Advances in Learning Theory:
Methods, Models and Applications

Edited by

J.A.K. Suykens
G. Horvath
S. Basu
C. Micchelli
J. Vandewalle

To be published by

IOS Press: NATO-ASI Series in Computer and Systems Sciences,
Amsterdam, The Netherlands, 2003

Preface

In recent years considerable progress has been made in the understanding of problems of learning and generalization. Intelligence in this context basically means the ability to perform well on new data after learning a model on the basis of given data. Such problems arise in many different areas and are becoming increasingly important and crucial towards many applications such as in bioinformatics, multimedia, computer vision and signal processing, internet search and information retrieval, datamining and textmining, finance, fraud detection, measurement systems and process control, and several others. Currently, the development of new technologies enables to generate massive amounts of data containing a wealth of information that remains to become explored. Often the dimensionality of the input spaces in these novel applications is huge such as in the analysis of microarray data where expression levels of thousands of genes need to be analysed given only a limited number of experiments. Without performing dimensionality reduction, the classical statistical paradigms show fundamental shortcomings at this point. Facing these new challenges, there is a need for new mathematical foundations and models such that the data can become processed in a reliable way. These subjects are very interdisciplinary and relate to problems studied in neural networks, machine learning, mathematics and statistics.

New methods, models and applications in Learning Theory were the central themes for a NATO Advanced Study Institute on *Learning Theory and Practice* which was taking place in Leuven Belgium July 8-19, 2002. This NATO Advanced Study Institute aims at creating a fascinating interplay between advanced theoretical results in learning theory and recent progress in several application areas. Lectures by 20 invited speakers were given who are leading authorities in the communities of neural networks, machine learning, mathematics, statistics, signal processing, and systems and control.

We highlight here a number of aspects from the invited speakers' lectures presented in the two-week period July 8-19, 2002. *Tomaso Poggio* explains the role of regularization in relation to statistical learning theory and illustrates the methods by several real-life applications including bioinformatics and vision. *Vladimir Vapnik* discusses the foundations of statistical learning theory and methods of support vector machines.

Especially the new elements of statistical learning theory versus classical statistics are emphasized. Benefits of these methods towards new applications in microarray data analysis and textmining are explained. Fields medal winner *Steve Smale* discusses the problem of learning and generalization and its mathematical foundations. The bias-variance problem is analysed and the difference between approximation and generalization theory is clarified. An application of the theory to modelling the evolution of language is presented. *Chris Bishop* presents probabilistic graphical models and their role in machine learning, including graphs, Markov Chain Monte Carlo methods, variational methods, the expectation-maximization algorithm and the relevance vector machine. *Bernhard Schölkopf* further discusses learning with kernels and shows the wide applicability of the so-called kernel trick in relation to positive definite kernels, including a kernel version of principal component analysis. An additional presentation in relation the Bernhard Schölkopf's talk is given by *Stéphane Canu* who shows how to avoid the requirement of positive definiteness for kernels in the context of kernel based learning. *Nello Cristianini* presents kernels which are tailored towards textmining applications and a kernel version of canonical correlation analysis. He further discusses methods of semidefinite programming for learning the kernel matrix and its use for transductive inference. *Massimiliano Pontil* presents ensembles of kernel machines with leave-one-out and stability analysis of the algorithms. *László Györfi* discusses nonparametric prediction of stationary time series for various loss functions. *Mathukumalli Vidyasagar* focuses on statistical learning theory with applications to bioinformatics such as fitting of hidden Markov models to data. *Yoram Singer* explains methods in machine learning for information retrieval and collaborative filtering, with emphasis on methods for multi-class categorization, boosting and ranking. *Rudolf Kulhavý* presents methods of Bayesian smoothing and information geometry with applications to local modelling in system identification. *Charles Micchelli* discusses properties of kernels and their use in approximation theory. Director *Johan Suykens* presents an overview of least squares support vector machines for classification, function estimation, unsupervised learning and recurrent networks including algorithms for large scale applications and links with several other kernel based methods. *Joerg Lemm* discusses Bayesian field theories with applications to density estimation, classification, regression and inverse quantum theory. *Vera Kurkova* gives an overview on approximation theoretical results for feedforward neural networks with new results towards large dimensional input spaces. *Luc Devroye* discusses methods of nonparametric learning from the viewpoint of statistics in relation to statistical learning theory. *Kristin Bennett* presents applications of support vector machines and kernel partial least squares to the virtual design of pharmaceuticals. *Peter Bartlett* gives an overview of policy gradient methods for reinforcement learning. Finally, co-director *Gabor Horvath* presents applications of neural networks and kernel based methods to instrumentation and measurement systems.

The total number of participants for this NATO-ASI was limited to 100 according to the NATO-ASI rules (a list of participants of this NATO-ASI can be found at the end of this book). Less than 50% of the submitted applications could be accepted taking into account the NATO restrictions. The following criteria were taken in the selection procedure as reflected by the NATO guidelines: good balancing between countries, balancing between topics, high quality of the work, selection of promising

post-docs and young researchers and good faculty people. The selected participants presented their work during one of the four poster sessions within the two-week period. More information about the final program, additional material of the invited speakers' lectures and a series of pictures showing the wonderful atmosphere during the meeting are available at

http://www.esat.kuleuven.ac.be/sista/natoasi/ltp2002.html .

As outcome of the NATO Advanced Study Institute, the present book entitled *Advances in Learning Theory; Methods, Models and Applications* has been prepared. In Chapter 1 *Vladimir Vapnik* presents an overview of statistical learning theory and its application to neural networks and support vector machines. In Chapter 2 *Felipe Cucker and Steve Smale* discuss the choice for regularization parameters in learning theory. This is done in view of the bias-variance trade-off in relation to analysis of the generalization error. In Chapter 3 *Charles Micchelli, Yuesheng Xu and Peixin Ye* present Cucker Smale Learning Theory in Besov Spaces. In Chapter 4 *Vera Kurkova* focuses on the analysis of the approximation error for neural networks and shows how to avoid the curse of dimensionality in high dimensional input spaces. In Chapter 5 *Stéphane Canu, Xavier Mary and Alain Rakotomamonjy* present functional learning through kernels and how to avoid the requirement that chosen kernels should be positive definite. In Chapter 6 *André Elisseeff and Massimiliano Pontil* present leave-one-out error and stability of learning algorithms with applications to kernel-based learning. In Chapter 7 *Ryan Rifkin, Gene Yeo, and Tomaso Poggio* discuss regularized least-squares Classification and comparisons with support vector machine classifiers and application to image classification. In Chapter 8 *Johan Suykens, Tony Van Gestel, Jos De Brabanter, Bart De Moor and Joos Vandewalle* give an overview on least squares support vector machine approaches to classification, regression, kernel PCA and kernel CCA with methods for imposing sparseness, robustness and handling large data sets. In Chapter 9 *Fernando Pérez-Cruz, Jason Weston, Daniel Herrmann and Bernhard Schölkopf* present an extension of the ν-SVM range for classification. In Chapter 10 *Nello Cristianini, Jaz Kandola, Alexei Vinokourov and John Shawe-Taylor* give an overview on kernels methods for text processing, including kernels which can be designed especially for textmining. In Chapter 11 *Kristin Bennett and Mark Embrechts* present kernel partial least squares methods and their application to pharmaceutical data. In Chapter 12 *Yoram Singer* discusses multiclass categorization in the context of information retrieval using support vector machines, boosting and decision trees. In Chapter 13 *Chris Bishop and Mike Tipping* study classification and regression from a Bayesian learning perspective and discuss Bayesian inference of parameterized kernel models using the relevance vector machine as an alternative to support vector machines. In Chapter 14 *Joerg Lemm* presents a framework of Bayesian field theory with application to density estimation, regression, and inverse quantum theory. In Chapter 15 *Rudolf Kulhavý* studies Bayesian smoothing and information geometry with emphasis on local, cased-based modeling. In Chapter 17 *László Györfi and Dominik Schäfer* consider non-parametric prediction of stationary time series for various loss functions. In Chapter 17 *Mathukumalli Vidyasagar* presents recent advances in the area of statistical learning theory including learning with inputs generated by a Markov chain

and prior information. In Chapter 18 *Gabor Horvath* confronts theoretical results from neural networks and learning theory with a practitioners point of view for applications in measurements systems.

Finally, we acknowledge support from NATO grant PST.ASI.978547, the IEEE Neural Networks Society, the fund for scientific research FWO Flanders (WOG-SRN Advanced Numerical Methods for Mathematical Modelling, FWO G.0407.02, JS post-doc), K.U. Leuven Research council GOA-Mefisto 666 and the Belgian interuniversity attraction poles IUAP IV-02, IUAP V-22. For help with local arrangements during the NATO-ASI we want to thank Lieveke Ameye, Peter Antal, Steven Bex, Tijl De Bie, Bart Hamers, Luc Hoegaerts, Chuan Lu, Lukas, Kristiaan Pelckmans, Qizheng Sheng Cynthia, Tony Van Gestel and Rita Vandewalle, and secretaries Ida Tassens, Ilse Pardon and Veerle Duchateau.

We are very grateful to all people who helped making this NATO Advanced Study Institute on Learning Theory and Practice a great success. We hope that this event may further lead to setting new milestones in this fascinating area!

Johan Suykens
Gabor Horvath
Sankar Basu
Charles Micchelli
Joos Vandewalle

Organizing committee

NATO-ASI Director:

Johan Suykens
K.U. Leuven
Dept. Elektrotechniek, ESAT-SCD-SISTA
Kasteelpark Arenberg 10
3001 Heverlee, Belgium
johan.suykens@esat.kuleuven.ac.be

NATO-ASI Partner Country Co-Director:

Gábor Horváth
Budapest Univ. of Technology & Economics
Dept. Measurement and Information Systems
Magyar tudósok Körútja 2
H-1117 Budapest, Hungary
horvath@mit.bme.hu

Other NATO-ASI organizing committee members:

Sankar Basu
National Science Foundation
CISE/CCR Division
4201 Wilson Blvd., Room 1145
Arlington, VA 22230, USA
sabasu@nsf.gov

Charles Micchelli
State University of New York
Dept. Mathematics & Statistics
University at Albany, SUNY
Albany, NY 12222, USA
cam@math.albany.edu

Joos Vandewalle
K.U. Leuven
Dept. Elektrotechniek, ESAT-SCD-SISTA
Kasteelpark Arenberg 10
3001 Heverlee, Belgium
joos.vandewalle@esat.kuleuven.ac.be

Welcome reception in the Arenberg castle at the NATO Advanced Study Institute on Learning Theory and Practice taking place in Leuven July 2002 (picture: Lukas)

List of chapter contributors

Kristin Bennett
Rensselaer Polytechnic Institute
Dept. Mathematical Sciences
Troy, New York, 12180-3590
USA
bennek@rpi.edu

Christopher Bishop
Microsoft Research Ltd.
7 J.J. Thomson Avenue
Cambridge CB3 OFB
U.K.
cmbishop@microsoft.com

Stéphane Canu
INSA Rouen
BP 8 Place E. Blondel
76131 Mont. St. Agnon, Cedex
France
scanu@insa-rouen.fr

Nello Cristianini
Royal Holloway, University of London
Dept. Computer Science
Egham, Surrey
TW 20 OEX, UK
nello@support-vector.net

Felipe Cucker
City University of Hong Kong
Department of Mathematics
83 Tat Chee Avenue, Kowloon
Hong Kong
macucker@math.cityu.edu.hk

Jos De Brabanter
K.U. Leuven
Dept. Elektrotechniek, ESAT-SCD-SISTA
Kasteelpark Arenberg 10
3001 Heverlee, Belgium
jos.debrabanter@esat.kuleuven.ac.be

Bart De Moor
K.U. Leuven
Dept. Elektrotechniek, ESAT-SCD-SISTA
Kasteelpark Arenberg 10
3001 Heverlee, Belgium
bart.demoor@esat.kuleuven.ac.be

Andre Elisseeff
Max Planck Institute for
Biological Cybernetics
Spemannstr. 38, 72076 Tübingen
Germany
andre.elisseeff@tuebingen.mpg.de

Mark Embrechts
Rensselaer Polytechnic Institute
Dept. Decision Sc. and Eng. Syst.
Troy, NY 12180
USA
embrem@rpi.edu

Laszlo Gyorfi
Budapest Univ. Techn. and Economics
Dept. Comp. sc. & inform. theory
Stoczek u.2.
Hungary
gyorfi@szit.bme.hu

Daniel Herrmann
Max Planck Institute for
Biological Cybernetics
Spemannstrasse 3, 72076 Tübingen
Germany
daniel.herrmann@tuebingen.mpg.de

Gábor Horváth
Budapest Univ. of Technology & Economics
Dept. Measurement and Information Systems
Magyar tudósok körútja 2
H-1117 Budapest, Hungary
horvath@mit.bme.hu

Jaz Kandola
Royal Holloway, University of London
Dept. Computer Science
Egham, Surrey
TW 20 OEX, UK
j.kandola@cs.rhul.ac.uk

Rudolf Kulhavy
Honeywell Prague Laboratory
Pod vodarenskou vezi 4
182 08 Prague 8
Czech Republic
rudolf.kulhavy@honeywell.com

Vera Kurkova
Academy of Sciences of the Czech Republic
Institute of Computer Science
Pod Vodarenskou vezi 2
182 07 Prague, Czech Republic
vera@cs.cas.cz

Jörg Lemm
Universität Münster
Institut für Theoretische Physik
Wilhelm-Klemm-Str. 9
48149 Münster, Germany
joerg.lemm@wgz-bank.de

Xavier Mary
INSA Rouen
BP 8 Place E. Blondel
76131 Mont. St. Agnon, Cedex
France
xavier.mary@insa-rouen.fr

Charles Micchelli
State University of New York
Dept. Mathematics & Statistics
University at Albany, SUNY
Albany, NY 12222, USA
cam@math.albany.edu

Fernando Perez-Cruz
Universidad Carlos III de Madrid
Dpto. Teori a de la Señal
y Comunicaciones. Leganes
Spain
fernandop@ieee.org

Tomaso Poggio
Massachusetts Inst. of Technology
Dept. Brain & Cognitive Sciences, AI Lab
45 Carleton Street, Cambridge MA 02142
USA
tp@ai.mit.edu

Massimiliano Pontil
Dipartimento di Ingegneria Informatica
Università di Siena
Via Roma 56, 53100 Siena
Italy
pontil@dii.unisi.it

Alain Rakotomamonjy
INSA Rouen
BP 8 Place E. Blondel
76131 Mont. St. Agnon, Cedex
France
alain.rakotomamonjy@insa-rouen.fr

Ryan Rifkin
Massachusetts Inst. of Technology
Dept. Brain & Cognitive Sciences, AI Lab
45 Carleton Street, Cambridge MA 02142
USA
rif@mit.edu

Dominik Schafer
Universität Stuttgart
Mathematisches Institut A
Pfaffenwaldring 57, D-70511 Stuttgart
Germany
schaefdk@mathematik.uni-stuttgart.de

Bernhard Schölkopf
Max Planck Institute for
Biological Cybernetics
Spemannstrasse 3, 72076 Tübingen
Germany
bernhard.schoelkopf@tuebingen.mpg.de

John Shawe-Taylor
Royal Holloway, University of London
Dept. Computer Science
Egham, Surrey
TW 20 OEX, UK
jst@cs.rhul.ac.uk

Yoram Singer
The Hebrew Univ., Givat-Ram campus
School of Computer Sc. & Eng.
Jerusalem 91904
Israel
singer@cs.huji.ac.il

Steve Smale
University of California
Dept. Mathematics
Berkeley, CA 94720-384
USA
smale@math.berkeley.edu

Johan Suykens
K.U. Leuven
Dept. Elektrotechniek, ESAT-SCD-SISTA
Kasteelpark Arenberg 10
3001 Heverlee, Belgium
johan.suykens@esat.kuleuven.ac.be

Michael Tipping
Microsoft Research Ltd.
7 J.J. Thomson Avenue
Cambridge CB3 OFB
U.K.
mtipping@microsoft.com

Joos Vandewalle
K.U. Leuven
Dept. Elektrotechniek, ESAT-SCD-SISTA
Kasteelpark Arenberg 10
3001 Heverlee, Belgium
joos.vandewalle@esat.kuleuven.ac.be

Tony Van Gestel
K.U. Leuven
Dept. Elektrotechniek, ESAT-SCD-SISTA
Kasteelpark Arenberg 10
3001 Heverlee, Belgium
tony.vangestel@esat.kuleuven.ac.be

Vladimir Vapnik
NEC Research
4 Independence Way
Princeton, NJ 08540
USA
vlad@research.nj.nec.com

M. Vidyasagar
Tata Consultancy Services
6th floor, Khan Lateefkahn Estate
Fateh Maidan Road, Hyderabad 500 001
India
sagar@atc.tcs.co.in

Alexei Vinokourov
Royal Holloway, University of London
Dept. Computer Science
Egham, Surrey
TW 20 OEX, UK
alexei@cs.rhul.ac.uk

Jason Weston
Max Planck Institute for
Biological Cybernetics
Spemannstrasse 3, 72076 Tübingen
Germany
jason.weston@tuebingen.mpg.de

Yuesheng Xu
Department of Mathematics
West Virginia University
Morgantown, West Virginia 26506
USA
yxu@math.wvu.edu

Peixin Ye
Institute of Mathematics
Academy of Math. & Syst. Sci.
Chinese Academy of Sciences
Beijing 100080, P.R. China
yepx@amss.ac.cn

Gene Yeo
Massachusetts Inst. of Technology
Dept. Brain & Cognitive Sciences, AI Lab
45 Carleton Street, Cambridge MA 02142
USA
geneyeo@ai.mit.edu

Contents

Chapter 1

An Overview of Statistical Learning Theory

Vladimir Vapnik[1]

Abstract. Statistical learning theory was introduced in the late 1960's. Until the 1990's it was a purely theoretical analysis of the problem of function estimation from a given collection of data. In the middle of the 1990's new types of learning algorithms (called support vector machines) based on the developed theory were proposed. This made statistical learning theory not only a tool for the theoretical analysis but also a tool for creating practical algorithms for estimating multidimensional functions. This article presents a very general overview of statistical learning theory including both theoretical and algorithmic aspects of the theory. The goal of this overview is to demonstrate how the abstract learning theory established conditions for generalization which are more general than those discussed in classical statistical paradigms and how the understanding of these conditions inspired new algorithmic approaches to function estimation problems. A more detailed overview of the theory (without proofs) can be found in Vapnik (1995). In Vapnik (1998) one can find a detailed description of the theory (including proofs).

[1]This chapter is reprinted with permission from Vladimir Vapnik, "An overview of statistical learning theory," IEEE Transactions on Neural Networks, Vol.10, No.5, pp.988-1000, 1999 (Copyright © 1999 IEEE). The author wants to thank Filip Mulier for discussions and help making the published article more clear and readable.

1.1 Setting of the Learning Problem

In this section we consider a model of learning and show that analysis of this model can be conducted in the general statistical framework of minimizing expected loss using observed data. We show that practical problems such as pattern recognition, regression estimation, and density estimation are particular case of this general model.

1.1.1 Function estimation model

The model of learning from examples can be described using three components:

1. A generator of random vectors x, drawn independently from a fixed but unknown distribution $P(x)$.

2. A supervisor that returns an output vector y for every input vector x, according to a conditional distribution function[2] $P(y|x)$, also fixed but unknown.

3. A learning machine capable of implementing a set of functions $f(x, \alpha)$, $\alpha \in \Lambda$.

The problem of learning is that of choosing from the given set of functions $f(x, \alpha)$, $\alpha \in \Lambda$, the one which predicts the supervisor's response in the best possible way. The selection is based on a training set of ℓ random independently identically distributed (i.i.d.) observations

$$(x_1, y_1), ..., (x_\ell, y_\ell) \tag{1.1}$$

drawn according to $P(x, y) = P(x)P(y|x)$.

1.1.2 Problem of risk minimization

In order to choose the best available approximation to the supervisor's response, one measures the *loss* or discrepancy $L(y, f(x, \alpha))$ between the response y of the supervisor to a given input x and the response $f(x, \alpha)$ provided by the learning machine. Consider the expected value of the loss, given by the *risk functional*

$$R(\alpha) = \int L(y, f(x, \alpha))dP(x, y). \tag{1.2}$$

The goal is to find the function $f(x, \alpha_0)$ which minimizes the risk functional $R(\alpha)$ over the class of functions $f(x, \alpha)$, $\alpha \in \Lambda$ in the situation where the joint probability distribution $P(x, y)$ is unknown and the only available information is contained in the training set (1.1).

1.1.3 Three main learning problems

This formulation of the learning problem is rather general. It encompasses many specific problems. Below we consider the main ones: the problems of pattern recognition, regression estimation, and density estimation.

[2]This is the general case which includes a case where the supervisor uses a function $y = f(x)$.

The Problem of Pattern Recognition. Let the supervisor's output y take only values $y \in \{0,1\}$ and let $f(x,\alpha),\ \alpha \in \Lambda$, be a set of *indicator* functions (functions which take on either value zero or one). Consider the following loss-function

$$L(y, f(x,\alpha)) = \begin{cases} 0 & \text{if } y = f(x,\alpha) \\ 1 & \text{if } y \neq f(x,\alpha). \end{cases} \tag{1.3}$$

For this loss function, the functional (1.2) provides the probability of classification error (i.e. when the answers y given by the supervisor and the answers given by the indicator function $f(x,\alpha)$ differ). Therefore, the problem is to find the function which minimizes the probability of classification errors when the probability measure $P(x,y)$ is unknown, but the data (1.1) are given.

The Problem of Regression Estimation. Let the supervisor's answer y be a real value, and let $f(x,\alpha)$, $\alpha \in \Lambda$ be a set of real functions which contains the *regression function*

$$f(x,\alpha_0) = \int y dP(y|x).$$

It is known that if $f(x,\alpha) \in L_2$ then the regression function is the one which minimizes the functional (1.2) with the the following loss-function

$$L(y, f(x,\alpha)) = (y - f(x,\alpha))^2. \tag{1.4}$$

Thus the problem of regression estimation is the problem of minimizing the risk functional (1.2) with the loss function (1.4) in the situation where the probability measure $P(x,y)$ is unknown but the data (1.1) are given.

The Problem of Density Estimation. Finally, consider the problem of density estimation from the set of densities $p(x,\alpha),\ \alpha \in \Lambda$. For this problem we consider the following loss-function

$$L(p(x,\alpha)) = -\log p(x,\alpha). \tag{1.5}$$

It is known that the desired density minimizes the risk functional (1.2) with the loss-function (1.5). Thus, again, in order to estimate the density from the data one has to minimize the risk-functional under the condition that the corresponding probability measure $P(x)$ is unknown but i.i.d. data

$$x_1, ..., x_n$$

are given.

The General Setting of the Learning Problem. The general setting of the learning problem can be described as follows. Let the probability measure $P(z)$ be defined on the space Z. Consider the set of functions $Q(z,\alpha), \alpha \in \Lambda$. The goal is: minimize the risk functional

$$R(\alpha) = \int Q(z,\alpha) dP(z), \quad \alpha \in \Lambda \tag{1.6}$$

for the probability measure $P(z)$ unknown but given an i.i.d. sample

$$z_1, ..., z_\ell. \tag{1.7}$$

The learning problems considered above are particular cases of this general problem of *minimizing the risk functional (1.6) on the basis of empirical data (1.7)*, where z denotes a pair (x, y) and $Q(z, \alpha)$ is the specific loss function (for example, either (1.3), (1.4) or (1.5)). In the sequel we will describe results obtained for the general statement of the problem. When applying this to specific problems, one has to substitute the corresponding loss-functions in the obtained formulas.

1.1.4 Empirical risk minimization induction principle

In order to minimize the risk functional (1.6) for an unknown probability measure $P(z)$ the following induction principle is usually employed.

The expected risk functional $R(\alpha)$ is replaced by the *empirical risk* functional

$$R_{emp}(\alpha) = \frac{1}{\ell}\sum_{i=1}^{\ell} Q(z_i, \alpha) \tag{1.8}$$

constructed on the basis of the training set (1.7). The principle is to approximate the function $Q(z, \alpha_0)$ which minimizes the risk (1.6) by the function $Q(z, \alpha_\ell)$ which minimizes the empirical risk (1.8). This principle is called the Empirical Risk Minimization induction principle (ERM principle).

1.1.5 Empirical risk minimization principle and the classical methods

The ERM principle is quite general. The classical methods for solving a specific learning problem, such as the least squares method in the problem of regression estimation or the maximum likelihood method in the problem of density estimation are realizations of the ERM principle for the specific loss functions considered above.

Indeed, in order to specify the regression problem one introduces an $n + 1$ dimensional variable $z = (x, y) = (x^1, ..., x^n, y)$ and uses loss function (1.4). Using this loss function in the functional (1.8) yields the functional

$$R_{emp}(\alpha) = \frac{1}{\ell}\sum_{i=1}^{\ell}(y_i - f(x_i, \alpha))^2$$

which one needs to minimize for finding the regression estimate (least square method).

In order to estimate a density function from a given set of functions $p(x, \alpha)$ one uses the loss function (1.5). Putting this loss function into (1.8) one obtains the maximum likelihood method with functional

$$R_{emp}(\alpha) = -\frac{1}{\ell}\sum_{i=1}^{\ell} \ln p(x_i, \alpha).$$

which one needs to minimize in order to find the approximation to the density.

Since the ERM principle is a general formulation of these classical estimation problems, any theory concerning the ERM principle applies to the classical methods as well.

1.1.6 Four parts of learning theory

Learning theory has to address the following four questions:

1. *What are the conditions for consistency of the ERM principle?*

   To answer this question one has to specify the *necessary and sufficient* conditions for convergence in probability[3]of the following sequences of the random values:

   - The values of risks $R(\alpha_\ell)$ converging to the minimal possible value of the risk $R(\alpha_0)$ (where $R(\alpha_\ell)$, $\ell = 1, 2, ...$ are the expected risks for functions $Q(z, \alpha_\ell)$ each minimizing the empirical risk $R_{emp}(\alpha_\ell)$)

   $$R(\alpha_\ell) \longrightarrow^{P}_{\ell\to\infty} R(\alpha_0), \tag{1.9}$$

   - The values of obtained empirical risks $R_{emp}(\alpha_\ell)$, $i = 1, 2, ...$ converging to the minimal possible value of the risk $R(\alpha_0)$

   $$R_{emp}(\alpha_\ell) \longrightarrow^{P}_{\ell\to\infty} R(\alpha_0). \tag{1.10}$$

   Equation (1.9) shows that solutions found using ERM converge to the best possible one. Equation (1.10) shows that empirical risk values converge to the value of the smallest risk.

2. *How fast does the sequence of smallest empirical risk values converge to the smallest actual risk?* In other words, what is the rate of generalization of a learning machine that implements the empirical risk minimization principle?

3. *How can one control the rate of convergence (the rate of generalization) of the learning machine?*

4. *How can one construct algorithms that can control the rate of generalization?*

The answers to these questions form the four parts of learning theory:

1. The theory of consistency of learning processes.

2. The non-asymptotic theory of the rate of convergence of learning processes.

3. The theory of controlling the generalization of learning processes.

4. The theory of constructing learning algorithms.

[3]Convergence in probability of values $R(\alpha_\ell)$ means that for any $\varepsilon > 0$ and for any $\eta > 0$ there exists a number $\ell_0 = \ell_0(\varepsilon, \eta)$ such that for any $\ell > \ell_0$ with probability at least $1 - \eta$ the inequality $R(\alpha_\ell) - R(\alpha_0) < \varepsilon$ holds true.

1.2 The Theory of Consistency of Learning Processes

The theory of consistency is an asymptotic theory. It describes *the necessary and sufficient conditions* for convergence of the solutions, obtained using the proposed method, to the best possible as the number of observations is increased. The following question arises:

> *Why do we need a theory of consistency if our goal is to construct algorithms for a small (finite) sample size?*

The answer is:

> *We need a theory of consistency because it provides not only sufficient but also necessary conditions for convergence of the empirical risk minimization inductive principle. Therefore, any theory of the empirical risk minimization principle must satisfy these necessary and sufficient conditions.*

In this section we introduce the main capacity concept (the so-called VC entropy) which defines the generalization ability of the ERM principle. In the next sections we show that the non-asymptotic theory of learning is based on different types of bounds that evaluate this concept for a fixed amount of observations.

1.2.1 The key theorem of the learning theory

The key theorem of the theory concerning the ERM based learning processes is the following [27]:

Theorem 1 (The Key Theorem) *Let $Q(z,\alpha)$, $\alpha \in \Lambda$ be a set of functions that has a bounded loss for probability measure $P(z)$*

$$A \leq \int Q(z,\alpha)P(z) \leq B \quad \forall \alpha \in \Lambda.$$

Then for the ERM principle to be consistent it is necessary and sufficient that the empirical risk $R_{emp}(\alpha)$ converges uniformly to the actual risk $R(\alpha)$ over the set $Q(z,\alpha)$, $\alpha \in \Lambda$ as

$$\lim_{\ell \to \infty} Prob\{\sup_{\alpha \in \Lambda} (R(\alpha) - R_{emp}(\alpha)) > \varepsilon\} = 0, \qquad \forall \varepsilon. \tag{1.11}$$

This type of convergence is called uniform one-sided convergence. In other words, according to the Key Theorem, the conditions for consistency of the ERM principle are equivalent to the conditions for existence of uniform one-sided convergence (1.11). This theorem is called the Key Theorem because it asserts that any analysis of the convergence properties of the ERM principle must be the *worst case analysis*. The necessary condition for consistency (not only the sufficient condition) depends on whether or not the deviation for the worst function over the given set of of functions

$$\Delta(\alpha_{worst}) = \sup_{\alpha \in \Lambda} (R(\alpha) - R_{emp}(\alpha))$$

converges to zero in probability. From this theorem it follows that the analysis of the ERM principle requires an analysis of the properties of uniform convergence of the expectations to their probabilities over the given set of functions.

1.2.2 The necessary and sufficient conditions for uniform convergence

To describe the necessary and sufficient condition for uniform convergence (1.11), we introduce a concept called *the entropy of the set of functions* $Q(z,\alpha)$, $\alpha \in \Lambda$, *on the sample of size* ℓ. We introduce this concept in two steps: first for sets of indicator functions and then for sets of real valued functions.

Entropy of the Set of Indicator Functions. Let $Q(z,\alpha)$, $\alpha \in \Lambda$ be a set of indicator functions (i.e. functions which take only the values zero or one). Consider a sample

$$z_1,\ldots,z_\ell. \tag{1.12}$$

Let us characterize the diversity of this set of functions $Q(z,\alpha),\alpha \in \Lambda$ on the given sample by a quantity $N^\Lambda(z_1,\ldots,z_\ell)$ that represents the number of different separations of this sample that can be obtained using functions from the given set of indicator functions. Let us write this in another form. Consider the set of ℓ-dimensional binary vectors

$$q(\alpha) = (Q(z_1,\alpha),\ldots,Q(z_\ell,\alpha)), \quad \alpha \in \Lambda$$

that one obtains when α takes various values from Λ. Then geometrically speaking $N^\Lambda(z_1,\ldots,z_\ell)$ is the number of different vertices of the ℓ-dimensional cube that can be obtained on the basis of the sample $z_1,\ldots,z_\ell$ and the set of functions $Q(z,\alpha),\alpha \in \Lambda$.Let us call the value

$$H^\Lambda(z_1,\ldots,z_\ell) = \ln N^\Lambda(z_1,\ldots,z_\ell)$$

the *random entropy*. The random entropy describes the diversity of the set of functions on the given data. $H^\Lambda(z_1,\ldots,z_\ell)$ is a random variable since it was constructed using random i.i.d. data. Now we consider the expectation of the random entropy over the joint distribution function $F(z_1,\ldots,z_\ell)$:

$$H^\Lambda(\ell) = E\ln N^\Lambda(z_1,\ldots,z_\ell).$$

We call this quantity the entropy of the set of indicator functions $Q(z,\alpha)$, $\alpha \in \Lambda$ on samples of size ℓ. It depends on the set of functions $Q(z,\alpha)$, $\alpha \in \Lambda$, the probability measure $P(z)$, and the number of observations ℓ. The entropy describes the expected diversity of the given set of indicator functions on the sample of size ℓ.

The main result of the theory of consistency for the pattern recognition problem (the consistency for indicator loss function) is the following theorem [24]:

Theorem 2 *For uniform two-sided convergence of the frequencies to their probabili-*

ties[4] *it is necessary and sufficient that the equality*

$$\lim_{\ell\to\infty}\frac{H^{\Lambda}(\ell)}{\ell}=0,\quad \forall\varepsilon>0 \tag{1.14}$$

holds.

By slightly modifying the condition (1.14) one can obtain the necessary and sufficient condition for one-sided uniform convergence (1.11).

Entropy of the Set of Real Functions. Now we generalize the concept of entropy to sets of real valued functions. Let $A\le Q(z,\alpha)\le B,\ \alpha\in\Lambda$, be a set of bounded loss functions. Using this set of functions and the training set (1.12) one can construct the following set of ℓ-dimensional real-valued vectors

$$q(\alpha)=(Q(z_1,\alpha),...,Q(z_\ell,\alpha)),\quad \alpha\in\Lambda. \tag{1.15}$$

This set of vectors belongs to the ℓ-dimensional cube with the edge $B-A$ and has a finite ε-net[5] in the metric C. Let $N=N^{\Lambda}(\varepsilon;z_1,...,z_\ell)$ be the number of elements of the minimal ε-net of the set of vectors $q(\alpha),\ \alpha\in\Lambda$. The logarithm of the (random) value $N^{\Lambda}(\varepsilon;z_1,...,z_\ell)$

$$H^{\Lambda}(\varepsilon;z_1,...,z_\ell)=\ln N^{\Lambda}(\varepsilon;z_1,...,z_\ell)$$

is called the *random VC-entropy*[6] of the set of functions $A\le Q(z,\alpha)\le B$ on the sample $z_1,...,z_\ell$. The expectation of the random VC-entropy

$$H^{\Lambda}(\varepsilon;\ell)=EH^{\Lambda}(\varepsilon;z_1,...,z_\ell)$$

is called the *VC-entropy* of the set of functions $A\le Q(z,\alpha)\le B,\ \alpha\in\Lambda$ on the sample of the size ℓ. Here expectation is taken with respect to product-measure $P(z_1,...,z_\ell)=P(z_1)\cdot...\cdot P(z_\ell)$.

The main results of the theory of uniform convergence of the empirical risk to the actual risk for bounded loss functions include the following theorem [24]:

Theorem 3 *For uniform two-sided convergence of the empirical risks to the actual risks*

$$\lim_{\ell\to\infty}Prob\{\sup_{\alpha\in\Lambda}(|R(\alpha)-R_{emp}(\alpha)|>\varepsilon\}=0,\qquad \forall\varepsilon. \tag{1.16}$$

---

[4]For sets of indicator functions $R(\alpha)$ defines probability and $R_{emp}(\alpha)$ defines frequency.

$$\lim_{\ell\to\infty}\mathrm{Prob}\{\sup_{\alpha\in\Lambda}|R(\alpha)-R_{emp}(\alpha)|>\varepsilon\}=0,\qquad \forall\varepsilon. \tag{1.13}$$

[5]The set of vectors $q(\alpha),\ \alpha\in\Lambda$ has minimal ε-net $q(\alpha_1),...,q(\alpha_N)$ if:

1. There exist $N=N^{\Lambda}(\varepsilon;z_1,...,z_\ell)$ vectors $q(\alpha_1),...,q(\alpha_N)$, such that for any vector $q(\alpha^*),\ \alpha^*\in\Lambda$ one can find among these N vectors one $q(\alpha_r)$ which is ε-close to this vector (in a given metric). For a C metric that means $\rho(q(\alpha^*),q(\alpha_r))=\max_{1\le i\le\ell}|Q(z_i\alpha^*)-Q(z_i,\alpha_r)|\le\varepsilon$.
2. N is minimal number of vectors which possess this property.

[6]Note that VC-entropy is different from classical metrical ε-entropy $H^{\Lambda}_{cl}(\varepsilon)=\ln N^{\Lambda}(\varepsilon)$ where $N^{\Lambda}(\varepsilon)$ is cardinality of the minimal ε-net of the set of functions $Q(z,\alpha),\alpha\in\Lambda$.

it is necessary and sufficient that the equality

$$\lim_{\ell\to\infty}\frac{H^{\Lambda}(\varepsilon,\ell)}{\ell}=0,\quad\forall\varepsilon>0 \tag{1.17}$$

be valid.

By slightly modifying the condition (1.16) one can obtain the necessary and sufficient condition for one-sided uniform convergence (1.11). According to the key assertion this implies the necessary and sufficient conditions for consistency of the ERM principle.

1.2.3 Three milestones in learning theory

In this section, we consider for simplicity reasons a set of indicator functions $Q(z,\alpha),\alpha\in\Lambda$ (i.e. we consider the problem of pattern recognition). The results obtained for sets of indicator functions can be generalized to sets of real-valued functions. In the previous section we introduced the entropy for sets of indicator functions

$$H^{\Lambda}(\ell)=E\ln N^{\Lambda}(z_1,\ldots,z_\ell).$$

Now, we consider two new functions that are constructed on the basis of the values $N^{\Lambda}(z_1,...,z_\ell)$: the *Annealed VC-entropy*

$$H^{\Lambda}_{ann}(\ell)=\ln EN^{\Lambda}(z_1,...,z_\ell)$$

and the *Growth function*

$$G^{\Lambda}(\ell)=\ln\sup_{z_1,\ldots,z_\ell}N^{\Lambda}(z_1,...,z_\ell).$$

These functions are determined in such a way that for any ℓ the inequalities

$$H^{\Lambda}(\ell)\le H^{\Lambda}_{ann}(\ell)\le G^{\Lambda}(\ell)$$

are valid. On the basis of these functions the three main milestones in Statistical Learning Theory are constructed. In the previous section we introduced the equation

$$\lim_{\ell\to\infty}\frac{H^{\Lambda}(\ell)}{\ell}=0$$

describing the *necessary and sufficient condition* for consistency of the ERM principle. This equation is the first milestone in learning theory: any machine that is minimizing empirical risk should satisfy it.

On the other hand, this equation says nothing about the rate of convergence of obtained risks $R(\alpha_\ell)$ to the minimal one $R(\alpha_0)$. It is possible that the ERM principle is consistent but has an arbitrary slow asymptotic rate of convergence. The question is then: *Under which conditions does one have a fast asymptotic rate of convergence?*

We say that the asymptotic rate of convergence is fast if for any $\ell > \ell_0$ the exponential bound

$$P\{R(\alpha_\ell) - R(\alpha_0) > \varepsilon\} < e^{-c\varepsilon^2\ell}$$

holds true, where $c > 0$ is some constant. The equation

$$\lim_{\ell\to\infty} \frac{H^{\Lambda}_{ann}(\ell)}{\ell} = 0$$

describes the *sufficient* condition for fast convergence[7]. This constitutes the second milestone in Statistical Learning Theory: guaranteeing a fast asymptotic rate of convergence. Note that both the equation describing the necessary and sufficient condition for consistency and the one that describes the sufficient condition for fast convergence of the ERM method are valid for a *given* probability measure $P(z)$ (both VC-entropy $H^{\Lambda}(\ell)$ and VC-annealed entropy $H^{\Lambda}_{ann}(\ell)$ are constructed using this measure).

However, our goal is to construct a learning machine for solving many different problems (i.e. for many different probability measures). The next question is then: *Under what conditions is the ERM principle consistent and rapidly converging independently of the probability measure?* The following equation describes the necessary and sufficient conditions for consistency of ERM for any probability measure:

$$\lim_{\ell\to\infty} \frac{G^{\Lambda}(\ell)}{\ell} = 0.$$

This condition is also sufficient for fast convergence. This equation is the third milestone in Statistical Learning Theory. It describes the conditions under which the learning machine implementing the ERM principle has an asymptotic high rate of convergence, independent of the problem to be solved.

These milestones form a foundation for constructing both distribution independent bounds and rigorous distribution dependent bounds for the rate of convergence of learning machines.

1.3 Bounds on the Rate of Convergence of the Learning Processes

In order to estimate the quality of the ERM method for a given sample size it is necessary to obtain non-asymptotic bounds on the rate of uniform convergence.

A non-asymptotic bound of the rate of convergence can be obtained using a new capacity concept, called the VC dimension (abbreviation for Vapnik-Chervonenkis dimension), which allows us to obtain a constructive bound for the growth function. The concept of VC-dimension is based on a remarkable property of the Growth-function $G^{\Lambda}(\ell)$.

[7]The necessity of this condition for fast convergence is an open question.

1.3.1 The structure of the growth function

Theorem 4 [23-24] *Any growth function either satisfies the equality*

$$G^{\Lambda}(\ell) = \ell \ln 2$$

or is bounded by the inequality

$$G^{\Lambda}(\ell) < h(\ln \frac{\ell}{h} + 1),$$

where h is an integer for which

$$G^{\Lambda}(h) = h \ln 2$$

$$G^{\Lambda}(h+1) \neq (h+1)\ln 2.$$

In other words the Growth function will be either a linear function or will be bounded by a logarithmic function. (e.g. it cannot be of the form $G^{\Lambda}(\ell) = c\sqrt{\ell}$). We say that the VC dimension of the set of indicator functions $Q(z, \alpha), \alpha \in \Lambda$ is infinite if the Growth function for this set of functions is linear. We say that the VC dimension of the set of indicator functions $Q(z, \alpha), \alpha \in \Lambda$ is finite and equals h if the Growth function is bounded by a logarithmic function with coefficient h.

The finiteness of the VC-dimension of the set of indicator functions implemented by the learning machine forms the necessary and sufficient condition for consistency of the ERM method independent of the probability measure. Finiteness of the VC-dimension also implies fast convergence.

1.3.2 Equivalent definition of the VC dimension

In this section we give an equivalent definition of the VC dimension of sets of indicator functions and we generalize this definition to sets of real functions.

The VC dimension of a set of indicator functions. The *VC-dimension of a set of indicator functions* $Q(z, \alpha)$, $\alpha \in \Lambda$, is the maximum number h of vectors $z_1, ..., z_h$ which can be separated in all 2^h possible ways using functions of this set[8] (*shattered* by this set of functions). If for any n there exists a set of n vectors which can be shattered by the set $Q(z, \alpha)$, $\alpha \in \Lambda$, then the VC-dimension is equal to infinity.

The VC dimension of a set of real valued functions. Let $a \leq Q(z, \alpha) \leq A$, $\alpha \in \Lambda$, be a set of real valued functions bounded by constants a and A (a can approach $-\infty$ and A can be equal to ∞). Let us consider along with the set of real valued functions $Q(z, \alpha)$, $\alpha \in \Lambda$, the set of indicator functions

$$I(z, \alpha, \beta) = \theta \{Q(z, \alpha) - \beta\}, \ \alpha \in \Lambda \tag{1.18}$$

[8] Any indicator function separates a set of vectors into two subsets: the subset of vectors for which this function takes value 0 and the subset of vectors for which it takes value 1.

where $a < \beta < A$ is some constant, $\theta(u)$ is a step function:

$$\theta(u) = \begin{cases} 0 & \text{if } u < 0 \\ 1 & \text{if } u \geq 0. \end{cases}$$

The *VC dimension of the set of real valued functions* $Q(z, \alpha)$, $\alpha \in \Lambda$, is defined to be the VC-dimension of the set of indicator functions (1.18).

1.3.3 Two important examples

Example 1

1. The VC-dimension of the set of *linear indicator functions*

$$Q(z, \alpha) = \theta\left\{\sum_{p=1}^{n} \alpha_p z_p + \alpha_0\right\}$$

in n-dimensional coordinate space $Z = (z_1, ..., z_n)$ is equal to $h = n + 1$, since using functions of this set one can shatter at most $n + 1$ vectors. Here $\theta\{\cdot\}$ is the step function, which takes value 1 if the expression between brackets is positive and takes value 0 otherwise.

2. The VC-dimension of the set of *linear functions*

$$Q(z, \alpha) = \sum_{p=1}^{n} \alpha_p z_p + \alpha_0, \quad \alpha_0, ..., \alpha_n \in (-\infty, \infty)$$

in n-dimensional coordinate space $Z = (z_1, ..., z_n)$ is also equal to $h = n + 1$ because the VC-dimension of the corresponding linear indicator functions is equal to $n+1$ (using $\alpha_0 - \beta$ instead of α_0 does not change the set of indicator functions).

Example 2

We call a hyperplane

$$(w^* \cdot x) - b = 0, \quad |w^*| = 1$$

the Δ-margin separating hyperplane if it classifies vectors x as follows

$$y = \begin{cases} 1 & \text{if } (w^* \cdot x) - b \geq \Delta \\ -1 & \text{if } (w^* \cdot x) - b \leq -\Delta. \end{cases}$$

Classifications of vectors x that fall within the margin $(-\Delta, \Delta)$ are undefined.

Theorem 5 [25, 20–22] *Let vectors $x \in X$ belong to a sphere of radius R, then the set of Δ-margin separating hyperplanes has VC dimension h bounded by the inequality*

$$h \leq \min\left(\left[\frac{R^2}{\Delta^2}\right], n\right) + 1.$$

These examples show that in general the VC dimension of the set of hyperplanes equals $n + 1$, where n is the dimensionality of the input space. However, the VC dimension of the set of Δ-margin separating hyperplanes (with a large value of margin Δ) can be less than $n + 1$. This fact will play an important role towards constructing new function estimation methods.

1.3.4 Distribution independent bounds for the rate of convergence of learning processes

Consider sets of functions which possess a finite VC-dimension h. We distinguish then between the following two cases:

1. The case where the set of loss functions $Q(z,\alpha),\ \alpha \in \Lambda$ is a set of *totally bounded functions*

2. The case where the set of loss functions $Q(z,\alpha),\ \alpha \in \Lambda$ is *not necessarily a set of totally bounded functions.*

Case 1 [The set of totally bounded functions] Without loss of generality, we assume that

$$0 \le Q(z,\alpha) \le B,\ \alpha \in \Lambda. \tag{1.19}$$

The main result in the theory of bounds for sets of totally bounded functions is the following [20–22]:

Theorem 6 *With probability at least* $1-\eta$*, the inequality*

$$R(\alpha) \le R_{emp}(\alpha) + \frac{B\varepsilon}{2}\left(1+\sqrt{1+\frac{4R_{emp}(\alpha)}{B\varepsilon}}\right), \tag{1.20}$$

holds true simultaneously for all functions of the set (1.19), where

$$\varepsilon = 4\frac{h(\ln\frac{2\ell}{h}+1)-\ln\eta}{\ell}. \tag{1.21}$$

For the set of indicator functions: $B=1$

This Theorem provides bounds for the risks of all functions of the set (1.18) (including the function $Q(z,\alpha_\ell)$ which minimizes the empirical risk (1.8)). The bounds follow from the bound on uniform convergence (1.13) for sets of totally bounded functions that have finite VC dimension.

Case 2 [The set of unbounded functions] Consider the set of (nonnegative) unbounded functions $0 \le Q(z,\alpha),\ \alpha \in \Lambda$. It is easy to show (by constructing an example) that, without additional information about the set of unbounded functions and/or probability measures, it is impossible to obtain an inequality of type (1.20). Below, we use the following information

$$\sup_{\alpha\in\Lambda}\frac{\left(\int Q^p(z,\alpha)dP(z)\right)^{\frac{1}{p}}}{\int Q(z,\alpha)dP(z)} \le \tau < \infty \tag{1.22}$$

where $p > 1$ is some fixed constant[9].

The main result for the case of unbounded sets of loss functions is the following [20–22]:

Theorem 7 *With probability at least* $1 - \eta$ *the inequality*

$$R(\alpha) \le \frac{R_{emp}(\alpha)}{(1 - a(p)\tau\sqrt{\varepsilon})_+}, \quad a(p) = \sqrt[p]{\frac{1}{2}\left(\frac{p-1}{p-2}\right)^{p-1}} \tag{1.23}$$

holds true simultaneously for all functions of the set, where ε *is determined by (1.21),* $(a)_+ = max(a, 0)$.

The Theorem bounds the risks for all functions of the set (1.15) (including the function $Q(z, \alpha_\ell)$).

1.3.5 Problem of constructing rigorous (distribution dependent) bounds

To construct rigorous bounds for the rate of convergence one has to take into account information about the probability measure. Let $\mathcal{P}_0$ be a set of all probability measures and let $\mathcal{P} \subset \mathcal{P}_0$ be a subset of the set $\mathcal{P}_0$. We say that one has prior information about an unknown probability measure $P(z)$ if one knows the set of measures $\mathcal{P}$ that contains $P(z)$. Consider the following generalization of the Growth function

$$\mathcal{G}^{\Lambda}_{\mathcal{P}}(\varepsilon, \ell) = \lg \sup_{P \in \mathcal{P}} E_P N^{\Lambda}(\varepsilon; z_1, ..., z_\ell).$$

For indicator functions $Q(z, \alpha), \alpha \in \Lambda$ and for the extreme case where $\mathcal{P} = \mathcal{P}_0$ the Generalized Growth function $\mathcal{G}^{\Lambda}_{\mathcal{P}}(\varepsilon, \ell)$ coincides with the Growth function $G^{\Lambda}(\ell)$. For another extreme case where $\mathcal{P}$ contains only one function $P(z)$ the Generalized growth function coincides with the annealed VC-entropy.

The following assertion is true [20, 26]:

Theorem 8 *Suppose that a set of loss-functions is bounded*

$$-\inf < A \le Q(z, \alpha) \le B < \infty, \qquad \alpha \in \Lambda.$$

Then for sufficiently large ℓ *the following inequality*

$$P\left\{\sup_{\alpha \in \Lambda}\left|\int Q(z, \alpha)dF(z) - \frac{1}{\ell}\sum_{i=1}^{\ell} Q(z_i, \alpha)\right| > \varepsilon\right\} \le$$

[9] This inequality describes some general properties of distribution functions of the random variables $\xi_\alpha = Q(z, \alpha)$, generated by the $P(z)$. It describes the "tails of distributions" (the probability of large values for the random variables ξ_α). If the inequality (1.22) with $p > 2$ holds, then the distributions have so-called "light tails" (large values do not occurs very often). In this case rapid convergence is possible. If, however, the inequality (1.22) holds only for $p < 2$ (large values of the random variables ξ_α occur rather often) then the rate of convergence will be small (it will be arbitrarily small if p is sufficiently close to one).

$$12\exp\left\{\left(\frac{G^{\Lambda}_{P}\Lambda_{ann}(\varepsilon/6(B-A),2\ell)}{\ell}-\frac{\varepsilon^2}{B-A}+\frac{\ln\ell}{\ell}\right)\ell\right\}.$$

holds true.

From this bound it follows that for sufficiently large ℓ with probability $1-\eta$ simultaneously for all $\alpha \in \Lambda$ (including the one that minimizes the empirical risk) the following inequality is valid:

$$\int Q(z,\alpha)dF(z) \leq \frac{1}{\ell}\sum_{i=1}^{\ell} Q(z_i,\alpha) + \sqrt{\frac{\mathcal{G}^{\Lambda}_{P}(\varepsilon/6(B_A),2\ell)-\ln\eta/12}{\ell}}.$$

However, this bound is non-constructive because the theory does not specify a method for evaluating the Generalized Growth function. In order to make this bound constructive and rigorous one has to estimate the Generalized Growth function for a given set of loss-functions and a given set of probability measures. This is one of the main subjects of the current learning theory research.

1.4 Theory for Controlling the Generalization of Learning Machines

The theory for controlling the generalization of a learning machine is devoted to constructing an induction principle for minimizing the risk functional which takes into account the *size of the training set* (an induction principle for a *"small" sample size*[10]). The goal is to specify methods which are appropriate for a given sample size.

1.4.1 Structural risk minimization induction principle

The ERM principle is intended for dealing with a large sample size. Indeed, the ERM principle can be justified by considering the inequalities (1.20). When ℓ/h is large, the second summand on the right hand side of inequality (1.20) becomes small. The actual risk is then close to the value of the empirical risk. In this case, a small value of the empirical risk provides a small value of (expected) risk. However, if ℓ/h is small, then even a small $R_{emp}(\alpha_\ell)$ does not guarantee a small value of risk. In this case the minimization for $R(\alpha)$ requires a new principle, based on the simultaneous minimization of the two terms in inequality (1.20), one of which depends on the value of the empirical risk while the second depends on the VC-dimension of the set of functions. To minimize the risk in this case it is necessary to find a method which, along with minimizing the value of empirical risk, controls the VC-dimension of the learning machine.

The following principle, which is called the principle of Structural Risk Minimization (SRM), is intended for minimizing the risk functional with respect to both empirical risk and VC-dimension of the set of functions. Let, the set S of functions $Q(z,\alpha)$, $\alpha \in \Lambda$,

[10]The sample size ℓ is considered to be small if ℓ/h is small, say $\ell/h < 20$.

be provided with a *structure*: so that S is composed of the nested subsets of functions $S_k = \{Q(z,\alpha),\ \alpha \in \Lambda_k\}$, such that

$$S_1 \subset S_2 \subset ... \subset S_n ... \tag{1.24}$$

and $S^* = \bigcup_k S_k$. An *admissible structure* is one that satisfies the following three properties:

1. The set S^* is everywhere dense in S.
2. The VC-dimension h_k of each set S_k of functions is finite.
3. Any element S_k of the structure contains totally bounded functions $0 \le Q(z,\alpha) \le B_k,\ \alpha \in \Lambda_k$.

The SRM principle suggests to do the following: for a given set of observations $z_1, ..., z_\ell$ choose the element of structure S_n with $n = n(\ell)$ and the particular function from S_n such that the guaranteed risk (1.20) is minimal. The SRM principle actually suggests a *trade-off between the quality of the approximation and the complexity of the approximating function*. As n increases, empirical risk minima decrease, but on the other hand the term responsible for the confidence interval (summand in (1.20)) increases. The SRM principle takes both factors into account.

The main results of the theory of SRM are the following [9, 22]:

Theorem 9 *For any distribution function the SRM method provides convergence to the best possible solution with probability one.*

In other words SRM method is universally strongly consistent.

Theorem 10 [22] *For admissible structures the method of structural risk minimization provides approximations $Q(z,\alpha_\ell^{n(\ell)})$ for which the sequence of risks $R(\alpha_\ell^{n(\ell)})$ converge to the best one $R(\alpha_0)$ with asymptotic rate of convergence*[11]

$$V(\ell) = r_{n(\ell)} + B_{n(\ell)}\sqrt{\frac{h_{n(\ell)}\ln\ell}{\ell}} \tag{1.25}$$

if the law $n = n(\ell)$ is such that

$$\lim_{\ell\to\infty} \frac{B^2_{n(\ell)} h_{n(\ell)} \ln\ell}{\ell} = 0. \tag{1.26}$$

In equation (1.25) B_n is the bound for functions from S_n and $r_n(\ell)$ is the rate of approximation

$$r_n = \inf_{\alpha\in\Lambda_n} \int Q(z,\alpha)dP(z) - \inf_{\alpha\in\Lambda} \int Q(z,\alpha)dP(z).$$

[11] We say that the random variables ξ_ℓ, $\ell = 1,2,...$ converge to the value ξ_0 with asymptotic rate $V(\ell)$ if there exists constant C such that $V^{-1}(\ell)\,|\xi_\ell - \xi_0| \longrightarrow^P_{\ell\to\infty} C$.

1.5 Theory of Constructing Learning Algorithms

In order to implement the SRM induction principle in learning algorithms one has to control two factors in the to be minimized bound (1.20):

1. The value of empirical risk
2. The capacity factor (to choose the element S_n with the appropriate value of VC dimension).

We confine ourselves now to the pattern recognition case and consider two type of learning machines:

1. Neural Networks (NN) that were inspired on the biological analogy with the brain
2. The support vector machines that were inspired on statistical learning theory.

We discuss how each corresponding machine can control these factors.

1.5.1 Methods of separating hyperplanes and their generalization

Consider first the problem of minimizing empirical risk on the set of *linear indicator functions*

$$f(x, w) = \theta\left\{\sum_{i=0}^{n} w_i x^i\right\}, \ w \in W. \tag{1.27}$$

Let

$$(x_1, y_1), ..., (x_\ell, y_\ell)$$

be a training set where $x_j = (x_j^1, ..., x_j^n)$ is a vector, $y_j \in \{0, 1\}$, $j = 1, ..., \ell$.

For minimizing the empirical risk one has to find the parameters $w = (w_1, ..., w_n)$ (weights) which minimize the empirical risk functional

$$R_{emp}(w) = \frac{1}{\ell}\sum_{j=1}^{\ell}(y_j - f(x_j, w))^2. \tag{1.28}$$

There are several methods for minimizing this functional. In the case when the minimum of the empirical risk is zero one can find the exact solution while when the minimum of this functional is nonzero one can find an approximate solution. Therefore, by constructing a separating hyperplane one can control the value of empirical risk. Unfortunately, the set of separating hyperplanes is not flexible enough to provide low empirical risk for many real life problems [13].

Two opportunities were considered to increase the flexibility of the sets of functions:

1. to use a richer set of indicator functions which are superpositions of linear indicator functions
2. to map the input vectors into a high dimensional feature space and construct in this space a Δ-margin separating hyperplane.

The first idea corresponds to the neural network. The second idea leads to support vector machines.

1.5.2 Sigmoid approximation of indicator functions and neural nets

For describing the idea behind the NN let us consider the method of minimizing the functional (1.28). It is impossible to use regular *gradient-based* methods of optimization for minimizing this functional. The gradient of the indicator function $R_{emp}(w)$ is either equal to zero or is undefined. The solution is to approximate the set of indicator functions (1.27) by so-called *sigmoid functions*

$$\overline{f}(x,w) = S\left\{\sum_{i=0}^{n} w_i x^i\right\} \tag{1.29}$$

where $S(u)$ is a smooth monotonic function such that $S(-\infty) = 0$, $S(+\infty) = 1$. For example, the functions

$$S_1(u) = \frac{1}{1+\exp^{-u}}, \qquad S_2(u) = \frac{2\text{arctan(u)} + \pi}{2\pi}.$$

are sigmoid functions.

For the set of sigmoid function, the empirical risk functional

$$R_{emp}(w) = \frac{1}{\ell}\sum_{j=1}^{\ell}(y_j - \overline{f}(x_j,w))^2 \tag{1.30}$$

is smooth in w. It has a gradient $\text{grad}R_{emp}(w)$ and therefore can be minimized using gradient-based methods. For example, the *gradient descent method* uses the following update rule

$$w_{new} = w_{old} - \gamma(\cdot)\text{grad } R_{emp}(w_{old})$$

where the data $\gamma(\cdot) = \gamma(n) \geq 0$ depend on the iteration number n. For convergence of the gradient descent method to a local minimum, it is enough that $\gamma(n)$ satisfy the conditions

$$\sum_{n=1}^{\infty}\gamma(n) = \infty, \qquad \sum_{n=1}^{\infty}\gamma^2(n) < \infty.$$

Thus, the idea is to use the sigmoid approximation at the stage of estimating the coefficients, and use the indicator functions with these coefficients at the stage of recognition.

The generalization of this idea leads to feedforward neural nets. In order to increase the flexibility of the set of decision rules of the learning machine one considers a set of functions which are the superposition of several linear indicator functions (networks of neurons) [13] instead of the set of linear indicator functions (single neuron). All indicator functions in this superposition are replaced by sigmoid functions.

A method for calculating the gradient of the empirical risk for the sigmoid approximation of neural nets, called the *back-propagation method*, was found [15],[12]. Using this gradient descent method, one can determine the corresponding coefficient values (weights) of all elements of the neural net. In the 1990s it was proven that the VC dimension of neural networks depends on the type of sigmoid functions and the number of weights in the neural net. Under some general conditions the VC dimension of the

neural net is bounded (although it is sufficiently large). Suppose that the VC dimension does not change during the neural network training procedure, then the generalization ability of neural net depends on how well the neural net minimizes the empirical risk using a sufficiently large number of training data.

The three main problems encountered when minimizing the empirical risk using the back-propagation method are:

1. The empirical risk functional has many local minima. Optimization procedures guarantee convergence to some local minimum. In general the function which is found using the gradient-based procedure can be far from the best one. The quality of the obtained approximation depends on many factors, in particular on the initial parameter values of the algorithm.

2. Convergence to a local minimum can be rather slow (due to the high dimensionality of the weight-space).

3. The sigmoid function has a scaling factor which affects the quality of the approximation. To choose the scaling factor one has to make a trade-off between quality of approximation and the rate of convergence.

Therefore, a good minimization of the empirical risk depends in many respects on the art of the researcher in this case.

1.5.3 The optimal separating hyperplanes

For introducing the method that serves as an alternative to the neural network, let us consider optimal separating hyperplanes [25]. Suppose the training data

$$(x_1, y_1), \ldots, (x_\ell, y_\ell), \quad x \in R^n, \quad y \in \{+1, -1\}$$

can be separated by a hyperplane:

$$(w \cdot x) - b = 0. \tag{1.31}$$

We say that this set of vectors is separated by the *Optimal hyperplane (or the Maximal Margin hyperplane)* if it is separated without error and the distance between the closest vector and the hyperplane is maximal. To describe the separating hyperplane let us use the following form:

$$\begin{aligned} (w \cdot x_i) - b \geq 1 \qquad & \text{if } y_i = 1\ , \\ (w \cdot x_i) - b \leq -1 \qquad & \text{if } y_i = -1. \end{aligned}$$

In the following we use a compact notation for these inequalities:

$$y_i[(w \cdot x_i) - b] \geq 1, \quad i = 1, \ldots, \ell. \tag{1.32}$$

It is easy to check that the Optimal hyperplane is the one that satisfies the conditions (1.32) and minimizes functional

$$\Phi(w) = \frac{1}{2}||w||^2 = \frac{1}{2}(w, w). \tag{1.33}$$

The minimization is taken with respect to both vector w and scalar b.

The solution to this optimization problem is given by the saddle point of the Lagrange functional (Lagrangian):

$$L(w, b, \alpha) = \frac{1}{2}(w \cdot w) - \sum_{i=1}^{\ell} \alpha_i \{[(x_i \cdot w) - b]y_i - 1\}, \tag{1.34}$$

where the α_i are the Lagrange multipliers. The Lagrangian has to be minimized with respect to w, b and maximized with respect to $\alpha_i \geq 0$.

In the saddle point, the solutions w_0, b_0, and α^0 should satisfy the conditions

$$\frac{\partial L(w_0, b_0, \alpha^0)}{\partial b} = 0, \quad \frac{\partial L(w_0, b_0, \alpha^0)}{\partial w} = 0.$$

Rewriting these equations in explicit form one obtains the following properties of the Optimal hyperplane:

(i) The coefficients α_i^0 for the Optimal hyperplane should satisfy the constraints

$$\sum_{i=1}^{\ell} \alpha_i^0 y_i = 0, \qquad \alpha_i^0 \geq 0, \qquad i = 1, \ldots, \ell \tag{1.35}$$

(ii) The parameters of the Optimal hyperplane (vector w_0) are a linear combination of the vectors of the training set with

$$w_0 = \sum_{i=1}^{\ell} y_i \alpha_i^0 x_i, \qquad \alpha_i^0 \geq 0, \qquad i = 1, \ldots, \ell \tag{1.36}$$

(iii) The solution must satisfy the following Kuhn–Tucker conditions,

$$\alpha_i^0 \{[(x_i \cdot w_0) - b_0]y_i - 1\} = 0, \quad i = 1, \ldots, \ell. \tag{1.37}$$

From these conditions it follows that only some training vectors in expansion (1.36) (called the *support vectors*) can have nonzero coefficients α_i^0 in the expansion of w_0. The support vectors are the vectors for which, in inequality (1.36), the equality is achieved. Therefore we obtain

$$w_0 = \sum_{\text{support vectors}} y_i \alpha_i^0 x_i, \qquad \alpha_i^0 \geq 0. \tag{1.38}$$

Substituting the expression for w_0 back into the Lagrangian and taking into account the Kuhn–Tucker conditions, one obtains the functional

$$W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j}^{\ell} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j). \tag{1.39}$$

It remains to maximize this functional in the non-negative quadrant

$$\alpha_i \geq 0, \quad i = 1, ..., \ell$$

under the constraint

$$\sum_{i=1}^{\ell} \alpha_i y_i = 0. \tag{1.40}$$

Putting the expression for w_0 in (1.31), we obtain the hyperplane as an expansion on support vectors

$$\sum_{i=1}^{\ell} \alpha_i^0 (x, x_i) + b_0 = 0. \tag{1.41}$$

To construct the Optimal hyperplane in the case when the data are linearly non-separable, we introduce non-negative variables $\xi_i \geq 0$ and the functional

$$\Phi(\xi) = (w, w) + C \sum_{i=1}^{\ell} \xi_i$$

which we minimize subject to the constraints

$$y_i((w \cdot x_i) - b) \geq 1 - \xi_i, \qquad i = 1, 2, \ldots, \ell.$$

Using the same formalism with Lagrange multipliers one can show that the optimal hyperplane also has an expansion (1.41) on support vectors. The coefficients α_i can be found by maximizing the same quadratic form as in the separable case (1.39) under slightly different constraints

$$0 \leq \alpha_i \leq C, \quad i = 1, \ldots, \ell, \text{ and } \sum_{i=1}^{\ell} \alpha_i y_i = 0. \tag{1.42}$$

1.5.4 The support vector network

The support vector network implements the following idea [21]: Map the input vectors into a very high dimensional feature space Z through some non-linear mapping chosen *a priori*. Then construct an optimal separating hyperplane in this space. The goal is to create the situation as described previously in example 2, where for Δ-margin separating hyperplanes the VC dimension is defined by the ratio R^2/Δ^2. In order to generalize well, we control (decrease) the VC dimension by constructing an optimal separating hyperplane (that maximizes the margin). To increase the margin we use very high dimensional spaces.

Example. Consider a mapping that allows us to construct decision polynomials in the input space. To construct a polynomial of degree two, one can create a feature space, Z, which has $N = \frac{n(n+3)}{2}$ coordinates of the form:

$$z_1 = x_1, \ldots, z_n = x_n \ , \qquad n \text{ coordinates} \ ,$$

$$z_{n+1} = x_1^2, \ldots, z_{2n} = x_n^2 , \qquad n \text{ coordinates} ,$$

$$z_{2n+1} = x_1 x_2, \ldots, z_N = x_n x_{n-1} , \qquad \frac{n(n-1)}{2} \text{ coordinates} ,$$

where $x = (x_1, \ldots, x_n)$. The separating hyperplane constructed in this space is a separating second degree polynomial in the input space. To construct a polynomial of degree k in an n dimensional input space one has to construct a $O(n^k)$ dimensional feature space, where one then constructs the optimal hyperplane. The problem then arises of how to computationally deal with such high-dimensional spaces: for constructing a polynomial of degree 4 or 5 in a 200 dimensional space it is necessary to construct hyperplanes in a billion dimensional feature space.

In 1992 it was noted [5] that both for describing the optimal separating hyperplane in the feature space (1.41) and estimating the corresponding coefficients of expansion of the separating hyperplane (1.39) one uses the inner product between two vectors $z(x_1)$ and $z(x_2)$, which are images in the feature space of the input vectors x_1 and x_2. Therefore, if one can estimate the inner product of the two vectors in the feature space $z(x_1)$ and $z(x_2)$ as a function of two variables in the input space

$$(z_i \cdot z) = K(x, x_i),$$

then it will be possible to construct the solutions which are equivalent to the optimal hyperplane in the feature space. To get this solution one only needs to replace the inner product (x_i, x_j) in equations (1.39) and (1.41) by the function $K(x_i, x_j)$. In other words, one constructs nonlinear decision functions in the input space

$$I(x) = \text{sign}\left(\sum_{\text{support vectors}} \alpha_i K(x_i, x) + b_0 \right) , \tag{1.43}$$

that are equivalent to the linear decision functions (1.33) in the feature space. The coefficients α_i in (1.43) are defined by solving the equation

$$W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j}^{\ell} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{1.44}$$

under constraints (1.42). In 1909 Mercer proved a theorem which defines the general form of inner products in Hilbert spaces.

Theorem 11 *The general form of the inner product in Hilbert space is defined by the symmetric positive definite function $K(x, y)$ that satisfies the condition*

$$\int K(x, y) z(x) z(y) dx dy \geq 0$$

for all functions $z(x)$, $z(y)$ satisfying the inequality

$$\int z^2(x) dx \leq \infty.$$

Therefore any function $K(x, y)$ satisfying Mercer's condition can be used for constructing rule (1.42) which is equivalent to constructing an optimal separating hyperplane in some feature space. The learning machines which construct decision functions of the type (1.43) are called *Support Vectors Networks or Support Vector Machines*[12].

Using different expressions for inner products $K(x, x_i)$ one can construct different learning machines with arbitrary types of (nonlinear in the input space) decision surfaces. For example to specify polynomials of any fixed order d one can use the following functions for the inner product in the corresponding feature space

$$K(x, x_i) = ((x \cdot x_i) + 1)^d.$$

Radial Basis Function machines with decision functions of the form

$$f(x) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i \exp\left\{-\frac{|x - x_i|^2}{\sigma^2}\right\}\right)$$

can be implemented by using a function of the type

$$K(x, x_i) = \exp\left\{-\frac{|x - x_i|^2}{\sigma^2}\right\}.$$

In this case the SVM machine will find both the centers x_i and the corresponding weights α_i.

The SVM possesses some useful properties:

- The optimization problem for constructing an SVM has a unique solution.
- The learning process for constructing an SVM is rather fast.
- Simultaneously with constructing the decision rule, one obtains the set of support vectors.
- Implementation of a new set of decision functions can be done by changing only one function (kernel $K(x_i, x)$), which defines the dot product in Z-space.

1.5.5 Why can neural networks and support vectors networks generalize?

The generalization ability of both Neural Networks and Support Vectors Networks is based on the factors described in the theory for controlling the generalization of the learning processes. According to this theory, to guarantee a high rate of generalization of the learning machine one has to construct a structure

$$S_1 \subset S_2 \subset ... \subset S$$

[12]This name stresses that for constructing this type of machine, the idea of expanding the solution on support vectors is crucial. In the SVM the complexity of construction depends on the number of support vectors rather than on the dimensionality of the feature space.

on the set of decision functions $S = \{Q(z, \alpha), \alpha \in \Lambda\}$ and then choose both an appropriate element S_k of the structure and a function $Q(z, \alpha_\ell^k) \in S_k$ within this element that minimizes bound (1.20). The bound (1.16) can be rewritten in the simple form

$$R(\alpha_\ell^k) \leq R_{emp}(\alpha_\ell^k) + \Omega(\frac{\ell}{h_k}) \tag{1.45}$$

where the first term is an estimate of the risk and the second term is the confidence interval for this estimate.

In designing a neural network, one determines a set of admissible functions with some VC-dimension h^*. For a given amount ℓ of training data the value h^* determines the confidence interval $\Omega(\frac{\ell}{h^*})$ for the network. Choosing the appropriate element of a structure is therefore a problem of designing the network for a given training set. During the learning process this network minimizes the first term in the bound (1.45) (the number of errors on the training set). If it happens that at the stage of designing the network one constructs a network that is too complex (for the given amount of training data), the confidence interval $\Omega(\frac{\ell}{h^*})$ will be large. In this case, even if one could minimize the empirical risk as small as zero, the amount of errors on the test set can become big. This case is called *overfitting.* To avoid overfitting (and get a small confidence interval) one has to construct networks with small VC-dimension. Therefore, for generalizing well by using a neural network, one must first suggest an appropriate architecture of the neural network, and second, find in this network the function that minimizes the number of errors on the training data. For neural networks these two problems are solved by using some heuristics (see remarks on the back-propagation method).

In support vector methods one can control both parameters: in the separable case one obtains the unique solution that minimizes the empirical risk (down to zero) using a Δ-margin separating hyperplane with the maximal margin (i.e., subset with the smallest VC dimension). In the general case one obtains the unique solution when one chooses the value of the trade-off parameter C.

1.6 Conclusion

This chapter presented a very general overview of statistical learning theory. It demonstrates how an abstract analysis allows us to discover a general model of generalization.

According to this model, the generalization ability of learning machines depends on capacity concepts which are more sophisticated than merely the dimensionality of the space or the number of free parameters of the loss function (these concepts are the basis for the classical paradigm of generalization).

The new understanding of the mechanisms behind generalization not only changes the theoretical foundation of generalization (for example from this new viewpoint, the Occam razor principle is not always correct), but also changes the algorithmic approaches to function estimation problems. The approach described is rather general. It can be applied for various function estimation problems including regression, density estimation, solving inverse equations and so on.

Statistical Learning Theory started more than 30 years ago. The development of this theory did not involve many researchers. After the success of the SVM in solving

real life problems, the interest in statistical learning theory significantly increased. For the first time, abstract mathematical results in statistical learning theory have a direct impact on algorithmic tools of data analysis. In the last three years a lot of articles have appeared that analyze the theory of inference and the SVM method from different perspectives. These include:

1. Obtaining improved constructive bounds instead of the classical ones described in this chapter (which are more in the spirit of the non-constructive bound based on the Growth function than on bounds based on the VC dimension concept). Success in this direction could lead, in particular, to creating machines that generalize better than the SVM based on the concept of optimal hyperplane.

2. Extending the SVM ideology to many different problems of function and data-analysis.

3. Developing a theory that allows us to create kernels that possess desirable properties (for example that can enforce desirable invariants).

4. Developing a new type of induction inference that is based on direct generalization from the training set to the test set, avoiding the intermediate problem of estimating a function (the transductive type inference).

The hope is that this very fast growing area of research will significantly boost all branches of data analysis.

Bibliography

[1] N. Alon, S. Ben-David, N. Cesa-Bianchi, and D. Haussler, Scale-sensitive dimensions, uniform convergence, and learnability, *Journal of the ACM* **44**(4) (1997) 617–631.

[2] P.L. Bartlett, P. Long, and R.C. Williamson, Fat-shattering and the learnability of real-valued functions, *Journal of Computer and System Sciences* **52**(3) (1996) 434–452.

[3] P.L. Bartlett and J. Shawe-Taylor, Generalization performance on support vector machines and other pattern classifiers, In B. Schölkopf, C. Burges, and A. Smola (ed) *Advances in Kernel Methods. Support Vector Learning*, The MIT Press (1999)

[4] A. Blumer, A. Ehrenfeucht, D. Haussler, and M.K. Warmuth, Learnability and the Vapnik-Chervonenkis Dimension, *Journal of the ACM* **36**(4) (1989) 929–965.

[5] B. Boser, I. Guyon, and V.N. Vapnik, A training algorithm for optimal margin classifiers, *Fifth Annual Workshop on Computational Learning Theory*, Pittsburgh ACM (1992) 144–152.

[6] C.J.C. Burges, Simplified support vector decision rule, *Proceedings of 13th Intl. Conf. on Machine Learning*, San Mateo, CA (1996) 71–77.

[7] C.J.C. Burges, Geometry and invariance in kernel based methods, In B. Schölkopf, C. Burges, and A. Smola (ed) *Advances in Kernel Methods. Support Vector Learning*, MIT Press (1999).

[8] C. Cortes and V.N. Vapnik, Support Vector Networks, *Machine Learning* **20** (1995) 273–297.

[9] L. Devroye, L. Gyorfi and G. Lugosi, *A Probability Theory of Pattern recognition*, Springer, N.Y. (1996).

[10] F. Girosi, An equivalence between sparse approximation and support vector machines, *Neural Computation* **10**(6) (1998) 1455–1480.

[11] F. Girosi, M. Jones, and T. Poggio, Regularization theory and neural networks architectures, *Neural Computation* **7**(2) (1995) 219–269.

[12] Y. Le Cun, Learning processes in an asymmetric threshold network, In E. Beinenstock, F. Fogelman-Soulie, and G. Weisbuch (ed) *Disordered systems and biological organizations*, Les Houches, France, Springer-Verlag (1986) 233–240.

[13] M.L. Minsky and S.A. Papert, *Perceptrons*, MIT Press (1969).

[14] M. Opper, On the annealed VC entropy for margin classifiers: a statistical mechanics study, In B. Schölkopf, C. Burges, and A. Smola (ed) *Advances in Kernel Methods, support Vector Learning,* MIT Press (1999).

[15] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, Learning internal representations by error propagation, *Parallel distributed processing: Explorations in macrostructure of cognition*, Vol. I, Badford Books, Cambridge, MA (1986) 318–362.

[16] J. Shawe-Taylor, P.L. Bartlett, R. C. Williamson, and M. Anthony, Structural risk minimization over data-dependent hierarchies, *IEEE Transactions on Information Theory* **44**(5) (1998) 1926–1940.

[17] B. Schölkopf, A Smola, and K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, *Neural Computation* **10** (1998) 1229–1319.

[18] A. Smola, B. Schölkopf and K-R. Müller, The connection between regularization operators and support vector kernels, *Neural Networks* 11 (1998) 637–649.

[19] M. Talagrand, The Glivenko-Cantelli problem, ten years later, *Journal of Theoretical Probability* **9**(2) (1996) 371–384.

[20] V.N. Vapnik, *Estimation of Dependencies Based on Empirical Data*, [in Russian], Nauka, Moscow (1979) (English translation: (1982) Springer-Verlag, New York).

[21] V.N. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, New-York (1995).

[22] V.N. Vapnik, *Statistical Learning Theory*, John Wiley, New-York (1998).

[23] V.N. Vapnik and A.Ja. Chervonenkis, On the uniform convergence of relative frequencies of events to their probabilities, Reports of Academy of Science USSR **181**(4) (1968).

[24] V.N. Vapnik and A.Ja. Chervonenkis, On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl* **16** (1971) 264–280.

[25] V.N. Vapnik and A.Ja. Chervonenkis, *Theory of Pattern Recognition* [in Russian], Nauka, Moscow (1974) (German translation: W. N. Wapnik and A. Ja. Cherwonenkis (1979) *Theorie der Zeichenerkennung*, Akademia-Verlag, Berlin)

[26] V.N. Vapnik and A.Ja. Chervonenkis, Necessary and sufficient conditions for the uniform convergence of the means to their expectations, *Theory Probab. Appl.* **26** (1981) 532–553.

[27] V.N. Vapnik and A.Ja. Chervonenkis, The necessary and sufficient conditions for consistency of the method of empirical risk minimization [in Russian] (1989) *Yearbook of the Academy of Sciences of the USSR* on *Recognition, Classification, and Forecasting* Nauka Moscow **2** (1989) 217–249 (English translation: (1991) *Pattern Recogn. and Image Analysis*, Vol. 1, No. 3, 284-305).

[28] M. Vidyasagar, *A theory of learning and generalization*, Springer, N.Y. (1997).

[29] G. Wahba, *Spline Models for observational data*, SIAM Vol. 59, Philadelphia (1990).

[30] R.C. Williamson, A. Smola, and B. Schölkopf, Entropy numbers, operators, and support vector kernels, In B. Schölkopf, C. Burges, and A. Smola (ed) *Advances in Kernel Methods. Support Vector Learning*. The MIT Press (1999).

Chapter 2

Best Choices for Regularization Parameters in Learning Theory: On the Bias-Variance Problem

Felipe Cucker and Steve Smale[1]

[1]This chapter is reprinted with permission from Felipe Cucker, Steve Smale, "Best Choices for Regularization Parameters in Learning Theory: On the Bias - Variance Problem," Foundations of Computational Mathematics, Vol.2, No.4, pp.413-428, 2002 (Copyright © 2002 Springer-Verlag). This work has been substantially funded by a grant from the Research Grants Council of the Hong Kong SAR (project number CityU 1002/99P). Also, the second named author expresses his appreciation to City University of Hong Kong for supporting his visit there in December 2001, when this paper was written. AMS classification: 68T05, 62J02.

2.1 Introduction

The goal of learning theory (and a goal in some other contexts as well) is to find an approximation of a function $f_\rho : X \to Y$ known only through a set of pairs $\mathbf{z} = (x_i, y_i)_{i=1}^m$ drawn from an unknown probability measure ρ on $X \times Y$ (f_ρ is the "regression function" of ρ).

An approach championed by Poggio (see e.g. [5]) with ideas going back to Ivanov [7] and Tikhonov [13] is to minimize

$$\frac{1}{m}\sum_{i=1}^{m}(f(x_i) - y_i)^2 + \gamma\|Af\|^2_{\mathcal{L}^2_\rho(X)}$$

where A is an differential operator and $\mathcal{L}^2_\rho(X)$ is the Hilbert space of square integrable functions on X with measure ρ_X on X defined via ρ. See [9] (in the sequel denoted by [CS]) for background on this and, even more importantly, for results used here.

This minimization is well-conditioned and solved by straightforward finite dimensional least squares linear algebra (see Theorem 1 below) to yield $f_{\gamma,\mathbf{z}} : X \to Y$. The problem is posed: *How good an approximation is $f_{\gamma,\mathbf{z}}$ to f_ρ, or measure the error $\int_X (f_{\gamma,\mathbf{z}} - f_\rho)^2$?* and *What is the best choice of γ to minimize this error?*

Our goal in this report is to give some answers to these questions.

Main result. *We exhibit, for each $m \in \mathbb{N}$ and $\delta \in [0,1)$, a function*

$$E_{m,\delta} = E : \mathbb{R}^+ \to \mathbb{R}$$

such that, for all $\gamma > 0$,

$$\int_X (f_{\gamma,\mathbf{z}} - f_\rho)^2 \le E(\gamma)$$

with confidence $1 - \delta$. There is a unique minimizer of $E(\gamma)$ which is found by an easy algorithm to yield the "best" regularization parameter $\gamma = \gamma^*$.

The bound $E(\gamma)$ found is a natural one, a bound which flows from the hypotheses and thus yields a γ^* which could be useful in the algorithmics for $f_{\gamma,\mathbf{z}}$. Of course, γ^* depends on the number of examples m, confidence $1 - \delta$, as well as the operator A and a simple invariant of ρ.

2.2 RKHS and Regularization Parameters

Let X be a compact domain or a manifold in Euclidean space and $Y = \mathbb{R}$ (one can extend all what follows to $Y = \mathbb{R}^k$ with $k \in \mathbb{N}$). Let ρ be a Borel probability measure on $Z = X \times Y$.

For every $x \in X$, let $\rho(y|x)$ be the conditional (w.r.t. x) probability measure on Y and ρ_X be the marginal probability measure on X, i.e. the measure on X defined by $\rho_X(S) = \rho(\pi^{-1}(S))$ where $\pi : X \times Y \to X$ is the projection. Notice that ρ, $\rho(y|x)$ and ρ_X are related as follows. For every integrable function $\varphi : X \times Y \to \mathbb{R}$ a version of Fubini's Theorem states that

$$\int_{X\times Y} \varphi(x,y)\, d\rho = \int_X \left(\int_Y \varphi(x,y)\, d\rho(y|x)\right) d\rho_X.$$

This "breaking" of ρ into the measures $\rho(y|x)$ and ρ_X corresponds to looking at Z as a product of an input domain X and an output set Y.

Define $f_\rho : X \to Y$ by

$$f_\rho(x) = \int_Y y \, d\rho(y|x).$$

The function f_ρ is called the *regression function* of ρ. For each $x \in X$, $f_\rho(x)$ is the average of the y coordinate of $\{x\} \times Y$.

In what follows we assume $f_\rho \in \mathcal{L}^2_\rho(X)$ is bounded. We also assume that

$$M_\rho = \inf\left\{\overline{M} \geq 0 \mid \{(x,y) \in Z \mid |y - f_\rho(x)| \geq \overline{M}\} \text{ has measure zero}\right\}$$

is finite. Note that this implies that

$$|y| \leq \mathrm{M} = \max\{\|f_\rho\|_\infty + M_\rho, 1\}$$

almost surely.

Recall, $\|f\|$ denotes, unless otherwise specified, the norm of f in $\mathcal{L}^2_\rho(X)$. Let K be a *Mercer kernel*. That is, $K : X \times X \to \mathbb{R}$ is continuous, symmetric and K is *positive semidefinite*, i.e. for all finite sets $\{x_1, \ldots, x_k\} \subset X$ the $k \times k$ matrix $K[\mathrm{x}]$ whose (i,j) entry is $K(x_i, x_j)$ is positive semidefinite. Then (cf. Chapter III of [CS]) K determines a linear operator $L_K : \mathcal{L}^2_\rho(X) \to \mathscr{C}(X)$ given by

$$(L_K f)(x) = \int K(x,t) f(t) dt$$

which is well-defined, positive, and compact. In addition, there exists a Hilbert space $\mathcal{H}_K$ of continuous functions on X (called *reproducing kernel Hilbert space*, RKHS for short) associated to K and X and independent of ρ such that the linear map $L_K^{1/2}$ is a Hilbert isomorphism between $\mathcal{L}^2_\rho(X)$ and $\mathcal{H}_K$. Here $L_K^{1/2}$ denotes the square root of L_K, i.e. the only linear operator satisfying $L_K^{1/2} \circ L_K^{1/2} = L_K$. Thus, we have the following diagram

$$\begin{array}{ccc} \mathcal{L}^2_\rho(X) & \xrightarrow{L^{1/2}_{K,\mathscr{C}}} & \mathscr{C}(X) \\ & \searrow^{L_K^{1/2}} \quad \approx & \uparrow I_K \\ & & \mathcal{H}_K \end{array}$$

where we write $L_{K,\mathscr{C}}$ to emphasize that the target is $\mathscr{C}(X)$ and I_K denotes the inclusion. If K is $\mathscr{C}^\infty$ then I_K is compact. In the sequel, K denotes a $\mathscr{C}^\infty$ Mercer kernel, and $\| \ \|_K$ the norm in $\mathcal{H}_K$.

Let $\mathbf{z} = (z_1, \ldots, z_m)$ with $z_i = (x_i, y_i) \in X \times Y$ for $i = 1, \ldots, m$. We also write $\mathbf{x} = (x_1, \ldots, x_m)$, $\mathbf{y} = (y_1, \ldots, y_m)$. Note that since K is a Mercer kernel $K[\mathrm{x}]$ is positive semidefinite.

For $\gamma > 0$ let $\Phi(\gamma)$ and $\Phi_{\mathbf{z}}(\gamma)$ be the problems respectively

$$\begin{array}{ll} \min & \displaystyle\int_X (f(x) - y)^2 + \gamma \|f\|_K^2 \\ \text{s.t.} & f \in \mathcal{H}_K \end{array}$$

and

$$\begin{aligned} \min \quad & \frac{1}{m}\sum_{i=1}^{m}(f(x_i)-y_i)^2+\gamma\|f\|_K^2 \\ \text{s.t.} \quad & f\in\mathcal{H}_K. \end{aligned}$$

For $x \in X$, let $K_x : X \to \mathbb{R}$ be given by $K_x(t) = K(x,t)$.

Theorem 1 *For all $\gamma > 0$, the minimizers f_γ and $f_{\gamma,\mathbf{z}}$ of $\Phi(\gamma)$ and $\Phi_{\mathbf{z}}(\gamma)$ respectively exist and are unique. In addition*

$$f_\gamma = (\mathrm{Id} + \gamma L_K^{-1})^{-1} f_\rho$$

and $f_{\gamma,\mathbf{z}}$ is given by

$$f_{\gamma,\mathbf{z}}(x) = \sum_{i=1}^{m} a_i K(x, x_i)$$

where $a = (a_1, \ldots, a_m)$ is the unique solution of the well-posed linear system in $\mathbb{R}^m$

$$(\gamma m \mathrm{Id} + K[\mathbf{x}])a = \mathbf{y}.$$

Finally, for $f = \sum_{i=1}^m a_i K_{x_i}$ we have $\|f\|_K^2 = a^{\mathrm{T}} K[\mathbf{x}]a$.

PROOF. See Propositions 7 and 8 and Theorem 2 in Chapter III of [CS] and its references, and [5] and its references. □

2.3 Estimating the Confidence

Define, for $f \in \mathcal{L}_\rho^2(X)$, its *error*

$$\mathcal{E}(f) = \int_Z (f(x) - y)^2$$

and, given a sample $\mathbf{z} \in Z^m$, its *empirical error*

$$\mathcal{E}_{\mathbf{z}}(f) = \frac{1}{m}\sum_{i=1}^{m}(f(x_i) - y_i)^2.$$

Note that from the equality $\mathcal{E}(f_{\gamma,\mathbf{z}}) = \mathcal{E}(f_{\gamma,\mathbf{z}}) - \mathcal{E}(f_\gamma) + \mathcal{E}(f_\gamma)$ we deduce

$$\mathcal{E}(f_{\gamma,\mathbf{z}}) \le |\mathcal{E}(f_{\gamma,\mathbf{z}}) - \mathcal{E}(f_\gamma)| + \mathcal{E}(f_\gamma)$$

We will call the first term in the right-hand side the *sample error* (this use of the expression slightly differs from the one in [CS]) and the second, the *approximation error*. Note that the sample error is a random variable on the space Z^m. In this section we will bound the confidence for the sample error to be small enough. The main result is Theorem 2 below.

For $r > 0$ let $B_r = \{f \in \mathcal{H}_K \mid \|f\|_K \leq r\}$ and $\mathcal{H}(r) = \overline{I_K(B_r)}$. Notice that this is a compact subset of $\mathscr{C}(X)$ so that, for every η, the *covering number*

$$\mathcal{N}(\mathcal{H}(r), \eta) = \min\{s \in \mathbb{N} \mid \exists\ s \text{ closed balls of radius } \eta \text{ in } \mathscr{C}(X) \text{ covering } \mathcal{H}(r)\}$$

is finite. Also, let

$$\mathbf{C}_K = \max\left\{1, \sup_{x,t\in X} |K(x,t)|\right\}$$

and

$$R_\gamma = \frac{\sqrt{\mathbf{C}_K}\|f_\rho\|_\infty}{\gamma}, \qquad \text{and} \qquad r_\gamma = \frac{\sqrt{\mathbf{C}_K}\mathbf{M}}{\gamma}.$$

Theorem 2 *For all $\gamma, \epsilon > 0$,*

$$\operatorname*{Prob}_{\mathbf{z}\in Z^m}\{|\mathcal{E}(f_\gamma) - \mathcal{E}(f_{\gamma,\mathbf{z}})| \leq \epsilon\} \geq 1 - 4\left[m + \mathcal{N}\left(\mathcal{H}(r_\gamma), \frac{\epsilon\gamma}{32\mathbf{M}(\gamma + \mathbf{C}_K)}\right)\right] e^{-\frac{m\epsilon^2\gamma^4}{128\mathbf{M}^4(\gamma+\mathbf{C}_K)^4}}.$$

The idea towards the proof of Theorem 2 is to write

$$\mathcal{E}(f_\gamma) - \mathcal{E}(f_{\gamma,\mathbf{z}}) = \mathcal{E}(f_\gamma) - \mathcal{E}_\mathbf{z}(f_\gamma) + \mathcal{E}_\mathbf{z}(f_\gamma) - \mathcal{E}_\mathbf{z}(f_{\gamma,\mathbf{z}}) + \mathcal{E}_\mathbf{z}(f_{\gamma,\mathbf{z}}) - \mathcal{E}(f_{\gamma,\mathbf{z}})$$

from which it follows that

$$|\mathcal{E}(f_\gamma) - \mathcal{E}(f_{\gamma,\mathbf{z}})| \leq |\mathcal{E}(f_\gamma) - \mathcal{E}_\mathbf{z}(f_\gamma)| + |\mathcal{E}_\mathbf{z}(f_\gamma) - \mathcal{E}_\mathbf{z}(f_{\gamma,\mathbf{z}})| + |\mathcal{E}_\mathbf{z}(f_{\gamma,\mathbf{z}}) - \mathcal{E}(f_{\gamma,\mathbf{z}})|.$$

We first, see Proposition 1 below, bound (with high confidence) the first and last terms in the sum above. Towards this end we give bounds on $\|f_\gamma\|_K, \|f_{\gamma,\mathbf{z}}\|_K, \|f_\gamma\|_\infty$, and $\|f_{\gamma,\mathbf{z}}\|_\infty$.

Lemma 1 *For all $\gamma > 0$*

$$\|f_\gamma\|_K \leq R_\gamma.$$

PROOF. Let $f_\rho = \sum c_i\phi_i$. Then

$$f_\gamma = \sum_{i=1}^{\infty}\left(1 + \frac{\gamma}{\lambda_i}\right)^{-1} c_i\phi_i = \sum_{i=1}^{\infty}\left(\frac{\lambda_i}{\gamma + \lambda_i}\right) c_i\phi_i$$

and therefore

$$\begin{aligned}
\|f_\gamma\|_K^2 &= \sum_{i=1}^{\infty}\frac{\lambda_i}{(\gamma+\lambda_i)^2}c_i^2 \\
&\leq \max_{i\geq 1}\frac{\lambda_i}{(\gamma+\lambda_i)^2}\sum_{i=1}^{\infty}c_i^2 \\
&\leq \frac{1}{\gamma^2}\max_{i\geq 1}\lambda_i\,\|f_\rho\|^2 \\
&\leq \frac{\mathbf{C}_K}{\gamma^2}\|f_\rho\|^2.
\end{aligned}$$

□

Lemma 2 *For all $\gamma > 0$*

$$\|f_{\gamma,\mathbf{z}}\|_K \leq r_\gamma.$$

PROOF. Since $f_{\gamma,\mathbf{z}} = \sum a_i K_{x_i}$ we have $\|f_{\gamma,\mathbf{z}}\|_K^2 = a^{\mathrm{T}} K[\mathbf{x}]a$.

Also, since $a = (\gamma m \mathrm{Id} + K[\mathbf{x}])^{-1}\mathbf{y}$ it follows that

$$\|a\| \leq \|\mathbf{y}\|\|(\gamma m \mathrm{Id} + K[\mathbf{x}])^{-1}\| \leq \sqrt{m}\mathbf{M}\frac{1}{\gamma m} = \frac{\mathbf{M}}{\gamma\sqrt{m}}$$

where $\|a\|$ and $\|\mathbf{y}\|$ refer to the Euclidean norm in $\mathbb{R}^m$. Therefore

$$\|f_{\gamma,\mathbf{z}}\|_K^2 \leq \|a\|^2\|K[\mathbf{x}]\| \leq \frac{\mathbf{M}^2}{\gamma^2 m}\mathbf{C}_K m = \frac{\mathbf{M}^2}{\gamma^2}\mathbf{C}_K$$

where $\|K[\mathbf{x}]\|$ denotes the operator norm of $K[\mathbf{x}] : \mathbb{R}^m \to \mathbb{R}^m$ with respect to the Euclidean norm in both domain and target space and we have used that, since each entry of $K[\mathbf{x}]$ is bounded in absolute value by $\mathbf{C}_K$, $\|K[\mathbf{x}]\| \leq \mathbf{C}_K m$. □

Corollary 1 *For all $\gamma > 0$, $\|f_\gamma\|_\infty \leq \frac{\mathbf{C}_K\|f_\rho\|_\infty}{\gamma}$ and $\|f_{\gamma,\mathbf{z}}\|_\infty \leq \frac{\mathbf{C}_K\mathbf{M}}{\gamma}$.*

PROOF. By Theorem 2 in Chapter III of [CS], $\|I_K\| \leq \sqrt{\mathbf{C}_K}$. □

Remark 1 *Note that for all $\gamma > 0$, $r_\gamma \geq R_\gamma$.*

Proposition 1 *For all $\epsilon, \gamma > 0$,*

(i)

$$\operatorname*{Prob}_{\mathbf{z}\in Z^m}\{|\mathcal{E}(f_\gamma) - \mathcal{E}_{\mathbf{z}}(f_\gamma)| \leq \epsilon\} \geq 1 - \mathcal{N}\left(\mathcal{H}(R_\gamma), \frac{\epsilon\gamma}{8\mathbf{M}(\gamma + \mathbf{C}_K)}\right) 2e^{-\frac{m\epsilon^2\gamma^4}{8\mathbf{M}^4(\gamma+\mathbf{C}_K)^4}}.$$

(ii)

$$\operatorname*{Prob}_{\mathbf{z}\in Z^m}\{|\mathcal{E}(f_{\gamma,\mathbf{z}}) - \mathcal{E}_{\mathbf{z}}(f_{\gamma,\mathbf{z}})| \leq \epsilon\} \geq 1 - \mathcal{N}\left(\mathcal{H}(r_\gamma), \frac{\epsilon\gamma}{8\mathbf{M}(\gamma + \mathbf{C}_K)}\right) 2e^{-\frac{m\epsilon^2\gamma^4}{8\mathbf{M}^4(\gamma+\mathbf{C}_K)^4}}.$$

PROOF. We use Theorem B of [CS] but proved with Hoeffding's inequality instead of Bernstein's. This yields, for a compact subset $\mathcal{H}$ of $\mathscr{C}(X)$ such that $|f(x) - y| \leq M$ a.e. for all $f \in \mathcal{H}$, the uniform estimate

$$\operatorname*{Prob}_{\mathbf{z}\in Z^m}\left\{\sup_{f\in\mathcal{H}}|\mathcal{E}(f) - \mathcal{E}_{\mathbf{z}}(f)| \leq \epsilon\right\} \geq 1 - \mathcal{N}\left(\mathcal{H}, \frac{\epsilon}{8M}\right) 2e^{-\frac{m\epsilon^2}{8M^4}}.$$

For (i) use this estimate applied to $\mathcal{H} = \mathcal{H}(R_\gamma)$, and

$$M = \|f_\gamma\|_\infty + M_\rho + \|f_\rho\|_\infty \leq \frac{\mathbf{C}_K\|f_\rho\|_\infty}{\gamma} + \mathbf{M} \leq \frac{\mathbf{M}(\gamma + \mathbf{C}_K)}{\gamma}.$$

and note that

$$\operatorname*{Prob}_{\mathbf{z}\in Z^m}\{|\mathcal{E}(f_\gamma) - \mathcal{E}_{\mathbf{z}}(f_\gamma)| \leq \epsilon\} \geq \operatorname*{Prob}_{\mathbf{z}\in Z^m}\left\{\sup_{f\in\mathcal{H}(R_\gamma)}|\mathcal{E}(f) - \mathcal{E}_{\mathbf{z}}(f)| \leq \epsilon\right\}.$$

A similar proof, now with $\mathcal{H} = \mathcal{H}(r_\gamma)$, and

$$M = \|f_\gamma\|_\infty + M_\rho + \|f_\rho\|_\infty \leq \frac{\mathrm{C}_K \mathrm{M}}{\gamma} + \mathrm{M} = \frac{\mathrm{M}(\gamma + \mathrm{C}_K)}{\gamma}.$$

yields (ii). □

We now proceed with the middle term $|\mathcal{E}_{\mathbf{z}}(f_\gamma) - \mathcal{E}_{\mathbf{z}}(f_{\gamma,\mathbf{z}})|$.

In what follows, for $f : X \to \mathbb{R}$ and $\mathrm{x} \in X^m$, we denote by $f[\mathrm{x}]$ the point $(f(x_1), \ldots, f(x_m)) \in \mathbb{R}^m$. Also, if $v \in \mathbb{R}^m$, we denote $\|v\|_{\max} = \max\{|v_1|, \ldots, |v_m|\}$.

Proposition 2 *For all $\gamma, \epsilon > 0$,*

$$\operatorname*{Prob}_{\mathbf{z} \in Z^m} \left\{ \|f_\gamma[\mathrm{x}] - f_{\gamma,\mathbf{z}}[\mathrm{x}]\|_{\max} \leq 2\epsilon \right\} \geq 1 - 4me^{-\frac{m\epsilon^2\gamma^4}{2\mathrm{C}_K^2 \mathrm{M}^2(\gamma + \mathrm{C}_K)^2}}.$$

Proof of Theorem 2. Recall,

$$|\mathcal{E}(f_\gamma) - \mathcal{E}(f_{\gamma,\mathbf{z}})| \leq |\mathcal{E}(f_\gamma) - \mathcal{E}_{\mathbf{z}}(f_\gamma)| + |\mathcal{E}_{\mathbf{z}}(f_\gamma) - \mathcal{E}_{\mathbf{z}}(f_{\gamma,\mathbf{z}})| + |\mathcal{E}_{\mathbf{z}}(f_{\gamma,\mathbf{z}}) - \mathcal{E}(f_{\gamma,\mathbf{z}})|.$$

The first and last terms are each bounded by ϵ with probabilities at least

$$1 - \mathcal{N}\left(\mathcal{H}(r_\gamma), \frac{\epsilon\gamma}{8\mathrm{M}(\gamma + \mathrm{C}_K)}\right) 2e^{-\frac{m\epsilon^2\gamma^4}{8\mathrm{M}^4(\gamma + \mathrm{C}_K)^4}}$$

by Proposition 1 and the fact that $r_\gamma \geq R_\gamma$. For the middle term note that

$$\begin{aligned} |\mathcal{E}_{\mathbf{z}}(f_\gamma) - \mathcal{E}_{\mathbf{z}}(f_{\gamma,\mathbf{z}})| &= \frac{1}{m}\left|\sum_{i=1}^m (f_\gamma(x_i) - y_i) - \sum_{i=1}^m (f_{\gamma,\mathbf{z}}(x_i) - y_i)\right| \\ &\leq \frac{1}{m}\sum_{i=1}^m |f_\gamma(x_i) - f_{\gamma,\mathbf{z}}(x_i)| \\ &\leq \|f_\gamma[\mathrm{x}] - f_{\gamma,\mathbf{z}}[\mathrm{x}]\|_{\max}. \end{aligned}$$

Now apply Proposition 2 to bound this term by 2ϵ with probability at least

$$1 - 4me^{-\frac{m\epsilon^2\gamma^4}{2\mathrm{C}_K^2 \mathrm{M}^2(\gamma + \mathrm{C}_K)^2}}$$

and the conclusion follows by noting that $2\mathrm{C}_K^2\mathrm{M}^2(\gamma + \mathrm{C}_K)^2 \leq 8\mathrm{M}^4(\gamma + \mathrm{C}_K)^4$ and by replacing ϵ by $\epsilon/4$. □

It only remains to prove Proposition 2. Towards this end, recall, Hoeffding's inequality states that if ξ is a random variable on a probability space Z bounded almost everywhere by M with mean μ then

$$\operatorname*{Prob}_{\mathbf{z} \in Z^m} \left\{ \left| \frac{1}{m}\sum_{i=1}^m \xi(z_i) - \mu \right| \leq \epsilon \right\} \geq 1 - 2e^{-\frac{m\epsilon^2}{2M}}.$$

Lemma 3 *For all $\gamma, \epsilon > 0$ and all $t \in X$,*

$$\operatorname*{Prob}_{\mathbf{z}\in Z^m}\left\{\left|\frac{1}{m\gamma}\sum_{i=1}^{m} K(x_i,t)(f_\rho(x_i)-y_i)\right| \le \epsilon\right\} \ge 1 - 2e^{-\frac{m\epsilon^2\gamma^2}{2(\mathbf{C}_K M_\rho)^2}}.$$

PROOF. Consider the random variable

$$z \mapsto \frac{1}{\gamma}K(x,t)(f_\rho(x)-y).$$

It is almost everywhere bounded by $\frac{1}{\gamma}\mathbf{C}_K M_\rho$. Its mean is 0 since, by Fubini's Theorem

$$\int_Z \frac{1}{\gamma}K(x,t)(f_\rho(x)-y) = \int_X \frac{1}{\gamma}K(x,t)\left(\int_Y f_\rho(x)-y\, d\rho(y|x)\right) d\rho_X$$

and the inner integral is 0 by definition of f_ρ. Now apply Hoeffding's inequality. □

Lemma 4 *For all $\gamma, \epsilon > 0$ and all $t \in X$,*

$$\operatorname*{Prob}_{\mathbf{z}\in Z^m}\left\{\left|f_\gamma(t) - \frac{1}{m\gamma}\sum_{i=1}^{m} K(x_i,t)(f_\rho(x_i)-f_\gamma(x_i))\right| \le \epsilon\right\} \ge 1 - 2e^{-\frac{m\epsilon^2\gamma^2}{2\mathbf{C}_K^2(\|f_\rho\|_\infty+\sqrt{\mathbf{C}_K}R_\gamma)^2}}.$$

PROOF. By Theorem 1,

$$\begin{aligned} f_\gamma = (\mathrm{Id}+\gamma L_K^{-1})^{-1} f_\rho &\Rightarrow f_\gamma + \gamma L_K^{-1} f_\gamma = f_\rho \\ &\Rightarrow L_K f_\gamma + \gamma f_\gamma = L_K f_\rho \\ &\Rightarrow f_\gamma = \frac{1}{\gamma} L_K(f_\rho - f_\gamma) \\ &\Rightarrow f_\gamma(t) = \int_X \left(\frac{1}{\gamma}K(x,t)(f_\rho(x)-f_\gamma(x))\right) d\rho_X. \end{aligned}$$

The function inside the last integral can be thus considered a random variable on X with mean $f_\gamma(t)$. It is bounded by $\frac{\mathbf{C}_K}{\gamma}(\|f_\rho\|_\infty + \sqrt{\mathbf{C}_K}R_\gamma)$. Again, apply Hoeffding's inequality. □

Lemma 5 *For all $\gamma, \epsilon > 0$,*

$$\operatorname*{Prob}_{\mathbf{z}\in Z^m}\left\{\left\|\left(\mathrm{Id}+\frac{K[\mathbf{x}]}{\gamma m}\right) f_\gamma[\mathbf{x}] - \frac{K[\mathbf{x}]\mathbf{y}}{\gamma m}\right\|_{\max} \le 2\epsilon\right\} \ge 1 - 4me^{-\frac{m\epsilon^2\gamma^4}{2\mathbf{C}_K^2\mathbf{M}^2(\gamma+\mathbf{C}_K)^2}}.$$

PROOF. From Lemmas 3 and 4 it follows that, with a probability at least

$$1 - 2\left(e^{-\frac{m\epsilon^2\gamma^2}{2(\mathbf{C}_K M_\rho)^2}} + e^{-\frac{m\epsilon^2\gamma^2}{2\mathbf{C}_K^2(\|f_\rho\|_\infty+\sqrt{\mathbf{C}_K}R_\gamma)^2}}\right)$$

for every $t \in X$,

$$\left|f_\gamma(t) + \frac{1}{m\gamma}\sum_{i=1}^{m} K(x_i,t)(f_\gamma(x_i)-y_i)\right| \le 2\epsilon.$$

Note that, since

$$\max\{M_\rho, \|f_\rho\|_\infty + \sqrt{\mathbf{C}_K} R_\gamma\} \leq \mathbf{M} + \sqrt{\mathbf{C}_K} r_\gamma = \frac{\mathbf{M}(\gamma + \mathbf{C}_K)}{\gamma}$$

the confidence above is at least

$$1 - 4e^{-\frac{m\epsilon^2\gamma^4}{2\mathbf{C}_K^2\mathbf{M}^2(\gamma+\mathbf{C}_K)^2}}.$$

Applying this to $t = x_1, \ldots, x_m$ and writing the m resulting inequalities in matrix form we obtain that, with confidence at least the one in the statement,

$$\left\| f_\gamma[\mathbf{x}] + \frac{1}{m\gamma} K[\mathbf{x}] f_\gamma[\mathbf{x}] - \frac{1}{m\gamma} K[\mathbf{x}][\mathbf{y}] \right\|_{\max} \leq 2\epsilon.$$

□

Lemma 6 *For all $\gamma, \epsilon > 0$,*

$$\left(\mathrm{Id} + \frac{K[\mathbf{x}]}{\gamma m}\right) f_{\gamma,\mathbf{z}}(x) = \frac{K[\mathbf{x}]\mathbf{y}}{\gamma m}.$$

PROOF. In Proposition 8, Chapter III of [CS] it is shown that

$$\begin{aligned} & f_{\gamma,\mathbf{z}}(t) = \sum_{i=1}^{m} \frac{y_i - f_{\gamma,\mathbf{z}}(x_i)}{\gamma m} K(x_i, t) \\ \Rightarrow \quad & \gamma m f_{\gamma,\mathbf{z}}(t) = \sum_{i=1}^{m} (y_i - f_{\gamma,\mathbf{z}}(x_i)) K(x_i, t) \\ \Rightarrow \quad & \gamma m f_{\gamma,\mathbf{z}}(t) + \sum_{i=1}^{m} f_{\gamma,\mathbf{z}}(x_i) K(x_i, t) = \sum_{i=1}^{m} y_i K(x_i, t). \end{aligned}$$

Applying this equality for $t = x_1, \ldots, x_m$ and writing the resulting m equalities in matrix form we obtain

$$\gamma m f_{\gamma,\mathbf{z}}[\mathbf{x}] + f_{\gamma,\mathbf{z}}[\mathbf{x}] K[\mathbf{x}] = K[\mathbf{x}]\mathbf{y}$$

from which the statement follows. □

PROOF OF PROPOSITION 2. From Lemmas 5 and 6 it follows that

$$\left\| \left(\mathrm{Id} + \frac{K[\mathbf{x}]}{\gamma m}\right) f_\gamma[\mathbf{x}] - \left(\mathrm{Id} + \frac{K[\mathbf{x}]}{\gamma m}\right) f_{\gamma,\mathbf{z}}[\mathbf{x}] \right\|_{\max} \leq 2\epsilon$$

i.e.

$$\left\| \left(\mathrm{Id} + \frac{K[\mathbf{x}]}{\gamma m}\right) (f_\gamma[\mathbf{x}] - f_{\gamma,\mathbf{z}}[\mathbf{x}]) \right\|_{\max} \leq 2\epsilon$$

with the stated confidence. The result now follows since $\frac{K[\mathbf{x}]}{\gamma m}$ is positive definite and therefore $\| \left(\mathrm{Id} + \frac{K[\mathbf{x}]}{\gamma m}\right)^{-1} \| \geq 1$. □

2.4 Estimating the Sample Error

The expression $|\mathcal{E}(f_\gamma) - \mathcal{E}(f_{\gamma,\mathbf{z}})|$ is called the *sample error* (of $f_{\gamma,\mathbf{z}}$). In the previous section we estimated the confidence of obtaining a small sample error when the sample size m and an error bound ϵ are given. In this section we will fix a confidence $1-\delta$ and a sample size m and obtain bounds for the sample error.

Lemma 7 *Let $c_1, c_2 > 0$ and $s > q > 0$. Then the equation*

$$x^s - c_1 x^q - c_2 = 0$$

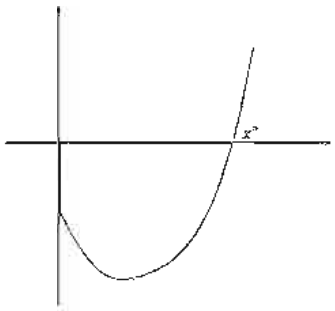
has a unique positive zero x^*. In addition

$$x^* \leq \max\left\{(2c_1)^{\frac{1}{s-q}}, (2c_2)^{\frac{1}{s}}\right\}.$$

PROOF. Let $\varphi(x) = x^s - c_1 x^q - c_2$. Then, taking the derivative with respect to x, $\varphi'(x) = sx^{s-1} - qc_1 x^{q-1} = x^{q-1}(sx^{s-q} - qc_1)$. Thus

$$\varphi'(x) = 0 \Leftrightarrow x^{s-q} = \frac{qc_1}{s}$$

and this derivative has a unique positive zero. The first statement follows since $\varphi(0) < 0$, $\varphi'(0^+) \leq 0$ and $\varphi(x) \to +\infty$ when $x \to +\infty$.



The second statement is a well-known bound (see [10, Theorem 4.2 (iv)]). □

Remark 2 *Note that, given c_1, c_2, s and t one can efficiently compute (a good approximation of) x^* using algorithms such as Newton's method.*

By Theorem 2, the sample error ϵ satisfies, with confidence $1-\delta$,

$$4m\mathcal{N}\left(\mathcal{H}(r_\gamma), \frac{\epsilon\gamma}{32\mathrm{M}(\gamma + \mathrm{C}_K)}\right) e^{-\frac{m\epsilon^2\gamma^4}{128\mathrm{M}^4(\gamma+\mathrm{C}_K)^4}} \geq \delta$$

i.e.

$$\frac{m\epsilon^2\gamma^4}{128\mathrm{M}^4(\gamma + \mathrm{C}_K)^4} - \ln\left(\frac{4m}{\delta}\right) - \ln\mathcal{N}\left(\mathcal{H}(r_\gamma), \frac{\epsilon\gamma}{32\mathrm{M}(\gamma + \mathrm{C}_K)}\right) \leq 0. \tag{2.1}$$

Now we recall (cf. Section 6 in Chapter I of [CS]) that, for every $t < 2$ there exists a constant C_t independent of ϵ and γ, such that

$$\ln\mathcal{N}\left(\mathcal{H}(r_\gamma), \frac{\epsilon\gamma}{32\mathrm{M}(\gamma+\mathrm{C}_K)}\right) \leq \left(\frac{r_\gamma C_t 32\mathrm{M}(\gamma+\mathrm{C}_K)}{\epsilon\gamma}\right)^t \leq \left(\frac{32C_t\mathrm{M}^2(\gamma+\mathrm{C}_K)^2}{\epsilon\gamma^2}\right)^t$$

(a different bound appears in [15]). Note, in the last inequality we replaced r_γ by its definition and used that $\sqrt{\mathrm{C}_K} \leq (\mathrm{C}_K+\gamma)$. Using this bound for the covering number, inequality (2.1) becomes

$$\frac{m\epsilon^2\gamma^4}{128\mathrm{M}^4(\gamma+\mathrm{C}_K)^4} - \ln\left(\frac{4m}{\delta}\right) - \left(\frac{32C_t\mathrm{M}^2(\gamma+\mathrm{C}_K)^2}{\epsilon\gamma^2}\right)^t \leq 0.$$

Write

$$v = \frac{\epsilon\gamma^2}{32\mathrm{M}^2(\gamma+\mathrm{C}_K)^2}.$$

Then the inequality above takes the form

$$c_0 v^2 - c_1 - c_2 v^{-t} \leq 0 \tag{2.2}$$

where $c_0 = \frac{m}{4}$, $c_1 = \ln\left(\frac{4m}{\delta}\right)$, $c_2 = C_t^t$, the tth power of C_t. Note that one could fix, for example, $t = 1$.

Now take the equality in (2.2) to obtain the equation

$$\varphi(v) = v^{t+2} - \frac{c_1}{c_0}v^t - \frac{c_2}{c_0} = 0$$

and note that this equation has only one positive zero by Lemma 7. Let $v^*(m,\delta)$ be this solution. Then, also by Lemma 7,

$$v^*(m,\delta) \leq \left\{\left(\frac{8\ln(4m)-\ln(\delta)}{m}\right)^{1/2}, \left(\frac{8C_t^t}{m}\right)^{t+2}\right\} \tag{2.3}$$

and

$$\epsilon = \frac{32\mathrm{M}^2(\gamma+\mathrm{C}_K)^2}{\gamma^2}v^*(m,\delta)$$

and we can conclude stating the following result.

Theorem 3 *Given $m \geq 1$ and $0 < \delta \leq 1$, for all $\gamma > 0$, the expression*

$$\mathscr{S}(\gamma) = \frac{32\mathrm{M}^2(\gamma+\mathrm{C}_K)^2}{\gamma^2}v^*(m,\delta)$$

bounds the sample error with confidence at least $1-\delta$. □

2.5 Choosing the optimal γ

We now focus on the approximation error $\mathcal{E}(f_\gamma)$. To do so we first apply part (1) of Theorem 3, Chapter II in [CS] with $H = \mathcal{L}^2_\rho(X)$, $s = 1$, $A = L_K^{1/2}$, and $a = f_\rho$, and use that $\|L_K^{-1/2} f\| = \|f\|_K$ to obtain that, for $0 < \theta < 1$ (e.g. for $\theta = 1/2$),

$$\min_{f \in \mathcal{L}^2_\rho(X)} (\|f - f_\rho\|^2 + \gamma \|f\|_K^2) \leq \gamma^\theta \|L_K^{-\theta/2} f_\rho\|^2.$$

Since the minimum above is attained at f_γ we deduce

$$\|f_\gamma - f_\rho\|^2 + \gamma \|f_\gamma\|_K^2 \leq \gamma^\theta \|L_K^{-\theta/2} f_\rho\|^2.$$

A basic result in [CS] (Proposition 1, Chapter I) states that, for all $f \in \mathcal{L}^2_\rho(X)$,

$$\mathcal{E}(f) = \int_X (f - f_\rho)^2 + \sigma_\rho^2 \tag{2.4}$$

where σ_ρ^2 is a non-negative quantity depending only on ρ. Therefore the approximation error $\mathcal{E}(f_\gamma)$ is bounded by $\mathscr{A}(\gamma) + \sigma_\rho^2$ where

$$\mathscr{A}(\gamma) = \gamma^\theta \|L_K^{-\theta/2} f_\rho\|^2.$$

Proof of the Main Result. Let

$$E(\gamma) = \mathscr{A}(\gamma) + \mathscr{S}(\gamma).$$

Recall

$$\mathcal{E}(f_{\gamma,\mathbf{z}}) \leq |\mathcal{E}(f_\gamma) - \mathcal{E}(f_{\gamma,\mathbf{z}})| + \mathcal{E}(f_\gamma).$$

Then the error $\mathcal{E}(f_{\gamma,\mathbf{z}})$ satisfies the bound

$$\mathcal{E}(f_{\gamma,\mathbf{z}}) \leq E(\gamma) + \sigma_\rho^2$$

and therefore, subtracting σ_ρ^2 from both sides and using (2.4) for $f = f_{\gamma,\mathbf{z}}$,

$$\int_X (f_{\gamma,\mathbf{z}} - f_\rho)^2 \leq E(\gamma).$$

This proves the first part of the Main Result. Note that this is actually a family of bounds parameterized by $t < 2$ and $0 < \theta < 1$ and depends on m, δ, K and f_ρ.

For a point $\gamma > 0$ to be a minimum of $E(\gamma) = \mathscr{S}(\gamma) + \mathscr{A}(\gamma)$ it is necessary that $\mathscr{S}'(\gamma) + \mathscr{A}'(\gamma) = 0$. Taking derivatives, we get

$$\mathscr{S}'(\gamma) = -64\mathbf{M}^2 v^*(m,\delta)\mathbf{C}_K \frac{\gamma + \mathbf{C}_K}{\gamma^3}$$

and

$$\mathscr{A}'(\gamma) = \theta \gamma^{\theta-1} \|L_K^{-\theta/2} f_\rho\|^2.$$

Thus the solutions of $\mathscr{A}'(\gamma) + \mathscr{S}'(\gamma) = 0$ are those of

$$\theta\gamma^{\theta+2}\|L_K^{-\theta/2} f_\rho\|^2 - 64\mathrm{M}^2 v^*(m,\delta)\mathrm{C}_K(\gamma + \mathrm{C}_K) = 0$$

i.e. those of

$$\gamma^{\theta+2} - \frac{64\mathrm{M}^2 v^*(m,\delta)\mathrm{C}_K}{\theta\|L_K^{-\theta/2} f_\rho\|^2}\gamma - \frac{64\mathrm{M}^2 v^*(m,\delta)\mathrm{C}_K^2}{\theta\|L_K^{-\theta/2} f_\rho\|^2} = 0. \tag{2.5}$$

Using again Lemma 7, we obtain a unique solution γ^* which is a minimizer of E since $E(\gamma) \to \infty$ as $\gamma \to 0$ or $\gamma \to \infty$. This finishes the proof of the Main Result. □

Corollary 2 *For every* $0 < \delta \leq 1$,

$$\lim_{m\to\infty} E(\gamma^*) = 0.$$

PROOF. The bound (2.3) shows that $v^*(m,\delta) \to 0$ when $m \to \infty$. Now, equation (2.5) shows that γ^* is the only positive root of

$$\gamma^{\theta+2} - Qv^*(m,\delta)\gamma - Qv^*(m,\delta)\mathrm{C}_K = 0 \tag{2.6}$$

where $Q = \frac{64\mathrm{M}^2\mathrm{C}_K}{\theta\|L_K^{-\theta/2} f_\rho\|^2}$. Then, by Lemma 7,

$$\gamma^* \leq \max\left\{(2Qv^*(m,\delta))^{\frac{1}{\theta+1}}, (2Qv^*(m,\delta)\mathrm{C}_K)^{\frac{1}{\theta+2}}\right\}$$

from which it follows that $\gamma^* \to 0$ when $m \to \infty$. Note that this implies that

$$\lim_{m\to\infty} \mathscr{A}(\gamma^*) = \lim_{m\to\infty} (\gamma^*)^\theta \|L_K^{-\theta/2} f_\rho\|^2 = 0.$$

Finally, it follows from equation (2.6) that

$$(\gamma^*)^\theta - (Q\gamma^* + Q\mathrm{C}_K)\left[\frac{v^*(m,\delta)}{(\gamma^*)^2}\right] = 0$$

and therefore, that $\left[\frac{v^*(m,\delta)}{(\gamma^*)^2}\right] \to 0$ when $m \to \infty$. This, together with Theorem 3, shows that $\lim_{m\to\infty} \mathscr{S}(\gamma^*) = 0$. □

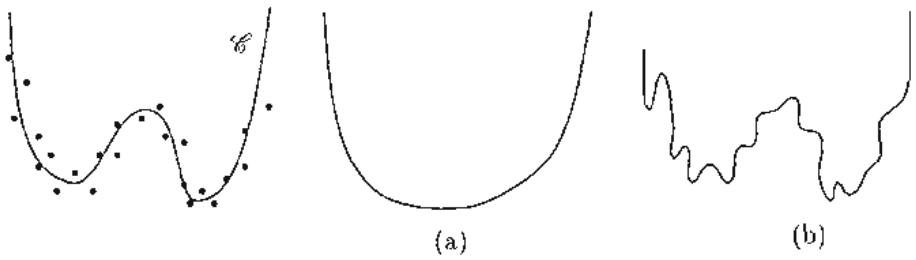
2.6 Final Remarks

(1) This report can be seen as a solution of one instance of the *Bias-Variance problem*. Roughly speaking, the "bias" of a solution f coincides with our approximation error, and its "variance" with the sample error. Quoting [3],

> A model which is too simple, or too inflexible, will have a large bias, while one which has too much flexibility in relation to the particular data set will have a large variance. Bias and variance are complementary quantities, and the best generalization [i.e. the smallest error] is obtained when we have the best compromise between the conflicting requirements of small bias and small variance.

As described in Section 3, Chapter II in [CS], the bias-variance problem amounts to the choice of a compact subspace $\mathcal{H}$ of $\mathscr{C}(X)$ over which $\mathcal{E}_{\mathbf{z}}$ is minimized. A too small space $\mathcal{H}$ will yield a large bias while one too large will yield a large variance. Several parameters (radius of balls, dimension, etc.) determine the "size" of $\mathcal{H}$ and different instances of the bias-variance problem are obtained by fixing all of them except one and minimizing the error over this non-fixed parameter. Our solution considers the ball of radius $r = \|f_{\gamma,\mathbf{z}}\|_K$ in $\mathcal{H}_K$ and $\mathcal{H} = \overline{I_K(B_r)}$ (a space over which $f_{\gamma,\mathbf{z}}$ minimizes $\mathcal{E}_{\mathbf{z}}$). The number r is our replacement of the VC-dimension. Since γ is inversely proportional to r, large γ corresponds to large bias or approximation error and small γ to large variance or sample error.

Failing to find a good compromise between bias and variance leads to what is called *underfitting* (large bias) or *overfitting* (large variance). As an example, consider the curve $\mathscr{C}$ in the figure below with the set of sample points and assume we want to approximate that curve with a polynomial of degree d (the parameter d determines in our case the dimension of $\mathcal{H}$). If d is too small, say $d = 2$, we obtain a curve as in (a) in the figure, which necessarily "underfits" the data points. If d is too large, we can tightly fit the data points but this "overfitting" yields a curve as in (b).



For more on the bias-variance problem see [3], the above mentioned section in [CS], [6], and the references in these papers. Note, however, that the literature on this problem is vast and we have only touched on it.

(2) One could interpret the main estimates in this paper in terms of algorithms for approximating solutions of integral equations by Monte Carlo methods. But for most algorithms in the theory of integral equations the points x_i, $i = 1, \ldots, m$, are not randomly chosen but taken for example as a set of lattice points of a domain $X \subset \mathbb{R}^n$ (this would correspond to *active learning* in the learning theory literature). Now one might take ρ_X as Lebesgue measure and the x_i from a uniform grid of points. The theory in our previous paper [CS] should permit modifications to deal with this situation and our main result here as well.

(3) Our work can be interpreted in the area of statistics known as regularized nonparametric least squares regression. A general reference for this area is the book by Sara van de Geer [14]. Besides the references in this book, the papers [2, 1, 8, 9] are also related to our work. A result somewhat similar in spirit to our main result appears in [11, 12]. Here a function $E(m, n)$ is exhibited bounding the error in terms of the number m of examples and the number n of basis functions in a

space of Gaussian radial basis functions and it is shown that, for each $m \in \mathbb{N}$, $E(m,n)$ has a unique minimizer n^*.

Addenda. Corrections to [CS].

(1) A regularity hypothesis on measure ρ_X on X requiring every open set on X to have positive measure is needed for our extension of Mercer Theorem and its applications. This is a mild hypothesis since open sets with zero measure can be deleted from X with no harm.

(2) In connection with the matrices associated to a Mercer kernel, the "positive definite" condition should be relaxed to "positive semidefinite."

Bibliography

[1] A.R. Barron, Approximation and estimation bounds for artificial neural networks, *Machine Learning* 14 (1994) 115–133.

[2] A.R. Barron, L. Birgé, and P. Massart, Risk bounds for model selection via penalisation, *Probability Theory and Related Fields* **113** (1995) 301–403.

[3] C.M. Bishop, *Neural Networks for Pattern Recognition*, Cambridge University Press (1995).

[4] F. Cucker and S. Smale, On the mathematical foundations of learning, *Bulletin Amer. Math. Soc.* **39** (2002) 1–49.

[5] T. Evgeniou, M. Pontil, and T. Poggio, Regularization Networks and Support Vector Machines, *Advances in Computational Mathematics* **13** (2000) 1–50.

[6] G. Golub, M. Heat, and G. Wahba, Generalized cross-validation as a method for choosing a good ridge parameter, *Technometrics* **21** (1979) 215–223.

[7] V.V. Ivanov, *The Theory of Approximate Methods and Their Application to the Numerical Solution of Singular Integral Equations*, Nordhoff International (1976).

[8] O.V. Lepskii, Asymptotically minimax adaptive estimation. I: Upper bounds, optimally adaptive estimates, *Theory Probab. Appl.* **36** (1991) 682–697.

[9] O.V. Lepskii, Asymptotically minimax adaptive estimation. II: Schemes without optimal adaption, adaptive estimators, *Theory Probab. Appl.* **37** (1992) 433–448.

[10] M. Mignotte, *Mathematics for Computer Algebra*, Springer-Verlag (1992).

[11] P. Niyogi and F. Girosi, On the relationship between generalization error, hypothesis complexity and sample complexity for radial basis functions, *Neural Computation* 8 (1996) 819–842.

[12] P. Niyogi and F. Girosi, Generalization bounds for function approximation from scattered noisy data, *Advances in Computational Mathematics* **50** (1999) 51–80.

[13] A.N. Tikhonov and V.Y. Arsenin, *Solutions of Ill-posed Problems*, W.H. Winston (1977).

[14] S. van de Geer, *Empirical Processes in M-Estimation*, Cambridge University Press (2000).

[15] D.-X. Zhou, The covering numbers in learning theory, To appear in *J. of Complexity*, (2001).

Chapter 3

Cucker Smale Learning Theory in Besov Spaces

Charles A. Micchelli[1], Yuesheng Xu[1,2] and Peixin Ye

Dedicated to Steve Smale with friendship and esteem

Abstract. Let $\mathbb{B} := (\mathbb{B}, \|\cdot\|_b)$ be a Banach space and $\mathbb{H} := (\mathbb{H}, \|\cdot\|_h)$ a dense subspace of $\mathbb{B}$. We are interested in the rate at which the distance of a given $x \in \mathbb{B}$ from the ball of radius t in $\mathbb{H}$ tends to zero as $t \to \infty$. This problem has numerous applications and its recent importance in the Cucker Smale theory of learning renewed our interest in this subject. We study two aspects of this problem. First, we obtain general results on the relation between the CS-functional and the Peetre K-functional. Secondly, we estimate it in the concrete case of Besov spaces by using functions of exponential type.

[1]Supported in part by the National Science Foundation under grant DMS-9973427

[2]Supported in part by the Chinese Academy of Sciences under program "Hundred Distinguished Young Scientists".

3.1 Introduction

Let $\mathbb{B} := (\mathbb{B}, \|\cdot\|_b)$ be a Banach space and $\mathbb{H} := (\mathbb{H}, \|\cdot\|_h)$ be a dense subspace of $\mathbb{B}$ with $\|x\|_b \le \|x\|_h$ for $x \in \mathbb{H}$. Given $x \in \mathbb{B}$ and $t > 0$, Cucker and Smale demonstrated in [5] the importance in *learning theory* of the rate at which the function

$$I(x,t) := \inf\{\|x-y\|_b : \|y\|_h \le t\}$$

tends to zero as $t \to \infty$. They referred to I as the *approximation error* with *regression function* x and *hypothesis space*

$$\mathbb{H}_t := \{y \in \mathbb{H} : \|y\|_h \le t\}.$$

So far, in learning theory this problem was studied, almost exclusively, when $\mathbb{H}$ is a Hilbert space, although, as pointed out in [11] it arises in a wide variety of problems in applied mathematics including optimal filter designs, control theory, approximation of operators and regularization, see [1], [17] for material exclusive to Tikhonov regularization and for Korneichuk and Kudryavtsev's study of best approximation between two Sobolev classes [8], [9]. In this chapter, we use results in [11] to give sharp error estimates for the CS-functional in terms of the Peetre K-functional. This is done in Section 3.2. In Section 3.3 we estimate the CS-functional for anisotropic Besov spaces containing hypothesis spaces which are the range of a convolution kernel by using results from Section 3.2 on entire functions of exponential type, thereby improving results in [15].

3.2 Cucker Smale Functional and the Peetre K-Functional

Let $\mathbb{X}$ be a linear space equipped with two norms $\|\cdot\|_i$, $i = 1, 2$. In this section we give exact bounds for the CS-functional

$$I(x,t) := \inf\{\|x-y\|_1 : \|y\|_2 \le t, y \in \mathbb{X}\}$$

in terms of the Peetre K-functional

$$K(x,t) := \inf\{\|x-y\|_1 + t\|y\|_2 : y \in \mathbb{X}\}$$

which plays a pivotal role in the theory of interpolation spaces, [2]. To this end, we find the following quantities useful. For every $\theta \in I^0 := (0,1)$ and $x \in \mathbb{X}$ we define

$$\|x\|_\theta := \sup\left\{K(x,t)t^{-\theta} : t \in \mathbb{R}_+\right\} \tag{3.1}$$

and

$$|x|_\theta := \sup\left\{I(x,t)^{1-\theta}t^{\theta} : t \in \mathbb{R}_+\right\} \tag{3.2}$$

where $\mathbb{R}_+ := \{t \in \mathbb{R} : t > 0\}$.

Theorem 1 *For every $x \in \mathbb{X}$ and $\theta \in I^0$*

$$\frac{|x|_\theta}{\|x\|_\theta} = h(\theta)$$

where

$$h(\theta) := \theta^\theta (1-\theta)^{1-\theta}, \quad \theta \in I^0.$$

PROOF. Using the first part of Proposition 2.3, page 49-50 of [11] we have that

$$\begin{aligned} K(x;t) &= \inf\{I(x;\sigma) + \sigma t : \sigma \in \mathbb{R}_+\} \\ &\leq \inf\{|x|_\theta^{(1-\theta)^{-1}} \sigma^{-\theta(1-\theta)^{-1}} + \sigma t : \sigma \in \mathbb{R}_+\} \\ &= |x|_\theta t^\theta \inf\{\tau + \tau^{-\theta(1-\theta)^{-1}} : \tau \in \mathbb{R}_+\} \\ &= \frac{|x|_\theta t^\theta}{h(\theta)}. \end{aligned}$$

This observation proves that

$$\|x\|_\theta \leq |x|_\theta h^{-1}(\theta). \tag{3.3}$$

Next, we use the second part of Proposition 2.3, page 49-50 of [11], which gives

$$I(x;t) = \sup\left\{\frac{\sigma K(x;\sigma^{-1}) - t}{\sigma} : \sigma \in \mathbb{R}_+\right\}$$

and so we conclude that

$$\begin{aligned} I(x;t) &\leq \sup\{\sigma^\theta \|x\|_\theta - t\sigma : \sigma \in \mathbb{R}_+\} \\ &= \|x\|_\theta^{(1-\theta)^{-1}} t^{-\theta(1-\theta)^{-1}} \sup\{\tau^\theta - \tau : \tau \in \mathbb{R}_+\} \\ &= h(\theta)^{(1-\theta)^{-1}} \|x\|_\theta^{(1-\theta)^{-1}} t^{-\theta(1-\theta)^{-1}}. \end{aligned}$$

In other words, we have established that

$$|x|_\theta \leq h(\theta)\|x\|_\theta$$

and with (3.3) the theorem. □

Since $h(I^0) = (\frac{1}{2}, 1)$, it follows from Theorem 1 for every $x \in \mathbb{X}$ and $\theta \in I^0$ that

$$\frac{1}{2} \leq \frac{|x|_\theta}{\|x\|_\theta} \leq 1. \tag{3.4}$$

This inequality is prominent in [15]. Note that Theorem 1 establishes that the ratio of the two quantities (3.1) and (3.2) appearing in (3.4) is *independent* of $x \in \mathbb{X}$.

The basis of the proof of Proposition 2.3 of [11] (and hence also Theorem 1 above) to pass from K to I and back uses the *Gagliardo diagram*, see, Chapters 3 and 7 of [2]. The use of Proposition 2.3 of [11] should also allow for comparisons of other decay rates for K and I, other than those considered in Theorem 1.

With this result, standard facts from approximation theory of interpolation spaces can be translated into estimates for the CS-functional. For example, identification of Besov spaces as an intermediate space leads to the following theorem. We use the notation $Q := [1, \infty]$.

Theorem 2 *If $r, s \in \mathbb{R}_+$, $p, q \in Q$ with $r < s$ and Ω is an open subset of $\mathbb{R}^d$ with minimally smooth boundary then there exists a constant $c \in \mathbb{R}_+$ such that for all $f \in B^r_{p,\infty}(\Omega)$*

$$\inf\{\|f - g\|_{L_p(\Omega)} : \|g\|_{\mathbb{H}} \le t\} \le c\|f\|_{B^r_{p,\infty}(\Omega)}(r/s)^{\frac{r}{s-r}}(1 - r/s)t^{-\frac{r}{s-r}}$$

where $\mathbb{H} = B^s_{p,q}(\Omega)$.

PROOF. The result follows by recalling from [2] that

$$(L_p(\Omega), B^s_{p,q}(\Omega))_{r/s,\infty} = B^r_{p,\infty}(\Omega).$$

Specifically, for $f \in B^r_{p,\infty}(\Omega)$ we obtain from Theorem 1 that

$$\begin{aligned} \inf\{\|f - g\|_{L_p(\Omega)} : \|g\|_{\mathbb{H}} \le t\} &\le \|f\|_{r/s}(r/s)^{\frac{r}{s-r}}(1 - r/s)t^{-\frac{r}{s-r}} \\ &\le c\|f\|_{B^r_{p,\infty}(\Omega)}(r/s)^{\frac{r}{s-r}}(1 - r/s)t^{-\frac{r}{s-r}}, \end{aligned}$$

which proves the result. □

Let us now turn to another special case of the above result. We restrict ourselves to a Hilbert space $(\mathbb{H}, \|\cdot\|)$ and a bounded symmetric nonnegative operator $\mathcal{A}$ on $\mathbb{H}$. For any positive numbers $\sigma, s \in \mathbb{R}_+$ with $\sigma < s$ we consider for $x \in \mathbb{H}$ the quantity

$$I_{s\sigma}(x; t) := \inf\{\|\mathcal{A}^\sigma x - \mathcal{A}^s y\| : \|y\| \le t\}$$

which was considered in [5], [15] where the estimate

$$\sup\left\{I_{s\sigma}(x; t)^{1-\theta}t^\theta : t \in \mathbb{R}_+\right\} \le \|x\| \tag{3.5}$$

for $\theta := \frac{\sigma}{s}$ and $x \in \mathbb{H}$ was derived by means of *Tikhonov regularization*. We now show another duality principle extracted from [11] whose roots lie in *optimal estimation* can be used to improve upon the estimate (3.5).

Theorem 3 *If $\theta = \frac{\sigma}{s}$ where $\sigma < s$, $\sigma, s \in \mathbb{R}_+$ and $x \in \mathbb{H}$, a Hilbert space, then*

$$\sup\left\{I_{s\sigma}(x, t)^{1-\sigma}t^\theta : t \in \mathbb{R}_+\right\} \le h(\theta)\|x\|.$$

PROOF. We consider the quantity

$$K_{s\sigma}(x, t) := \inf\{\|\mathcal{A}^\sigma x - \mathcal{A}^s y\| + t\|y\| : y \in \mathbb{H}\}$$

and estimate it from above by using Theorem 3.1 of [11]. To this end, we let $I := [0, 1]$, $(\cdot, \cdot)$ be the inner product on $\mathbb{H}$ and observe that

$$\begin{aligned} K_{s\sigma}(x, t) &= \sup\{(\mathcal{A}^\sigma x, y) : \|y\| \le 1, \|\mathcal{A}^s y\| \le t\} \\ &= \inf\left\{\sup\{(\mathcal{A}^\sigma x, y) : \rho\|y\|^2 + t^{-2}(1 - \rho)\|\mathcal{A}^s y\|^2 \le 1\} : \rho \in I\right\} \\ &= \inf\left\{\sup\{\mathcal{A}^\sigma x, (\rho I + (1 - \rho)t^{-2}\mathcal{A}^{-1/2})^{1/2}y) : \|y\| \le 1\} : \rho \in I\right\} \\ &= \inf\left\{\|\mathcal{A}^\sigma(\rho I + (1 - \rho)t^{-2}\mathcal{A}^{2s})^{-1/2}x\| : \rho \in I\right\} \\ &\le \inf\left\{\|\mathcal{A}^\sigma(\rho I + (1 - \rho)t^{-2}\mathcal{A}^{2s})^{-1/2}\| : \rho \in I\right\}\|x\|. \end{aligned}$$

Consequently, we have proved that

$$\begin{aligned}\frac{K_{s\sigma}(x;t)}{\|x\|} &\leq \inf\left\{\sup\left\{\frac{u^\sigma}{(\rho+(1-\rho)t^{-2}u^{2s})^{1/2}} : u\in\mathbb{R}_+\right\} : \rho\in I\right\}\\ &= \inf\left\{\frac{\rho^{\theta/2-1}}{(1-\rho)^{\theta/2}}t^\theta : \rho\in I\right\}\sup\left\{v^{\theta/2}(1+v)^{-1} : v\in\mathbb{R}_+\right\}\\ &= h(\theta/2)t^\theta\inf\left\{\rho+\rho^{\frac{\theta-2}{\theta}} : \rho\in\mathbb{R}_+\right\} = t^\theta.\end{aligned}$$

In other words, we have established that

$$K_{s\sigma}(x,t)\leq t^\theta\|x\|. \tag{3.6}$$

Combining this estimate with Theorem 1 yields the desired result. □

We remark that in [5] and [15] the operator $\mathcal{A}$ is assumed to be strictly positive, a condition which not required here. From Theorem 3 and the fact that $h(I^0)=(\frac{1}{2},1)$, inequality (3.5) follows directly. Note that the upper bound in (3.6) depends *only* on θ and x but *not* on $\mathcal{A}$. We have, of course, used in its derivation the hypothesis that the spectrum of $\mathcal{A}$ satisfies $\sigma(\mathcal{A})\subseteq\mathbb{R}_+$. Additional information about the spectrum of $\mathcal{A}$ would lead to shaper bounds.

We can bound $K_{s\sigma}$ in another way which still leads to an improvement on inequality (3.5). Indeed, discussions in [11] suggest that we choose $\lambda\in\mathbb{R}_+$ and set

$$y := (\lambda I+\mathcal{A}^{2s})^{-1}\mathcal{A}^{s+\sigma}x. \tag{3.7}$$

Therefore, we have that

$$\frac{K_{s\sigma}(x,t)}{\|x\|}\leq\inf\left\{\lambda\|\mathcal{A}^\sigma(\mathcal{A}^{2s}+\lambda I)^{-1}\|+t\|\mathcal{A}^{s+\sigma}(\mathcal{A}^{2s}+\lambda I)^{-1}\| : \lambda\in\mathbb{R}_+\right\}.$$

Each norm can be bounded from above, since $\sigma(\mathcal{A})\subseteq\mathbb{R}_+$, to obtain the estimate for the Peetre K-functional

$$\begin{aligned}\frac{K_{s\sigma}(x,t)}{\|x\|} &\leq \inf\left\{h\left(\frac{\theta}{2}\right)\lambda^{\theta/2}+t\lambda^{\theta/2-1/2}h\left(\frac{1+\theta}{2}\right) : \lambda\in\mathbb{R}_+\right\}\\ &= t^\theta h\left(\frac{1+\theta}{2}\right)^\theta h\left(\frac{\theta}{2}\right)^{1-\theta}\inf\left\{\rho+\rho^{-\frac{1-\theta}{\theta}} : \rho\in\mathbb{R}_+\right\}\\ &= \frac{t^\theta h\left(\frac{1+\theta}{2}\right)^\theta h\left(\frac{\theta}{2}\right)^{1-\theta}}{h(\theta)}\end{aligned}$$

and our estimate for the CS-functional is

$$\sup\left\{I_{s\sigma}(x,t)^{1-\theta}t^\theta : t\in\mathbb{R}_+\right\}\leq h\left(\frac{1+\theta}{2}\right)^\theta h\left(\frac{\theta}{2}\right)^{1-\theta}\|x\|.$$

The choice of the vector (3.7) was done with care. Indeed, the following is a standard fact which is worthwhile to include here.

Theorem 4 *Let $(\mathbb{H}, \|\cdot\|)$ be a Hilbert space and $\mathcal{A} : \mathbb{H} \to \mathbb{H}$ a bounded normal operator on $\mathbb{H}$ with dense range. Suppose $x \notin R(\mathcal{A}) :=$ range of $\mathcal{A}$ and*

$$I(x,t) := \inf\{\|x - \mathcal{A}y\| : \|y\| \le t\}.$$

There exists a unique vector $w \in \mathbb{H}$ which achieves this infimum and is given by

$$w = (\lambda I + \mathcal{A}^*\mathcal{A})^{-1}\mathcal{A}^*x$$

where $\lambda \in \mathbb{R}_+$ is chosen uniquely to satisfy the equation $\|w\| = t$.

PROOF. By Theorem 2.2 of [11], for each $x \in \mathbb{H}$, $I(x,\cdot)$ is a convex, *strictly* decreasing function on $\mathbb{R}_+$ with range $(0, \|x\|)$. By weak compactness of the unit ball in $\mathbb{H}$ the infimum is attained for any $t \in \mathbb{R}_+$. Call the element in $\mathbb{H}$ which achieves the minimum w. If $\|w\| < t$ then by a variational argument $x - \mathcal{A}w \perp R(\mathcal{A})$ and so $x \in R(\mathcal{A})$, a contradiction to our hypothesis about x. Thus we conclude that $\|w\| = t$. Therefore, by the method of Lagrange multipliers there is a $\lambda \in \mathbb{R}$ such that

$$\mathcal{A}^*(x - \mathcal{A}w) = \lambda w.$$

Since $x \notin R(\mathcal{A})$ it must be the case that $\lambda \neq 0$. By a direct variational argument we also have that $(x - \mathcal{A}w, \mathcal{A}w) \ge 0$. Combining this inequality with the above equation gives

$$\lambda t^2 = \lambda(w,w) = (x - \mathcal{A}w, \mathcal{A}w) \ge 0,$$

that is, we obtain that $\lambda \in \mathbb{R}_+$ and

$$w = (\lambda I + \mathcal{A}^*\mathcal{A})^{-1}\mathcal{A}x.$$

Consequently, we have that

$$\|w\|^2 = \int_S \frac{\rho}{(\lambda+\rho)^2} d\mu(\rho)$$

where $S := \sigma(\mathcal{A}^*\mathcal{A})$ and $d\mu$ is the spectral measure of $\mathcal{A}^*\mathcal{A}$ at x. The function of λ on the right hand side of the above equation is *strictly* increasing on $\mathbb{R}_+$ and so there is indeed a *unique* value of $\lambda \in \mathbb{R}_+$ such that $\|w\| = t$. This observation completes the proof of the result. □

3.3 Estimates for the CS-Functional in Anisotropic Besov Spaces

In this section we develop general error estimates for the CS-functional relative to *anisotropic Besov spaces*. Let us begin by recalling their definitions. We have a set $Z_d := \{1, 2, \ldots, d\}$, a vector $\mathbf{p} := (p_j : j \in Z_d) \in Q^d$ and a subinterval J of $\mathbb{R}$. With these at hand, we denote by $L_{\mathbf{p}}(J^d)$ the Banach space of measurable functions f on J^d with the norm

$$\|f\|_{L_{\mathbf{p}}(J^d)} := \left(\int_J \left(\int_J \cdots \left(\int_J |f(\mathbf{t})|^{p_1} dt_1 \right)^{p_2/p_1} \cdots dt_{d-1} \right)^{p_d/p_{d-1}} dt_d \right)^{1/p_d}.$$

When $\mathbf{p} = p\mathbf{e}$ where $\mathbf{e} := (1, 1, \ldots, 1) \in \mathbb{Z}^d$ and p is a scalar in Q, the space $L_{\mathbf{p}}(J^d)$ coincides with the usual L_p space on J^d.

For the choice $J := \mathbb{R}$, we denote this norm by simply writing $\|\cdot\|_{\mathbf{p}}$. For $f \in L_{\mathbf{p}}(\mathbb{R}^d)$, we define the ℓ-th partial difference of f in the k-th coordinate direction $\mathbf{e}_k$ at the point $\mathbf{x} \in \mathbb{R}^d$ with step $t_k \in \mathbb{R}$ by the formula

$$\Delta_{t_k}^{\ell} f(\mathbf{x}) = \sum_{j \in Z_{\ell+1}} (-1)^{j-1+\ell} \binom{\ell}{j-1} f(\mathbf{x} + (j-1)t_k \mathbf{e}_k). \tag{3.8}$$

We choose $\mathbf{k} := (k_j : j \in Z_d) \in \mathbb{Z}_+^d$, $\mathbf{r} := (r_j : j \in Z_d) \in \mathbb{R}_+^d$ satisfying $0 < \mathbf{r} < \mathbf{k}$, $\theta \in V := Q \setminus \{\infty\}$ and define the semi-norms

$$|f|_{B^{r_j}_{\mathbf{p},\theta,j}} := \left\{ \int_{\mathbb{R}} \left(\frac{\|\Delta_{t_j}^{k_j} f\|_{\mathbf{p}}}{|t_j|^{r_j}} \right)^{\theta} \frac{dt_j}{|t_j|} \right\}^{1/\theta}$$

and

$$|f|_{B^{r_j}_{\mathbf{p},\infty,j}} := \sup \left\{ \frac{\|\Delta_{t_j}^{k_j} f\|_{\mathbf{p}}}{|t_j|^{r_j}} : t_j \neq 0 \right\}$$

where we set $\mathbb{Z}_+ := \mathbb{R}_+ \cap \mathbb{Z}$. By [7] and [8], this function space is a Banach space with the norm given by

$$\|f\|_{B^{\mathbf{r}}_{\mathbf{p},\theta}(\mathbb{R}^d)} := \|f\|_{\mathbf{p}} + \sum_{j \in Z_d} |f|_{B^{r_j}_{\mathbf{p},\theta,j}}.$$

(Equivalent spaces result from any choice of $\mathbf{k} > \mathbf{r}$.) Alternatively, these anisotropic Besov spaces can be realized as interpolation spaces between $L_{\mathbf{p}}(\mathbb{R}^d)$ and anisotropic Sobolev spaces. When $\mathbf{r} = r\mathbf{e}$, $r \in \mathbb{R}_+$, we obtain the standard isotropic Besov spaces.

We now return to estimating the CS-functional. A kernel $K \in L_1(\mathbb{R}^d)$ determines a linear operator $\mathcal{K}$ defined for $\mathbf{x} \in \mathbb{R}^d$ and $f \in L_{\mathbf{p}}(\mathbb{R}^d)$ by the equation

$$(\mathcal{K}f)(\mathbf{x}) := (K * f)(\mathbf{x}) = \int_{I^d} K(\mathbf{x} - \mathbf{t}) f(\mathbf{t}) d\mathbf{t},$$

where $I := [0, 1]$. The CS-functional which interests us is defined for $t \in \mathbb{R}_+$ and $f \in L_{\mathbf{p}}(\mathbb{R}^d)$ by the equation

$$I_{\mathbf{p}}(f, t) := \inf\{\|f - \mathcal{K}g\|_{L_{\mathbf{p}}(I^d)} : \|g\|_{L_p(I^d)} \leq t\}.$$

We shall study this quantity only under the hypothesis that the kernel K is *positive definite*. Recall that this mean that for any finite set of points $\{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^N\} \subseteq \mathbb{R}^d$ the symmetric matrix

$$(K(\mathbf{x}^i - \mathbf{x}^j))_{i,j \in Z_N}$$

is positive semi-definite. The celebrated theorem of Bochner [4] provides a complete characterization of such functions in the form

$$K(\mathbf{x}) := \int_{\mathbb{R}^d} e^{i\mathbf{x}\cdot\mathbf{y}} d\mu(\mathbf{y}), \quad \mathbf{x} \in \mathbb{R}^d \tag{3.9}$$

where $d\mu$ is a Borel measure on $\mathbb{R}^d$ such that

$$\int_{\mathbb{R}^d} d\mu(\mathbf{x}) < \infty.$$

Let us recall a result from [10] which ensures that CS-functional above tends to zero as $t \to \infty$ for *nearly* all such kernels. We need a condition on the measure which appears in the Bochner representation (3.9) of K. Indeed, if $d\mu$ were a discrete measure with only finite number of atoms then in general the CS-functional would not tend to zero. Our sufficient condition on the support of $d\mu$ uses the notion of the *uniqueness* set for entire functions. We say that $D \subseteq \mathbb{R}^d$ is a uniqueness set if any entire function F which vanishes on D is identically zero on $\mathbb{C}^d$. The following theorem from [10] is sufficient to establish that the CS-functional goes to zero when the support of the measure $d\mu$ in the representation (3.8) is a uniqueness set. We include the proof for the convenience of the reader.

Theorem 5 *If K is positive definite where the support of $d\mu$ is a uniqueness set and X is a compact set in $\mathbb{R}^d$ then the two linear spaces*

$$M_1 := \text{span } \{K(\cdot - \mathbf{y}) : \mathbf{y} \in X\} \tag{3.10}$$

and

$$M_2 := \{\mathcal{K}g : g \in C(X)\} \tag{3.11}$$

are dense in $C(X)$.

PROOF. We first prove that space M_1 defined by (3.10) is dense in $C(X)$. By the Hahn-Banach duality, it suffices to prove that whenever $d\nu$ is a *sign* measure on X such that for all $\mathbf{x} \in X$

$$\int_X K(\mathbf{x} - \mathbf{y}) d\nu(\mathbf{y}) = 0, \tag{3.12}$$

it follows that $d\nu = 0$. To this end, we note that

$$\int_X \int_X K(\mathbf{x} - \mathbf{y}) d\nu(\mathbf{y}) d\nu(\mathbf{x}) = 0. \tag{3.13}$$

Substituting representation (3.9) for K into equation (3.13) yields the equation

$$\int_X \int_X \int_{\mathbb{R}^d} e^{-i(\mathbf{x}-\mathbf{y})\cdot \mathbf{t}} d\mu(\mathbf{t}) d\nu(\mathbf{y}) d\nu(\mathbf{x}) = 0,$$

which is equivalent to the condition that

$$\int_{\mathbb{R}^d} \left| \int_X e^{-i\mathbf{x}\cdot\mathbf{y}} d\nu(\mathbf{x}) \right|^2 d\mu(\mathbf{y}) = 0. \tag{3.14}$$

Now, we define an entire function

$$F(\mathbf{y}) := \int_X e^{-i\mathbf{x}\cdot\mathbf{y}} d\nu(\mathbf{x}), \quad \mathbf{y} \in \mathbb{C}^d,$$

and conclude from (3.14) that F vanishes on the support of $d\mu$. Since the support of $d\mu$ is a uniqueness set, we have that F is identically zero on $\mathbb{R}^d$. This implies that $d\nu = 0$.

It remains to prove that the space M_2 defined by (3.11) is dense in $C(X)$. If for all $g \in C(X)$ we have that

$$\int_{\mathbb{R}^d} (\mathcal{K}g)(\mathbf{x}) d\nu(\mathbf{x}) = 0,$$

then for all $g \in C(X)$ it follows that

$$\int_X g(\mathbf{x}) \left(\int_X K(\mathbf{x} - \mathbf{y}) d\nu(\mathbf{y}) \right) d\mathbf{x} = 0.$$

This ensures for all $\mathbf{x} \in \mathbb{R}^d$ that

$$\int_X K(\mathbf{x} - \mathbf{y}) d\nu(\mathbf{y}) = 0.$$

Appealing to the already verified density of space M_1 in $C(X)$, we conclude that the space M_2 is also dense in $C(X)$. □

Our goal is to develop useful estimates for $I_{\mathbf{p}}(f,t)$ when $f \in B^{\mathbf{r}}_{\mathbf{p},\theta}(\mathbb{R}^d)$. To this end, we rely upon the inequality

$$I_{\mathbf{p}}(f,t) \le \|f - h\|_{L_{\mathbf{p}}(I^d)} + \|h - \mathcal{K}g\|_{L_{\mathbf{p}}(I^d)} \tag{3.15}$$

where, depending on f, we shall choose h carefully in the space of entire functions of exponential type and then a g with $\|g\|_{L_{\mathbf{p}}(I^d)} \le t$ for which $\mathcal{K}g$ approximates h well. We begin with a description of how to choose h .

For any vector $\mathbf{v} := (v_j : j \in Z_d)$ with positive coordinates we say an entire function h is of *exponential type* $\mathbf{v}$ provided that for every $\epsilon > 0$ there exists a positive number c such that for all complex vectors $\mathbf{z} := (z_j : j \in Z_d) \in \mathbb{C}^d$ we have the bound

$$|h(\mathbf{z})| \le c \exp\left(\sum_{j \in Z_d} (v_j + \epsilon)|z_j| \right).$$

We denote by $M_{\mathbf{v}}(\mathbb{R}^d)$ the subset of all entire functions of exponential type which are *bounded* on $\mathbb{R}^d$ and for any $\mathbf{p} \in Q^d$ we let

$$M_{\mathbf{v},\mathbf{p}}(\mathbb{R}^d) := M_{\mathbf{v}}(\mathbb{R}^d) \cap L_{\mathbf{p}}(\mathbb{R}^d).$$

Every vector $\mathbf{v} = (v_j : j \in Z_d) \in \mathbb{R}^d_+$ determines the rectangle

$$I^d_{\mathbf{v}} := \prod_{j \in Z_d} [-v_j, v_j]$$

and when $\mathbf{v} = v\mathbf{e}$, $v \in \mathbb{R}_+$, we denote it by I^d_v. According to the Schwartz theorem, we have that

$$M_{\mathbf{v},\mathbf{p}}(\mathbb{R}^d) = \left\{ f \in L_{\mathbf{p}}(\mathbb{R}^d) : \operatorname{supp} \widehat{f} \subseteq I_{\mathbf{v}} \right\},$$

which in the special case $p = 2\mathbf{e}$ reduces to the Paley-Wiener theorem, cf., [16]. For a fixed $v \in \mathbb{R}_+$, we define

$$M_{v,\mathbf{p}}(\mathbb{R}^d) := \bigcup \left\{ M_{\mathbf{v},\mathbf{p}}(\mathbb{R}^d) : \mathbf{v} = (v_j : j \in Z_d), \prod_{j \in Z_d} v_j \le v \right\}.$$

To estimate the CS-functional in anisotropic Besov spaces, we recall the definition of an operator which appears in [13] which we shall use here. To this end, for $u \in \mathbb{R}_+$, we choose a function w_u which is *positive* a.e. on $\mathbb{R}_+$ such that

$$\int_{\mathbb{R}} w_u(t)dt = 1. \tag{3.16}$$

For $\mathbf{u} := (u_i : i \in Z_d) \in \mathbb{R}_+^d$, $\mathbf{x} \in \mathbb{R}^d$ and $f \in B^{\mathbf{r}}_{\mathbf{p},\theta}(\mathbb{R}^d)$, $i \in Z_d$, we define

$$(\mathcal{T}_{u_i} f)(\mathbf{x}) := (-1)^{k_i+1} \int_{\mathbb{R}} w_{u_i}(t_i)(\Delta_{t_i}^{k_i} f)(\mathbf{x})dt_i + f(\mathbf{x}). \tag{3.17}$$

By equation (3.8), there exists constants d_{ij}, $j \in Z_{k_i}$, $i \in Z_d$ such that for all functions f

$$\sum_{j \in Z_{k_i}} d_{ij} f(\mathbf{x} + jt_i\mathbf{e}_i) - f(\mathbf{x}) = (-1)^{k_i+1} \Delta_{t_i}^{k_i} f(\mathbf{x}). \tag{3.18}$$

For any $i \in Z_d$, we remark that

$$\sum_{j \in Z_{k_i}} d_{ij} = 1$$

and

$$\sum_{j \in Z_{k_i}} |d_{ij}| = 2^{k_i} - 1.$$

When $t \in \mathbb{R}$ and $i \in Z_d$, we let

$$G_{u_i}(t) := \sum_{j \in Z_{k_i}} \frac{d_{ij}}{j} w_{u_i}\left(\frac{t}{j}\right) \tag{3.19}$$

and observe from formulas (3.17)-(3.19) that $\mathcal{T}_{u_i}$ has the alternative representation

$$(\mathcal{T}_{u_i} f)(\mathbf{x}) := \int_{\mathbb{R}} G_{u_i}(t_i) f(\mathbf{x} + jt_i\mathbf{e}_i)dt_i, \quad \mathbf{x} \in \mathbb{R}^d. \tag{3.20}$$

We define the value of a kernel $G_{\mathbf{u}}$ at $\mathbf{t} := (t_i : i \in Z_d) \in \mathbb{R}^d$ by the formula

$$G_{\mathbf{u}}(\mathbf{t}) := \prod_{j \in Z_d} G_{u_j}(t_j)$$

and introduce the operator

$$\mathcal{T}_{\mathbf{u}} := \mathcal{T}_{u_1} \circ \cdots \circ \mathcal{T}_{u_d}.$$

Consequently, $\mathcal{T}_{\mathbf{u}}$ is given by

$$(\mathcal{T}_{\mathbf{u}} f)(\mathbf{x}) = \int_{\mathbb{R}^d} G_{\mathbf{u}}(\mathbf{t}) f(\mathbf{x} + \mathbf{t})d\mathbf{t}, \quad \mathbf{x} \in \mathbb{R}^d. \tag{3.21}$$

In the next two theorems, we gather together required facts about the operator $\mathcal{T}_{\mathbf{u}}$.

Theorem 6 *If $\mathbf{p} \in Q^d$, $\theta \in V$, $\mathbf{u}, \mathbf{r} \in \mathbb{R}_+^d$ then operator $\mathcal{T}_{\mathbf{u}}$ is bounded both from $L_{\mathbf{p}}(\mathbb{R}^d)$ to $L_{\mathbf{p}}(\mathbb{R}^d)$ and from $B^{\mathbf{r}}_{\mathbf{p},\theta}(\mathbb{R}^d)$ to $B^{\mathbf{r}}_{\mathbf{p},\theta}(\mathbb{R}^d)$.*

PROOF. For $i \in Z_d$, by the definition of G_{u_i} and (3.16) we have that

$$\int_{\mathbb{R}} |G_{u_i}(t)|dt \le \sum_{j \in Z_i} |d_{ij}| = 2^{k_i} - 1$$

from which it follows that

$$\int_{\mathbb{R}^d} |G_{\mathbf{u}}(\mathbf{t})|d\mathbf{t} < 2^{\sigma}$$

where

$$\sigma := \sum_{j \in Z_d} k_j.$$

Therefore, by the Young inequality, we have for $f \in L_{\mathbf{p}}(\mathbb{R}^d)$ that

$$\|\mathcal{T}_{\mathbf{u}} f\|_{\mathbf{p}} \le 2^{\sigma} \|f\|_{\mathbf{p}}. \tag{3.22}$$

Since $\mathcal{T}_{\mathbf{u}}$ is a convolution operator it commutes with difference operators and therefore employing estimate (3.22), we obtain for any $t_j \in \mathbb{R}$ and $j \in Z_d$ the estimate

$$\|\Delta_{t_j}^{k_j} \mathcal{T}_{\mathbf{u}} f\|_{\mathbf{p}} = \|\mathcal{T}_{\mathbf{u}} \Delta_{t_j}^{k_j} f\|_{\mathbf{p}} \le 2^{\sigma} \|\Delta_{t_j}^{k_j} f\|_{\mathbf{p}},$$

which implies that

$$|\mathcal{T}_{\mathbf{u}} f|_{B^{r_j}_{\mathbf{p},\theta,j}} \le 2^{\sigma} |f|_{B^{r_j}_{\mathbf{p},\theta,j}}, \quad j \in Z_d.$$

These estimates, together with (3.22) and the definition of Besov norms, yield the inequality

$$\|\mathcal{T}_{\mathbf{u}} f\|_{B^{\mathbf{r}}_{\mathbf{p},\theta}} \le 2^{\sigma} \|f\|_{B^{\mathbf{r}}_{\mathbf{p},\theta}}$$

and complete the proof. □

In the next theorem, we provide an estimate for approximation by the operator $\mathcal{T}_{\mathbf{u}}$. The estimate we present below uses the function

$$\widetilde{w}_{u_j}(t) := |t|^{r_j + \theta^{-1}} w_{u_j}(t), \quad t \in \mathbb{R}, \tag{3.23}$$

where $\theta \in V$.

Theorem 7 *If* $\mathbf{r}, \mathbf{u} \in \mathbb{R}^d_+$, $j \in Z_d$, $\mathbf{p} \in Q^d$, $\theta \in V$ *and* $1/\theta + 1/\theta' = 1$ *then*

$$\|f - \mathcal{T}_{u_j} f\|_{\mathbf{p}} \le \|\widetilde{w}_{u_j}\|_{L_{\theta'}(\mathbb{R})} |f|_{B^{r_j}_{\mathbf{p},\theta,j}(\mathbb{R}^d)}.$$

PROOF. We know from (3.17) that

$$f(\mathbf{x}) - (\mathcal{T}_{u_j} f)(\mathbf{x}) = (-1)^{k_j} \int_{\mathbb{R}} w_{u_j}(t_j)(\Delta_{t_j}^{k_j} f)(\mathbf{x}) dt_j, \quad \mathbf{x} \in \mathbb{R}^d.$$

Therefore, the Minkowskii inequality gives the estimate

$$\|f - \mathcal{T}_{u_j} f\|_{\mathbf{p}} \le \int_{\mathbb{R}} w_{u_j}(t_j) \|\Delta_{t_j}^{k_j} f(\cdot)\|_{\mathbf{p}} dt_j. \tag{3.24}$$

Hence, by the Hölder inequality and the definition of $\widetilde{w}_{u_j}$ we obtain the desired result.

□

We specialize this theorem to the special weight function

$$w_{su}(t) := \mu^{-1} \left| \frac{\sin ut}{ut} \right|^s, \quad t \in \mathbb{R}, \tag{3.25}$$

where $s \in W := V \setminus \{1\}$, $v \in \mathbb{R}_+$ and μ is chosen so that equation (3.16) is satisfied. When $s - r_j \in W, j \in Z_d$, we obtain the estimate

$$\|\widehat{w}_{su_j}\|_{L_{\theta'}(\mathbb{R})} \le \rho u_j^{-r_j}, \tag{3.26}$$

for all $u_j \in \mathbb{R}_+$, where

$$\rho := \left(\frac{2^s}{(s - r_j - 1)\theta'} \right)^{1/\theta'}.$$

We wish to combine this bound for the special weight function (3.25) and Theorem 7 to estimate the efficiency of approximation by the operator $\mathcal{T}_{\mathbf{u}}$. We shall do this only for a *restricted* set of vectors $\mathbf{u} \in \mathbb{R}_+^d$. Specifically, for every $v \in \mathbb{R}_+$, we define the coordinates of the vector $\mathbf{u} \in \mathbb{R}_+^d$ by the equations

$$u_j^{r_j} = v^{\gamma(\mathbf{r})}, \quad j \in Z_d, \tag{3.27}$$

where

$$\gamma^{-1}(\mathbf{r}) := \sum_{j \in Z_d} \frac{1}{r_j}. \tag{3.28}$$

Note that these definitions imply that

$$\prod_{j \in Z_d} u_j = v.$$

With this choice of the vector $\mathbf{u}$ and the weight function (3.25) we denote the corresponding operator $\mathcal{T}_{\mathbf{u}}$ by $\mathcal{S}_v$ and use the quantity $\overline{r} := \max\{r_j : j \in Z_d\}$ below.

Theorem 8 *If $\mathbf{r} \in \mathbb{R}_+^d$, $v \in \mathbb{R}_+$, $\mathbf{p} \in Q^d$, $\theta \in V$, $s - \overline{r} \in W$ and $f \in L_{\mathbf{p}}(\mathbb{R}^d)$ then $\mathcal{S}_v f \in M_{v,\mathbf{p}}(\mathbb{R}^d)$.*

PROOF. The first step of the proof is to observe that $G_{\mathbf{u}}$ is an entire function in $M_{\mathbf{u},\mathbf{p}}(\mathbb{R}^d)$ for *all* $\mathbf{p} \in Q^d$ when $s \in W$. Since $\mathcal{S}_v$ is a convolution operator, its Fourier transform is supported on the cube $I_{\mathbf{u}}^d$. Also, from Theorem 6 we conclude that $\mathcal{S}_v \in L_{\mathbf{p}}(\mathbb{R}^d)$. Therefore, it remains to show that $\mathcal{S}_v f \in L_\infty(\mathbb{R}^d)$ which follows from the Hölder inequality and the fact that $G_{\mathbf{u}} \in L_{\mathbf{p}}(\mathbb{R}^d)$ for all $\mathbf{p} \in Q^d$. □

The next theorem describes the approximation property of the operator $\mathcal{S}_v$. For this purpose, we let $\beta := d\rho 2^\sigma$ and $\eta := (1/\beta)^{1/\gamma(\mathbf{r})}$.

Theorem 9 *If $\mathbf{r} \in \mathbb{R}_+^d$, $v \in \mathbb{R}_+$, $\mathbf{p} \in V^d$, $\theta \in V$ and $f \in B^{\mathbf{r}}_{\mathbf{p},\theta}(\mathbb{R}^d)$ then operator $\mathcal{S}_v$ has the property that*

$$\|f - \mathcal{S}_v f\|_{\mathbf{p}} \le \beta v^{-\gamma(\mathbf{r})} \|f\|_{B^{\mathbf{r}}_{\mathbf{p},\theta}(\mathbb{R}^d)}.$$

Moreover, for all $v \ge \eta$ we have that

$$\|\mathcal{S}_v f\|_{\mathbf{p}} \le \|f\|_{B^{\mathbf{r}}_{\mathbf{p},\theta}(\mathbb{R}^d)}.$$

PROOF. Let $\mathbf{t}_0 := \mathbf{0}$ and for $j \in Z_d$, we define the vectors

$$\mathbf{t}_j := \sum_{k \in Z_j} t_k \mathbf{e}_k.$$

For $\mathbf{x} \in \mathbb{R}^d_+$ and $f \in B^{\mathbf{r}}_{\mathbf{p},\theta}(\mathbb{R}^d)$ we also make use of the quantity

$$E_j(\mathbf{x}, \mathbf{t}_{j-1}) := f(\mathbf{x} + \mathbf{t}_{j-1}) - (\mathcal{T}_{u_j} f)(\mathbf{x} + \mathbf{t}_{j-1}).$$

From Theorem 7, the estimate (3.26) and the translation invariance of the $L_{\mathbf{p}}(\mathbb{R}^d)$ norm, we have that

$$\|E_j(\cdot, \mathbf{t}_{j-1})\|_{\mathbf{p}} \le \rho v^{-\gamma(\mathbf{r})} |f|_{B^{r_j}_{\mathbf{p},\theta,j}}. \tag{3.29}$$

Since

$$I - \mathcal{S}_v = \sum_{j \in Z_d} (\mathcal{T}_{u_0} \circ \cdots \circ \mathcal{T}_{u_{j-1}} - \mathcal{T}_{u_0} \circ \cdots \circ \mathcal{T}_{u_j})$$

where we set $\mathcal{T}_{u_0} := I$, we obtain for $\mathbf{x} \in \mathbb{R}^d$ that

$$f(\mathbf{x}) - (\mathcal{S}_v f)(\mathbf{x}) = \sum_{j \in Z_d} \int_{\mathbb{R}} \cdots \int_{\mathbb{R}} \prod_{i \in Z_{j-1}} G_{u_i}(t_i) E_j(\mathbf{x}, \mathbf{t}_{j-1}) d\mathbf{t}_{j-1}.$$

Using the Minkowskii inequality, (3.29) and the above equation we conclude for all $f \in B^{\mathbf{r}}_{\mathbf{p},\theta}(\mathbb{R}^d)$ and all $v \in \mathbb{R}_+$ that

$$\|f - \mathcal{S}_v f\|_{\mathbf{p}} \le \beta v^{-\gamma(\mathbf{r})} \sum_{j \in Z_d} |f|_{B^{\mathbf{r}}_{\mathbf{p},\theta,j}(\mathbb{R}^d)},$$

which proves the first part of the theorem. Finally, we have that

$$\|\mathcal{S}_v f\|_{\mathbf{p}} \le \|f\|_{\mathbf{p}} + \beta v^{-\gamma(\mathbf{r})} \sum_{j \in Z_d} |f|_{B^{\mathbf{r}}_{\mathbf{p},\theta,j}(\mathbb{R}^d)}$$

and the second part of this theorem follows immediately. □

When $\mathbf{p} = p\mathbf{e}$, $p \in \mathbb{R}_+$ this theorem essentially appears in [7]. The operator $\mathcal{S}_v$ was designed to give a good approximation to a function $f \in B^{\mathbf{r}}_{\mathbf{p},\theta}(\mathbb{R}^d)$. This will take care of our choice for the function h in inequality (3.15). With this in hand, we now turn our attention to obtaining a good choice of the function g in inequality (3.15).

As in [15] we use a *collocation scheme* on a *uniform grid* to obtain a function g from h and provide estimates for a larger class of kernels than those considered there. For a given function φ which is positive definite and of compact support on $\mathbb{R}^d$ we consider a function g of the form

$$g := \sum_{\mathbf{j} \in Z^d_N} d_{\mathbf{j}} \varphi(N \cdot - \mathbf{j}). \tag{3.30}$$

Let h be any function defined everywhere on I^d and bounded there. We choose the coefficient vector $\mathbf{d} = (d_{\mathbf{j}} \colon \mathbf{j} \in Z^d_N)$ so that

$$(\mathcal{K}g)(\mathbf{j}/N) = h(\mathbf{j}/N), \quad \mathbf{j} \in Z^d_N. \tag{3.31}$$

Thus, we have that

$$\mathcal{K}g = \sum_{\mathbf{j}\in Z_N^d} d_{\mathbf{j}}\Phi(\cdot - \mathbf{j}/N) \tag{3.32}$$

where

$$\Phi := K * \varphi(N\cdot). \tag{3.33}$$

We define the matrix

$$\mathbf{A} := (\Phi(\mathbf{j}/N - \mathbf{k}/N))_{\mathbf{j},\mathbf{k}\in Z_N^d}$$

and the vector

$$\mathbf{b} := (h(\mathbf{j}/N) : \mathbf{j} \in Z_N^d)$$

so that the coefficient vector $\mathbf{d}$ is determined by the linear system

$$\mathbf{A}\mathbf{d} = \mathbf{b}. \tag{3.34}$$

We display the (linear) dependency of g on h by writing $g := \mathcal{G}_N h$ and turn our attention to estimating the norm of $\mathcal{G}_N h$. To this end, for *any* function ϕ we introduce the one-periodic function

$$\phi^0 := \sum_{\mathbf{k}\in Z^d} \phi(\cdot + \mathbf{k}),$$

the N-periodic function

$$\phi_N := \sum_{\mathbf{k}\in Z^d} \widehat{\phi}(\cdot + 2N\pi\mathbf{k})$$

and the min-function defined for $\mathbf{v} \in \mathbb{R}^d$ by the formula

$$\underline{\phi}_N(\mathbf{v}) := \inf\left\{\phi_N(\mathbf{t}) : \mathbf{t} \in I_{\mathbf{v}}^d\right\}.$$

The theorem below not only estimates the norm of $\mathcal{G}_N h$ but also establishes that the matrix $\mathbf{A}$ is nonsingular when $\underline{\Phi}_N(N\pi\mathbf{e}) > 0$. In the statement of the result, we use the set

$$Y^d := \{\mathbf{p} \in \mathbb{R}^d : \mathbf{e} \le \mathbf{p} \le 2\mathbf{e}\} \cup \{\mathbf{p} \in \mathbb{R}^d : 2\mathbf{e} \le \mathbf{p}\},$$

which is, for $d \in W$, not the same as V^d.

Theorem 10 *For $\mathbf{p} \in Y^d$ and $\mathcal{G}_N h$ defined by equations (3.31) and (3.32) where the vector $\mathbf{d}$ is given by the equation (3.34), we have that*

$$\|\mathcal{G}_N h\|_{L^p(I^d)} \le \frac{N^{(d/2-1/\gamma(\mathbf{p}))_+}}{\underline{\Phi}_N(N\pi\mathbf{e})}\|\varphi^0\|_{L^\infty(I^d)}\|h\|_{L^\infty(I^d)}.$$

PROOF. We obtain an estimate for the norm of $\mathbf{d}$ by observing that

$$\mathbf{d}^T\mathbf{A}\mathbf{d} = \frac{1}{(2\pi)^d}\int_{I_{N\pi\mathbf{e}}} \Phi_N(\mathbf{t})\left|\sum_{\mathbf{j}\in Z_N^d} d_{\mathbf{j}} e^{i\mathbf{j}/N\cdot\mathbf{t}}\right|^2 d\mathbf{t},$$

and

$$\frac{1}{(2\pi)^d}\int_{I_{N\pi\mathbf{e}}}\left|\sum_{\mathbf{j}\in Z_N^d} d_{\mathbf{j}} e^{i\mathbf{j}/N\cdot\mathbf{t}}\right|^2 d\mathbf{t} = N^d\|\mathbf{d}\|_2^2,$$

from which we conclude that

$$\mathbf{d}^T\mathbf{A}\mathbf{d} \geq \underline{\Phi}_N(N\pi\mathbf{e})\|\mathbf{d}\|_2^2. \tag{3.35}$$

Multiplying equation (3.34) by $\mathbf{d}$ from the left yields the formula

$$\mathbf{d}^T\mathbf{A}\mathbf{d} = \mathbf{d}^T\mathbf{h}. \tag{3.36}$$

Consequently, by the Cauchy-Schwarz inequality we have that

$$\mathbf{d}^T\mathbf{A}\mathbf{d} \leq N^{d/2}\|\mathbf{d}\|_2\|h\|_{L_\infty(I^d)}. \tag{3.37}$$

Combining inequalities (3.35) and (3.37) gives the estimate

$$\|\mathbf{d}\|_2 \leq \frac{N^{d/2}}{\underline{\Phi}_N(N\pi\mathbf{e})}\|h\|_{L_\infty(I^d)}. \tag{3.38}$$

To continue the proof, we use the inequality

$$\|g\|_{\mathbf{p}} \leq N^{-1/\gamma(\mathbf{p})}\|\varphi^0\|_{L_{\mathbf{p}}(I^d)}\|\mathbf{d}\|_{\mathbf{p}}, \tag{3.39}$$

valid for any function g given in the form (3.30), which was proved in [6] for the case $\mathbf{p} = p\mathbf{e}$, $p \in V$. However, since the proof given there is based on the Young inequality for convolution sequences, which is valid for arbitrary vectors, inequality (3.39) also holds in this generality.

We also need the fact that for any vectors $\mathbf{d}, \mathbf{r}, \mathbf{s} \in \mathbb{R}^d$ with $\mathbf{r} \geq \mathbf{s}$ and a function f there holds the inequalities

$$\|\mathbf{d}\|_{\mathbf{r}} \leq \|\mathbf{d}\|_{\mathbf{s}}$$

and

$$\|f\|_{L_{\mathbf{s}}(I^d)} \leq \|f\|_{L_{\mathbf{r}}(I^d)}.$$

Therefore, for $\mathbf{e} \leq \mathbf{p} \leq 2\mathbf{e}$, we have that

$$\|\mathcal{G}_N h\|_{L_{\mathbf{p}}(I^d)} \leq N^{-d/2}\|\varphi^0\|_{L_2(I^d)}\|\mathbf{d}\|_2 \leq \frac{\|\varphi^0\|_{L_\infty(I^d)}\|h\|_{L_\infty(I^d)}}{\underline{\Phi}_N(N\pi\mathbf{e})}.$$

While for $2\mathbf{e} \leq \mathbf{p}$, we have that

$$\|\mathcal{G}_N h\|_{L_{\mathbf{p}}(I^d)} \leq N^{-1/\gamma(\mathbf{p})}\|\varphi^0\|_{L_\infty(I^d)}\|\mathbf{d}\|_2 \leq \frac{N^{d/2-1/\gamma(\mathbf{p})}}{\underline{\Phi}_N(N\pi\mathbf{e})}\|\varphi^0\|_{L_\infty(I^d)}\|h\|_{L_\infty(I^d)},$$

proving the theorem. □

In the next theorem we relate various the norms of *any* entire function h of a given *exponential type*. For this purpose, we use the vector

$$\overline{\mathbf{p}} := \frac{\mathbf{p}}{2-\mathbf{p}}$$

and denote by P^d the set of all vectors $\mathbf{p} \in \mathbb{R}^d$ such that $\mathbf{e} \leq \mathbf{p} \leq 2\mathbf{e}$ whose coordinates of $\mathbf{p}$ form a *nonincreasing sequence*. Moreover, whenever we have a $\mathbf{v} = (v_j : j \in Z_d)$ we set $\|\mathbf{v}\|_\infty := \max\{v_j : j \in Z_d\}$.

Theorem 11 *If $h \in M_{\mathbf{v},\mathbf{p}}(\mathbb{R}^d)$, $\mathbf{p} \in P^d$ and $q - 1 \in Q$ then*

$$\|h\|_q \le (2\pi)^{-d/q'+\gamma^{-1}(\mathbf{p}')} \prod_{j\in Z_d} (2v_j)^{(1-2/q)1/p_j+2/q\overline{p}_j} \|h\|_{\mathbf{p}}.$$

Moreover, when $\|\mathbf{v}\|_\infty \le N\pi$ then

$$\|h\|_q \le (2\pi)^{(2/q)\gamma^{-1}(\mathbf{p})-d/q} N^{(1+2/q)\gamma^{-1}(\mathbf{p})-2d/q} \|h\|_{\mathbf{p}}.$$

PROOF. By the Fourier inversion formula and Hölder inequality, we have for an $\mathbf{x} \in \mathbb{R}^d$ that

$$|h(\mathbf{x})| \le (2\pi)^{-d} \int_{I_\mathbf{v}^d} |\widehat{h}(\mathbf{y})| d\mathbf{y} \le (2\pi)^{-d} \|\widehat{h}\|_{\mathbf{p}'} \prod_{j\in Z_d} (2v_j)^{1/p_j}.$$

However, the Hausdorff-Young inequality, as it appears in [3], ensures under the hypothesis of the theorem that

$$\|\widehat{h}\|_{\mathbf{p}'} \le (2\pi)^{\gamma^{-1}(\mathbf{p}')} \|h\|_{\mathbf{p}},$$

which proves the inequality for $q = \infty$. The proof of the case $q = 2$ is similar. Indeed, we have that

$$\begin{aligned} \|h\|_2 = (2\pi)^{-d/2} \left(\int_{I_\mathbf{v}^d} |\widehat{h}(\mathbf{y})|^2 d\mathbf{y} \right)^{1/2} &\le (2\pi)^{-d/2} \prod_{j\in Z_d} (2v_j)^{1/\overline{p}_j} \|\widehat{h}\|_{L_{\mathbf{p}'}(I_\mathbf{v}^d)} \\ &\le (2\pi)^{-d/2+\gamma^{-1}(\mathbf{p}')} \prod_{j\in Z_d} (2v_j)^{1/\overline{p}_j} \|h\|_{\mathbf{p}}. \end{aligned}$$

The general case is proved by combining the first two in a straight forward manner. Moreover, using the condition on the vector $\mathbf{v}$ the remaining inequality follows by a direct computation. □

We now turn to the task of bounding the error $h - \mathcal{K}g$ by using facts concerning interpolation in a reproducing kernel Hilbert space which we describe next. The idea is to interpret the interpolation problem which determines $\mathcal{K}g$ from h as an *optimal* interpolant in an appropriate Hilbert space. To this end, we let $\mathbb{H}$ be a Hilbert space of real-valued functions on some domain D with *reproducing kernel* $K(\cdot,\cdot)$. This kernel should not be confused with the kernel that determines the convolution operator $\mathcal{K}$. For every finite set $D_N := \{y^j : j \in Z_N\} \subseteq D$ we let

$$K(Y) = \det\,(K(y^i, y^j) : i, j \in Z_N)$$

and assume that $K(D_N) > 0$. (It is nonnegative since K is a reproducing kernel for $\mathbb{H}$.) We wish to *estimate* a function $f \in \mathbb{H}$ from its values on D_N, that is, from the *information* vector $I(f) := (f(y) : y \in D_N)$ known to lie in the *hypothesis space*, $\{f : \|f\|_{\mathbb{H}} \le 1\}$. From the point of view of the theory of *optimal estimation* [12], we consider the effect of *any* algorithm $A : I(\mathbb{H}) \to \mathbb{R}$ which produces the error

$$\sup\{|f(x) - A(I(f))| : \|f\|_{\mathbb{H}} \le 1\}.$$

An *optimal* algorithm is one which minimizes this expression over *all* algorithms and the value of the infimum, which we denote by $\mathcal{E}(\mathbb{H}, D_N)$, is call the *intrinsic* error

of estimating a function f in the hypothesis space $\{f : \|f\|_{\mathbb{H}} \leq 1\}$ at $x \in D$ from it values on the set D_N. It is known from [12] that

$$\mathcal{E}(\mathbb{H}, D_N) = \max\{f(x) : I(f) = 0,\ \|f\|_{\mathbb{H}}\| \leq 1\}$$

and the Hahn-Banach duality gives the alternative expression for the intrinsic error

$$\mathcal{E}(\mathbb{H}, D_N) = \min\left\{\left\|K(x,\cdot) - \sum_{y\in D_N} d_y K(y,\cdot)\right\| : d_y \in \mathbb{R}\right\}.$$

Moreover, we can describe in concrete terms an optimal algorithm which is, in fact, *linear* in the information operator. To this end, the *optimal* interpolant to a function f with domain D relative to the set D_N is the function p of the form

$$p = \sum_{y\in D_N} d_y K(y,\cdot)$$

chosen so that $f(y) = p(y)$ for all $y \in D_N$. It then follows that

$$|f(x) - p(x)| \leq \min\left\{\left\|K(x,\cdot) - \sum_{y\in D_N} d_y K(y,\cdot)\right\| : d_y \in \mathbb{R}\right\}\|f\|_{\mathbb{H}}.$$

Moreover, the choice of the constants $d_y(x), x \in D_N$ which minimize the above expression are characterized by the requirement that $d_y(y') = \delta_{yy'}$ for $y, y' \in D_N$ and $d_y \in \text{span}\{K(y,\cdot) : y \in D_N\}$. That is, $\{d_y : y \in D_N\}$ are *Lagrange* functions for this interpolation problem.

Although we shall not require it here we point out that the minimum above can be evaluated as a ratio of two Fredholm determinants of the kernel K. Specifically, we obtain for any $x \in D$ the inequality

$$|f(x) - p(x)| \leq \sqrt{\frac{K(x \cup D_N)}{K(D_N)}}\|f\|_{\mathbb{H}}.$$

We shall now apply this principle in the current context. This requires the assumption that $\widehat{\Phi}$ is *positive* a.e. on $\mathbb{R}^d$ so that $\Phi(\mathbf{x}-\mathbf{y})$, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ is a reproducing kernel for functions on $\mathbb{R}^d$ with inner product

$$\langle f, g\rangle_\Phi := \frac{1}{(2\pi)^d}\int_{\mathbb{R}^d} \frac{\widehat{f}(\mathbf{t})\overline{\widehat{g}(\mathbf{t})}}{\widehat{\Phi}(\mathbf{t})}d\mathbf{t}.$$

Since, the collocation scheme described above to generate the function Kg is, by equations (3.31) and (3.32), the optimal interpolant to the function h relative to the inner product $\langle\cdot,\cdot\rangle_\Phi$ we obtain the following estimate

$$\|h - Kg\|_{L_\infty(I^d)} \leq E(\Phi; N)\|h\|_\Phi, \tag{3.40}$$

where

$$E(\Phi; N) := \sup\left\{\inf\left\{\sqrt{Q_\Phi(\mathbf{w}, \mathbf{x})} : \mathbf{w} \in \mathbb{R}^{Nd}\right\} : \mathbf{x} \in I^d\right\}.$$

and $Q_\Phi(\cdot,\cdot)$ is the function defined for $\mathbf{w} := (w_\mathbf{k} : \mathbf{k} \in Z_N^d) \in \mathbb{R}^{N^d}$ and $\mathbf{x} \in \mathbb{R}^d$ by the formula

$$Q_\Phi(\mathbf{w}, \mathbf{x}) := (2\pi)^{-d} \int_{\mathbb{R}^d} \widehat{\Phi}(\mathbf{y}) \left| e^{i\mathbf{x}\cdot\mathbf{y}} - \sum_{\mathbf{k}\in Z_N^d} w_\mathbf{k} e^{i(\mathbf{k}/N)\cdot\mathbf{y}} \right|^2 d\mathbf{y}. \tag{3.41}$$

By our remarks above, for any vector $\mathbf{x} \in \mathbb{R}^d$, the quadratic form $Q_\Phi(\cdot, \mathbf{x})$ takes its minimum over $\mathbb{R}^{N^d}$ at the vector $\mathbf{u} = (u_\mathbf{k}(\mathbf{x}) : \mathbf{k} \in Z_N^d)$ where according to equations (3.31) and (3.32), its coordinates have the form

$$u_\mathbf{k}(\mathbf{x}) = \sum_{\mathbf{j}\in Z_N^d} \left(\mathbf{A}^{-1}\right)_{\mathbf{jk}} \Phi(\mathbf{x} - \frac{\mathbf{j}}{N}). \tag{3.42}$$

Therefore, we conclude that

$$\mathcal{K}g = \sum_{\mathbf{k}\in Z_N^d} u_\mathbf{k} h(\mathbf{k}/N).$$

When $h \in M_\mathbf{v}(\mathbb{R}^d)$ it is important to realize that the estimate (3.40) can be *improved.*

Theorem 12 *If $h \in M_\mathbf{v}(\mathbb{R}^d)$ with $\|\mathbf{v}\|_\infty \le N\pi$ and $\mathbf{p} \in P^d$ then*

$$\|h - \mathcal{K}g\|_{L_\infty(I^d)} \le (2\pi)^{-d/2+\gamma^{-1}(\mathbf{p}')} \prod_{j\in Z_d} (2v_j)^{1/\overline{p}_j} \underline{\Phi}_N^{-1}(N\pi\mathbf{e}) E(\Phi; N) \|h\|_\mathbf{p}.$$

Consequently, we have that

$$\|h - \mathcal{K}g\|_{L_\infty(I^d)} \le (2\pi)^{\gamma^{-1}(\overline{\mathbf{p}})/2} N^{\gamma^{-1}(\overline{\mathbf{p}})} \underline{\Phi}_N^{-1}(N\pi\mathbf{e}) E(\Phi; N) \|h\|_\mathbf{p}.$$

PROOF. For any $\mathbf{x} \in \mathbb{R}^d$ we have that

$$h(\mathbf{x}) - (\mathcal{K}g)(\mathbf{x}) = (2\pi)^{-d} \int_{I_\mathbf{v}^d} \widehat{h}(\mathbf{y}) \left(e^{i\mathbf{x}\cdot\mathbf{y}} - \sum_{\mathbf{k}\in Z_N^d} u_\mathbf{k}(\mathbf{x}) e^{i(\mathbf{k}/N)\cdot\mathbf{y}} \right) d\mathbf{y}.$$

Therefore, the equation

$$\int_{I_{N\pi\mathbf{e}}} \Phi_N(\mathbf{y}) \left(e^{i\mathbf{x}\cdot\mathbf{y}} - \sum_{\mathbf{k}\in Z_N^d} u_\mathbf{k}(\mathbf{x}) e^{i(\mathbf{k}/N)\cdot\mathbf{y}} \right)^2 d\mathbf{y} = \int_{\mathbb{R}^d} \Phi(\mathbf{y}) \left(e^{i\mathbf{x}\cdot\mathbf{y}} - \sum_{\mathbf{k}\in Z_N^d} u_\mathbf{k}(\mathbf{x}) e^{i(\mathbf{k}/N)\cdot\mathbf{y}} \right)^2 d\mathbf{y}$$

and the Cauchy-Schwarz inequality gives the desired estimate

$$\|h - \mathcal{K}g\|_{L_\infty(I^d)} \le \left((2\pi)^{-d} \int_{I_\mathbf{v}^d} \frac{|\widehat{h}(\mathbf{y})|^2}{\Phi_N(\mathbf{y})} d\mathbf{y} \right)^{1/2} E(\Phi; N).$$

Hence, we conclude that

$$\|h - \mathcal{K}g\|_{L_\infty(I^d)} \le \underline{\Phi}_N^{-1}(N\pi\mathbf{e}) E(\Phi; N) \|h\|_2$$

and we obtain the desired result by applying Theorem 11 to estimate the quantity $\|h\|_2$ in terms of $\|h\|_{\mathbf{p}}$. □

We now combine all our estimates to obtain the following bound for the CS-functional. For the statements we use the notation $\underline{r} := \min\{r_j : j \in Z_d\}$ whenever $\mathbf{r} = (r_j : j \in \mathbb{R}^d)$. One possible consequence of the previous estimates is the following fact.

Theorem 13 *Let $f \in B^{\mathbf{r}}_{\mathbf{p},\theta}(\mathbb{R}^d)$, $\mathbf{p} \in P^d$, $\mathbf{r} \in \mathbb{R}^d_+$ and*

$$t \geq \frac{\|\varphi^0\|_{L_\infty(I^d)}}{\underline{\Phi}_N(N\pi\mathbf{e})} N^{\gamma^{-1}(\mathbf{p})} \|f\|_{\mathbf{p}}$$

then there holds the inequality

$$I_{\mathbf{p}}(f,t) \leq \beta(N\pi)^{-\underline{r}} \|f\|_{B^{\mathbf{r}}_{\mathbf{p},\theta}(\mathbb{R}^d)} + (2\pi)^{\gamma^{-1}(\overline{\mathbf{p}})/2} N^{\gamma^{-1}(\overline{\mathbf{p}})} \underline{\Phi}_N^{-1}(N\pi\mathbf{e}) E(\Phi; N) \|f\|_{\mathbf{p}}.$$

PROOF. We return to inequality (3.15) and now make our choices for $h := \mathcal{S}_v f$ and $g := \mathcal{G}_N \mathcal{S}_v f$ where we require $0 < v < (N\pi)^{\frac{\underline{r}}{\gamma(\mathbf{r})}}$. From Theorem 9 we have that

$$\|f - h\|_{\mathbf{p}} \leq \beta(N\pi)^{-\underline{r}} \|f\|_{B^{\mathbf{r}}_{\mathbf{p},\theta}(\mathbb{R}^d)}.$$

Since by hypothesis $\mathbf{p} \in P^d$ we can appeal to Theorem 10 to conclude that

$$\|g\|_{L_{\mathbf{p}}(I^d)} \leq \frac{\|\varphi^0\|_{L_\infty(I^d)} \|\mathcal{S}_v f\|_{L_\infty(I^d)}}{\underline{\Phi}_N(N\pi\mathbf{e})}.$$

Using the bound

$$\|\mathcal{S}_v f\|_{L^\infty(I^d)} \leq N^{\gamma^{-1}(\mathbf{p})} \|f\|_{\mathbf{p}}$$

obtained from Theorem 11 we see that our hypothesis implies that $\|g\|_{L_{\mathbf{p}}(I^d)} \leq t$. The remainder of the proof also follows from Theorems 11 and 12. □

A more complex and potentially better bound follows by taking advantage of the freedom in the choice of the parameter v. The formulation of such an improvement should be clear to the reader. Although the bound above improves upon those given in [15] the method of proof closely follows the one given there. In the same spirit there follow the next two theorems.

Theorem 14 *If the hypothesis of Theorem 13 holds and K is the gaussian kernel $K(\mathbf{x}) = e^{-|\mathbf{x}|^2}$, $\mathbf{x} \in \mathbb{R}^d$, then*

$$I_{\mathbf{p}}(f,t) = O\left((\ln t)^{-\underline{r}/2}\right), \quad t \to \infty.$$

Theorem 15 *If the hypothesis of Theorem 13 holds, $\alpha > d$ and $K(\mathbf{x}) = (1+|\mathbf{x}|^2)^{-\alpha/2}$, $\mathbf{x} \in \mathbb{R}^d$, then*

$$I_{\mathbf{p}}(f,t) = O\left((\ln t)^{-\underline{r}}\right), \quad t \to \infty.$$

Bibliography

[1] V. V. Arestov, Approximation of unbounded operators by bounded operators and related extremal problems, *Russian Math. Surveys* **51** (1996) 1093-1126.

[2] J. Bergh and J. Löfström, *Interpolation Spaces, An Introduction*, Springer-Verlag, New York (1976).

[3] A. Benedec and R. Panzone, The spaces L_p with mixed norm, *Duke Math J.* **28** (1961) 301–324.

[4] S. Bochner, Hilbert distances and positive definite functions, *Annals Math.* **42** (1941) 647–656.

[5] F. Cucker and S. Smale, On the mathematical foundations of learning, *Bull. Amer. Math. Soc.* **39** (2002) 1–49.

[6] R. Q. Jia and C. A. Micchelli, Using the refinement equation for the construction of pre-wavelets II, Powers of Two, in *Curves and Surfaces* , P. J. Laurent, A. Le Méhauté and L. L. Schumaker (eds.), Academic Press, New York (1991) 209–246.

[7] Y. Jiang and Y. Liu, Average width and optimal recovery of multivariate Besov classes in $L_p(\mathbb{R}^d)$, *J. Approx. Theory* **102** (2000) 155–170.

[8] N. P. Korneichuk, *Extremal Problems of Approximation Theory,* Nauka, Moscow (1976) (Russian).

[9] S. N. Kudryavtsev, Some problems in approximation theory for a class of functions of finite smoothness, *Russian Acad. Sci. Sb. Math* **75**(1) (1993).

[10] C. A. Micchelli, Universal learning kernels, in preparation.

[11] C. A. Micchelli and A. Pinkus, Variational problems arising from balancing several error criteria, *Rend. Mat. Appl. Roma* **14** (1994) 37-86.

[12] C. A. Micchelli and T. J. Rivlin, A survey of optimal recovery, in *Optimal Estimation in Approximation Theory*, Plenum Press, New York (1977) 1–54.

[13] S. M. Nikolskii, *Approximation of Functions of Several Variables and Imbedding Theorem,* Nauka, Moscow (1977).

[14] S. Smale, Mathematical problems for the next century, In *Mathematics: Frontiers and Perspectives*, V. Arnold, M. Atiyah, P. Lax and B. Mazur, eds., AMS (2000) 271–294.

[15] S. Smale and D. Zhou, Estimating the approximation error in learning theory, preprint (2001).

[16] E. M. Stein and G. Weiss, *Introduction to Fourier Analysis on Euclidean Space*, Princeton Univ. Press, Princeton, NJ (1971).

[17] A. N. Tikhonov and V. Y. Arsenin, *Methods for Solving Ill-posed Problems*, Wiley, New York (1979).

Chapter 4

High-dimensional Approximation by Neural Networks

Věra Kůrková[1]

Abstract. Approximation of high-dimensional mappings by neural networks is investigated in the context of nonlinear approximation theory. It is shown that the "curse of dimensionality" can be avoided when functions to be approximated have small special norms, which are tailored to the type of computational units. Properties of such norms and method of derivation of their estimates are described. Estimates of rates of nonlinear approximation are applied to neural network learning formalized as approximate minimization of regularized empirical error functionals.

[1]This work was partially supported by GA ČR grant 201/02/0428.

4.1 Introduction

In contrast to classical artificial intelligence, which has modeled cognitive tasks using rule-based manipulation of symbols, connectionistic computational models employ learning in systems of distributed representations. While symbolic representations are typically low-dimensional, distributed ones tend to be *high-dimensional* as they describe objects in terms of many parameters. Many connectionistic systems can be formally described as mappings between such representations. For example NETtalk [47], which performs "reading aloud", maps binary vectors of length over 200 (coding segments of written text) to vectors with 26 real entries (coding phonemes). A vowel-recognizer performing "lip-reading" [48] transforms vectors of length 500 (corresponding to pixels from video) to a phonetic code of length 32.

The primary computational technique used in such systems is approximation of multivariable functions implemented in neural networks of various types, which are often simulated on classical computers. Approximation is sufficient as neural networks performance is usually followed by an error-correcting after-processing. One of the goals of research in computational intelligence is to understand which properties make neural networks efficient and flexible tools for learning reasonable approximations of high-dimensional mappings. Theoretical understanding can provide guidelines for design of computationally feasible procedures that can perform a large variety of tasks by merely changing network parameters during a learning mode.

Such a competence of a computational model is often called *universality* with respect to computations to be performed. A classical example of a universal computational model is the Turing machine, which is a procedure capable of realizing any algorithm. Within the last decade, it has been shown that many types of feedforward networks (including all standard types that are popular in applications as well as many others that may not have been considered by experimentalists) form universal computational models with respect to approximation of mappings between subsets of Euclidean spaces (see, e.g., [43], [31], [32] and the references there).

However for some high-dimensional tasks, implementation of theoretically optimal approximation procedures becomes unfeasible because of an unmanageably large number of parameters. Such tasks are limited by the "curse of dimensionality" [5], i.e., an exponentially fast scaling of the number of parameters with the number of variables. Nevertheless, experience has shown that some feedforward networks of moderate complexity performed quite well some tasks depending on hundreds of variables (e.g., [47], [48]).

Approximation theory offers some explanation of feasibility of such implementations. It has derived various upper bounds on complexity of approximating systems depending on the number of variables of functions to be approximated together with their other characteristics such as smoothness. Inspection of such bounds shows that we can compensate for increase in the number of variables by increase in smoothness. More precisely, for linear approximation there are tight estimates on the speed of decrease of worst-case errors of functions from balls in Sobolev spaces of the order of $\mathcal{O}(n^{-s/d})$, where d is the number of variables and s is the smoothness parameter for the Sobolev space, while n is the number of parameters of the linear approximating family [42]. Thus in linear approximation, the curse of dimensionality can be only

avoided when smoothness is increased together with the number of variables.

In this chapter, we describe ways to cope with the curse of dimensionality in nonlinear approximation schemes corresponding to connectionistic systems. We study such systems in the framework of *variable-basis approximation,* which includes feedforward networks as well as other nonlinear systems with free parameters. We show that for variable-basis approximation, the role of the characteristics to be kept low to cope with the curse of dimensionality is played by a *norm tailored to the particular basis.* In the case of feedforward networks, such a basis corresponds to the type of computational units (e.g., perceptrons or radial units). We show that in contrast to linear approximation, where with increasing number of variables the sets of functions that do not exhibit the curse of dimensionality are more and more constrained (as the requirements on the degree of their smoothness are increasing), in the case of feedforward networks, such sets of d-variable functions can be embedded into corresponding sets of $d+1$-variable functions. We describe properties of such sets and methods of estimation of norms that define them. Finally, we apply the results derived for nonlinear approximation to network learning formally described as minimization of regularized empirical error functionals over admissible sets of functions computable by neural networks. We derive estimates of rates of decrease of approximate infima of such functionals with increasing number of computational units.

The chapter is organized as follows. In section 4.2, concepts and notations are introduced concerning fixed and variable-basis approximation, feedforward neural networks and minimization of functionals. Section 4.3 is devoted to Maurey-Jones-Barron's theorem which is used as a main tool for derivation of estimates of rates of approximation. In section 4.4, a concept of "variational" norm tailored to a basis is defined and used to describe sets of multivariable functions that do not exhibit the curse of dimensionality. In section 4.5, estimates of rates of approximation are applied to learning from data modelled as approximate minimization of regularized empirical error functionals. In section 4.6, properties of such sets are compared with properties of sets playing a similar role in linear approximation. In sections 4.7 and 4.8, upper and lower bounds, resp., on variational norm are derived. The concept of variation is illustrated in section 4.9 by examples of real-valued Boolean functions with small variational norm with respect to perceptron networks.

4.2 Variable-basis Approximation and Optimization

By a normed linear space $(X, \|.\|)$ we mean a *real normed linear space.* $\mathcal{R}$ denotes the set of real numbers, $\mathcal{R}_+$ the set of positive reals and $\mathcal{N}_+$ the set of positive integers. Sequences (of real numbers, sets or elements of normed linear spaces) are denoted by $\{x_n\}$ instead of $\{x_n : n \in \mathcal{N}_+\}$.

A ball of radius r centered at $h \in X$ is denoted by $B_r(h, \|.\|) = \{f \in X : \|f - h\| \leq r\}$. We write shortly $B_r(\|.\|) = B_r(0, \|.\|)$ and $B_r(h) = B_r(h, \|.\|)$, $B_r = B_r(0)$ when it is clear which norm is used. A sphere is denoted $S_r(h, \|.\|) = \{f \in X : \|h - f\| = r\}$ and analogous shortenings as for balls are used.

For a subset M of a normed linear space $(X, \|.\|)$, $cl\, M$ denotes the *closure* of M in the norm-induced topology, for $f \in X$ the *distance* of f from M is denoted by $\|f - M\|$.

A functional $\Phi : (X, \|.\|) \to (-\infty, +\infty)$ is *continuous* at $f \in X$ if for all $\varepsilon > 0$ there exists $\eta < 0$ such that $\|f - g\| < \eta$ implies $|\Phi(f) - \Phi(g)| < \varepsilon$. *Modulus of continuity* of Φ at f is the function $\omega : \mathcal{R}_+ \to \mathcal{R}_+$ defined as $\omega(a) = \sup\{|\Phi(f) - \Phi(g)| : \|f - g\| \leq a\}$.

Φ is *convex* on $M \subseteq X$ if for all $h, g \in M$ and all $\lambda \in [0, 1]$, $\Phi(\lambda h + (1 - \lambda)g) \leq \lambda\Phi(h) + (1 - \lambda)\Phi(g)$. Φ is *strictly uniformly convex* on $M \subseteq (X, \|.\|)$ if there exists a function $\delta : \mathcal{R}_+ \to \mathcal{R}_+$ such that $\delta(0) = 0$, for all $t > 0$, $\delta(t) > 0$ and for all $h, g \in M$ and all $\lambda \in [0, 1]$, $\Phi(\lambda h + (1 - \lambda)g) \leq \lambda\Phi(h) + (1 - \lambda)\Phi(g) - \lambda(1 - \lambda)\delta(\|h - g\|)$. Any such function δ is called a *modulus of convexity* of Φ.

Using standard notation (see, e.g., [14]), we denote by (M, Φ) the problem of infimizing a functional Φ over a subset M of X. M is called a set of *admissible solutions* or an *admissible set*. We denote by $argmin\,(M, \Phi) = \{g \in M : \Phi(g) = \inf_{g \in M} \Phi(g)\}$ the set of *argminima* of the problem (M, Φ) and for $\eta > 0$, we denote by $argmin_\eta(M, \Phi) = \{g \in M : \Phi(g) < \inf_{g \in M} \Phi(g) + \eta\}$ the set of its *η-near argminima*.

A sequence $\{g_n\}$ of elements of M is called *Φ-minimizing over M* if $\lim_{n \to \infty} \Phi(g_n) = \inf_{g \in M} \Phi(g)$. The term "minimizing" is used here since it is widespread in the literature, although in general the sequence is only "infimizing", as the minimum might not be achieved. By the definition of infimum, for any problem (M, Φ) with M nonempty, there always exists a minimizing sequence.

Tasks representable as mappings can be performed by devices computing functions depending on two vector variables: an input vector (corresponding to a coded pattern to be recognized or transformed) and a parameter vector (to be adjusted during learning mode). Due to error-correcting afterprocessing (such as "best guess" [47]) it is sufficient when such devices compute mappings that perform tasks only approximately.

Classical linear approximation theory has explored properties of parametrized sets formed by linear combinations of the first n elements of a set of basis functions with a fixed ordering (e.g., polynomials ordered by degree or sines ordered by frequencies). Thus it can be called *fixed-basis approximation*.

Connectionistic systems exploit parametrized families with a flexible choice of basis functions, which form *variable-basis approximation* schemes. Formally, such a scheme is defined for any subset G of a real linear space X and any positive integer n as the set of all linear combinations of at most n elements of G, which is denoted $span_n G = \{\sum_{i=1}^n w_i g_i : w_i \in \mathcal{R}, g_i \in G\}$. With proper choices of G it includes feedforward networks with a single linear output unit and any number of hidden layers as well as other nonlinear families with free inner parameters such as free-nodes splines, rational functions with free poles, and trigonometric sums with free-frequencies.

Since in applications all parameters are bounded, we shall also consider variable-basis approximation schemes with constraints on coefficients of linear combinations of basis functions. If such a constraint is defined in terms of a norm on the space $\mathcal{R}^n$ of coefficient vectors, then the set of functions computable by such a scheme is, for a proper scalar c, contained in the set $conv_n G(c)$ of all convex combinations of at most n elements of the scaled set $G(c) = \{wg : g \in G, |w| \leq c\}$. This can be easily checked for a constraint defined in terms of l_1-norm. As all norms on $\mathcal{R}^n$ are equivalent, inclusion into $conv_n G$ holds for any norm on $\mathcal{R}^n$. We denote $conv\, G = \cup_{n \in \mathcal{N}_+} conv_n G$ and $span\, G = \cup_{n \in \mathcal{N}_+} span_n G$.

An important class of variable-basis functions are functions computable by feedforward networks. They are composed from *computational units*, which can be formally

described as functions $\phi : A \times \Omega \to \mathcal{R}$, where $A \subseteq \mathcal{R}^q$ is the set of "inner" parameters of the unit, while $\Omega \subseteq \mathcal{R}^d$ is the set of its inputs (we shall restrict our considerations to inputs in either the Euclidean cube $[0,1]^d$ or the Boolean one $\{0,1\}^d$).

We denote by $G_\phi(\Omega, A) = \{\phi(a,.) : \Omega \to \mathcal{R} : a \in A\}$ the set of functions on Ω computable by a computational unit ϕ with all possible choices of parameters $a \in A$. For $A = \mathcal{R}^d$, we write only $G_\phi(\Omega)$, while for $A = \mathcal{R}^d$ and $\Omega = [0,1]^d$, we write only G_ϕ. The set of functions computable by a neural network with n hidden units computing ϕ is $span_n G_\phi$ if all coefficients in linear combinations are allowed (i.e., output weights are unconstrained) or it is a subset of $conv_n G_\phi$ if a constraint in the form of a norm on output weights is imposed.

Standard types of computational units used in neurocomputing are perceptrons and RBF units, but this formalism also includes other types of parametrized mappings like trigonometric functions with frequencies playing the role of parameters. A *perceptron* with an activation function $\psi : \mathcal{R} \to \mathcal{R}$ computes $\phi((v,b),x) = \psi(v \cdot x + b) : A \times \Omega \to \mathcal{R}$ and *RBF unit* with radial function $\mathcal{R}_+ \to \mathcal{R}$ computes $\phi((v,b),x) = \psi(b(\|x - v\|) : A \times \Omega \to \mathcal{R}$. Standard types of activation functions are sigmoidal functions, i.e., nondecreasing functions $\sigma : \mathcal{R} \to \mathcal{R}$ with $\lim_{t \to -\infty} \sigma(t) = 0$ and $\lim_{t \to +\infty} \sigma(t) = 1$. Heaviside activation function ϑ is defined as $\vartheta(t) = 0$ for $t < 0$ and $\vartheta(t) = 1$ for $t \geq 0$. A standard radial function is the Gaussian function e^{-t^2}.

We denote by $P_d(\psi)$ the *set of functions computable by perceptrons with activation* ψ, i.e., $P_d(\psi) = G_\phi$ for $\phi((v,b),x) = \psi(v \cdot x + b) : A \times [0,1]^d \to \mathcal{R}$. Similarly, by $F_d(\psi, \|.\|)$ is denoted the *set of functions computable by RBF-units with radial function* ψ, i.e., $F_d(\psi) = G_\phi$ for $\phi((v,b),x) = \psi(b(\|x - v\|) : \mathcal{R}^{d+1} \times [0,1]^d \to \mathcal{R}$.

Since the set of *functions computable by perceptrons with Heaviside activation* $P_d(\vartheta)$ is equal to the set of characteristic functions of closed half-spaces of $\mathcal{R}^d$ restricted to $[0,1]^d$, we use a shorter notation H_d instead of $P_d(\sigma)$.

Many classes of feedforward networks are "universal approximators", i.e., for many types of computational units ϕ, the sets $\cup_{n \in \mathcal{N}_+} span_n G_\phi$ are dense in the space of all continuous functions $\mathcal{C}([0,1]^d)$ with the supremum norm or $\mathcal{L}_p([0,1]^d)$ with $\mathcal{L}_p$-norms, $p \in [1, \infty)$ (see [31], [32], [43]). Although density guarantees arbitrarily close approximation of all functions from $\mathcal{C}([0,1]^d)$ or $\mathcal{L}_p([0,1]^d)$, for practical purposes, its implications are limited to functions for which a sufficient accuracy can be achieved by $span_n G_\phi$ with n small enough to allow implementation.

4.3 Maurey-Jones-Barron's Theorem

In many practical applications of neural networks, tasks depending on hundreds of variables have been performed quite well by networks with only a moderate number of hidden units (see, e.g., [47], [48]). However, estimates on rates of approximation by neural networks derived from constructive proofs of the universal approximation property have not been able to explain such successes - the constructions used in such arguments lead to networks with a number of hidden units growing exponentially with the number of inputs. Such growth of complexity is called the "curse of dimensionality" [5].

It has been shown that in linear approximation of all continuous or $\mathcal{L}_p$-functions on

$[0,1]^d$, the curse of dimensionality is unavoidable and that one can only cope with it by restricting approximation to certain subsets that are shrinking with increasing number of variables [42]. Description of such sets has been derived from tight estimates of rates of approximation. Inspection of such estimates shows which constraints on functions to be approximated have to be imposed to compensate for increase of the number of variables.

For variable-basis approximation, such constraints can be derived from an upper bound on rates of variable-basis obtained by Jones [22] (in fact, his estimate is formulated for approximation by $conv_n G$, but as $conv_n G \subseteq span_n G$, it can be applied to variable-basis approximation). The same result has been earlier proven by Maurey by a probabilistic argument (it has been quoted in [44], see also [4]). Barron [4] has improved Jones' [22] upper bound and applied it to neural networks. Here, we state their result (we shall refer to it as MJB theorem or MJB bound) in a slightly reformulated way with a proof which is a simplification of Barron's argument. In the next section, we derive a corollary of MJB theorem in terms of a norm tailored to a basis and use it to describe conditions on multivariable functions that guarantee variable-basis approximation without the curse of dimensionality.

Theorem 1 *Let G be a bounded subset of a Hilbert space $(X, \|.\|)$ and $s_G = \sup_{g\in G} \|g\|$, then for every $f \in cl\, conv\, G$ and for every positive integer n, $\|f - conv_n G\| \leq \sqrt{\frac{s_G^2 - \|f\|^2}{n}}$.*

PROOF. Since the distance from $conv_n G$ is continuous on $(X, \|.\|)$ (see, e.g., [46]), it is sufficient to verify the statement for $f \in conv\, G$. Let $f = \sum_{j=1}^m a_j h_j$ be a representation of f as a convex combination of elements of G. Set $c = s_G^2 - \|f\|^2$. We show by induction that there exist a sequence $\{g_i\}$ of elements of G such that the barycenters $f_n = \sum_{i=1}^n \frac{g_i}{n}$ satisfy $e_n^2 = \|f - f_n\|^2 \leq \frac{c}{n}$.

First check that there exists $g_1 \in G$ such that $f_1 = g_1$ satisfies $e_1^2 = \|f - f_1\|^2 \leq c$. As $\sum_{j=1}^m a_j \|f - h_j\|^2 = \|f\|^2 - 2f \cdot \sum_{j=1}^m a_j h_j + \sum_{j=1}^m a_j \|h_j\|^2 \leq s_G^2 - \|f\|^2 = c$, there must exist at least one $j \in \{1, \ldots, m\}$ for which $\|f - h_j\|^2 \leq c$ and we set $g_1 = h_j$.

Assuming that we already have $g_1, \ldots, g_n$, we express e_{n+1}^2 in terms of e_n^2 as $e_{n+1}^2 = \|f - f_{n+1}\|^2 = \|\frac{n}{n+1}(f - f_n) + \frac{1}{n+1}(f - g_{n+1})\|^2 = \frac{n^2}{(n+1)^2} e_n^2 + \frac{2n}{(n+1)^2}(f - f_n) \cdot (f - g_{n+1}) + \frac{1}{(n+1)^2}\|f - g_{n+1}\|^2$.

Similarly as in the first step, we consider convex combination of the last two terms from the formula expressing e_{n+1}^2 in terms of e_n^2: $\sum_{j=1}^m a_j \left(\frac{2n}{(n+1)^2}(f - f_n) \cdot (f - h_j) + \frac{1}{(n+1)^2}\|f - h_j\|^2\right)$ $= \frac{2n}{(n+1)^2}(f - f_n) \cdot (f - \sum_{j=1}^m a_j h_j) + \frac{1}{(n+1)^2}\left(\|f\|^2 - 2f \cdot (\sum_{j=1}^m a_j h_j) + \sum_{j=1}^m a_j \|h_j\|^2\right) = \frac{1}{(n+1)^2}$ $(\sum_{j=1}^m a_j g_j - \|f\|^2) \leq \frac{1}{(n+1)^2}(s_G^2 - \|f\|^2) = \frac{c}{(n+1)^2}$.
Thus there must exist some $j \in \{1, \ldots, m\}$ such that $\frac{2n}{(n+1)^2}(f - f_n) \cdot (f - g_{n+1}) + \frac{1}{(n+1)^2}\|f - g_{n+1}\|^2 \leq \frac{c}{(n+1)^2}$. Setting $g_j = h_j$, we get $e_{n+1}^2 \leq \frac{n^2}{(n+1)^2} e_n^2 + \frac{c}{(n+1)^2}$.

It can be easily verified by induction that this recursive formula together with $e_1^2 \leq c$ gives $e_n^2 \leq \frac{c}{n}$. □

The same upper bound as in Theorem 1 has been derived by Maurey using a probabilistic argument (see [44], [4]) in which a representation of a function f as an element of the set $conv\, G$, $f = \sum_{j=1}^m a_j h_i$, defines a finite probability distribution P

on the set G by setting $P(g = h_j) = a_j$. Then for $f_n = \sum_{i=1}^n \frac{g_i}{n}$, a random variable corresponding to the barycenter of an n-tuple of elements of G chosen randomly with respect to the probability P, the mean value of $\|f - f_n\|^2$ is bounded from above by $\frac{s_G^2 - \|f\|^2}{n}$. Hence there must exist at least one n-tuple $(g_1, \ldots, g_n)$ of elements of G for which $\|f - \sum_{i=1}^n \frac{g_i}{n}\| \leq \sqrt{\frac{s_G^2 - \|f\|^2}{n}}$.

Note that both these arguments prove that rates bounded from above by $\frac{s_G^2 - \|f\|^2}{n}$ can be even achieved when merely barycenters of n-tuples of elements of G are used as approximators. More precisely, they show that $\|f - bar_n G\| \leq \sqrt{\frac{s_G^2 - \|f\|^2}{n}}$, where $bar_n G = \{f \in X : f = \sum_{i=1}^n \frac{g_i}{n}, (g_1, \ldots, g_n) \in G^n\}$.

Moreover, as a byproduct, the above constructive proof of Theorem 1 gives bounds on rates of convergence of incremental learning algorithms (i.e., algorithms that in each step add a new hidden unit) [30].

MJB theorem was extended to $\mathcal{L}_p$-spaces, $p \in (1, \infty]$, in [10] and [12] with a similar recursive argument as above with peak functionals replacing inner products. The following theorem is a slight reformulation of an estimate from [10].

Theorem 2 *Let G be a bounded subset of $\mathcal{L}_p([0,1]^d)$, $p \in (1, \infty)$, $f \in cl\,conv\,G$ and $r > 0$ be such that $G \subseteq B_r(f, \|.\|)$. Then for every positive integer n, $\|f - span_n G\|_p \leq \frac{2^{1/\overline{p}} r}{n^{1/\overline{q}}}$, where $q = p/(p-1)$, $\overline{p} = \min(p, q)$, $\overline{q} = \max(p, q)$.*

The probabilistic argument by Maurey was modified by various authors to obtain extensions of MJB theorem to spaces of continuous functions with the supremum norm [3], [17], [20], [40], [38]. A more sophisticated probabilistic argument in [40] even allows to derive a tight improvement of MJB theorem for G formed by functions computable by sigmoidal perceptrons. This tightness result was extended to sets G with "proper" behaviour of covering numbers in [36].

For orthonormal bases, a similar estimate as MJB bound was derived using a simpler proof technique based on a rearrangement of a basis [41] and its tightness was investigated in [38] and [34].

4.4 Variation with respect to a Set of Functions

Since $conv_n G \subseteq span_n G$, MJB bound also applies to rates of approximation by $span_n G$. Moreover for any $c > 0$, $span_n G$ contains $conv_n G(c)$, where $G(c) = \{wg : g \in G, |w| \leq c\}$. Thus we obtain an estimate for any element of $\cup_{c \in \mathcal{R}_+} cl\,conv\,G(c)$. This approach can be mathematically formulated in terms of a norm tailored to a set G.

Let $(X, \|.\|)$ be a normed linear space and G be its subset, then *G-variation* (variation with respect to G) is defined as the Minkowski functional of the set $cl\,conv(G \cup -G)$, i.e.,

$$\|f\|_G = \inf\{c > 0 : f/c \in cl\,conv(G \cup -G)\}.$$

G-variation is a norm on the subspace $\{f \in X : \|f\|_G < \infty\} \subseteq X$. Next proposition states its basic properties [35].

Proposition 3 *Let $(X, \|.\|)$ be a normed linear space, G and F be its subsets, and $s_G = \sup_{g\in G} \|g\|$. Then*
(i) for all $f \in X$, $\|f\| \leq s_G \|f\|_G$;
(ii) if G is finite with $\mathrm{card}\, G = m$ and $f \in \mathrm{span}\, G$, then $\|f\|_G = \min\Big\{\sum_{i=1}^m |w_i| : f = \sum_{i=1}^m w_i g_i, g_i \in G, w_i \in \mathcal{R}\Big\}$;
(iii) $\|.\|_G \leq c\|.\|_F$ if and only if for all $h \in F$, $\|h\|_G \leq c$.

Variation with respect to a set of functions was introduced in [29] as an extension of the concept of variation with respect to half-spaces (H_d-variation) defined in [3]. For functions of one variable, H_1-variation coincides, up to a constant, with the notion of total variation studied in the integration theory, which inspired the term "variation."

Moreover in $\mathcal{L}_p$-spaces, H_d-variation is equal to $P_d(\sigma)$-variation for any sigmoidal activation function σ [33] and thus for application of MJB bound to perceptron networks, it is sufficient to study H_d-variation.

Besides being a generalization of the concept of total variation, G-variation also generalizes the notion of l_1-norm. Let G be an orthonormal basis of a separable Hilbert space $(X, \|.\|)$, then the *l_1-norm with respect to G* of $f \in X$ is defined as $\|f\|_{1,G} = \sum_{g\in G} |f \cdot g|$. It was shown in [34] that for G an orthonormal basis of a separable Hilbert space, l_1-norm with respect to G is equal to G-variation.

The following corollary from [29] and [35] is a reformulation of MJB theorem in terms of worst-case errors formalized by the concept of *deviation* $\delta(B, Y) = \sup_{f\in B} \|f - Y\|$.

Corollary 1 *Let G be a bounded subset of a Hilbert space $(X, \|.\|)$, $s_G = \sup_{g\in G} \|g\|$, $b > 0$, $0 \leq r \leq s_G b$ and n be a positive integer. Then*
(i) for every $f \in X$, $\|f - span_n G\| \leq \sqrt{\frac{(s_G\|f\|_G)^2 - \|f\|^2}{n}}$;
(ii) $\delta(\{f \in B_b(\|.\|_G) : \|f\| \geq r\}, span_n G) \leq \sqrt{\frac{(s_G b)^2 - r^2}{n}}$;
(iii) $\delta(B_b(\|.\|_G), span_n G) \leq \frac{s_G b}{\sqrt{n}}$.

This corollary shows how to cope with the curse of dimensionality in variable-basis approximation. If $\{G_d \subset \mathcal{L}_2([0,1]^d) : d \in \mathcal{N}_+\}$ is a sequence of sets of functions of d variables, we can keep rate of approximation by $span_n G_d$ within $r/\sqrt{n}$ for all d by restricting approximation to functions from balls $B_r(\|.\|_{G_d})$ of a fixed radius r in G_d-variation.

In $\mathcal{L}_p$-spaces, G-variation plays a similar role as it does in Hilbert spaces. The following estimate is a corollary of Theorem 2. By $\|.\|_{G,p}$ is denoted G-variation in $\mathcal{L}_p$-space (its unit ball is $cl_p(conv(G \cup -G))$, where cl_p denotes the closure in $\mathcal{L}_p$-space).

Corollary 2 *Let G be a bounded subset of $\mathcal{L}_p([0,1]^d)$, $p \in (1, \infty)$, $q = p/(p-1)$, $\overline{p} = \min(p, q)$, $\overline{q} = \max(p, q)$, $s_G = \sup_{g\in G} \|g\|_p$ and n be a positive integer. Then*
(i) for every $f \in X$, $\|f - span_n G\|_p \leq \frac{2^{\frac{1}{\overline{p}}+1} s_G \|f\|_{G,p}}{n^{1/\overline{q}}}$;
(ii) $\delta(B_1(\|.\|_{G,p}), span_n G) \leq \frac{2^{\frac{1}{\overline{p}}+1} s_G \|f\|_{G,p}}{n^{1/\overline{q}}}$.

PROOF. (i) Set $A = G(\|f\|_{G,p}) = \{w\,g : g \in G, |w| \leq \|f\|_{G,p}\}$ and $r = 2\,s_G\,\|f\|_{G,p}$. By Proposition 3 (i), for every $h \in A$ and every $f \in X$, $\|h - f\|_p \leq \|h\|_p + \|f\|_p \leq s_G\,\|f\|_{G,p} + s_G\,\|f\|_{G,p} = r$.

Hence $A \subseteq B_r(f, \|.\|)$ and so by Theorem 2 for every $f \in X$ and every positive integer n, there exists $f_n \in conv_n\,A \subseteq span_n\,G$ such that $\|f - f_n\| \leq \frac{2^{\frac{1}{p}+1} s_G\,\|f\|_{G,p}}{n^{1/q}}$.

(ii) follows directly from (i). □

4.5 Rates of Approximate Optimization over Variable Basis Functions

Estimates of rates of approximation in terms of G-variation can be applied to learning from data modelled as approximate minimization of regularized empirical error functionals. Learning from empirical data, which are given by a finite set of input/output pairs $\{(x_i, y_i) \in \mathcal{R}^d \times \mathcal{R}, i = 1, \ldots, m\}$ can be formalized as a minimization of the *empirical error* (*empirical risk*) functional defined as $\mathcal{E}(f) = \frac{1}{m}\sum_{i=1}^{m} |f(x_i) - y_i|^2$. A minimization is usually performed over a parametrized set of functions implemented in a suitable device such as a neural network. Typically, such an optimization problem is ill-posed and thus it requires regularization. *Tychonov regularization* replaces the functional $\mathcal{E}$ with $\mathcal{E}_{\gamma,\Psi} = \mathcal{E} + \gamma\,\Psi$, where Ψ is a functional called *stabilizer* and γ is a *regularization parameter* (see, e.g., [49], [7], [51]).

A common class of stabilizers is formed by functionals of the form $\|.\|_K^2$, where $\|.\|_K$ is the norm on a reproducing kernel Hilbert space (RKHS). A RKHS $\mathcal{H}_K(\Omega)$ is a Hilbert space of functions on a set Ω such that for every $x \in \Omega$, the evaluation functional $\mathcal{F}_x$, defined for any $f \in \mathcal{H}_K(\Omega)$ as $\mathcal{F}_x(f) = f(x)$, is bounded.

Recall that a mapping $K : \Omega \times \Omega \to \mathcal{R}$ is *positive semidefinite* on Ω if for all positive integers m, all $(w_1, \ldots, w_m) \in \mathcal{R}^m$ and all $(x_1, \ldots, x_m) \in \Omega^m$, we have $\sum_{i,j=1}^{m} w_i\,w_j\,K(x_i, x_j) \geq 0$. To every RKHS one can associate a unique symmetric, positive semidefinite mapping $K : \Omega \times \Omega \to \mathcal{R}$, called *kernel*, such that for every $f \in \mathcal{H}_K$, $f(x) = \langle f, K(x, .)\rangle_K$ (see, e.g., [1], [6]).

A kernel $K : \Omega \times \Omega \to \mathcal{R}$ is called *Mercer kernel* if Ω is compact and K is symmetric, continuous and positive semidefinite. For Ω compact and K a Mercer kernel, the space $\mathcal{H}_K(\Omega)$ is contained in the space $\mathcal{C}(\Omega)$ of continuous functions on Ω and for every $f \in \mathcal{H}_K(\Omega)$, $\|f\|_K \leq \sqrt{c_K}\|f\|_{\mathcal{C}}$, where $\|.\|_{\mathcal{C}}$ denotes the supremum norm on $\mathcal{C}(\Omega)$ and $c_K = \sup_{x,y\in\Omega} |K(x, y)|$ [9, p.33]. A kernel K is called *convolution* or *translation invariant* if there exists a function $k : \Omega \to \mathcal{R}$ such that $K(x, y) = k(x - y)$.

With $\|.\|_K^2$ as a stabilizer the regularized functional obtained from $\mathcal{E}$ has the form

$$\mathcal{E}_{\gamma,K}(f) = \frac{1}{m}\sum_{i=1}^{m} |f(x_i) - y_i|^2 + \gamma\|f\|_K^2.$$

$\mathcal{E}_{\gamma,K}$ is continuous and strictly uniformly convex [37] and thus it has unique argminimum over any convex closed set [50]. Next theorem, which is a version of the Representer Theorem from [9, p.42], describes the unique argminimum of the problem $(\mathcal{H}_K(\Omega), \mathcal{E}_{\gamma,K})$.

Theorem 3 *Let $\Omega \subset \mathcal{R}^d$ be compact, $K : \Omega \times \Omega \to \mathcal{R}$ be a Mercer kernel, $\mathcal{H}_K(\Omega)$ be the RKHS defined by K, $x = (x_i, \ldots, x_m) \in \Omega^m$, $y = (y_m, \ldots, y_m) \in \mathcal{R}^m$, $\mathcal{E}(f) = \frac{1}{m}\sum_{i=1}^m |f(x_i) - y_i|^2$, and $\gamma > 0$. Then there exists a unique argminimum g^o of the problem $(\mathcal{H}_K(\Omega), \mathcal{E}_{\gamma,K})$ such that $g^o(x) = \sum_{i=1}^m a_i K(x, x_i)$, where $a = (a_1, \ldots, a_m)$ is the unique solution of the well-posed linear system $(\mathcal{K}(x) + \gamma\, m\, I)a = y$, I is the $m \times m$ identity matrix, and $\mathcal{K}(x)$ is the $m \times m$ matrix defined as $\mathcal{K}(x)_{i,j} = K(x_i, x_j)$.*

Notice that the unique argminimum described in this theorem is of the form of a variable-basis function from $span_m G_K$, where $G_K = \{K(x, \cdot) : x \in \Omega\}$. In particular, for the convolution kernel defined by the Gaussian function, the argminimum can be computed by a Gaussian radial-basis function network with m hidden units.

It has been argued in [18] that "the regularization principles lead to approximation schemes that are equivalent to networks with one layer of hidden units". Indeed, various versions of the Representer Theorem (Theorem 3 and an analogous theorem from [18]) show that for suitable stabilizers the unique function minimizing the regularized empirical error is of the form of a one-hidden layer network with a linear output and hidden units computing functions corresponding to the type of the stabilizer.

However, for large m such network might not be implementable. Next theorem allows us to estimate quality of an approximate solution of the regularized learning problem achievable using networks with a fixed number n of hidden units. It was derived in [37] from a more general theorem on approximate minimization of continuous functionals over sets of the form $M \cap span_n G$ with M convex. Estimates derived in [37] take advantage of MJB theorem combined with Lipschitz continuity of certain Minkowski functionals. Application of these estimates to kernel stabilizers utilizes the fact that for every norm $\|.\|$ on a Hilbert space, the functional $\|.\|^2$ is strictly uniformly convex with the modulus of convexity t^2 and that the functional $\mathcal{E}_{\gamma,K}$ is continuous with quadratic modulus of continuity [37].

Theorem 4 *Let $\Omega \subset \mathcal{R}^d$ be compact, $K : \Omega \times \Omega \to \mathcal{R}$ be a Mercer kernel, $s_K = \sup_{x\in\Omega}\sqrt{K(x,x)}$, $(\mathcal{H}_K(\Omega), \|.\|_K)$ be the RKHS defined by K, $\mathcal{E}(f) = \frac{1}{m}\sum_{i=1}^m |f(x_i) - y_i|^2$, where $(x_1, \ldots, x_m) \in \Omega^m$, $(y_1, \ldots, y_m) \in \mathcal{R}^m$, $\gamma > 0$, $g^o(x) = \sum_1^m a_i K(x, x_i)$ be the unique argminimum of $(\mathcal{H}_K(\Omega), \mathcal{E}_{\gamma,K})$ given by the Representer Theorem, $\{\varepsilon_n\}$ be a sequence of positive reals such that $\lim_{n-\infty} \varepsilon_n = 0$ and $\{g_n\}$ be a sequence of ε_n-argminima of problems $(span_n G_K, \mathcal{E}_{\gamma,K})$. Then for every positive integer n the following hold:*

(i) $\inf_{g\in span_n G_K} \mathcal{E}_{\gamma,K}(g) - \mathcal{E}_{\gamma,K}(g^o) \le \alpha\left(\sqrt{\frac{(s_K\|g^o\|_{G_K})^2 - \|g^o\|_K^2}{n}}\right)$;

(ii) if $\|g^o\|_G < \infty$, *then* $\{g_n\}$ *is an* $\mathcal{E}_{\gamma,K}$*-minimizing sequence over* $\mathcal{H}_K(\Omega)$ *and* $\mathcal{E}_{\gamma,K}(g_n) - \mathcal{E}_{\gamma,K}(g^o) \le \alpha\left(\sqrt{\frac{(s_K\|g^o\|_{G_K})^2 - \|g^o\|_K^2}{n}}\right) + \varepsilon_n$;

(iii) $\|g_n - g^o\|_K^2 \le \alpha\left(\sqrt{\frac{(s_K\|g^o\|_{G_K})^2 - \|g^o\|_K^2}{n}}\right) + \varepsilon_n$;

(iv) $\|g_n - g^o\|_C^2 \le \sqrt{c_K}\left(\alpha\left(\sqrt{\frac{(s_K\|g^o\|_{G_K})^2 - \|g^o\|_K^2}{n}}\right) + \varepsilon_n\right)$;

where $c_K = \sup_{x,y\in\Omega}|K(x,y)|$, $\alpha(t) = a_2 t^2 + a_1 t$, $a_1 = 2\big(m\|g^o\|_K c_K + mb\sqrt{c_K} + \gamma\|g^o\|_K\big)$, $a_2 = m\,c_K + \gamma$ *and* $b = \max\{|y_i| : i = 1, \ldots, m\}$.

Recall that a method of solving an optimization problem (M, Φ) is called *direct* if it constructs a Φ-minimizing sequence $\{g_i\} \subseteq M$ such that $\{g_i\}$ converges to some g in M and $\lim_{i\to\infty} \Phi(g_i) = \Phi(g)$ [19]. So for any Mercer kernel K, any algorithm constructing a sequence $\{g_n\}$ of ε_n-argminima of $\mathcal{E}_{\gamma,K}$ over $span_n G_K$ is a direct method of approximate solution of the problem of minimization of $\mathcal{E}_{\gamma,K}$ over $\mathcal{H}_K(\Omega)$. Moreover, Theorem 4 estimates speed of convergence of this sequence to the global argminimum g^o and of $\mathcal{E}_{\gamma,K}(g_n)$ to $\mathcal{E}_{\gamma,K}(g^o)$ in terms of G-variation and $\|.\|_K$-norm of the global argminimum g^o.

4.6 Comparison with Linear Approximation

Variation with respect to a set of functions plays an analogous role in variable-basis approximation as Sobolev norms do in linear approximation, where one can achieve rates of the order of $\mathcal{O}(n^{-1})$ by restricting approximation of d-variables functions only to functions from balls in Sobolev norms of degree $s = d$.

More precisely, for Ω an open subset of $\mathcal{R}^d$, $p \in [1, \infty)$, the Sobolev space $W_{s,p}(\Omega)$ is the subspace of $\mathcal{L}_p(\Omega)$ formed by all functions $f : \Omega \to \mathcal{R}$ such that for every multi-index $\alpha = (\alpha_1, \ldots, \alpha_d)$ satisfying $0 \leq |\alpha| \leq s$, $D^\alpha f \in \mathcal{L}_p(\Omega)$, where $D^\alpha = D_1^{\alpha_1} \ldots D_d^{\alpha_d}$ is a partial derivative in weak (distributional) sense (see [2, p.44]). The norm of $f \in W_{s,p}(\Omega)$ is defined as $\|f\|_{s,p}^d = (\sum_{0\leq|\alpha|\leq s} \|D^\alpha f\|_p^p)^{1/p}$.

The worst-case error in approximation of the unit ball in $W_{s,p}(\Omega)$ with $[0,1]^d \subset \Omega$ by an n-dimensional linear subspace X_n of $\mathcal{L}_p(\Omega)$ is of the order of $\mathcal{O}(n^{-s/d})$ [42]. More precisely, the Kolmogorov's n-width $d_n(B_1(\|.\|_{s,p}^d) = \inf_{X_n} \delta(B_1(\|.\|_{s,p}^d), X_n) \sim \mathcal{O}(n^{-s/d})$, where the infimum is taken over all n-dimensional subspaces of $\mathcal{L}_p(\Omega)$. So if we are free to choose for each n an "optimal" n-dimensional subspace, then we can approximate all functions from Sobolev balls of the order $s \geq d$ within the accuracy $\mathcal{O}(n^{-1})$.

With d increasing, balls in Sobolev norms of the order d are "shrinking" as some functions in the unit ball in the Sobolev norm $\|.\|_{d,p}^d$ with "large" $d+1$-th derivatives cannot be extended to functions of $d+1$ variables from the unit ball in the Sobolev norm $\|.\|_{d+1,p}^{d+1}$. In contrast to balls in Sobolev norms of degree d, balls in G_d-variation need not to be shrinking with d increasing.

Proposition 4 *Let d be a positive integer, $p \in (1, \infty)$, G_d and G_{d+1} be bounded subsets of $\mathcal{L}_p([0,1]^d)$ and $\mathcal{L}_p([0,1]^{d+1})$, resp., such that for every $g \in G_d$, $\overline{g} : [0,1]^{d+1} \to \mathcal{R}$ defined as $\overline{g}(x_1, \ldots, x_{d+1}) = g(x_1, \ldots, x_d)$ belongs to G_{d+1}. Then there exists an embedding $\nu_d : B_1(\|.\|_{G_d}) \to B_1(\|.\|_{G_{d+1}})$.*

PROOF. Define ν_d on $B_1(\|.\|_{G_d}) = cl\, conv(G_d \cup -G_d)$ as $\nu_d(g) = \overline{g}$, where $\overline{g}(x_1, \ldots, x_{d+1}) = g(x_1, \ldots, x_d)$. As ν_d is an embedding on G_d, it is an embedding on $cl\, conv(G_d \cup -G_d)$, too. □

It is easy to check that for any activation or radial function ψ, balls in variation with respect to perceptrons, $P_d(\psi)$-variation, and RBF, $F_d(\psi)$-variation, satisfy the assumptions of Proposition 4. So in neural network approximation, there exist families of sets of functions of d-variables approximable with rates $n^{-1/2}$, which are not shrinking with d increasing.

Sets of functions computable by networks with n sigmoidal perceptrons are much larger than a single n-dimensional subspace (they are formed by the union of all n-dimensional subspaces spanned by n-tuples of elements of G). However upper bounds on linear approximation cannot be automatically extended to such unions of n-dimensional subspaces as they may not contain "optimal" n-dimensional subspaces for linear approximation of balls in Sobolev norms.

Nevertheless, for a sufficiently large radius (depending on both d and s), a ball in variation with respect to half-spaces (H_d-variation) contains the unit ball in the Sobolev norm of the order s. For a bounded open set Ω such that $[0,1]^d \subset \Omega$ and $s > d/2 + 1$, $B_r(\|.\|^d_{s,2})$ is contained in $B_{2rb_s}(\|.\|_{H_d(\Omega)})$, where $b_s = \left(\int_{\mathcal{R}^d}(1 + \|y\|_2^{2(s-1)})^{-1}\,dy\right)^{1/2}$ is finite for $2(s-1) > d$, [4, pp. 935, 941]. So for $s > d/2 + 1$, the unit balls in Sobolev norms $\|.\|^d_{s,2}$ are contained in balls in H_d-variation of the radius $2b_s$.

By an improvement of MJB theorem for $G = H_d$ from [40], worst-case errors in approximation of such balls by neural networks with n sigmoidal perceptrons are bounded from above by $b_s n^{-(1/2+1/d)}$. On the other hand, the worst-case errors in approximation by optimal n-dimensional linear subspaces of such Sobolev balls (Kolmogorov's n-widths) are of the order of $\mathcal{O}(n^{-s/d})$, which for $s > d/2+1$ is bounded from above by $\mathcal{O}(n^{-(1/2+1/d)})$. These two results give upper bounds of the same order for linear and neural network approximation. However, only the bound for the linear case is tight, while the one for neural networks was obtained using a rather rough estimate based on embeddings of Sobolev balls into balls in H_d-variation.

Better estimates for neural networks than those for the linear case were obtained in [4] and [35] for balls in variation with respect to sigmoidal or periodic activations. For such balls, Kolmogorov's n-width is larger than upper bounds on worst-case errors following from MJB theorem.

In addition to comparing estimates on rates of linear and variable-basis approximation, we can also compare properties of projection operators in these two types of approximation. In the linear case, best approximation exists and is unique and thus the metric projection operator is always defined, single-valued, continuous, and homogeneous. However, the geometrical properties of variable basis approximating sets (nonconvexity) imply that the metric projection operators are set-valued mappings with no continuous "selection". Thus it is not possible to choose in a continuous way a best approximation or even a nearly best approximation in $span_n G$ for various sets G corresponding to neural networks [24], [25], [27]. So lower bounds exhibiting the curse of dimensionality derived using topological methods [11] (which can be applied to linear approximation due to continuity of projections) cannot be used in the case of neural networks, which opens a possibility of better rates of approximation than those achievable using linear methods.

4.7 Upper Bounds on Variation

Besides being a generalization of the concept of total variation, G-variation also extends the notion of l_1-norm: for G a countable orthonormal basis of X, $\|f\|_G = \|f\|_{G,1} = \sum_{g\in G} |f \cdot g|$ [35]. Thus for G countable orthonormal, the unit ball in G-variation is equal to the unit ball in l_1-norm. Even for some nonorthonormal G, properties of balls

in G-variation can be investigated using images of balls in l_1 or $\mathcal{L}_1$-norms under certain linear operators.

The following proposition shows that when G is finite, balls in G-variation are images of balls in $l_1(\mathcal{R}^m)$, where $m = card\, G$. By $\|.\|_1$ is denoted l_1-norm.

Proposition 5 *Let $G = \{g_1, \ldots, g_m\}$ be a subset of a normed linear space $(X, \|.\|)$ and $T : \mathcal{R}^m \to X$ be a linear operator defined for every $w = (w_1, \ldots, w_m) \in \mathcal{R}^m$ as $T(w) = \sum_{i=1}^m w_i g_i$. Then for every $r > 0$, $B_r(\|.\|_G) = T(B_r(\|.\|_1))$.*

PROOF. By Proposition 3 (ii), $\|f\|_G = \min\{\|w\|_1 : w \in \mathcal{R}^m, f = T(w)\}$. Thus $T(B_r(\|.\|_1)) \subseteq B_r(\|.\|_G)$. To show the opposite inclusion, consider $f \in B_r(\|.\|_G)$. By the definition of G-variation, $f = \lim_{j\to\infty} f_j$, where $f_j = \sum_{i=1}^m w_{ji} g_i$ and $\|w_j\|_1 = \sum_{i=1}^m |w_{ji}| \leq r$. Hence there exist $w = (w_1, \ldots, w_m) \in \mathcal{R}^m$ such that for all $i = 1, \ldots, m$, the sequences $\{w_{ji}\}$ converge to w_i subsequentially. So the sequence $\{f_j\}$ converges subsequentially to $f' = \sum_{i=1}^m w_i g_i$. Since every norm-induced topology is Hausdorff, $f = f'$ and so $f = T(w)$. □

Even when G is infinite, G-variation can be estimated in terms of $\mathcal{L}_1$-norm. For $\phi \in \mathcal{L}_2(A \times \Omega)$, where $A \subset \mathcal{R}^q$, $\Omega \subset \mathcal{R}^d$ and $w \in \mathcal{L}_2(A)$ define $T_\phi(w) = \int_A w(a)\phi(a, x)da$. Then $T_\phi : \mathcal{L}_2(A) \to \mathcal{L}_2(\Omega)$ is a bounded (continuous) linear operator [15, p.138]. When A, Ω are compact, T_ϕ is a compact operator and when $\phi \in \mathcal{C}(A \times \Omega)$, then $T_\phi : \mathcal{C}(A) \to \mathcal{C}(\Omega)$ [15, p.188].

Intuitively, any function in $T_\phi(\mathcal{L}_2(A))$ can be represented as a "neural network with a continuum of hidden units computing ϕ", $\int_A w(a)\phi(a, x)da$. Next theorem shows that G_ϕ-variation of such functions can be estimated from above by the $\mathcal{L}_1$-norm of the "output weight function" w (its proof is a modification of an argument used in [33] to derive a similar result for $\mathcal{C}(\Omega)$).

Theorem 5 *Let d, m be positive integers, $A \subset \mathcal{R}^m$ and $\Omega \subset \mathcal{R}^d$ be compact, $\phi \in \mathcal{L}_2(A \times \Omega)$ and $f \in \mathcal{L}_2(\Omega)$ be such that $f = T_\phi(w)$ for some $w \in \mathcal{L}_2(A)$. Then $\|f\|_{G_\phi} \leq \|w\|_{\mathcal{L}_1(A)} = \int_A |w(a)|da$.*

Integral representations of the form of a "neural network with a continuum of hidden units" were originally studied as a tool for proving the universal approximation property for various types of neural networks [8], [21]. In [4], Fourier representation was combined with MJB bound to obtain an upper bound on approximation by sigmoidal perceptrons via estimates for the cosine activation. For functions with compactly supported Fourier transforms and continuous partial derivatives of the order $s > d/2$, an integral representation of the form of a neural network with Gaussian RBF units was derived in [16].

Next theorem from [26] (see also [33]) gives an integral representation of the form of a Heaviside perceptron network. For $e \in S^{d-1}$ and $b \in \mathcal{R}$, we denote $H_{e,b} = \{x \in \mathcal{R}^d : e \cdot x + b = 0\}$. The half-spaces bounded by this hyperplane are denoted $H^+_{e,b} = \{x \in \mathcal{R}^d : e \cdot x + b \geq 0\}$ and $H^-_{e,b} = \{x \in \mathcal{R}^d : e \cdot x + b < 0\}$. By Δ is denoted the Laplacian operator $\Delta(h) = \sum_{i=1}^d \frac{\partial^2 h}{\partial x_i^2}$.

Theorem 6 *Let d be a positive integer and let $f : \mathcal{R}^d \to \mathcal{R}$ be compactly supported and $d+2$-times continuously differentiable. Then*

$$f(x) = \int_{S^{d-1}\times\mathcal{R}} w_f(e,b)\vartheta(e\cdot x+b)dedb,$$

where for d odd, $w_f(e,b) = a_d \int_{H^-_{e,b}} \triangle^{k_d} f(y)dy$, $k_d = \frac{d+1}{2}$, and a_d is a constant independent of f, while for d even, $w_f(e,b) = a_d \int_{H^-_{e,b}} \triangle^{k_d} f(y)\eta(e\cdot y+b)dy$, where $\eta(t) = -t\log|t| + t$ for $t \neq 0$ and $\eta(0) = 0$, $k_d = \frac{d+2}{2}$, and a_d is a constant independent of f.

4.8 Lower Bounds on Variation

The following theorem from [38] gives a geometric characterization of G-variation in a Hilbert space. By $G^\perp$ is denoted the orthogonal complement of G, i.e., $G^\perp = \{f \in X : (\forall g \in G)(f\cdot g = 0)\}$.

Theorem 7 *Let $(X, \|.\|)$ be a Hilbert space, G be its bounded non-empty subset and $f \in X$ be such that $\|f\|_G < \infty$. Then $\|f\|_G = \sup_{h\in X-G^\perp} \frac{f\cdot h}{\sup_{g\in G}|g\cdot h|}$.*

This theorem implies a lower bound on G-variation of the form $\|f\|_G \geq \frac{\|f\|^2}{\sup_{g\in G}|g\cdot f|}$. So functions that have small inner products with all elements of G (are "almost orthogonal" to G) have large G-variation.

This bound was used in [38] to show that certain Boolean functions have variations with respect to Heaviside perceptrons at least of the order of $\mathcal{O}(2^{d/6})$. The proof was based on properties of rather special Boolean functions called bent functions. They can be extended to smooth functions defined on $[0,1]^d$ with H_d-variation of the same order.

Another method of demonstrating existence of functions with large variations is based on comparison of cardinality of G with certain covering numbers. Define a binary relation ρ on the unit ball $S_1(\|.\|)$ of a Hilbert space $(X, \|.\|)$ by $\rho(f,g) = \arccos|f\cdot g|$. It is easy to see that ρ is a pseudometrics measuring distance as the minimum of the two angles between f and g and between f and $-g$ (it is a pseudometrics as the distance of antipodal vectors is zero). For a subset G of $S_1(\|.\|)$ define *extent* of G as

$$\alpha_G = \inf\{\alpha \in [0,\pi/2] : (\forall f \in S_1(\|.\|))(\exists g \in G)(\rho(f,g) \leq \alpha)\}.$$

Note that α_G is the infimum of all α for which G is an α-net in $S_1(\|.\|)$, i.e., balls of radius ε centered at elements of G cover $S_1(\|.\|)$. For G compact, we can replace infimum in the definition of α_G by minimum. This is the case of H_d, which is compact in $\mathcal{L}_p([0,1]^d)$ for any $p \in [1,\infty)$ and any positive integer d [20].

When α_G is small, G is "almost dense" in $S_1(\|.\|)$, while when α_G is large, G is "sparse" in $S_1(\|.\|)$. If α_G is close to $\frac{\pi}{2}$, then there exists an element in $S_1(\|.\|)$ which has a large "angular distance" from all elements of G (is almost orthogonal to G) and hence it has a large G-variation.

Metric entropy [28] measures "size" of sets in Banach spaces by the number of balls of a given radius needed to cover them. The size of the smallest α-net in $S_1(\|.\|)$ with the angular pseudometrics ρ is the *covering number* $cov_\alpha(S_1(\|.\|), \rho)$. When for some α close to $\pi/2$, the cardinality of G is smaller than α-covering number of $S_1(\|.\|)$, then there exists a function in $S_1(\|.\|)$ with "large" G-variation.

Proposition 6 *Let G be a subset of the unit sphere $S_1(\|.\|)$ in a Hilbert space $(X, \|.\|)$ and $\alpha \in [0, \pi/2]$ be such that $card\, G < cov_\alpha(S_1(\|.\|), \rho)$. Then there exists $f \in S_1(\|.\|)$, for which $\|f\|_G \geq 1/\cos\alpha$.*

PROOF. If $card\, G < \cos\alpha$, then $\alpha_G \leq \alpha$. So there exists $f \in S_1(\|.\|)$ such that for all $g \in G$, $\rho(f, g) \geq \alpha$ and hence $|f \cdot g| \leq \cos\alpha$. So by Theorem 7, $\|f\|_G \geq 1/\cos\alpha$. □

Proposition 6 can be applied to the space of real-valued Boolean functions $\mathcal{B}(\{0,1\}^d)$, which is studied in the next section. $\mathcal{B}(\{0,1\}^d)$ is isomorphic to $\mathcal{R}^{2^d}$ and for fixed α, α-covering numbers of S^{m-1} with the angular pseudometrics ρ grow asymptotically exponentially with the dimension m [23]. Thus Proposition 6 can be applied to show that in any subset G of $\mathcal{B}(\{0,1\}^d)$ of cardinality depending on 2^d only polynomially there exists a function with "large" G-variation.

4.9 Rates of Approximation of Real-valued Boolean Functions

One of the simplest cases where application of MJB theorem gives description of interesting sets of functions that can be approximated by neural networks without the curse of dimensionality is the space of real-valued Boolean function. This space, denoted by $\mathcal{B}(\{0,1\}^d)$ and endowed with the inner product defined for $f, g \in \mathcal{B}(\{0,1\}^d)$, as $f \cdot g = \sum_{x \in \{0,1\}^d} f(x)g(x)$, is a finite-dimensional Hilbert space isomorphic to the 2^d-dimensional Euclidean space $\mathcal{R}^{2^d}$ with the l_2-norm.

Let $\overline{H}_d$ denotes the set of functions on $\{0,1\}^d$ computable by signum perceptrons, i.e., $\overline{H}_d = \{f : \{0,1\}^d \to \mathcal{R} : f(\mathbf{x}) = \mathrm{sgn}(\mathbf{v} \cdot \mathbf{x} + b), \mathbf{v} \in \mathcal{R}^d, b \in \mathcal{R}\}$. From technical reasons we consider perceptrons with the signum (bipolar) activation function, defined as $\mathrm{sgn}(t) = -1$ for $t < 0$ and $\mathrm{sgn}(t) = 1$ for $t \geq 0$, instead of the more common Heaviside function that assigns zero to negative numbers.

For an orthonormal basis G, G-variation can be more easily estimated because it is equivalent to l_1-norm with respect to G [35]. Expressing elements of such a basis as a linear combination of signum perceptrons, we can obtain description of subsets of $\mathcal{B}(\{0,1\}^d)$ which can be approximated by perceptron networks without the curse of dimensionality.

We use two orthonormal bases. The first one is the *Euclidean orthonormal basis* defined as $E_d = \{e_u : u \in \{0,1\}^d\}$, where $e_u(u) = 1$ and for every $x \in \{0,1\}^d$ with $x \neq u$, $e_u(x) = 0$. The second one is the *Fourier orthonormal basis* (see, e.g., [52]) defined as $F_d = \left\{\frac{1}{\sqrt{2^d}}(-1)^{u \cdot x} : u \in \{0,1\}^d\right\}$. Every $f \in \mathcal{B}(\{0,1\}^d)$ can be represented as $f(x) = \frac{1}{\sqrt{2^d}} \sum_{u \in \{0,1\}^d} \widehat{f}(u)(-1)^{u \cdot x}$, where $\widehat{f}(u) = \frac{1}{\sqrt{2^d}} \sum_{x \in \{0,1\}^d} f(x)(-1)^{u \cdot x}$. For a subset $I \subset \{0,1\}^d$, I-parity is defined by $p_I(u) = 1$ if $\sum_{i \in I} u_i$ is odd, and $p_I(u) = 0$

otherwise. If we interpret the output 1 as -1 and 0 as 1, then the elements of the Fourier basis F_d correspond to the generalized parity functions. As F_d and E_d are orthonormal, for all $f \in \mathcal{B}(\{0,1\}^d)$, $\|f\|_{1,E_d} = \|f\|_{E_d}$ and $\|\widetilde{f}\|_1 = \|f\|_{1,F_d} = \|f\|_{F_d}$ [34], [38].

It is easy to see that $span_n F_d \subseteq span_{dn+1}\overline{H}_d$ (as $f_u(x) = -1^{u\cdot x} = \frac{1+(-1)^d}{2} + \sum_{j=1}^{d}(-1)^j \mathrm{sgn}(u\cdot x - j + \frac{1}{2})$) and that $span_{n+1}E_d \subseteq span_n \overline{H}_d$ (as $e_u(x) = \frac{\mathrm{sgn}(v\cdot x+b)+1}{2}$ for appropriate v and b). Thus by MJB theorem, $\|f - span_{dn+1}\overline{H}_d\| \leq \sqrt{\frac{\|\widetilde{f}\|_1^2 - \|f\|^2}{n}}$ and $\|f - span_{n+1}\overline{H}_d\| \leq \sqrt{\frac{\|f\|_{1,E_d}^2 - \|f\|^2}{n}}$. So "fast" rates of approximation are guaranteed for functions with either "small" variation with respect to signum perceptrons, "small" spectral norm, or "small" norm with respect to the Euclidean basis.

More interesting classes of functions that can be approximated quite well by networks with a moderate number of Boolean signum perceptrons, are functions with a small number of nonzero Fourier coefficients. The following estimate can be obtained from the embedding of $span_n F_d$ into $span_{nd+1}H_d$ combined with the Cauchy-Schwartz inequality [38].

Proposition 7 *Let d, n, and m be positive integers, $m \leq 2^d$, $c > 0$ and $f \in \mathcal{B}(\{0,1\}^d)$ be a function with at most m Fourier coefficients nonzero and with $\|f\| \leq c$. Then $\|f - span_{dn+1}\overline{H}_d\| \leq \frac{c}{2}\sqrt{\frac{m}{n}}$.*

Another example of functions that can be efficiently approximated by perceptron networks are functions representable by "small" decision trees [39]. A *decision tree* is a binary tree with labeled nodes and edges. The *size* of a decision tree is the number of its leaves. A function $f : \{0,1\}^d \to \mathcal{R}$ is representable by a decision tree if there exists a tree with internal nodes labeled by variables $x_1, \ldots, x_d$, all pairs of edges outgoing from a node labeled by 0s and 1s, and all leaves labeled by real numbers, such that f can be computed by this tree as follows. The computation starts at the root and after reaching an internal node labeled by x_i, it continues along the edge whose label coincides with the actual value of the variable x_i and finally when a leaf is reached, the value $f(x_1, \ldots, x_d)$ is equal to the label of this leaf. Next proposition follows from an estimate derived in [38].

Proposition 8 *Let d, s be positive integers, $b \geq 0$, $f \in \mathcal{B}(\{0,1\}^d)$ be representable by a decision tree of size s such that for all $x \in \{0,1\}^d$, $f(x) \neq 0$ and $\frac{\max_{x\in\{0,1\}^d}|f(x)|}{\min_{x\in\{0,1\}^d}|f(x)|}\|f\| \leq b$. Then $\|f - span_{dn+1}\overline{H}_d\| \leq \frac{sb}{2\sqrt{n}}$.*

Bibliography

[1] N. Aronszjan, Theory of reproducing kernels, *Transactions of AMS* **12** (1950) 15–27.

[2] R.A. Adams, *Sobolev Spaces*, New York: Academic Press (1975).

[3] A.R. Barron, Neural net approximation, In *Proceedings of the 7th Yale Workshop on Adaptive and Learning Systems* (1992) 69–72.

[4] A.R. Barron, Universal approximation bounds for superposition of a sigmoidal function, *IEEE Transactions on Information Theory* **39** (1993) 930–945.

[5] R. Bellman, *Dynamic Programming*, Princeton: Princeton University Press (1957).

[6] C. Berg, J.P.R. Christensen, P. Ressel, *Harmonic Analysis on Semigroups*, New-York: Springer-Verlag (1984).

[7] M. Bertero, Regularization methods for linear inverse problems, In *Inverse Problems* (Ed. Taylor, C. G.), Berlin: Springer-Verlag (1986).

[8] S.M. Carroll, B.W. Dickinson, Construction of neural nets using the Radon transform, In *Proceedings of IJCNN'89* New York: IEEE Press (1989) 607–611.

[9] F. Cucker, S. Smale, On the mathematical foundations of learning, *Bulletin of AMS* **39** (2002) 1–49.

[10] C. Darken, M. Donahue, L. Gurvits, E. Sontag, Rate of approximation results motivated by robust neural network learning, In *Proceedings of the 6th Annual ACM Conference on Computational Learning Theory*, New York: ACM (1993) 303–309.

[11] R. DeVore, R. Howard, C. Micchelli, Optimal nonlinear approximation, *Manuscripta Mathematica* **63** (1989) 469–478.

[12] M. Donahue, L. Gurvits, C. Darken, E. Sontag, Rates of convex approximation in non-Hilbert spaces, *Constructive Approximation* **13** (1997) 187–220.

[13] A.L. Dontchev, *Perturbations, Approximations and Sensitivity Analysis of Optimal Control Systems*, Lecture Notes in Control and Information Sciences, vol. 52. Berlin: Springer-Verlag (1983).

[14] A.L. Dontchev, T. Zolezzi, *Well-Posed Optimization Problems*, Lecture Notes in Mathematics, vol. 1543, Berlin: Springer-Verlag (1993).

[15] A. Friedman, *Foundations of Modern Analysis*, New York: Dover (1992).

[16] F. Girosi, G. Anzellotti, Rates of convergence for radial basis functions and neural networks, In *Artificial Neural Networks for Speech and Vision*, London: Chapman & Hall (1993) 97–113.

[17] F. Girosi, Approximation error bounds that use VC-bounds, In *Proceedings of ICANN'95*, Paris: EC2 & Cie (1995) 295–302.

[18] F. Girosi, M. Jones, T. Poggio, Regularization theory and neural architectures, *Neural Computation* **7** (1995) 219–269.

[19] I.M. Gelfand, S.V. Fomin, *Calculus of Variations*, Englewood Cliffs: Prentice Hall (1963).

[20] L. Gurvits, P. Koiran, Approximation and learning of convex superpositions, *Journal of Computer and System Sciences* **55** (1997) 161–170.

[21] Y. Ito, Representations of functions by superpositions of a step or sigmoid function and their applications to neural network theory, *Neural Networks* **4** (1991) 385–394.

[22] L.K. Jones, A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training, *Annals of Statistics* **20** (1992) 608–613.

[23] P.C. Kainen, V. Kůrková, Quasiorthogonal dimension of Euclidean spaces, *Applied Mathematics Letters* **6** (1993) 7–10.

[24] P.C. Kainen, V. Kůrková, A. Vogt, Approximation by neural networks is not continuous, *Neurocomputing* **29** (1999) 47–56.

[25] P.C. Kainen, V. Kůrková, A. Vogt, Best approximation by Heaviside perceptron networks, *Neural Networks* **13** (2000) 645–647.

[26] P.C. Kainen, V. Kůrková, A. Vogt, An integral formula for Heaviside neural networks, *Neural Network World* **10** (2000) 313–319.

[27] P.C. Kainen, V. Kůrková, A. Vogt, Continuity of approximation by neural networks in $\mathcal{L}_p$-spaces, *Annals of Operational Research* **101** (2001) 143–147.

[28] L.N. Kolmogorov, Asymptotic characteristics of some completely bounded metric spaces, *Doklady Akademii Nauk* **108** (1956) 385–389.

[29] V. Kůrková, Dimension-independent rates of approximation by neural networks, In *Computer-Intensive Methods in Control and Signal Processing: Curse of Dimensionality* (Eds. Warwick, K., Kárný, M.) Boston: Birkhauser (1997) 261–270.

[30] V. Kůrková, Incremental approximation by neural networks, In *Complexity: Neural Network Approach*, (Eds. K. Warwick, M. Kárný, V. Kůrková:) London: Springer-Verlag (1998) 177–188.

[31] V. Kůrková, Universality and complexity of approximation of multivariable functions by feedforward networks, In *Softcomputing and Industry - Recent Applications* (Eds. R. Roy, M. Koeppen, S. Ovaska, T. Furuhashi, F. Hoffmann) London: Springer-Verlag (2002) 13–24.

[32] V. Kůrková, Neural networks as universal approximators, In *The Handbook of Brain Theory and Neural Networks* II (Ed. M. A. Arbib) Cambridge: MIT Press (2002) 1180–1183.

[33] V. Kůrková, P.C. Kainen, V. Kreinovich, Estimates of the number of hidden units and variation with respect to half-spaces, *Neural Networks* **10** (1997) 1061–1068.

[34] V. Kůrková, M. Sanquinetti, Bounds on rates of variable-basis and neural network approximation, *IEEE Transactions on Information Theory* **47** (2001) 2659–2665.

[35] V. Kůrková, M. Sanquinetti, Comparison of worst-case errors in linear and neural network approximation, *IEEE Transactions on Information Theory* **48** (2002) 264–275.

[36] V. Kůrková, M. Sanquinetti, Tight bounds on rates of variable-basis approximation via estimates of covering numbers, Research Report ICS-02-865, Institute of Computer Science, Prague (2002)

[37] V. Kůrková, M. Sanguineti, Error estimates for approximate optimization over variable-basis functions, Research Report ICS-02-882, Institute of Computer Science, Prague (2002).

[38] V. Kůrková, P. Savický, K. Hlaváčková, Representations and rates of approximation of real-valued Boolean functions by neural networks, *Neural Networks* 11 (1998) 651–659.

[39] E. Kushilewicz, Y. Mansour, Learning decision trees using the Fourier spectrum, *SIAM Journal on Computing* **22** (1993) 1331–1348.

[40] Y. Makovoz, Random approximants and neural networks, *Journal of Approximation Theory* **85** (1996) 98–109.

[41] H.N. Mhaskar, C.A. Micchelli, Dimension-independent bounds on the degree of approximation by neural networks, *IBM Journal of Research and Development* **38** (1994) 277–283.

[42] A. Pinkus, *n-Width in Approximation Theory*, Berlin: Springer-Verlag (1986).

[43] A. Pinkus, Approximation theory of the MLP model in neural networks, *Acta Numerica* **8** (1998) 277–283.

[44] G. Pisier, Remarques sur un resultat non publié de B. Maurey, In *Seminaire d'Analyse Fonctionelle* I., n.12 (1981)

[45] W. Rudin, *Functional Analysis*, New York: McGraw-Hill (1973).

[46] I. Singer, *Best approximation in normed linear spaces by elements of subspaces*, Berlin: Springer-Verlag (1970).

[47] T.J. Sejnowski, C. Rosenberg, Parallel networks that learn to pronounce English text. *Complex Systems* **1** (1987) 145–168.

[48] T.J. Sejnowski, B.P. Yuhas, Mappings between high-dimensional representations of acoustic and visual speech signals, In *Computation and Cognition*, Philadelphia: Siam (1991) 52–68.

[49] A.N. Tikhonov, V.Y. Arsenin, *Solutions of Ill-posed Problems*, Washington DC : W. H. Winston (1977).

[50] A.A. Vladimirov, Y.E. Nesterov, Y.N. Chekanov, On uniformly convex functionals, *Moscow University Computational Mathematics and Cybernetics* **3** (1979) 12–23.

[51] G. Wahba, *Splines Models for Observational Data*, Series in Applied Mathematics, vol. 59. Philadelphia: SIAM (1990).

[52] H.J. Weaver, *Applications of Discrete and Continuous Fourier Analysis*. New York: John Wiley (1983).

Chapter 5

Functional Learning through Kernels

Stéphane Canu[1], Xavier Mary and Alain Rakotomamonjy

Abstract. This chapter reviews the functional aspects of statistical learning theory. The main point under consideration is the nature of the hypothesis set when no prior information is available but data. Within this framework we first discuss about the hypothesis set: it is a vectorial space, it is a set of pointwise defined functions, and the evaluation functional on this set is a continuous mapping. Based on these principles an original theory is developed generalizing the notion of reproduction kernel Hilbert space to non hilbertian sets. Then it is shown that the hypothesis set of any learning machine has to be a generalized reproducing set. Therefore, thanks to a general "representer theorem", the solution of the learning problem is still a linear combination of a kernel. Furthermore, a way to design these kernels is given. To illustrate this framework some examples of such reproducing sets and kernels are given.

[1]Part of this work has been realized while the authors were visiting B. Schölkopf in Tuebingen. The section dedicated to the representer theorem benefits from O. Bousquet ideas. This work also benefits from comments and discussion with NATO-ASI on Learning Theory and Practice students in Leuven.

5.1 Some Questions Regarding Machine Learning

Kernels and in particular Mercer or reproducing kernels play a crucial role in statistical learning theory and functional estimation. But very little is known about the associated hypothesis set, the underlying functional space where learning machines look for the solution. How to choose it? How to build it? What is its relationship with regularization? The machine learning community has been interested in tackling the problem the other way round. For a given learning task, therefore for a given hypothesis set, is there a learning machine capable of learning it? The answer to such a question allows to distinguish between learnable and non-learnable problem. The remaining question is: is there a learning machine capable of learning any learnable set.

We know since [13] that learning is closely related to the approximation theory, to the generalized spline theory, to regularization and, beyond, to the notion of reproducing kernel Hilbert space ($r.k.h.s$). This framework is based on the minimization of the empirical cost plus a stabilizer (*i.e.* a norm is some Hilbert space). Then, under these conditions, the solution to the learning task is a linear combination of some positive kernel whose shape depends on the nature of the stabilizer. This solution is characterized by strong and nice properties such as universal consistency.

But within this framework there remains a gap between theory and practical solutions implemented by practitioners. For instance, in $r.k.h.s$, kernels are positive. Some practitioners use hyperbolic tangent kernel $\tanh(\mathbf{w}^\top \mathbf{x} + w_0)$ while it is not a positive kernel: but it works. Another example is given by practitioners using non-hilbertian framework. The sparsity upholder uses absolute values such as $\int |f| d\mu$ or $\sum_j |\alpha_j|$: these are L^1 norms. They are not hilbertian. Others escape the hilbertian approximation orthodoxy by introducing prior knowledge (*i.e.* a stabilizer) through information type criteria that are not norms.

This chapter aims at revealing some underlying hypothesis of the learning task extending the reproducing kernel Hilbert space framework. To do so we begin with reviewing some learning principle. We will stress that the hilbertian nature of the hypothesis set is not necessary while the reproducing property is. This leads us to define a non hilbertian framework for reproducing kernel allowing non positive kernel, non-hilbertian norms and other kinds of stabilizers.

The chapter is organized as follows. The first point is to establish the three basic principles of learning. Based on these principles and before entering the non-hilbertian framework, it appears necessary to recall some basic elements of the theory of reproducing kernel Hilbert space and how to build them from non reproducing Hilbert space. Then the construction of non-hilbertian reproducing space is presented by replacing the dot (or inner) product by a more general duality map. This implies distinguishing between two different sets put in duality, one for hypothesis and the other one for measuring. In the hilbertian framework these two sets are merged in a single Hilbert space.

But before going into technical details we think it advisable to review the use of $r.k.h.s$ in the learning machine community.

5.2 *r.k.h.s* Perspective

5.2.1 Positive kernels

The interest of *r.k.h.s* arises from its associated kernel. As it were, a *r.k.h.s* is a set of functions entirely defined by a kernel function. A Kernel may be characterized as a function from $\mathcal{X} \times \mathcal{X}$ to $\mathbb{R}$ (usually $\mathcal{X} \subseteq R^d$). Mercer [11] first establishes some remarkable properties of a particular class of kernels: positive kernels defining an integral operator. These kernels have to belong to some functional space (typically $L^2(\mathcal{X} \times \mathcal{X})$, the set of square integrable functions on $\mathcal{X} \times \mathcal{X}$) so that the associated integral operator is compact. The positivity of kernel K is defined as follows:

$$K(x,y) \text{ positive} \quad \Leftrightarrow \quad \forall f \in L^2, \ \ \langle \langle K, f \rangle_{L^2}, f \rangle_{L^2} \geq 0$$

where $\langle ., . \rangle_{L^2}$ denotes the dot product in L^2. Then, because it is compact, the kernel operator admits a countable spectrum and thus the kernel can be decomposed. Based on that, the work by Aronszajn [2] can be presented as follows. Instead of defining the kernel operator from L^2 to L^2 Aronszajn focuses on the *r.k.h.s* H embedded with its dot product $\langle ., . \rangle_H$. In this framework the kernel has to be a pointwise defined function. The positivity of kernel K is then defined as follows:

$$K(x,y) \text{ positive} \quad \Leftrightarrow \quad \forall g \in H, \ \ \langle \langle K, g \rangle_H, g \rangle_H \geq 0 \tag{5.1}$$

Aronszajn first establishes a bijection between kernel and *r.k.h.s*. Then L. Schwartz [16] shows that this was a particular case of a more general situation. The kernel doesn't have to be a genuine function. He generalizes the notion of positive kernels to weakly continuous linear application from the dual set E^* of a vector space E to itself. To share interesting properties the kernel has to be positive in the following sense:

$$K \text{ positive} \quad \Leftrightarrow \quad \forall h \in E^* \ \ ((K(h), h)_{E,E^*} \geq 0$$

where $(., .)_{E,E^*}$ denotes the duality product between E and its dual set E^*. The positivity is no longer defined in terms of scalar product. But there is still a bijection between positive Schwartz kernels and Hilbert spaces.

Of course this is only a short part of the story. For a detailed review on *r.k.h.s* and a complete literature survey see [3, 14]. Moreover some authors consider non-positive kernels. A generalization to Banach sets has been introduced [4] within the framework of the approximation theory. Non-positive kernels have been also introduced in Kre˘in spaces as the difference between two positive ones ([1] and [16] section 12).

5.2.2 *r.k.h.s* and learning in the literature

The first contribution of *r.k.h.s* to the statistical learning theory is the regression spline algorithm. For an overview of this method see Wahba's book [20]. In this book two important hypothesis regarding the application of the *r.k.h.s* theory to statistics are stressed. These are the nature of pointwise defined functions and the continuity of the evaluation functional. An important and general result in this framework is the so-called representer theorem [9]. This theorem states that the solution of some class

of approximation problems is a linear combination of a kernel evaluated at the training points. But only applications in one or two dimensions are given. This is due to the fact that, in that work, the way to build *r.k.h.s* was based on some derivative properties. For practical reason only low dimension regressors were considered by this means.
Poggio and Girosi extended the framework to large input dimension by introducing radial functions through regularization operator [13]. They show how to build such a kernel as the green functions of a differential operator defined by its Fourier transform.
Support vector machines (SVM) perform another important link between kernel, sparsity and bounds on the generalization error [19]. This algorithm is based on Mercer's theorem and on the relationship between kernel and dot product. It is based on the ability for positive kernel to be separated and decomposed according to some generating functions. But to use Mercer's theorem the kernel has to define a compact operator. This is the case for instance when it belongs to L^2 functions defined on a compact domain.
Links between green functions, SVM and reproducing kernel Hilbert space were introduced in [8] and [17]. The link between *r.k.h.s* and bounds on a compact learning domain has been presented in a mathematical way by Cucker and Smale [5].

Another important application of *r.k.h.s* to learning machines comes from the bayesian learning community. This is due to the fact that, in a probabilistic framework, a positive kernel is seen as a covariance function associated to a gaussian process.

5.3 Three Principles on the Nature of the Hypothesis Set

5.3.1 The learning problem

A supervised learning problem is defined by a learning domain $\mathcal{X} \subseteq \mathbb{R}^d$ where d denotes the number of explicative variables, the learning codomain $\mathcal{Y} \subseteq \mathbb{R}$ and a n dimensional sample $\{(\mathbf{x}_i, y_i), i = 1, n\}$: the training set.

Main stream formulation of the learning problem considers the loading of a learning machine based on empirical data as the minimization of a given criterion with respect to some hypothesis lying in a hypothesis set $\mathcal{H}$. In this framework hypotheses are functions f from $\mathcal{X}$ to $\mathcal{Y}$ and the hypothesis space $\mathcal{H}$ is a functional space.

Hypothesis H_1 : $\mathcal{H}$ is a functional vector space

Technically a convergence criterion is needed in $\mathcal{H}$, *i.e.* $\mathcal{H}$ has to be embedded with a topology. In the remaining, we will always assumed $\mathcal{H}$ to be a convex topological vector space.

Learning is also the minimization of some criterion. Very often the criterion to be minimized contains two terms. The first one, C, represents the fidelity of the hypothesis with respect to data while Ω, the second one, represents the compression required to make a difference between memorizing and learning. Thus the learning machine solves the following minimization problem:

$$\min_{f \in \mathcal{H}} C(f(x_1), ..., f(x_n), y) + \Omega(f) \tag{5.2}$$

The fact is, while writing this cost function, we implicitly assume that the value of function f at any point x_i is known. We will now discuss the important consequences this assumption has on the nature of the hypothesis space $\mathcal{H}$.

5.3.2 The evaluation functional

By writing $f(x_i)$ we are assuming that function f can be evaluated at this point. Furthermore if we want to be able to use our learning machine to make a prediction for a given input x, $f(x)$ has to exist for all $x \in \mathcal{X}$: we want pointwise defined functions. This property is far from being shared by all functions. For instance function $\sin(1/t)$ is not defined in 0. Hilbert space L^2 of square integrable functions is a quotient space of functions defined only almost everywhere (*i.e.* not on the singletons $\{x\}, x \in \mathcal{X}$). L^2 functions are not pointwise defined because the L^2 elements are equivalence classes.

To formalize our point of view we need to define $\mathbb{R}^{\mathcal{X}}$ as the set of all pointwise defined functions from $\mathcal{X}$ to $\mathbb{R}$. For instance when $\mathcal{X} = \mathbb{R}$ all finite polynomials (including constant function) belong to $\mathbb{R}^{\mathcal{X}}$. We can lay down our second principle:

Hypothesis H_2 : $\mathcal{H}$ is a set of pointwise defined function (*i.e.* a subset of $\mathbb{R}^{\mathcal{X}}$)

Of course this is not enough to define a hypothesis set properly and at least another fundamental property is required.

5.3.3 Continuity of the evaluation functional

The pointwise evaluation of the hypothesis function is not enough. We want also the pointwise convergence of the hypothesis. If two functions are closed in some sense we don't want them to disagree on any point. Assume t is our unknown target function to be learned. For a given sample of size n a learning algorithm provides a hypothesis f_n. Assume this hypothesis converges in some sense to the target hypothesis. Actually the reason for hypothesis f_n is that it will be used to predict the value of t at a given x. For any x we want $f_n(x)$ to converge to $t(x)$ as follows:

$$f_n \xrightarrow{\mathcal{H}} t \Longrightarrow \forall x \in \mathcal{X},\ f_n(x) \xrightarrow{\mathbb{R}} t(x)$$

We are not interested in global convergence properties but in local convergence properties. Note that it may be rather dangerous to define a learning machine without this property. Usually the topology on $\mathcal{H}$ is defined by a norm. Then the pointwise convergence can be restated as follows:

$$\forall x \in \mathcal{X},\ \exists M_x \in \mathbb{R}^+ \text{ such that } |f(x) - t(x)| \leq M_x \, ||f - t||_{\mathcal{H}} \tag{5.3}$$

At any point x, the error can be controlled.

It is interesting to restate this hypothesis with the evaluation functional

Definition 1 *the evaluation functional*

$$\begin{array}{rccc} \delta_x : & \mathcal{H} & \longrightarrow & \mathbb{R} \\ & f & \longmapsto & \delta_x f = f(x) \end{array}$$

Applied to the evaluation functional our prerequisite of pointwise convergence is equivalent to its continuity.

$$\text{Hypothesis } H_3: \text{ the evaluation functional is continuous on } \mathcal{H}$$

Since the evaluation functional is linear and continuous, it belongs to the topological dual of $\mathcal{H}$. We will see that this is the key point to get the reproducing property.

Note that the continuity of the evaluation functional does not necessarily imply uniform convergence. But in many practical cases it does. To do so one additional hypothesis is needed, the constants M_x have to be bounded: $\sup_{x\in\mathcal{X}} M_x < \infty$. For instance this is the case when the learning domain $\mathcal{X}$ is bounded. Differences between uniform convergence and evaluation functional continuity is a deep and important topic for learning machine but out of the scope of this chapter.

5.3.4 Important consequence

To build a learning machine we do need to choose our hypothesis set as a reproducing space to get the pointwise evaluation property and the continuity of this evaluation functional. But the hilbertian structure is not necessary. Embedding a set of functions with the property of continuity of the evaluation functional has many interesting consequences. The most useful one in the field of learning machine is the existence of a kernel K, a two-variable function with generation property[2]:

$$\forall f \in \mathcal{H},\ \exists \ell \in \mathbb{N},\ (\alpha_i)_{i=1,\ell} \text{ such that } f(x) \approx \sum_{i=1}^{\ell} \alpha_i K(x, x_i)$$

Note that for practical reasons f may have a different representation.

If the evaluation set is also a Hilbert space (a vector space embedded with a dot product) it is a reproducing kernel Hilbert space ($r.k.h.s$). Although not necessary, $r.k.h.s$ are widely used for learning because they have a lot of nice practical properties. Before moving on more general reproducing sets, let's review the most important properties of $r.k.h.s$ for learning.

5.3.5 $\mathbb{R}^{\mathcal{X}}$ the set of the pointwise defined functions on $\mathcal{X}$

In the following, the function space of the pointwise defined functions $\mathbb{R}^{\mathcal{X}} = \{f : \mathcal{X} \to \mathbb{R}\}$ will be seen as a topological vector space embedded with the topology of simple convergence.
$\mathbb{R}^{\mathcal{X}}$ will be put in duality with $\mathbb{R}^{[\mathcal{X}]}$ the set of all functions on $\mathcal{X}$ equal to zero everywhere except on a finite subset $\{x_i, i \in I\}$ of $\mathcal{X}$. Thus all functions belonging to $\mathbb{R}^{[\mathcal{X}]}$ can be written in the following way:

$$g \in \mathbb{R}^{[\mathcal{X}]} \Longleftrightarrow \exists \{\alpha_i\}, i = 1, n \text{ such that } g(x) = \sum_i \alpha_i \mathbb{1}_{x_i}(x)$$

[2]this property means that the set of all finite linear combinations of the kernel is dense in $\mathcal{H}$.

were the indicator function $\mathbb{1}_{x_i}(x)$ is null everywhere except on x_i where it is equal to one.

$$\forall x \in \mathcal{X} \quad \mathbb{1}_{x_i}(x) = 0 \text{ if } x \neq x_i \text{ and } \mathbb{1}_{x_i}(x) = 1 \text{ if } x = x_i$$

Note that the indicator function is closely related to the evaluation functional since they are in bijection through:

$$\forall f \in \mathbb{R}^{\mathcal{X}}, \forall x \in \mathcal{X}, \quad \delta_x(f) = \sum_{y \in \mathcal{X}} \mathbb{1}_x(y) f(y) = f(x)$$

But formally, $\left(\mathbb{R}^{\mathcal{X}}\right)' = \text{span}\{\delta_x\}$ is a set of linear forms while $\mathbb{R}^{|\mathcal{X}|}$ is a set of pointwise defined functions.

5.4 Reproducing Kernel Hilbert Space (*r.k.h.s*)

Definition 2 (Hilbert space) *A vector space H embedded with the positive definite dot product $\langle .,. \rangle_H$ is a Hilbert space if it is complete for the induced norm $\|f\|_H^2 = \langle f, f \rangle_H$ (*i.e. *all Cauchy sequences converge in H).*

For instance $\mathbb{R}^n$, $\mathcal{P}_k$ the set of polynomials of order lower or equals to k, L^2, ℓ^2 the set of square summable sequences seen as functions on $\mathbb{N}$ are Hilbert spaces. L^1 and the set of bounded functions L^∞ are not.

Definition 3 (reproducing kernel Hilbert space (*r.k.h.s*)) *A Hilbert space $(\mathcal{H}, \langle .,. \rangle_{\mathcal{H}})$ is a r.k.h.s if it is defined on $\mathbb{R}^{\mathcal{X}}$ (pointwise defined functions) and if the evaluation functional is continuous on H (see the definition of continuity equation (5.3)).*

For instance $\mathbb{R}^n$, $\mathcal{P}_k$ as any finite dimensional set of genuine functions are *r.k.h.s*. ℓ^2 is also a *r.k.h.s*. The Cameron-Martin space defined in the examples is a *r.k.h.s* while L^2 is not because it is not a set of pointwise functions.

Definition 4 (positive kernel) *A function from $\mathcal{X} \times \mathcal{X}$ to $\mathbb{R}$ is a positive kernel if it is symmetric and if for any finite subset $\{x_i\}, i = 1, n$ of $\mathcal{X}$ and any sequence of scalar $\{\alpha_i\}, i = 1, n$*

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j K(x_i, y_j) \geq 0$$

This definition is equivalent to Aronszajn definition of positive kernel given equation (5.1).

Proposition 1 (bijection between *r.k.h.s* and Kernel) *Corollary of proposition 23 in [16] and theorem 1.1.1 in [20]. There is a bijection between the set of all possible r.k.h.s and the set of all positive kernels.*

PROOF.

$\Rightarrow$ from *r.k.h.s* to Kernel. Let $(\mathcal{H}, (.,.)_{\mathcal{H}})$ be a *r.k.h.s*. By hypothesis the evaluation functional δ_x is a continuous linear form so that it belongs to the topological dual of $\mathcal{H}$. Thanks to the Riesz theorem we know that for each $x \in \mathcal{X}$ there exists a function $K_x(.)$ belonging to $\mathcal{H}$ such that for any function $f(.) \in \mathcal{H}$:

$$\delta_x(f(.)) = \langle K_x(.), f(.) \rangle_{\mathcal{H}}$$

$K_x(.)$ is a function from $\mathcal{X} \times \mathcal{X}$ to $\mathbb{R}$ and thus can be written as a two variable function $K(x,y)$. This function is symmetric and positive since, for any real finite sequence $\{\alpha_i\}, i = 1, \ell$, $\sum_{i=1}^{\ell} \alpha_i K(x, x_i) \in \mathcal{H}$, we have:

$$\begin{aligned} \| \textstyle\sum_{i=1}^{\ell} \alpha_i K(., x_i) \|^2_{\mathcal{H}} &= \langle \textstyle\sum_{i=1}^{\ell} \alpha_i K(., x_i), \sum_{j=1}^{\ell} \alpha_j K(., x_j) \rangle_{\mathcal{H}} \\ &= \displaystyle\sum_{i=1}^{\ell} \sum_{j=1}^{\ell} \alpha_i \alpha_j K(x_i, x_j) \end{aligned}$$

$\Leftarrow$ from kernel to *r.k.h.s*. For any couple $(f(.), g(.))$ of $\mathbb{R}^{[\mathcal{X}]}$ (there exist two finite sequences $\{\alpha_i\} i = 1, \ell$ and $\{\beta_j\}, j = 1, m$ and two sequence of $\mathcal{X}$ points $\{x_i\}, i = 1, \ell$, $\{y_j\}, j = 1, m$ such that $f(x) = \sum_{i=1}^{\ell} \alpha_{i=1}^{\ell} \mathbb{I}_{x_i}(x)$ and $g(x) = \sum_{j=1}^{m} \beta_j \mathbb{I}_{y_j}(x)$) we define the following bilinear form:

$$\langle f(.), g(.) \rangle_{[\mathcal{X}]} = \sum_{i=1}^{\ell} \sum_{j=1}^{m} \alpha_i \beta_j K(x_i, y_j)$$

Let $\mathcal{H}_0 = \{f \in \mathbb{R}^{[\mathcal{X}]} |\ \langle f(.), f(.) \rangle_{[\mathcal{X}]} = 0\}$. $\langle .,. \rangle_{[\mathcal{X}]}$ defines a dot product on the quotient set $\mathbb{R}^{[\mathcal{X}]}/\mathcal{H}_0$. Now let's define $\mathcal{H}$ as the $\mathbb{R}^{[\mathcal{X}]}$ completion for the corresponding norm. $\mathcal{H}$ is a *r.k.h.s* with kernel K by construction.

□

Proposition 2 (from basis to Kernel) *Let $\mathcal{H}$ be a r.k.h.s. Its kernel K can be written:*

$$K(x,y) = \sum_{i \in I} e_i(x)\, e_i(y)$$

for all orthonormal basis $\{e_i\}_{i \in I}$ of $\mathcal{H}$, I being a set of indices possibly infinite and non-countable.

PROOF. $K \in \mathcal{H}$ implies there exits a real sequence $\{\alpha_i\}_{i \in I}$ such that $K(x,.) = \sum_{i \in I} \alpha_i e_i(x)$. Then for all $e_i(x)$ element of the orthonormal basis:

$$\begin{array}{lrcll} & \langle K(.,y), e_i(.) \rangle_{\mathcal{H}} &=& e_i(y) & \text{reproducing property of } K \\ \text{and} & \langle K(.,y), e_i(.) \rangle_{\mathcal{H}} &=& \langle \sum_{j \in I} \alpha_j e_j(.), e_i(.) \rangle_{\mathcal{H}} & \\ & &=& \sum_{j \in I} \alpha_j \langle e_j(.), e_i(.) \rangle_{\mathcal{H}} & \\ & &=& \alpha_i & \{e_i\}_{i \in I} \text{ is orthonormal basis} \end{array}$$

by identification we have $\alpha_i = e_i(y)$. □

Remark 1 *Thanks to this results it is also possible to associate to any positive kernel a basis, possibly uncountable. Consequently to proposition 1 we now how to associate a r.k.h.s to any positive kernel and we get the result because every Hilbert space admit an orthonormal basis.*

The fact that the basis is countable or uncountable (that the corresponding *r.k.h.s* is separable or not) has no consequences on the nature of the hypothesis set (see examples). Thus Mercer kernels are a particular case of a more general situation since every Mercer kernel is positive in the Aronszajn sense (definition 4) while the converse is false. Consequently, when possible functional formulation is preferable to kernel formulation of learning algorithm.

5.5 Kernel and Kernel Operator

5.5.1 How to build *r.k.h.s*?

It is possible to build *r.k.h.s* from a $L^2(G,\mu)$ Hilbert space where G is a set (usually $G = \mathcal{X}$) and μ a measure. To do so, an operator S is defined to map L^2 functions onto the set of the pointwise valued functions $\mathbb{R}^{\mathcal{X}}$. A general way to define such an operator consists in remarking that the scalar product performs such a linear mapping. Based on that remark this operator is built from a family Γ_x of $L^2(G,\mu)$ functions when $x \in \mathcal{X}$ in the following way:

Definition 5 (Carleman operator) *Let $\Gamma = \{\Gamma_x, x \in \mathcal{X}\}$ be a family of $L^2(G,\mu)$ functions. The associated Carleman operator S is*

$$\begin{array}{rccl} S: & L^2 & \longrightarrow & \mathbb{R}^{\mathcal{X}} \\ & f & \longmapsto & g(.) = (Sf)(.) = \langle \Gamma_{(.)}, f \rangle_{L^2} = \displaystyle\int_G \Gamma_{(.)}\, f\, d\mu \end{array}$$

That is to say $\forall x \in \mathcal{X},\ g(x) = \langle \Gamma_x, f \rangle_{L^2}$. To make apparent the bijective restriction of S it is convenient to factorize it as follows:

$$S : L^2 \longrightarrow L^2/\mathrm{Ker}(S) \xrightarrow{T} \mathrm{Im}(S) \xrightarrow{i} \mathbb{R}^{\mathcal{X}} \tag{5.4}$$

where $L^2/\mathrm{Ker}(S)$ is the quotient set, T the bijective restriction of S and i the canonical injection.

This class of integral operators is known as Carleman operators [18]. Note that this operator unlike Hilbert-Schmidt operators need not be compact neither bounded. But when G is a compact set or when $\Gamma_x \in L^2(G \times G)$ (it is a square integrable function with respect to both of its variables) S is a Hilbert-Schmidt operator. As an illustration of this property, see the gaussian example on $G = \mathcal{X} = \mathbb{R}$ in Table 5.1. In that case $\Gamma_x(\tau) \notin L^2(\mathcal{X} \times \mathcal{X})$[3].

[3]To clarify the not so obvious notion of pointwise defined function, whenever possible, we use the notation f when the function is not a pointwise defined function and $f(.)$ denotes $\mathbb{R}^{\mathcal{X}}$ functions. Here $\Gamma_x(\tau)$ is a pointwise defined function with respect to variable x but not with respect to variable τ. Thus, whenever possible, the confusing notation (τ) is omitted.

Proposition 3 (bijection between Carleman operators and the set of *r.k.h.s*) *- Proposition 21 in [16] or theorems 1 and 4 in [14]. Let S be a Carleman operator. Its image set $\mathcal{H} = Im(S)$ is a r.k.h.s. If $\mathcal{H}$ is a r.k.h.s there exists a measure μ on some set G and a Carleman operator S on $L^2(G,\mu)$ such that $\mathcal{H} = Im(S)$.*

PROOF.

$\Rightarrow$ Consider T the bijective restriction of S defined in equation (5.4). $\mathcal{H} = \mathrm{Im}(S)$ can be embedded with the induced dot product defined as follows:

$$\begin{aligned} \forall g_1(.), g_2(.) \in \mathcal{H}^2, \quad \langle g_1(.), g_2(.)\rangle_{\mathcal{H}} &= \langle T^{-1}g_1, T^{-1}g_2\rangle_{L^2} \\ &= \langle f_1, f_2\rangle_{L^2} \end{aligned}$$

where $g_1(.) = Tf_1$ and $g_2(.) = Tf_2$. With respect to the induced norm, T is an isometry. To prove $\mathcal{H}$ is a *r.k.h.s*, we have to check the continuity of the evaluation functional. This works as follows:

$$\begin{aligned} g(x) &= (Tf)(x) \\ &= \langle \Gamma_x, f\rangle_{L^2} \leq \|\Gamma_x\|_{L^2} \|f\|_{L^2} \\ &\leq M_x \|g(.)\|_{\mathcal{H}} \end{aligned}$$

with $M_x = \|\Gamma_x\|_{L^2}$. In this framework $\mathcal{H}$ reproducing kernel K verifies $S\Gamma_x = K(x,.)$. It can be built based on Γ:

$$\begin{aligned} K(x,y) &= \langle K(x,.), K(y,.)\rangle_{\mathcal{H}} \\ &= \langle \Gamma_x, \Gamma_y\rangle_{L^2} \end{aligned}$$

$\Leftarrow$ Let $\{e_i\}, i \in I$ be a $L^2(G,\mu)$ orthonormal basis and $\{h_j(.)\}, j \in J$ an orthonormal basis of $\mathcal{H}$. We admit there exists a couple (G,μ) such that card(I) $\geq$ card(J) (take for instance the counting measure on the suitable set). Define $\Gamma_x = \sum_{j\in J} h_j(x)e_j$ as a L^2 family. Let T be the associated Carleman operator. The image of this Carleman operator is the *r.k.h.s* span by $h_j(.)$ since:

$$\begin{aligned} \forall f \in L^2, \quad (Tf)(x) &= \langle \Gamma_x, f\rangle_{L^2} \\ &= \langle \sum_{j\in J} h_j(x)e_j, \sum_{i\in I} \alpha_i e_i\rangle_{L^2} \quad \text{because } f = \sum_{i\in I} \alpha_i e_i \\ &= \sum_{j\in J} h_j(x) \sum_{i\in I} \alpha_i \langle e_j, e_i\rangle_{L^2} \\ &= \sum_{j\in J} \alpha_j h_j(x) \end{aligned}$$

and family $\{h_i(.)\}$ is orthonormal since $h_i(.) = Te_i$.

□

To put this framework at work the relevant function Γ_x has to be found. Some examples with popular kernels illustrating this definition are shown in Table 5.1.

5.5.2 Carleman operator and the regularization operator

The same kind of operator has been introduced by Poggio and Girosi in the regularization framework [13]. They proposed to define the regularization term $\Omega(f)$ (defined in

Name	$\Gamma_x(u)$	$K(x,y)$
Cameron Martin	$\mathbb{1}_{\{x \leq u\}}$	$\min(x,y)$
Polynomial	$e_0(u) + \sum_{i=1}^{d} x_i e_i(u)$	$\mathbf{x}^\top \mathbf{y} + 1$
Gaussian	$\frac{1}{Z} \exp^{-\frac{(x-u)^2}{2}}$	$\frac{1}{Z'} \exp^{-\frac{(x-y)^2}{4}}$

Table 5.1: Examples of Carleman operator and their associated reproducing kernel. Note that functions $\{e_i\}_{i=1,d}$ are a finite subfamily of a L^2 orthonormal basis. Z and Z' are two constants.

(5.2)) by introducing a regularization operator P from hypothesis set $\mathcal{H}$ to L^2 such that $\Omega(f) = \|Pf\|^2_{L^2}$. This framework is very attractive since operator P models the prior knowledge about the solution defining its regularity in terms of derivative or Fourier decomposition properties. Furthermore the authors show that, in their framework, the solution of the learning problem is a linear combination of a kernel (a representer theorem). They also give a methodology to build this kernel as the green function of a differential operator. Following [2] in its introduction the link between green function and *r.k.h.s* is straightforward when green function is a positive kernel. But a problem arises when operator P is chosen as a derivative operator and the resulting kernel is not derivable (for instance when P is the simple derivation, the associated kernel is the non-derivable function $\min(x,y)$). A way to overcome this technical difficulty is to consider things the other way round by defining the regularization term as the norm of the function in the *r.k.h.s* built based on Carleman operator T. In this case we have $\Omega(f) = \|f\|_H = \|T^{-1}g\|^2_{L^2}$. Thus since T is bijective we can define operator P as: $P = T^{-1}$. This is no longer a derivative operator but a generalized derivative operator where the derivation is defined as the inverse of the integration (P is defined as T^{-1}).

5.5.3 Generalization

It is important to notice that the above framework can be generalized to non L^2 Hilbert spaces. A way to see this is to use Kolmogorov's dilation theorem [7]. Furthermore, the notion of reproducing kernel itself can be generalized to non-pointwise defined function by emphasizing the role played by continuity through positive generalized kernels called Schwartz or hilbertian kernels [16]. But this is out of the scope of our work.

5.6 Reproducing Kernel Spaces (*r.k.k.s*)

By focusing on the relevant hypothesis for learning we are going to generalize the above framework to non-hilbertian spaces.

5.6.1 Evaluation spaces

Definition 6 (ES)
Let $\mathcal{H}$ be a real topological vector space (t.v.s.) on an arbitrary set $\mathcal{X}$, $\mathcal{H} \subset \mathbb{R}^{\mathcal{X}}$. $\mathcal{H}$ is

an evaluation space if and only if:

$$\forall x \in \mathcal{X}, \quad \begin{array}{rcl} \delta_x : \mathcal{H} & \longrightarrow & \mathbb{R} \\ f & \longmapsto & \delta_x(f) = f(x) \end{array} \quad \textit{is continuous}$$

ES are then topological vector spaces in which δ_t (the evaluation functional at t) is continuous, *i.e.* belongs to the topological dual $\mathcal{H}^*$of $\mathcal{H}$.

Remark 2 *Topological vector space $\mathbb{R}^{\mathcal{X}}$ with the topology of simple convergence is by construction an ETS (evaluation topological space).*

In the case of normed vector space, another characterization can be given:

Proposition 4 (normed ES or BES)
Let $(\mathcal{H}, \|.\|_{\mathcal{H}})$ be a real normed vector space on an arbitrary set $\mathcal{X}$, $\mathcal{H} \subset \mathbb{R}^{\mathcal{X}}$. $\mathcal{H}$ is an evaluation kernel space if and only if the evaluation functional:

$$\forall x \in \mathcal{X}, \ \exists M_x \in \mathbb{R}, \ \forall f \in \mathcal{H}, \ |f(x)| \leq M_x \|f\|_{\mathcal{H}}$$

if it is complete for the corresponding norm it is a Banach evaluation space (BES).

Remark 3 *In the case of a Hilbert space, we can identify $\mathcal{H}^*$ and $\mathcal{H}$ and, thanks to the Riesz theorem, the evaluation functional can be seen as a function belonging to $\mathcal{H}$: it is called the reproducing kernel.*

This is an important point: thanks to the hilbertian structure the evaluation functional can be seen as a hypothesis function and therefore the solution of the learning problem can be built as a linear combination of this reproducing kernel taken different points. Representer theorem [9] demonstrates this property when the learning machine minimizes a regularized quadratic error criterion. We shall now generalize these properties to the case when no hilbertian structure is available.

5.6.2 Reproducing kernels

The key point when using Hilbert space is the dot product. When no such bilinear positive functional is available its role can be played by a duality map. Without dot product, the hypothesis set $\mathcal{H}$ is no longer in self duality. We need another set $\mathcal{M}$ to put in duality with $\mathcal{H}$. This second set $\mathcal{M}$ is a set of functions measuring how the information I have at point x_1 helps me to measure the quality of the hypothesis at point x_2. These two sets have to be in relation through a specific bilinear form. This relation is called a duality.

Definition 7 (Duality between two sets) *Two sets $(\mathcal{H}, \mathcal{M})$ are in duality if there exists a bilinear form $\mathcal{L}$ on $\mathcal{H} \times \mathcal{M}$ that separates $\mathcal{H}$ and $\mathcal{M}$ (see [10] for details on the topological aspect of this definition).*

Let $\mathcal{L}$ be such a bilinear form on $\mathcal{H} \times \mathcal{M}$ that separate them. Then we can define a linear application $\gamma_{\mathcal{H}}$ and its reciprocal $\theta_{\mathcal{H}}$ as follows:

$$\begin{array}{rccc} \gamma_{\mathcal{H}}: & \mathcal{M} & \longrightarrow & \mathcal{H}^* \\ & f & \longmapsto & \gamma_{\mathcal{H}} f = \mathcal{L}(.,f) \end{array} \qquad \begin{array}{rccc} \theta_{\mathcal{H}}: & \mathrm{Im}\,(\gamma_{\mathcal{H}}) & \longrightarrow & \mathcal{M} \\ & g = \mathcal{L}(.,f) & \longmapsto & \theta_{\mathcal{H}} g = f \end{array}$$

where $\mathcal{H}^*$ (resp. $\mathcal{M}^*$) denotes the dual set of $\mathcal{H}$ (resp. $\mathcal{M}$).

Let's take an important example of such a duality.

Proposition 5 (duality of pointwise defined functions) *Let $\mathcal{X}$ be any set (not necessarily compact). $\mathbb{R}^{\mathcal{X}}$ and $\mathbb{R}^{[\mathcal{X}]}$ are in duality*

PROOF. Let's define the bilinear application $\mathcal{L}$ as follows:

$$\begin{array}{rccc} \mathcal{L}: & \mathbb{R}^{\mathcal{X}} \times \mathbb{R}^{[\mathcal{X}]} & \longrightarrow & \mathbb{R} \\ & \left(f(.), g(.) = \sum_{i \in I} \alpha_i \mathbb{I}_{x_i}(.)\right) & \longmapsto & \sum_{i \in I} \alpha_i f(x_i) = \sum_{x \in \mathcal{X}} f(x) g(x) \end{array}$$

□

Another example is shown in the two following functional spaces:

$$L^1 = \left\{ f \;\middle|\; \int_{\mathcal{X}} |f|\, d\mu < \infty \right\} \qquad \text{and} \qquad L^\infty = \left\{ f \;\middle|\; \operatorname*{ess\,sup}_{x \in \mathcal{X}} |f| < \infty \right\}$$

where for instance μ denotes the Lebesgue measure. Theses two spaces are put in duality through the following duality map:

$$\begin{array}{rccc} \mathcal{L}: & L^1 \times L^\infty & \longrightarrow & \mathbb{R} \\ & f, g & \longmapsto & \mathcal{L}(f,g) = \int_{\mathcal{X}} f\, g\, d\mu \end{array}$$

Definition 8 (Evaluation subduality) *Two sets $\mathcal{H}$ and $\mathcal{M}$ form an evaluation subduality iff:*

- *they are in duality through their duality map $\gamma_{\mathcal{H}}$,*
- *they both are subsets of $\mathbb{R}^{\mathcal{X}}$*
- *the continuity of the evaluation functional is preserved through:*

$$Span(\delta_x) = \gamma_{\mathbb{R}^{\mathcal{X}}}\left(\left(\mathbb{R}^{\mathcal{X}}\right)'\right) \subseteq \gamma_{\mathcal{H}}(\mathcal{M}) \quad \text{and} \quad \gamma_{\mathbb{R}^{\mathcal{X}}}\left(\left(\mathbb{R}^{\mathcal{X}}\right)'\right) \subseteq \theta_{\mathcal{H}}(\mathcal{H})$$

The key point is the way of preserving the continuity. Here the strategy to do so is first to consider two sets in duality and then to build the (weak) topology such that the dual elements are (weakly) continuous.

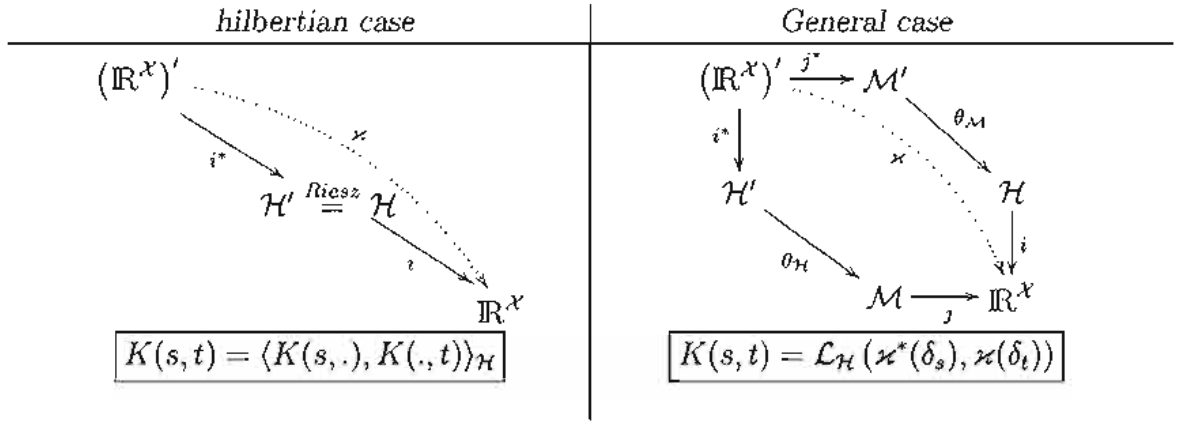


Figure 5.1: Illustration of the subduality map.

Proposition 6 (Subduality kernel) *A unique weakly continuous linear application $\varkappa$ is associated to each subduality. This linear application, called the subduality kernel, is defined as follows:*

$$\begin{array}{rccl} \varkappa: & (\mathbb{R}^{\mathcal{X}})' & \longrightarrow & \mathbb{R}^{\mathcal{X}} \\ & \sum_{i\in I}\delta_{x_i} & \longmapsto & i\circ\theta_{\mathcal{M}}\circ j^*(\sum_{i\in I}\delta_{x_i}) \end{array}$$

where i and j^* are the canonical injections from $\mathcal{H}$ to $\mathbb{R}^{\mathcal{X}}$ and respectively from $(\mathbb{R}^{\mathcal{X}})'$ to $\mathcal{M}'$ (figure 5.1).

PROOF. for details see [10]. □

We can illustrate this mapping detailing all performed applications as in Figure 5.1:

$$\begin{array}{ccccccccc} (\mathbb{R}^{\mathcal{X}})' & \longrightarrow & \mathbb{R}^{[\mathcal{X}]} & \xrightarrow{j^*} & \mathcal{M}' & \xrightarrow{\theta_{\mathcal{M}}} & \mathcal{H} & \xrightarrow{i} & \mathbb{R}^{\mathcal{X}} \\ \delta_x & \longmapsto & \mathbb{1}_{\{x\}} & \longmapsto & \mathcal{L}(K_x,.) & \longrightarrow & K_x(.) & \longmapsto & K(x,.) \end{array}$$

Definition 9 (Reproducing kernel of an evaluation subduality) *Let $(\mathcal{H},\mathcal{M})$ be an evaluation subduality with respect to map $\mathcal{L}_{\mathcal{H}}$ associated with subduality kernel $\varkappa$. The reproducing kernel associated with this evaluation subduality is the function of two variables defined as follows:*

$$\begin{array}{rccl} K: & \mathcal{X}\times\mathcal{X} & \longrightarrow & \mathbb{R} \\ & (x,y) & \longmapsto & K(x,y) = \mathcal{L}_{\mathcal{H}}(\varkappa^*(\delta_y), \varkappa(\delta_x)) \end{array}$$

This structure is illustrated in Figure 5.1. Note that this kernel no longer needs to be definite positive. If the kernel is definite positive it is associated with a unique *r.k.h.s.* However, as shown in the examples it can also be associated with evaluation subdualities. A way of looking at things is to define $\varkappa$ as the generalization of the Schwartz kernel while K is the generalization of the Aronszajn kernel to non hilbertian structures. Based on these definitions the important expression property is preserved.

Proposition 7 (generation property) *$\forall f\in\mathcal{H}$, $\exists(\alpha_i)_{i\in I}$ such that $f(x)\approx\sum_{i\in I}\alpha_i K(x,x_i)$ and $\forall g\in\mathcal{M}$, $\exists(\alpha_i)_{i\in I}$ such that $g(x)\approx\sum_{i\in I}\alpha_i K(x_i,x)$*

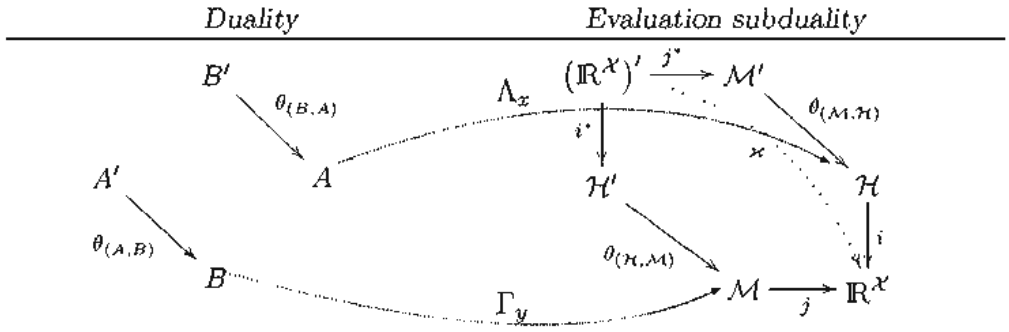


Figure 5.2: Illustration of the building operators for reproducing kernel subduality from a duality (A, B).

PROOF. This property is due to the density of $\mathrm{Span}\{K(.,x), x \in \mathcal{X}\}$ in $\mathcal{H}$. For more details see [10] Lemma 4.3. □

Just like *r.k.h.s*, another important point is the possibility to build an evaluation subduality, and of course its kernel, starting from any duality.

Proposition 8 (building evaluation subdualities) *Let (A, B) be a duality with respect to map $\mathcal{L}_A$. Let $\{\Gamma_x, x \in \mathcal{X}\}$ be a total family in A and $\{\Lambda_x, x \in \mathcal{X}\}$ be a total family in B. Let S (reps. T) be the linear mapping from A (reps. B) to $\mathbb{R}^{\mathcal{X}}$ associated with Γ_x (reps. Λ_x) as follows:*

$$\begin{array}{rccl} S: & A & \longrightarrow & \mathbb{R}^{\mathcal{X}} \\ & g & \longmapsto & Sg(x) = \mathcal{L}_A(g, \Lambda_x) \end{array} \qquad \begin{array}{rccl} T: & B & \longrightarrow & \mathbb{R}^{\mathcal{X}} \\ & f & \longmapsto & Tf(x) = \mathcal{L}_A(\Gamma_x, f) \end{array}$$

Then S and T are injective and $(S(A), T(B))$ is an evaluation subduality with the reproducing kernel K defined by:

$$K(x, y) = \mathcal{L}_A(\Gamma_x, \Lambda_y)$$

PROOF. see [10] Lemma 4.5 and proposition 4.6 □

An example of such subduality is obtained by mapping the (L^1, L^∞) duality to $\mathbb{R}^{\mathcal{X}}$ using *injective* operators defined by the families $\Gamma_x(\tau) = \mathbb{1}_{\{x<\tau\}}$ and $\Lambda_y(\tau) = \mathbb{1}_{\{y<\tau\}}$:

$$\begin{array}{rccl} T: & L^1 & \longrightarrow & \mathbb{R}^{\mathcal{X}} \\ & f & \longmapsto & Tf(x) = (\Gamma_x, f)_{L^\infty, L^1} = \int \mathbb{1}_{\{x<\tau\}} f(\tau)\, d\tau \end{array}$$

and

$$\begin{array}{rccl} S: & L^\infty & \longrightarrow & \mathbb{R}^{\mathcal{X}} \\ & g & \longmapsto & Sg(y) = (g, \Lambda_y)_{L^\infty, L^1} = \int g(\tau)\mathbb{1}_{\{y<\tau\}}\, d\tau \end{array}$$

In this case $\mathcal{H} = Im(T)$, $\mathcal{M} = Im(S)$ and $K(y, x) = \int \Lambda(y, \tau)\Gamma(x, \tau)\, d\tau = \min(x, y)$.

We define the duality map between $\mathcal{H}$ and $\mathcal{M}$ through:

$$\mathcal{L}_{\mathcal{X}}(g_1, g_2) = \mathcal{L}_{\mathcal{X}}(Sf_1, Tf_2) = \mathcal{L}(f_1, f_2)$$

See examples for details.

All useful properties of *r.k.h.s* – pointwise evaluation, continuity of the evaluation functional, representation and building technique – are preserved. A missing dot product has no consequence on this functional aspect of the learning problem.

5.7 Representer Theorem

Another issue is of paramount practical importance: determining the shape of the solution. To this end the representer theorem states that, when $\mathcal{H}$ is a *r.k.h.s*, the solution of the minimization of the regularized cost defined equation (5.2) is a linear combination of the reproducing kernel evaluated at the training examples [9, 15]. When hypothesis set $\mathcal{H}$ is a reproducing space associated with a subduality we have the same kind of result. The solution lies in a finite n-dimensional subspace of $\mathcal{H}$. But we don't know yet how to systematically build a convenient generating family in this subspace.

Theorem 1 (representer) *Assume $(\mathcal{H}, \mathcal{M})$ is a subduality of $\mathbb{R}^{\mathcal{X}}$ with kernel $K(x, y)$. Assume the stabilizer Ω is convex and differentiable (∂_Ω denotes its subdifferential set). If $\partial_\Omega(\sum \alpha_i K(x_i, x)) \subseteq \{\sum \beta_i \delta_{x_i}\} \in \mathcal{H}^*$ then the solution of cost minimization lies in a n-dimensional subspace of $\mathcal{H}$.*

PROOF. Define a $\mathcal{M}$ subset $M_1 = \{\sum_{i=1}^n \alpha_i K(x_i, .)\}$. Let $H_2 \subset \mathcal{H}$ be the M_1 orthogonal in the sense of the duality map (*i.e.* $\forall f \in H_2, \forall g \in M_1 \ \mathcal{L}(f, g) = 0$). Then for all $f \in H_2, f(x_i) = 0, i = 1, n$. Now let H_1 be the complement vector space defined such that

$$\mathcal{H} = H_1 \oplus H_2 \quad \Leftrightarrow \forall f \in \mathcal{H} \ \exists f_1 \in H_1 \text{ and } f_2 \in H_2 \quad \text{such that } f = f_1 + f_2$$

The solution of the minimizing problem lies in H_1 since:

- $\forall f_2 \in H_2, C(f_2) = \text{constant}$
- $\Omega(f_1 + f_2) \geq \Omega(f_1) + (\partial_\Omega(f_1), f_2)_{\mathcal{M},\mathcal{H}}$ (thanks to the convexity of Ω)
- and $\forall f_2 \in H_2, ; (\partial_\Omega(f_1), f_2)_{\mathcal{M},\mathcal{H}} = 0$ by hypothesis

By construction H_1 a n-dimensional subspace of $\mathcal{H}$. □

The nature of vector space H_1 depends on kernel K and on regularizer Ω. In some cases it is possible to be more precise and retrieve the nature of H_1. Let's assume regularizer $\Omega(f)$ is given. $\mathcal{H}$ may be chosen as the set of functions such that $\Omega(f) < \infty$. Then, if it is possible to build a subduality $(\mathcal{H}, \mathcal{M})$ with kernel K such that

$$E = \underbrace{\text{Vect}\{K(x_i, .)\}}_{H_1} \oplus \underbrace{(\ \text{Vect}\{K(., x_i)\})^\top}_{M_1^\top}$$

and if the vector space spanned by the kernel belongs to the regularizer subdifferential $\partial\Omega(f)$:

$$\forall f \in H_1, \quad \exists g \in M_1 \text{ such that } g \in \partial\Omega(f)$$

then solution f^* of the minimization of the regularized empirical cost is a linear combination of the kernel:

$$f^*(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x).$$

An example of such result is given with the following regularizer based on the p-norm on $G = [0, 1]$:

$$\Omega(f) = \int_0^1 (f')^p \ d\mu$$

The hypothesis set is Sobolev space H^p (the set of functions defined on $[0,1]$ whose generalized derivative is p-integrable) put in duality with H^q (with $1/p + 1/q = 1$) through the following duality map:

$$\mathcal{L}(f,g) = \int_0^1 f'g' \, d\mu$$

The associated kernel is just like in Cameron Martin case $K(x,y) = \min(x,y)$. Some tedious derivations lead to:

$$\forall h \in \mathcal{H} \quad \mathcal{L}(h, \partial\Omega(f)) = \int_0^1 h' \, p(f')^{p-1} \, d\mu$$

Thus the kernel verifies $p(K(.,y)')^{p-1} \propto K(x,.)$

This question of the representer theorem is far from being closed. We are still looking for a way to derive a generating family from the kernel and the regularizer. To go more deeply into general and constructive results, a possible way to investigate is to go through Ω Fenchel dual.

5.8 Examples

5.8.1 Examples in Hilbert space

The examples in this section all deal with r.k.h.s included in a L^2 space.

1. Schmidt ellipsoid:
   Let $(\mathcal{X}, \mu)$ be a measure space, $\{e_i, i \in I\}$ a basis of $L^2(\mathcal{X}, \mu)$ I being a countable set of indices. Any sequence $\{\alpha_i, i \in I, \quad \sum_{i\in I} \alpha_i^2 < +\infty\}$ defines a Hilbert-Schmidt operator on $L^2(\mathcal{X}, \mu)$ with kernel function $\Gamma(x,y) = \sum_{i\in I} \alpha_i e_i(x) e_i(y)$, thus a reproducing kernel Hilbert space with kernel function:

   $$\forall (x,y) \in \mathcal{X}^2, \quad K(x,y) = \sum_{i\in I} \alpha_i^2 e_i(x) e_i(y)$$

   The closed unit ball $\mathfrak{B}_H$ of the $r.k.h.s$ verifies

   $$\mathfrak{B}_H = T(\mathfrak{B}_{L^2}) = \left\{ f \in L^2, f = \sum_{i\in I} f_i e_i, \quad \sum_{i\in I} \left(\frac{f_i}{\alpha_i}\right)^2 \leq 1 \right\}$$

   and is then a Schmidt ellipsoid in L^2. An interesting discussion about Schmidt ellipsoids and their applications to sample continuity of Gaussian measures may be found in [6].

2. Cameron-Martin space:
   Let T be the Carleman integral operator on $L^2([0,1], \mu)$ (μ is the Lebesgue measure) with kernel function

   $$\Gamma(x,y) = Y(x-y) = \mathbb{1}_{\{y \leq x\}}$$

it defines a *r.k.h.s* with reproducing kernel $K(x,y) = \min(x,y)$. The space $(H;\langle .,.\rangle_H)$ is the Sobolev space of degree 1, also called the Cameron-Martin space.

$$\begin{cases} H = \{f \text{ absolutely continuous}, \exists f' \in L^2([0,1]),\ f(x) = \int_0^x f' d\mu\} \\ \langle f,g\rangle_H = \langle f', g'\rangle_{L^2} \end{cases}$$

3. A Carleman but non Hilbert-Schmidt operator:
   Let T be the integral operator on $L^2(\mathbb{R},\mu)$ (μ is the Lebesgue measure) with kernel function
   $$\Gamma(x,y) = \exp^{-\frac{1}{2}(x-y)^2}$$
   It is a Carleman integral operator, thus we can define a r.k.h.s $(H;\langle .,.\rangle_H) = Im(T)$, but T is not a Hilbert-Schmidt operator. H reproducing kernel is:
   $$K(x,y) = \frac{1}{Z}\exp^{-\frac{1}{4}(x-y)^2}$$
   where Z is a suitable constant.

4. Continuous kernel:
   This example is based on Theorem 3.11 in [12]. Let $\mathcal{X}$ be a compact subspace of $\mathbb{R}$, $K(.,.)$ a continuous symmetric positive definite kernel. It defines a *r.k.h.s* $(H;\langle .,.\rangle_H)$ and any Radon measure μ of full support is kernel-injective. Then, for any such μ, there exists a Carleman operator T on $L^2(\mathcal{X},\mu)$ such that $(H;\langle .,.\rangle_H) = Im(T)$.

5. Hilbert space of constants:
   Let $(H;\langle .,.\rangle_H)$ be the Hilbert space of constant functions on $\mathbb{R}$ with scalar product $\langle f,g\rangle_H = f(0)g(0)$. It is obviously a *r.k.h.s* with reproducing kernel $K(.,.) \equiv 1$. For any probability measure μ on $\mathbb{R}$ let:
   $$\forall f \in L^2(\mathbb{R},\mu), \quad Tf = \int_{\mathbb{R}} f(s)\mu(ds)$$
   Then $H = T(L^2(\mathbb{R},\mu))$ and $\forall f,g \in H,\ \langle f,g\rangle_H = \langle f,g\rangle_{L^2}$.

6. A non-separable *r.k.h.s* - the L^2 space of almost surely null functions:
   Define the positive definite kernel function on $\mathcal{X} \subset \mathbb{R}$ by $\forall s,t \in \mathcal{X},\ K(s,t) = \mathbb{1}_{\{s=t\}}$. It defines a *r.k.h.s* $(H;\langle .,.\rangle_H)$ and its functions are null except on a countable set. Define a measure μ on $(\mathcal{X},\mathcal{B})$ where $\mathcal{B}$ is the Borel σ-algebra on $\mathcal{X}$ by $\mu(t) = 1\ \forall t \in \mathcal{X}$. μ verifies: $\mu(\{t_1,\cdots,t_n\}) = n$ and $\mu(A) = +\infty$ for any non-finite $A \in \mathcal{B}$. The kernel function is then square integrable and H is injectively included in $L^2(\mathcal{X},\mathcal{B},\mu)$. Moreover, $K(s,t) = \int_{\mathcal{X}} K(t,u)K(u,s)d\mu(u)$ with K Carleman integrable and $T = Id_{L^2}$ (note that the identity is a non-compact Carleman integral operator). Finally, $(H;\langle .,.\rangle_H) = L^2(\mathcal{X},\mathcal{B},\mu)$.

7. Separable *r.k.h.s* :
   Let H be a separable *r.k.h.s* . It is well known that any separable Hilbert space is isomorphic to ℓ^2. Then there exists T kernel operator $Im(T) = H$. It is easy

to construct effectively such a T: let $\{h_n(.), n \in \mathbf{N}\}$ be an orthonormal basis of H and define T kernel operator on ℓ^2 with kernel $\Gamma_x \rightarrow \{h_n(x), n \in \mathbf{N}\}(\in l^2)$. Then $Im(T) = H$.

5.8.2 Other examples

Applications to non-hilbertian spaces are also feasible:

1. (L^1, L^∞) - "Cameron-Martin" evaluation subduality:
   Let T be the kernel operator on $L^1([0,1], \mu)$ (μ is the Lebesgue measure) with kernel function

   $$\Gamma(t,s) = Y(t-s) = \mathbb{1}_{\{s \leq t\}}, \quad \Gamma(t,.) \in L^\infty$$

   it defines an evaluation duality $(H_1; H_\infty)$ with reproducing kernel

   $$\forall (s,t) \in \mathcal{X}^2, \quad K(s,t) = \min(s,t)$$

   $$\begin{cases} H_1 = \{f \text{ absolutely continuous}, \exists f' \in L^1([0,1]),\ f(t) = \int_0^t f'(s)ds\} \\ \|f\|_{H_1} = \|f'\|_{L^1} \end{cases}$$

   and

   $$\begin{cases} H_\infty = \{f \text{ absolutely continuous}, \exists f' \in L^\infty([0,1]),\ f(t) = \int_0^t f'(s)ds\} \\ \|f\|_{H_\infty} = \|f'\|_{L^\infty} \end{cases}$$

2. $\left(\mathbb{R}^{\mathcal{X}}, \mathbb{R}^{|\mathcal{X}|}\right)$:
   We have seen that $\mathbb{R}^{\mathcal{X}}$ endowed with the topology of simple convergence is an ETS. However, $\mathbb{R}^{\mathcal{X}}$ endowed with the topology of almost sure convergence is never an ETS unless every singleton of $\mathcal{X}$ has strictly positive measure.

5.9 Conclusion

It is always possible to learn without kernel. But even if it is not visible, one is hidden somewhere! We have shown, from some basic principles (we want to be able to compute the value of a hypothesis at any point and we want the evaluation functional to be continuous), how to derive a framework generalizing *r.k.h.s* to non-hilbertian spaces. In our reproducing kernel dualities, all *r.k.h.s* nice properties are preserved except the dot product replaced by a duality map. Based on the generalization of the hilbertian case, it is possible to build associated kernels thanks to simple operators. The construction of evaluation subdualities without Hilbert structure is easy within this framework (and rather new). The derivation of evaluation subdualities from any kernel operator has many practical outcome. First, such operators on separable Hilbert spaces can be represented by matrices, and we can build any separable *r.k.h.s* from well-known ℓ^2 structures (like wavelets in a L^2 space for instance). Furthermore, the set of kernel operators is a vector space whereas the set of evaluation subdualities is not (the set of *r.k.h.s* is for instance a convex cone), hence practical combination of

such operators are feasible. On the other hand, from the bayesian point of view, this result may have many theoretical and practical implications in the theory of Gaussian or Laplacian measures and abstract Wiener spaces.

Unfortunately, even if some work has been done, a general representer theorem is not available yet. We are looking for an automatic mechanism designing the *shape* of the solution of the learning problem in the following way:

$$f(\mathrm{x}) = \sum_{i=1}^{m} \alpha_i K(\mathrm{x}_i, \mathrm{x}) + \sum_{j=1}^{k} \beta_j \varphi_j(\mathrm{x})$$

where Kernel K, number of component m and functions $\varphi_k(\mathrm{x}), j = 1, k$ are derivated from regularizer Ω. The remaining questions being: how to learn the coefficients and how to determine cost function?

Bibliography

[1] D.A. Alpay, Some Krein spaces of analytic functions and an inverse scattering problem, *Michigan Journal of Mathematics* **34** (1987) 349–359.

[2] N. Aronszajn. Theory of reproducing kernels, *Transactions of the American Society* **68** (1950) 337–404.

[3] M. Attéia, *Hilbertian kernels and spline functions*, North-Holland (1992).

[4] M. Attéia and J. Audounet, Inf-compact potentials and banachic kernels, In *Banach space theory and its applications*, volume 991 of *Lecture notes in mathematics*, Springer-Verlag (1981) 7–27.

[5] F. Cucker and S. Smale, On the mathematical foundations of learning. *Bulletin of the American Mathematical Society* **39** (2002) 1–49.

[6] R. M. Dudley, *Uniform central limit theorems*, Cambridge university press (1999).

[7] D. Evans and J.T. Lewis, Dilations of irreversible evolutions in algebraic quantum theory, *Communications of the Dublin Institute for advanced Studies, Series A.*, 24 (1977).

[8] F. Girosi, An equivalence between sparse approximation and support vector machines, *Neural Computation* **10**(6) (1998) 1455–1480.

[9] G. Kimeldorf and G. Wahba, Some results on Tchebycheffian spline functions, *J. Math. Anal. Applic.* **33** (1971) 82–95.

[10] X. Mary, D. De Brucq and S. Canu, Sous-dualités et noyaux (reproduisants) associés, Technical report PSI 02-002 (2002). available at `asi.insa-rouen.fr/~scanu`

[11] J. Mercer, Functions of positive and negative type and their connection with the theory of integral equations, *Transactions of the London Philosophical Society A* **209** (1909) 415–446.

[12] J. Neveu, *Processus aléatoires gaussiens*, Séminaires de mathématiques supérieures, Les presses de l'université de Montréal (1968).

[13] T. Poggio and F. Girosi, A theory of networks for approximation and learning, Technical Report AIM-1140 (1989).

[14] S. Saitoh, *Theory of reproducing kernels and its applications*, volume 189. Longman scientific and technical (1988).

[15] B. Schölkopf, A generalized representer theorem, Technical Report 2000-81, NeuroColt2 Technical Report Series (2000).

[16] L. Schwartz, Sous espaces hilbertiens d'espaces vectoriels topologiques et noyaux associés, *Journal d'Analyse Mathématique* (1964) 115–256.

[17] A.J. Smola and B. Schölkopf, From regularization operators to support vector kernels, In M.I. Jordan, M.J. Kearns, and S.A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. The MIT Press (1998).

[18] G.I. Targonski, On Carleman integral operators, *Proceedings of the American Mathematical Society* **18**(3) (1967) 450–456.

[19] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, N.Y (1995).

[20] G. Wahba, *Spline Models for Observational Data*, Series in Applied Mathematics, Vol. 59, SIAM, Philadelphia (1990).

Chapter 6

Leave-one-out Error and Stability of Learning Algorithms with Applications

André Elisseeff and Massimiliano Pontil[1]

Abstract. The leave-one-out error is an important statistical estimator of the performance of a learning algorithm. Unlike the empirical error, it is almost unbiased and is frequently used for model selection. We review attempts aiming at justifying the use of the leave-one-out error in Machine Learning. We especially focus on the concept of stability of a learning algorithm and show how this can be used to formally link the leave-one-out error to the generalization error. Stability has also motivated recent work on averaging techniques similar to bagging which we briefly summarize in the chapter. The ideas we develop are illustrated in some detail in the context of kernel-based learning algorithms.

[1]Alessio Ceroni read the chapter and made useful remarks.

6.1 Introduction

Assessing the performance of different learning algorithms on a dataset is at the core of Machine Learning. If this task is done properly, the best algorithm can be selected and the problem of generalization is partially solved. For that reason, many studies have focused on the design of different model selection strategies. As examples, we could cite techniques based on minimum description length [37], margin bounds [7], Bayesian evidence [31], metric structures [40], etc. A complete list would include many more techniques.

When it comes to practice, many practitioners use cross validation methods. This is remarkable from two related standpoints. First, cross validation has been seldom studied in the literature compared to some of the above mentioned techniques. Second, as described in [27], "In spite of the practical importance of this estimate [cross validation], relatively little is known about its theoretical properties".

In this chapter, we try to gather information about one particular instance of cross validation, namely the leave-one-out error, in the context of Machine Learning and mostly from stability considerations. By limiting the scope of this presentation, we hope to provide a consistent view albeit not complete as we are aware that we might miss relevant contributions.

Outline of the chapter

After a general discussion about the leave-one-out error (Section 6.2), we present different analyses trying to explain why and how this technique works well (Section 6.3). In particular we focus on concepts of stability which allow to derive probabilistic bounds on the generalization error of learning algorithms. The bounds say that the difference between the generalization error and the leave-one-out error is small when the algorithm is stable. This is discussed in Section 6.3 where we also present other attempts to justify the leave-one-out error. In Section 6.4 we illustrate the use of the leave-one-out error in the context of kernel machines, a family of learning algorithms which has gained great popularity over the past few years. In this case it is possible to derive estimates of the leave-one-out error which only require the knowledge of the machine trained once on the full dataset. In Section 6.5 we overview the use of leave-one-out error in learning problems other than classification and regression.

Notation

In the following, calligraphic font is used for sets and capital letters refer to numbers unless explicitly defined. Let $\mathcal{X}$ and $\mathcal{Y}$ be two Hilbert spaces and define $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. $\mathcal{X}$ is identified as the input space and $\mathcal{Y}$ as the output space. Given a learning algorithm A, we define $f_{\mathcal{D}}$ to be the solution of the algorithm when the training set $\mathcal{D} = \{z_i = (x_i, y_i),\ i = 1, \ldots, m\} \in \mathcal{Z}^m$ drawn i.i.d. from a distribution $\mathbb{P}$ is used. A is thus interpreted as a function from $\mathcal{Z}^m$ to $\mathcal{Y}^{\mathcal{X}}$ - the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$ - and we use the notation: $A(\mathcal{D}) = f_{\mathcal{D}}$. We denote by $\mathcal{D}^i$ the training set obtained by removing the point (x_i, y_i) from $\mathcal{D}$. $f_{\mathcal{D}}$ is sometimes denoted by f and $f_{\mathcal{D}^i}$ by f^i. The expectation with respect to (w.r.t.) the sample $\mathcal{D}$ is denoted by $\mathbf{E}_{\mathcal{D}}[\cdot]$.

For any point (x, y) and function f we denote by $\ell(f(x), y)$ the error made when $f(x)$ is predicted instead of y (ℓ is the loss function). We also sometimes write $\ell(f, z)$ instead, where $z = (x, y)$. The generalization error of f w.r.t. loss ℓ is

$$R_{\text{gen}}(f) = \mathbf{E}_z \ell(f, z)$$

where $\mathbf{E}_z \cdot$ denotes the expectation w.r.t. z. We define as well the *empirical error* w.r.t. the same loss function by

$$R_{\text{emp}}(f) = \frac{1}{m} \sum_{i=1}^{m} \ell(f, z_i).$$

We focus on regression, $\mathcal{Y} = \mathbb{R}$ and on binary classification, $\mathcal{Y} = \{-1, 1\}$. In the latter case we denote by $\theta(\cdot)$ the Heavyside function and let the *misclassification error* of function f be the generalization error of f w.r.t. loss $\theta(-yf(x))$.

6.2 General Observations about the Leave-one-out Error

It seems that the leave-one-out error (also called *deleted estimate* or *U-method*) has been defined in the late sixties/mid-seventies. It appears in different papers by Lachenbruch [29], Luntz and Brailovsky [30], Cover [10], and Stone [42]. For a learning algorithm A producing an outcome $f_{\mathcal{D}}$, it is defined as

Definition 1 (Leave-one-out error)

$$R_{\text{loo}}(f_{\mathcal{D}}) = \frac{1}{m} \sum_{i=1}^{m} \ell(f^i, z_i)$$

and is supposed to be an "almost" unbiased estimate of the generalization error of $f_{\mathcal{D}}$. Because it seems to share many properties with a technique called Jackknife introduced by Tukey [43, 35], it is worth pointing out that the leave-one-out error is different. The latter concerns indeed a learning algorithm whose output is computed on a point that has not been used during training. The former consists in using repeatedly the whole training set but one point, computing many estimators and combining them at the end. This combination should lead to a new estimator whose bias is supposed to be low. The Jackknife can be used to derive an estimate of the generalization error based on the empirical error but things are then more complicated than for the leave-one-out estimate[2] (see, e.g., [36]).

The first result that is generally presented to motivate the leave-one-out error is the following:

Theorem 1 (Luntz and Brailovsky [30]) *The leave-one-out estimate is almost unbiased in the following sense:*

$$\mathbf{E}_{\mathcal{D}} \left[R_{\text{loo}}(f_{\mathcal{D}}) \right] = \mathbf{E}_{\mathcal{D}'} \left[R_{\text{gen}}(f_{\mathcal{D}'}) \right] \tag{6.1}$$

where $\mathcal{D}'$ is set of $m - 1$ points in $\mathcal{Z}$.

[2]Note that we use leave-one-out error or leave-one-out estimate interchangeably. The term leave-one-out error estimate refers to an approximation of the leave-one-out error.

On average, the leave-one-out error should be relatively informative about the generalization error of the same algorithm when trained on $m-1$ points. In practice, it is generally admitted that removing one point of the training set does not change much the outcome of the algorithm. In theory, such an assumption cannot be made so easily. Consider $\mathcal{D}' \subset \mathcal{D}$ a training set of $m-1$ points. If we assume that $f_{\mathcal{D}} \approx f_{\mathcal{D}'}$ for all training sets $\mathcal{D}$, then $R_{\text{emp}}(f_{\mathcal{D}}) \approx R_{\text{loo}}(f_{\mathcal{D}})$ and using the leave-one-out estimate is roughly the same as using the empirical error. The latter however is well known to be highly biased and does not give a good indication on the generalization performance of the algorithm. The apparent contradiction in this reasoning could have been removed if we had defined a more precise definition of what we mean by stable. For simplicity, let us consider until the end of the section that the learning problem is a classification problem. We define:

Definition 2 (Hypothesis stability [14]) *The hypothesis stability $\beta(f_{\mathcal{D}})$ of a symmetric[3] learning algorithm A whose outcome is $f_{\mathcal{D}}$ is defined as:*[4]

$$\beta(f_{\mathcal{D}}) = \mathbf{E}_{\mathcal{D},z}\left[|\ell(f_{\mathcal{D}}, z) - \ell(f_{\mathcal{D}^i}, z)|\right]. \tag{6.2}$$

This definition captures a notion of stability that suits our purpose: if one point is removed, the difference in the outcome of the learning algorithm will be measured by the averaged absolute difference of the losses. We can now naturally relate the generalization error of $f_{\mathcal{D}}$ with the one of $f_{\mathcal{D}^i}$. We have indeed:

$$|\mathbf{E}_{\mathcal{D}}\left[R_{\text{gen}}(f_{\mathcal{D}}) - R_{\text{gen}}(f_{\mathcal{D}^i})\right]| = |\mathbf{E}_{\mathcal{D},z}\left[\ell(f_{\mathcal{D}}, z) - \ell(f_{\mathcal{D}^i}, z)\right]| \leq \beta(f_{\mathcal{D}})$$

so that the bias of the leave-one-out error is bounded by the hypothesis stability of the learning algorithm. Note that the hypothesis stability does not imply that the empirical error is close to the leave-one-out error.

The following example shows how to compute hypothesis stability for the $k-$nearest neighbor ($k-$NN) algorithm.

Example 1 (Hypothesis Stability of $k-$NN) *With respect to the classification loss, $k-NN$ is $\frac{k}{m}$ stable. This can be seen via symmetrization arguments. For sake of simplicity, we give here the proof for the $1-NN$ only. Let v_i be the neighborhood of z_i such that the closest point of the training set to any point in v_i is z_i. The nearest neighbor algorithm computes its output via the following equation (we assume here that the probability that x_i appears twice in the training set is negligible):*

$$f_{\mathcal{D}}(x) = \sum_{i=1}^{m} y_i 1_{x \in v_i}(x)$$

where 1_A is the indicator function of the set A. The difference between the losses $\ell(f_{\mathcal{D}}, z)$ and $\ell(f_{\mathcal{D}^i}, z)$ is then defined by the set v_i. Here we assume that ℓ is the classification loss. We have then:

$$\mathbf{E}_z[|\ell(f_{\mathcal{D}^i}, z) - \ell(f_{\mathcal{D}}, z)|] \leq \mathbb{P}(v_i).$$

[3] An algorithm is said to be symmetric if its outcome, $f_{\mathcal{D}}$, does not change when the elements of the training set are permuted.

[4] Note that, since the algorithm is symmetric, when averaging over $\mathcal{D}$, specifying which point is left out is irrelevant: the r.h.s of Eq. (6.2) is the same for all $i = 1, .., m$.

Note that v_i depends on $\mathcal{D}$. Now averaging over $\mathcal{D}$ we need to compute $\mathbf{E}_{\mathcal{D}}[\mathbb{P}(v_i)]$ which is the same for all i because the z_i are drawn i.i.d. from the same distribution. But, we have,

$$1 = \mathbf{E}_{\mathcal{D},z}\left[|f_{\mathcal{D}}(x)|\right] = \mathbf{E}_{\mathcal{D},z}\left[\left|\sum_{i=1}^{m} y_i \mathbf{1}_{x\in v_i}(x)\right|\right] = \mathbf{E}_{\mathcal{D},z}\left[\sum_{i=1}^{m} \mathbf{1}_{x\in v_i}(x)\right].$$

The last inequality comes from the fact that for fixed $\mathcal{D}$ and z, only one $\mathbf{1}_{x\in v_i}(x)$ is non-zero. We have then:

$$1 = \mathbf{E}_{\mathcal{D},z}\left[\sum_{i=1}^{m} \mathbf{1}_{x\in v_i}(x)\right] = m\mathbf{E}_{\mathcal{D}}\left[\mathbb{P}(v_i)\right].$$

So that: $\mathbf{E}_{\mathcal{D}}[\mathbb{P}(v_i)] \frac{1}{m}$. And finally, the $1-NN$ has a hypothesis stability bounded above by $1/m$.

Thus for nearest neighbor methods, the leave-one-out error is truly almost unbiased: for the 1-NN, the average difference between $f_{\mathcal{D}'}$ and $f_{\mathcal{D}}$ is bounded by $1/m$. This statement does not hold for all algorithms: when the algorithm is unstable, the use of the leave-one-out error is not recommended. Consider for instance a (quite stupid) algorithm whose outcome is the constant function 0 when m is odd and the constant function 1 when m is even. The leave-one-out estimate would then be totally wrong. One might object that this example is quite artificial but actually such instabilities might occur when the value of m is small. A hard margin support vector machine (SVM) separating 3 points in a two dimensional input space is unstable: if one point is removed, the outcome of the SVM usually changes a lot.

Another case where the leave-one-out estimate might not be recommended has been pointed out by Shao [41] who studied model selection with a linear regression method. Shao proved that, asymptotically when m tends to infinity, the probability to select the optimal model based on the leave-one-out error is not equal to one: model selection based on the leave-one-out estimate tends to choose unnecessarily large models. This failure is not the privilege of the leave-one-out error. The latter is indeed equivalent to other criterion that inherit as well from this inconsistency. These criterion are the Akaike information criterion [1] and Mallows's Cp [32], which are proved to be equivalent to the leave-one-out error in the particular setting of quadratic linear regression when m tends to infinity.

A common belief that should decrease the interest in the leave-one-out error is that it has a large variance: when different training sets are sampled from the same distribution, the variance of the leave-one-out error computed over these samplings is generally larger than 10-fold cross validation. This statement has been observed practically by Kohavi [26]. The latter also showed examples where the leave-one-out error fails completely due to the instability of the learning algorithm. Theoretically, the variance of the leave-one-out error has been computed in [44] (p.236) in the special case of a Gaussian noise and for linear models. It is shown that the variance of the leave-one-out error in this setting is equivalent to the variance of a hold-out estimate.[5]

[5]A hold-out estimate is an average of the errors computed on a set of data points that have not been used during learning. For these variance calculations to be valid, the number of points held out

At last, except for few special cases, the leave-one-out estimate is very time-consuming. For that reason, many works have tried to to derive easy-to-compute bounds. These bounds seem however to be specific to linear models. This rules out many algorithms for which the leave-one-out error will still be long to compute.

Despite all these handicaps, the leave-one-out estimate is used by many practitioners. It is generally believed to be a fairly good estimator albeit not the best and it has been used successfully for model selection (see for instance the work of Chapelle *et al.* [8]). All these facts might then seem contradictory. Following the no free lunch theorem [48], we are tempted to conclude like Goutte [20] that the leave-one-out error is as bad as any other estimator but that we have not found practical problems for which it completely fails. This observation might indicate that the leave-one-out estimate has a small bias in most practical cases. Coming back to stability considerations, this can be translated into the hypothesis that most methods that are currently used by practitioners are actually stable with respect to the removal of one point in the training set. Next section shows more precisely how stability and the leave-one-out error are related to generalization.

6.3 Theoretical Attempts to Justify the Use of the Leave-one-out Error

In the previous section we have mainly focused on basic facts that are observed from practice or that have been deduced from the classical statistical framework. In this section, we will present theoretical approaches that have been inspired by machine learning and which seem to have been directed by the need to understand why this estimate was better than empirical error.

6.3.1 Early work in non-parametric statistics

In the late seventies, Rogers, Devroye and Wagner wrote a series of papers about k-nearest neighbor ($k-$NN), local, histograms and potential rules. For $k-$NN and $k-$local rules (i.e. rules which compute their output from the k closest points in the training set), they derived exponential bounds on the probability that the leave-one-out error[6] deviates from the generalization error [14].

Theorem 2 (Devroye and Wagner [14])

$$\mathbb{P}\left(R_{\text{loo}}\left(k-NN\right) - R_{\text{gen}}\left(k-NN\right) > \epsilon\right) \leq 2e^{-m\epsilon^2/18} + 6e^{-m\epsilon^3/(108k(2+\gamma_d))} \tag{6.3}$$

where γ_d is the maximum number of distinct points in $\mathbb{R}^d$ which can share the same nearest neighbor.

must be greater than m minus the dimension of the input space. Under general conditions, it can be assumed that the training set is the same set used to compute the leave-one-out error. Note that in his book, Vapnik uses the term "moving control estimator" rather than leave-one-out estimate.

[6]Note that the leave-one-out error for $k-$NN is very easy to compute.

The capacity term also called VC-dimension (see [45] for an account about Vapnik Chervonenkis dimension and the theory that is derived) that generally occurs in statistical learning theory bounds is replaced here by a metric concept γ_d whose value seems quite difficult to compute (see [14]). This value does not depend on m but goes to infinity when d tends to infinity (it is lower bounded by the d). This means that such bound does not motivate the use of $k-$NN in infinite dimensional reproducing kernel Hilbert spaces, that is, $k-$NN used with kernels (see Section 6.4). Although a similar bound has been derived for potential function rules that include classifiers based on Parzen window density estimators (see [15] for more details), these results do not extend to all classifiers. It is therefore interesting to consider the following theorem valid for any classifier $f_{\mathcal{D}}$:

Theorem 3 (Devroye *et al.* [13, 38])

$$\mathbf{E}_{\mathcal{D}}\left[\left(R_{\text{loo}}\left(f_{\mathcal{D}}\right)-R_{\text{gen}}\left(f_{\mathcal{D}}\right)\right)^2\right] \leq \frac{1}{m}+\mathbb{P}\left(f_{\mathcal{D}} \neq f_{\mathcal{D}^i}\right). \tag{6.4}$$

In order to get a small variance, this bound suggests that the stability should be small. Since with Tchebytchev's inequality a bound on the variance induces a bound on the difference between the leave-one-out and the generalization error, it is straightforward to relate stability to generalization and to support the common intuition that if an algorithm is stable then its leave-one-out error is close to its generalization error. Such stability considerations have been reconsidered in the late nineties with the work of Kearns and Ron [25].

6.3.2 Relation to VC-theory

The work of Kearns and Ron aims at proving sanity check bounds for the leave-one-out error. They prove that the bounds derived for the leave-one-out error are at least as good as those derived for the empirical error. This may seem quite reassuring in the sense that theory does not say that the leave-one-out error is worse than the empirical error. Their result is based on the following definition of stability:

Definition 3 (Error Stability [25]) *We say that a deterministic (and symmetric) algorithm A has error stability (β_1, β_2) if:*

$$\mathbb{P}\left(\left|R_{\text{gen}}\left(f_{\mathcal{D}}\right)-R_{\text{gen}}\left(f_{\mathcal{D}^i}\right)\right| \geq \beta_2\right) \leq \beta_1.$$

They showed that this notion combined with another notion of "overestimating the empirical error" which controls how much the leave-one-out error overestimates the empirical error, leads to the following bound: $\forall \delta > 0$, with probability $1-\delta$ over the sampling of the training set, assuming that $f_{\mathcal{D}}$ is an algorithm minimizing the empirical error over a set of functions with VC dimension d, we have:

$$\left|R_{\text{loo}}\left(f_{\mathcal{D}}\right)-R_{\text{gen}}\left(f_{\mathcal{D}}\right)\right| \leq \left(8\sqrt{\frac{(d+1)(\ln(9m/d)+2)}{m}}\right)/\delta. \tag{6.5}$$

This bound is very similar to those that are derived for the empirical error [44]. Actually it is not so surprising since the analysis of Kearns and Ron is based on VC-theory. They

also showed that there exists an algorithm minimizing the empirical error over a set of VC dimension d for which the left-hand side of Eq. (6.5) is lower bounded in $\Omega(d/m)$. This statement shows that if a bound on the difference between the leave-one-out and the generalization error is found then it must be very specific on the algorithm or it will be like for the empirical error, in $\Omega(d/m)$.

Another related contribution to the theory of cross validation is by Holden [23]. Unfortunately, the results do not apply to the leave-one-out error and holds only for cross validation when the fold left out is sufficiently large.

It seems that so far the best success that theory has been able to achieve for the leave-one-out error has been met by the work of Devroye, Rogers and Wagner and by the notion of stability.

6.3.3 Stability

Recently, stability considerations have been revisited by Bousquet and Elisseeff [5] who proved exponential bounds for stable algorithms.

Definition 4 (Uniform stability [5]) *Let $f_{\mathcal{D}}$ be the outcome of a symmetric and deterministic learning algorithm. We define the uniform stability $\beta(f_{\mathcal{D}})$ with respect to a loss function ℓ by*

$$\beta(f_{\mathcal{D}}) = \sup_{\mathcal{D}} \|\ell(f_{\mathcal{D}}, .) - \ell(f_{\mathcal{D}^i}, .)\|_\infty \tag{6.6}$$

From this definition, it is possible to prove the following theorem:

Theorem 4 (Bousquet and Elisseeff [5]) *Let $f_{\mathcal{D}}$ be the outcome of an algorithm with uniform stability $\beta(f_{\mathcal{D}})$ with respect to a loss function ℓ such that $0 \leq \ell(f_{\mathcal{D}}, y) \leq M$, for all $y \in \mathcal{Y}$ and all set $\mathcal{D}$. Then, for any $m \geq 1$, and any $\eta \in (0, 1)$, the following bound holds with probability at least $1 - \eta$ over the random draw of the sample $\mathcal{D}$,*

$$\mathbb{P}\left(|R_{\text{gen}}\left(f_{\mathcal{D}}\right) - R_{\text{loo}}\left(f_{\mathcal{D}}\right)| \geq \beta_m + \epsilon\right) \leq e^{-2m\epsilon^2/(4m\beta(f_{\mathcal{D}})+M)^2} \tag{6.7}$$

The bound presented in this theorem is interesting only if the stability $\beta(f_{\mathcal{D}})$ decreases as $1/m^a$ with $a > 1/2$.

The uniform stability of regularization algorithms such as regularization networks [33] or support vector machines [4, 45] are computed in [5]. This extends the results of Devroye, Rogers and Wagner to other learning techniques but at the cost of a more restrictive definition of stability. Hypothesis stability is indeed upper bounded by uniform stability and classic algorithms such as $k-$NN do not have an interesting uniform stability with respect to the classification loss: it is equal to 1 for the 1−NN. It seems however that uniform stability is the cost to pay to get exponential bounds rather than bounds derived with Tchebytchev's inequality as it is the case for the hypothesis stability.

A less restrictive notion of stability that has been introduced lately by Kutin and Niyogi [28] might provide a better notion than uniform stability.

Definition 5 (Partial stability [28]) *Let $f_{\mathcal{D}}$ be the outcome of a symmetric and deterministic learning algorithm A. We say that A is (δ, β) partially stable with respect to a loss function ℓ if:*

$$\mathbb{P}(\forall i \in \{1, \ldots, m\},\ \|\ell(f_{\mathcal{D}}, .) - \ell(f_{\mathcal{D}^i}, .)\|_\infty \leq \beta) \geq 1 - \delta. \tag{6.8}$$

Kutin and Niyogi have derived bounds for the empirical error but it is possible to easily extend their results to bound the leave-one-out error as in (6.7), except that the uniform stability is replaced by the partial stability. We refer the reader to [28] for more details.

We see that there exist many notions of stability that lead to bounds on the generalization error in terms of the leave-one-out estimate. The choice of the stability measure is influenced by the desired type of bound (Tchebytchev's type with Hypothesis stability and exponential bounds with Uniform and partial stability) and the knowledge of the learning algorithm which will guide the computation of its stability. In next section, we show as an aside a way of improving the stability of a learning algorithm by averaging techniques.

6.3.4 Stability of averaging techniques

Suppose that a learning algorithm A has uniform stability $\beta_m = O(1/m^a), a > 0$. When $a = 1$ we say that A is stable. In this case Theorem 4 implies that

$$|R_{\text{gen}}(f_{\mathcal{D}}) - R_{\text{loo}}(f_{\mathcal{D}})| = O(1/\sqrt{m}).$$

If A is not stable, a way to stabilize it is by averaging its solution trained on small bootstrap subsets of the training set [17]. In particular, consider the function

$$F^k_{\mathcal{D}}(x) = \mathbf{E}_S[f_S(x)] \tag{6.9}$$

where $\mathbf{E}_S$ denotes the expectation with respect to k points sampled in $\mathcal{D}$ with the uniform distribution, i.e. S contains k points of $\mathcal{D}$ obtained by uniform sampling without replacement. When $k = m$ and sampling is done with replacement, this method is Breiman's bagging [6]. The next theorem provides a link between the uniform stability of the average combination and that of algorithm A.

Theorem 5 (Evgeniou *et al.* [17]) *Assume that $f_{\mathcal{D}}$ has stability β_m. Then, the stability, $\widehat{\beta}_m$ of the average combination F^k in Eq. (6.9) is bounded as*

$$\widehat{\beta}_m \leq \frac{k}{m}\beta_k.$$

This theorem implies that the average combination is always stable in the sense that $\widehat{\beta}_m = O(1/m)$. In practice the average combination is replaced by a finite combination. In this case the stability theory does not hold because of the randomization of the solution. Recent work in [16] extends the stability concepts to randomized algorithms and presents a closer look at averaging techniques.

6.4 Kernel Machines

In this section we focus on kernel machines [45, 18, 11, 39], a family of learning algorithms based on reproducing kernel Hilbert spaces. We begin by recalling the main features of kernel machines. Then, we present two sides that make them appealing when used with leave-one-out techniques: first, their leave-one-out error can be easily computed, second their stability can be (roughly) bounded.

6.4.1 Background on kernel machines

Kernel machines are the minimizers of regularization functionals of the form:

$$\frac{1}{m}\sum_{i=1}^{m}\ell(f(x_i),y_i)+\lambda\|f\|_K^2 \tag{6.10}$$

where f is a function $\mathcal{X}\to\mathbb{R}$ belonging to a reproducing kernel Hilbert space (RKHS) $\mathcal{H}_K$ defined by a symmetric and positive definite kernel $K:\mathcal{X}\times\mathcal{X}\to\mathbb{R}$, and $\|f\|_K^2$ is the norm of f in this space. More precisely, the RKHS is the completion of the span of the set $\{K(x,\cdot),x\in\mathcal{X}\}$ with the inner product induced by

$$\langle K(x,\cdot),K(t,\cdot)\rangle=K(x,t). \tag{6.11}$$

Thus, if $t_1,\ldots,t_n\in\mathcal{X}$, the function $\sum_{i=1}^n c_iK(t_i,x)$ belongs to the RKHS and its squared norm is $\sum_{i,j=1}^n c_ic_jK(t_i,t_j)$. In particular $\|K(x,\cdot)\|^2=K(x,x)$. The next fundamental property of $\mathcal{H}_K$ follows from Eq. (6.11):

$$f(x)=\langle f,K(x,\cdot)\rangle,\quad\text{for every } f\in\mathcal{H},\ x\in\mathcal{X}.$$

Using Cauchy-Schwartz inequality we obtain another important property of the RKHS:

$$|f(x)|=|\langle f,K(x,\cdot)\rangle|\le\|f\|_K\sqrt{K(x,x)}. \tag{6.12}$$

The RKHS norm is a measure of smoothness of the function, e.g. a Sobolev norm and, so, the regularization parameter $\lambda>0$ trades-off between small empirical error and smoothness of the solution. A correct choice of λ prevents from overfitting. In practice λ is selected by means of cross validation. RKHS are discussed in [3]. A nice introduction relevant to kernel machines can be found in [12]. For more information on positive definite kernels see [19].

The loss function ℓ is assumed to be convex. Its choice determines different learning techniques, each leading to a different learning algorithm. Important cases are regularization networks [33] and support vector machines (SVMs) [45]. They are summarized in Table 6.1 (left column). When $\mathcal{X}=\mathbb{R}^n$ and $K(x,t)$ is the Euclidean inner product and ℓ is the square loss, we have the older ridge regression technique [22].

Learning method	$\ell(f,y)$	$S(\alpha)$	Constraints
Regularization network	$(y-f)^2$	$y_i\alpha_i-\frac{1}{4C}\alpha_i^2$	NO
SVM classification	$(1-yf)_+$	α_i	$0\le\alpha_i\le C$
SVM regression	$(\lvert y-f\rvert-\epsilon)_+$	$y_i\alpha_i-\epsilon\lvert\alpha_i\rvert$	$\lvert\alpha_i\rvert\le C$

Table 6.1: Some kernel machines used in practice. Function $(z)_+$ equals z when $z>0$ and zero otherwise. The coefficients α_i and the function S are introduced in Equations (6.13), respectively.

It can be showns that the minimizer of (6.10) is unique and has the form

$$f(x)=\sum_{i=1}^{m}\alpha_iK(x_i,x). \tag{6.13}$$

The coefficients α_i in Eq. (6.13) are learned by solving the optimization problem:

$$\max_\alpha \left\{ \sum_{i=1}^m S(\alpha_i) - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j K_{ij} \right\} \tag{6.14}$$

where $S(\cdot)$ is a concave function whose form depends on ℓ, and we used the shorthand K_{ij} for $K(x_i, x_j)$. For SVM classification, $S(\alpha) = \alpha$ if $\alpha \in [0, C]$ and infinite otherwise. Here $C = 1/(2m\lambda)$. Thus, SVMs solve the following quadratic programming problem

$$\begin{aligned} &\max_\alpha \left\{ \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j K_{ij} \right\} \\ &\text{subject to : } 0 \leq \alpha_i \leq C, \ i = 1, \ldots, m. \end{aligned} \tag{6.15}$$

The points for which $\alpha_i > 0$ are called support vectors. They are those points for which $y_i f(x_i) \leq 1$[7].

In the case of the square loss, it can be easily verified that the coefficients α_i solve the linear system of equations

$$(K + \lambda m I)\alpha = \mathbf{y} \tag{6.16}$$

where $\mathbf{y} = (y_1, \ldots, y_m)$ and, abusing notation, we denoted by K the $m \times m$ matrix with elements $K_{ij} = K(x_i, x_j)$.

6.4.2 Leave-one-out error for the square loss

For regularization networks the α parameters are the solution of the linear system (6.16). It is then possible to use standard linear algebra results to obtain the following lemma.

Lemma 1 *Let f be the minimizer of the functional in Eq. (6.10), where we choose ℓ to be the square loss and f^i the minimizer of the same functional when point i is removed from the training set. Let $B = K(K + \lambda I)^{-1}$. Then*

$$y_i - f^i(x_i) = \frac{y_i - f(x_i)}{1 - B_{ii}}.$$

The interesting feature of this result is that it allows an exact computation of the leave-one-out error on the base of the solution obtained by training on the full dataset only. The following theorem is an immediate consequence of the lemma.

Theorem 6 *Under the same notation of Lemma 1, the leave-one-out error of a regularization network is*

$$\frac{1}{m} \sum_{i=1}^m (y_i - f^i(x_i))^2 = \frac{1}{m} \sum_{i=1}^m \left(\frac{y_i - f(x_i)}{1 - B_{ii}} \right)^2. \tag{6.17}$$

When the training set is large, computing B_{ii} may be time consuming. In this case it is convenient to use the generalized cross-validation approximation [46]

$$\frac{1}{m} \sum_{i=1}^m (y_i - f^i(x_i))^2 \approx \frac{1}{m} \sum_{i=1}^m \left(\frac{y_i - f(x_i)}{1 - \frac{\text{trace} B}{m}} \right)^2.$$

For more information on generalized cross-validation see [46].

[7] In some special cases, which rarely occur in practice, it may happen that $y_i f(x_i) = 1$ and $\alpha_i = 0$.

6.4.3 Bounds on the leave-one-out error and stability

When the loss function in Eq. (6.10) is different from the square loss, the leave-one-out error cannot be computed on the base of the solution trained on the full training set only. However, it is possible to derive an upper bound on the error which achieves this. The derivation is based on the following lemma:

Lemma 2 (Zhang [49]) *Suppose function S in (6.14) is concave. Let f be the solution of (6.14) and f^i the solution of the same problem when point i is removed from the training set. Then*

$$\|f - f^i\|_K \leq |\alpha_i| \sqrt{K(x_i, x_i)}.$$

Now, property (6.12) implies that

$$|f(x_i) - f^i(x_i)| \leq |\alpha_i| K(x_i, x_i). \tag{6.18}$$

This immediately gives:

Theorem 7 *Under the same hypothesis of Lemma 2, the leave-one-out error of a kernel machine which solves problem (6.14) is upper bounded as*

$$\frac{1}{m} \sum_{i=1}^{m} \ell(f^i(x_i), y_i) \leq \frac{1}{m} \sum_{i=1}^{m} \max_{|\mu| \leq 1} \ell(f(x_i) + \mu \alpha_i K(x_i, x_i), y_i).$$

In particular, for binary classification

$$\frac{1}{m} \sum_{i=1}^{m} \theta(-y_i f^i(x_i)) \leq \frac{1}{m} \sum_{i=1}^{m} \theta(|\alpha_i| K(x_i, x_i) - y_i f(x_i)).$$

Similar results were derived in [24, 8] for classification. In this case, last inequality says that a data point is counted as a leave-one-out error if it is either misclassified by f or if by removing its contribution to f changes the sign of classification. For SVMs, the number of leave-one-out errors is no more than the number of support vectors. However if $|\alpha_i| K(x_i, x_i) \ll 1$, support vectors which are correctly classified are likely not to be counted as leave-one-out errors. Thus, the leave-one-out error is close to the empirical error if $C\kappa \ll 1$, with $\kappa = \sup_x K(x, x)$. In this case the SVM is stable. Since $C \propto 1/\lambda$, increasing λ increases stability. This is something we could expect, because the larger λ the smoother the solution. The following result makes this intuition precise.

Theorem 8 (Bousquet and Elisseeff [5]) *The uniform stability of a kernel machine w.r.t the loss function ℓ is bounded as*

$$\beta_m \leq \frac{\kappa \sigma^2}{2m\lambda}$$

where $\kappa = \sup_{x \in \mathcal{X}} K(x, x)$, and σ is a constant such that $|\ell(f, y) - \ell(g, y)| \leq \sigma |f - g|$, for every $y \in \mathbb{R}$ and $f, g \in \mathcal{H}_K$.

Notice that the uniform stability of a kernel machines depends on how the regularization parameter scales with m. If λ is a constant, we have $\beta_m = O(1/m)$. Usually, however, λ decreases with m and the kernel machine is not stable. In this case averaging helps to improve stability as we discussed. Some experiments which verify this effect are presented in [2].

6.5 The Use of the Leave-one-out Error in Other Learning Problems

Leave-one-out error is useful to study other learning problems than the standard classification and regression ones. This section analyzes two important cases in the context of kernel-based learning algorithms.

6.5.1 Transduction

The transduction problem is discussed in [45]. Like in the supervised learning problem, we collect a training set $\mathcal{D}$, but our goal is now to compute only the outputs of a finite set $X = \{x'_1, \ldots, x'_\ell\} \subset \mathcal{X}$ (the unlabelled set). When the size of X is small this problem can be significantly different form the standard learning problem. An extreme case is $X = \{x'\}$. In this case we only need to compute the optimal output of x' as opposed to the problem of computing a function with optimal average performance on future data.

Let $\mathbf{y}' = (y'_1, \ldots, y'_\ell)$ be the output variables of the unlabelled set. [9] proposes to compute $\mathbf{y}'$ by minimizing the leave-one-out error of the set $\mathcal{T} = \mathcal{D} \cup \{(x'_1, y'_1), \ldots, (x'_\ell, y'_\ell)\}$, which we denote by $L(\mathbf{y}')$. [9] uses ridge regression with $m + \ell$ basis functions, each being a Gaussian centered on a data in $\mathcal{T}$. An advantage of this approach is that the leave-one-out error can be computed as in Eq. (6.17). Let $\mathbf{y}^0$ be the outputs assigned to the unlabeled set by means of standard ridge regression on the training set $\mathcal{D}$. We compute $\mathbf{y}'$ as to be the minimizer of

$$L(\mathbf{y}') + \gamma \|\mathbf{y}' - \mathbf{y}^0\|^2$$

where γ is a positive parameter. The second term serves as a regularization term which tends to favor solutions close to the standard ridge regression one. [9] validates this method on two datasets showing an improvement w.r.t. standard ridge regression.

We remark that a related but different problem is that of learning from partially labelled data. In this case the training set is formed of both labelled and unlabelled data and the goal is to compute a function which minimizes the generalization error. Such a system would be very useful in real applications where labelled data are an expensive commodity. A possibility would be to compute the outputs of the unlabelled points as before and then to retrain on the extended training set.

6.5.2 Feature selection and rescaling

In feature selection and rescaling, an additional set of parameters σ is introduced which multiplies the coordinate of x (we assume here $\mathcal{X} = \mathbb{R}^n$) and the goal is to compute the optimal value of these parameters. In feature selection $\sigma \in \{0, 1\}^n$, while in feature rescaling $\sigma \in \mathbb{R}^n$. We focus on the second problem. This also serves as an intermediate step to solve the first problem, since we can select the features whose computed scaling parameters are larger than a threshold.

[8] discusses feature rescaling with SVMs for binary classification. First the SVM is trained on the initial data. Second, the scaling parameters are updated by performing

a gradient step so that the leave-one-out error decreases. These two steps are then repeated till a minimum of the leave-one-out error is reached (usually a stopping criterion is used). An interesting finding of [8] was that rather simplified leave-one-out error bounds are sufficient for computing a good solution. In particular the bounds in Theorem 7 can be further upper bounded by $R^2\|f\|_K^2$, where R^2 is the maximum value of $K(x,x)$ among the set of support vectors. This has also the advantage that $R^2\|f\|_K^2$ can be differentiated exactly – see [8] for more details. [47] contains more experiments on the feature selection problem. [21] discusses a similar feature selection method which is applied to image classification problems.

Note that the leave-one-out bounds may be also useful to adapt/build a kernel function in a way similar to the feature selection problem. For example the kernel function could be a convex combination of some known kernel functions. The coefficients in the convex combination could be computed by means of the algorithm above, with an additional positivity constraint.

6.6 Discussion

We have reviewed existing results on the leave-one-out error in Machine Learning. In particular we discussed attempts aimed at relating the leave-one-out error to the generalization error. These attempts may or may not explain the success of the leave-one-out error in practical problems, but we hope at least they will motivate further studies on this interesting topic.

6.6.1 Sensitivity analysis, stability, and learning

Leave-one-out error and stability are a "specific" instance of the more general problem of studying how the solution of learning algorithm changes in response to some perturbations of its parameters.

The leave-one-out perturbation consists in removing one point from the training set. The results in Section 6.3 indicate that there is a general relation between the stability of a learning algorithm and its generalization error. Other kind of perturbations may be important to better understand the properties of the learning algorithm, both statistical and numerical. [34] studies how the SVM solution changes when the regularization parameter is modified but it would be also interesting to know the sensitivity of the solution w.r.t. modification of the outputs of the training points, kernel parameters, etc...

6.6.2 Open problems

We conclude with few open questions.

Question 1 (consistency of the leave-one-out error) We have seen that stability considerations provide, for a fixed algorithm, sufficient conditions for the leave-one-out error to tend to the generalization error. What are the general necessary conditions? The answer would give interesting insights into which

properties of an algorithm are required for its leave-one-out estimate to be close to its generalization error.

Question 2 (empirical stability) Can we use the leave-one-out error to say something practical about stability? In particular, is the difference between the leave-one-out error and the training error a good estimator of the "stability" of a learning algorithm? An answer to this question would also help to apply the stability bounds to other learning algorithms such as neural networks and decision trees where it seems difficult to compute hypothesis stability.

Question 3 (stability and invariances) Is there a way to incorporate prior knowledge via stability? For instance, would it help to add virtual inputs and to constrain the algorithm to be very stable on these new inputs, e.g. when one point is removed from the training set, the algorithm should give the same output on the virtual samples.

Bibliography

[1] H. Akaike, Information theory and an extension of the maximum likelihood principle, In B.N. Petrov and F. Csaki, editors, *2nd Int. Symp. on Inform. Theory*, Akademia Kiado, Budapest (1973) 267–281.

[2] S. Andonova, A. Elisseeff, T. Evgeniou, and M. Pontil, A simple algorithm for learning stable machines, In *15-th European Conference on Artificial Intelligence* (2002).

[3] N. Aronszajn, Theory of reproducing kernels, *Trans. Amer. Math. Soc.* **686** (1950) 337–404.

[4] B. Boser, I. Guyon, and V. Vapnik, A training algorithm for optimal margin classifiers, In *Fifth Annual Workshop on Computational Learning Theory*, Pittsburgh, ACM (1992) 144–152.

[5] O. Bousquet and A. Elisseeff, Stability and generalization, *J. of Machine Learning Res.* **2** (2002) 499–526.

[6] L. Breiman, Bagging predictors, *Machine Learning* **24**(2) (1996) 123–140.

[7] O. Chapelle and V. Vapnik, Model selection for support vector machines, In T.K. Leen S.A. Solla and K.-R. Muller, editors, *Advances in Neural Information Processing Systems 12*, MIT Press (2000).

[8] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, Choosing multiple parameters for support vector machines, *Machine Learning* **46**(1-3) (2002) 131–159.

[9] O. Chapelle, V. Vapnik, and J. Weston, Transductive inference for estimating values of functions. In T.K. Leen S.A. Solla and K.-R. Muller, editors, *Advances in Neural Information Processing Systems 12*, MIT Press (2000).

[10] T.M. Cover, Learning in pattern recognition, In S. Watanabe, editor, *Methodologies of Pattern Recognition*, Academic Press (1969) 111–132.

[11] N. Cristianini and J. Shawe-Taylor, *Introduction to Support Vector Machines*, Cambridge University Press (2000).

[12] F. Cucker and S. Smale, On the mathematical foundations of learning, *Bull. Amer. Math. Soc.* **39**(1) (2002) 1–49.

[13] L.P. Devroye and T.J. Wagner, Nonparametric discrimination and density estimation, Technical Report 183, Information Systems Research Laboratory, University of Texas, Austin (1976).

[14] L.P. Devroye and T.J. Wagner, Distribution-free inequalities for the deleted and holdout error estimates, *IEEE Trans. Inform. Theory* **25**(2) (1979) 202–207.

[15] L.P. Devroye and T.J. Wagner, Distribution-free performance bounds for potential function rules, *IEEE Trans. Inform. Theory* **25**(5) (1979) 601–604.

[16] A. Elisseeff, T. Evgeniou, and M. Pontil, Stability of randomized algorithms with an application to bootstrap methods, Preprint (2002).

[17] T. Evgeniou, M. Pontil, and A. Elisseeff, Leave one out error, stability, and generalization of voting combinations of classifiers, *Machine Learning* (2002) to appear.

[18] T. Evgeniou, M. Pontil, and T. Poggio, Regularization networks and support vector machines. *Advances in Computational Mathematics* **13** (2000) 1–50.

[19] C.H. FitzGerald, C. A. Micchelli, and A. Pinkus, Functions that preserves families of positive definite functions. *Linear Algebra and its Appl.* **221** (1995) 83–102.

[20] C. Goutte, Note on free lunches and cross-validation, *Neural Computation* **9**(6) (1997) 1245–1249.

[21] B. Heisele, T. Poggio, and M. Pontil, Face detection in still gray images, AI-Memo 1687, MIT (2000).

[22] A.E. Hoerl and R. Kennard, Ridge regression: Biased estimation for nonorthogonal problmes, *Technometrics* **12** (1970) 55–67.

[23] S.B. Holden, Pac-like upper bounds for the sample complexity of leave-one-out crossvalidation, In *Ninth Annual Conference on Computational Learning Theory*, ACM Press (1996) 41–50.

[24] T. Jaakkola and D. Haussler, Probabilistic kernel regression models, In *Proc. of the 7-th Int. Workshop on Artificial Intelligence and Statistics* (1999).

[25] M. Kearns and D. Ron, Algorithmic stability and sanity check bounds for leave-one-out cross validation bounds, *Neural Computation* **11**(6) (1999) 1427–1453.

[26] R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection, In *IJCAI* (1995) 1137–1145.

[27] S. Kulkarni, G. Lugosi, and S. Venkatesh, Learning pattern classification—a survey, *1948–1998 Special Commemorative Issue of IEEE Transactions on Information Theory* **44** (1998) 2178–2206.

[28] S. Kutin and P. Niyogi, Almost-everywhere algorithmic stability and generalization error, Technical report TR-2002-03, University of Chicago (2002).

[29] P.A. Lachenbruch, An almost unbiased method for the probability of misclassification in discriminant analysis, *Biometrics* **23** (1967) 639–645.

[30] A. Luntz and V. Brailovsky, On estimation of characters obtained in statistical procedure of recognition (in russian), *Technicheskaya Kibernetica* **3** (1969).

[31] D. MacKay, *Bayesian Methods for Adaptive Models*, PhD thesis (1992).

[32] C.L. Mallows, Some comments on Cp, *Technometrics* **15**(4) (1973) 661–675.

[33] T. Poggio and F. Girosi, Regularization algorithms for learning that are equivalent to multilayer networks, *Science* **247** (1990) 978–982.

[34] M. Pontil and A. Verri, Properties of support vector machines, *Neural Computation* **10** (1998) 955–974.

[35] M.H. Quenouille, Approximate tests of correlation in time-series, *J. Roy. Statist. Soc. B* **11** (1949) 68–84.

[36] B.D. Ripley, *Pattern Recognition and Neural Networks*, Cambridge University Press (1996).

[37] J. Rissanen, Modeling by shortest data description, *Automatica* **14** (1978) 465–471.

[38] W.H. Rogers and T.J. Wagner, A finite sample distributio-free preformance bound for local discrimination rule, *Annals of Statistics* **6** (1978) 506–514.

[39] B. Schölkopf and A. Smola, *Learning with Kernels*, The MIT Press, Cambridge, MA (2002).

[40] D. Schuurmans, A new metric-based approach to model selection, In *AAAI/IAAI* (1997) 552–558.

[41] J. Shao, Linear model selection by cross-validation, *J. Amer. Statist. Assoc.* **88** (1993) 486–494.

[42] M. Stone, Cross-validatory choice and assessment of statistical predictions (with discussion), *Journal of the Royal Statistical Society B* **36** (1974) 111–147.

[43] J.W. Tukey, Bias and confidence in not-quite large samples, *Annals of Math. Stat.* 29 (1958).

[44] V.N. Vapnik, *Estimation of Dependencies Based on Empirical Data*, Springer, N.Y. (1982).

[45] V.N. Vapnik, *Statistical Learning Theory*, John Wiley & Sons, N.Y. (1998).

[46] G. Wahba, Splines models for observational data, *Series in Applied Mathematics, SIAM, Philadelphia*, 59 (1990).

[47] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik, Feature selection for support vector machines, In T. Leen T. Dietterich and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*. MIT Press (2001).

[48] D.H. Wolpert and W.G. Macready, No free lunch theorems for search, Technical Report SFI-TR-95-02-010, Santa Fe, NM (1995).

[49] T. Zhang, A leave-one-out cross-validation bound for kernel methods with applications in learning, In *14th Annual Conference on Computational Learning Theory* (2001) 427–443.

Chapter 7

Regularized Least-Squares Classification

Ryan Rifkin, Gene Yeo and Tomaso Poggio

Abstract. We consider the solution of binary classification problems via Tikhonov regularization in a Reproducing Kernel Hilbert Space using the square loss, and denote the resulting algorithm Regularized Least-Squares Classification (RLSC). We sketch the historical developments that led to this algorithm, and demonstrate empirically that its performance is equivalent to that of the well-known Support Vector Machine on several datasets. Whereas training an SVM requires solving a convex quadratic program, training RLSC requires only the solution of a single system of linear equations. We discuss the computational tradeoffs between RLSC and SVM, and explore the use of approximations to RLSC in situations where the full RLSC is too expensive. We also develop an elegant leave-one-out bound for RLSC that exploits the geometry of the algorithm, making a connection to recent work in algorithmic stability.

7.1 Introduction

We assume that X and Y are two sets of random variables. We are given a **training set** $S = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)$, consisting of ℓ independent identically distributed samples drawn from the probability distribution on $X \times Y$. The joint and conditional probabilities over X and Y obey the following relation:

$$p(x, y) = p(y|x) \cdot p(x)$$

It is crucial to note that we view the joint distribution $p(x, y)$ as **fixed** but **unknown**, since we are only given the ℓ examples.

In this chapter, we consider the n-dimensional binary classification problem, where the ℓ training examples satisfy $\mathbf{x}_i \in \mathbb{R}^n$ and $y_i \in \{-1, 1\}$ for all i. Denoting the training set by S, our goal is to learn a function f_S that will generalize well on new examples. In particular, we would like

$$Pr(\mathrm{sgn}(f_S(\mathbf{x}))) \neq y$$

to be as small as possible, where the probability is taken over the distribution $X \times Y$, and $\mathrm{sgn}(f(\mathbf{x}))$ denotes the sign of $f(\mathbf{x})$. Put differently, we want to minimize the *expected risk*, defined as

$$I_{exp}[f] = \int_{X \times Y} i_{\mathrm{sgn}(f(x)) \neq y} dP,$$

where i_p denotes an indicator function that evaluates to 1 when p is true and 0 otherwise. Since we do not have access to $p(x, y)$, we cannot minimize $I_{exp}[f]$ directly. We instead consider the *empirical risk minimization* problem, which involves the minimization of:

$$I_{emp}[f] = \frac{1}{\ell} \sum_{i=1}^{\ell} i_{\mathrm{sgn}(f(\mathbf{x}_i)) \neq y_i}.$$

This problem is ill-defined, because we have not specified the set of functions that we consider when minimizing $I_{emp}[f]$. If we require that the solution f_S lie in a bounded convex subset of a Reproducing Kernel Hilbert Space H defined by a positive definite kernel function K [5, 6] (the norm of a function in this space is denoted by $||f||_K$), the regularized empirical risk minimization problem (also known as Ivanov Regularization) is well-defined:

$$\min_{f_S \in H, ||f_S||_K \leq R} \frac{1}{\ell} \sum_{i=1}^{\ell} i_{f(\mathbf{x}_i) \neq y_i}.$$

For Ivanov regularization, we can derive (probabilistic) generalization error bounds [7] that bound the difference between $I_{exp}[f_S]$ and $I_{emp}[f_S]$; these bounds will depend on H and R (in particular, on the VC-dimension of $\{f : f \in H \wedge ||f||_K^2 \leq R\}$) and ℓ. Since $I_{emp}[f]$ can be measured, this in turn allows us to (probabilistically) upper-bound $I_{exp}[f]$. The minimization of the (non-smooth, non-convex) 0-1 loss $i_{f(\mathbf{x}) \neq y}$ induces an NP-complete optimization problem, which motivates replacing $i_{\mathrm{sgn}(f(\mathbf{x})) \neq y}$ with a smooth, convex loss function $V(y, f(\mathbf{x}))$, thereby making the problem well-posed [1, 2, 3, 4]. If V *upper bounds* the 0-1 loss function, then the upper bound we derive on any $I_{exp}[f_S]$ with respect to V will also be an upper bound on $I_{exp}[f_S]$ with respect to the 0-1 loss.

In practice, although Ivanov regularization with a smooth loss function V is not necessarily intractable, it is much simpler to solve instead the closely related (via Lagrange multipliers) Tikhonov minimization problem

$$\min_{f \in H} \frac{1}{\ell} \sum_{i=1}^{\ell} V(y_i, f(\mathbf{x}_i)) + \lambda ||f||_K^2. \tag{7.1}$$

Whereas the Ivanov form minimizes the empirical risk subject to a bound on $||f||_K^2$, the Tikhonov form smoothly trades off $||f||_K^2$ and the empirical risk; this tradeoff is controlled by the regularization parameter λ. Although the algorithm does not explicitly include a bound on $||f||_K^2$, using the fact that the all-0 function $f(\mathbf{x}) \equiv 0 \in H$, we can easily show that the function f^* that solves the Tikhonov regularization problem satisfies $||f||_K^2 \leq \frac{B}{\ell}$, where B is an upper bound on $V(y, 0)$, which will always exist given that $y \in \{-1, 1\}$. This allows us to immediately derive bounds for Tikhonov regularization that have the same form as the original Ivanov bounds (with weaker constants). Using the notion of uniform stability developed by Bousquet and Elisseef [8], we can also more directly derive bounds that apply to Tikhonov regularization in an RKHS.

For our purposes, there are two key facts about RKHS's that allow us to greatly simplify (7.1). The first is the Representer Theorem [9, 10], stating that, under very general conditions on the loss function V, the solution f^* to the Tikhonov regularization problem can be written in the following form:

$$f^*(\mathbf{x}) = \sum_{i=1}^{\ell} c_i K(\mathbf{x}, \mathbf{x}_i).$$

The second is that, for functions in the above form,

$$||f||_K^2 = \mathbf{c}^T K \mathbf{c},$$

where K now denotes the ℓ-by-ℓ matrix whose (i, j)'th entry is $K(\mathbf{x}_i, \mathbf{x}_j)$.[1] The Tikhonov regularization problem becomes the problem of finding the c_i:

$$\min_{\mathbf{c} \in \mathbb{R}^\ell} \frac{1}{\ell} \sum_{i=1}^{\ell} V(y_i, \sum_{j=1}^{\ell} c_i K(\mathbf{x}_i, \mathbf{x}_j)) + \lambda \mathbf{c}^T K \mathbf{c}. \tag{7.2}$$

A specific learning scheme (algorithm) is now determined by the choice of the loss function V. The most natural choice of loss function, from a pure learning theory perspective, is the L_0 or misclassification loss, but as mentioned previously, this results in an intractable NP-complete optimization problem. The Support Vector Machine [7] arises by choosing V to be the hinge loss

$$V(y, f(\mathbf{x})) = \begin{cases} 0 & \text{if } yf(\mathbf{x}) \geq 1 \\ 1 - yf(\mathbf{x}) & \text{otherwise} \end{cases}$$

[1] This overloading of the term K to refer to both the kernel function and the kernel matrix is somewhat unfortunate, but the usage is clear from context, and the practice is standard in the literature.

The Support Vector Machine leads to a convex quadratic programming problem in ℓ variables, and has been shown to provide very good performance in a wide range of contexts.

In this chapter, we focus on the simple square-loss function

$$V(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2. \tag{7.3}$$

This choice seems very natural for regression problems, in which the y_i are real-valued, but at first glance seems a bit odd for classification — for examples in the positive class, large positive predictions are almost as costly as large negative predictions. However, we will see that empirically, this algorithm performs as well as the Support Vector Machine, and in certain situations offers compelling practical advantages.

7.2 The RLSC Algorithm

Substituting (7.3) into (7.2), and dividing by two for convenience, the RLSC problem can be written as:

$$\min F(\mathbf{c}) = \min_{\mathbf{c} \in \mathbb{R}^\ell} \frac{1}{2\ell} \sum_{i=1}^{\ell} (\mathbf{y} - K\mathbf{c})^T (\mathbf{y} - K\mathbf{c}) + \frac{\lambda}{2} \mathbf{c}^T K \mathbf{c}.$$

This is a convex differentiable function, so we can find the minimizer simply by taking the derivative with respect to $\mathbf{c}$:

$$\begin{aligned} \nabla F_{\mathbf{c}} &= \frac{1}{\ell}(\mathbf{y} - K\mathbf{c})^T(-K) + \lambda K \mathbf{c} \\ &= -\frac{1}{\ell} K\mathbf{y} + \frac{1}{\ell} K^2 \mathbf{c} + \lambda K \mathbf{c}. \end{aligned}$$

The kernel matrix K is positive semidefinite, so $\nabla F_{\mathbf{c}} = 0$ is a necessary and sufficient condition for optimality of a candidate solution $\mathbf{c}$. By multiplying through by K^{-1} if K is invertible, or reasoning that we only need a single solution if K is not invertible, the optimal K can be found by solving

$$(K + \lambda \ell I)\mathbf{c} = \mathbf{y}, \tag{7.4}$$

where I denotes an appropriately-sized identity matrix. We see that a Regularized Least-Squares Classification problem can be solved by solving a single system of linear equations.

Unlike the case of SVMs, there is no *algorithmic* reason to define the dual of this problem. However, by deriving a dual, we can make some interesting connections. We rewrite our problems as:

$$\begin{aligned} \min_{\mathbf{c} \in \mathbb{R}^\ell} \quad & \tfrac{1}{2\ell} \xi^T \xi + \tfrac{\lambda}{2} \mathbf{c}^T K \mathbf{c} \\ \text{subject to :} \quad & K\mathbf{c} - \mathbf{y} = \xi \end{aligned}$$

We introduce a vector of dual variables $\mathbf{u}$ associated with the equality constraints, and form the Lagrangian:

$$L(\mathbf{c}, \xi, \mathbf{u}) = \frac{1}{2\ell}\xi^T\xi + \frac{\lambda}{2}\mathbf{c}^T K\mathbf{c} - \mathbf{u}^T(K\mathbf{c} - \mathbf{y} - \xi).$$

We want to minimize the Lagrangian with respect to $\mathbf{c}$ and ξ, and maximize it with respect to $\mathbf{u}$. We can derive a "dual" by eliminating $\mathbf{c}$ and ξ from the problem. We take the derivative with respect to each in turn, and set it equal to zero:

$$\frac{\partial L}{\partial \mathbf{c}} = \lambda K\mathbf{c} - K\mathbf{u} = 0 \implies \mathbf{c} = \frac{\mathbf{u}}{\lambda} \tag{7.5}$$

$$\frac{\partial L}{\partial \xi} = \frac{1}{\ell}\xi + \mathbf{u} = 0 \implies \xi = -\ell\mathbf{u} \tag{7.6}$$

Unsurprisingly, we see that both $\mathbf{c}$ and ξ are simply expressible in terms of the dual variables $\mathbf{u}$. Substituting these expressions into the Lagrangian, we arrive at the reduced Lagrangian

$$\begin{aligned} L^R(\mathbf{u}) &= \frac{\ell}{2}\mathbf{u}^T\mathbf{u} + \frac{1}{2\lambda}\mathbf{u}^T K\mathbf{u} - \mathbf{u}^T(K\frac{\mathbf{u}}{\lambda} - \mathbf{y} + \ell\mathbf{u}) \\ &= -\frac{\ell}{2}\mathbf{u}^T\mathbf{u} - \frac{1}{2\lambda}\mathbf{u}^T K\mathbf{u} + \mathbf{u}^T\mathbf{y}. \end{aligned}$$

We are now faced with a differentiable concave maximization problem in $\mathbf{u}$, and we can find the maximum by setting the derivative with respect to $\mathbf{u}$ equal to zero:

$$\nabla L^R_{\mathbf{u}} = -\ell\mathbf{u} - \frac{1}{\lambda}K\mathbf{u} + y = 0 \Longrightarrow (K + \lambda\ell I)\mathbf{u} = \lambda\mathbf{y}.$$

After we solve for $\mathbf{u}$, we can recover $\mathbf{c}$ via Equation (7.5). It is trivial to check that the resulting $\mathbf{c}$ satisfies (7.4). While the exercise of deriving the dual may seem somewhat pointless, its value will become clear in later sections, where it will allow us to make several interesting connections.

7.3 Previous Work

The square loss function is an obvious choice for regression. Tikhonov and Arsenin [3] and Schönberg [11] used least-squares regularization to restore well-posedness to ill-posed regression problems. In 1988, Bertero, Poggio and Torre introduced regularization in computer vision, making use of Reproducing Kernel Hilbert Space ideas [12]. In 1989, Girosi and Poggio [13, 14] introduced classification and regression techniques with the square loss in the field of supervised learning. They used pseudodifferential operators as their stabilizers; these are essentially equivalent to using the norm in an RKHS. In 1990, Wahba [4] considered square-loss regularization for regression problems using the norm in a Reproducing Kernel Hilbert Space as a stabilizer.

More recently, Fung and Mangasarian considered *Proximal Support Vector Machines* [15]. This algorithm is essentially identical to RLSC in the case of the linear kernel, although the derivation is very different: Fung and Mangasarian begin with

the SVM formulation (with an unregularized bias term b, as is standard for SVMs), then modify the problem by penalizing the bias term and changing the inequalities to equalities. They arrive at a system of linear equations that is identical to RLSC up to sign changes, but they define the right-hand-side to be a vector of all 1's, which somewhat complicates the intuition. In the nonlinear case, instead of penalizing $\mathbf{c}^T K \mathbf{c}$, they penalize $\mathbf{c}^T \mathbf{c}$ directly, which leads to a substantially more complicated algorithm. For linear RLSC where $n << \ell$, they show how to use the Sherman-Morrison-Woodbury to solve the problem rapidly (see Section 7.4).

Suykens also uses the square loss in his *Least-Squares Support Vector Machines* [16], but allows the unregularized free bias term b to remain, leading to a slightly different optimization problem.

We chose a new name for this old algorithm, Regularized Least-Squares Classification, to emphasize both the key connection with regularization, and the fact RLSC is not a standard Support Vector Machine in the sense that there is no sparsity in the c vector. The connection between RLSC and SVM is that *both* are instances of Tikhonov regularization, with different choices of the loss function V.

7.4 RLSC vs. SVM

Whereas training an SVM requires solving a convex quadratic program, training RLSC requires only the solution of a single system of linear equations, which is conceptually much simpler. However, this does not necessarily mean that training RLSC is faster. To solve a general, nonlinear RLSC problem, we must first compute the K matrix, which requires time $O(\ell^2 n)$ and space $O(\ell^2)$.[2] We must then solve the linear system, which requires $O(\ell^3)$ operations. The constants associated with the O notation in RLSC are small and easily measured. The resource requirements of the SVM are not nearly so amenable to simple analysis. In general, convex quadratic programming problems are solvable in polynomial time [17], but the bounds are not tight enough to be practically useful. Empirically, Joachims [18] has observed a scaling of approximately $O(\ell^{2.1})$, although this ignores the dependence on the dimensionality n. Modern SVM algorithms are not amenable to direct formal analysis, and depend on a complex tradeoff between the number of data points, the number of Support Vectors, and the amount of memory available. However, as a rule of thumb, for large nonlinear problems, we expect the SVM to be much more tractable than the direct approach to the RLSC problem. In particular, for large problems, we can often solve the SVM problem in less time than it takes to compute every entry in K. Furthermore, if we do not have $O(\ell^2)$ memory available to store K, we cannot solve the RLSC problem at all via direct methods (we can still solve it using conjugate gradient methods, recomputing the kernel matrix K at every iteration, but this will be intractably slow for large problems). Even if we do manage to train the RLSC system, using the resulting function at test time will be much slower — for RLSC, we will have to compute all ℓ kernel products of training points with a new test point, whereas for SVM, we will only have to compute the kernel products with the support vectors, which are usually a small fraction of the training set.

[2] We make the assumption (satisfied by all frequently used kernel functions) that computing a single kernel product takes $O(n)$ operations.

In Section 7.6, we consider various approximations to the RLSC problem [19, 20, 21], but whether these approximations allow us to recover a high quality solution using an amount of time and memory equivalent to the SVM is an intriguing open question.

In the case of linear RLSC, the situation is very different. If we place our data in the ℓ-by-n matrix A, (7.4) becomes

$$(AA^T + \lambda \ell I)\mathbf{c} = \mathbf{y}.$$

If $n << \ell$, we can solve the problem in $O(\ell n^2)$ operations using either the Sherman-Morrison-Woodbury formula [15] or a product-form Cholesky factorization (suggested by Fine and Scheinberg in the context of interior point methods for SVMs [22]); the essential observation is that we can transform the rank ℓ problem into a rank n problem. If n is small compared to ℓ, this leads to a huge savings in computational cost. If n is large compared to ℓ, this technique is not helpful in general. However, in the important subcase where the data points are *sparse* (have very few nonzero elements per vector), for example in the case of a text categorization task, we can still solve the problem very rapidly using the Conjugate Gradient algorithm [23]. In particular, an explicit representation of $(AA^T + \lambda \ell I)$, which would require $O(\ell^2)$ storage, is not required — instead, we can form a matrix-vector product $(AA^T + \lambda \ell I)\mathbf{x}$ in $O(\ell \overline{n})$ operations, where $\overline{n}$ is the *average* number of nonzero entries per data point by first computing $\mathbf{i} = (\lambda \ell I)\mathbf{x}$ and $\mathbf{t} = A^T\mathbf{x}$, then using

$$(AA^T + \lambda \ell I)\mathbf{x} = A\mathbf{t} + \mathbf{i}.$$

As an example of the speed advantages offered by this technique, we compared SVMs and RLSCs trained using the Conjugate Gradient approach on the `20newsgroups` data set (described in Section 7.5 below) containing 15,935 data points. It took 10,045 seconds (on a Pentium IV running at 1.5 GhZ) to train 20 one-vs-all SVMs, and only 1,309 seconds to train the equivalent RLSC classifiers.[3]. At test time, both SVM and RLSC yield a linear hyperplane, so testing times are equivalent.

It is also worth noting that the same techniques that can be used to train linear RLSC systems very quickly can be used to train linear SVMs quickly as well. Using an interior point approach to the quadratic optimization problem [22], we can solve the SVM training problem as a sequence of linear systems of the form $(K + dI)\mathbf{x} = \mathbf{b}$, where d changes with each iteration. Each system can be solved using the techniques discussed above. In this case, the RLSC system will be faster than the SVM system by a factor of the number of systems the SVM has to solve (as the RLSC approach only needs to solve a single system); whether this approach to linear SVM will be faster than the current approaches is an open question.

7.5 Empirical Performance of RLSC

Fung and Mangasarian [15] perform numerous experiments on benchmark datasets from the UCI Machine Learning Repository [24], and conclude that RLSC performs as well as SVMs. We consider two additional, large-scale examples. The first example is

[3]The SVMs were trained using the highly-optimized SvmFu system (`http://fpn.mit.edu/SvmFu`), and kernel products were stored between different SVMs. The RLSC code was written in Matlab.

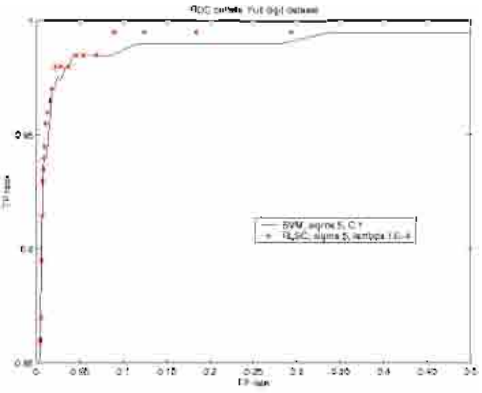


Figure 7.1: ROC curves for SVM and RLSC on the usps dataset.

	800		250		100		30	
	SVM	RLSC	SVM	RLSC	SVM	RLSC	SVM	RLSC
OVA	0.131	0.129	0.167	0.165	0.214	0.211	0.311	0.309
BCH 63	0.125	0.129	0.164	0.165	0.213	0.213	0.312	0.316

Table 7.1: A comparison of SVM and RLSC accuracy on the 20newsgroups multiclass classification task. The top row indicates the number of documents/class used for training. The left column indicates the multiclass classification scheme. Entries in the table are the fraction of misclassified documents.

the US Postal Service handwritten database (referred to here as usps), consisting of 7,291 training and 2,007 testing points; the task we consider is to discriminate images of the digit "4" from images of other digits. We first optimized the parameters (C and σ) for the SVM, then optimized the λ parameter of RLSC using the same σ ($\sigma = 5$). In Figure 7.1, we see that the full RLSC performed as well or better than the full SVM across the entire range of the ROC curve.

The second example is a pair of multiclass text categorization tasks, referred to as 20Newsgroups and Sector105. Rennie and Rifkin [25] used linear SVMs on these datasets, using both a one-vs-all and a BCH coding scheme for combining the binary classifiers into a multiclass classification system, obtaining the best published results on both datasets. The datasets were split into nested training and testing sets 10 times; details of the preprocessing scheme are available in [25]. Rifkin [21] used linear RLSC with $\lambda\ell = 1$ (solved using Conjugate Gradient) instead of SVMs; the experimental setup is otherwise identical. Tables 7.1 and 7.2 give the average accuracy of the SVM and RLSC schemes, for differing numbers of training documents per class. In all cases, the accuracy of the RLSC scheme is essentially identical to that of the SVM scheme; we note again that these results are better than all previously published results on these datasets. Given that linear RLSC can be trained very fast, it seems that linear RLSC shows great promise as a method of choice for very large-scale text categorization problems.

	52		20		10		3	
	SVM	RLSC	SVM	RLSC	SVM	RLSC	SVM	RLSC
OVA	0.072	0.066	0.176	0.169	0.341	0.335	0.650	0.648
BCH 63	0.067	0.069	0.176	0.178	0.343	0.344	0.653	0.654

Table 7.2: A comparison of SVM and RLSC accuracy on the `sector105` multiclass classification task. The top row indicates the number of documents/class used for training. The left column indicates the multiclass classification scheme. Entries in the table are the fraction of misclassified documents.

7.6 Approximations to the RLSC Algorithm

Although the RLSC algorithm is conceptually simple and yields high accuracy, for general nonlinear classification, it is often many orders of magnitude slower than an SVM. For this reason, we explore here the idea of solving approximations to RLSC. Hopefully, these approximations will yield a vast increase in speed with very little loss in accuracy.

Several authors have suggested the use of low-rank approximations to the kernel matrix in order to avoid explicit storage of the entire kernel matrix [19, 26, 20, 27, 22]. These techniques can be used in a variety of methods, including Regularized Least Squares Classification, Gaussian Process regression and classification, and interior point approaches to Support Vector Machines.

The approaches all rely on choosing a subset of m of the training points (or a subset of size m, abusing notation slightly) representing those points exactly in the Hilbert space, and representing the remaining points approximately as a linear combination of the points in m. The methods differ in the approach to choosing the subset, and in the matrix math used to represent the hypothesis. Both Smola and Schölkopf [19] and Williams and Seeger [20] suggest the use of the approximate kernel matrix

$$\widetilde{K} = K_{\ell m} K_{mm}^{-1} K_{m\ell}.$$

The assumption is that m is small enough so that K_{mm} is invertible. If we use a method that does not need the entries of $\widetilde{K}$, but only needs to multiply vectors by $\widetilde{K}$, we can form the matrix-vector produce by taking advantage of the $\widetilde{K}$'s representation as three separate matrices:

$$\begin{aligned} \widetilde{K}\mathbf{x} &= (K_{\ell m} K_{mm}^{-1} K_{m\ell})\mathbf{x} \\ &= (K_{\ell m}(K_{mm}^{-1}(K_{m\ell}\mathbf{x}))). \end{aligned}$$

This approximation is known as the *Nyström approximation* and is justified theoretically in [20]. If we approximate the kernel matrix using a subset of size m, we can show that we are actually approximating the first m eigenfunctions of the integral operator induced by the kernel (evaluated at all the data points). The quality of this approximation will of course depend on the rate of decay of the eigenvalues of the kernel matrix.

The two sets of authors differ in their suggestions as to how to choose the subset. Williams and Seeger simply pick their subset randomly. Smola and Schölkopf suggest

a number of greedy approximation methods that iteratively reduce some measure of the difference between K and $\widetilde{K}$, such as the trace of $K - \widetilde{K}$. They also suggest approximations to the full greedy approach in order to speed up their method. However, all their methods are likely to involve computing nearly all of the entries of K over the course of their operation (for reasonable subset sizes), implying that forming a matrix approximation in this manner will already be slower than training an SVM.

It is easy to show that (however m is selected) $\widetilde{K}$ has the property that $\widetilde{K}_{mm} = K_{mm}$, $\widetilde{K}_{m\ell} = K_{m\ell}$, and (trivially by symmetry) $\widetilde{K}_{\ell m} = K_{\ell m}$. In other words, the approximated kernel product $\widetilde{K}(\mathbf{x_1}, \mathbf{x_2})$ for a pair of examples $\mathbf{x_1}$ and $\mathbf{x_2}$ will be exact if either of $\mathbf{x_1}$ or $\mathbf{x_2}$ are in m. It is also easy to show that $\widetilde{K}_{nn} = K_{\ell m} K_{mm}^{-1} K_{m\ell}$.

Inverting the matrix K_{mm} explicitly takes $O(m^3)$ steps formally, and in practice, performing the inversion may be ill-conditioned, so this approach does not not seem to be a good choice for real-world applications.

Fine and Scheinberg [22] instead suggest the use of an incomplete Cholesky factorization. Noting that a positive semidefinite kernel matrix can always be represented as $K = R^T R$ where R is an upper-triangular matrix, the incomplete Cholesky factorization is formed by using only the first m rows of R. Fine and Scheinberg cleverly choose the subset on the fly — at each iteration, they pivot a pair of rows of the matrix so that the largest diagonal element becomes the next element in the subset. The total running time of their approach is only $O(mk^2)$.[4] In terms of the feature space, this pivoting technique is equivalent to, at each step, adding to the subset the data point which is most poorly represented by the current subset. Although the final matrix contains only m nonzero rows, it is still upper-triangular, and we write

$$\widetilde{K} = R_m^T R_m.$$

However, there is a serious problem with this approach. Although Fine and Scheinberg claim that their method "takes the full advantage of the greedy approach for for the best reduction in the approximation bound $\mathrm{tr}(\Delta Q)$", this is not the case. To reduce the approximation bound, we need to consider which element not in the basis will best represent (the currently unrepresented portion of) all remaining non-basis elements. This is what Smola and Schölkopf attempt in [19], at too large a computational cost. The Fine and Scheinberg approach, in contrast, adds to the basis the element which is most poorly represented by the current basis elements. If we believe that the trace of $K - \widetilde{K}$ is a useful measure of the quality of the approximation, this turns out to be a very poor choice, at least for Gaussian kernels. In particular, on two different datasets (see Section 7.6.2), we find that the portion of the trace accounted for by the Fine and Scheinberg approximation is consistently smaller than the portion of the trace accounted for by selecting the subset at random. This is not too hard to explain. Because the Fine and Scheinberg approach adds the most poorly represented element to the basis, under the Gaussian kernel, it will tend to add outliers — data points that are far from any other data point. The trace would be reduced much more by adding elements which are not as far away, but which are themselves close to a large number

[4] We have not formally defined "operations" (is a multiplication more expensive than an addition or a memory access?), but it is clear that the constant involved in this approach is small compared to the methods suggested in [19].

of additional points. Speaking informally, we want to add the points which are centers of clusters to the basis, not point which are outliers.

We now consider the use of a low-rank approximation, obtained by any of the above means, in RLSC.

7.6.1 Low-rank approximations for RLSC

The most obvious approach (and indeed, the one suggested in [20] in the context of Gaussian process classification and [22] in the context of Support Vector Machine classification) is to simply use the matrix $\widetilde{K}$ in place of the original K, resulting in the system of linear equations:

$$(\widetilde{K} + \lambda \ell I) = \mathbf{y}.$$

These equations can be solved using the conjugate gradient method, taking advantage of the factorization of $\widetilde{K}$ to avoid having to store an ℓ-by-ℓ matrix. From a machine learning standpoint, this approach consists of taking a subset of the points, estimating the kernel values at the remaining points, then treating the estimate as correct and solving the original problem over all the data points. In practice, we found that this worked quite poorly, because the matrix eigenvalues do not decay sufficiently quickly (see Section 7.6.2).

Instead, we consider the following modified *algorithm*, alluded to (among other places) in [19]. We minimize the empirical risk over all points, but allow the c_i to be nonzero only at a specific subset of the points (identical to the points used in the low-rank kernel approximation). This leads to the following modified problem:

$$\begin{aligned}
&\min F(\mathbf{c_m}) \\
&= \min_{\mathbf{c_m} \in \mathbb{R}^m} \quad \frac{1}{\ell}(\mathbf{y} - K_{\ell m}\mathbf{c_m})^T(\mathbf{y} - K_{\ell m}\mathbf{c_m}) + \lambda \mathbf{c_m}^T K \mathbf{c_m} \\
&= \min_{\mathbf{c_m} \in \mathbb{R}^m} \quad \frac{1}{2\ell}\left(\mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T K_{\ell m}\mathbf{c_m} + \mathbf{c}^T K_{m\ell}K_{\ell m}\mathbf{c}\right) + \frac{\lambda}{2}\mathbf{c_m}^T K \mathbf{c_m}.
\end{aligned}$$

We take the derivative with respect to $\mathbf{c_m}$ and set it equal to zero:

$$\begin{aligned}
&\nabla F_{\mathbf{c_m}} = \frac{1}{\ell}\left(K_{m\ell}\mathbf{y} + K_{m\ell}K_{\ell m}\mathbf{c_m}\right) + \lambda K_{mm}\mathbf{c_m} = 0 \\
\implies \quad &(K_{m\ell}K_{\ell m} + K_{mm}\lambda\ell)\mathbf{c_m} = K_{m\ell}\mathbf{y}.
\end{aligned}$$

We see that when we only allow a subset of size m points to have nonzero coefficients in the expansion, we can solve a m by m system of equations rather than an ℓ-by-ℓ system. As with the standard full RLSC, we were able to find the system of equations that defines the optimal solution by setting the derivative equal to zero. Again, suppose for the sake of argument we decide to derive a "dual" problem. We introduce a vector of dual variables $\mathbf{u}$ — it is important to note that this vector is of length ℓ, not m. We form the Lagrangian:

$$L(\mathbf{c_m}, \xi, \mathbf{u}) = \frac{1}{2\ell}\xi^T\xi + \frac{\lambda}{2}\mathbf{c_m}^T K_{mm}\mathbf{c_m} - \mathbf{u}^T(K_{\ell m}\mathbf{c_m} - \mathbf{y} - \xi).$$

We take the derivative with respect to ξ:

$$\begin{aligned} \frac{\partial L}{\partial \xi} &= \frac{1}{\ell}\xi + \mathbf{u} = 0 \\ \Longrightarrow \quad \xi &= -\ell \mathbf{u} \end{aligned}$$

We take the derivative with respect to $\mathbf{c_m}$:

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{c_m}} &= \lambda K_{mm}\mathbf{c_m} - K_{m\ell}\mathbf{u} = 0 \\ \Longrightarrow \quad \mathbf{c_m} &= \frac{1}{\lambda}K_{mm}^{-1}K_{m\ell}\mathbf{u}. \end{aligned}$$

Substituting these equations into L, we reduce the Lagrangian to:

$$\begin{aligned} & L(\mathbf{u}) \\ = \quad & \frac{\ell}{2}\mathbf{u}^T\mathbf{u} + \frac{1}{2\lambda}\mathbf{u^T}K_{\ell m}K_{mm}^{-1}K_{m\ell}\mathbf{u} - \frac{1}{\lambda}\mathbf{u^T}K_{\ell m}K_{mm}^{-1}K_{m\ell}\mathbf{u} + \mathbf{u}^T\mathbf{y} - \ell\mathbf{u}^T\mathbf{u} \\ = \quad & \frac{\ell}{2}\mathbf{u}^T\mathbf{u} + \frac{1}{2\lambda}\mathbf{u^T}\widetilde{K}\mathbf{u} - \frac{1}{\lambda}\mathbf{u^T}\widetilde{K}\mathbf{u} + \mathbf{u}^T\mathbf{y} - \ell\mathbf{u}^T\mathbf{u} \\ = \quad & -\frac{\ell}{2}\mathbf{u}^T\mathbf{u} - \frac{1}{2\lambda}\mathbf{u^T}\widetilde{K}\mathbf{u} + \mathbf{u}^T\mathbf{y}. \end{aligned}$$

Setting the derivative to zero yields

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{u}} &= -\ell\mathbf{u} - \frac{1}{\lambda}\widetilde{K}\mathbf{u} + \mathbf{y} = 0 \\ \Longrightarrow \quad (\widetilde{K} &+ \lambda\ell I)\mathbf{u} = \lambda\mathbf{y} \end{aligned}$$

If we solve this system for $\frac{\mathbf{u}}{\lambda}$, *we are solving exactly the same system of equations as if we had used the Nyström approximation at the subset m directly.* However, in order to find the optimal solution, we need to recover the vector $\mathbf{c_m}$ using the equation

$$\mathbf{c_m} = \frac{1}{\lambda}K_{mm}^{-1}K_{m\ell}\mathbf{u}. \tag{7.7}$$

To summarize, we suggest that instead of using the matrix $\widetilde{K}$ directly in place of K, we consider a modified problem in which we require our function to be expressed in terms of the points in the subset m. This leads to an algorithm that doesn't directly involve $\widetilde{K}$, but uses only the component matrices K_{mm}, $K_{\ell m}$ and $K_{m\ell}$. Although it is not strictly necessary, we can take the Lagrangian dual of this problem, at which point we solve a system that is identical to the original, full RLSC problem with K replaced with $\widetilde{K}$. However, we do not use the resulting $\mathbf{u}$ vector directly, instead recovering the c_m by means of (7.7).

7.6.2 Nonlinear RLSC application: image classification

To test the ideas discussed above, we compare various approaches to nonlinear RLSC on two different datasets. The first dataset is the US Postal Service handwritten

database, used in [20] and communicated to us by the authors, and referred to here as `usps`. This dataset consists of 7,291 training and 2,007 testing points; the task we consider is to discriminate images of the digit "4" from images of other digits. The training set contains 6,639 negative examples and 652 positive examples. The testing set contains 1,807 negative examples and 200 positive example. The second data set is a face recognition data set that has been used numerous times at the Center for Biological and Computational Learning at MIT [28, 29], referred to here as `faces`. The training set contains 2,429 faces and 4,548 non-faces. The testing set contains 472 faces and 23,573 non-faces.

Although RLSC on the entire dataset will produce results essentially equivalent to those of the SVM [21], they will be substantially slower. We therefore turn to the question of whether an *approximation* to RLSC results can produce results as good as the full RLSC algorithm. We consider three different approximation methods. The first method, which we call `subset`, involves merely selecting (at random) a subset of the points, and solving a reduced RLSC problem on only these points. The second method, which we call `rectangle`, is the primary algorithm discussed in Section 7.6.1: we choose a subset of the points, allow only those points to have nonzero coefficients in the function expansion, but minimize the loss over all points simultaneously. The third method is the `Nyström` method, also discussed briefly in Section 7.6.1 and presented more extensively in [20] (and in a slightly different context in [22]): in this method we choose a subset of the points, use those points to approximate the entire kernel matrix, and then solve the full problem using this approximate kernel matrix. In all experiments, we try four different subset sizes (1,024, 512, 256, and 128 for the `usps` dataset, and 1,000, 500, 250 and 125 for the `faces` dataset), and the results presented are the average of ten independent runs.

The results for the `usps` data set are given in Figure 7.2. We see that `rectangle` performs best, followed by `subset`, followed by `nystrom`. We suggest in passing that the extremely good results reported for `nystrom` in [20] may be a consequence of looking only at the error rate (no ROC curves are provided) for a problem with a highly skewed distribution (1,807 negative examples, 200 positive examples). For `rectangle` performance is very good at both the 1,024 and 512 sizes, but degrades noticeably for 256 samples. The results for the `faces` dataset, shown in Figure 7.3, paint a very similar picture. Note that the overall accuracy rates are much lower for this dataset, which contains many difficult test examples.

In all the experiments described so far, the subset of points was selected at random. Fine and Scheinberg [22] suggests selecting the points iteratively using an optimal greedy heuristic: at each point, the example is selected which will minimize the trace of the difference between the approximate matrix and the true matrix. Because the method simply amounts to running an incomplete Cholesky factorization for some number of steps (with pivoting at each step), we call this method `ic`. As mentioned earlier, if we believe that smaller values of $\mathrm{tr}(K - \widetilde{K})$ are indicative of a better approximation, this method appears to produce a worse approximation than choosing the subset at random (see Figure 7.4 and Figure 7.5). As pointed out by Scheinberg in personal communication, a smaller value of $\mathrm{tr}(K - \widetilde{K})$ is not necessarily indicative of a better matrix for learning, so we conducted experiments comparing `ic` to choosing a subset of the data at random randomly. Figure 7.6 shows the results for the `usps`

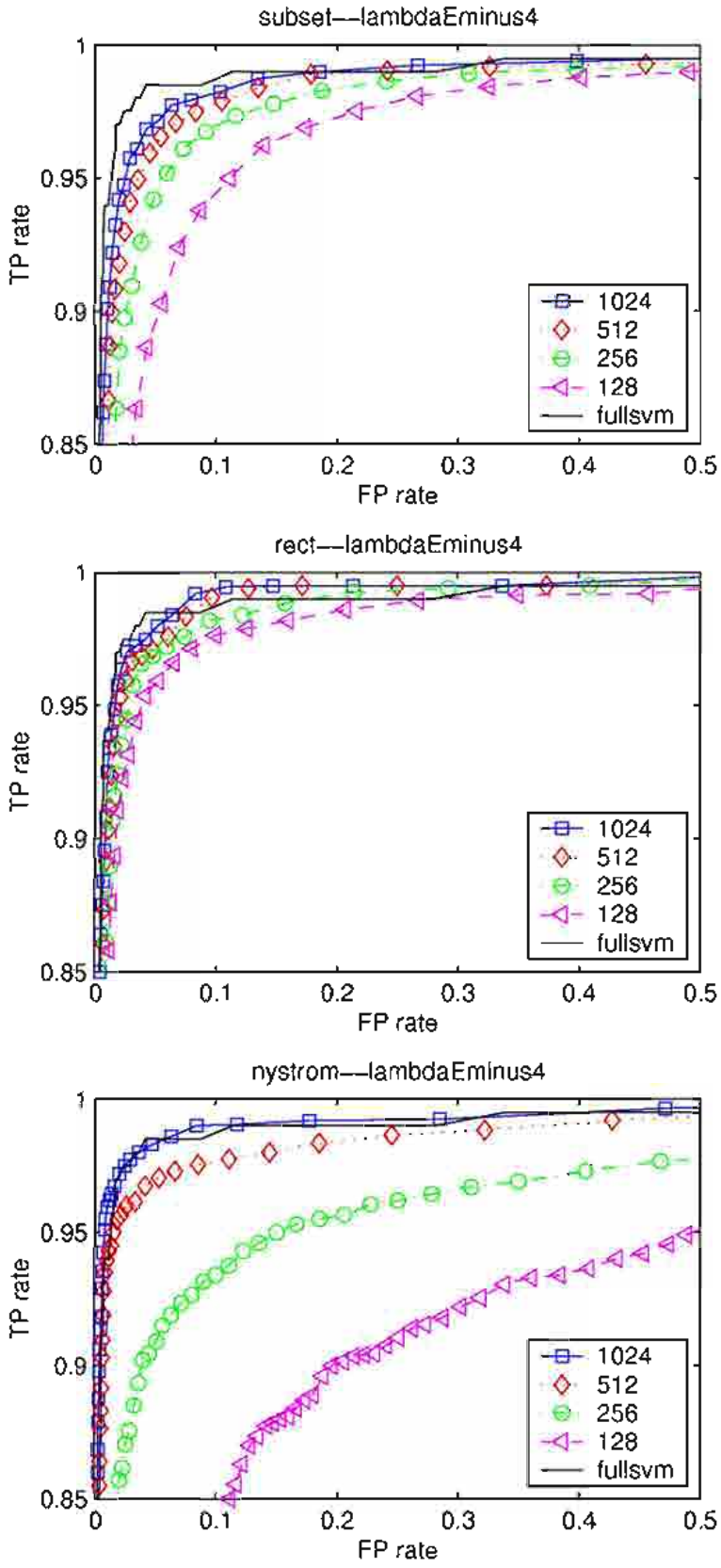


Figure 7.2: A comparison of the `subset`, `rectangle` and `nystrom` approximations to RLSC on the `usps` dataset. Both SVM and RLSC used $\sigma = 5$.

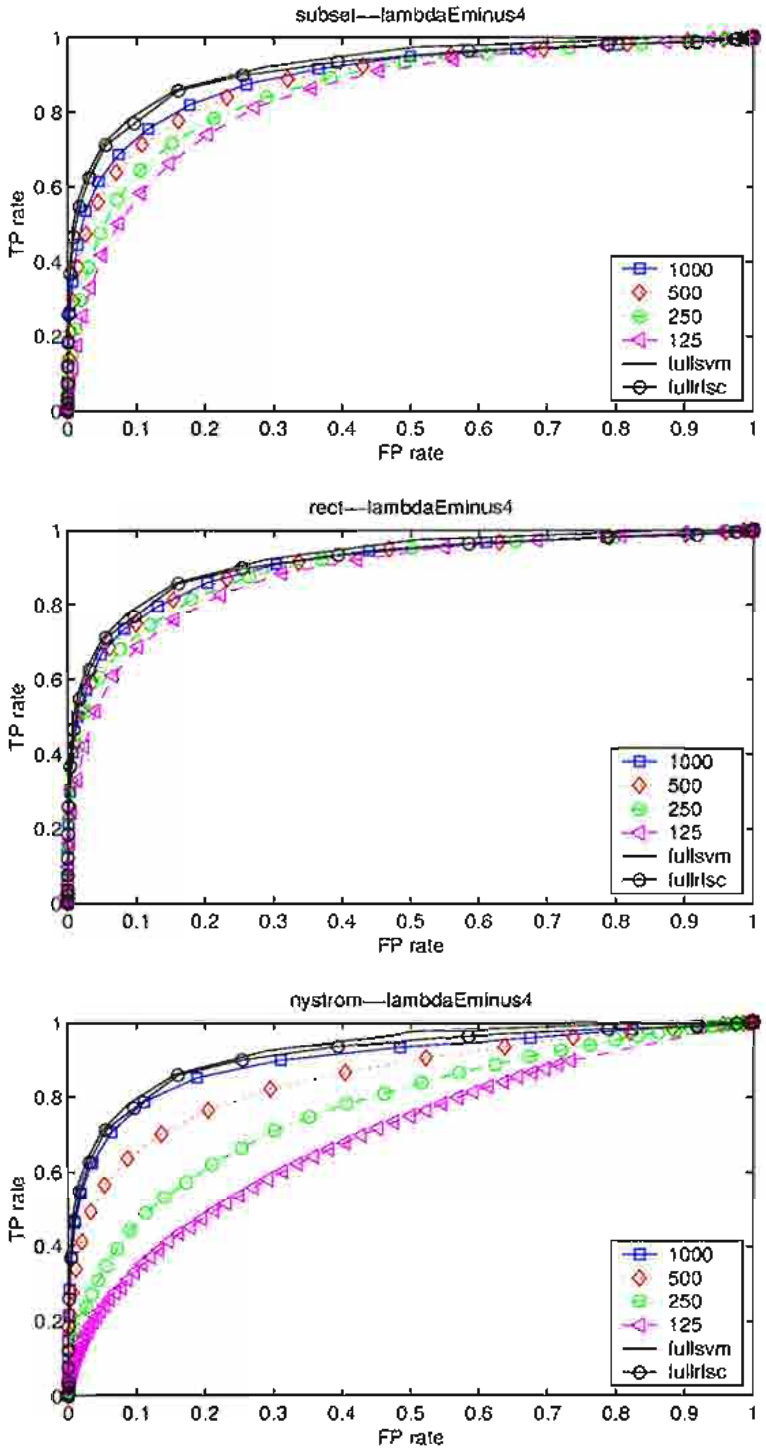


Figure 7.3: A comparison of the `subset`, `rectangle` and `nystrom` approximations to RLSC on the `faces` dataset. SVM used $\sigma = 5$, RLSC used $\sigma = 2$.

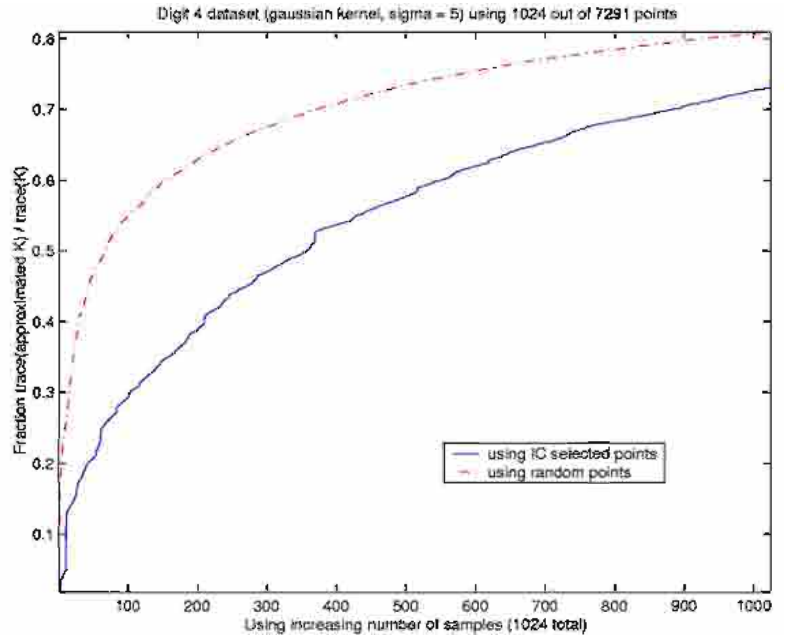


Figure 7.4: A comparison between the portion of the trace of K accounted for by using a subset of the points chosen using `ic` and a random subset, for the `usps` dataset.

data, and Figure 7.7 shows the results for the `faces` data. In all these experiments, we compare the average performance of the ten classifiers trained on a random subset to the performance of a single classifier trained on the subset of the same size obtained using the `ic` method. In these experiments, it does not seem that selecting the "optimal" subset using the `ic` method improves performance on learning problems. In the interests of computational efficiency, we may be better off simply choosing a random subset of the data.

The experiments contained in this section are somewhat preliminary. In particular, although using subsets of size 128 is faster than running the SVM, using subsets of size 1024 are already slower. Further experiments in which both techniques are optimized more thoroughly are necessary to fully answer the question of whether these approximations represent a viable approach to solving large-scale optimization problems.

7.7 Leave-one-out Bounds for RLSC

Leave-one-out bounds allow us to estimate the generalization error of a learning system when we do not have access to an independent test set. In this section, we define f_S to be the function obtained when the entire data set is used, and f_{S^i} to be the function obtained when the ith data point is removed and the remaining $\ell - 1$ points are used. We use G to denote $(K + \lambda \ell I)^{-1}$. Classical results (for example see [30]) yield the following exact computation of the leave-one-out value:

$$y_i - f_{S^i}(\mathbf{x}_i) = \frac{y_i - f_S(\mathbf{x}_i)}{1 - G_{ii}}.$$

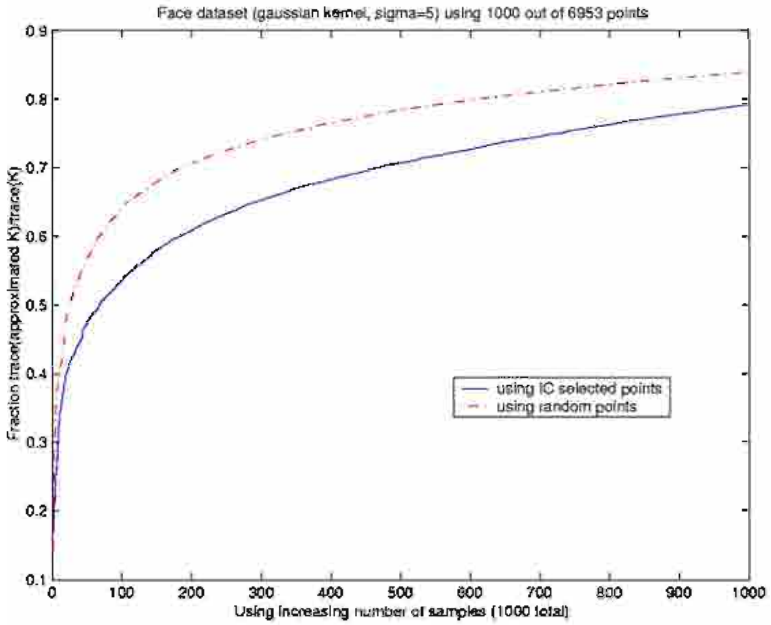


Figure 7.5: A comparison between the portion of the trace of K accounted for by using a subset of the points chosen using `ic` and a random subset, for the `faces` dataset.

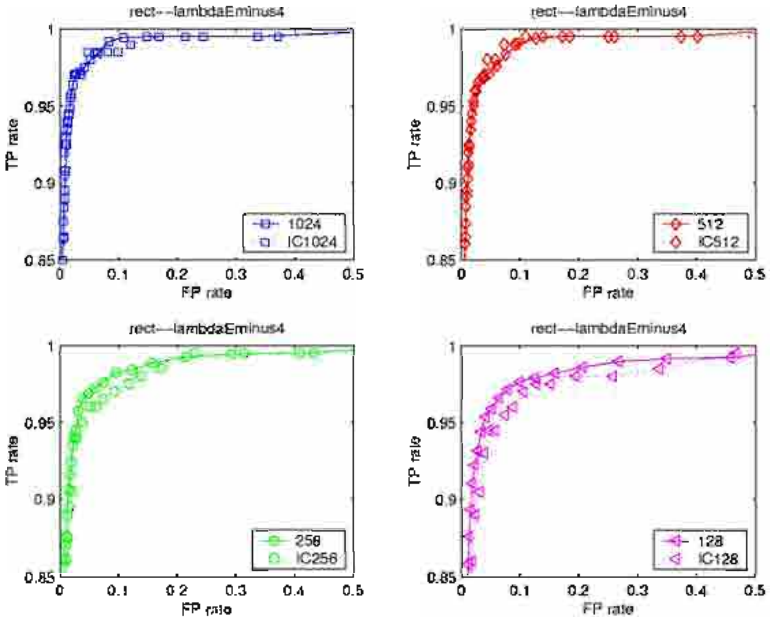


Figure 7.6: A comparison between random subset selection and the `ic` method, classifying the `usps` dataset using the `rectangle` method.

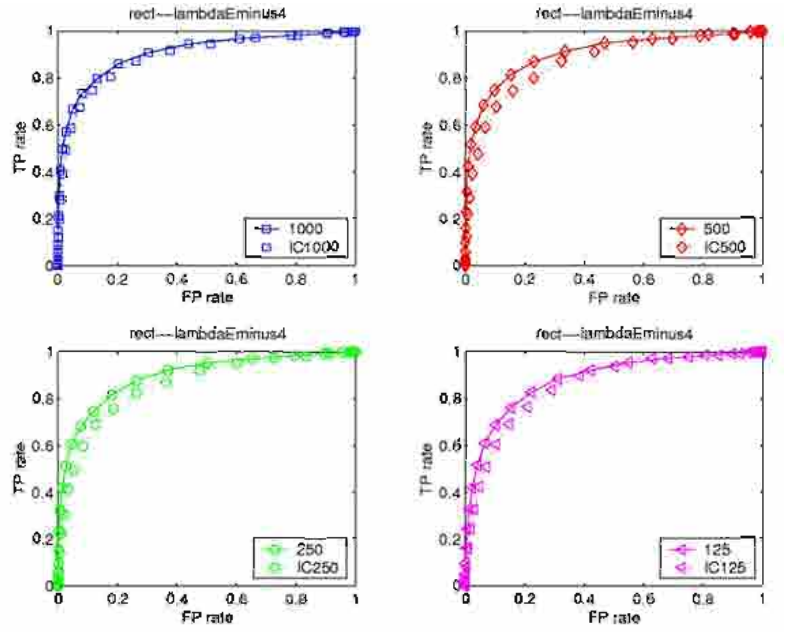


Figure 7.7: A comparison between random subset selection and the `ic` method, classifying the `faces` dataset using the `rectangle` method.

Unfortunately, using this equation requires access to the inverse matrix G, which is often computationally intractable and numerically unstable.

An alternative approach, introduced by Craven and Wahba [31], is known as the *generalized approximate cross-validation*, or GACV for short. Instead of actually using the entries of the inverse matrix G directly, an approximation to the leave-one-out value is made using the *trace* of G:

$$y_i - f_{S^i}(\mathbf{x}_i) \approx \frac{y_i - f_S(\mathbf{x}_i)}{1 - \frac{1}{\ell}\mathrm{tr}(G)}$$

This approach, while being only an approximation rather than an exact calculation, has an important advantage. The trace of a matrix is the sum of its eigenvalues. If the eigenvalues of K are known, the eigenvalues of G can be computed easily for any value of λ, and we can then easily select the value of λ which minimizes the leave-one-out bound. Unfortunately, computing all the eigenvalues of K is in general much too expensive, so again, this technique is not practical for large problems.

Jaakkola and Haussler introduced an interesting class of simple leave-one-bounds [32] for kernel classifiers. In words, their bound says that if a given training point $\mathbf{x}$ can be classified correctly by f_S *without using the contribution of* $\mathbf{x}$ *to* f_S, then $\mathbf{x}$ cannot be a leave-one-out error — $f_{S^i}(\mathbf{x}_i) \geq 0$. Although the Jaakkola and Haussler bound does not apply directly to RLSC (because the c_i are not sign-constrained), Rifkin [21] showed that their bound is valid for RLSC via a slightly more involved proof, and that the number of leave-one-out errors is bounded by

$$|\mathbf{x}_i : y_i \sum_{j \neq i} c_j K(\mathbf{x}_i, \mathbf{x}_j) \leq 0|$$

This bound can be computed directly given the c_i; no retraining is required. The same bound holds for SVMs (without an unregularized b term), via the Jaakkola and Haussler proof. However, there is a simple geometric condition in RLSC that allows us to provide a more elegant bound.

Using (7.5) and (7.6), we see that

$$\xi_i = -\ell\lambda c_i.$$

Combining this with the definition of the ξ_i,

$$\xi_i = f(\mathbf{x}_i) - y_i,$$

we find that

$$c_i = \frac{y_i - f(\mathbf{x}_i)}{\ell\lambda}.$$

Using this, we can eliminate c_i from the bound, arriving at

$$\begin{aligned} & |\mathbf{x}_i : y_i \left(f(\mathbf{x}_i) - \left(\frac{y_i - f(\mathbf{x}_i)}{\ell\lambda} \right) K(\mathbf{x}_i, \mathbf{x}_i) \right) \leq 0| \\ \Longrightarrow \quad & |\mathbf{x}_i : y_i \left((1 + \frac{K(\mathbf{x}_i, \mathbf{x}_i)}{\ell\lambda}) f(\mathbf{x}_i) - \frac{y_i}{\ell\lambda} K(\mathbf{x}_i, \mathbf{x}_i) \right) \leq 0|. \end{aligned}$$

Supposing $y_i = 1$, for some i, the condition reduces to

$$\begin{aligned} & (1 + \frac{K(\mathbf{x}_i, \mathbf{x}_i)}{\ell\lambda}) f(\mathbf{x}_i) \leq \frac{1}{\ell\lambda} K(\mathbf{x}_i, \mathbf{x}_i) \\ \Longrightarrow \quad & f(\mathbf{x}_i) \leq \frac{K(\mathbf{x}_i, \mathbf{x}_i)}{\ell\lambda + K(\mathbf{x}_i, \mathbf{x}_i)}. \end{aligned}$$

Reasoning similarly for the case where $y_i = -1$, we find that we can bound the number of leave-one-out errors for RLSC via

$$|\mathbf{x}_i : y_i f(\mathbf{x}_i) \leq \frac{K(\mathbf{x}_i, \mathbf{x}_i)}{K(\mathbf{x}_i, \mathbf{x}_i) + \ell\lambda}|.$$

In this form, there is a nice connection to the recent algorithmic stability work of Bousquet and Elisseef [8], where it is shown that Tikhonov regularization algorithms have "stability" of $O(\frac{1}{\ell\lambda})$ — very loosely, that when a single data point is removed from the training set, the change in the function obtained is $O(\frac{1}{\lambda\ell})$. For RLSC, we are able to exploit the geometry of the problem to obtain a leave-one-out bound that directly mirrors the stability theory results.

Bibliography

[1] T. Evgeniou, M. Pontil and T. Poggio, Regularization networks and support vector machines, *Advances in Computational Mathematics* **13** (2000) 1–50.

[2] F. Cucker and S. Smale, On the mathematical foundations of learning, *Bulletin of the American Mathematical Society* **39** (2002) 1–49.

[3] A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-posed problems*, W. H. Winston, Washington D.C. (1977).

[4] G. Wahba, *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*, Society for Industrial & Applied Mathematics (1990).

[5] N. Aronszajn, Theory of reproducing kernels, *Transactions of the American Mathematical Society* **68** (1950) 337–404.

[6] G. Wahba, Support vector machines, reproducing kernel hilbert spaces and the randomized gacv, Technical Report 984rr, University of Wisconsin, Department of Statistics (1998).

[7] V.N. Vapnik, *Statistical Learning Theory*, John Wiley & Sons (1998).

[8] O. Bousquet and André Elisseeff, Stability and generalization, *Journal of Machine Learning Research* **2** (2002) 499–526.

[9] F. Girosi, An equivalence between sparse approximation and support vector machines, *Neural Computation* **10**(6) (1998) 1455–1480.

[10] B. Schölkopf, R. Herbrich and Alex J. Smola, A generalized representer theorem. In *Proceedings of the 14th Annual Conference on Computational Learning Theory* (2001) 416–426.

[11] I. Schönberg, Spline functions and the problem of graduation, *Proceedings of the National Academy of Science* (1964) 947–950.

[12] M. Bertero, T. Poggio and V. Torre, Ill-posed problems in early vision, *Proceedings of the IEEE* **76** (1988) 869–889.

[13] T. Poggio and F. Girosi, A theory of networks for approximation and learning, Technical Report A.I. Memo No. 1140, C.B.C.L Paper No. 31, Massachusetts Institute of Technology, Artificial Intelligence Laboratory and Center for Biological and Computational Learning, Department of Brain and Cognitive Sciences, July (1989).

[14] T. Poggio and F. Girosi, Networks for approximation and learning, *Proceedings of the IEEE* **78**(9) (1990) 1481–1497.

[15] G. Fung and O.L. Mangasarian, Proximal support vector machine classifiers, Technical report, Data Mining Institute (2001).

[16] J.A.K. Suykens and J. Vandewalle, Least squares support vector machine classifiers, *Neural Processing Letters* **9**(3) (1999) 293–300.

[17] Y. Nesterov and A. Nemirovskii, *Interior Point Polynomial Algorithms in Convex Programming*, volume 13 of *Studies In Applied Mathematics*. SIAM (1994).

[18] T. Joachims, Making large-scale svm learning practical, Technical Report LS VIII-Report, Universität Dortmund (1998).

[19] A.J. Smola and B. Schölkopf, Sparse greedy matrix approximation for machine learning, In *Proceedings of the International Conference on Machine Learning* (2000).

[20] C.K.I. Williams and M. Seeger, Using the Nyström method to speed up kernel machines, In *Neural Information Processing Systems* (2000).

[21] R.M. Rifkin, *Everything Old Is New Again: A Fresh Look at Historical Approaches to Machine Learning*, PhD thesis, Massachusetts Institute of Technology (2002).

[22] S. Fine and K. Scheinberg, Efficient application of interior point methods for quadratic problems arising in support vector machines using low-rank kernel representation, *Submitted to Mathematical Programming* (2001).

[23] J.R. Shewchuk, An introduction to the conjugate gradient method without the agonizing pain, http://www-2.cs.cmu.edu/~jrs/jrspapers.html (1994).

[24] C.J. Merz and P.M. Murphy, Uci repository of machine learning databases, http://www.ics.uci.edu/~mlearn/MLRepository.html (1998).

[25] J. Rennie and R. Rifkin, Improving multiclass classification with the support vector machine. Technical Report A.I. Memo 2001-026, C.B.C.L. Memo 210, MIT Artificial Intelligence Laboratory, Center for Biological and Computational Learning (2001).

[26] Y.-J. Lee and O. Mangasarian, Rsvm: Reduced support vector machines, In *SIAM International Conference on Data Mining* (2001).

[27] T. Van Gestel, J. Suykens, B. De Moor, and J. Vandewalle. Bayesian inference for ls-svms on large data sets using the Nyström method. In *International Joint Conference on Neural Networks* (2002).

[28] B. Heisele, T. Poggio, and M. Pontil, Face detection in still gray images, Technical Report A.I. Memo No. 2001-010, C.B.C.L. Memo No. 197, MIT Center for Biological and Computational Learning (2000).

[29] M. Alvira and R. Rifkin, An empirical comparison of snow and svms for face detection, Technical Report A. I. Memo No. 2001-004, C.B.C.L. Memo No. 193, MIT Center for Biological and Computational Learning (2001).

[30] P.J. Green and B.W. Silverman, *Nonparametric Regression and Generalized Linear Models*, Number 58 in Monographs on Statistics and Applied Probability. Chapman & Hall (1994).

[31] P. Craven and G. Wahba, Smoothing noisy data with spline functions, *Numerical Mathematics* **31** (1966) 377–390.

[32] T. Jaakkola and D. Haussler, Probabilistic kernel regression models, In *Advances in Neural Information Processing Systems 11* (1998).

Chapter 8

Support Vector Machines: Least Squares Approaches and Extensions

Johan A.K. Suykens, Tony Van Gestel, Jos De Brabanter,
Bart De Moor and Joos Vandewalle[1]

Abstract. Support Vector Machines (SVMs) is a powerful methodology for nonlinear classification, function estimation and density estimation. However, due to its non-parametric nature, the original SVM methodology is more difficult to extend to other problem settings than e.g. classical multilayer perceptrons. In this Chapter we give a short overview on Least Squares SVMs (LS-SVMs) and demonstrate how several extensions are possible. LS-SVMs for classification and function estimation are closely related to kernel Fisher discriminant analysis, regularization networks and Gaussian processes, and aim at exploiting primal-dual formulations from the viewpoint of optimization theory. A Bayesian framework with three levels of inference is developed together with ways for achieving sparseness and robustness. The framework with primal-dual support vector machine formulations is extended to kernel-PCA, -CCA, -PLS, recurrent networks and optimal control. For very large scale problems, on-line learning and transductive inference, a method of fixed-size LS-SVM is proposed which allows modelling in the primal and the dual space in relation to a Nyström approximation with active selection of support vectors and leads to sparse representations.

[1]Research supported by *Research Council KUL*: GOA-Mefisto 666, IDO (IOTA oncology, genetic networks), several PhD/postdoc & fellow grants; *Flemish Government*: FWO: PhD/postdoc grants, G.0407.02 (support vector machines), projects G.0115.01 (microarrays/oncology), G.0240.99 (multilinear algebra), G.0080.01 (collective intelligence), G.0413.03 (inference in bioi), G.0388.03 (microarrays for clinical use), G.0229.03 (ontologies in bioi), G.0197.02 (power islands), G.0141.03 (identification and cryptography), G.0491.03 (control for intensive care glycemia), G.0120.03 (QIT), research communities (ICCoS, ANMMM); AWI: Bil. Int. Collaboration Hungary, Poland, South Africa; IWT: PhD Grants, STWW-Genprom (gene promotor prediction), GBOU-McKnow (knowledge management algorithms), GBOU-SQUAD (quorum sensing), GBOU-ANA (biosensors); Soft4s (softsensors) *Belgian Federal Government*: DWTC (IUAP IV-02 (1996-2001) and IUAP V-22 (2002-2006); PODO-II (CP/40: TMS and sustainibility); *EU*: CAGE; ERNSI; Eureka 2063-IMPACT; Eureka 2419-FliTE; *Contract Research/agreements*: Data4s, Electrabel, Elia, LMS, IPCOS, VIB; JS is a professor at K.U.Leuven Belgium and a postdoctoral researcher with FWO Flanders. TVG is postdoctoral researcher with FWO Flanders. BDM and JWDW are full professors at K.U.Leuven Belgium.

8.1 Introduction

In recent years considerable progress has been made in kernel based learning methods, after the introduction of Support Vector Machines (SVMs) [5, 30, 42, 43]. Up till now standard SVMs as proposed by Vapnik have been mainly developed for classification, nonlinear function estimation and density estimation. Interesting features of SVMs are sparse approximation by solving convex optimization problems and the primal-dual optimization problem formulations with use of Mercer's theorem. Furthermore, the use of the kernel trick (i.e. application of the Mercer theorem), has also been demonstrated for other problems such as principal component analysis and clustering [28, 29, 30]. In this way, the investigated models and methods have become important for the areas of neural networks, machine learning, pattern recognition, signal processing, datamining, systems and control with aspects relevant to mathematics, linear algebra, optimization and statistics. An interesting new aspect of SVM methods in relation to many classical methods is for example that the models are able to learn and generalize in huge dimensional input spaces which is important e.g. for novel applications in bioinformatics and textmining.

On the other hand, classical neural network methodologies such as multilayer perceptrons (MLP) [3] are currently still more flexible as a general plug-in solution thanks to the parametric nature of the model. For MLPs it is straightforward to parameterize classifiers, static nonlinear models, recurrent networks, unsupervised learning methods, controllers etc. SVM methods have often outperformed many other techniques but on the other hand it is mainly restricted to classification and static function estimation problems. For this purpose, least squares approaches are explored with the following motivations:

- Formulating SVM methods in terms of least squares cost functions and with equality constraints instead of inequality constraints usually leads to solving linear systems. Such problems are better understood from a theoretical, algorithmical and numerical point of view than solving quadratic programming problems. In view of interior point algorithms the obtained KKT systems can be considered as a core problem. Towards adaptive signal processing applications recursive least squares algorithms are well understood, e.g. in relation to Kalman filtering techniques.

- In terms of the simpler formulations it may become easier to understand the links between the new generation of SVM techniques and the related methods in different areas within the wide interdisciplinary context.

- It is shown that several extensions to the standard SVM formulations are possible in terms of least squares such as for PCA, PCR, PLS, CCA analysis problems and recurrent networks and control. In these formulations the primal-dual SVM optimization problem interpretations remain preserved. In general this leads to a new avenue of primal-dual modelling thinking where, for a given problem at hand, one may either solve the primal or the dual problem. The dual space is interesting for large dimensional inputs (e.g. classification of microarray data in bioinformatics), while the primal space can be interesting for a large number of training data (e.g. large scale datamining problems).

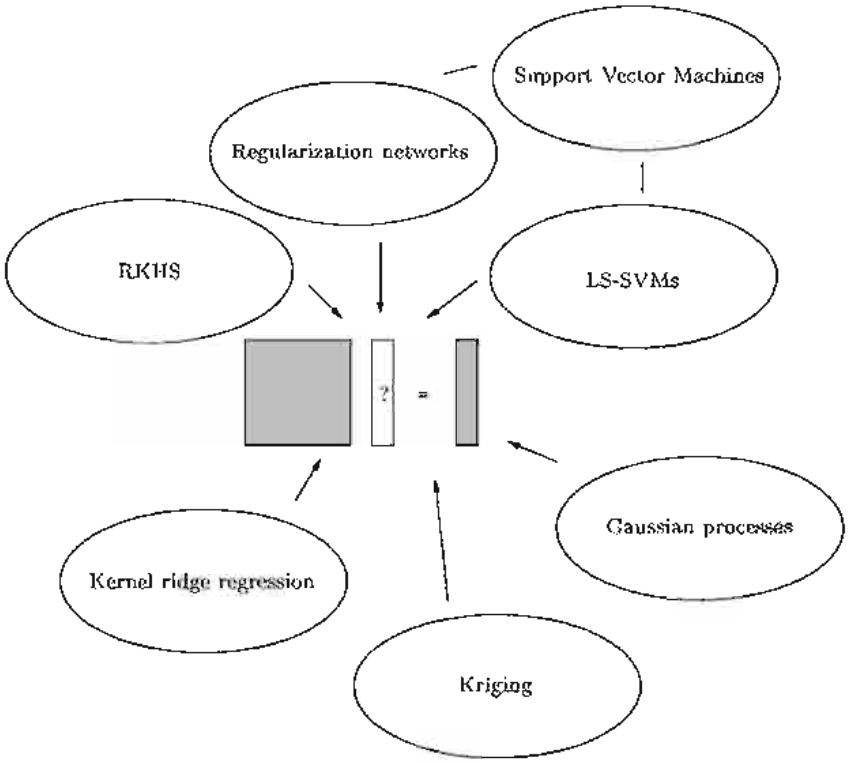


Figure 8.1: LS-SVMs are closely related to regularization networks, Gaussian processes, kriging and kernel ridge regression. The emphasis in the LS-SVM 'language' is on primal-dual optimization problem formulations as in standard SVMs.

In this chapter, we give an overview of developments in least squares support vector machines (LS-SVMs). In the function estimation case the solutions are mathematically equivalent to regularization networks and Gaussian processes (but it is more straightforward to handle bias terms in the model). The emphasis in LS-SVM models is on the primal-dual interpretations as for standard SVMs (Fig. 8.1). The classifier formulation is closely related to a kernel version of Fisher discriminant analysis. The use of least squares may have potential drawbacks such as the lack of sparseness and the robustness with respect to outliers or non-Gaussian noise distributions. Several ways are mentioned to overcome these drawbacks, such as the application of pruning techniques and the incorporation of methods from robust statistics. Probabilistic interpretations of the models are made within a Bayesian framework with three levels of inference. In the primal space, SVM models can be viewed as parametric (size of unknown vector is fixed) while in the dual space it becomes non-parametric (size of solution vector grows with the number of data). At the different levels of inference, one considers the unknown weights in the primal problem, next the hyperparameters and finally the tuning parameters of the kernel. Furthermore, support vector machine formulations are shown for PCA and CCA analysis and their kernel versions. Besides these extensions also recurrent versions of LS-SVMs have been formulated and use in optimal control. For very large scale problems and adaptive learning a method of fixed size LS-SVM has been proposed which makes use of the Nyström method with active selection of support vectors. Further details can be found in [33].

8.2 Least Squares SVMs for Classification and Function Estimation

8.2.1 LS-SVM classifiers and link with kernel FDA

Consider a given training set $\{x_k, y_k\}_{k=1}^N$ with input data $x_k \in \mathbb{R}^n$ and output data with class labels $y_k \in \{-1, +1\}$. The standard nonlinear SVM classifier [42] which takes the form

$$y(x) = \text{sign}\left[w^T\varphi(x) + b\right] \tag{8.1}$$

in the primal space with $\varphi(\cdot)$: $\mathbb{R}^n \to \mathbb{R}^{n_h}$, where n_h is the dimension of the (potentially infinite dimensional) feature space, and is determined by the optimization problem

$$\left[\begin{array}{lll} \boxed{\text{P}}: & \min\limits_{w,b,\xi} J_{\text{P}}(w,\xi) = & \frac{1}{2}w^Tw + c\sum\limits_{k=1}^N \xi_k \\ & \text{such that} & y_k\left[w^T\varphi(x_k) + b\right] \geq 1 - \xi_k, \quad k = 1,...,N \\ & & \xi_k \geq 0, \quad k = 1,...,N \end{array}\right] \tag{8.2}$$

where ξ_k are slack variables needed for tolerating misclassifications. In the objective function one penalizes the maximization of the soft-margin (by minimization of the regularization term) with respect to the sum of these slack variables, subject to the set of inequalities for correct classification of the training data, except for an amount of misclassified points determined by tuning parameter c. One solves then the dual problem in the unknown Lagrange multipliers associated to this problem.

For least squares support vector machine classifiers [34] this is reformulated into

$$\left[\begin{array}{lll} \boxed{\text{P}}: & \min\limits_{w,b,e} J_{\text{P}}(w,e) = & \frac{1}{2}w^Tw + \gamma\frac{1}{2}\sum\limits_{k=1}^N e_k^2 \\ & \text{such that} & y_k\left[w^T\varphi(x_k) + b\right] = 1 - e_k, \; k = 1,...,N \end{array}\right] \tag{8.3}$$

where the set of inequalities are replaced by equalities. Instead of considering the value 1 as a threshold value, it is taken as a target value upon which an error variable is considered which is penalized in a least squares sense within the cost function with regularization constant γ.

The Lagrangian for the problem is

$$\mathcal{L}(w,b,e;\alpha) = J_{\text{P}}(w,e) - \sum_{k=1}^N \alpha_k\{y_k\left[w^T\varphi(x_k) + b\right] - 1 + e_k\} \tag{8.4}$$

where the α_k values are the Lagrange multipliers, which can be positive or negative due to the equality constraints.

The conditions for optimality yield

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial w} = 0 \;\rightarrow\; w = \sum_{k=1}^{N} \alpha_k y_k \varphi(x_k) \\ \frac{\partial \mathcal{L}}{\partial b} = 0 \;\rightarrow\; \sum_{k=1}^{N} \alpha_k y_k = 0 \\ \frac{\partial \mathcal{L}}{\partial e_k} = 0 \;\rightarrow\; \alpha_k = \gamma e_k, \qquad k = 1, ..., N \\ \frac{\partial \mathcal{L}}{\partial \alpha_k} = 0 \;\rightarrow\; y_k \left[w^T \varphi(x_k) + b \right] - 1 + e_k = 0, \quad k = 1, ..., N. \end{cases} \tag{8.5}$$

Defining $y = [y_1; ...; y_N]$, $1_v = [1; ...; 1]$, $e = [e_1; ...; e_N]$, $\alpha = [\alpha_1; ...; \alpha_N]$ and eliminating w, e, one obtains the following linear Karush-Kuhn-Tucker (KKT) system

$$\left[\begin{array}{l} \boxed{\mathrm{D}}: \;\text{solve in } \alpha, b: \\ \\ \left[\begin{array}{c|c} 0 & y^T \\ \hline y & \Omega + I/\gamma \end{array} \right] \left[\begin{array}{c} b \\ \hline \alpha \end{array} \right] = \left[\begin{array}{c} 0 \\ \hline 1_v \end{array} \right] \end{array} \right] \tag{8.6}$$

where $\Omega_{kl} = y_k y_l K(x_k, x_l)$ for $k, l = 1, ..., N$ and $K(x_k, x_l) = \varphi(x_k)^T \varphi(x_l)$ is a positive definite kernel, according to the Mercer theorem. In the dual space the classifier takes the form

$$y(x) = \mathrm{sign} \left[\sum_{k=1}^{N} \alpha_k y_k K(x, x_k) + b \right] \tag{8.7}$$

where α, b follows from solving the linear system. Any positive definite kernel can be taken such as e.g. the linear kernel $K(x_k, x_l) = x_k^T x_l$ and RBF kernel $K(x_k, x_l) = \exp(-\|x_k - x_l\|_2^2/\sigma^2)$.

For a larger number of data points, iterative methods such as Conjugate Gradient (CG) algorithms [11] can be applied for solving the linear system. Due to the fact that a bias term is considered in the model, the linear system matrix is indefinite. However, it is straightforward to transform it into a positive definite system such that CG algorithms can be applied to it [33, 39]. Also SMO optimization has been developed for LS-SVM classifiers [47]. LS-SVM classifiers have been successfully tested on 20 UCI (binary and multiclass) benchmark data sets. The method consistently performs very well in comparison with many other methods on all these data sets and its performance is comparable to the standard SVM performance.

SVM models have neural network interpretations both in the primal and the dual space. In the primal space the problem is essentially parametric, while in the dual space it becomes non-parametric because the size of the solution vector α grows with the number of data points (Fig. 8.2). In this LS-SVM formulation, every data point is a support vector but some points contribute more than others as follows from the conditions for optimality with $\alpha_k = \gamma e_k$. For standard SVMs the support vectors are typically located close to the decision boundary, while in the LS-SVM case the points with large $|\alpha_k|$ values are located close and far from the decision boundary.

The LS-SVM classifier formulation can be related to a kernel version of Fisher Discriminant Analysis (FDA) [2, 23, 40]. In linear FDA the data are projected as

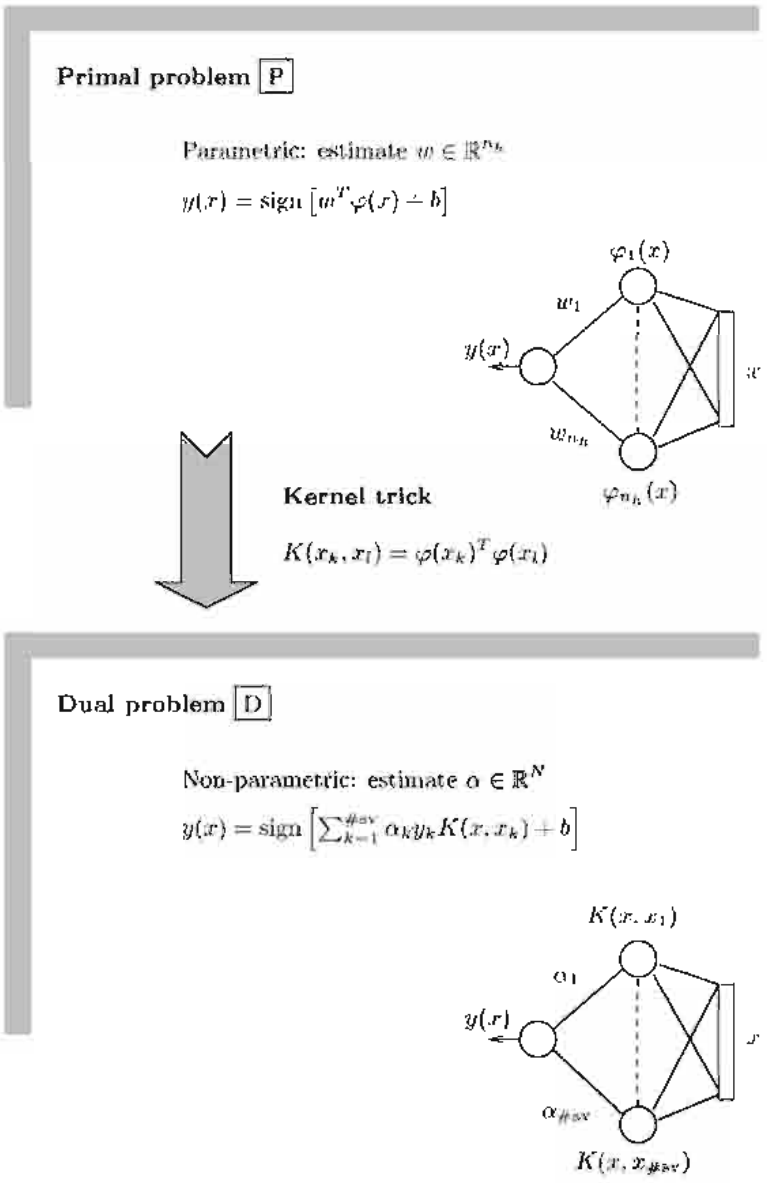


Figure 8.2: Primal-dual neural network interpretations of support vector machines.

follows

$$z = w^T x + b \tag{8.8}$$

where w is determined by optimizing the Rayleigh quotient

$$\max_w J_{\text{FD}}(w) = \frac{w^T \Sigma_B w}{w^T \Sigma_W w} \tag{8.9}$$

with Σ_B, Σ_W the between class and within class covariance matrix, respectively. In the dual space, the kernel FDA version with Fisher target ± 1 is closely related to the LS-SVM classifier solution.

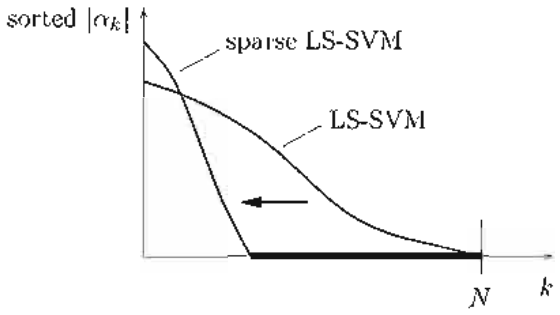


Figure 8.3: Pruning techniques as a simple method for imposing sparseness in LS-SVM models.

8.2.2 Function estimation case and equivalence to a regularization network solution

The LS-SVM function estimation formulation is given by

$$\left[\begin{array}{l} \boxed{\mathrm{P}}: \ \min_{w,b,e} J_{\mathrm{P}}(w,e) = \frac{1}{2}w^Tw + \gamma\frac{1}{2}\sum_{k=1}^{N} e_k^2 \\ \qquad \text{such that} \qquad y_k = w^T\varphi(x_k) + b + e_k, \ \ k = 1,...,N \end{array}\right] \tag{8.10}$$

which corresponds to a form of ridge regression [27]. Taking conditions for optimality with Lagrangian $\mathcal{L}(w,b,e;\alpha) = \mathcal{J}_{\mathrm{P}}(w,e) - \sum_{k=1}^{N}\alpha_k\{w^T\varphi(x_k) + b + e_k - y_k\}$ gives the dual problem after taking the conditions for optimality $\partial\mathcal{L}/\partial w = 0$, $\partial\mathcal{L}/\partial b = 0$, $\partial\mathcal{L}/\partial e_k = 0$, $(k = 1,..,N)$, $\partial\mathcal{L}/\partial\alpha_k = 0$, $(k = 1,..,N)$,

$$\left[\begin{array}{l} \boxed{\mathrm{D}}: \ \text{solve in } \alpha, b: \\ \qquad \left[\begin{array}{c|c} 0 & 1_v^T \\ \hline 1_v & \Omega + I/\gamma \end{array}\right]\left[\begin{array}{c} b \\ \hline \alpha \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline y \end{array}\right] \end{array}\right] \tag{8.11}$$

with $\Omega_{kl} = K(x_k, x_l)$ for $k,l = 1,...,N$. The solution is mathematically equivalent to regularization networks [8, 7, 24] although in the LS-SVM formulation it is more straightforward to take into account a bias term. Regularization networks are formulated in the context of functional analysis, while (LS)-SVMs are derived in view of primal-dual optimization problem formulations and invoke the Mercer theorem for linking the model between the primal and the dual space. LS-SVM and regularization network solutions also correspond to kernel ridge regression, Gaussian processes for function estimation and kriging [4, 19, 20, 44, 45].

8.2.3 Issues of sparseness and robustness

Potential drawbacks of the use of least squares cost functions are the lack of sparseness and the lack of robustness with respect to outliers and non-Gaussian noise distributions.

On the other hand there are several possible ways to overcome these drawbacks. A simple way to sparsify the obtained LS-SVM models is to apply neural network

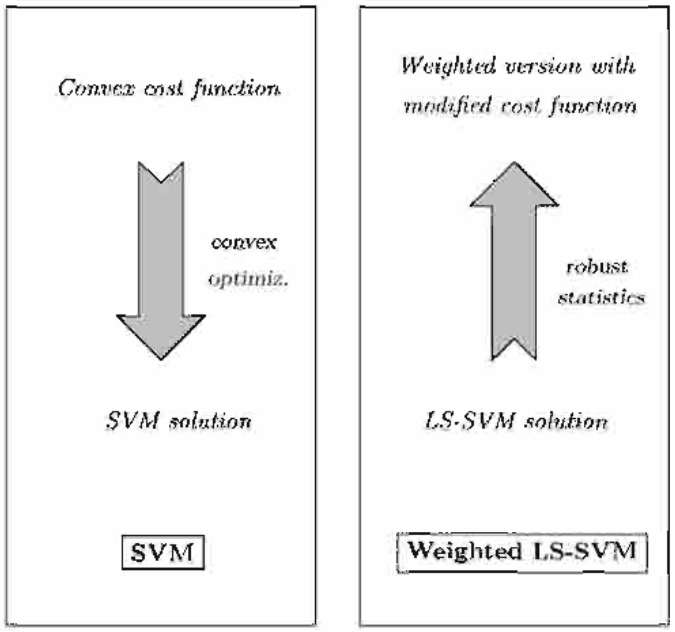


Figure 8.4: In standard SVMs one takes a top-down approach by choosing a convex cost function and finds the SVM solution by convex optimization. On the other hand one can also follow a bottom-up approach by starting from LS-SVM and define a weighting in terms of the error distribution based on robust statistics which implicitly corresponds to a modified cost function. In this way one aims at implicitly finding a most suitable cost function in view of robust statistics, with a good robustness-efficiency trade-off.

techniques of pruning (such as optimal brain damage and optimal brain surgeon) [13, 3]. Less relevant weights are gradually removed and the model is retrained (Fig. 8.3). A simple version with pruning of the smallest $|\alpha_k|$ values has been successfully applied to many classification and function estimation problem (although the obtained degree of sparsity is not extremely high due to the simplicity of this procedure). In this way one reduces the size of the model [35]. Another approach with a fixed size method will be explained in the sequel of this chapter. Furthermore, it is also known that there is a close connection between SVMs and sparse approximation in general [10].

In the standard SVM function estimation case the Vapnik epsilon insensitive loss function is usually employed. Essentially, this is an L_1 estimator (which ensures robustness) but with an epsilon insensitive zone around the origin for achieving sparseness. The SVM methodology has been extended to any convex cost function (Fig. 8.4). In order to robustify LS-SVM models it is straightforward to apply an additional weighted version. The weightings are determined as a function of the empirical distribution of the error variables e_k with a robust scale estimation based on robust statistics [16]. In standard SVMs a *convex* cost function is chosen which gives the solution in a top-down fashion by applying a convex optimization algorithm, e.g. by using interior point algorithms. The reduced KKT system to be solved within interior point algorithms takes the same form as one single LS-SVM KKT system. Hence, conceptually one may view it as solving a sequence of LS-SVM KKT systems. In the weighted LS-SVM case

one rather works in a bottom-up way, by first solving the unweighted LS-SVM and then solve a weighted version, where one additional weighting step is often sufficient [35]. Implicitly this weighted version corresponds to solving a modified cost function. At this point one should note that several popular loss functions in robust statistics (Andrews, Hampel, Tukey) are non-convex and cannot be plugged-in into standard SVM formulations, except for the Huber loss function [12].

8.2.4 Bayesian inference of LS-SVMs and Gaussian processes

A Bayesian framework with three levels of inference has been developed for LS-SVM classifiers and regression (Fig. 8.5) [40]. At this point there are a few differences with related work in the area of Gaussian Processes (GP) [45]. In the LS-SVM models it is straightforward to handle a bias term, while in GP this should be done by taking an additional constant within the kernel. The LS-SVM classifier case can be treated as a regression problem with targets ± 1 while the classifier case in GP needs the application of sampling techniques. Furthermore, tuning parameters of the kernel are treated at a different level of inference. The Bayesian framework for LS-SVM models is done in the same way as the framework developed by MacKay for MLPs [22].

At the first level, inference is considered in the primal weight parameters w, b. It leads to a probabilistic interpretation of the output and an additional correction for the choice of the bias term. At the second level, inference of the hyperparameters is done. In addition to the regularization constant γ, which is associated to the least squares cost function, an additional hyperparameter is needed which is associated which is related to the regularization term (hyperparameters μ, ζ). This leads to expressions for the effective number of parameters in terms the eigenvalues of the centered kernel matrix. At the third level of inference different models $\mathcal{H}_\sigma$ (assume e.g. models parameterized by a kernel tuning parameter σ) can be compared, where the kernel width σ is selected by maximizing the level 3 posterior. By taking as many σ_i values as components of the input vector (equivalent to taking a weighted norm [24]), these σ_i values can be inferred at level 3. In this way one can assess the relevance of the input variables of the model. This procedure is called Automatic Relevance Determination (ARD).

8.3 Primal-dual Formulations to Kernel PCA and CCA

8.3.1 Kernel PCA as a one-class modelling problem and a primal-dual derivation

In view of the least squares formulations for classification and nonlinear function estimation, extensions of the framework can be made towards kernel Principal Component Analysis (PCA) [28], kernel canonical correlation analysis (CCA) [1] and kernel partial least squares (PLS) [26].

In order to obtain a support vector machine formulation of kernel PCA with primal-dual interpretation [38], the PCA analysis problem is viewed as a one-class modelling problem with target zero (instead of targets ± 1 as for the classification case) upon which

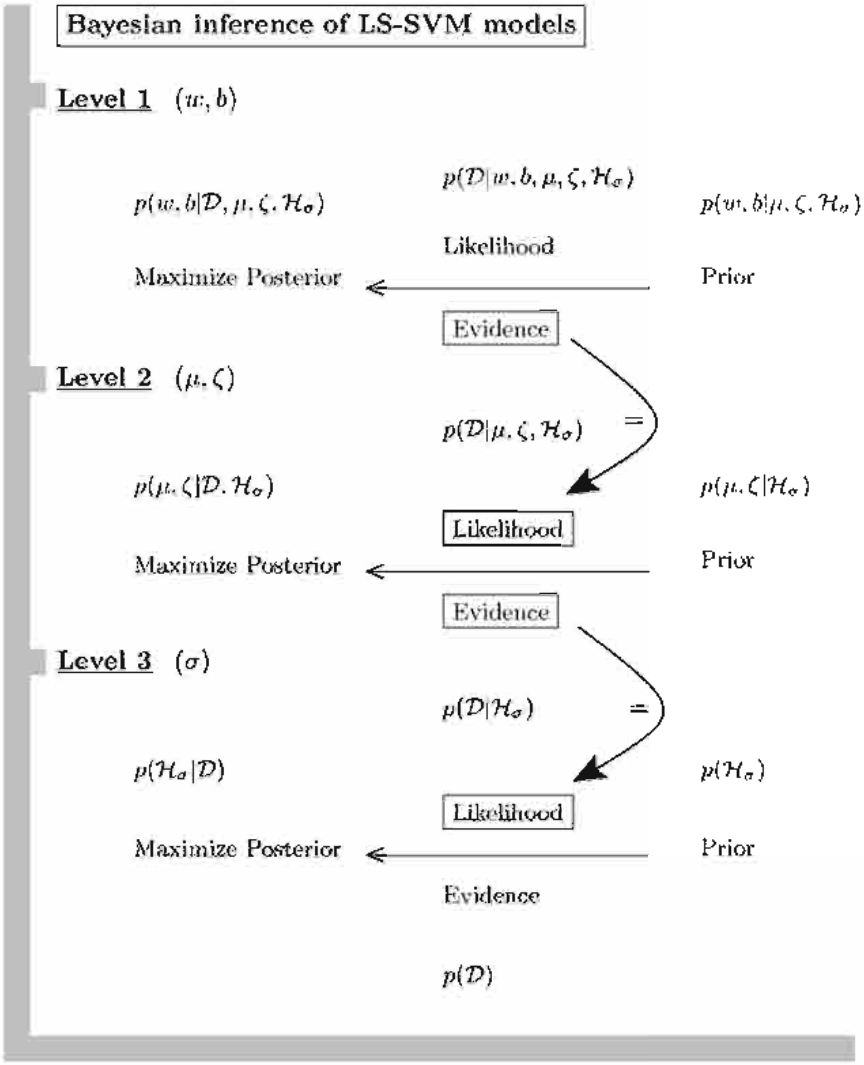


Figure 8.5: Illustration of Bayesian inference for LS-SVM models with several levels of inference.

an error variable is considered and the sum squared error is maximized (in comparison with minimization for the classification case). These error variables correspond to the score variables (Fig. 8.6).

The objective is then the following

$$\max_{w} \sum_{k=1}^{N} \left(0 - w^T(\varphi(x_k) - \widehat{\mu}_\varphi)\right)^2 \tag{8.12}$$

with $\widehat{\mu}_\varphi = (1/N)\sum_{k=1}^{N} \varphi(x_k)$ and $\varphi(\cdot)$: $\mathbb{R}^n \to \mathbb{R}^{n_h}$ the mapping to a high dimensional feature space which might be infinite dimensional. Instead of this centering approach, also a bias term can be taken. The following optimization problem is formulated in

the primal weight space

$$\left[\begin{array}{ll} \boxed{\mathrm{P}}: & \max_{w,e} J_{\mathrm{P}}(w,e) = \gamma\frac{1}{2}\sum_{k=1}^{N} e_k^2 - \frac{1}{2}w^T w \\ & \text{such that} \qquad e_k = w^T\left(\varphi(x_k) - \widehat{\mu}_\varphi\right),\ k=1,...,N. \end{array}\right] \tag{8.13}$$

This gives the Lagrangian

$$\mathcal{L}(w,e;\alpha) = \gamma\frac{1}{2}\sum_{k=1}^{N} e_k^2 - \frac{1}{2}w^T w - \sum_{k=1}^{N}\alpha_k\left(e_k - w^T(\varphi(x_k) - \widehat{\mu}_\varphi)\right) \tag{8.14}$$

with conditions for optimality

$$\left\{\begin{array}{lll} \frac{\partial\mathcal{L}}{\partial w} = 0 & \rightarrow & w = \sum_{k=1}^{N}\alpha_k(\varphi(x_k) - \widehat{\mu}_\varphi) \\ \frac{\partial\mathcal{L}}{\partial e_k} = 0 & \rightarrow & \alpha_k = \gamma e_k, \qquad\qquad k=1,...,N \\ \frac{\partial\mathcal{L}}{\partial\alpha_k} = 0 & \rightarrow & e_k - w^T(\varphi(x_k) - \widehat{\mu}_\varphi) = 0, \quad k=1,...,N. \end{array}\right. \tag{8.15}$$

By elimination of the variables e, w one obtains

$$\frac{1}{\gamma}\alpha_k - \sum_{l=1}^{N}\alpha_l(\varphi(x_l) - \widehat{\mu}_\varphi)^T(\varphi(x_k) - \widehat{\mu}_\varphi) = 0\ ,\ k=1,...,N. \tag{8.16}$$

Defining $\lambda = 1/\gamma$ one gets the following dual problem

$$\left[\begin{array}{ll} \boxed{\mathrm{D}}: & \text{solve in } \alpha\text{:} \\ & \Omega_c\alpha = \lambda\alpha \end{array}\right] \tag{8.17}$$

with

$$\Omega_c = \left[\begin{array}{ccc} (\varphi(x_1) - \widehat{\mu}_\varphi)^T(\varphi(x_1) - \widehat{\mu}_\varphi) & \ldots & (\varphi(x_1) - \widehat{\mu}_\varphi)^T(\varphi(x_N) - \widehat{\mu}_\varphi) \\ \vdots & & \vdots \\ (\varphi(x_N) - \widehat{\mu}_\varphi)^T(\varphi(x_1) - \widehat{\mu}_\varphi) & \ldots & (\varphi(x_N) - \widehat{\mu}_\varphi)^T(\varphi(x_N) - \widehat{\mu}_\varphi) \end{array}\right] \tag{8.18}$$

with as elements for the centered kernel matrix

$$\Omega_{c,kl} = (\varphi(x_k) - \widehat{\mu}_\varphi)^T(\varphi(x_l) - \widehat{\mu}_\varphi), \qquad k,l=1,...,N. \tag{8.19}$$

For the centered kernel matrix one can apply the kernel trick for given points x_k, x_l:

$$\begin{aligned} &(\varphi(x_k) - \widehat{\mu}_\varphi)^T\ (\varphi(x_l) - \widehat{\mu}_\varphi) \\ &= K(x_k,x_l) - \frac{1}{N}\sum_{r=1}^{N}K(x_k,x_r) - \frac{1}{N}\sum_{r=1}^{N}K(x_l,x_r) + \frac{1}{N^2}\sum_{r=1}^{N}\sum_{s=1}^{N}K(x_r,x_s). \end{aligned} \tag{8.20}$$

For the linear PCA case this new formulation shows the link with principal co-ordinate analysis [17] as the dual problem. In this sense, kernel PCA as proposed by Schölkopf *et al.* corresponds to the kernel version of principal co-ordinate analysis [17].

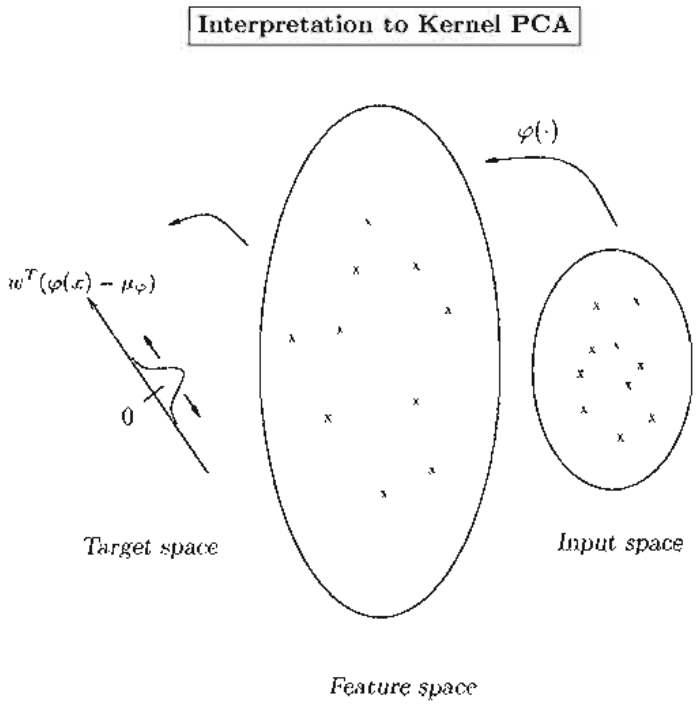


Figure 8.6: Interpretation of kernel PCA towards a primal-dual support vector machine formulation.

8.3.2 A support vector machine formulation to Kernel CCA

The problem of canonical correlation analysis (CCA) is closely related to the PCA analysis problem [17]. In CCA analysis (originally studied by Hotelling [15]) one is interested in finding maximal correlation between projected variables $z_x = w^T x$ and $z_y = v^T y$ where $x \in \mathbb{R}^{n_x}, y \in \mathbb{R}^{n_y}$ denote given random vectors with zero mean. Linear CCA analysis has been applied e.g. in subspace algorithms for system identification, with links to system theory, information theory and signal processing.

The objective function is to maximize the correlation coefficient

$$\begin{aligned} \max_{w,v} \rho &= \frac{\mathcal{E}\left[z_x z_y\right]}{\sqrt{\mathcal{E}\left[z_x z_x\right]}\sqrt{\mathcal{E}\left[z_y z_y\right]}} \\ &= \frac{w^T C_{xy} v}{\sqrt{w^T C_{xx} w}\sqrt{v^T C_{yy} v}} \end{aligned} \tag{8.21}$$

with $C_{xx} = \mathcal{E}\left[xx^T\right]$, $C_{yy} = \mathcal{E}\left[yy^T\right]$, $C_{xy} = \mathcal{E}\left[xy^T\right]$.

Towards a kernel CCA version [1] and its primal-dual support vector machine formulation, one considers the score variables [33]

$$\begin{aligned} z_x &= w^T\left(\varphi_1(x) - \widehat{\mu}_{\varphi_1}\right) \\ z_y &= v^T\left(\varphi_2(y) - \widehat{\mu}_{\varphi_2}\right) \end{aligned} \tag{8.22}$$

where $\varphi_1(\cdot)$: $\mathbb{R}^{n_x} \to \mathbb{R}^{n_{h_x}}$ and $\varphi_2(\cdot)$: $\mathbb{R}^{n_y} \to \mathbb{R}^{n_{h_y}}$ are mappings (which can be chosen to be different) to high dimensional feature spaces and $\widehat{\mu}_{\varphi_1} = (1/N)\sum_{k=1}^{N}\varphi_1(x_k)$, $\widehat{\mu}_{\varphi_2} = (1/N)\sum_{k=1}^{N}\varphi_2(y_k)$. It is important to take centering into account for the nonlinear CCA problem.

One starts from the primal problem

$$\left[\begin{array}{lll} \boxed{\mathrm{P}}: & \max\limits_{w,v,e,r} & \gamma\sum\limits_{k=1}^{N} e_k r_k - \nu_1\frac{1}{2}\sum\limits_{k=1}^{N} e_k^2 - \nu_2\frac{1}{2}\sum\limits_{k=1}^{N} r_k^2 - \frac{1}{2}w^T w - \frac{1}{2}v^T v \\ & \text{such that} & e_k = w^T\left(\varphi_1(x_k) - \widehat{\mu}_{\varphi_1}\right), \qquad k = 1,...,N \\ & & r_k = v^T\left(\varphi_2(y_k) - \widehat{\mu}_{\varphi_2}\right), \qquad k = 1,...,N \end{array}\right] \tag{8.23}$$

with Lagrangian

$$\begin{aligned} \mathcal{L}(w,v,e,r;\alpha,\beta) = \gamma\sum_{k=1}^{N} e_k r_k - \nu_1\frac{1}{2}\sum_{k=1}^{N} e_k^2 - \nu_2\frac{1}{2}\sum_{k=1}^{N} r_k^2 - \frac{1}{2}w^T w - \\ \frac{1}{2}v^T v - \sum_{k=1}^{N}\alpha_k\left[e_k - w^T\left(\varphi_1(x_k) - \widehat{\mu}_{\varphi_1}\right)\right] - \sum_{k=1}^{N}\beta_k\left[r_k - v^T\left(\varphi_2(y_k) - \widehat{\mu}_{\varphi_2}\right)\right] \end{aligned} \tag{8.24}$$

where α_k, β_k are Lagrange multipliers. Note that w and v might be infinite dimensional.

The conditions for optimality are

$$\left\{\begin{array}{lll} \frac{\partial\mathcal{L}}{\partial w} = 0 & \to & w = \sum\limits_{k=1}^{N}\alpha_k\left(\varphi_1(x_k) - \widehat{\mu}_{\varphi_1}\right) \\ \frac{\partial\mathcal{L}}{\partial v} = 0 & \to & v = \sum\limits_{k=1}^{N}\beta_k\left(\varphi_2(y_k) - \widehat{\mu}_{\varphi_2}\right) \\ \frac{\partial\mathcal{L}}{\partial e_k} = 0 & \to & \gamma v^T\left(\varphi_2(y_k) - \widehat{\mu}_{\varphi_2}\right) = \nu_1 w^T\left(\varphi_1(x_k) - \widehat{\mu}_{\varphi_1}\right) + \alpha_k \\ & & \qquad k = 1,...,N \\ \frac{\partial\mathcal{L}}{\partial r_k} = 0 & \to & \gamma w^T\left(\varphi_1(x_k) - \widehat{\mu}_{\varphi_1}\right) = \nu_2 v^T\left(\varphi_2(y_k) - \widehat{\mu}_{\varphi_2}\right) + \beta_k \\ & & \qquad k = 1,...,N \\ \frac{\partial\mathcal{L}}{\partial\alpha_k} = 0 & \to & e_k = w^T\left(\varphi_1(x_k) - \widehat{\mu}_{\varphi_1}\right) \\ & & \qquad k = 1,...,N \\ \frac{\partial\mathcal{L}}{\partial\beta_k} = 0 & \to & r_k = v^T\left(\varphi_2(y_k) - \widehat{\mu}_{\varphi_2}\right) \\ & & \qquad k = 1,...,N \end{array}\right. \tag{8.25}$$

which results in the following dual problem after defining $\lambda = 1/\gamma$

$$\left[\begin{array}{l} \boxed{\mathrm{D}}: \text{ solve in } \alpha,\beta: \\ \\ \begin{bmatrix} 0 & \Omega_{c,2} \\ \Omega_{c,1} & 0 \end{bmatrix}\begin{bmatrix} \alpha \\ \beta \end{bmatrix} \\ \\ = \lambda\begin{bmatrix} \nu_1\Omega_{c,1} + I & 0 \\ 0 & \nu_2\Omega_{c,2} + I \end{bmatrix}\begin{bmatrix} \alpha \\ \beta \end{bmatrix} \end{array}\right] \tag{8.26}$$

where

$$\begin{aligned} \Omega_{c,1_{kl}} &= (\varphi_1(x_k) - \widehat{\mu}_{\varphi_1})^T (\varphi_1(x_l) - \widehat{\mu}_{\varphi_1}) \\ \Omega_{c,2_{kl}} &= (\varphi_2(y_k) - \widehat{\mu}_{\varphi_2})^T (\varphi_2(y_l) - \widehat{\mu}_{\varphi_2}) \end{aligned} \tag{8.27}$$

are the elements of the centered Gram matrices for $k, l = 1, ..., N$. In practice these matrices can be computed by $\Omega_{c,1} = M_c \Omega_1 M_c$, $\Omega_{c,2} = M_c \Omega_2 M_c$ with centering matrix $M_c = I - 1_v 1_v^T / N$. The eigenvalues and eigenvectors that give an optimal correlation coefficient value are selected. The resulting score variables can be computed by applying the kernel trick with kernels $K_1(x_k, x_l) = \varphi_1(x_k)^T \varphi_1(x_l)$, $K_2(y_k, y_l) = \varphi_2(y_k)^T \varphi_2(y_l)$. This kernel CCA formulation can also be further related to kernel partial least squares (PLS), which has been studied in [14, 26].

8.4 Large Scale Methods and On-line Learning

8.4.1 Nyström method

For linear (LS)-SVMs one may in fact solve the primal problem instead of the dual. The primal is advantageous for solving problems with a large number of given training data N, while the dual is suitable for large dimensional input spaces. For the nonlinear case on the other hand the situation is more complicated. For many choices of the kernel, $\varphi(\cdot)$ may become infinite dimensional and hence also the w vector. However, one may still try to find meaningful estimates for $\varphi(x_k)$.

A technique to find such estimates is the Nyström method, which is well known in the area of integral equations and has been successfully applied in the context of Gaussian processes by Williams & Seeger in [46]. The method is related to finding a low rank approximation to the given kernel matrix by randomly choosing M rows/columns of the kernel matrix. Let us denote the *big* kernel matrix by $\Omega_{(N,N)} \in \mathbb{R}^{N \times N}$ and the *small* kernel matrix based on the random subsample $\Omega_{(M,M)} \in \mathbb{R}^{M \times M}$ with $M < N$ (in practice often $M \ll N$). Consider the eigenvalue decomposition of the *small* kernel matrix $\Omega_{(M,M)}$

$$\Omega_{(M,M)} \overline{U} = \overline{U}\, \overline{\Lambda} \tag{8.28}$$

where $\overline{\Lambda} = \mathrm{diag}([\overline{\lambda}_1; ...; \overline{\lambda}_M])$ contains the eigenvalues and $\overline{U} = [\overline{u}_1 ... \overline{u}_M] \in \mathbb{R}^{M \times M}$ the corresponding eigenvectors. This is related to eigenfunctions $\phi_i(x)$ and eigenvalues λ_i of the integral equation

$$\int K(x, x') \phi_i(x) p(x) dx = \lambda_i \phi_i(x') \tag{8.29}$$

as follows

$$\begin{aligned} \widehat{\lambda}_i &= \tfrac{1}{M} \overline{\lambda}_i \\ \widehat{\phi}_i(x_k) &= \sqrt{M}\, \overline{u}_{ki} \\ \widehat{\phi}_i(x') &= \tfrac{\sqrt{M}}{\overline{\lambda}_i} \sum_{k=1}^{M} \overline{u}_{ki} K(x_k, x') \end{aligned} \tag{8.30}$$

where $\widehat{\lambda}_i$ and $\widehat{\phi}_i$ are estimates to λ_i and ϕ_i, respectively, for the integral equation, and $\overline{u}_{ki}$ denotes the ki-th entry of the matrix $\overline{U}$. This can be understood from sampling the

integral by M points $x_1, x_2, ..., x_M$. For the *big* kernel matrix one has the eigenvalue decomposition

$$\Omega_{(N,N)}\, \widetilde{U} = \widetilde{U}\, \widetilde{\Lambda}. \tag{8.31}$$

Furthermore, as explained in [46] one has

$$\begin{aligned} \widetilde{\lambda}_i &= \tfrac{N}{M}\overline{\lambda}_i \\ \widetilde{u}_i &= \sqrt{\tfrac{N}{M}}\tfrac{1}{\overline{\lambda}_i}\Omega_{(N,M)}\overline{u}_i. \end{aligned} \tag{8.32}$$

One can then show that

$$\Omega_{(N,N)} \simeq \Omega_{(N,M)}\Omega^{-1}_{(M,M)}\Omega_{(M,N)} \tag{8.33}$$

where $\Omega_{(N,M)}$ is the $N \times M$ block matrix taken from $\Omega_{(N,N)}$. These insights are used then for approximately solving the linear system in combination with applying the matrix inversion lemma.

8.4.2 Basis construction in the feature space using fixed size LS-SVM

In the Nyström method an approximate solution to the linear system is computed based upon a random subsample of the given training data set. In the method of fixed size LS-SVM [33] the number of support vectors is decided beforehand and the support vectors are actively selected instead of taken at random. The method makes use of the Nyström method but additionally links it to entropy criteria and density estimation. The support vectors are selected according to an entropy criterion and the estimation is done in the primal space by exploiting the primal-dual LS-SVM formulations (Fig. 8.7). In this way one selects a basis in the feature space. Other methods related to basis construction in the feature space are e.g. [6, 29, 32].

The estimation in the primal weight space is done in the unknowns w, b

$$\min_{w\in\mathbb{R}^{n_h}, b\in\mathbb{R}} \frac{1}{2}w^T w + \gamma\, \frac{1}{2}\sum_{k=1}^{N}\left(y_k - (w^T\varphi(x_k) + b)\right)^2 \tag{8.34}$$

with estimations for $\varphi(x_k)$ provided by the Nyström method. One chooses a fixed size M ($M \leq N$ and typically $M \ll N$) for a working set of support vectors where the value of M is related to the Nyström subsample. Using the expression (8.30) one obtains

$$\varphi_i(x') = \sqrt{\overline{\lambda}_i}\,\widehat{\phi}_i(x') = \frac{\sqrt{M}}{\sqrt{\overline{\lambda}_i}}\sum_{k=1}^{M}\overline{u}_{ki}K(x_k, x'). \tag{8.35}$$

Hence one constructs the $M \times M$ kernel matrix, takes its eigenvalue decomposition, and computes the eigenfunctions based upon the eigenvalue decomposition of the kernel matrix which gives the expression for $\varphi(x')$ evaluated at any point x'. This can be applied both to function estimation and classification problems.

The model takes the form

$$\begin{aligned} y(x) &= w^T\varphi(x) + b \\ &= \sum_{i=1}^{M} w_i \frac{\sqrt{M}}{\sqrt{\lambda_i}} \sum_{k=1}^{M} \overline{u}_{ki} K(x_k, x). \end{aligned} \tag{8.36}$$

The support values corresponding to the number of M support vectors are then

$$\alpha_k = \sum_{i=1}^{M} w_i \frac{\sqrt{M}}{\sqrt{\lambda_i}} \overline{u}_{ki} \tag{8.37}$$

if one represents the model as

$$y(x) = \sum_{k=1}^{M} \alpha_k K(x_k, x). \tag{8.38}$$

This approach gives explicit links between the primal and the dual representation.

In order to make a more suitable selection of the support vectors instead of a random selection, one can relate the Nyström method to kernel principal component analysis, density estimation and entropy criteria, as discussed in [9] (Fig. 8.8). An analysis is done of the quadratic Renyi entropy [25]

$$H_R = -\log \int p(x)^2 dx \tag{8.39}$$

in relation to kernel PCA and density estimation with

$$\int \widehat{p}(x)^2 dx = \frac{1}{N^2} 1_v^T \Omega 1_v \tag{8.40}$$

where a normalized kernel is assumed with respect to density estimation. One chooses a fixed size M then and actively selects points from the pool of training data as candidate support vectors (Fig. 8.7). In the working set of support vectors a point is randomly selected and replaced by a randomly selected point from the training data set if the new point improves the entropy criterion. Illustrative examples for a regression and classification problem are shown in Fig. 8.9-8.10.

The fixed size LS-SVM method is suitable for *adaptive* signal processing applications where on-line updating of w, b and recursive methods for the eigenvalue decomposition can be applied. Both for recursive least squares and singular value decomposition updating, various efficient algorithms have been developed in the literature, which can be used at this point. Also for transductive inference the use of fixed size LS-SVM is natural due to the fact that the search of support vectors is done in an unsupervised way. In *transductive inference* [41, 43] one is interested in finding a model which is performing well on a specific future data point (or set of data points) rather than a general model (as obtained by inductive inference). If for fixed size LS-SVM one knows future data (in addition to the given training data) on which the model should perform well, then one may take these points into account (without class labels or target values) for possible selection of support vectors.

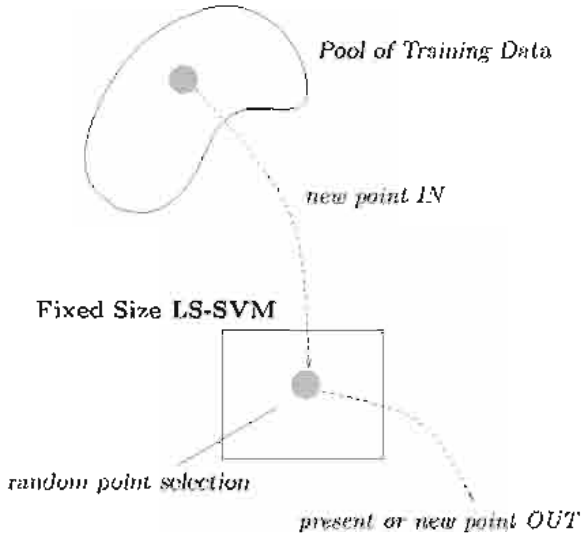


Figure 8.7: Fixed size LS-SVM: the number of support vectors is fixed beforehand and the support vectors are actively selected from the pool of training data. After estimating eigenfunctions the model is computed in the primal space with calculation of w, b. In the working set of support vectors a point is randomly selected and replaced by a randomly selected point from the training data if this new point improves an entropy criterion which is related to density estimation and kernel PCA.

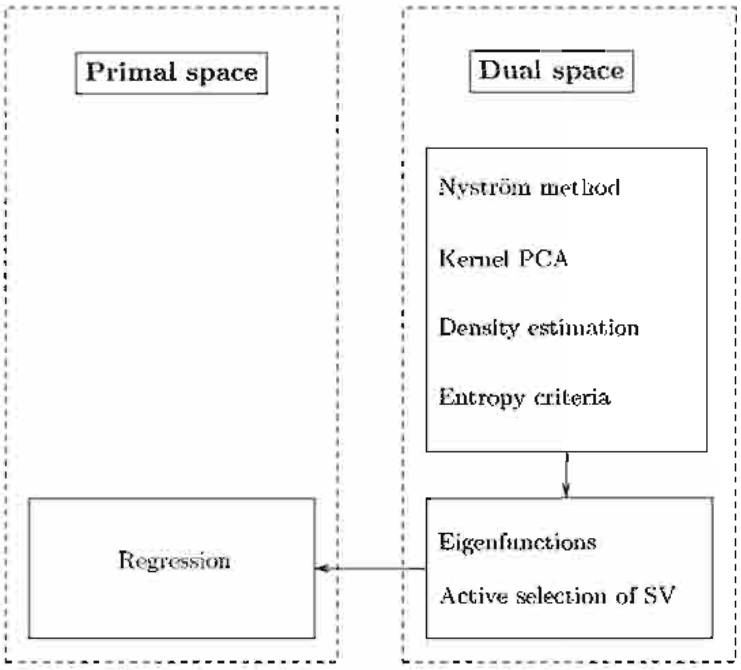


Figure 8.8: In the method of fixed size LS-SVM the Nyström method, kernel PCA and density estimation are linked to estimation of eigenfunctions and active selection of support vectors. Regression in the primal space is done in a next step.

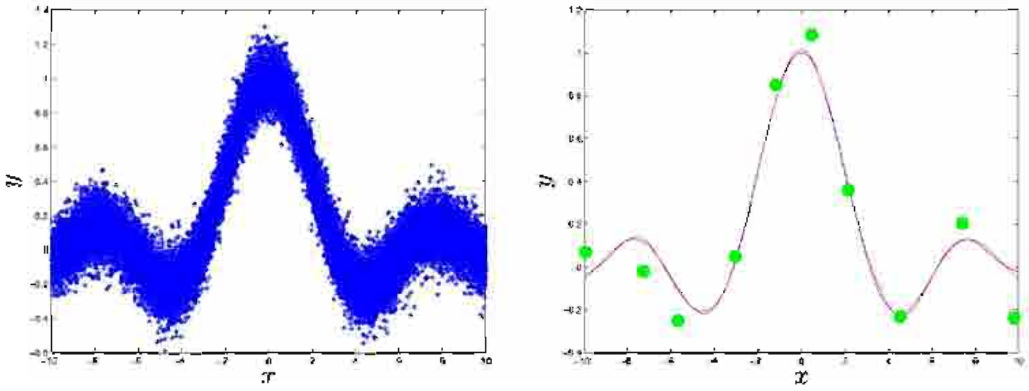


Figure 8.9: Fixed size LS-SVM for a noisy sinc function with $M = 10$ and $N = 20000$. (Left) given training set; (Right) true sinc function (solid line), estimate of Fixed Size LS-SVM (dashed line), and support vectors (grey dots).

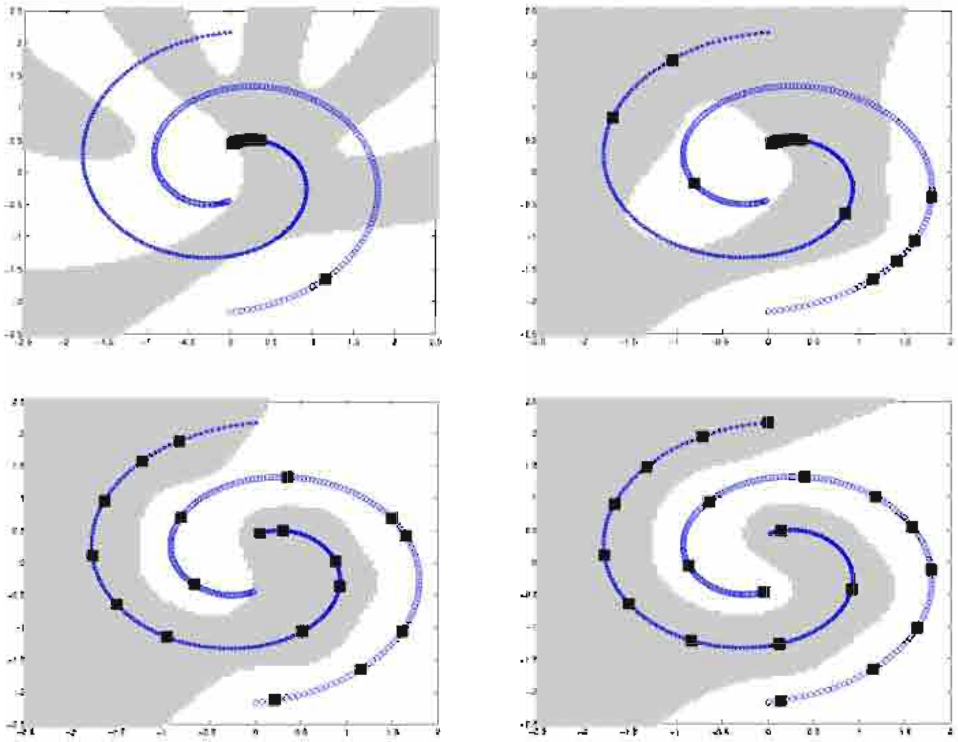


Figure 8.10: Illustration of fixed size LS-SVM on a double spiral classification problem with $M = 20$ support vectors and $N = 400$ given training data. The figures show different stages in the learning process.

8.5 Recurrent Networks and Control

For LS-SVM models also extensions have been made towards recurrent networks and use for optimal control. From a traditional systems identification point of view, one usually first considers a certain model structure (such as NARX, NARMAX, NOE,

nonlinear state space etc.) and then one parameterizes the model structure and solves an optimization problem in the unknown parameter vector [31]. While for classical parametric models such as MLPs and RBF networks one can parameterize both NARX (Nonlinear ARX) and NOE (Nonlinear Output Error) models in a straightforward way (which leads to feedforward and recurrent networks, respectively), the situation is more difficult in the SVM context, due to the non-parametric nature of the method.

In [36] it has been a recurrent LS-SVM model has been formulated, which preserves certain primal-dual interpretations and uses of the kernel trick. However, due to the fact that the equality constraints of the problem formulation are nonlinear in the unknowns instead of linear, the recurrent LS-SVM problem becomes non-convex. The approach has been successfully illustrated on trajectory learning and prediction of chaotic systems.

The fact that SVM theory makes use of elements of optimization theory and optimal control theory is closely related to optimization theory (in the area of model predictive control these links are even more explicit), the study and use of SVM methodology within control theory becomes a natural open question. In [37] first attempts and results are made in this direction, where LS-SVM formulations for nonlinear state feedback controllers are merged with finite time horizon optimal control problems. After formulating the primal problems and taking conditions for optimality from the Lagrangian, the solution is characterized by a set of nonlinear equations for which the kernel trick can be applied.

8.6 Conclusions

In this chapter a brief overview is given of least squares approaches to SVMs. Several extensions to present SVM formulations in classifications and static nonlinear function estimation can be made in terms of least squares and equality constraints based formulations. In this way it becomes straightforward to extend the methods to nonlinear kernel versions of existing linear techniques in pattern recognition and statistics (such as FDA, PCA, CCA, PLS) and make extensions to areas as recurrent networks and control. In addition the links between various related kernel based methods as SVMs, regularization networks, Gaussian processes, kriging and kernel ridge regression become often more explicit. For LS-SVMs the primal-dual optimization problem formulations are emphasized and exploited. It leads for example to a fixed-size version which is capable of handling very large data sets, being related to the Nyström method, and is at the same time very natural towards adaptive signal processing and transductive inference. This fixed-size LS-SVM technique enables to consider modelling in the primal space (parametric) as well as the dual space (non-parametric). The former representation is suitable for handling large data sets while the latter is convenient for problems with large dimensional inputs. It also offers a framework for studying such problems and its many related applications in an highly interdisciplinary manner[2].

[2]For a more extensive discussion of this chapter we refer to the book [33]. A related Matlab/C toolbox LS-SVMlab is available at **http://www.esat.kuleuven.ac.be/sista/lssvmlab/**.

Bibliography

[1] F.R. Bach, M.I. Jordan, Kernel independent component analysis, *Journal of Machine Learning Research* **3** (2002) 1–48.

[2] G. Baudat, F. Anouar, Generalized discriminant analysis using a kernel approach, *Neural Computation* **12** (2000) 2385–2404.

[3] C.M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press (1995)

[4] N. Cressie, *Statistics for Spatial Data*, John Wiley & Sons, New York (1993)

[5] N. Cristianini, J. Shawe-Taylor, *An Introduction to Support Vector Machines*, Cambridge University Press (2000)

[6] L. Csató, M. Opper, Sparse on-line Gaussian processes, *Neural Computation* 14(3) (2002) 641–668.

[7] F. Cucker, S. Smale, On the mathematical foundations of learning theory, *Bulletin of the AMS* **39** (2002) 1–49.

[8] T. Evgeniou, M. Pontil, T. Poggio, Regularization networks and support vector machines, *Advances in Computational Mathematics* **13**(1) (2000) 1–50.

[9] M. Girolami, Orthogonal series density estimation and the kernel eigenvalue problem, *Neural Computation* 14(3) (2002) 669–688.

[10] F. Girosi, An equivalence between sparse approximation and support vector machines, *Neural Computation* **10**(6) (1998) 1455–1480.

[11] G.H. Golub, C.F. Van Loan, *Matrix Computations*, Baltimore MD: Johns Hopkins University Press (1989)

[12] F.R. Hampel, E.M. Ronchetti, P.J. Rousseeuw, W.A. Stahel, *Robust Statistics, the Approach Based on Influence Functions*, John Wiley & Sons, New York (1986)

[13] B. Hassibi, D.G. Stork, Second order derivatives for network pruning: optimal brain surgeon, In Hanson, Cowan, Giles (Eds.) *Advances in Neural Information Processing Systems*, 5, San Mateo, CA: Morgan Kaufmann (1993) 164–171.

[14] L. Hoegaerts, J.A.K. Suykens, J. Vandewalle, B. De Moor, Kernel PLS variants for regression, *European Symposium Artificial Neural Networks* (ESANN 2003) (2003) to appear.

[15] H. Hotelling, Relations between two sets of variates, *Biometrica* **28** (1936) 321–377.

[16] P.J. Huber, *Robust Statistics*, John Wiley & Sons, New York (1981)

[17] I.T. Jolliffe, *Principal Component Analysis*, Springer Series in Statistics, Springer-Verlag (1986)

[18] S.S. Keerthi, S.K. Shevade, SMO algorithm for Least Squares SVM formulations, *Neural Computation*, to appear (2003)

[19] G.S. Kimeldorf, G. Wahba, A correspondence between Bayesian estimation on stochastic processes and smoothing by splines, *Ann. Math. Statist.* **2** (1971) 495–502.

[20] D.G. Krige, A statistical approach to some basic mine valuation problems on the Witwatersrand, *J. Chem. Metall. Mining Soc. S. Africa* **52**(6) (1951) 119–139.

[21] Y. Le Cun, J.S. Denker, S.A. Solla, Optimal brain damage, In Touretzky (Ed.) *Advances in Neural Information Processing Systems* San Mateo, CA: Morgan Kaufmann 2 (1990) 598–605.

[22] D.J.C. MacKay, Bayesian interpolation, *Neural Computation* **4**(3) (1992) 415–447.

[23] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, K.-R. Müller, Fisher discriminant analysis with kernels, In Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, editors, *Neural Networks for Signal Processing IX*, IEEE (1999) 41–48.

[24] T. Poggio, F. Girosi, Networks for approximation and learning, *Proceedings of the IEEE* **78**(9) (1990) 1481–1497.

[25] J. Principe, J. Fisher, D. Xu, Information theoretic learning, in S. Haykin (Ed.), *Unsupervised Adaptive Filtering*, John Wiley & Sons, New York (2000).

[26] R. Rosipal, L.J. Trejo, Kernel partial least squares regression in reproducing kernel Hilbert space, *Journal of Machine Learning Research* **2** (2001) 97–123.

[27] C. Saunders, A. Gammerman, V. Vovk, Ridge regression learning algorithm in dual variables, *Proc. of the 15th Int. Conf. on Machine Learning (ICML-98)*, Madison-Wisconsin (1998) 515–521.

[28] B. Schölkopf, A. Smola, K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, *Neural Computation* **10** (1998) 1299–1319.

[29] B. Schölkopf, S. Mika, C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, A. Smola, Input space vs. feature space in kernel-based methods, *IEEE Transactions on Neural Networks* **10**(5) (1999) 1000–1017.

[30] B. Schölkopf, A. Smola, *Learning with Kernels*, MIT Press, Cambridge, MA (2002).

[31] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P.-Y. Glorennec, H. Hjalmarsson, A. Juditsky, Nonlinear black-box modeling in system identification: a unified overview, *Automatica* **31**(12) (1995) 1691–1724.

[32] A.J. Smola, B. Schölkopf, Sparse greedy matrix approximation for machine learning, in P. Langley (Ed.) *Proc. 17th International Conference on Machine Learning,* 911–918, San Francisco, Morgan Kaufman (2000) 911–918,

[33] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, *Least Squares Support Vector Machines*, World Scientific, Singapore (2002).

[34] J.A.K. Suykens, J. Vandewalle, Least squares support vector machine classifiers, *Neural Processing Letters* **9**(3) (1999) 293–300.

[35] J.A.K. Suykens, J. De Brabanter, L. Lukas, J. Vandewalle, Weighted least squares support vector machines: robustness and sparse approximation, *Neurocomputing*, Special issue on fundamental and information processing aspects of neurocomputing **48**(1-4) (2002) 85–105.

[36] J.A.K. Suykens, J. Vandewalle, Recurrent least squares support vector machines, *IEEE Transactions on Circuits and Systems-I* **47**(7) (2000) 1109–1114.

[37] J.A.K. Suykens, J. Vandewalle, B. De Moor, Optimal control by least squares support vector machines, *Neural Networks* **14**(1) (2001) 23–35.

[38] J.A.K. Suykens, T. Van Gestel, J. Vandewalle, B. De Moor, A support vector machine formulation to PCA analysis and its kernel version, *IEEE Transactions on Neural Networks* (2003) in press.

[39] T. Van Gestel, J.A.K. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B. De Moor, J. Vandewalle, Benchmarking least squares support vector machine classifiers, *Machine Learning* (2003) in press.

[40] T. Van Gestel, J.A.K. Suykens, G. Lanckriet, A. Lambrechts, B. De Moor, J. Vandewalle, A Bayesian framework for least squares support vector machine classifiers, Gaussian processes and kernel Fisher discriminant analysis, *Neural Computation* **14**(5) (2002) 1115–1147.

[41] V. Vapnik, *Estimation of Dependencies based on Empirical Data*, Springer-Verlag, New York (1982).

[42] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, New York (1995).

[43] V. Vapnik, *Statistical Learning Theory*, John Wiley & Sons, New York (1998)

[44] G. Wahba, *Spline Models for Observational Data*, Series in Applied Mathematics **59**, SIAM, Philadelphia (1990)

[45] C.K.I. Williams, C.E. Rasmussen, Gaussian processes for regression, In D.S. Touretzky, M.C. Mozer, and M.E. Hasselmo (Eds.), *Advances in Neural Information Processing Systems* 8 MIT Press (1996) 514–520.

[46] C.K.I. Williams, M. Seeger, Using the Nyström method to speed up kernel machines, In T.K. Leen, T.G. Dietterich, and V. Tresp (Eds.), *Advances in Neural Information Processing Systems*, 13, MIT Press (2001) 682–688.

Chapter 9

Extension of the ν-SVM Range for Classification

Fernando Pérez-Cruz, Jason Weston, Daniel J. L. Herrmann and Bernhard Schölkopf[1]

Abstract. The ν-Support Vector Machine for classification (ν-SVC) has been presented as a different formulation for solving SVMs in which the C parameter is transformed by a more meaningful parameter ν, that roughly represents the fraction of support vectors. The value of ν cannot always take all possible values between 0 and 1, which limits the range of possible solutions. Either, because the training set is non-separable in the feature space, or because the classes are unbalanced. In this chapter, we will deal with both restrictions, presenting a new Extended ν-SVC, in which the value of ν can move from 0 to 1 in any circumstance. The modification to extend the range up to 1 is trivial, we only need to modify the cost associated to the margin errors to balance the classes. The modification to extend the range down to zero is far more complex. We will first need to revisit how maximum margin classifiers can be obtained for a separable training set, to enable us to construct "hard" margin classifiers for non-separable datasets. This can be achieved by finding the separation in which incorrectly classified samples have the smallest *negative* margin. This re-interpretation of the maximum margin classifier, when viewed as a soft margin formulation, will allow us to extend the range of ν-SVC to any number of support vectors. Experiments with real and synthetic data confirm the validity of the proposed approach.

[1]We would like to express our gratitude to Chih-Jen Lin for its helpful comments for making the chapter more readable and avoiding the introduction of major mistakes.

9.1 Introduction

Support Vector Machines (SVMs) are state-of-the-art tools for linear and nonlinear knowledge discovery [14]. They were initially developed for linearly separable problems, known as the optimal hyperplane decision rule (OHDR) [14]. In a nutshell, the OHDR finds the classification boundary that linearly separates the given data and is furthest from the data. The OHDR is computed as the maximization of the minimum distance of the samples to the separating hyperplane. This minimum distance is known as the margin, and the OHDR is also known as the maximum margin classifier. The maximum margin classifiers were generalized to cover nonlinear problems, through the "kernel trick" [14, 2]; and non-separable problems, via slack variables that relax the conditions in the original formulation [29, 6].

The SVM, when solved for nonlinear problems, has to set the value of a weight parameter C which measures the trade off between the training errors and the maximization of the margin. This weight is hard to choose a priori and it is difficult to infer which result can be expected for a C value over any given problem. There is an alternative formulation, known as ν-SVM, in which the weight parameter is replaced by another more intuitive parameter ν. This parameter roughly represents the fraction of expected support vectors, therefore for any given $\nu \in (0, 1]$, we will know a priori how the classifier will be formed. Also, it allows to easily scan the whole range of possible solutions, because choosing ν between 0 and 1 will give all the possible outcomes.

The ν-SVM for classification (ν-SVC) has a limitation in terms of its usable range. The value of ν can be upper bounded by a value less than 1, if the classes are not balanced [7], and it can be lower bounded by a value greater than 0 for some data sets, if the VC dimension of the used classifier is not infinite [4]. These two limitations also exist in the formulation using the C parameter, although they are not explicit with this parameter, explaining why they had not been previously addressed as limitations. The first limitation can be easily avoided if one requires so, as we will show herein. The second is not readily overcome and the main body of this article is devoted to it.

We will propose a reinterpretation on how maximum margin hyper-planes are constructed. This reinterpretation will lead to a unified formulation for both separable and inseparable sets: a maximum positive margin solution, if the training data set is separable, and a minimum negative margin solution (to be described in the following sections), if the training samples are not. This unified formulation will be constructed using a ν-SVM type parameterization and, consequently, we will be able to control the number of SVs for the full range of possible values of ν. We will refer to it as Extended ν-SVM (Eν-SVM). Therefore, we will be able to select the SVM optimal solution from the whole range of possible solutions, i.e., all the solutions with any number of SVs.

We will start with a full description of the ν-SVM and its relationship with SVMs (using the C parameter, also known as C-SVM) in Section 9.2. Then in Section 9.3, we will focus on its two limitations, the upper and lower bounds over the value of ν. The first can be easily overcome by re-weighting the errors in the ν-SVM. The second will need a further study, we will define a negative margin classifier and how they have to be solved in Section 9.4. We will show that the negative margin classifiers can be expressed in a unified formulation similar to ν-SVM in Section 9.5, which we will refer to as Extended ν-SVM. In Section 9.6, We show by means of computer experiments

the validity of the proposed approach using real and synthetic data. We will end with some concluding remarks in Section 9.7.

9.2 ν Support Vector Classifiers

The Support Vector Machine for binary classification (C-SVC) [20] finds the optimum of a quadratic constrained problem:

$$\min_{\mathbf{w},\xi_i,b} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \tag{9.1}$$

subject to

$$y_i(\phi^T(\mathbf{x}_i)\mathbf{w} + b) \geq 1 - \xi_i \qquad \forall i = 1,\ldots,n \tag{9.2}$$

$$\xi_i \geq 0 \qquad \forall i = 1,\ldots,n \tag{9.3}$$

where the data set $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots (\mathbf{x}_n, y_n)$ ($\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{\pm 1\}$) has been sampled independently and identically distributed (i.i.d.) from a joint probability density function $p(\mathbf{x}, y)$ that relates each vector $\mathbf{x}_i$ with its corresponding class label y_i. The nonlinear mapping $\phi(\cdot)$ ($\mathbb{R}^d \xrightarrow{\phi(\cdot)} \mathbb{R}^{\mathcal{H}}$) transforms the input data to a higher dimensional space, the feature space $\mathcal{H}$. The linear classifier $(\mathbf{w}, b)$ in the feature space is usually nonlinear in the input space, unless $\phi(\mathbf{x}) = \mathbf{x}$.

In the above formulation, C is a parameter determining the trade-off between two conflicting goals: minimizing the training error, and maximizing the margin. Unfortunately, C is a rather unintuitive parameter, and we have no a priori way to select it. Therefore, a modification was proposed in [16], which replaces C by a parameter ν; the latter will turn out to control the number of margin errors and, consequently, the Support Vectors (SVs).

As a primal problem for this approach, termed the ν-SVM for classification (ν-SVC), we consider

$$\min_{\mathbf{w},\xi_i,\rho,b} \frac{1}{2}\|\mathbf{w}\|^2 - \nu\rho + \frac{1}{n}\sum_{i=1}^{n}\xi_i \tag{9.4}$$

subject to (9.3) and

$$y_i(\phi^T(\mathbf{x}_i)\mathbf{w} + b) \geq \rho - \xi_i \qquad \forall i = 1,\ldots,n \tag{9.5}$$

$$\rho \geq 0 \tag{9.6}$$

A new non-negative variable ρ has been included in the objective functional and has to be minimized. However, it has been shown [7] that the constraint enforcing ρ to be positive, (9.6), is unnecessary and that the above optimization problem will always end with a ρ greater or equal than 0. Intuitively, if a solution with $\rho \leq 0$ is feasible, we can set $\mathbf{w} = 0$, $b = 0$ and $\xi_i = 0$, which will fulfill the constraint in (9.5), and will give the lowest possible values of the first and third terms of (9.4), $\rho = 0$ being the one that minimizes it most. Therefore, a negative value of ρ cannot reduce the value of (9.4). Also, it can be shown that the functional (9.4) cannot become negative (it can

Table 9.1: Fractions of errors and SVs, along with the margins of class separation, for the toy example. Note that ν upper bounds the fraction of errors and lower bounds the fraction of SVs, and that increasing ν, i.e., allowing more margin errors, increases the margin.

ν	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
fraction of errors	0.00	0.07	0.25	0.32	0.39	0.50	0.61	0.71
fraction of SVs	0.29	0.36	0.43	0.46	0.57	0.68	0.79	0.86
margin $\rho/\|\mathbf{w}\|$	0.005	0.018	0.115	0.156	0.364	0.419	0.461	0.546

be readily seen from the dual of this problem), therefore the solution in which (9.4) is zero cannot be improved.

To explain the significance of ν, let us first define the term *margin error*: by this, we denote points with $\xi_i > 0$. These are points which are either training errors ($\xi_i > \rho$), or lie within the margin ($\xi_i \in (0, \rho]$). Formally, the fraction of margin errors is

$$R^{\rho}_{\text{emp}}[\mathbf{w}, b] := \frac{1}{n} \left| \{ i | y_i (\phi^T(\mathbf{x}_i)\mathbf{w} + b) < \rho \} \right| . \tag{9.7}$$

The following proposition was stated and proven in [16] and it allows to understand the role of ν and what to expect once the solution has been reached.

Proposition 1 *Suppose we run ν-SVC with a given kernel on some data with the result that $\rho > 0$. Then*

(i) ν is an upper bound on the fraction of margin errors.

(ii) ν is a lower bound on the fraction of SVs.

(iii) Suppose the data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ were generated i.i.d. from a distribution $p(\mathbf{x}, y) = p(\mathbf{x})p(y|\mathbf{x})$, such that neither $p(\mathbf{x}, y = 1)$ nor $p(\mathbf{x}, y = -1)$ contains any discrete component. Suppose, moreover, that the kernel used is analytic and non-constant. With probability 1, asymptotically, ν equals both the fraction of SVs and the fraction of errors.

We would like to show with a toy example the solutions that one can expect when solving the ν-SVM, before explaining how it is actually solved. We show in Figure 9.1 the solution for various different ν for a two dimensional problem solved with a Gaussian kernel. The fraction of margin errors and support vectors, discussed in the previous proposition, can be seen in Table 9.1.

The ν-SVM for classification can include the linear restrictions in the objective functional using Lagrange multipliers, requiring one to minimize

$$L(\mathbf{w}, \xi_i, b, \rho, \alpha_i, \mu_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \nu\rho + \frac{1}{n}\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} (\alpha_i (y_i (\phi^T(\mathbf{x}_i)\mathbf{w} + b) - \rho + \xi_i) + \mu_i \xi_i), \tag{9.8}$$

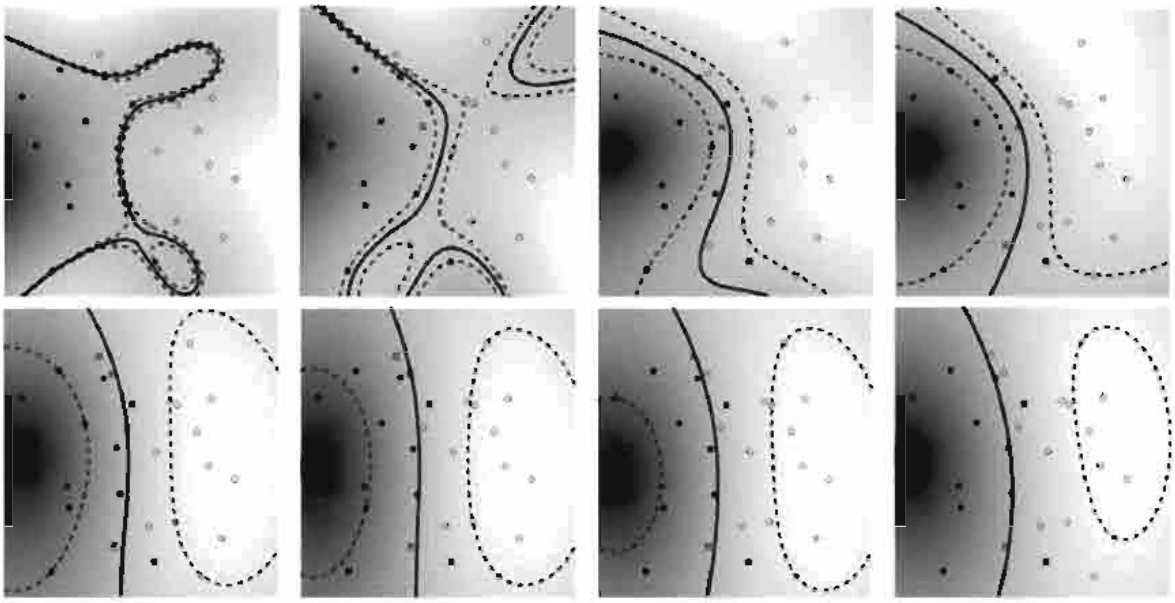


Figure 9.1: Toy problem (task: separate circles from disks) solved using ν-SV classification, with parameter values ranging from $\nu = 0.1$ (top left) to $\nu = 0.8$ (bottom right). The larger we make ν, the more points are allowed to lie inside the margin (depicted by dotted lines). Results are shown for a Gaussian kernel, $k(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2)$ [14].

with respect to $\mathbf{w}$, ξ_i, ρ and b and maximize it with respect to the Lagrange multipliers, $\alpha_i, \beta_i \geq 0$. We have not imposed the condition in (9.6), following [7]. The solution to this problem is given by the Karush-Kuhn-Tucker Theorem [9], that imposes the following conditions: (9.5), (9.3) and

$$\frac{\partial L_p}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \phi(\mathbf{x}_i) = \mathbf{0} \tag{9.9}$$

$$\frac{\partial L_p}{\partial b} = -\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{9.10}$$

$$\frac{\partial L_p}{\partial \rho} = \sum_{i=1}^{n} \alpha_i - \nu = 0 \tag{9.11}$$

$$\frac{\partial L_p}{\partial \xi_i} = \frac{1}{n} - \alpha_i - \mu_i = 0 \qquad \forall i = 1, \ldots, n \tag{9.12}$$

$$\alpha_i, \mu_i \geq 0 \qquad \forall i = 1, \ldots, n \tag{9.13}$$

$$\alpha_i \{ y_i (\phi^T(\mathbf{x}_i)\mathbf{w} + b) - \rho + \xi_i \} = 0 \qquad \forall i = 1, \ldots, n \tag{9.14}$$

$$\mu_i \xi_i = 0 \qquad \forall i = 1, \ldots, n \tag{9.15}$$

which are known as the KKT conditions.

The ν-SVC, like the C-SVC, gives the solution as a linear combination of the samples in the feature space (9.9), called the SV expansion. The α_i that are non-zero correspond to a constraint (9.5) which is precisely met.

The regular way of solving Support Vector Machines is by substituting (9.9) to (9.12) into L in (9.8), leaving us with the following quadratic optimization problem for

ν-SV classification:

$$\max_{\alpha_i} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i,\mathbf{x}_j) \tag{9.16}$$

subject to

$$0 \le \alpha_i \le \frac{1}{n}, \tag{9.17}$$

$$\sum_{i=1}^{n}\alpha_i y_i = 0, \tag{9.18}$$

$$\sum_{i=1}^{n}\alpha_i = \nu. \tag{9.19}$$

The $k(\cdot,\cdot)$ represents a dot product (kernel) of two variables in the feature space, $k(\cdot,\cdot) = \boldsymbol{\phi}^T(\cdot)\boldsymbol{\phi}(\cdot)$. The conditions for any function to be a kernel in a Hilbert space is given by the Mercer theorem [3].

The resulting decision function can be expressed as a linear combination of kernels:

$$\begin{aligned} f(x) &= \operatorname{sgn}\left(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}) + b\right) = \\ &= \operatorname{sgn}\left(\sum_{i=1}^{n}\alpha_i y_i \boldsymbol{\phi}^T(\mathbf{x}_i)\boldsymbol{\phi}(\mathbf{x}) + b\right) = \operatorname{sgn}\left(\sum_{i=1}^{n}\alpha_i y_i k(\mathbf{x},\mathbf{x}_i) + b\right). \end{aligned} \tag{9.20}$$

Therefore, we do not need to specify the whole nonlinear mapping, only its kernel.

To compute the threshold b and the margin parameter ρ, we consider two sets $S_\pm$, containing SVs $\mathbf{x}_i$ with $0 < \alpha_i < 1/n$ and $y_i = \pm 1$, respectively. We choose $s = \min(|S_+|, |S_-|)$, and limit the larger $S_\pm$ set to contain s elements. Then, due to the KKT conditions, (9.5) becomes an equality with $\xi_i = 0$ for all the samples in $S_\pm$. Hence, in terms of kernels,

$$b = -\frac{1}{2s}\sum_{\mathbf{x}\in S_+\cup S_-}\sum_{j=1}^{n}\alpha_j y_j k(\mathbf{x},\mathbf{x}_j), \tag{9.21}$$

$$\rho = \frac{1}{2s}\Big(\sum_{\mathbf{x}\in S_+}\sum_{j=1}^{n}\alpha_j y_j k(\mathbf{x},\mathbf{x}_j) - \sum_{\mathbf{x}\in S_-}\sum_{j=1}^{n}\alpha_j y_j k(\mathbf{x},\mathbf{x}_j)\Big). \tag{9.22}$$

Note that for the decision function, only b is actually required.

In the case that either S_+ or S_- are the empty set, they will be, respectively, formed by a one element set:

$$S_+ = \underset{\mathbf{x}_i|\alpha_i\neq 0, y_i=1}{\arg\max} \left\{\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i)\right\}$$

and

$$S_- = \underset{\mathbf{x}_i|\alpha_i\neq 0, y_i=-1}{\arg\min} \left\{\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i)\right\}$$

as detailed in [8] for the C-SVM.

A connection to standard SV classification, and a somewhat surprising interpretation of the regularization parameter C, is described by the following result:

Proposition 2 (Connection ν-SVC — C-SVC [16]) *If ν-SV classification leads to $\rho > 0$, then C-SV classification, with C set a priori to $1/\rho$, leads to the same decision function.*

The proof of this proposition can be found in [16]. This proposition ensures that any ν providing a non-trivial solution[2] with $\rho \neq 0$, a C value can be obtained for the C-SVC formulation that will lead to the same solution obtained with such ν, up to a scaling factor, which is the value of C. For further details on the connection between ν-SVMs and C-SVMs see [7, 1].

9.3 Limitation in the Range of ν

A complete account of the relation between the C-SVM and ν-SVM has been given in [4], where they have shown that the value of ν can not always take the full range from 0 to 1. They have stated and proven the following theorem in which the maximum and minimum value of ν are bounded:

Theorem 1 *We can define*

$$\nu_* = \lim_{C \to \infty} \frac{1}{nC} \sum_{i=1}^{n} \alpha_i^C \tag{9.23}$$

and

$$\nu^* = \lim_{C \to 0} \frac{1}{nC} \sum_{i=1}^{n} \alpha_i^C \tag{9.24}$$

where α_i^C are the Lagrange multipliers associated with the constraints in (9.2) *in the C-SVM and $\nu_* > 0$ and $\nu^* \leq 1$. For any $\nu > \nu^*$* (9.16) *is infeasible and for any $0 < \nu \leq \nu_*$* (9.16) *is feasible with zero optimal objective value (the trivial solution). For $\nu_* < \nu \leq \nu^*$* (9.16) *is feasible and its solution is equal) to the solution of the dual of* (9.1)*, up to a scaling factor ($\alpha_i^C = Cn\alpha_i''$).*

We will not enter in the demonstration of the theorem, which is detailed in [4], but we will give some intuitions of the results provided by the theorem. The minimal value of ν for which (9.4) is nonzero (greater than 0) was discussed earlier in this section, once ρ becomes zero (with $\mathbf{w} = 0$ and $b = 0$) there is no incentive in the objective functional (9.4) to make ρ go negative. Therefore, it can be seen that this will only happen if the kernel matrix $\mathbf{H}$ ($(\mathbf{H})_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$) is singular (not full-rank), because otherwise (9.16) will only be zero iff $\alpha_i = 0\ \forall i$. The solutions of the ν-SVM in which ν lies between ν_* and ν^* are feasible and meaningful, and its relationship with C-SVM is the one stated by Proposition 2.

Finally, the infeasibility of (9.16) for ν greater than ν^* is easily understood when examining restrictions (9.18) and (9.19). These two restrictions enforce that the sum of the α_i of class $+1$ has to be equal to $\nu/2$ and equal to the sum of the α_i of class -1. Therefore, as the maximum contribution of each sample to this sum is $1/n$ (see KKT

[2] We understand by a trivial solution a value of ν that forces $\rho = 0$ and, consequently, $\mathbf{w} = 0$ and $b = 0$.

condition (9.12)), then the maximum value ν can take is $2\min(n^+, n^-)/n$, where n^+ and n^- are, respectively, the number of sample in class $+1$ and -1. If the classes are not balanced ν has to be less than 1, because $2\min(n^+, n^-) < n$.

In this contribution, we are interested in extending the range of ν-SVM for classification. We will first show that extending the range up to 1 is readily obtained and we will dedicate the rest of the chapter to detail how the range can be extended for a value of ν less than ν_* without ending in the trivial solution ($\rho = 0$, $\mathbf{w} = 0$ and $b = 0$).

To extended the value of ν up to 1 ($\nu = 1$), we will need that $\sum_{i=1|y_i=1}^{n} \alpha_i = \sum_{i=1|y_i=-1}^{n} \alpha_i = 1/2$, which can be obtained if we set, respectively, $\alpha_i = 1/(2n^+)$ and $\alpha_i = 1/(2n^-)$, if the samples belongs to class $+1$ or class -1. As the maximum value α_i can take is the multiplicative factor for ξ_i in (9.4), we can then extend the range by considering a different penalty factor for the ξ_i of positive and negative samples, leading to the modification of (9.4) by:

$$\min_{\mathbf{w},\boldsymbol{\xi},\rho,b} \frac{1}{2}\|\mathbf{w}\|^2 - \nu\rho + \frac{1}{2n^+}\sum_{i=1|y_i=1}^{n} \xi_i + \frac{1}{2n^-}\sum_{i=1|y_i=-1}^{n} \xi_i \tag{9.25}$$

This modification is similar to the one proposed in [13, 11] for solving the C-SVM with a different cost for positive and negative classes in unbalanced problems, to obtain a good balanced error performance.

Proposition 1 still holds and it will also hold for each class independently. Using (9.25) the value of ν will be an upper bound for the fraction margin errors of class $+1$ (class -1) and will be a lower bound for the fraction of support vectors of class $+1$ (class -1), which did not hold for the previous formulation.

9.4 Negative Margin Minimization

We will now address the problem of reducing the value of ν below ν_* without being led to the trivial solution. To do so, we will need to revisit the regular SVM solution for linearly separable data sets and try to find a different solution for non-separable sets.

The SVM enforces a solution in which for positive class samples, $\mathbf{x}_i^T\mathbf{w} + b \geq \rho$, and for negative class samples, $\mathbf{x}_i^T\mathbf{w} + b \leq -\rho$, where $\mathbf{w}$ and b define a linear classifier with $\rho > 0$. Among the solutions that fulfill these requirements the SVM picks the one in which the samples are the furthest apart from the classification boundary (the maximum margin solution) [20], as shown in Figure 9.2a. To construct the maximum margin solution, the SVM fixes ρ to 1 and minimizes $\|\mathbf{w}\|^2$. For non-separable problems the SVM includes slack variables in the previous constraints and minimizes the one-norm of the slack variables (to approximately minimize the number of training errors), leading to the optimization of (9.1) subject to (9.2) and (9.3).

A typical solution of a non-separable problem is shown in Figure 9.2b, in which there are 17 SVs. This solution presents the least number of SVs, and no other value of C will reduce it, therefore the minimum value of ν will be between $14/60 < \nu \leq 17/60$. But, analyzing the obtained solution for the separable problem in Figure 9.2a, one could expect the solution for a non-separable problem to be the one shown in Figure 9.2c, in which the solution is obtained with the extreme vectors as in Figure 9.2a, instead of

the one shown in Figure 9.2b. In Figure 9.2a we have an exclusion zone (no training sample is allowed to have a machine output, $\mathbf{x}_i^T\mathbf{w} + b$, between $-\rho$ and $+\rho$) and in Figure 9.2c we have an intersection zone, into which samples from both classes can be placed without becoming SVs. This intersection zone can be implemented through the use of the following constraint:

$$\begin{aligned} \mathbf{x}_i^T\mathbf{w} + b &\geq -1 \qquad \text{if } y_i = +1 \\ \mathbf{x}_i^T\mathbf{w} + b &\leq +1 \qquad \text{if } y_i = -1 \end{aligned}$$

To obtain the maximum margin solution in Figure 9.2a, we maximize the exclusion zone ($\min \|\mathbf{w}\|^2$) forcing the samples to be as far apart from the classification boundary as possible. To obtain the solution in Figure 9.2c, we would like the intersection zone to be as small as possible to reduce the number of samples that lie inside it (i.e to minimize the number of potential errors). To reduce the intersection zone, we will have to minimize $1/\|\mathbf{w}\|$ ($\min -\|\mathbf{w}\|^2$).

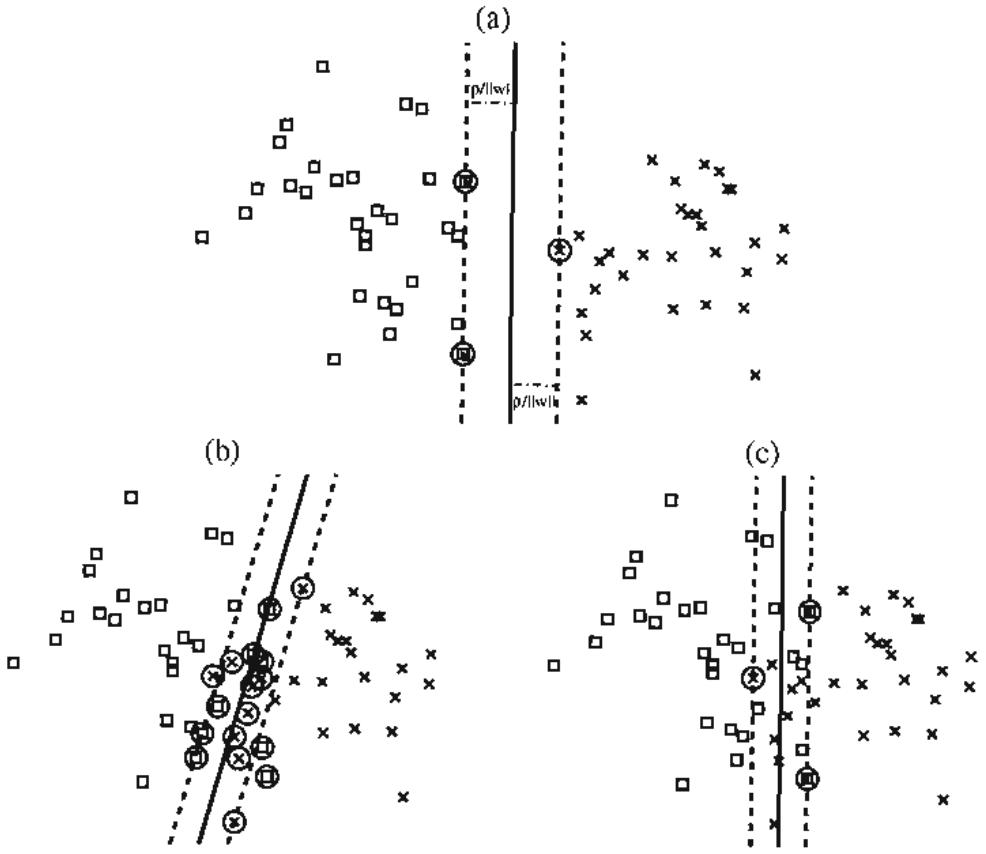


Figure 9.2: The solid lines represent the classification boundary, the dashed lines the $\pm\rho/\|\mathbf{w}\|$ margins. Class $+1$ is shown by crosses, Class -1 by squares, and the SVs are ringed. In (a) we show the maximum margin solution for a linearly separable data set. In (b) we show the SVM solution for a non-linearly separable data set. In (c) we show the solution with the negative margin classifier in which the solution is constructed by the extreme vectors as in (a).

The maximum of the positive margin and the minimum of the negative margin can be unified in a single optimization problem by looking back at how the maximum margin is constructed. In Figure 9.2a the solution can be obtained by solving:

$$\max_{\rho,\mathbf{w},b} \frac{\rho}{\|\mathbf{w}\|}$$

subject to

$$y_i(\mathbf{x}_i^T\mathbf{w} + b) \geq \rho$$

As there is a multiplicative factor between $\rho/\|\mathbf{w}\|$, there are infinitely many different solutions that only differ by a scaling factor. To resolve this, one can fix ρ and maximize $1/\|\mathbf{w}\|$ (or minimize $\|\mathbf{w}\|^2$ as the SVM does). Note that this only works for positive ρ, since for negative ρ, we would have to minimize $1/\|\mathbf{w}\|$. Alternatively one can fix $\|\mathbf{w}\|$, which accounts for a non-convex constrain, and maximize ρ. If this is the case and the problem is separable we will end up with the maximum margin solution and a positive ρ, for non-separable problems we will end up with a negative ρ and with the least possible intersection zone.

9.5 Extended ν-SVM

In the previous section, we have motivated how the hard maximum margin can be modified to deal with separable and non-separable training data sets. In this section, we will formulate the new learning problem with a ν-SVM like formulation (soft margin) that will allow us to control ρ in an intuitive way. We solve the problem:

$$\min_{\rho,\mathbf{w},b,\boldsymbol{\xi},} -n\nu\rho + \sum_{i=1}^{n} \xi_i \tag{9.26}$$

subject to:

$$y_i(\mathbf{x}_i^T\mathbf{w} + b) \geq \rho - \xi_i \qquad \forall i = 1,\ldots,n \tag{9.27}$$

$$\xi_i \geq 0 \qquad \forall i = 1,\ldots,n \tag{9.28}$$

$$\frac{1}{2}\|\mathbf{w}\|^2 = 1 \tag{9.29}$$

The objective function is linear but there is a non-convex constraint (9.29); therefore we can expect local minima in its solution[3], which have to be dealt with either using several initializations or starting at a controlled point. Note that in this case ν can not run up to 1 unless the classes are balanced. The modification needed is the same one proposed at the end of Section 9.3, but we have not included it to make the development of the Eν-SVM clearer.

We can solve this problem directly for linear classifiers by linearizing the quadratic constraint in (9.29). We select a starting point labeled as $\widetilde{\mathbf{w}}$ and we then replace (9.29)

[3]If the two data sets are not linearly separable, the problem of finding two half-spaces with parallel boundary and minimal overlap, containing the respective classes, is not convex. So this is not a weakness of the chosen approach, but an important aspect of the considered problem.

Table 9.2: Eν-SVC algoritmic implementation.

0. Initialize $\widetilde{\mathbf{w}}$.
1. Solve the linear problem:

$$\min_{\rho,\mathbf{w},b,\xi_i} -n\nu\rho + \sum_{i=1}^{n} \xi_i$$

subject to:

$$y_i(\mathbf{x}_i^T\mathbf{w} + b) \geq \rho - \xi_i,$$
$$\xi_i \geq 0 \text{ and } \widetilde{\mathbf{w}}^T\mathbf{w} = 2.$$

2. Compute $\widetilde{\mathbf{w}} = \gamma\widetilde{\mathbf{w}} + (1-\gamma)\mathbf{w}$.
3. If $\widetilde{\mathbf{w}} = \mathbf{w}$ end, otherwise go to Step 1.

by: $\widetilde{\mathbf{w}}^T\mathbf{w} = 2$. We thus get a linear problem that can be solved using any linear programming tool such as `linprog` from MATLAB®. Once we have computed the solution, we obtain a new $\widetilde{\mathbf{w}}$ and continue iterating until there is no further modification in either $\mathbf{w}$, b or ρ. To construct the new $\widetilde{\mathbf{w}}$ we do not directly use the value of $\mathbf{w}$ due to the linearizing step. We will construct it using a convex combination between $\widetilde{\mathbf{w}}$ and $\mathbf{w}$: $\widetilde{\mathbf{w}} = \gamma\widetilde{\mathbf{w}} + (1-\gamma)\mathbf{w}$. We have found experimentally that $\gamma = \frac{9}{10}$ is a good compromise value. We have written down an algorithmic implementation of the proposed approach in Table 9.2. The initial value of $\widetilde{\mathbf{w}}$ can be a random guess (not very convenient) or a solution to the ν-SVC with a ν above ν_*. The benefits of using this starting point is that we will be looking for a solution close to the best solution provided by the ν-SVC, which should do as a good starting point for avoiding local minima (or at least a bad local minima).

9.5.1 Kernelization in the dual

The above learning machine can be used to solve nonlinear problems using kernels. In order to construct a nonlinear classifier, we need to map non-linearly the vectors $\mathbf{x}_i$ to the feature space, through a nonlinear transformation $\phi(\cdot)$. To solve (9.26) with a possibly unknown $\phi(\cdot)$, we will introduce it in the constraints in (9.27)-(9.29) using Lagrange multipliers. This requires us to minimize

$$L_p = -n\nu\rho + \sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left(y_i(\phi^T(\mathbf{x}_i)\mathbf{w} + b) - \rho + \xi_i\right) - \sum \mu_i\xi_i + \lambda\left(\frac{1}{2}\|\mathbf{w}^2\| - 1\right) \quad (9.30)$$

with respect to $\rho, \mathbf{w}, b, \xi_i$ and maximize it with respect to the Lagrange multipliers, α_i, μ_i and λ. The solution to this problem is given by the Karush-Kuhn-Tucker (KKT)

theorem [9], which imposes the following conditions over (9.30):

$$\frac{\partial L_p}{\partial \rho} = -n\nu - \sum_{i=1}^{n} \alpha_i = 0 \tag{9.31}$$

$$\frac{\partial L_p}{\partial \mathbf{w}} = \lambda \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{\phi}(\mathbf{x}_i) = \mathbf{0} \tag{9.32}$$

$$\frac{\partial L_p}{\partial b} = -\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{9.33}$$

$$\frac{\partial L_p}{\partial \xi_i} = 1 - \alpha_i - \mu_i = 0 \qquad \forall i = 1, \ldots, n \tag{9.34}$$

$$\alpha_i, \mu_i \geq 0 \qquad \forall i = 1, \ldots, n \tag{9.35}$$

$$\alpha_i \{ y_i (\boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w} + b) - \rho + \xi_i \} = 0 \qquad \forall i = 1, \ldots, n \tag{9.36}$$

$$\mu_i \xi_i = 0 \qquad \forall i = 1, \ldots, n \tag{9.37}$$

$$\lambda \left(\frac{1}{2} \|\mathbf{w}\|^2 - 1 \right) = 0 \tag{9.38}$$

The dual formulation, which is the usual way of solving SVMs, cannot be used for solving the Eν-SVC, because it is not a convex problem and the dual formulation only holds for convex problems. But in our problem if $\lambda > 0$, the constraint in (9.29) can be transformed to $\frac{1}{2}\|\mathbf{w}\|^2 \leq 1$, which makes the problem convex. Therefore for positive λ, we can obtain the dual formulation by substituting (9.31), (9.32), (9.33), (9.34) into (9.30), in which one needs to maximize with respect to α_i and λ:

$$L_D = -\frac{1}{2\lambda} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \boldsymbol{\phi}^T(\mathbf{x}_i) \boldsymbol{\phi}(\mathbf{x}_j) - \lambda \tag{9.39}$$

subject to (9.31), (9.33) and $0 \leq \alpha_i \leq 1$. This problem cannot be easily solved because λ depends on α. This dependence can be obtain using (9.29) and (9.32):

$$\lambda = + \sqrt{\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j)} \tag{9.40}$$

To solve (9.39), we fix λ for example equal to 1. If in the solution the optimal objective value is nonzero one can compute λ as shown by (9.40). If it is zero, it will mean that the solution of the Eν-SVC will be obtained with a negative λ and the dual cannot be used. If we replace this new value in the functional the solution will be the same. Therefore, if λ is positive we only need to solve:

$$L_D = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \boldsymbol{\phi}^T(\mathbf{x}_i) \boldsymbol{\phi}(\mathbf{x}_j)$$

which is the same functional used for the ν-SVM. Therefore if λ is positive the solution for the ν-SVC and the Eν-SVC is the same one up to a scaling factor.

For negative values of λ, however, if one fixes λ and tries to optimize (9.39), will obtain $\sum_i \sum_j y_i y_j \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) = 0$ and one therefore one will not get a feasible solution. This is something already known, because for negative λ the constraint in (9.29) is non-convex and the dual problem can not be stated. Therefore, if the dual ends with a nonzero λ the solution of the Eν-SVC and ν-SVC are equal, up to a scaling factor, and if $\lambda = 0$ the solution of the Eν-SVC will have to be obtained by other means.

9.5.2 Kernelization in the primal

We need a different approach to solve the Eν-SVC using kernels and without explicitly computing λ. We will use a kernel PCA (KPCA) decomposition of the input data using the selected kernel. KPCA [15, 14] computes the principal components of the mapped data and it will give at most n (the number of training samples) principal components. Once we have performed KPCA, we will map the data onto these components, so we will have a finite dimensional representation of each training vector in the feature space. Consequently, we can use the KPCA representation of the input data to train a linear classifier using (9.26), which will become a nonlinear machine in the original representation of the data (input space). A similar procedure has been used for incorporating invariances in SVMs [5].

When we have solved the linear problem with the KPCA representation, we can obtain the values of α_i and λ using the KKT conditions and the obtained solution ($\mathbf{w}$, b, ρ and ξ_i), which we use to distinguish between solutions that can be obtained with the classic ν-SVM or SVM ($\lambda > 0$) and those solutions which are not feasible with them ($\lambda < 0$).

We believe that this learning algorithm can be used together with the new trend in the machine learning community in which the kernel matrix is learnt instead [10]. Once the learning matrix has been optimized for a given problem, if it is not full-rank, all the values of ν might not be feasible and the Eν-SVC will provide a wider range of possible solutions to be evaluated.

The theorem in [16] in which it is stated that ν is an upper bound in the fraction of bounded SVs and a lower bound in the fraction of SVs also holds for this learning algorithm.

We conclude this section by noting that a this approach is related to one used for boosting [12] (related to 1-norm SVMs), but in which ρ was constrained positive, although it is not a necessary constraint for solving the problem.

9.6 Experiments

We have shown in the previous section that the ν-SVM can be modified to be able to extend the range of ν value down to zero, by allowing ρ to take negative values. We first show a simple 2D example, in which we can picture the solution given by the Extended ν-SVM. Class $+1$ samples are drawn equally from two normal distributions of means [2 0] and [0 2] and Class -1 from a normal distribution of zero mean. Both classes are equally likely and their variance matrices are the identity. In Figure 9.3, we show the obtained solution using a linear classifier for different values of ν. Of the

ones shown, the solution in Figure 9.3a ($\nu = 0.69$) is the only one that can be achieved by the classic ν-SVM and SVM described in [14]. The solution depicted in Figure 9.3b ($\nu = 0.51$) actually presents a ρ which is still positive, but the associated value of λ is negative, so this solution cannot be obtained with the classic ν-SVM. In Figure 9.3c ($\nu = 0.33$), we show a solution for $\rho \simeq 0$ in which the solution is constructed by all and only incorrectly classified samples. Finally, in Figure 9.3d ($\nu = 0.01$), we show the hard negative margin solution in which all the slack variables are equal to zero.

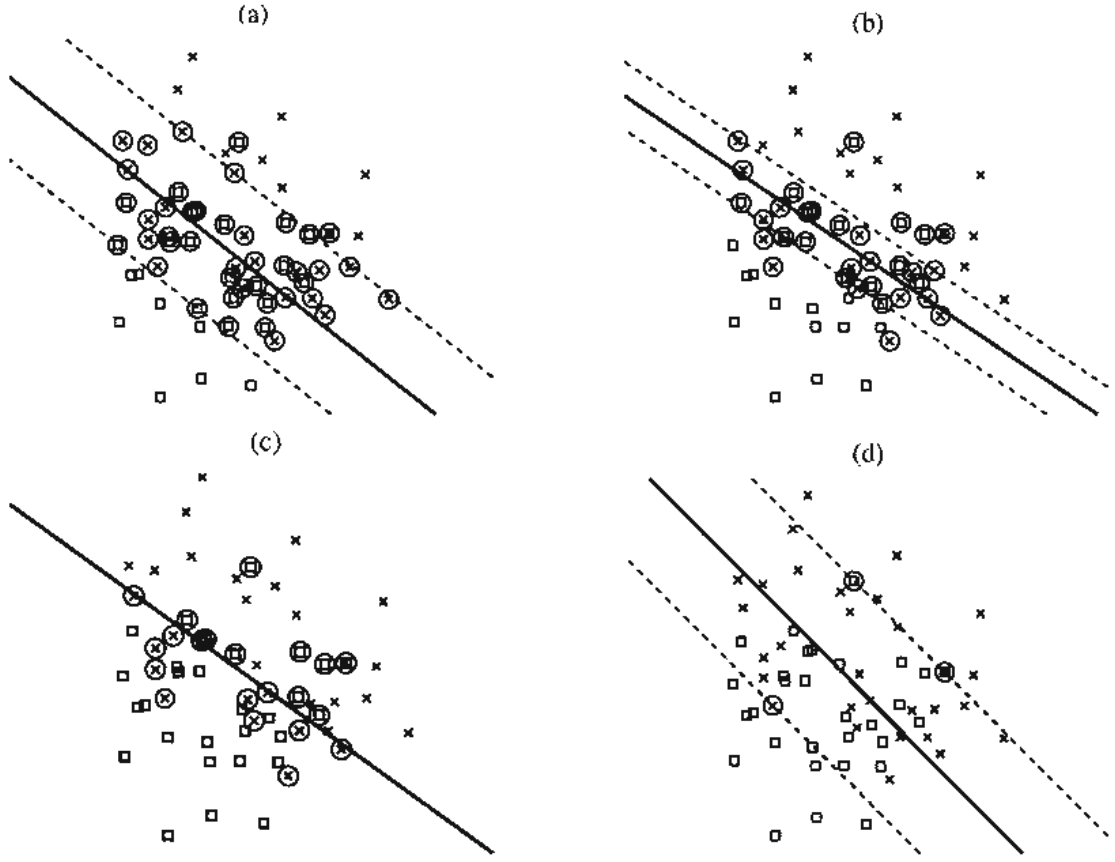


Figure 9.3: The solid lines represent the classification boundary and the dashed lines the $\pm\rho$ margins. Class $+1$ is shown by crosses, Class -1 by squares, and the SVs are ringed. In (a) we show the solution for $\nu = 0.69$ ($\rho = 1.59$ and $\lambda = 1.78$). In (b) we show the solution for $\nu = 0.51$ ($\rho = 0.69$ and $\lambda = -4.60$). In (c) we show the solution for $\nu = 0.33$ ($\rho = -0.04$ and $\lambda = -7.01$). And, in (d) we show the solution for $\nu = 0.01$ ($\rho = -1.79$ and $\lambda = -0.54$).

We have also solved this problem with two types of nonlinear kernels for $\nu = 0.51$, which is a value unreachable for the classic ν-SVM. In Figure 9.4a we show the obtained solution with an inhomogeneous polynomial kernel of second degree and in Figure 9.4b, an RBF kernel with standard deviation $\sigma = 8$. The probability of error of the Bayesian classifier is 0.200 and the achieved solution for the polynomial and RBF kernels are, respectively, 0.205 and 0.217, which are the best possible results for the whole range of ν.

We performed experiments on a real application using a dataset from the UCI

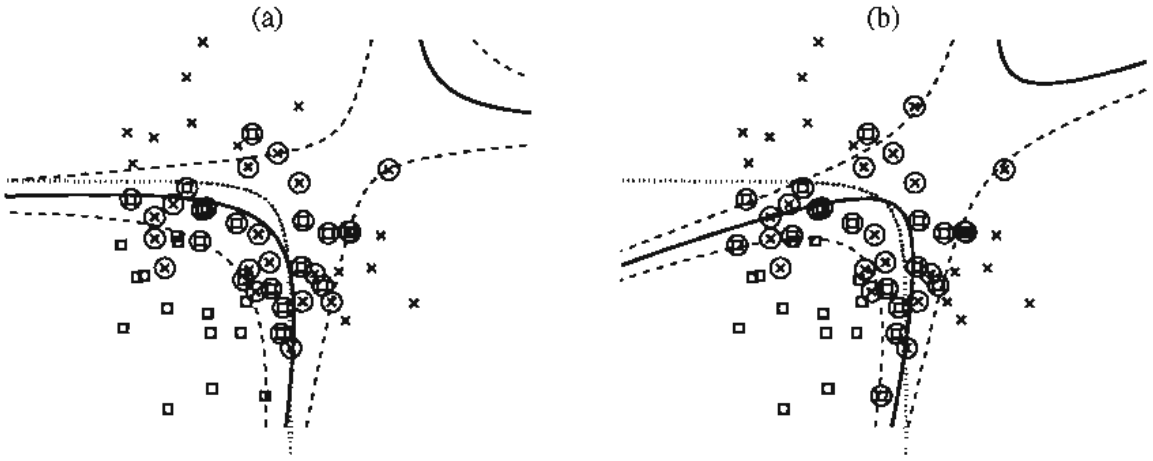


Figure 9.4: We show the results for polynomial and RBF kernels for $\nu = 0.51$, respectively in (a) and (b). The solid lines represent the classification boundary and the dashed lines the $\pm\rho$ margins. Class $+1$ is shown by crosses, Class -1 by squares, and the SVs are ringed. The dotted lines represent the Bayes classifier.

ν	0.01	0.16	0.26	0.31	0.36	0.41	0.56	0.71	0.76	0.81
ρ	-1.12	-0.27	-0.07	0.02	0.09	0.15	0.34	0.54	0.69	1.09
λ	-0.78	-13.3	-15.7	-15.8	-15.5	-14.6	-9.59	-0.63	3.59	9.46
fSVs	.025	.167	.265	.312	.361	.420	.569	.716	.767	.811

Table 9.3: The values of ρ, λ, and the fraction of support vectors (fSVs) mean values for the 5 split of the data.

Machine Learning Repository (http://www.ics.uci.edu/~mlearn). We have chosen the Liver-disorder database, because it is a noisy dataset with a large classification error, which significantly limits the minimum value of ν for the classic ν-SVM. We preprocessed the dataset so every feature has zero mean and unit standard deviation. Then, we solved the problem using a linear kernel and measured success using 5-fold cross-validation over different possible values from the whole range of ν.

In Figure 9.5, we have plotted the mean training and test error for the 5 splits of the data. In this problem as the classification error is so high the allowed values of ν in the classic ν-SVM formulation are restricted to the two highest ($\nu = 0.76$ and $\nu = 0.81$). Therefore the best solution for the classic ν-SVM is obtained for $\nu = 0.76$, achieving a probability of error of 0.325. The Extended ν-SVM is able to obtain the solution for any ν value so we can find that the best solution is found for $\nu = 0.41$ with a probability of error of 0.293, a reduction of more than three percentage points.

We have reported in Table 9.3 the values of ρ, λ and the fraction of SVs for the tested values of ν. In it, we can see that for $\nu \leq 0.71$ the values of λ are negative and therefore this solution cannot be achieved by the classic ν-SVM nor by the regular SVM (recall that the two types of SVMs can be shown to provide identical solutions for suitable parameter values [14]). It can be also pointed out that the value of ρ remains

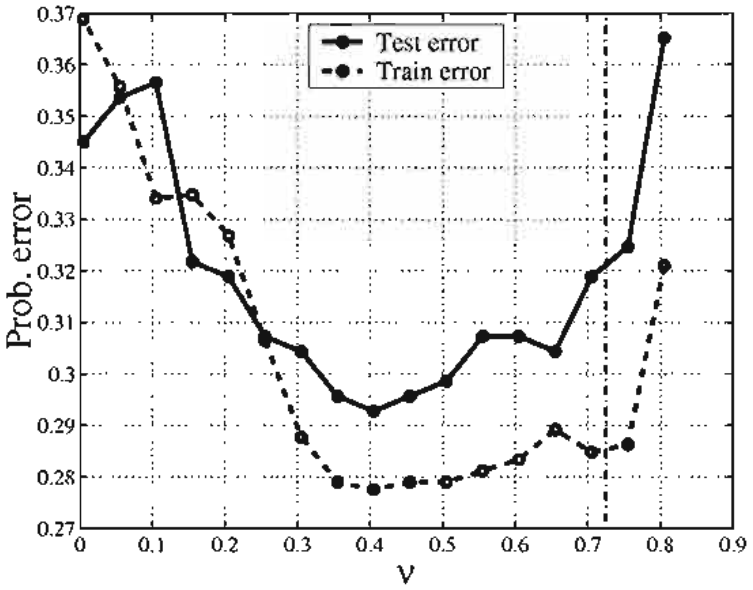


Figure 9.5: The training and test error are represented, respectively, by the dashed and solid lines for the liver disorder database. The dash-dotted line separates the results of the classic ν-SVM (to the right) and those that can be obtained only with the Extended ν-SVM.

positive for ν ranging from 0.71 to 0.31. In this case we will have a positive margin but we will want to minimize it, because the number of SVs correctly classified do not outnumber the incorrectly classified ones, but there are still training samples that are correctly classified and are SVs. For values of $\nu \leq 0.26$ the solution will be constructed exclusively using incorrectly classified samples and, consequently, the margin ρ will be negative.

9.7 Conclusions and Further Work

In this chapter, we have reinterpreted how the maximum margin can be constructed for separable training data sets, and so we were able to obtain an alternative SVM solution for non-separable data sets. Moreover, we have developed a formulation like the ν-SVM, Extended ν-SVM, in which the former is contained as a special case (for $\lambda > 0$). This extended ν-SVM allows us to construct the machine with any number of SVs and if the best possible solution lies in the zone in which $\lambda < 0$ we can obtain a solution that can be better than the best solution achieved by the classic SVM. And it is possible that the best solution can be obtained for a $\lambda < 0$ because it has been recently proven that the optimal ν is equal to twice the Bayes error [17].

There are various possible extensions of the work presented. The training procedure has to be improved so it is easier to solve when using kernels. And, there are also possible theoretical extensions, to prove bounds and convergence rates as well as to determine the role of the Lagrange multiplier λ.

Bibliography

[1] K. Bennett and E. Bredensteiner, Duality and geometry in SVM classifiers, In *Proceedings of the Seventeenth International Conference on Machine Learning*, San Francisco, CA, December (2000) 57–64.

[2] B. E. Boser, I. M. Guyon, and V. N. Vapnik, A training algorithm for optimal margin classifiers, In D. Haussler, editor, *5th Annual ACM Workshop on COLT*, Pittsburgh, PA, ACM Press (1992) 144–152.

[3] C. J. C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, *Knowledge Discovery and Data Mining* **2**(2) (1998) 121–167.

[4] C.-C. Chang and C.-J. Lin, Training ν-support vector classifiers: Theory and algorithms, *Neural Computation* **13**(9) (2001) 2119–2147.

[5] O. Chapelle and B. Schölkopf, Incorporating invariances in non-linear SVMs, In *Advances in Neural Information Processing Systems 14*, Cambridge, MA, M.I.T. Press (2001).

[6] C. Cortes and V. N. Vapnik, Support Vector Networks, *Machine Learning* **20** (1995) 273–297.

[7] D.J. Crisp and C.J.C. Burges, A geometric interpretation of nu-SVM classifiers, In *Advances in Neural Information Processing Systems 12*, Cambridge, MA, M.I.T. Press (1999).

[8] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machine*, Cambridge University Press (1999).

[9] R. Fletcher, *Practical Methods of Optimization*, Wiley, Chichester, 2^{nd} edition (1987).

[10] G.R.G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M.I. Jordan, Learning the kernel matrix with semi-definite programming. In *Proceedings of the nineteenth International Conference on Machine Learning*, Sidney, Australia (2002) 57–64.

[11] Y. Lin, Y. Lee, and G. Wahba, Support vector machines for classification in nonstandard situations, *Machine Learning* **46** (2002) 191–202.

[12] G. Rätsch, B. Schölkopf, A. J. Smola, S. Mika, T. Onoda, and K.-R. Müller, Robust ensemble learning, In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, M.I.T. Press, Cambridge MA (2000) 207–220.

[13] M. Schmidt and H. Gish, Speaker identification via support vector classifiers, In *Proceedings ICASSP'96*, Atlanta, GA (1996) 105–108.

[14] B. Schölkopf and A. Smola, *Learning with kernels*, M.I.T. Press (2001).

[15] B. Schölkopf, A. Smola, and K. R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, *Neural Computation* **10**(15) (1998) 1299–1319.

[16] B. Schölkopf, A. Smola, R. Williamson, and P. L. Bartlett, New support vector algorithms, *Neural Computation* **12**(5) (2000) 1207–1245.

[17] I. Steinwart, On the optimal choice for the nu-support vector machines, Technical report, Driedrich-Schiller-Universitat (2002).

[18] V. N. Vapnik, *Estimation of Dependences Based on Empirical Data*, Springer–Verlag (1982).

[19] V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer–Verlag, New York (1995).

[20] V. N. Vapnik, *Statistical Learning Theory*, Wiley, New York (1998).

Chapter 10

Kernels Methods for Text Processing

Nello Cristianini, Jaz Kandola, Alexei Vinokourov and John Shawe-Taylor[1]

Abstract. Kernel Methods are used to perform pattern analysis on datasets by embedding them into a vector space where relations can be more easily discovered. The choice of the kernel function determines the properties of such embedding, and ultimately decides the quality of the relations detected in the dataset. In the case of text processing, documents need to be embedded in a way that reflects their semantic content, so that documents on a similar topic will be mapped to nearby locations in the vector space. Designing a kernel that incorporates semantic information is therefore a crucial objective, that can be - at least partially - accomplished by learning it from data. Variations of the traditional "vector space" representation of text are presented here to address this problem, all aimed at learning the optimal embedding directly from the corpus, exploiting co-occurrence information, or cross-linguistic correlations. We also present kernels based on string matching, and kernels obtained by combining text and links information for hypertext documents.

[1]This work was supported in part by the EU under the KerMIT project, IST-2001-25431.

10.1 Introduction

Following the explosion of availability of digital text in the last decade, automatically processing natural language text documents has become one of the main focuses of AI and computer science in general. In a way, after the vectorial form, the most important data format in applications is perhaps natural language text, and its particular features mean that it deserves specific attention.

Although we are far from a full understanding of the semantic content of a text document, it is however possible to tackle simpler tasks like extracting some coarse-grained information about the topic of a document, and use it for tasks such as retrieval: after all deciding if two given documents are on a similar topic is a much easier task than actually understanding it (imagine you are looking at documents in a foreign language). Such notions of similarity, suitably refined, can be turned into kernels and hence can be used for a number of applications other than just retrieval: classification, clustering, ranking, correlation analysis.

Classic techniques from Information Retrieval, like the rich class of Vector Space Models, can be naturally reinterpreted as kernel methods, and in this new perspective they can be better understood and implemented, as well as greatly extended and improved. Based just on detecting and exploiting statistical patterns in the documents, kernel based methods are increasingly used in this domain, and furthermore they provide a natural common framework for some new as well as classical approaches. Furthermore, in the Vector Space representation, the primal-dual dialectics common in kernel methods has an interesting counterpart in terms of term-based vs. document-based representations. In this chapter we analyze mainly kernels based on the Vector Space family and its generalizations, but we also look at a kernel based on string matching. Throughout this chapter the aim is just to discuss how to embed a text document into a space in such a way as to capture as much as possible of its semantic content. The use that we want to make of such a kernel is not discussed in this chapter in detail, although some applications are mentioned. The reader can refer to the algorithms discussed in other parts of this collection, such as Support Vector Machines, or those referred to in the conclusions.

10.2 Overview of Kernel Methods

Kernels methods for pattern analysis work by embedding data into a vector space where patterns can be more easily detected. This is achieved in a computationally efficient way by implicitly computing the inner products between the images of the data items in this space, rather than their coordinates. Several pattern recognition algorithms exist that only require the knowledge of inner products between data, and it is often the case that the inner product between feature vectors is much easier to compute than their explicit representation in coordinates. The function that returns the inner product between images of two data items in some embedding space is called the Kernel function.

Of course the quality of the pattern analysis will depend on the quality of the embedding provided by the chosen kernel. Ideally, we would like the embedding to be such

that documents that are semantically related end up in nearby locations of the feature space. The issue is how to build a map from documents to vectors that incorporates some knowledge of semantics, or equivalently a kernel between two documents. Once such a kernel is available, it can be used in combination with classification, clustering, ranking, retrieval and other algorithms.

Formally, we will call any function that calculates the inner product between mapped examples in a feature space "a kernel function". For any mapping $\phi : D \rightarrow F$, from the input space D to the feature space F, we will denote the kernel by $K(d_i, d_j) = \langle \phi(d_i), \phi(d_j) \rangle$. The mapping ϕ transforms an example $d \in D$ (e.g., a document) into an N dimensional feature vector,

$$\phi(d) = (\phi_1(d), \ldots, \phi_N(d)).$$

The explicit extraction of features in a feature space generally has very high computational cost but a kernel function provides a way to handle this problem. A kernel function is a symmetric positive definite function, that is the $n \times n$ matrix with entries of the form $K_{ij} = K(d_i, d_j)$ (known as the kernel matrix) is always a symmetric, positive definite matrix. It is interesting to note that this matrix is the main source of information for KMs and these methods use only this information to learn a classifier.

There are ways of combining simple kernels to obtain more complex ones, possibly at each step refining the quality of the embedding [9]. For example given a kernel K and a set of n vectors the polynomial construction is given by

$$K_{poly}(d_i, d_j) = (K(d_i, d_j) + c)^p$$

where p is a positive integer and c is a non-negative constant. Clearly, we incur a small computational cost, to define a new feature space. The feature space corresponding to a degree p polynomial kernel includes all products of at most p input features. Hence polynomial kernels create images of the examples in feature spaces having a very large numbers of dimensions. Furthermore, Gaussian kernels define feature space with an infinite number of dimensions though the function

$$K_{gauss}(d_i, d_j) = \exp\left(\frac{-\|d_i - d_j\|^2}{2\sigma^2}\right)$$

In this sense, a Gaussian kernel can allow an algorithm to learn a linear classifier in an infinite dimensional feature space. In this chapter we will examine different ways to construct kernels that capture something of the semantic content of a document. Some of them will be obtained by successive embeddings, while others will be learned from data and others yet will be obtained by comparing common substrings in the documents.

10.3 From Bag of Words to Semantic Space

Although the topic of analyzing a document from the point of view of its meaning belongs to the field of Natural Language Processing, it has been the distinct Information Retrieval community that developed the simple representation that is today mostly

used to compare the topics between documents, and that forms the basis of this chapter. This is called the Vector Space model, or the Bag of Words approach [26, 25].

By regarding those similarity measures as kernels [16], it is possible to extend their applicability from mere retrieval tasks to the full range of possibilities of the kernel approach: correlation analysis, novelty detection, classification (known as categorization in the text community), ranking, etc.

The simplest possible approximated representation of a document is as *a bag of words* (a bag is a set where repeated elements are allowed, so that not only the presence of an element but also its frequency is considered). In this way all the positional information (given by grammar and syntax) is discarded, and simple statistics about the frequency of words are considered. This approach is very popular in information retrieval and leads to a very natural representation of documents as vectors in a 'terms space' (a space in which each dimension is associated to a term of the dictionary). In order to improve such a method, it is sometimes more natural to replace words by 'concepts', mapping documents into a lower-dimensional 'concept space', as discussed in the later sections. Of course proximity information between words in the text should count, as well as grammatical relations, but at the moment it is not clear how to exploit them, and this approach chooses to discard them altogether.

Another important issue besides the relative positions of words in the document, is the one of the semantic relations between words: synonymy, homonymy, etc should be considered when comparing two documents. Such information can introduce some degree of semantics into the embedding, and will be discussed later in this chapter. It can be either based on some existing knowledge-base like a thesaurus or a semantic net, or it can be inferred directly from a corpus of documents, e.g. by means of statistical considerations.

Other operations can improve the quality of the embedding, for example assigning a weight to each term based on its importance (for example removing uninformative terms known as 'stop words' in IR literature, such as: "and", "of", "with"); or declaring two words to be equivalent. These operations can be performed in sequence, creating a series of successive embeddings, each of which adds some level of semantic to the system, and that can at the end be regarded all together as a single map. Successive embeddings are a powerful technique in applications where domain knowledge should be introduced into the kernel design.

An interesting feature of the Vector Space approach will be to provide an interpretation of the duality between representations of kernel algorithms. Here the dual description corresponds to a document-by-document view of the problem, while the primal description to a term by term view of it. In the same way a document can be seen as the count of the terms that appear in it, a term can be regarded as the count of the documents in which it appears. A term-by-document matrix (and its transpose document-by-term) can represent this.

Since it is often the case that the number of documents in a corpus is less than the number of different terms used in them, for computational reasons it is often useful to work in a document-based (or dual) perspective, whereas for intuition and interpretation it is usually better to derive algorithms in the term-based (primal) representation. We can sometimes exploit the full power of duality in kernel methods, by stating the problem in the intuitive term representation, and then dualizing it to the document-

based perspective. Many operations for extracting semantic information about terms from a corpus will follow this line.

10.4 Vector Space Representations

Given a document, we map it to a bag of terms (or bag of words) by simply considering all the terms contained in it and counting the number of occurrences of each term. A bag of words has its natural representation as a vector in the following way: the number of dimensions is the same as the number of different terms in the corpus (the set of all different terms is sometimes referred to as the lexicon, or dictionary), each entry of the vector is assigned to a specific term, and we simply count occurrences in a document. Further normalization and term weighting operations can be viewed as further embeddings. In this way a document is represented by a (column) vector d in which each entry records how many times a particular term is used in the document. Typically d can have tens of thousands of entries, often comparable with or even larger than the number of training examples. Furthermore, for a particular document the representation is typically extremely sparse, having only a few non-zero entries. This preliminary embedding can then be refined by successive operations that we will examine later in this chapter. Before we continue, some definitions and notation are necessary:

Definition 1 *A document is represented, in the vector-space model, by a vertical vector d indexed by all the elements of the dictionary, and a corpus by a matrix D, whose columns are indexed by the documents and whose rows are indexed by the terms, $D = [d_1, ..., d_m]$. We also call the data matrix D the "term by document" matrix.*

Definition 2 *We define the "document by document" matrix as $G = D'D$ and the "term by term" matrix $T = DD'$. K is a kernel matrix, which can in general be different from G: further transformations of the document vectors can lead to different representations.*

In general, we will consider transformations of the document vectors $\phi(d)$. The simplest case involves linear transformations of the type $\phi(d) = Pd$, where P is any matrix, so that the kernels have the form

$$K(d_1, d_2) = d_1' P' P d_2$$

It should be noted that, for two vectors x_1 and x_2, and for any linear mapping denoted by the matrix P, the function $K_P(x_1, x_2) = x_1' P' P x_2$ is always a kernel, the Gram matrix (the matrix of inner products) being given by $X'P'PX$ for $X = (x_1, ..., x_m)$ which is by definition symmetric and positive definite. Different choices of the matrix P lead to different variants of the vector space models. We will examine some variants in the next subsections. For this particular framework, the work of [15] is appropriate.

A simple kernel between documents can be obtained by the dot product between their respective (possibly normalized) vectors, that is their cosine, a quantity that is easy to regard as a kernel:

$$K(d_1, d_2) = \langle d_1, d_2 \rangle = d_1' d_2$$

It is clear that there are several problems with this first representation of documents. For example, document length will affect the vector's norm, and this should be unrelated to the topic hence one could then define a kernel that discards this information as follows:

$$K(d_1, d_2) = \left\langle \frac{d_1}{||d_1||}, \frac{d_2}{||d_2||} \right\rangle$$

Also this representation has some clear problems: for example not all words should have the same importance in establishing the topic of a document. The entropy or the frequency of a word across the documents can be used to quantify the amount of information carried by a word (e.g. the word 'the' will be equally distributed in all documents). This is an 'absolute' measure of the information content of a word, but one could also define a relative measure of the importance of a word, with respect to the given task. A measure would be the mutual information, but more commonly one uses a class of term weighting measures known as *tf-idf*. One can consider a weight vector w associated with the dictionary, and this can be used to define a different kernel as follows:

$$K(d_1, d_2) = d_1' W' W d_2$$

where $W = diag(w)$ is a diagonal matrix with w on its diagonal. This representation is advantageous since it takes care of *downweighting* irrelevant terms, but nonetheless it is still not capable of recognizing when two terms are semantically related, and hence of establishing a connection between two documents that share no terms but that are about the same topic, because they use synonyms. The problem is to introduce a notion of 'semantic similarity' between terms. One way is to define a 'proximity' matrix P such that entry $P_{ij} > 0$ iff term i is related (eg a synonym) to term j. If such a matrix is available, then one can define the following kernel:

$$K(d_1, d_2) = d_1' P' P d_2$$

and it is easy to see that the document represented by the (sparse) vector d is mapped to a (less sparse) vector $\phi(d)$ that has non-zero entries corresponding to terms semantically similar to the ones present in d. In information retrieval techniques of this type are sometimes known as 'query expansion' [2].

Remark 1 *Although conceptually falling within the semantic similarity issue, in practice the relation between different inflections of the same word is obtained by stemming. Word stemming reduces words to some root form. For example, the words "retrieve", "retrieval" and "retrieving" would be reduced to the word stem "retriev". This can be considered to be a form of dimensionality reduction.*

Remark 2 *The use of a matrix P can in theory give rise to stemming, whilst a W matrix encodes stop words removal. It is also necessary to add semantic relations perhaps in the same way. An important research issue is how to obtain these matrices W and P.*

In this chapter we will discuss methods for obtaining a semantic proximity matrix P directly from a corpus. We will try to perform all these operations as efficiently as possible, ideally in dual form, without writing down a matrix P explicitly but through the use of a kernel matrix.

A general way to define term weights is as follows: a component reflects the importance of the term in the document (term presence), while the other the discriminative power of the term within the corpus (term importance). The first part is usually a function of the term frequency tf within the given document, while the other is a function of the inverse document frequency idf: $\frac{m}{d(\text{term})}$, that is the total number of documents in the corpus divided by the number of documents that contain the given term. So if for example a word appears exactly 20 times in each document, this would not be regarded as a very informative one. Its distance from the uniform distribution is a good estimation of its importance, but better methods can be obtained by studying the typical term distributions within documents and corpora. The simplest case is just given by: $df * \log(idf)$. Other measures can be obtained from information theoretic quantities, or from empirical models of term frequency [2]. Since these measures do not use label information, they could also be estimated from an external, larger unlabeled corpus, that provides the background knowledge to the system.

The issue of learning P is more complicated, and the rest of the chapter is devoted to it. The different choices of matrix P characterize the different vector space models proposed so far.

Remark 3 *The mapping discussed above, obtained by multiplication with the matrix P, is a particularly simple and useful case. However it is also possible to consider nonlinear mappings by means of kernels. For polynomial kernels, it will be sufficient to use $K_{\text{poly}}(d_1, d_2) = (K(d_1, d_2) + 1)^p$ and similarly for a Gaussian kernel.*

10.4.1 Basic vector space model

The conventional *VSM* introduced by Salton (see [26]), and first used as a kernel by [16], makes direct use of the vector representation, with no further mapping. In other words the matrix $P = I$ (where I denotes the identity matrix), hence only the term weighting matrix W is applied. The performance of retrieval systems based on such a simple representation is surprisingly good. Since the representation of each document as a vector is very sparse, special techniques can be deployed to facilitate the storage and the computation of dot products between such vectors. Joachims [16] and Dumais et al. [11], also used polynomial and Gaussian kernels on this representation, so further mapping the data into a richer space, for classification tasks with Support Vector Machines. In particular, the use of polynomial kernels can be seen as checking for n-tuples of terms. One of the problems with this representation is that it treats terms as uncorrelated, assigning them to orthogonal directions of the feature space. This means that it clusters together in the feature space documents that share many terms. But in reality words are correlated, sometimes synonymous. This then raises the issue of how to incorporate semantic information, so as to map to nearby locations documents that share related terms. This can be achieved efficiently through the use of a semantic proximity matrix. One idea would be to perform a kind of document

expansion, adding to the expanded version all synonymous (or just related) words to the existing terms. In effect this can be viewed either as mapping documents to sets of document or to larger documents. An equivalent approach would be to replace terms by concepts. There are many ways to address this problem. One can either exploit external knowledge about correlations, for example from a semantic network, or use statistical information about term-term correlations derived from the corpus itself, or from an external reference corpus. The approaches discussed in the following subsections can all be regarded as kernel methods, and aim at performing this operation in a somewhat automatic way.

10.4.2 Generalised vector space model

An early attempt to overcome the limitations of VSMs was proposed by [33] under the name of Generalised VSM (GVSM). This method aims at capturing some term-term correlations by looking at co-occurrence information: two terms are considered semantically related if they co-occur often in the same documents. This has the effect that two documents can be seen as similar even if they do not share any terms. A simple technique can provide one such metric, and it is easy to see that it also constitutes a kernel function.

If the data matrix is D, (term-document) then

$$K(d_1, d_2) = (D'd_1)(D'd_2) = d_1' DD' d_2$$

the matrix DD' has a nonzero entry in the cell ij if and only if there is a document in the corpus where the i and the j th terms co-occur. So two terms co-occurring in a document are considered related. The new metric accounts for this co-occurrence information. This method is faster but not as effective as Latent Semantic Indexing (LSI) described below. In the common case when there are less documents than terms, this has the effect of dimensionality reduction: the terms are all mapped to a lower dimensional space, via a kind of bottle-neck mapping, and compared in the resulting space or alternatively they can be mapped back to the original space in an augmented form. A further dimensionality reduction in GVSM can be achieved with the simple technique of sparsification: before computing the inner product, all but the k largest entries are set to zero.

10.4.3 Semantic smoothing for vector space models

Another way to incorporate semantic information is by directly using an external source, like a semantic network. In this section we briefly describe one such approach. In the next section we will see how a more refined co-occurrence analysis than the simple GVSM, also bears many relations with semantic networks [28].

In order to incorporate some semantic information into VSMs, Solias and D'Alche-Buc used a semantic network (Word-net) [28] as a way to obtain term-similarity information. Such a network encodes for each word of a dictionary its relation with the other words in a hierarchical fashion (e.g.: synonym; hypernym; etc). For example both the word 'husband' and 'wife' are special cases of their hypernym 'spouse'. In this way, the distance between two terms in the hierarchical tree provided by Wordnet

gives an estimation of their semantic proximity, and can be used to modify the metric of the vector space where the documents are mapped by the bag-of-words approach.

In order to insert such knowledge into the kernel, they have handcrafted the matrix P in such a way that the ij-th entry expresses the semantic proximity between the terms i and j. Such a quantity is fixed to be equal to the inverse of their topological distance in the graph, that is of the number of edges of the shortest path connecting them. The vector space metric is then dictated by the following kernel:

$$K(d_1, d_2) = d_1' P' P d_2$$

or by the following distance

$$\begin{aligned} \|Pd_1 - Pd_2\|^2 &= \|P(d_1 - d_2)\|^2 \\ &= (d_1 - d_2)' P' P (d_1 - d_2) \\ &= (d_1 - d_2) S (d_1 - d_2) \end{aligned}$$

Obviously on top of this first linear mapping one can then perform another mapping, for example with polynomial kernels or – as is done by [28] with Gaussian kernels. Notice that GVSMs can be regarded as a special case of this, when the proximity is given by the data matrix itself: two terms are 'declared' semantically similar if they co-occur in the same documents.

10.4.4 Latent semantic kernels

As mentioned in the previous section we now consider how co-occurrence analysis can be used to generate a semantic proximity matrix. This very effective vector space representation of documents, also proposed in order to capture semantic information, is known as Latent Semantic Indexing (LSI) [10]. So, conceptually, LSI follows similar ideas as GVSMs. But the technique used to extract such information is very different and makes use of Singular Value Decomposition (SVD). We will see that this amounts to a very special choice of the matrix P, and that such matrix has some useful properties [7].

We can obtain a suitable matrix P from the data as follows: given a term-document matrix D, by using SVD we factor it into 3 parts:

$$D = U \Sigma V'$$

where Σ is a diagonal matrix, and U and V are unitary (i.e. $U'U = I$). The new kernel becomes:

$$K(d_1, d_2) = (I_k U' d_1)'(I_k U' d_2) = d_1' U I_k U' d_2 = d_1' U_k U_k' d_2 = d_1' P' P d_2$$

where I_k is the identity with only the first k diagonal elements nonzero and U_k is the matrix U with all but the first k columns set to zero. In this way we achieve dimension reduction, by projecting the data onto a subspace spanned by the principal axes (another bottleneck mapping, like GVSM, into a k dimensional space). The motivation for this particular mapping is that highly correlated dimensions, i.e. terms

that co-occur very often in the same documents of the corpus, are merged into a single dimension of the new space. This creates a new similarity metric based on context information. In terms of proximity matrix, as defined in the case of the semantic network, described in the previous section, we could instead define P as $U'U_k$.

What is more interesting for kernel methods is that the same mapping, instead of acting on term-term matrices, can be obtained implicitly by working with smaller document-document matrices. The original term by document matrix D gives rise to the dot product matrix

$$K = D'D$$

since the feature vector for document j is the j-th column of D. The SVD decomposition is related to the eigenvalue decomposition of K,

$$K = D^T D = V\Sigma U^T U\Sigma V^T = V\Sigma^2 V^T = V\Lambda V^T$$

so that the i-th column of V is the eigenvector of K, with corresponding eigenvalue $\Lambda_{ii} = \lambda_i = \sigma_i^2$. The feature space created by choosing the first k singular values in the LSI approach corresponds to mapping a feature vector x to the vector $U_k U_k^T x$, where U_k is obtained by taking the first k columns of U. This corresponds to an orthogonal projection onto the subspace spanned by the first k columns of U. Hence, the feature vectors for the training set are the columns of the matrix $U_k U_k^T D$ and so the new kernel matrix is given by

$$\begin{aligned}\widehat{K} &= D^T U_k U_k^T U_k U_k^T D \\ &= V\Sigma U^T U_k U_k^T U\Sigma V^T \\ &= V\Sigma\widehat{I}\Sigma V^T = V\widehat{\Lambda}V^T\end{aligned}$$

where $\widehat{I}$ and $\widehat{\Lambda}$ are respectively the identity matrix I and the matrix Λ with diagonal entries beyond the k-th set to zero.

This new kernel matrix is obtained directly from K by applying an eigenvalue decomposition of K and re-multiplying the component matrices having set all but the first k eigenvalues to zero. Hence, we can obtain the kernel corresponding to the LSI feature space without actually ever computing the features.

If we consider the possible outputs of a linear classifier with bounded weight vectors w on the training set, before the projection we had the set

$$\begin{aligned}\left\{w^T D : \|w\| \leq 1\right\} &= \left\{w^T U\Sigma V^T : \|w\| \leq 1\right\} \\ &= \left\{u^T\Sigma V^T : \|u\| \leq 1\right\}\end{aligned}$$

This set is an ellipsoid with principle axes given by the columns of V and dimension given by the corresponding singular values. After the LSI projection the corresponding set is

$$\begin{aligned}\left\{w^T U_k U_k^T D : \|w\| \leq 1\right\} &= \left\{w^T U_k U_k^T U\Sigma V^T : \|w\| \leq 1\right\} \\ &= \left\{w^T U_k\Sigma_k V^T : \|w\| \leq 1\right\} \\ &= \left\{u^T\widehat{\Sigma}V^T : \|u\| \leq 1\right\}\end{aligned} \tag{10.1}$$

where $\widehat{\Sigma}$ and Σ_k are the matrix Σ with diagonal entries beyond the k-th set to zero and the matrix composed of its first k rows. Hence, the effect of the LSI projection is to flatten the ellipse into the space spanned by its first k principal axes.

The method therefore produces a new kernel, which can be used in any kernel based method provided we are able to evaluate the solution on a new input x. Hence, we must evaluate functions of the form

$$\begin{aligned} f(x) &= \sum_{i=1}^{m} \alpha_i y_i \left(x^T U_k U_k^T U_k U_k^T D\right)_i \\ &= \sum_{i=1}^{m} \alpha_i y_i \left(x^T U_k U_k^T U \Sigma V^T\right)_i \\ &= \sum_{i=1}^{m} \alpha_i y_i \left(x^T U_k \Sigma_k V^T\right)_i \end{aligned}$$

Note that we would like to avoid working with the feature vector x and only make use of the original dot products which we are able to compute

$$t^T := x^T D = x^T U \Sigma V^T$$

We must therefore express $x^T U_k \Sigma_k$ in terms of t. But we have

$$\begin{aligned} t^T V \widehat{I} &= x^T U \Sigma \widehat{I} \\ &= x^T U_k \Sigma_k \end{aligned} \tag{10.2}$$

and hence we can evaluate $f(x)$ as follows

$$\begin{aligned} f(x) &= \sum_{i=1}^{m} \alpha_i y_i \left(x^T U_k \Sigma_k V^T\right)_i \\ &= \sum_{i=1}^{m} \alpha_i y_i \left(t^T V \widehat{I} V^T\right)_i \end{aligned}$$

10.4.5 Semantic diffusion kernels

As we have already highlighted, the standard representation of text documents as bags of words suffers from well known limitations, mostly due to its inability to exploit semantic similarity between terms: documents sharing terms that are different but semantically related will be considered as unrelated. In this section we show how semantic similarity from a corpus can be inferred in two different ways. The first one defines word-similarity based on document-similarity and vice versa, giving rise to a system of equations whose equilibrium point we use to obtain a semantic similarity measure. The second method models semantic relations by means of a diffusion process on a graph defined by lexicon and co-occurrence information. Both approaches produce valid kernel functions parametrised by a real number. The alignment measure introduced to the kernel community in [8] can be used to successfully perform model selection over this parameter [20] [19].

Kernel based methods can be considered to be attractive choices for inferring relations from textual data since they enable us to work in a document-by-document setting rather than in a term-by-term one [16]. In the vector space model, a document is represented by a vector indexed by the terms of the corpus. Hence, the vector will typically be sparse with non-zero entries for those terms occurring in the document. Two documents that use semantically related but distinct words will therefore show no similarity. As in Latent Semantic Kernels (see section 10.4.4) the aim of a semantic proximity matrix is to correct for this by indicating the strength of the relationship between terms that even though distinct are semantically related.

The semantic proximity matrix P is indexed by pairs of terms a and b, with the entry $P_{ab} = P_{ba}$ giving the strength of their semantic similarity. If the vectors corresponding to two documents are d_i, d_j, their inner product is now evaluated through the kernel

$$k(d_i, d_j) = d_i' P d_j,$$

where $\mathbf{x}'$ denotes the transpose of the vector or matrix $\mathbf{x}$. Note that there has been a slight change of notation in that P here is what had previously been denoted $P'P$. The symmetry of P ensures that the kernel is symmetric. We must also require that P is positive semi-definite in order to satisfy Mercer's conditions. In this case we can decompose $P = R'R$ for some matrix R, so that we can view the semantic similarity as a projection into a semantic space

$$\phi : d \longmapsto Rd, \quad \text{since} \quad k(d_i, d_j) = d_i' P d_j = \langle Rd_i, Rd_j \rangle.$$

Two methods for inferring (or refining) the similarity measure between examples that are based on two different observations can be analyzed. The first method exploits the fact that the standard representation of text documents as bags of words gives rise to an interesting duality: while documents can be seen as bags of words, simultaneously terms can be viewed as bags of documents – the documents that contain them. In such a model, two documents that have highly correlated term-vectors are considered as having similar content. Similarly, two terms that have a correlated document-vector will have a semantic relation. This is of course only a first order approximation since the knock-on effect of the two similarities on each other needs to be considered. It is possible to define term-similarity based on document-similarity, and vice versa, to obtain a system of equations that can be solved in order to obtain a semantic proximity matrix P.

The second method exploits the representation of a lexicon (the set of all words in a given corpus) as a graph, where the nodes are indexed by words and where co-occurrence is used to establish links between nodes. Such a representation has been studied recently giving rise to a number of topological properties [12]. We consider the idea that higher order correlations between terms can affect their semantic relations as a diffusion process on such a graph. Although there can be exponentially many paths connecting two given nodes in the graph, the use of diffusion kernels [22] enables us to obtain the level of semantic relation between any two nodes efficiently, so inferring the semantic proximity matrix from data.

Equilibrium equations for semantic similarity

We consider the first approach outlined above. The aim is to create recursive equations for the relations between documents and between terms. Let X be the feature example (term/document in the case of text data) matrix in a possibly kernel-defined feature space, so that $X'X$ gives the kernel matrix K and XX' gives the correlations between different features over the training set. We will denote this latter matrix with G. Consider the similarity matrices defined recursively by

$$\widehat{K} = \lambda X'\widehat{G}X + K \quad \text{and} \quad \widehat{G} = \lambda X'\widehat{K}X + G \tag{10.3}$$

We can interpret this as augmenting the similarity given by K through indirect similarities measured by G and vice versa. The factor $\lambda < \|K\|^{-1}$ ensures that the longer range effects decay exponentially. The following result characterizes the solution of the above recurrences.

Proposition 1 *Provided $\lambda < \|K\|^{-1} = \|G\|^{-1}$, the kernels $\widehat{K}$ and $\widehat{G}$ that solve the recurrences (10.3) are given by*

$$\widehat{K} = K(I - \lambda K)^{-1} \text{ and } \widehat{G} = G(I - \lambda G)^{-1}$$

In view of the form of the solution the following definition can be introduced:

Definition 3 [von Neumann Kernel] *Given a kernel K the derived kernel $\widehat{K}(\lambda) = K(I - \lambda K)^{-1}$ will be referred to as the von Neumann kernel.*

Note that we can view $\widehat{K}(\lambda)$ as a kernel based on the semantic proximity matrix $P = \lambda\widehat{G} + I$ since

$$X'PX = X'(\lambda\widehat{G} + I)X = \lambda X'\widehat{G}X + K = \widehat{K}(\lambda).$$

Hence, the solution $\widehat{G}$ defines a refined similarity between terms/features. In the next section, we will consider the second method of introducing semantic similarity derived from viewing the terms and documents as vertices of a weighted graph.

Semantic similarity as a diffusion process

Graph like structures within data occur frequently in many diverse settings. In the case of language, the topological structure of a lexicon graph has recently been analyzed [12]. Such a graph has nodes indexed by all the terms in the corpus, and the edges are given by the co-occurrence between terms in documents of the corpus. Although terms that are connected are likely to have related meaning, terms with a higher degree of separation would not be considered as being related.

A diffusion process on the graph can also be considered as a model of semantic relations existing between indirectly connected terms. Although the number of possible paths between any two given nodes can grow exponentially, results from spectral graph theory have been recently used by [22] to show that it is possible to compute the similarity between any two given nodes efficiently *without* examining all possible paths. It is also possible to show that the similarity measure obtained in this way is a valid

kernel function. The exponentiation operation used in the definition, naturally yields the Mercer conditions required for valid kernel functions.

An alternative insight into semantic similarity is afforded if we multiply out the expression for $\widehat{K}(\lambda)$, $\widehat{K}(\lambda) = K(I - \lambda K)^{-1} = \sum_{t=1}^{\infty} \lambda^{t-1} K^t$. The entries in the matrix K^t are then given by

$$K_{ij}^t = \sum_{\substack{u \in \{1,\ldots,m\}^t \\ u_1 = i, u_t = j}} \prod_{\ell=1}^{t-1} K_{u_\ell u_{\ell+1}}.$$

The entries can be viewed as the sum of the products of the weights over all paths of length t that start at vertex i and finish at vertex j in the weighted graph defined on the examples. If we view these "connection strengths" as channel capacities, the entry K_{ij}^t in the matrix can be seen to measure the sum over all routes of the products of the capacities. If the entries satisfy that they are all positive and for each vertex the sum of the connections is 1, we can view the entry as the probability that a random walk beginning at vertex i is at vertex j after t steps. It is for these reasons that the kernels defined using these combinations of powers of the kernel matrix have been termed diffusion kernels [22]. A similar equation holds for G^t. Hence, examples that both lie in a cluster of similar examples become more strongly related, and similar features that occur in a cluster of related features are drawn together in the semantic proximity matrix P.

The kernel $\widehat{K}$ combines these indirect link kernels with an exponentially decaying weight. This suggests an alternative weighting scheme that shows faster decay for increasing path length,

$$\widetilde{K}(\lambda) = K \sum_{t=1}^{\infty} \frac{\lambda^t K^t}{t!} = K \exp(\lambda K)$$

The next proposition gives the semantic proximity matrix corresponding to $\widetilde{K}(\lambda)$.

Proposition 2 *Let $\widetilde{K}(\lambda) = K \exp(\lambda K)$. Then $\widetilde{K}(\lambda)$ corresponds to a semantic proximity matrix $\exp(\lambda G)$.*

PROOF. Let $X = U\Sigma V'$ be the singular value decomposition of X, so that $K = V\Lambda V'$ is the eigenvalue decomposition of K, where $\Lambda = \Sigma'\Sigma$. We can write $\widetilde{K}$ as

$$\begin{aligned} \widetilde{K} &= V\Lambda \exp(\lambda\Lambda)V' = X'U\Sigma^{-1}\Lambda \exp(\lambda\Lambda)\Sigma^{-1}U'X \\ &= X'U \exp(\lambda\Lambda)U'X = X' \exp(\lambda G)X, \end{aligned}$$

as required. □

The above leads to the definition of the second kernel that we consider.

Definition 4 *Given a kernel K the derived kernels $\widehat{K}(\lambda) = K \exp(\lambda K)$ will be referred to as the exponential kernels.*

The two new kernel adaptations described above, in both cases are parameterized by a positive real parameter λ. In order to apply these kernels on real text data, a method needs to be developed for choosing the parameter λ. Of course one possibility would be just to use cross-validation as considered by [22]. Rather than adopt this rather expensive methodology an alternative quantitative measure of agreement between the diffusion kernels and the learning task known as *alignment*, which measures the degree of agreement between a kernel and target [8] can be used.

Definition 5 [Alignment] *The (empirical) alignment of a kernel k_1 with a kernel k_2 with respect to the sample S is the quantity*

$$A(S, k_1, k_2) = \frac{\langle K_1, K_2 \rangle_F}{\sqrt{\langle K_1, K_1 \rangle_F \langle K_2, K_2 \rangle_F}},$$

where K_i is the kernel matrix for the sample S using kernel k_i.

where we use the following definition of inner products between Gram matrices

$$\langle K_1, K_2 \rangle_F = \sum_{i,j=1}^{m} K_1(x_i, x_j) K_2(x_i, x_j) \tag{10.4}$$

corresponding to the Frobenius inner product. From a text categorization perspective this can also be viewed as the cosine of the angle between two bi-dimensional vectors K_1 and K_2, representing the Gram matrices. If we consider $K_2 = yy'$, where y is the vector of outputs (+1/-1) for the sample, then

$$A(S, K, yy') = \frac{\langle K, yy' \rangle_F}{\sqrt{\langle K, K \rangle_F \langle yy', yy' \rangle_F}} = \frac{y'Ky}{m\|K\|_F} \tag{10.5}$$

However it should be observed that the parameterization is non-linear in λ so that we cannot solve for the optimal value. Instead an optimal value is sought using a line search over the range of possible values of λ for the value at which the derivative of the alignment with respect to λ is zero. For a full description of this and experimental results, see [20].

10.5 Learning Semantics from Cross Language Correlations

In this section we describe how we can learn a semantic space by identifying directions in the feature spaces for two different languages for which there is maximal correlation. This in turn implies a semantic similarity measure. In cross-language information retrieval, vector space models are used with partially aligned corpora, that is corpora formed by pairs of documents that are the translations of each other. This enables the system to learn term-term associations, that can then be used for retrieving relevant documents in a language as a response to a query in another language. Using a paired corpus (a set of pairs of documents, each pair being formed by two versions of the same text in two different languages), after merging each pair into a single 'document', we can

interpret frequent co-occurrence of two terms in the same document as an indication of cross-linguistic correlation [23]. In this framework, a common vector-space, including words from both languages, is created and then the training set is analysed in this space using SVD. This method, termed CL-LSI, has been discussed in [23]. More generally, many other statistical and linear algebra methods have been used to obtain an improved semantic representation of text data over LSI [29]. The problem of learning a semantic representation of text from a paired bilingual corpus is considered both for mono-lingual and cross-lingual applications. This problem can be regarded either as an unsupervised problem with paired documents, or as a supervised monolingual problem with very complex labels (i.e. the label of an English document could be its French counterpart). In either way, the data can be readily obtained without an explicit labeling effort, and furthermore there is not the loss of information due to compressing the meaning of a document into a discrete label. Kernel Canonical Correlation Analysis (KCCA) [1] can be used to learn a representation of text that captures aspects of its meaning. Given a paired bilingual corpus, this method defines two embedding spaces for the documents of the corpus, one for each language, and an obvious one-to-one correspondence between points in the two spaces. KCCA then finds projections in the two embedding spaces for which the resulting projected values are highly correlated. In other words, it looks for particular combinations of words that appear to have the same co-occurrence patterns in the two languages. Our hypothesis is that finding such correlations across a paired crosslingual corpus will locate the underlying semantics, since we assume that the two languages are 'conditionally independent', or that the only thing they have in common is their meaning. The directions would carry information about the *concepts* that stood behind the process of generation of the text and, although expressed differently in different languages, are, nevertheless, semantically equivalent. To illustrate such representation we have printed a few of the most probable (most typical) words in each language for two kernel canonical correlation components found for bilingual 36^{th} Canadian Parliament corpus (Hansards) (left column is English space and right column is French space) (Notice that not all the components display strong semantic similarity within the language, but all of them achieve strong semantic relations across the two languages. See paper [29] for details about this table):

PENSIONS PLAN?		CANADIAN LANDS?	
pension	regime	park	parc
plan	pensions	land	autochtones
cpp	rpc	aboriginal	terres
canadians	prestations	yukon	ches
benefits	canadiens	marine	vall
retirement	retraite	government	ressources
fund	cotisations	valley	yukon
tax	fonds	water	nord
investment	discours	boards	gouvernement
income	impôt	territories	offices
finance	revenu	board	marin
young	jeunes	north	eaux
years	ans	parks	territoires
rate	pension	resource	parcs
superannuation	argent	agreements	nations
disability	regimes	northwest	territoriales
taxes	investissement	resources	revendications
mounted	milliards	development	ministre
future	prestation	treaty	cheurs

This representation is then used for retrieval tasks, providing better performance than existing techniques. Such directions are then used to calculate the coordinates of the documents in a 'language independent' way. Of course, particular statistical care is needed for excluding 'spurious' correlations. We show that the correlations we find are not the effect of chance, and that the resulting representation significantly improves performance of retrieval systems [31]. It can be observed that the correlation existing between certain sets of words in English and French documents cannot be explained as a random correlation. Hence we need to explain it by means of relations between the generative processes of the two versions of the documents, that we assume to be conditionally independent given the *topic* or *content*. Under such assumptions, hence, such correlations detect similarities in content between the two documents, and can be exploited to derive a semantic representation of the text. This representation is then used for retrieval tasks, providing better performance than existing techniques. In [31] the method has been applied to cross-lingual information retrieval and the comparison of the performance with a related approach based on latent semantic indexing (LSI) [23] has also been made. The projection obtained by CL-KCCA as a semantic smoothing for use in a multilingual classification task obtained very encouraging results [31], [30].

We first give a definition:

Definition 6 *For zero-mean multivariate random variables $x \in \mathbb{R}^n$ and $z \in \mathbb{R}^m$ we define the covariance matrix as follows:*

$$C_{xz} = E[\mathbf{x}\mathbf{z}']$$

We will use also the empirical form of $C_{xz} = \frac{1}{l}\sum \mathbf{x}_i\mathbf{z}_i'$ where $\mathbf{x}_i, \mathbf{z}_i$ are vectors. In

general, given two sets of vectors $\mathbf{x}_i, \mathbf{z}_i$ *we can define the matrix*

$$C = \begin{pmatrix} C_{xx} & C_{xz} \\ C_{zx} & C_{zz} \end{pmatrix} = E\left(\begin{pmatrix} \mathbf{x} \\ \mathbf{z} \end{pmatrix}\begin{pmatrix} \mathbf{x} \\ \mathbf{z} \end{pmatrix}'\right)$$

where C_{xx} *and* C_{zz} *are the within class covariances, and* C_{xz} *and* C_{zx} *are the between class covariances.*

For us, those multivariate random variables will correspond to document-vectors (in the bags of words representation) in English and French, and there will be a one-to-one relation between them corresponding to documents that are translations of each other. We will now consider sets of words that are correlated between the two languages (sets of words in the two languages that have a correlated pattern of appearance in the corpus). We will assume that such sets approximate the notion of 'concepts' in each language, and that such concepts are the translation of each other. Rather than considering plain sets, we will consider terms to have a degree of membership to a given set. In other words, the term t_i will be assigned a weight w_i for each concept we consider, and every concept will correspond to a vector $w_x \in \mathbb{R}^n$ in English, and a vector $w_z \in \mathbb{R}^m$ in French. We will use that weight α_i to form linear combinations of terms, so that they can define a direction in the term space.

Thus, in this study our aim is to find an appropriate language-independent representation. Suppose as for CL-LSI we are given *aligned* texts in, for simplicity, two languages, i.e., every text in one language $x_i \in \mathbb{R}^n$ is a translation of text $z_i \in \mathbb{R}^m$ in another language or vice versa. Our hypothesis is that having aligned texts $S_x = (x_1, ..., x_\ell) \subseteq \mathbb{R}^n$ and $S_z = (z_1, ..., z_\ell) \subseteq \mathbb{R}^m$ we can learn (semantic) directions $\widehat{w}_x$ and $\widehat{w}_z$ where we use the notation $\widehat{w} = \frac{w}{\|w\|}$ so that the projections $\widehat{w}_x' x$ and $\widehat{w}_z' z$ of input data images from the different languages would be maximally correlated. These new random variables are univariate, and linear combinations of the previous ones. We consider optimizing this quantity with respect to the choice of $\widehat{w}_x \in \mathbb{R}^n$ and $\widehat{w}_z \in \mathbb{R}^m$. This leads to the following objective functions and optimization problems:

$$\rho = \max_{w_x, w_z} \operatorname{corr}(\widehat{w}_x' x, \widehat{w}_z' z)$$

This optimization problems can be transformed into a generalized eigenvalue problems as follows. One is looking for the maximum correlation directions:

$$\begin{array}{ll} \text{maximize} & \rho = \dfrac{E[\mathbf{w}_x' \mathbf{C}_{xz}' \mathbf{w}_z]}{\sqrt{E[\mathbf{w}_x' \mathbf{C}_{xx}' \mathbf{w}_x] E[\mathbf{w}_z' \mathbf{C}_{zz}' \mathbf{w}_z]}} \\ \text{subject to} & \|\mathbf{w}_x\| = \|\mathbf{w}_z\| = 1 \end{array}$$

where we are using the covariance matrix:

$$C = \begin{pmatrix} C_{xx} & C_{xz} \\ C_{zx} & C_{zz} \end{pmatrix} = E\left(\begin{pmatrix} \mathbf{x} \\ \mathbf{z} \end{pmatrix}\begin{pmatrix} \mathbf{x} \\ \mathbf{z} \end{pmatrix}'\right)$$

The solutions of this problem can be obtained by solving a related generalized eigenproblem

$$A\mathbf{w} = \lambda B\mathbf{w} \tag{10.6}$$

and the solution $\mathbf{w}$ directly provides the directions $\mathbf{w}_x$ and $\mathbf{w}_z$ of maximum correlation:

$$\begin{aligned} C_{xz}\mathbf{w}_z &= \frac{\mathbf{w}_x' C_{xz}\mathbf{w}_z}{\mathbf{w}_x' C_{xx}\mathbf{w}_x} C_{xx}\mathbf{w}_x \\ C_{zx}\mathbf{w}_x &= \frac{\mathbf{w}_x' C_{xz}\mathbf{w}_z}{\mathbf{w}_z' C_{zz}\mathbf{w}_z} C_{zz}\mathbf{w}_z \end{aligned}$$

Imposing $\mathbf{w}_x' C_{xx}\mathbf{w}_x = \mathbf{w}_z' C_{zz}\mathbf{w}_z = 1$ removes a degree of freedom, and we finally obtain the following solution:

$$\begin{aligned} A &= \begin{pmatrix} 0 & C_{xz} \\ C_{zx} & 0 \end{pmatrix} \\ B &= \begin{pmatrix} C_{xx} & 0 \\ 0 & C_{zz} \end{pmatrix} \\ \mathbf{w} &= \begin{pmatrix} \mu_x \mathbf{w}_x \\ \mu_z \mathbf{w}_z \end{pmatrix} \end{aligned}$$

Note that if λ is an eigenvalue, so is $-\lambda$ thus the spectrum is $\{\lambda_1, -\lambda_1, ..., \lambda_N, -\lambda_N\}$.

Dualization of the problem. One can easily consider the dual representation of these problems by substituting: $\mathbf{w}_x = K_x\alpha_x$ and $\mathbf{w}_z = K_z\alpha_z$ where K_x and K_z being respectively the *kernels* of the two feature spaces, i.e. matrices of the inner products between images of all the data points [9]. After substitution we have: $\mathrm{var}x = \frac{1}{N}\alpha_x' K_x K_x \alpha_x$, $\mathrm{var}z = \frac{1}{N}\alpha_z' K_z K_z \alpha_z$ and $\mathrm{cov}(x,z) = \frac{1}{N}\alpha_x' K_x K_z \alpha$. The generalized eigenvalue problem transforms then into $A\alpha = \lambda B\alpha$ where

$$A = \begin{pmatrix} O & K_x K_z \\ K_z K_x & O \end{pmatrix} \tag{10.7}$$

$$B_{corr} = \begin{pmatrix} K_x(K_x + \gamma I) & O \\ O & K_z(K_z + \gamma I) \end{pmatrix} \tag{10.8}$$

The regularization parameter γ, which can be chosen experimentally [31], not only makes the generalized eigenvalue problem well-posed numerically but also provides a way to control the capacity of the correlation mapping.

10.6 Hypertext

Consider the co-citation matrix of a set of documents. This concept was first introduced in the context of bibliometrics to analyse impact factors or to detect clusters of related documents. Two documents have a positive score if they are cited by the same document. It is natural to extend this idea to a co-link matrix for hypertext documents, possibly also considering out-links as well as in-links. Chang et al. [6] used the co-citation matrix for a problem of retrieval in citation analysis and Chakrabarti at al. [5] exploited similar information in their classification model. Furthermore, one could define link-weighting schemes in a similar way to the term-weighting schemes used in text categorisation, where some links are more informative than others.

Given a linked set of documents, a link matrix M specifies their connectivity: element M_{ij} is non-zero if and only if there is a link from document i to document j. M_{ij} can be set to 1, or maybe to 1 over the total number of references in document i. For a given document i, the algorithm HITS [21] defines a_i (resp. h_j) to be its authority (resp. hub) score. The vector a (resp. h) is defined as $a = M'h = M'Ma$ (resp. $h = Ma = MM'h$).

This definition of course means that h and a are in a relation of duality with each other. A large hub score means that the documents points to many good authorities, and a large authority score means that the document is pointed to by many good hubs. The definition also suggests an iterative procedure for the calculation of a (resp. h), guaranteed to converge to the principal eigenvector of $M'M$ (resp. MM'): starting from an initial value a_0 (resp. h_0), at each step the value of a_t (resp. h_t) is calculated $a_{t+1} \leftarrow M'Ma_t$ and $a_{t+1} \leftarrow \frac{a_{t+1}}{\|a_{t+1}\|_1}$ (similarly for h). This information is also exploited in the PageRank estimator at the basis of Google's search engine [3].

It is possible to define the similarity between two documents based on their connectivity structure: a document can be considered as an indicator vector on the set of documents, d, and its image in the feature space can be obtained simply by $\phi(d) = Md$, so that a kernel is given by $K(d_1, d_2) = d_1MM'd_2'$ (and similarly using the transpose of M). Other ways are possible, for example considering an undirected graph.

On this representation of hypertext documents it is possible to perform all the standard operations discussed in this chapter, like semantic focussing, and the others discussed in other sections [18].

The non-principal eigenvectors of this kernel matrix provide information about the different contexts in which a document appears. Overall, this analysis is similar to the co-citation analysis in bibliometrics, or the co-occurrence analysis that motivated semantic indexing. Each principal component of $M'M$ represents a different 'context'. Similarity between documents can also depend on how many contexts they share.

Remark 4 *Of course the weighting of the links can also be assigned with techniques similar to the term weighting methods used in the vector space models. So a site that is linked by everybody would be less informative than others, and a site that is linked many times within a given document is relevant for that document (link-to-site-i = using-term-i). Measures like idf, df, or entropy, should be very helpful, and in the case of text categorization are well known to be much more relevant than - for example - the kernel parameters, the normalization, the kernel family or other quantities.*

10.7 String Matching Kernels

In this section we describe a kernel between two text documents. The idea is to compare them by means of the substrings they contain: the more substrings in common, the more similar they are. An important part is that such substrings do not need to be contiguous, and the degree of contiguity of one such substring in a document determines how much weight it will have in the comparison [24].

For example: the substring 'c-a-r' is present both in the word 'card' and in the word 'custard', but with different weighting. For each such substring there is a dimension of

the feature space, and the value of such coordinate depends on how frequently and how compactly such string is embedded in the text. In order to deal with non-contiguous substrings, it is necessary to introduce a decay factor $\lambda \in (0,1)$ that can be used to weight the presence of a certain feature in a text (see Definition 7 for more details).

Example. Consider - as simple documents - the words *cat*, *car*, *bat*, *bar*. If we consider only $k = 2$, we obtain an 8-dimensional feature space, where the words are mapped as follows:

	c-a	c-t	a-t	b-a	b-t	c-r	a-r	b-r
ϕ(cat)	λ^2	λ^3	λ^2	0	0	0	0	0
ϕ(car)	λ^2	0	0	0	0	λ^3	λ^2	0
ϕ(bat)	0	0	λ^2	λ^2	λ^3	0	0	0
ϕ(bar)	0	0	0	λ^2	0	0	λ^2	λ^3

Hence, the unnormalised kernel between *car* and *cat* is K(car,cat) $= \lambda^4$, whereas the normalised version is obtained as follows: K(car,car) $= K$(cat,cat) $= 2\lambda^4 + \lambda^6$ and hence K(car,cat) $= \lambda^4/(2\lambda^4 + \lambda^6) = 1/(2 + \lambda^2)$. Note that in general the document will contain more than one word, but the mapping for the whole document is into one feature space: the catenation of all the words and the spaces (ignoring the punctuation) is considered as a unique sequence.

Example. We can compute the similarity between the two parts of a famous line by Kant.

K("science is organized knowledge","wisdom is organized life")

The values for this kernel, and values of $k = 1, 2, 3, 4, 5, 6$ are: $K_1 = 0.580$, $K_2 = 0.580$, $K_3 = 0.478$, $K_4 = 0.439$, $K_5 = 0.406$, $K_6 = 0.370$

However, for interesting substring sizes (eg $k > 4$) and normal sized documents, direct computation of all the relevant features would be impractical (even for moderately sized texts) and hence explicit use of such representation would be impossible. But it turns out that a kernel using such features can be defined and calculated in a very efficient way by using dynamic programming techniques. We derive the kernel by starting from the features and working out their inner product. In this case there is no need to prove that it satisfies Mercer's conditions (symmetry and positive semi-definiteness) since they will follow automatically from its definition as an inner product. This kernel named as string subsequence kernel (SSK) is based on work [32, 13] mostly motivated by bioinformatics applications. It maps strings to a feature vector indexed by all k tuples of characters. A k-tuple will have a non-zero entry if it occurs as a subsequence anywhere (not necessarily contiguously) in the string. The weighting of the feature will be the sum over the occurrences of the k-tuple of a decaying factor of the length of the occurrence.

Definition 7 (String subsequence kernel- SSK) *Let Σ be a finite alphabet. A string is a finite sequence of characters from Σ, including the empty sequence. For strings s, t, we denote by $|s|$ the length of the string $s = s_1 \dots s_{|s|}$, and by st the string obtained by concatenating the strings s and t. The string $s[i:j]$ is the substring $s_i \dots s_j$ of s. We say that u is a subsequence of s, if there exist indices $\mathbf{i} = (i_1, \dots, i_{|u|})$, with $1 \leq i_1 < \dots < i_{|u|} \leq |s|$, such that $u_j = s_{i_j}$, for $j = 1, \dots, |u|$, or $u = s[\mathbf{i}]$ for short.*

The length $l(\mathbf{i})$ *of the subsequence in* s *is* $i_{|u|} - i_1 + 1$. *We denote by* Σ^n *the set of all finite strings of length* n, *and by* Σ^* *the set of all strings*

$$\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n. \tag{10.9}$$

We now define feature spaces $F_n = \mathbb{R}^{\Sigma^n}$. *The feature mapping* ϕ *for a string* s *is given by defining the* u *coordinate* $\phi_u(s)$ *for each* $u \in \Sigma^n$. *We define*

$$\phi_u(s) = \sum_{\mathbf{i}:u=s[\mathbf{i}]} \lambda^{l(\mathbf{i})}, \tag{10.10}$$

for some $\lambda \leq 1$. *These features measure the number of occurrences of subsequences in the string* s *weighting them according to their lengths. Hence, the inner product of the feature vectors for two strings* s *and* t *give a sum over all common subsequences weighted according to their frequency of occurrence and lengths*

$$\begin{aligned} K_n(s,t) &= \sum_{u\in\Sigma^n} \langle \phi_u(s) \cdot \phi_u(t) \rangle = \sum_{u\in\Sigma^n} \sum_{\mathbf{i}:u=s[\mathbf{i}]} \lambda^{l(\mathbf{i})} \sum_{\mathbf{j}:u=t[\mathbf{j}]} \lambda^{l(\mathbf{j})} \\ &= \sum_{u\in\Sigma^n} \sum_{\mathbf{i}:u=s[\mathbf{i}]} \sum_{\mathbf{j}:u=t[\mathbf{j}]} \lambda^{l(\mathbf{i})+l(\mathbf{j})}. \end{aligned}$$

A direct computation of these features would involve $O(|\sum|^n)$ time and space, since this is the number of features involved. It is also clear that most of the features will have non zero components for large documents. In order to derive an effective procedure for computing such kernel, we introduce an additional function which will aid in defining a recursive computation for this kernel. Let

$$\begin{aligned} K'_i(s,t) &= \sum_{u\in\Sigma^i} \sum_{\mathbf{i}:u=s[\mathbf{i}]} \sum_{\mathbf{j}:u=t[\mathbf{j}]} \lambda^{|s|+|t|-i_1-j_1+2}, \\ i &= 1,\ldots,n-1, \end{aligned}$$

that is counting the length from the beginning of the particular sequence through to the end of the strings s and t instead of just $l(\mathbf{i})$ and $l(\mathbf{j})$. We can now define a recursive computation for K'_i and hence compute K_n.

Definition 8 *Recursive computation of the subsequence kernel.*

$$\begin{aligned} K'_0(s,t) &= 1, \text{ for all } s,t, \\ K'_i(s,t) &= 0, \text{ if } \min(|s|,|t|) < i, \\ K_i(s,t) &= 0, \text{ if } \min(|s|,|t|) < i, \\ K'_i(sx,t) &= \lambda K'_i(s,t) + \sum_{j:t_j=x} K'_{i-1}(s,t[1:j-1])\lambda^{|t|-j+2}, \\ & \qquad i = 1,\ldots,n-1, \\ K_n(sx,t) &= K_n(s,t) + \sum_{j:t_j=x} K'_{n-1}(s,t[1:j-1])\lambda^2. \end{aligned}$$

Notice that we need the auxiliary function K' since it is only the interior gaps in the subsequences that are penalised. The correctness of this recursion follows from observing how the length of the strings has increased, incurring a factor of λ for each extra length unit. Hence, in the formula for $K'_i(sx,t)$, the first term has one fewer character, so requiring a single λ factor, while the second has $|t|-j+2$ fewer characters. For the last formula the second term requires the addition of just two characters, one to s and one to $t[1:j-1]$, since x is the last character of the n-sequence. If we wished to compute $K_n(s,t)$ for a range of values of n, we would simply perform the computation of $K'_i(s,t)$ up to one less than the largest n required, and then apply the last recursion for each $K_n(s,t)$ that is needed using the stored values of $K'_i(s,t)$. We can of course create a kernel $K(s,t)$ that combines the different $K_n(s,t)$ giving different (positive) weightings for each n.

Once we have created such a kernel it is natural to normalise to remove any bias introduced by document length. We can produce this effect by normalising the feature vectors in the feature space. Hence, we create a new embedding $\widehat{\phi}(s) = \frac{\phi(s)}{\|\phi(s)\|}$, which gives rise to the kernel

$$\begin{aligned}\widehat{K}(s,t) &= \left\langle \widehat{\phi}(s)\cdot\widehat{\phi}(t)\right\rangle = \left\langle \frac{\phi(s)}{\|\phi(s)\|}\cdot\frac{\phi(t)}{\|\phi(t)\|}\right\rangle \\ &= \frac{1}{\|\phi(s)\|\,\|\phi(t)\|}\langle\phi(s)\cdot\phi(t)\rangle = \frac{K(s,t)}{\sqrt{K(s,s)K(t,t)}}\end{aligned}$$

10.7.1 Efficient computation of SSK

SSK measures the similarity between documents s and t in a time proportional to $n|s||t|^2$, where n is the length of the sequence. It is evident from the description of the recursion in Definition 8, as the outermost recursion is over the sequence length and for each length and each additional character in s and t a sum over the sequence t must be evaluated. However it is possible to speed up the computation of SSK. We now present an efficient recursive computation of SSK that reduce the complexity of the computation to $O(n|s||t|)$, by first evaluating

$$K''_i(sx,t) = \sum_{j:t_j=x} K'_{i-1}(s,t[1:j-1])\lambda^{|t|-j+2}$$

and observing that we can then evaluate $K'_i(s,t)$ with the $O(|s||t|)$ recursion,

$$K'_i(sx,t) = \lambda K'_i(s,t) + K''_i(sx,t).$$

Now observe that $K''_i(sx,tu) = \lambda^{|u|}K''_i(sx,t)$, provided x does not occur in u, while

$$K''_i(sx,tx) = \lambda\left(K''_i(sx,t) + \lambda K'_{i-1}(s,t)\right).$$

These observations together give an $O(|s||t|)$ recursion for computing $K''_i(s,t)$. Hence, we can evaluate the overall kernel in $O(n|s||t|)$ time.

10.7.2 n-grams- a language independent approach

n-grams is a language independent text representation technique. It transforms documents into high dimensional feature vectors where each feature corresponds to a contiguous substring. n-grams are n adjacent characters (substring) from the alphabet A. Hence, the number of distinct n-grams in a text is less than or equal to $|A|^n$. This shows that the dimensionality of the n-grams feature vector can be very high even for moderate values of n. However all these n-grams are not present in a document, thus reducing the dimensionality substantially. For example there are 8727 unique tri-grams (excluding stop words) in the Reuters dataset. Generally during n-grams feature vector formation all the upper-case characters are converted into lower-case characters and space is assumed for punctuation. The feature vectors are then normalised. This is illustrated in the following example.

Example Consider an example that compute a tri-gram, and quad-gram feature vector.

d = "support vector"

The 3-grams are sup upp ppo por ort rt␣ t␣v ␣ve vec ect cto tor, while the 4-grams are supp uppo ppor port ort␣ rt␣v t␣ve ␣vec ecto ctor.

where ␣ represents a space. Systems based on this technique have been applied in situations where the text suffers from errors such as misspelling [4]. The choice of an optimal n varies with text corpora.

10.8 Conclusions

In this chapter we have presented a number of different methods for constructing kernels that capture some of the semantic content in a document, exploiting statistical information about the occurrence and co-occurrence of words. Such kernels have been used with success in categorization, retrieval, cross-language and hypertext analysis. A number of other kernels for text have also recently been proposed for example: Hofmann [14] and Saunders et al. [27] that we did not have space to discuss here. However, the interested reader is referred to the webpage **www.support-vector.net/text-kernels.html** for constantly updated list of papers and links to state of the art work in this direction.

Throughout this chapter we deliberately adopted the approach not to emphasize how to use such kernels, since the inherent modularity of KMs makes it possible to combine this mapping with a number of methodologies, some of them discussed elsewhere in this book, and others in related books (see for example [9, 17]). We dealt explicitly with the Generalized Vector Space family of approaches, although other representations of text involving strings or probabilistic models are important.

Many known Information Retrieval systems, and many new generative models, or pattern matching systems, can be regarded as special cases of kernel methods. Once this is achieved, a whole set of techniques can be used: not just retrieval, but also classification, clustering, novelty detection, regression, density estimation and a large set of operations designed to enhance the performance in kernel methods can be applied as well. A number of key ideas have been presented in this chapter, ranging from the pioneering work of Salton [26] and its recent adaptation by Joachims [16] to kernel methods. Recent advances by Kandola et al. [20] and Vinokourov et al.

[31] have shown that semantic similarity and cross-lingual approaches can be viewed in a kernel framework. In summary, Kernel methods can be considered to provide a natural framework for Text and Hypertext pattern recognition. Deciding if two documents, sentences or words have a similar meaning is much easier than guessing their meaning. This similarity measure can be easily obtained using methods from different disciplines, and can be exploited by kernel-based learning systems. There is still much to be learned about the possibility but also the limitations of such systems.

Bibliography

[1] F. R. Bach and M. I. Jordan, Kernel independent component analysis, *Journal of Machine Learning Research* **3** (2002) 1–48.

[2] R. A. Baeza-Yates & B. A. Ribeiro-Neto, Modern information retrieval, ACM Press, Addison-Wesley (1999).

[3] S. Brin and L. Page, The anatomy of a large-scale hypertextual web search engine, *Computer Networks and ISDN Systems* (1998).

[4] W. B. Cavnar and J. M. Trenkle, N-gram based text categorization, In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval* (1994).

[5] S. Chakrabarti, B. Dom, and P. Indyk, Enhanced hypertext categorization using hyperlinks, In *ACM SIGMOD* (1998).

[6] H. Chang, D. Cohn, and A. McCallum, Creating customized authority lists, In *International Conference on Machine Learning* (2000).

[7] N. Cristianini, J. Shawe-Taylor, and H. Lodhi, Latent semantic kernels, *Journal of Intelligent Information Systems (JJIS)*, 18 (2002).

[8] N. Cristianini, A. Elisseeff, J. Shawe-Taylor, and J. Kandola, On kernel target alignment, In *Neural Information Processing Systems 14 (NIPS 14)* (2001).

[9] N. Cristianini and J. Shawe-Taylor, *An introduction to Support Vector Machines*, Cambridge University Press, Cambridge, UK (2000).

[10] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman, Indexing by latent semantic analysis, *Journal of the American Society for Information Science* **41** (1990) 391–407.

[11] S. T. Dumais, J. Platt, D. Heckerman, and M. Sahami, Inductive learning algorithms and representations for text categorisation, In *Proceedings of the Seventh International Conference on Information and Knowledge Management* (1998) 148–155.

[12] R. Ferrer and R.V. Sole, The small world of human language, *Proceedings of the Royal Society of London Series B - Biological Sciences* (2001) 2261–2265.

[13] D. Haussler, Convolution kernels on discrete structures, Technical Report UCSC-CRL-99-10, UCSC- University of California in Santa Cruz (1999).

[14] T. Hofmann, Learning the similarity of documents: an information-geometric approach to document retrieval and categorization, In *Proceedings of Advances in Neural Information Processing Systems (NIPS'99)*, volume 12, MIT Press (2000).

[15] F. Jiang and M. Littman, Approximate dimension equalization in vector-based information retrieval, In *Proceedings of XXVII International Conference on Machine Learning* (2000).

[16] T. Joachims, Text categorization with support vector machines, In *Proceedings of European Conference on Machine Learning (ECML)* (1998).

[17] T. Joachims, *Learning to Classify Text using Support Vector Machines*, Kluwer (2002).

[18] T. Joachims, N. Cristianini, and J. Shawe-Taylor, Combination kernels for hypertext, In *ICML 2001 - Proceedings of the International Conference on Machine Learning* (2001).

[19] J. Kandola, J. Shawe-Taylor, and N. Cristianini, Learning semantic kernels, (2002) in preparation.

[20] J. Kandola, J. Shawe-Taylor, and N. Cristianini, Learning semantic similarity, In *Neural Information Processing Systems 15 (NIPS 15)* (2002).

[21] J. M. Kleinberg, Authoritative sources in a hyperlinked environment, *Journal of the ACM* **46**(5) (1999) 604–632.

[22] R.I. Kondor and J. Lafferty, Diffusion kernels on graphs and other discrete structures, In *Proceedings of Intenational Conference on Machine Learning (ICML 2002)* (2002).

[23] M. L. Littman, S. T. Dumais, and T. K. Landauer, Automatic cross-language information retrieval using latent semantic indexing, In G. Grefenstette, editor, *Cross language information retrieval*, Kluwer (1998).

[24] H. Lodhi, C. Saunders, N. Cristianini, C. Watkins, and J. Shawe-Taylor, String matching kernels for text classification, *Journal of Machine Learning Research* **2** (2002) 419–444.

[25] G. Salton, J. Allen, C. Buckley, and A. Singhal, Automatic analysis, theme generation, and summarization of machine-readable texts, *Science* **264** (1994) 1421–1426.

[26] G. Salton, A.Wang, and C. Yang, A vector space model for information retrieval, *Journal of the American Society for Information Science* **18** (2000) 613–620.

[27] C. Saunders, J. Shawe-Taylor, and A. Vinokourov, String kernels, Fisher kernels and finite state automata, In *Advances of Neural Information Processing Systems 15* (2002) to appear.

[28] G. Siolas and F. d'Alch Buc, Support vector machines based on a semantic kernel for text categorization, In *IEEE-IJCNN 2000* (2000).

[29] A. Vinokourov and M. Girolami, A probabilistic framework for the hierarchic organisation and classification of document collections, *Journal of Intelligent Information Systems* (Special Issue on Automated Text Categorization) **18** (2002) 153–172.

[30] A. Vinokourov, J. Shawe-Taylor, and N. Cristianini, Cross-language correlation analyis and semantic mapping (2002) in preparation.

[31] A. Vinokourov, J. Shawe-Taylor, and N. Cristianini, Inferring a semantic representation of text via cross-language correlation analysis, In *Advances of Neural Information Processing Systems 15* (2002) to appear.

[32] C. Watkins, Dynamic alignment kernels, Technical Report CSD-TR-98-11, Royal Holloway, University of London (2000).

[33] S.K.M. Wong, W. Ziarko, and P.C.N. Wong, Generalized vector space model in information retrieval, In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'85)* (1985) 18–25.

Chapter 11

An Optimization Perspective on Kernel Partial Least Squares Regression

Kristin P. Bennett and Mark J. Embrechts[1]

Abstract. This work provides a novel derivation based on optimization for the partial least squares (PLS) algorithm for linear regression and the kernel partial least squares (K-PLS) algorithm for nonlinear regression. This derivation makes the PLS algorithm, popularly and successfully used for chemometrics applications, more accessible to machine learning researchers. The work introduces Direct K-PLS, a novel way to kernelize PLS based on direct factorization of the kernel matrix. Computational results and discussion illustrate the relative merits of K-PLS and Direct K-PLS versus closely related kernel methods such as support vector machines and kernel ridge regression.

[1]This work was supported by NSF grant number IIS-9979860. Many thanks to Roman Rosipal, Nello Cristianini, and Johan Suykens for many helpful discussions on PLS and kernel methods, Sean Ekans from Concurrent Pharmaceutical for providing molecule descriptions for the Albumin data set, Curt Breneman and N. Sukumar for generating descriptors for the Albumin data, and Tony Van Gestel for an eficient Gaussian kernel implementation algorithm.

11.1 Introduction

Partial Least Squares (PLS) has proven to be a popular and effective approach to problems in chemometrics such as predicting the bioactivity of molecules in order to facilitate discovery of novel pharmaceuticals. PLS-type algorithms work very well. The algorithms are very resistant to overfitting, fast, easy to implement and simple to tune. Chemometric problems frequently have training data with few points and very high dimensionality. On this type of problem, simple linear least squares regression fails, but linear PLS excels. This ability to do inference in high-dimensional space effectively makes PLS an ideal candidate for a kernel approach. Rosipal and Trejo extended PLS to nonlinear regression using kernels functions [22]. As demonstrated in this chapter, kernel partial least squares (K-PLS) is a very effective general purpose regression approach.

The goal of this work is to make PLS and K-PLS more accessible to machine learning researchers in order to promote the use and extension of this powerful approach. The first step is to understand where PLS comes from and how it works. Thus, we develop a relatively simple yet rigorous derivation of PLS and two K-PLS variants from an optimization perspective. Many published papers contain PLS algorithms and analysis of these algorithms. But, typically no derivation of the algorithm is provided. The second step is to understand the strengths and weaknesses of PLS and K-PLS. So we provide a computational comparison with other kernel regression methods and discuss the relative merits of the approaches and directions for future work. The third step is to get researchers to try K-PLS. The K-PLS code is publicly available as part of the Analyze/StripMiner software at www.drugmining.com.

For simplicity, we focus on the simple regression problem of predicting a single response variable and derive the version of PLS that has been previously kernelized. But the analysis performed here can be easily altered to produce other PLS and K-PLS variations both existing and new. To demonstrate this, we derive the novel DK-PLS algorithm. The mathematical models underlying PLS are closely related to those of principal component analysis (PCA) regression. Thus in Section 11.2.1, we begin with a brief review of PCA. PCA computes components based on input data but does not take into account the response. In Section 11.2.2, we show how PLS can be derived from a logical extension of PCA to include information about the response. In Section 11.3, we investigate two possible methods for constructing a nonlinear PLS algorithm via kernels. In Section 11.4 we discuss practical aspects of a successful K-PLS implementation. In Sections 11.5-11.7, we provide a computational comparison between K-PLS and other kernel regression algorithms such as support vector machines (SVM) and kernel ridge regression (KRR). We conclude with a discussion of the relative merits of the PLS and K-PLS approaches, and future directions for research.

We limit our discussion to regression problems with a single dependent variable. More specifically, given a training set of data $((\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_\ell, \mathbf{y}_\ell))$ $\mathbf{x}_i \in R^{n \times 1}$, $\mathbf{y}_i \in R$, the problem is to construct some function f such that $f(x_i)$ approximately equals $\mathbf{y}_i$ and the function generalizes well on future data. Note PLS and K-PLS are also applicable to multiple regression problems with $\mathbf{y}_i \in R^c$, $c > 1$. The following notation is used. Each data point, $\mathbf{x}_i$, is represented as the i^{th} row in the data matrix $\mathbf{X}$. The i^{th} response is denoted by $\mathbf{y}_i$ where $\mathbf{y}$ is a column vector. Let ℓ denote the number of

points, and $\mathbf{n}$ denote the dimensionality of the data, so $\mathbf{X} \in R^{\ell \times \mathbf{n}}$ and $\mathbf{y} \in R^{\ell \times 1}$. In general, matrices are denoted by bold capital letters such as $\mathbf{A}$. Vectors are denoted by bold lower case letters like $\mathbf{u}$. With the exception of the data points $\mathbf{x}_i$, vectors are presumed to be column vectors unless otherwise noted. The 2-norm or Euclidean norm is denoted by $||\mathbf{y}||$ if $\mathbf{y}$ is a vector, thus $||\mathbf{y}||^2 = \mathbf{y}'\mathbf{y}$. For a matrix $\mathbf{A}$, $||\mathbf{A}||^2$ denotes the square of the Frobenius norm which equals $\sum_i \sum_j (\mathbf{A}_{ij})^2$. Greek letters are used to denote scalars. The transpose of a matrix is denoted by $\mathbf{X}'$. The dot product of two column vectors $\mathbf{u}$ and $\mathbf{v}$ is denoted by $\mathbf{u}'\mathbf{v}$. The outer product of $\mathbf{u}$ and $\mathbf{v}$ is denoted by $\mathbf{u}\mathbf{v}'$. The reader should be careful to distinguish the use of dot products from that of outer products. Note that we assume a data vector $\mathbf{x}$ is a row vector. Thus $\mathbf{x}\mathbf{w}$ denotes the inner product of $\mathbf{x}$ and $\mathbf{w}$. Subscripts are used to indicate components of a matrix or a vector. Superscripts indicate values of a matrix or vector at each iteration.

11.2 PLS Derivation

As the name implies, PLS is based on least-squares regression. Consider the problem of constructing a linear function consisting of the inner product of $\mathbf{x}$ with $\mathbf{w}$, $f(\mathbf{x}) = \mathbf{x}\mathbf{w}$, such that $f(\mathbf{x})$ is approximately $\mathbf{y}$. Note that here $\mathbf{x}$ is a row vector and $\mathbf{w}$ is a column vector. We assume that $\mathbf{y}$ and the data matrix $\mathbf{X}$ have been scaled to have zero column means so that no bias term is necessary. A least squares optimization method constructs $\mathbf{w}$ by minimizing the sum of the squared residuals between the predicted and actual response. The simplest least squares problem is:

$$\min_{\mathbf{w}} \quad \sum_{i=1}^{\ell} \frac{1}{2}(\mathbf{x}_i\mathbf{w} - \mathbf{y}_i)^2 \tag{11.1}$$

where $\mathbf{x}_i$ is a row vector representing the i^{th} data point. For high-dimensional data, the problem must be regularized to prevent overfitting. In ridge regression [12] this is accomplished by penalizing large values of $||\mathbf{w}||^2$ to yield:

$$\min_{\mathbf{w}} \quad \sum_{i=1}^{\ell} \frac{1}{2}(\mathbf{x}_i\mathbf{w} - \mathbf{y}_i)^2 \; + \; \frac{\lambda}{2}||\mathbf{w}||^2. \tag{11.2}$$

PLS's method of regularization or capacity control distinguishes it from other least squares algorithms. In Principal Component Analysis (PCA) regression, the capacity control is performed by mapping the data to a lower-dimensional space and constructing the least squares estimate in that space. PCA ignores all information about $\mathbf{y}$ while constructing this mapping. PLS utilizes information from the response $\mathbf{y}$ to help select a mapping better suited for regression problems. Since PLS is very similar to PCA, we begin with a review of PCA in the next section and then show how this type of analysis can be extended to PLS in the subsequent section.

11.2.1 PCA regression review

PCA regression consists of two steps. First a linear projection or mapping of the data is constructed, using standard PCA. Then the final regression function is constructed

by minimizing the least squares error between the projected data and the response $\mathbf{y}$. Selecting a lower dimensional subspace for the mapping restricts the set of possible regression functions, thus limiting the capacity of the resulting function to overfit the data. This is a form of regularization.

PCA constructs a linear projection of the data that preserves the characteristics of the data as much as possible. The first component of PCA computes the linear projection of the data with maximum variance. But there are alternative derivations of PCA. The first principal component can be derived by minimizing the distance between a data point and its projection on a vector. These are equivalent problems.

If we take a data point $\mathbf{x}_i \in R^{1\times n}$ and project it on the vector $\mathbf{w}' \in R^{1\times n}$, the projected point is $(\mathbf{x}_i\mathbf{w})\mathbf{w}'$. To compute the linear projection that minimizes the distortion between $\mathbf{x}$ and its projection, one can minimize

$$\min_{\mathbf{w}} \sum_{i=1}^{\ell} ||\mathbf{x}_i - \mathbf{x}_i\mathbf{w}\mathbf{w}'||^2 \ s.t.\ \mathbf{w}'\mathbf{w} = 1. \tag{11.3}$$

Note that $\mathbf{x}_i$ is a row vector, $\mathbf{w}$ is a column vector, and $\mathbf{w}\mathbf{w}'$ is an outer product. By simple linear algebra, Problem (11.3) is equivalent to maximizing the variance of the projected data, i.e.

$$\max_{\mathbf{w}} var(\mathbf{X}\mathbf{w}) \ s.t.\ \mathbf{w}'\mathbf{w} = 1. \tag{11.4}$$

Using the constraint to simplify the objective, problem (11.3) can be rewritten as

$$\begin{array}{ll} \min_{\mathbf{w}} & \sum_i [(\mathbf{x}_i - \mathbf{x}_i\mathbf{w}\mathbf{w}')(\mathbf{x}_i' - \mathbf{w}\mathbf{w}'\mathbf{x}_i')] = \sum_i (\mathbf{x}_i\mathbf{x}_i' - \mathbf{x}_i\mathbf{w}\mathbf{w}'\mathbf{x}_i') \\ s.t. & \mathbf{w}'\mathbf{w} = 1. \end{array} \tag{11.5}$$

After dropping the constant terms and converting the problem to a maximization problem, the problem becomes:

$$\min_{\mathbf{w}} -\sum_i \mathbf{x}_i\mathbf{w}\mathbf{w}'\mathbf{x}_i' \ \ s.t.\ \ \mathbf{w}'\mathbf{w} = 1 \tag{11.6}$$

or equivalently

$$\max_{\mathbf{w}} \sum_i ||\mathbf{x}_i\mathbf{w}||^2 \ \ s.t.\ \ \mathbf{w}'\mathbf{w} = 1. \tag{11.7}$$

Assuming that $\mathbf{x}_i$ has mean $\mathbf{0}$, this problem is equivalent to Problem (11.4).

The optimal solution for $\mathbf{w}$ can be easily constructed using the first order optimality conditions [18]. To derive the optimality conditions for Problem (11.7), convert the problem to a minimization problem, construct the Lagrangian function, and set the derivative of the Lagrangian with respect to $\mathbf{w}$ to zero. With λ as the Lagrangian multiplier of the constraint, the Lagrangian function is

$$L(\mathbf{w}, \lambda) = -\sum_i ||\mathbf{x}_i\mathbf{w}||^2 + \lambda(\mathbf{w}'\mathbf{w} - 1).$$

The optimality conditions are

$$\mathbf{X}'\mathbf{X}\mathbf{w} = \lambda\mathbf{w} \qquad \mathbf{w}'\mathbf{w} = 1. \tag{11.8}$$

So we know the optimal $\mathbf{w}$ is just the maximal eigenvalue of the covariance matrix $\mathbf{X}'\mathbf{X}$.

But this yields only a one-dimensional representation of the data. A series of orthogonal projections can be computed by the NIPALS (Nonlinear Iterative Partial Least Squares) algorithm [34]. The data matrix is reduced in order to account for the part of the data explained by $\mathbf{w}$. The data matrix is "deflated" by subtracting away the part explained by $\mathbf{w}$. So at the next iteration the method computes the best linear projection $\mathbf{w}^2$ of $[\mathbf{X}^1 - \mathbf{X}^1\mathbf{w}^1\mathbf{w}^{1\prime}]$ which exactly corresponds to the eigenvector with second largest eigenvalue of $\mathbf{X}'\mathbf{X}$. If $\widehat{M} < n$ orthogonal projects are desired, this procedure is repeated $\widehat{M}$ times. If the projection vectors $\mathbf{w}^i$ are represented as columns in the matrix $\mathbf{W}$, the reduced dimensionality data or latent variables are now $\mathbf{XW}$. We call each column of $\mathbf{XW}$ a latent variable. The matrix $\mathbf{X}$ has now been approximated by the low-rank matrix $\mathbf{XWW}'$. By construction $\mathbf{W}'\mathbf{W} = I$.

In PCA regression, the least-squares loss function is used to compute the final regression function. The least-squares problem in the projected space:

$$\min_{\mathbf{v}\in R^{\widehat{M}}} \frac{1}{2}||\mathbf{XWv} - \mathbf{y}||^2 \tag{11.9}$$

has optimality conditions

$$\mathbf{W}'\mathbf{X}'\mathbf{XWv} - \mathbf{W}'\mathbf{Xy} = 0 \tag{11.10}$$

The optimal value of the regression coefficients in the projected space is $\overline{\mathbf{v}}$. In the original space, the coefficients of the final linear function, $f(\mathbf{x}) = \mathbf{xb}$, are

$$\mathbf{b} = \mathbf{W}\overline{\mathbf{v}} = \mathbf{W}(\mathbf{W}'\mathbf{X}'\mathbf{XW})^{-1}\mathbf{W}'\mathbf{Xy}. \tag{11.11}$$

The final regression function is

$$f(\mathbf{x}) = \mathbf{xb} = \mathbf{xW}(\mathbf{W}'\mathbf{X}'\mathbf{XW})^{-1}\mathbf{W}'\mathbf{Xy} \tag{11.12}$$

assuming $\mathbf{x}$ is a row vector. The final regression function looks complicated as an equation, but conceptually it is fairly simple.

In PCA regression, the degree of capacity control or regularization is controlled by $\widehat{M}$, the sole parameter in PCA. $\widehat{M}$ controls the number of principal components $\mathbf{w}$, or equivalently the number of latent variables $\mathbf{xw}$. If $\widehat{M} = n$, then the method reduces to simple least squares.

11.2.2 PLS analysis

Like PCA, PLS also constructs a mapping of the data to a $\widehat{M} \leq n$ dimensional space and then solves a least squares regression problem in the $\widehat{M}$ dimensional space to calculate the final regression function. Like PCA, this mapping is achieved by constructing a low-rank approximation of the data matrix $\mathbf{X}$. If $\widehat{M} = n$, PLS also becomes simple least-squares regression. PLS methods differ from PCA in only two respects: the mathematical model for computing the mapping and the mathematical model used to compute the final regression coefficients. Unlike PCA, PLS utilizes both the input and the response data, $\mathbf{X}$ and $\mathbf{y}$ respectively, to construct the mapping to a lower dimensional space. In this process it constructs low-rank approximations for both $\mathbf{X}$ and $\mathbf{y}$. The final optimization model optimization utilizes these approximations plus the linear projections to compute the final regression coefficients.

11.2.3 Linear PLS

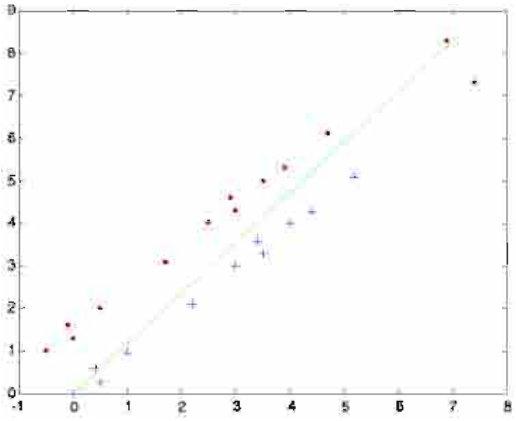


Figure 11.1: Line is direction of maximum variance (w) constructed by PCA.

The entire PLS procedure can be derived by making two simple but profound changes to the optimization problems underlying PCA Regression. The primary problem with PCA Regression, is that PCA does not take into account the response variable when constructing the principal components or latent variables. Thus even for easy classification problems such as in Figure 11.1, the method may select poor latent variables. Thus, the first change is to incorporate information about the response in the model used to construct the latent variables. To simplify notation, we switch to matrix notation and write the PCA Problem (11.4) as

$$\min_{\mathbf{w}} ||\mathbf{X} - \mathbf{X}\mathbf{w}\mathbf{w}'||^2 s.t. \mathbf{w}'\mathbf{w} = 1 \tag{11.13}$$

where $||\mathbf{X} - \mathbf{X}\mathbf{w}\mathbf{w}'||^2 = \sum_i \sum_i (\mathrm{x}_{ij} - \mathrm{x}_i \mathbf{w} \mathbf{w}_j)^2$. If $\mathbf{X}\mathbf{w}$ is approximately $\mathbf{y}$, the regression problem is solvable using only one latent variable. So we simply substitute $\mathbf{y}$ for $\mathbf{X}\mathbf{w}$ into the PCA Problem (11.7) to yield

$$\min_{\mathbf{w}} ||\mathbf{X} - \mathbf{y}\mathbf{w}'||^2 \ s.t. \ \mathbf{w}'\mathbf{w} = 1. \tag{11.14}$$

Does this problem make sense? Figure 11.2 illustrates the resulting PLS direction on the above sample problem. Now the regression function (here for a classification problem) can be constructed using a single latent variable. We know Problem (11.14) constructs the rank-one function of y that mostly nearly maps to the input data x. How does this relates to the regression problem of finding a function of the inputs x that maps to the response y? Using $||\mathbf{w}|| = 1$ and the Cauchy-Schwartz Theorem, we can show that $||\mathbf{X} - \mathbf{y}\mathbf{w}'||^2$ bounds the usual least-squares regression loss function. More precisely for each point training point $(\mathrm{x}_i, \mathrm{y}_i)$:

$$||\mathrm{x}_i\mathbf{w} - \mathrm{y}_i||^2 = ||(\mathrm{x}_i - \mathrm{y}_i\mathbf{w}')\mathbf{w}||^2 \leq ||\mathrm{x}_i - \mathrm{y}_i\mathbf{w}'||^2||\mathbf{w}||^2 = ||\mathrm{x}_i - \mathrm{y}_i\mathbf{w}'||^2.$$

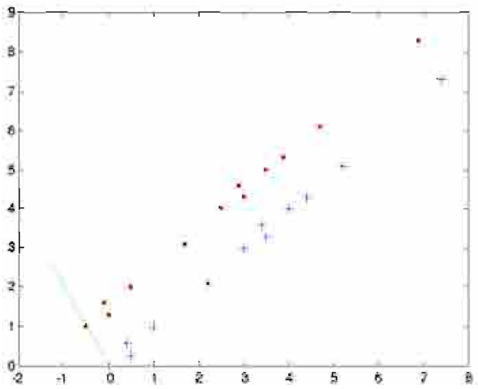


Figure 11.2: Line is direction w constructed by PLS using response (1 or -1) information.

Thus Problem (11.14) minimizes a bound on the least squares loss function but the choice of $\mathbf{w}$ is now greatly restricted.

To our knowledge, Problem (11.14) has not been used before to derive PLS in the literature. It is well known that in PLS the first latent variable $\mathbf{Xw}$ maximizes the sample covariance of $\mathbf{Xw}$ and $\mathbf{y}$. Just as in PCA, the covariance and low rank approximation problems can be shown to be equivalent. Note that this derivation depends on the assumptions that $\mathbf{y'y} = \mathbf{w'w} = 1$, $mean(\mathbf{X}) = \mathbf{0}$, and $mean(\mathbf{y}) = 0$. For any data point $\mathbf{x}$,

$$\begin{aligned} ||\mathbf{x} - \mathbf{yw'}||^2 &= (\mathbf{x} - \mathbf{wy'})(\mathbf{x} - \mathbf{yw'}) = \mathbf{x'x} - 2\mathbf{wx'y} + \mathbf{yw'wy'} \\ &= \mathbf{x'x} - 2\mathbf{wx'y} + \mathbf{yy'} = -2cov(\mathbf{xw}, y) + constant. \end{aligned}$$

Thus after converting the problem to a maximization problem and removing constant terms, Problem (11.14) is equivalent to

$$\max_{\mathbf{w}} cov(\mathbf{Xw}, \mathbf{y}) \ s.t. \ \mathbf{w'w} = 1. \tag{11.15}$$

A wonderful quality of these problems is that they have a closed form solution. The optimality conditions are $\mathbf{X'y} = \lambda \mathbf{w}$ and $\mathbf{w'w} = 1$, so the optimal solution is $\mathbf{w} = \frac{\mathbf{X'y}}{(\mathbf{y'XX'y})}$. The latent variable can be computed in linear time. It is easy to show that $\mathbf{w}$ is the eigenvector of $\mathbf{X'yy'X}$.

Note that these problems are well defined even if the constraint $||\mathbf{w}|| = 1$ is dropped. We would like to derive a version of PLS that is convenient for kernel functions when $\mathbf{w}$ is in feature space. So we will not use the normalized form of $\mathbf{w}$. For the rest of the chapter we will use the unnormalized $\mathbf{w} = \mathbf{X'y}$. We can now derive the rest of the PLS algorithm using the analogous approaches to PCA regression, but we must alter any step that assumes $\mathbf{W'W} = I$.

As in PCA we want to compute a series of directions that provide good low-rank approximations of our data. We can now deflate our data matrix to take into account the explained data. Our current approximation of the original matrix ($\mathbf{X^1}$) is $\mathbf{X^1 w^1 w^{1'}}$

But we can make this approximation a bit better. Define $\mathbf{t}^1 = \frac{\mathbf{X}^1\mathbf{w}^1}{||\mathbf{X}^1\mathbf{w}^1||}$. The best approximation is the matrix $\mathbf{t}^1\mathbf{p}^{1\prime}$ where $\mathbf{p}^1$ solves

$$\min_{\mathbf{p}} ||\mathbf{X}^1 - \mathbf{t}^1\mathbf{p}'||^2. \tag{11.16}$$

By Lemma 1 in the appendix, $\mathbf{p}^1 = \mathbf{X}^{1\prime}\mathbf{t}^{1\prime}$. Thus deflating $\mathbf{X}^1$ can be accomplished as follows:

$$\mathbf{X}^2 = \mathbf{X}^1 - \mathbf{t}^1\mathbf{p}^{1\prime} = \mathbf{X}^1 - \mathbf{t}^1\mathbf{t}^{1\prime}\mathbf{X}^1.$$

This process will generate a series of vectors $\mathbf{t}^i$ that are orthogonal. The matrix $\mathbf{T}$, created using $\mathbf{t}^i$ as the columns of $\mathbf{T}$, is orthogonal.

Recall that $\mathbf{X}^i\mathbf{w}^i$ is also an approximation of $\mathbf{y}$. So it seems reasonable to calculate the residual for the current estimate of $\mathbf{y}$ since that part of $\mathbf{y}$ is in some sense explained. Since $\mathbf{t}^1$ is proportional to $\mathbf{X}^i\mathbf{w}^i$, we calculate the best least-squares fit of $\mathbf{t}^1$ to $\mathbf{y}^1 = \mathbf{y}$. So our best estimate of $\mathbf{y}^1$ is $\mathbf{t}^1\mathbf{c}^{1\prime}$ such that $\mathbf{c}^1 \in R$ solves

$$\min_{\mathbf{c}} ||\mathbf{y}^1 - \mathbf{t}^1\mathbf{c}'||^2.$$

By Lemma 1, $\mathbf{c}^1 = \mathbf{t}^{1\prime}\mathbf{y}^1$. So the residual is just

$$\mathbf{y}^2 = \mathbf{y}^1 - \mathbf{t}^1\mathbf{c}^{1\prime} = \mathbf{y}^1 - \mathbf{t}^1\mathbf{t}^{1\prime}\mathbf{y}^1.$$

For the next latent variable, this process can be repeated using $\mathbf{X}^2$ and $\mathbf{y}^2$ to compute $\mathbf{w}^2$, $\mathbf{t}^2$ etc. until the desired number of latent variables, $\widehat{M}$ is reached.

11.2.4 Final regression components

Once the latent variables have been constructed, the next step is to compute the final regression functions. PLS and PCA use different mathematical models to compute the final regression coefficients. PCA Regression maps the original data into a lower-dimensional space using the projection matrix $\mathbf{W}$ and then computes the least squares solution in this space. Recall that PLS computes an orthogonal factorization of both the input $\mathbf{X}$ and response $\mathbf{y}$ data in the process of computing $\mathbf{W}$. This factorization constructs low rank approximations of $\mathbf{X}$ and $\mathbf{y}$. *The least squares model for PLS is based on these approximations of the input and response data, not the original data.* This is the other key difference between K-PLS and PCA regression. The use of the approximate data instead of the actual training data contributes both to the robustness and computational efficiency of the approach.

Use of the data factorization makes computing the final regression coefficients more efficient. Let $\mathbf{T} \in R^{\ell \times K}$ denote the matrix formed by using each $\mathbf{t}^i$ as a column and similarly for $\mathbf{W}$. We know $\mathbf{T}$ was constructed to be a good factorization of both $\mathbf{X}$ and $\mathbf{y}$. By Lemma 1 in the Appendix and the fact that $\mathbf{T}'\mathbf{T} = I$, one can show $\mathbf{P} = \mathbf{X}'\mathbf{T}$ solves $\min_{\mathbf{P}} ||\mathbf{X} - \mathbf{T}\mathbf{P}'||^2$ and therefore $\mathbf{T}\mathbf{P}'$ is the best approximation of $\mathbf{X}$ based on $\mathbf{T}$. Similarly the best approximation of $\mathbf{y}$ is $\mathbf{T}\mathbf{C}'$ where $\mathbf{C} = \mathbf{y}'\mathbf{T}$ solves $\min_{\mathbf{C}} ||\mathbf{y} - \mathbf{T}\mathbf{C}'||^2$. The final regression problem is: construct $\mathbf{v}$ such that

$$\min_{\mathbf{v}} ||(\mathbf{T}\mathbf{P}')\mathbf{W}\mathbf{v} - \mathbf{T}\mathbf{C}'||^2 \quad = \quad \min_{\mathbf{v}} ||\mathbf{T}(\mathbf{P}'\mathbf{W}\mathbf{v} - \mathbf{C}')||^2 \tag{11.17}$$

Note this is identical to the problem solved in PCA Regression except the approximation of $\mathbf{X}$ and $\mathbf{y}$ are used instead of the actual data. If $(\mathbf{P}'\mathbf{W}\mathbf{v} = \mathbf{C}')$ has a solution $\overline{\mathbf{v}}$, then $\mathbf{v}$ is the optimal solution of Problem (11.17) since the objective will then be zero, the lowest possible value. It turns out that $\mathbf{P}'\mathbf{W}$ is always a lower triangular matrix (see [22, 13]) and thus nonsingular. Thus we know that $\overline{\mathbf{v}}$ exists satisfying $\mathbf{P}'\mathbf{W}\mathbf{v} = \mathbf{C}'$. The solution $\widehat{\mathbf{v}}$ can be computed efficiently using forward substitution. For notational convenience we will say $\overline{\mathbf{v}} = (\mathbf{P}'\mathbf{W})^{-1}\mathbf{C}'$. But in fact the inverse matrix need not be calculated explicitly.

Having found $\overline{\mathbf{v}}$, the final regression function can be computed using the same approach in PCA Regression (see (11.11),(11.12)). The final regression functions is $f(\mathbf{x}) = \mathbf{x}\mathbf{b}$ where

$$\mathbf{b} = W(\mathbf{P}'\mathbf{W})^{-1}\mathbf{C}' = W(\mathbf{T}'\mathbf{X}\mathbf{W})^{-1})\mathbf{T}'\mathbf{y}. \tag{11.18}$$

By exploiting the facts that $\mathbf{C}' = \mathbf{T}'\mathbf{y}$ and $\mathbf{P}' = \mathbf{T}'\mathbf{X}$, $\mathbf{b}$ can be calculated solely in terms of $(\mathbf{X}, \mathbf{y}, \mathbf{T}, \mathbf{W})$.

To summarize, the final PLS algorithm for computing a single response variable is:

Algorithm 1 Basic PLS Algorithm *Assume data* $\mathbf{X}^1 = \mathbf{X}$ *and response* $\mathbf{y}^1 = \mathbf{y}$ *have been normalized by column to have mean* 0 *and standard deviation* 1. *The only algorithm parameter is the number of latent variables,* $\widehat{M}$.

1. *For* $m = 1$ *to* $\widehat{M}$
2. $\mathbf{w}^m = \mathbf{X}^{m\prime}\mathbf{y}^m$
3. $\mathbf{t}^m = \mathbf{X}^m\mathbf{w}^m$
4. $\mathbf{t}^m = \mathbf{t}^m/||\mathbf{t}^m||$
5. $\mathbf{t}^{m+1} = \mathbf{X}^m - \mathbf{t}^m\mathbf{t}^{m\prime}\mathbf{X}^m$
6. $\mathbf{y}^{m+1} = \mathbf{y}^m - \mathbf{t}^m\mathbf{t}^{m\prime}\mathbf{y}^m$
7. $\mathbf{y}^{m+1} = \mathbf{y}^{m+1}/||\mathbf{y}^{m+1}||$
8. *Compute final regression coefficients* $\mathbf{b}$

$$\mathbf{b} = \mathbf{W}(\mathbf{T}'\mathbf{X}\mathbf{W})^{-1})\mathbf{T}'\mathbf{y}.$$

where the m^{th} *columns of* $\mathbf{W}$ *and* $\mathbf{T}$ *are* $\mathbf{w}^m$ *and* $\mathbf{t}^m$ *respectively.*

The final regression function is $f(\mathbf{x}) = \mathbf{x}\mathbf{b}$ where the data vector $\mathbf{x}$ is a row vector. PLS requires far less computational time than PCA because PLS computes the latent vector by simply multiplying the data matrix by the residual, e.g. $\mathbf{w}^m = \mathbf{X}^{m\prime}\mathbf{y}^m$. while PCA must compute the eigenvectors of $\mathbf{X}^{m\prime}\mathbf{X}^m$ at each iteration. Because the solution can be represented in terms of the data matrix and works very well on high-dimensional collinear data, PLS is a natural candidate for a kernel method.

11.3 Nonlinear PLS via Kernels

While other nonlinear extensions to PLS have previously been proposed [1, 35, 36, 2], PLS has only just recently been extended to nonlinear regression through the use of kernels. K-PLS exhibits the elegance that only linear algebra is required, once the kernel matrix has been determined. There are two general approaches for kernelizing PLS. The first approach by Rosipal and Trejo is based on the now classic methodology used in SVM [30, 22]. Each point is mapped nonlinearly to a higher dimensional feature space. A linear regression function is constructed in the mapped space corresponding to a nonlinear function in the original input space. In the dual space, the mapped data only appears as dot products and these dot products can be replaced by kernel functions in the final K-PLS algorithm. The second approach is to use PLS to factorize the kernel matrix directly. K-PLS as first introduced by Rosipal and Trejo [22] is derived using approaches analogous to those used for kernel PCA [23]. Direct Kernel PLS (DK-PLS), introduced here, is based on a direct factorization of the kernel matrix. DK-PLS explicitly produces a low rank approximation of the kernel matrix. Thus it is more closely related to other kernel matrix approximation approaches based on sampling or factorization [16, 10, 17, 9, 27, 24]. DK-PLS has the advantage that the kernel does not need to be square. When combined with sampling of the columns of the kernel matrix such as in [17, 16, 10], it is more scalable than the original KPLS.

11.3.1 Feature space K-PLS

A full discussion of the derivation of K-PLS from PLS using the approach of mapping the data to feature space and constructing a linear function in feature space is given in [22]. To generate this approach one defines a mapping Φ and replaces $\mathbf{X}$ with $\Phi(\mathbf{X})$ and propagates the changes required in the algorithm. Thus the basic problem becomes:

$$\min_{\mathbf{w}} ||\Phi(\mathbf{X}) - \mathbf{y}\mathbf{w}'||^2 \;\; s.t. \;\; ||\mathbf{w}||^2 = 1. \tag{11.19}$$

PLS Algorithm 1 can be easily kernelized. Using approaches such as for LS-SVM [26], the optimality conditions of this problem can be constructed in the dual space. This produces a formulation equivalent to [22] and the reader should consult that paper for more details. We just summarize here and provide the simplified algorithm for one response variable. In Algorithm 1, there is no need to explicitly calculate $\mathbf{W}$. Steps 2 and 3 can be combined. The final regression coefficient can be rewritten to exploit the fact that $\mathbf{w}^m = \mathbf{X}^m\mathbf{y}^m = \mathbf{X}\mathbf{y}^m$. In feature space, step 6 cannot be explicitly performed. But since $\mathbf{X}^m$ only appears in the expression $\mathbf{X}^m\mathbf{X}^{m\prime}$, the kernel matrix can be deflated directly. The K-PLS simplified for one response becomes:

Algorithm 2 Kernel Partial Least Squares [22]

Let $\mathbf{K}^0 = \Phi(\mathbf{X})\Phi(\mathbf{X})'$, *i.e.* $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$, *be the Gram matrix in feature space. Let* $\mathbf{K}^1$ *be the centered form of* $\mathbf{K}^0$. *Let response* $\mathbf{Y}^1 = \mathbf{y}$ *be normalized to have mean* 0 *and standard deviation* 1. *Let* $\widehat{M}$ *be the desired number of latent variables.*

1. *For* $m = 1$ *to* $\widehat{M}$
2. $\mathbf{t}^m = \mathbf{K}^m\mathbf{y}^m$

3. $\mathbf{t}^m = \mathbf{t}^m / ||\mathbf{t}^m||$

4. $\mathbf{K}^{m+1} = (I - \mathbf{t}^m \mathbf{t}^{m\prime}) \mathbf{K}^k (I - \mathbf{t}^m \mathbf{t}^{m\prime})$

5. $\mathbf{y}^{m+1} = \mathbf{Y}^m - \mathbf{t}^m \mathbf{t}^{m\prime} \mathbf{Y}^m$

6. $\mathbf{y}^{m+1} = \mathbf{y}^{m+1} / ||\mathbf{y}^{m+1}||$

7. *Compute final regression coefficients* α

$$\alpha = \mathbf{Y}(\mathbf{T}'\mathbf{K}^{1\prime}\mathbf{Y})^{-1})\mathbf{T}'\mathbf{y}$$

where the m^{th} *columns of* $\mathbf{Y}$ *and* $\mathbf{T}$ *are* $\mathbf{y}^m$ *and* $\mathbf{t}^m$ *respectively.*

$$f(x) = \sum_{i=1}^{\ell} \widehat{K}(\mathbf{x}, \mathbf{x}_i)\alpha_i$$

Note that the training and testing kernels must be centered. See equation (11.21).

11.3.2 Direct kernel partial least squares

Direct Kernel Partial Least Squares factorizes the kernel matrix directly and then computes the final regression function based on this factorization. Let $\mathbf{K} = \Phi(\mathbf{X})\Phi(\mathbf{X})'$, i.e. $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$, be the Gram matrix in feature space. Assume $\mathbf{K}$ and $\mathbf{y}$ have been centered so that no bias term is required. The underlying problem for DK-PLS is:

$$\min_{\alpha} ||\mathbf{K} - \mathbf{y}\alpha'||^2 \tag{11.20}$$

DK-PLS constructs a low rank approximation of the kernel matrix, and then uses this approximation to construct the final function. Strategies for improving kernel methods through factorization have been receiving increasing attention [29, 9]. DK-PLS not only computes a factorization very efficiently relative to eigenvector methods, but it produces a low-rank approximation biased for good performance on the regression task. Algorithm 1 is converted to DK-PLS by simply substituting $\mathbf{K}$ for $\mathbf{X}$. Any variant of PLS can be adapted using the direct kernel approach. The resulting algorithms may be more efficient especially if the kernel matrix is not square. DK-PLS does not assume that the kernel matrix is square so the data maybe sampled to construct the kernel matrix such as in RSVM [10]. When coupled with such a sampling strategy, DK-PLS is more scalable than K-PLS.

Algorithm 3 Direct Kernel Partial Least Squares

Let $\mathbf{K}^0 = \Phi(\mathbf{X})\Phi(\mathbf{X})'$, *i.e.* $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$, *be the Gram matrix in feature space. Let* $\mathbf{K}^1$ *be the centered form of* $\mathbf{K}^0$. *Let response* $\mathbf{Y}^1 = \mathbf{y}$ *be normalized to have mean* 0 *and standard deviation* 1. *Let* $\widehat{M}$ *be the desired number of latent variables.*

1. *For* $k = 1$ *to* $\widehat{M}$

2. $\mathbf{t}^m = \mathbf{K}^m \mathbf{K}^{m\prime} \mathbf{y}^m$

3. $\mathbf{t}^m = \mathbf{t}^m / ||\mathbf{t}^m||$

4. $\mathbf{K}^{m+1} = \mathbf{K}^m - \mathbf{t}^m \mathbf{t}^{m\prime} \mathbf{K}^m$

5. $\mathbf{y}^{m+1} = \mathbf{y}^m - \mathbf{t}^m \mathbf{t}^{m\prime} \mathbf{Y}^m$

6. $\mathbf{y}^{m+1} = \mathbf{y}^{m+1} / ||\mathbf{y}^{m+1}||$

7. *Compute final regression coefficients* α

$$\alpha \;=\; \mathbf{K}^1\mathbf{Y}(\mathbf{T}'\mathbf{K}^1\mathbf{K}^{1\prime}\mathbf{Y})^{-1}\mathbf{T}'\mathbf{y}$$

where the m^{th} *columns of* $\mathbf{Y}$ *and* $\mathbf{T}$ *are* $\mathbf{y}^m$ *and* $\mathbf{t}^m$ *respectively.*

$$f(x) = \sum_{i=1}^{\ell} \widehat{K}(x_i, x)\alpha_i$$

Note that the testing kernel must be centered.

11.4 Computational Issues in K-PLS

K-PLS requires that the user specifies the number of latent variables. Selecting a modest number of latent variables effectively constitutes regularization for K-PLS. The number of latent variables can be determined by either i) tuning on a validation set or ii) adhering to a policy. It has been our experience that for particular types of datasets the optimal number of latent variables does not change by much and the prediction performance is generally not extremely sensitive to the optimal choice for the number of latent variables. On chemometric data, we generally adopt the policy of using five latent variables, regardless of the dataset (but Mahalanobis scale the data first). For large datasets or data that are known to be highly nonlinear (e.g., twisted spiral), the optimal number of latent variables is estimated using a validation set.

As in Kernel PCA, centering the kernel is important to make K-PLS (and especially DK-PLS) work properly. The idea of kernel centering is to force the bias term to be zero. The important thing to remember when centering kernels is that the training set and the test set kernels should be centered in a consistent manner. Kernel centering can be implemented using the following formulas for centering the training kernel and test kernel as suggested by Wu et al. [39, 23, 22]. It is important to note that the equation for the test kernel is based on the un-centered training kernel:

$$\begin{array}{rcl} K^{train}_{center} & = & (\mathbf{I} - \frac{1}{\ell}\mathbf{1}\mathbf{1}')K^{train}(\mathbf{I} - \frac{1}{\ell}\mathbf{1}\mathbf{1}') \\ K^{test}_{center} & = & (K^{test} - \frac{1}{\ell}\mathbf{1}\mathbf{1}'K^{train})(\mathbf{I} - \frac{1}{\ell}\mathbf{1}\mathbf{1}') \end{array} \tag{11.21}$$

where 1 is a vector of ones and I is an identity matrix of appropriate dimension. Above expressions for kernel centering are mathematically elegant and can be rapidly programmed in MATLAB, but are numerically inefficient. In our implementation we essentially adhere to the formulation above, but micro-encode for optimal performance.

A numerically equivalent but more efficient kernel centering proceeds as follows. For the centered training kernel subtract the average for each column of the training

kernel from the column entries, and then subtract the average row value of the modified training kernel row value from each row entity. The training average column values can be kept in memory and used to center the test kernel in a similar way. For centering the test kernel, the average value of the columns of the training kernel is subtracted from each column entry in the test kernel, succeeded by subtracting the average row value from each row entry of the recently modified test kernel. Note that the kernel need not be square.

11.5 Comparison of Kernel Regression Methods

In this section, we compare several different types of kernel regression methods with kernel PLS.

11.5.1 Methods

Different methods considered in this benchmark study include: i) Linear Partial Least Squares [32, 33, 34, 37] (PLS); ii) Linear Proximal Support Vector Machines [10] (P-SVM Lin); iii) K-PLS algorithm as proposed by Rosipal and Trejo [22] with kernel centering (K-PLS); iv) direct kernel PLS (DK-PLS), which factorizes the centered kernel matrix directly as described above; v) LS-SVM also known as Kernel Ridge Regression [26, 7, 8] (LS-SVM) applied to the centered kernel; vi) the reduced form of Least-Squares Support Vector Machines [10] (LS-RSVM) applied to the centered kernel; and viii) classic SVM as implemented in SVM-Torch [30, 5]. The kernel was not centered for SVM-TORCH. More precisely, the LS-SVM solution is produced by solving the following set of equations (using notation in [3]):

$$(\mathbf{K} + \lambda \mathbf{I})\alpha = \mathbf{y} \tag{11.22}$$

to produce the following function

$$f(\mathbf{x}) = y'(\mathbf{K} + \lambda \mathbf{I})^{-1}\mathbf{k} \tag{11.23}$$

where $\mathbf{k}_i = K(x, x_i), i = 1, \ldots, m$. The LS-RSVM method is constructed by solving the following equations:

$$(\mathbf{K}'\mathbf{K}\alpha - \lambda \mathbf{I})\alpha = \mathbf{K}'\mathbf{y} \tag{11.24}$$

to produces the final regression functions:

$$f(\mathbf{x}) = y'\mathbf{K}(\mathbf{K}'\mathbf{K} + \lambda \mathbf{I})^{-1}\mathbf{k} \tag{11.25}$$

where $\mathbf{k}_i = K(x, x_i), i = 1, \ldots, m$. For all kernel methods except SVM-Torch, the kernel was centered. Note that LS-RSVM does not require the kernel matrix to be square to be well defined. All the computer coding was efficiently implemented in C in the Analyze/StripMiner code available from the DDASSL website [6]. The Least Squares SVM variants (LS-SVM, LS-RSVM and LS-RSVM lin) apply an extension of scaled conjugate gradient method introduced by Möller (in a very different context) [19] for fast equation solving. The scaled conjugate gradient method is effectively a Krylov method [14] and the computation time for solving a linear set of ℓ equations scales roughly as $50\ell^2$, rather than ℓ^3 for traditional equation solvers.

11.5.2 Benchmark cases

The benchmark studies comprise four binary classification problems and three true regression cases. The classification cases were treated as if they were regression problems. The classification benchmark data sets were obtained from the UCI Machine Learning Repository [20] and are BUPA Liver, Ionosphere, Tic-Tac-Toe, and Mushroom. The regression cases include Boston Housing, Abalone, and Albumin. The Boston Housing data were obtained from the UCI data repository. The Abalone data, which relate to the prediction of the age for horseshoe crabs [21], were obtained from http://ssi.umh.ac.be/abalone.html. The Albumin dataset [4] is a public QSAR drug-design-related dataset and can be obtained from the DDASSL homepage [6]. The aim of the Albumin dataset is predicting the binding affinities of small molecules to the human serum albumin.

With an exception for the mushroom data, all the computations were performed on a 300 MHz Pentium II with 128 MB of memory. The calculations for the mushroom and abalone data were performed on a 1.8 GH Pentium IV. Because the compiler was not optimized for a Pentium IV Processor, the reported execution times for the mushroom and abalone data were recalibrated to the estimated run time on a 300 MHz Pentium II.

11.5.3 Data preparation and parameter tuning

All data were first Mahalanobis scaled before generating the kernel. The kernel function used in this study is a Gaussian kernel ($K(u,v) = \exp(-||u-v||^2/2\sigma^2)$. The value of σ for each dataset was tuned on a validation set before proceeding with the calculations. The λ parameter for penalizing the two-norm of the weights in all the least squares methods (LS-SVM, LS-RSVM, and LS-RSVM lin) were heuristically determined by the following relationship:

$$\lambda = 0.05\left(\frac{\ell}{200}\right)^{\frac{3}{2}} \tag{11.26}$$

where ℓ is the number of training data. We found that this particular policy choice for λ gives near optimal performance for 40 different benchmark datasets that we have tested so far. SVM-Torch is executed in a regression mode and the C parameter is heuristically determined by the above formula, but now $C = 1/\lambda$. The ϵ parameter for SVM-Torch was tuned on a validation set. All cases were executed in a 100-times leave-10-percent-out mode (100xLOO10). The number of latent variables for PLS and the K-PLS methods was held fixed by policy to five for most cases, but tuned on a validation set if the optimal value was considered to be relevantly different. The results for the binary classification benchmark cases are summarized in Table 11.1, and the results for the regression cases are shown in Table 11.2. The best performance for prediction accuracy are indicated in bold. The results for both regression and classification are summarized in Table 11.3.

For the binary classification cases, the results are presented as the number of cases that were misclassified without providing any details on the false positive/false negative ratio. For the regression cases the results are presented by the least-mean square error and Q2 defined below. Q2 is the square of the correlation between the actual and

Method	Data Set mxn	BUPA Liver 345x6	Ionosphere 351x34	Tic-Tac-Toe 958x9	Mushroom 8124x22
PLS (linear)	Test (%)	30.0	12.7	31.1	12.3
	Time (s)	0.52	1.89	2.69	56.18
P-SVM (linear)	Test	30.0	13.7	31.5	12.4
	Times	0.05	4.26	1.37	37.6
K-PLS	Test	27.9	**4.2**	**0.1**	**0.0**
	Time	47.47	63.48	1044.24	$2.01*10^5$
DK-PLS	Test	28.1	5.5	6.8	3.75
	Time	40.51	54.62	799.31	$1.05*10^5$
LS-SVM	Test	29.0	5.5	3.9	0.04
	Time	77.00	54.40	590.4	$6.76 *10^4$
LS-RSVM	Test	**27.5**	4.3	9.5	0.11
	Time	211.55	300.08	4313	$1.12 *10^5$
SVM-Torch	Test	28.9	5.0	1.5	0.08
	Time	258.03	242.00	1161.81	$1.44*10^5$

Table 11.1: Binary Classification Benchmark Cases Average Misclassification Rate (100 x, leave 10% out mode)

predicted response. Since Q2 is independent on the scaling of the data [11], it is generally useful to compare the relative prediction performance for different datasets for regression. Let $\mathbf{y}_i$ be the i^{th} response, $\widehat{\mathbf{y}}_i$ be the predicted response, $\mu_{\mathbf{y}}$, $\sigma_{\mathbf{y}}$, $\mu_{\widehat{\mathbf{y}}}$, and $\sigma_{\widehat{\mathbf{y}}}$ be the sample means and standard deviations of the actual and predicted responses, then

$$Q^2 = \frac{\sum_{ij}((\widehat{\mathbf{y}}_i - \mu_{\widehat{\mathbf{y}}})(\mathbf{y}_j - \mu_{\mathbf{y}}))}{\sigma_{\widehat{\mathbf{y}}}\sigma_{\mathbf{y}}} \tag{11.27}$$

11.5.4 Results and discussion

The best performance method and particular choice of tuning parameters for each benchmark problem is summarized in Table 11.4. The reported execution times in this table are for K-PLS only, to allow for a consistent comparison between different datasets. It can be observed that K-PLS generally compares well in prediction performance and computing time with other kernel methods.

The K-PLS method has the general advantage that the tuning is robust and simpler than other methods. Other than the choice of kernel, the only parameter is the number of latent variables. One need only consider a few discrete values as opposed to the continuous parameters in SVM and Ridge Regression. Only for datasets that exhibit pronounced non-linearity was it necessary to choose more than 5 latent variables. As currently implemented, K-PLS and DK-PLS have the restriction that they require a machine with sufficient memory to store the full kernel matrix in memory. To allow processing of large sets in DK-PLS, a smaller rectangular kernel can be constructed using sampling. This was not required in these experiments. Results for DK-PLS are generally comparable to those obtained from K-PLS.

Method	Data Set nxn	Boston Housing 506x13	Albumin 94x551	Abalone 4177x8
PLS (linear)	Test Q2	0.28	0.36	0.49
	LMSE	4.92	0.35	2.22
	Time(s)	1.35	10.73	17.75
P-SVM (linear)	Test Q2	0.28	0.42	0.49
	LMSE	4.88	0.4	2.22
	Times(s)	1.18	69.7	5.85
K-PLS	Test Q2	**0.13**	0.35	**0.44**
	LMSE	3.40	0.35	2.12
	Time(s)	181.05	34.24	26224
DK-PLS	Test Q2	0.18	**0.33**	0.45
	LMSE	3.9	0.34	2.12
	Time(s)	147.99	32.37	14202
LS-SVM	Test Q2	0.14	0.35	0.47
	LMSE	3.49	0.35	2.18
	Time(s)	167.60	36.71	8286
LS-RSVM	Test Q2	0.18	0.33	0.45
	LMSE	3.88	0.34	2.14
	Time(s)	661.49	39.17	170925
SVM-Torch	Test Q2	0.16	0.38	0.44
	LMSE	3.70	0.37	2.15
	Time(s)	212.43	368.0	6675

Table 11.2: Regression Benchmark Cases Q2 Error (100 x leave 10% out mode)

Data Set	nxm	σ	λ	K	type	error	method	Time (s)
BUPA Liver	345x6	3.5	0.09695	5	class	27.5%	LS-SVM	47.47
Ionosphere	351x34	3.5	0.09930	5	class	4.2%	K-PLS	63.48
Tic-Tac-Toe	958x9	2.0	0.44817	20	class	0.1%	K-PLS	1044.00
Mushroom	8124x22	2.0	11.05298	12	class	0.0%	K-PLS	$2.01*10^5$
Boston Housing	506x13	5.0	0.17214	12	reg	0.13	K-PLS	181.00
Albumin	94x551	40.0	0.01385	5	reg	0.33	DK-PLS	34.00
Abalone	4177x8	4.0	4.07574	12	reg	0.44	K-PLS	26224.00

Table 11.3: Classification and Regression Results Summary

Table 11.4: Benchmark Summary Results

	Classified as negative class	Classified as positive class
Belonging to negative class	998 (TN)	165 (FP)
Belonging to positive class	17 (FN)	122 (TP)

Table 11.5: Confusion matrix for the checkerboard example of Fig. 11.3 with a zero threshold for discriminating between the negative class and the positive outlier class

11.6 Case Study for Classification with Uneven Classes

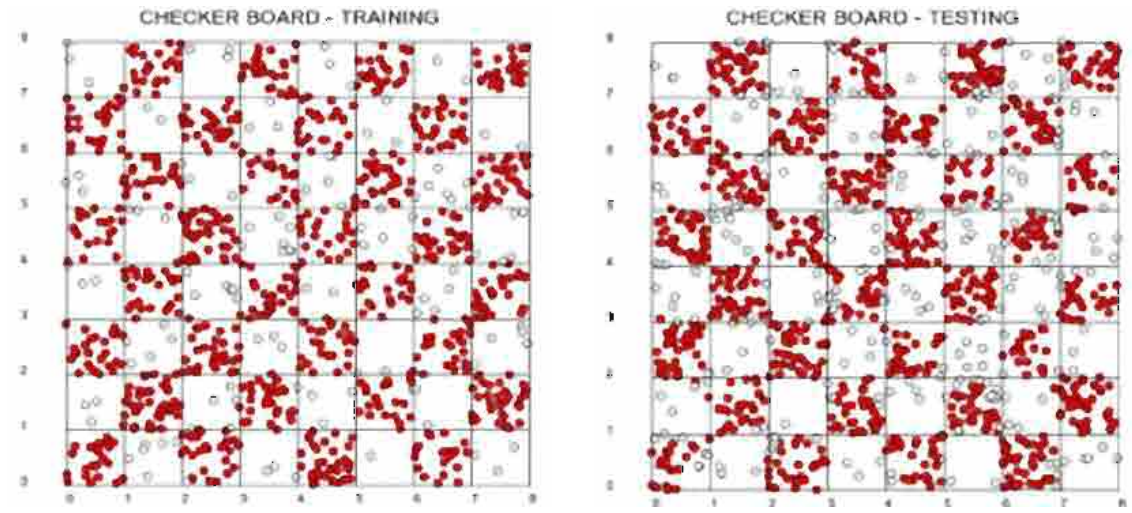


Figure 11.3: Training set and test set performance for an 8x8 checkerboard toy problem. where the red class (the 1 class) is the majority class. There are 1000 training samples (including 121 minority patterns) and 1302 test samples (including 139 minority patterns).

Results from applying K-PLS to a 8x8 checkerboard problem for uneven classes are shown in Figure 11.3. In this case we used 12 latent variables and a Gaussian kernel with a kernel width of 0.08. There are 1000 training samples (including 121 minority patterns) and 1302 test samples (including 139 minority patterns). The results obtained from LS-SVM and LS-RSVM were comparable to those obtained with K-PLS and are not shown. The confusion matrix for the test set for a zero threshold for classification between the +1 and the 1 patterns is shown in Table 11.5.

11.7 Feature Selection with K-PLS

Feature selection, the process of determining which features to include in the input space, can further enhance the performance of K-PLS. Analyze/StripMiner has a fea-

ture selection method incorporated based on sensitivity analysis as described by Kewley and Embrechts [15]. Sensitivity analysis monitors the response as the features are tweaked one-at-a-time within their allowable range, while holding the other input features constant at their average value. Features that cause a larger variation in the response are deemed more important. Sensitivity analysis for feature selection generally requires multiple passes where just a small fraction of the features (10-30%) are dropped at a time. The features selection method implemented in Analyze/StripMiner operates in a leave-several-out mode where the sensitivities are averaged over multiple bootstraps.

To evaluate the relative importance of a feature, the dataset can be augmented with a random gauge variable: features that are less sensitive than the random gauge variables are not likely to be important and dropped from the dataset. After these features have been dropped a new model is built, the sensitivities are determined again, and more features are dropped. This process is then repeated until all features show higher sensitivities than the random gauge variable.

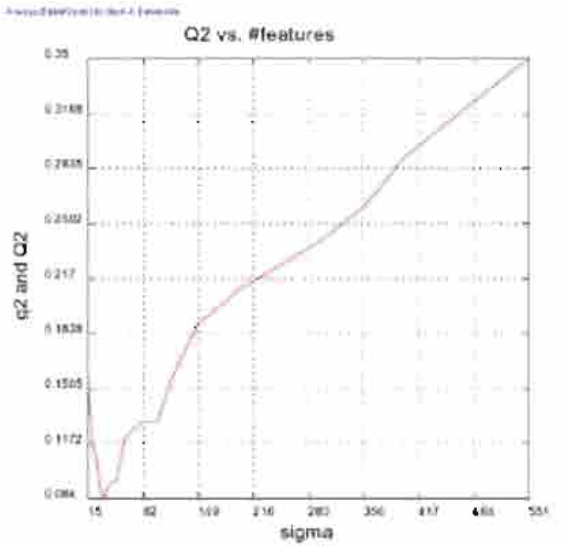


Figure 11.4: Increased prediction performance on the test set for the albumin dataset for 5 latent variables as indicated by the Q2 measure as a function of the reduced number of selected features by successively applying sensitivity analysis to K-PLS. The optimal number of features is 35 with a Q2 of 0.084.

In this section we will report on feature reduction studies on the albumin dataset. The Albumin dataset is a pharmaceutical dataset for predicting binding affinities to the human serum Albumin [4]. The basic descriptive features consist of 511 MOE and wavelet descriptors generated and made available as a public benchmark dataset in the Drug Design and Semi-Supervised Learning (DDASSL) project [6].

Figure 11.4 shows the change of Q2 as a function of the number of features. The optimal performance is for 35 features with a Q2 of 0.084 for 5 latent variables. This performance changes relatively little until the number of features is further reduced to

20. Figure 11.4 shows the scatterplot for the test set for the albumin data when the number of features is successively reduced from the original 511 to 35 features. Figure 11.5 shows the change in Q2 when the number of latent variables changes from 1 to 12.

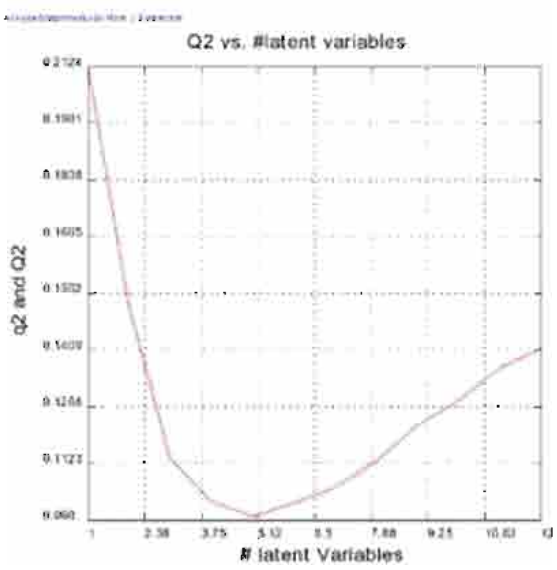


Figure 11.5: Prediction performance on the test set for the albumin dataset as indicated by the Q2 measure as a function of the selected number of latent variables for the optimal selection of 35 features.

11.8 Thoughts and Conclusions

This chapter presents a novel derivation of PLS from an optimization perspective. This derivation provides a basis for the machine learning researcher to better understand how and why PLS works and how it can be improved. PLS has two basic tricks. PLS produces low-rank approximations of the data that are aligned with the response and then uses low-rank approximations of both the input data and response data to produce the final regression model. We showed that from this perspective there are two general options for introducing kernels into PLS. The K-PLS algorithm of Rosipal and Trejo results from applying PLS to the data mapped into feature space. An alternative approach is to use PLS to make low-rank approximations of the kernel or Gram matrix. This produces the Direct K-PLS algorithm. DK-PLS approach can be used with any of the many PLS variants. Rectangular kernels can be factorized with DK-PLS. Combined with sampling of the kernel columns, this is a significant advantage on large datasets where full evaluation of the kernel is not possible. Several benchmark studies show that K-PLS and DK-PLS are comparable with other kernel-based support vector machine approaches in prediction performance and execution time, and tend to show a slightly better performance in general. PLS and variants are

much easier to tune than SVM methods, because the only parameter (beyond choice of kernel) is the number of latent variables. Various options for K-PLS are implemented in the Analyze/StripMiner code, which operates on Windows and Linux platforms. The executable code and complimentary JAVA scripts for graphics are available for academic evaluation and scholarly use from www.drugmining.com. K-PLS and DK-PLS are outstanding methods when the kernel matrix can fit in memory.

Future research is need to fully exploit the potential of K-PLS type algorithms. Statistical learning theory may be able to shed light on why K-PLS generalizes well. Variants of K-PLS may help address its limitations. Current implementations require the full kernel matrix in memory because of the lack of sparsity in the solution and to support kernel centering and deflation. New variants of K-PLS with sparse solutions are needed that do not require full evaluation of the kernel matrix for both training and prediction of new points. Kernel centering could be eliminated by introducing bias when constructing the latent variables. Efficient algorithms for deflating the kernel matrix are needed.

Appendix

In this chapter we are frequently required to compute the best rank-one approximation of the matrix $\mathbf{X} \in R^{\ell \times n}$ given the matrix $\mathbf{t} \in R^{\ell \times 1}$. Specifically we want to find $\mathbf{p} \in R^{n \times k}$ that solves

$$\min_{\mathbf{p}} \sum_{i=1}^{\ell} ||\mathbf{X}_i - \mathbf{t}\mathbf{p}'||^2 = \sum_i \sum_j (\mathbf{X}_{ij} - \mathbf{t}_i \mathbf{p}_j)^2 \tag{11.28}$$

Thus we prove the following Lemma.

Lemma 1 *The solution of problem (11.28) is* $\mathbf{p} = \frac{\mathbf{X}'\mathbf{t}}{||\mathbf{t}||^2}$.

PROOF. Taking the partial derivatives and setting them equal to 0 yields the optimality conditions:

$$\sum_i (\mathbf{t}_i \mathbf{X}_{ij} - \mathbf{t}_i^2 \mathbf{p}_j) = 0 \quad \text{for } j = 1, \ldots, n.$$

Thus

$$\sum_i (\mathbf{t}_i \mathbf{X}_{ij}) = ||\mathbf{t}||^2 \mathbf{p}_j \quad \text{for } j = 1, \ldots, n.$$

□

Bibliography

[1] G. Baffi, E. B. Martin, and A. J. Morris, Non-Linear Projection to Latent Structures Revisited: the Quadratic PLS Algorithm, *Computers and Chemical Engineering* **23** (1999) 395–411.

[2] A. Berglund and S. Wold, INLR, Implicit Non-Linear Latent Variable Regression, *Journal of Chemometrics* **11** (1997) 141–156.

[3] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning methods*, Cambridge: Cambridge University Press (2000).

[4] G. Colmenarejo, A. Alvarez Pedraglio, and J.-L. Lavandera, Chemoinformatic Models to Predict Binding Affinities to Human Serum Albumin, *Journal of Medicinal Chemistry* **44** (2000) 4370–4378.

[5] R. Collobert, and S. Bengio, Support Vector Machines for Large-Scale Regression Problems, IDIAP-RR-00-17 (2000).

[6] M. J. Embrechts, K. P. Bennett, and C. Breneman, www.drugmining.com (2002).

[7] T. Evgeniou, M. Pontil, and T. Poggio, T. Statistical Learning Theory: A Primer, *International Journal of Computer Vision* **38**(1) (2000) 9–13.

[8] T. Evgeniou, M. Pontil, and T. Poggio, Regularization Networks and Support Vector Machines, in *Advances in Large Margin Classifiers*, MIT Press, Boston (2000).

[9] S. Fine and K. Scheinberg, Efficient SVM Training Using Low-Rank Kernel Representations, *Journal of Machine Learning Research* **2** (2001) 243–264.

[10] G. Fung and O. L. Mangasarian, Proximal Support Vector Machine Classifiers, in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2001) 77–86.

[11] A. Golbraikh and A. Tropsha, Beware of q2!, *Journal of Molecular Graphics and Modelling* **20** (2002) 269–276.

[12] A.E. Hoerl and R.W. Kennard, Ridge regression: Biased estimation for nonorthogonal problems, *Technometrics* **12**(3) (1970) 55–67.

[13] A. Höskuldsson, PLS Regression Methods, *Journal of Chemometrics* **2** (1998) 211–218.

[14] I. C. F. Ipsen, and C. D. Meyer, The Idea behind Krylov Methods, *American Mathematical Monthly* **105** (1998) 889–899.

[15] R. H. Kewley, and M. J. Embrechts, Data Strip Mining for the Virtual Design of Pharmaceuticals with Neural Networks, *IEEE Transactions on Neural Networks* 11(3) (2000) 668–679.

[16] Y.-J. Lee and O. L. Mangasarian, RSVM: Reduced Support Vector Machine Classifiers, in *CD Proceedings of the SIAM International Conference on Data Mining*, SIAM, Philadelphia (2001).

[17] K-M Ling and C-J Lin, A study on Reduced Support Vector Machines, Technical Report, Dep. of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan (2002).

[18] D. G. Luenberger, *Linear and Nonlinear Programming*, Addison-Wesley, Reading, MA (1989).

[19] M. F. Möller, A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning, *Neural Networks* **6** (1993) 525–534.

[20] P. M. Murphy and D. W. Ahu, UCI Repository of Machine Learning Databases, www.ics.uci.edu/∼mlearn/MLRepository.html (2002).

[21] W. J. Nash, T. L. Sellers, S. R. Talbot, A. J. Cawthorn and W. B. Ford, The Population Biology of Abalone (Haliotis species) in Tasmania. I. Blacklip Abalone (H. rubra) from the North Coast and Islands of Bass Strait, Sea Fisheries Division, Technical Report No. 48 (ISSN 1034-3288) (1994).

[22] R. Rosipal and L.J. Trejo, Kernel Partial Least Squares Regression in Reproducing Kernel Hilbert Space, *Journal of Machine Learning Research* **2** (2001) 97–123.

[23] B. Schölkopf, A. Smola, and K.-R. Müller, Nonlinear Component Analysis as a Kernel Eigenvalue Problem, *Neural Computation* **10** (1998) 1299–1319.

[24] B. Schölkopf, S. Mika, C.J.C. Burges, P. Knirsch, K.-R. Müller, G. Raetsch and A.J. Smola, Input space versus feature space in kernel-based methods, *IEEE Transactions On Neural Networks* **10**(5) (1999) 1000–1017.

[25] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press (2001).

[26] J. A. K. Suykens and J. Vandewalle, Least Squares Support Vector Machine Classifiers, *Neural Processing letters* **9**(3) (1999) 293–300.

[27] A. J. Smola and B. Schölkopf, Sparse Greedy Matrix Approximation for Machine Learning, *Proceedings of the Seventeenth Intenational Conference on Machine Learning*, Morgan Kaufman, Stanford (2000) 911–918.

[28] J. A. Swets, R. M. Dawes, and J. Monahan, Better Decisions through Science, *Scientific American* (2002) 82–87.

[29] V. Tresp and A. Schwaighofer, Scalable kernel systems, In *International Conference on Artificial Neural Networks* (2001).

[30] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, New York (1995).

[31] H. Wold, Estimation of principal components and related models by iterative least squares, In *Multivariate Analysis*, Academic Press, NY (1966) 391–420.

[32] H. Wold, Soft Modeling, The Basic Design and Some Extensions, In *Systems under Direct Observation Causality-Structure Prediction*, North Holland, Amsterdam (1981).

[33] H. Wold in *Food Research and Drug Analysis*, Applied Science Publishers (1983).

[34] S. Wold, Cross-Validatory Estimation of the Number of Components in Factor and Principal Component Models, *Technometrics* **20**(4) (1987) 397–405.

[35] S. Wold, N. Kettanch-Wold, and B. Skegerberg, Non-Linear PLS Modelling, *Chemometrics and Intelligent Laboratory Systems* **7** (1989) 53–65.

[36] S. Wold, Non-Linear Partial Least Squares Modelling II: Spline Inner Function, *Chemometrics and Intelligent Laboratory Systems* **14** (1992) 71–84.

[37] S. Wold, PLS-Regression: a Basic Tool of Chemometrics, *Chemometrics and Intelligent Laboratory Systems* **58** (2001) 109–130.

[38] W. Wu, D. L. Massarat and S. de Jong, The Kernel PCA Algorithm for Wide Data. Part I: Theory and Algorithms, *Chemometrics and Intelligent Laboratory Systems* **36** (1977) 165–172.

[39] W. Wu, D. L. Massarat and S. de Jong, The Kernel PCA Algorithm for Wide Data. Part II: Fast Cross-Validation and Application in Classification of NIR Data, *Chemometrics and Intelligent Laboratory Systems* **37** (1977) 271–280.

Chapter 12

Multiclass Learning with Output Codes

Yoram Singer[1]

Abstract. In this chapter we review a technique for solving multiclass categorization problems based on output codes. The first part describes an algorithmic framework with accompanying analysis for using output codes with margin classifiers. The second part gives an overview of methods for designing and improving output codes.

[1]This overview is solely based on numerous papers written in collaboration with Koby Crammer, Rob Schapire, and Ofer Dekel. I am in debt to Koby and Rob for the wonderful time we have spent working on learning with output codes. I would also like to deeply thank Ofer Dekel with whom I have been working recently on on multiclass learning by embeddings.

12.1 Introduction

Many supervised machine learning tasks can be cast as the problem of assigning elements to a finite set of classes or categories. For example, the goal of optical character recognition (OCR) systems is to determine the digit value $(0, \ldots, 9)$ from its image. The number of applications that include a multiclass categorization ingredient is immense. A few examples for such applications are text and speech categorization, natural language processing tasks such as part-of-speech tagging, and gesture and object recognition in machine vision.

In designing machine learning algorithms, it is often easier first to devise algorithms for distinguishing between only two classes. Some machine learning algorithms, such as C4.5 [24] and CART [2], can then be naturally extended to handle the multiclass case. For other algorithms, such as AdaBoost [15, 27] and the support vector machines (SVM) algorithm [29, 6], a direct extension to the multiclass case may be problematic. Typically, in such cases, the multiclass problem is reduced to multiple binary classification problems that can be solved separately. Connectionist models [26], in which each class is represented by an output neuron, are a notable example; each output neuron serves as a discriminator between the class it represents and all of the other classes. Thus, this training algorithm is based on a reduction of the multiclass problem to k binary problems, where k is the number of classes.

There are many ways to reduce a multiclass problem to multiple binary classification problems. In the simple approach mentioned above, each class is compared to all others. Hastie and Tibshirani [18] suggest a different approach in which all pairs of classes are compared to each another. Dietterich and Bakiri [13] presented a general framework in which the classes are partitioned into opposing subsets using error-correcting codes. For all of these methods, after the binary classification problems have been solved, the resulting set of binary classifiers must then be combined in some way. In this chapter, we overview a general framework, that is a simple extension of Dietterich and Bakiri's framework, that unifies all of these methods of reducing a multiclass problem to a binary problem.

In this chapter we review methods for reducing a single multiclass problem to multiple binary problems using a binary learning algorithm. We pay particular attention to the case in which the binary learning algorithm is one that is based on the *margin* of a training example. Roughly speaking, the margin of a training example is a number that is positive if and only if the example is correctly classified by a given classifier and whose magnitude is a measure of confidence in the prediction. Several well known algorithms work directly with margins. For instance, the SVM algorithm [29, 6] attempts to maximize the minimum margin of any training example. There are many more algorithms that attempt to minimize some loss function of the margin. AdaBoost [15, 27] is one example: it can be shown that AdaBoost is a greedy procedure for minimizing an exponential loss function of the margins. In Section 12.2, we catalog many other algorithms that also can be viewed as margin-based learning algorithms, including regression, logistic regression and decision-tree algorithms.

The simplest method of combining the binary classifiers (which is called *Hamming decoding*) ignores the loss function that was used during training as well as the confidences attached to predictions made by the classifier. In Section 12.3, we overview a

general technique for combining classifiers that was suggested in [1] and does not suffer from defects. This method is called *loss-based decoding*.

We next overview some of the theoretical properties of these methods in Section 12.4. In particular, for both of the decoding methods, we overview general bounds on the training error on the multiclass problem in terms of the empirical performance on the individual binary problems. These bounds indicate that loss-based decoding is superior to Hamming decoding. Also, these bounds depend on the manner in which the multiclass problem has been reduced to binary problems. For the one-against-all approach, the bounds are linear in the number of classes, but for a reduction based on random partitions of the classes, the bounds are *independent* of the number of classes. We conclude the section with an overview of a bound on the generalization error of the method when the binary learner is AdaBoost.

The results discussed through Section 12.4 assume the existence of a *predefined* code. To contrast the positive results, we conclude in Section 12.5 in which we overview the results on constructing good output codes. Finally, we conclude in Section 12.6 in which we discuss current research directions and open questions.

The chapter is an overview of known results and thus no proofs are given. Readers who are interested in the proof techniques are welcome to read the original publications [1, 8, 7, 9, 11].

12.2 Margin-based Learning Algorithms

We study methods for handling multiclass problems using a general class of binary algorithms that attempt to minimize a margin-based loss function. In this section, we describe that class of learning algorithms with several examples.

A *binary margin-based learning algorithm* takes as input binary labelled training examples $(x_1, y_1), \ldots, (x_m, y_m)$ where the *instances* x_i belong to some domain $\mathcal{X}$ and the *labels* $y_i \in \{-1, +1\}$. Such a learning algorithm uses the data to generate a real-valued function or *hypothesis* $f : \mathcal{X} \to \mathbb{R}$ where f belongs to some *hypothesis space* $\mathcal{F}$. The *margin* of an example (x, y) with respect to f is $yf(x)$. Note that the margin is positive if and only if the sign of $f(x)$ agrees with y. Thus, if we interpret the sign of $f(x)$ as its prediction on x, then

$$\frac{1}{m}\sum_{i=1}^{m}[\![y_i f(x_i) \leq 0]\!]$$

is exactly the training error of f, where, in this case, we count a zero output ($f(x_i) = 0$) as a mistake. (Here and throughout this chapter, $[\![\pi]\!]$ is 1 if predicate π holds and 0 otherwise.)

Although minimization of the training error may be a worthwhile goal, in its most general form the problem is intractable (see for instance [19]). It is therefore often advantageous to instead minimize some other nonnegative *loss function* of the margin, that is, to minimize

$$\frac{1}{m}\sum_{i=1}^{m} L(y_i f(x_i)) \tag{12.1}$$

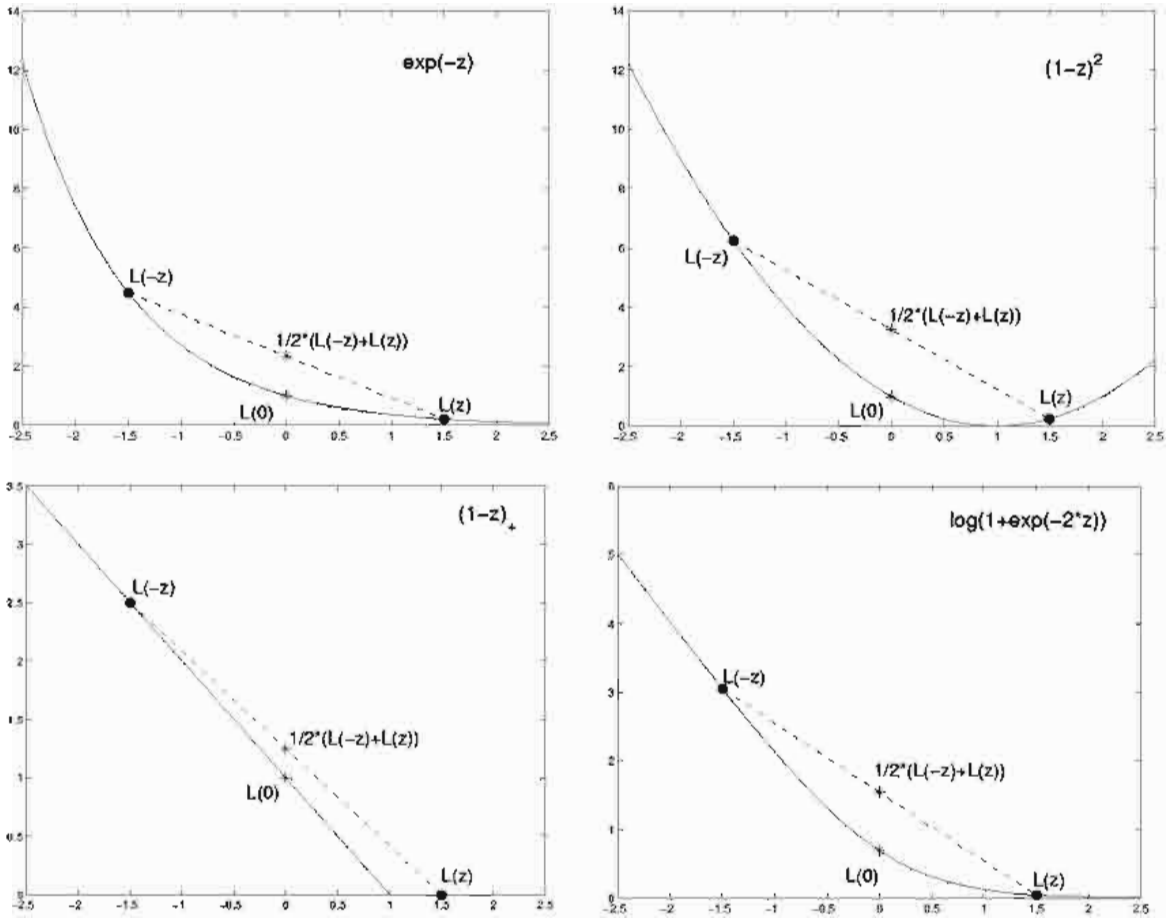


Figure 12.1: Some of the margin-based loss functions described in the chapter: the exponential loss used by AdaBoost, the square loss used in least-squares regression, the "hinge" loss used by support vector machines, the logistic loss used in logistic regression.

for some loss function $L : \mathbb{R} \to [0, \infty)$. Different choices of the loss function L and different algorithms for (approximately) minimizing (12.1) over some hypothesis space lead to various well-studied learning algorithms. Below we list several examples. The work of Allwein et al. [1] was general and applicable to any learning algorithm that can be used for minimizing a margin-based loss function. Thus the focus of that work was on the loss function itself whose properties yields a theorem on the effectiveness of output coding methods for multiclass problems.

Support Vector Machines. For training data that may not be linearly separable, the support vector machines (SVM) algorithm [29, 6] seeks a linear classifier $f : \mathbb{R}^n \to \mathbb{R}$ of the form $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$ that minimizes the objective function

$$\frac{1}{2}\|\mathbf{w}\|_2^2 + C \sum_{i=1}^{m} \xi_i \ ,$$

for some parameter C, subject to the linear constraints

$$y_i((x_i \cdot \mathbf{w}) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \ .$$

Put another way, the SVM solution for $\mathbf{w}$ is the minimizer of the regularized empirical loss function

$$\frac{1}{2}||\mathbf{w}||_2^2 + C\sum_{i=1}^{m}(1 - y_i((\mathbf{w}\cdot x_i) + b))_+ ,$$

where $(z)_+ = \max\{z, 0\}$. (For a more formal treatment see for instance [28].) Although the role of the L_2 norm of $\mathbf{w}$ in the objective function is fundamental in order for SVM to work, the analysis presented in the next section (and the corresponding multiclass algorithm) depends only on the loss function (which is a function of the margins). Thus, SVM can be viewed here as a binary margin-based learning algorithm which seeks to achieve small empirical risk for the loss function $L(z) = (1 - z)_+$.

AdaBoost. The algorithm AdaBoost [15, 27] builds a hypothesis f that is a linear combination of *weak* or *base hypotheses* h_t:

$$f(x) = \sum_t \alpha_t h_t(x).$$

The hypothesis f is built up in a series of rounds on each of which an h_t is selected by a *weak* or *base learning algorithm* and $\alpha_t \in \mathbb{R}$ is then chosen. It has been observed by Breiman [3, 4] and other authors [16, 22, 25, 27] that the method in which the h_t's and α_t's are chosen has the effect of greedily minimizing

$$\frac{1}{m}\sum_{i=1}^{m} e^{-y_i f(x_i)}.$$

Thus, AdaBoost is a binary margin-based learning algorithm in which the loss function is $L(z) = e^{-z}$.

AdaBoost with randomized predictions. In a little studied variant of AdaBoost [15], AdaBoost was allowed to output randomized predictions in which the predicted label of a new example x is chosen randomly to be $+1$ with probability $1/(1 + e^{-2f(x)})$. The loss suffered then is the probability that the randomly chosen predicted label disagrees with the correct label y. Let $p(x) \stackrel{\text{def}}{=} 1/(1+e^{-2f(x)})$. Then the loss is $p(x)$ if $y = -1$ and $1 - p(x)$ if $y = +1$. Using a simple algebraic manipulation, the loss can be shown to be $1/(1 + e^{2yf(x)})$. So for this variant of AdaBoost, the loss $L(z)$ is set to $1/(1 + e^{2z})$. However, in this case, note that the learning algorithm is not directly attempting to minimize this loss (it is instead minimizing the exponential loss described above).

Regression. There are various algorithms, such as neural networks and least squares regression, that attempt to minimize the squared error loss function $(y - f(x))^2$. When the y's are in $\{-1, +1\}$, this function can be rewritten as

$$\begin{aligned}(y - f(x))^2 &= y^2(y - f(x))^2 \\ &= (yy - yf(x))^2 \\ &= (1 - yf(x))^2.\end{aligned}$$

Thus, for binary problems, minimizing squared error fits the framework where $L(z) = (1-z)^2$.

Logistic regression. In logistic regression and related methods such as Iterative Scaling [10, 23, 21], and LogitBoost [16], one posits a logistic model for estimating the conditional probability of a positive label:

$$\Pr[y = +1|x] = \frac{1}{1+e^{-2f(x)}}.$$

One then attempts to maximize the likelihood of the labels in the sample, or equivalently, to minimize the log loss

$$-\log(\Pr[y|x]) = \log(1+e^{-2yf(x)}).$$

Thus, for logistic regression and related methods, the loss $L(z)$ is set to $\log(1+e^{-2z})$.

Decision trees. The most popular decision tree algorithms can also be naturally linked to loss functions. For instance, Quinlan's C4.5 [24], in its simplest form, for binary classification problems, splits decision nodes in a manner to greedily minimize

$$\sum_{\text{leaf}\, j} \left(p_j^+ \ln\left(\frac{p_j^- + p_j^+}{p_j^+}\right) + p_j^- \ln\left(\frac{p_j^- + p_j^+}{p_j^-}\right)\right) \tag{12.2}$$

where p_j^+ and p_j^- are the fraction of positive and negative examples reaching leaf j, respectively. The prediction at leaf j is then $\text{sign}(p_j^+ - p_j^-)$. Viewed differently, imagine a decision tree that instead outputs a real number f_j at each leaf with the intention of performing logistic regression as above. Then the empirical loss associated with logistic regression is

$$\sum_{\text{leaf}\, j} \left(p_j^+ \ln(1+e^{-2f_j}) + p_j^- \ln(1+e^{2f_j})\right).$$

This is minimized, over choices of f_j, when $f_j = (1/2)\ln(p_j^+/p_j^-)$. Plugging in this choice gives exactly (12.2), and thresholding f_j gives the hard prediction rule used earlier. Thus, C4.5, in this simple form, can be viewed as a margin-based learning algorithm that is naturally linked to the loss function used in logistic regression.

By similar reasoning, CART [2], which splits using the Gini index, can be linked to the square loss function, while Kearns and Mansour's [20] splitting rule can be linked to the exponential loss used by AdaBoost.

The analysis reviewed in the next section might also hold for other algorithms that tacitly employ a function of the margin. For instance, Freund's BrownBoost algorithm [14] implicitly uses an instance potential function that satisfies the condition we impose on L. Therefore, it can also be combined with output coding and used to solve multiclass problems. To conclude this section, Figure 12.1 shows some of the loss functions discussed above.

12.3 Output Coding for Multiclass Problems

In the last section, we discussed margin-based algorithms for learning binary problems. Suppose now that we are faced with a multiclass learning problem in which each label y is chosen from a set $\mathcal{Y}$ of cardinality $k > 2$. How can a binary margin-based learning algorithm be modified to handle a k-class problem?

Several solutions have been proposed for this question. Many involve reducing the multiclass problem, in one way or another, to a set of binary problems. For instance, perhaps the simplest approach is to create one binary problem for each of the k classes. That is, for each $r \in \mathcal{Y}$, we apply the given margin-based learning algorithm to a binary problem in which all examples labelled $y = r$ are considered positive examples and all other examples are considered negative examples. We then end up with k hypotheses that somehow must be combined. We call this the *one-against-all* approach.

Another approach, suggested by Hastie and Tibshirani [18], is to use the given binary learning algorithm to distinguish each pair of classes. Thus, for each distinct pair $r_1, r_2 \in \mathcal{Y}$, we run the learning algorithm on a binary problem in which examples labelled $y = r_1$ are considered positive, and those labelled $y = r_2$ are negative. All other examples are simply ignored. Again, the $\binom{k}{2}$ hypotheses that are generated by this process must then be combined. We call this the *all-pairs* approach.

A more general suggestion on handling multiclass problems was given by Dietterich and Bakiri [13]. Their idea is to associate each class $r \in \mathcal{Y}$ with a row of a "coding matrix" $\mathbf{M} \in \{-1, +1\}^{k \times \ell}$ for some ℓ. The binary learning algorithm is then run once for each column of the matrix on the induced binary problem in which the label of each example labelled y is mapped to $M(y, s)$. This yields ℓ hypotheses f_s. Given an example x, we then predict the label y for which row y of matrix $\mathbf{M}$ is "closest" to $(f_1(x), \ldots, f_\ell(x))$. This is the method of *error correcting output codes* (ECOC).

In this section, we review a unifying generalization of all three of these methods applicable to any margin-based learning algorithm. This generalization is closest to the ECOC approach of Dietterich and Bakiri [13] but differs in that the coding matrix is taken from the larger set $\{-1, 0, +1\}^{k \times \ell}$. That is, some of the entries $M(r, s)$ may be zero, indicating that we don't care how hypothesis f_s categorizes examples with label r.

Thus, the scheme for learning multiclass problems using a binary margin-based learning algorithm $\mathcal{A}$ works as follows. We begin with a given *coding matrix*

$$\mathbf{M} \in \{-1, 0, +1\}^{k \times \ell} .$$

For $s = 1, \ldots, \ell$, the learning algorithm $\mathcal{A}$ is provided with labelled data of the form $(x_i, M(y_i, s))$ for all examples i in the training set but omitting all examples for which $M(y_i, s) = 0$. The algorithm $\mathcal{A}$ uses this data to generate a hypothesis $f_s : \mathcal{X} \to \mathbb{R}$.

For example, for the one-against-all approach, $\mathbf{M}$ is a $k \times k$ matrix in which all diagonal elements are $+1$ and all other elements are -1. For the all-pairs approach, $\mathbf{M}$ is a $k \times \binom{k}{2}$ matrix in which each column corresponds to a distinct pair (r_1, r_2). For this column, $\mathbf{M}$ has $+1$ in row r_1, -1 in row r_2 and zeros in all other rows.

As an alternative to calling $\mathcal{A}$ repeatedly, in some cases, we may instead wish to add the column index s as a distinguished attribute of the instances received by $\mathcal{A}$, and then learn a *single* hypothesis on this larger learning problem rather than ℓ hypotheses

on smaller problems. That is, we provide $\mathcal{A}$ with instances of the form $((x_i, s), M(y_i, s))$ for all training examples i and all columns s for which $M(y_i, s) \neq 0$. Algorithm $\mathcal{A}$ then produces a *single* hypothesis $f : \mathcal{X} \times \{1, \ldots, \ell\} \to \mathbb{R}$. However, for consistency with the preceding approach, we define $f_s(x)$ to be $f(x, s)$. We call these two approaches in which $\mathcal{A}$ is called repeatedly or only once the *multi-call* and *single-call* variants, respectively.

We note in passing that there are no fundamental differences between the single and multi-call variants. Most previous work on output coding employed the multi-call variant due to its simplicity. The single-call variant becomes handy when an implementation of a classification learning algorithm that outputs a single hypothesis of the form $f : \mathcal{X} \times \{1, \ldots, \ell\} \to \mathbb{R}$ is available.

For either variant, the algorithm $\mathcal{A}$ attempts to minimize the loss L on the induced binary problem(s). Recall that L is a function of the margin of an example so the loss of f_s on an example x_i with induced label $M(y_i, s) \in \{-1, +1\}$ is $L(M(y_i, s)\ f_s(x_i))$. When $M(y_i, s) = 0$, we want to entirely ignore the hypothesis f_s in computing the loss. We can define the loss to be any constant in this case, so, for convenience, we choose the loss to be $L(0)$ so that the loss associated with f_s on example i is $L(M(y_i, s)\ f_s(x_i))$ in all cases.

Thus, the average loss over all choices of s and all examples i is

$$\frac{1}{m\ell} \sum_{i=1}^{m} \sum_{s=1}^{\ell} L(M(y_i, s)\ f_s(x_i)). \tag{12.3}$$

We call this the *average binary loss* of the hypotheses f_s on the given training set with respect to coding matrix $\mathbf{M}$ and loss L. It is the quantity that the calls to $\mathcal{A}$ have the implicit intention of minimizing. We will see in the next section how this quantity relates to the misclassification error of the final classifier that we build on the original multiclass training set.

Let $\mathbf{M}(r)$ denote row r of M and let $\mathbf{f}(x)$ be the vector of predictions on an instance x:

$$\mathbf{f}(x) = (f_1(x), \ldots, f_\ell(x)).$$

Given the predictions of the f_s's on a test point x, which of the k labels in $\mathcal{Y}$ should be predicted? While several methods of combining the f_s's can be devised, the work of Allwein et al. [1] focuses on two that are very simple to implement and for which it possible to analyze the empirical risk of the original multiclass problem. The basic idea of both methods is to predict with the label r whose row $\mathbf{M}(r)$ is "closest" to the predictions $\mathbf{f}(x)$. In other words, predict the label r that minimizes $d(\mathbf{M}(r), \mathbf{f}(x))$ for some distance d. This formulation begs the question, however, of how we measure distance between the two vectors.

One way of doing this is to count up the number of positions s in which the sign of the prediction $f_s(x)$ differs from the matrix entry $M(r, s)$. Formally, this means that the distance measure is

$$d_H(\mathbf{M}(r), \mathbf{f}(x)) = \sum_{s=1}^{\ell} \left(\frac{1 - \text{sign}(M(r, s) f_s(x))}{2} \right) \tag{12.4}$$

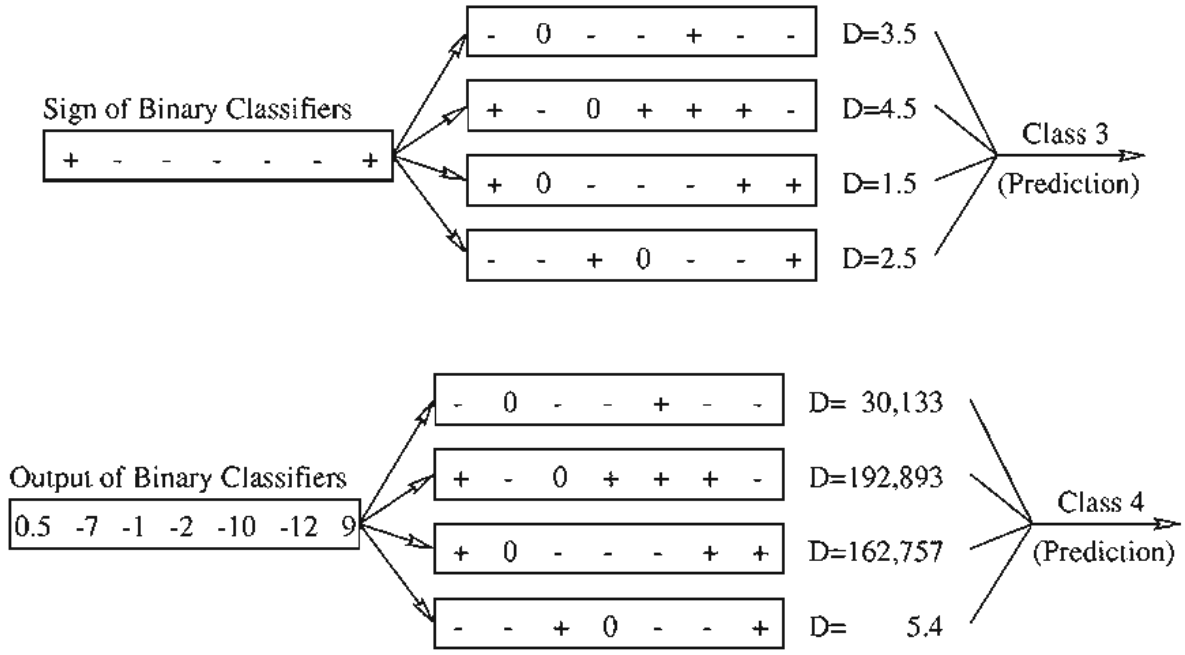


Figure 12.2: An illustration of the multiclass prediction procedure for Hamming decoding (top) and loss-based decoding (bottom) for a 4-class problem using a code of length 7. The exponential function was used for the loss-based decoding.

where $\text{sign}(z)$ is $+1$ if $z > 0$, -1 if $z < 0$, and 0 if $z = 0$. This is essentially like computing Hamming distance between row $\mathbf{M}(r)$ and the signs of the $f_s(x)$'s. However, note that if either $M(r,s)$ or $f_s(x)$ is zero then that component contributes $1/2$ to the sum. For an instance x and a matrix M, the predicted label $\widehat{y} \in \{1,\ldots,k\}$ is therefore

$$\widehat{y} = \arg\min_r d_H(\mathbf{M}(r), \mathbf{f}(x)) \ .$$

We call this method of combining the f_s's *Hamming decoding*.

A disadvantage of this method is that it ignores entirely the magnitude of the predictions which can often be an indication of a level of "confidence." Our second method for combining predictions takes this potentially useful information into account, as well as the relevant loss function L which is ignored with Hamming decoding. The idea is to choose the label r that is most consistent with the predictions $f_s(x)$ in the sense that, if example x were labelled r, the total loss on example (x,r) would be minimized over choices of $r \in \mathcal{Y}$. Formally, this means that the distance measure is the total loss on a proposed example (x,r):

$$d_L(\mathbf{M}(r), \mathbf{f}(x)) = \sum_{s=1}^{\ell} L(M(r,s) f_s(x)) \ . \tag{12.5}$$

Analogous to Hamming decoding, the predicted label $\widehat{y} \in \{1,\ldots,k\}$ is

$$\widehat{y} = \arg\min_r d_L(\mathbf{M}(r), \mathbf{f}(x)) \ .$$

We call this approach *loss-based decoding*. An illustration of the two decoding methods is given in Figure 12.2. The figure shows the decoding process for a problem with 4 classes using an output code of length $\ell = 7$. For clarity we denote in the figure the entries of the output code matrix by $+$, $-$ and 0 (instead of $+1$, -1 and 0). Note that in the example, the predicted class of the loss-based decoding (which, in this case, uses exponential loss) is different than that of the Hamming decoding.

12.4 Training Error Bounds

In this section, we review training error bounds for the output coding methods described in the last section. Specifically, we give the bound the training error of the two decoding methods in terms of the average binary loss as defined in (12.3), as well as a measure of the minimum distance between any pair of rows of the coding matrix. Here, we use a simple generalization of the Hamming distance for vectors over the set $\{-1, 0, +1\}$. Specifically, the distance between two rows $\mathbf{u}, \mathbf{v} \in \{-1, 0, +1\}^\ell$ is defined to be

$$\begin{aligned}
\Delta(\mathbf{u}, \mathbf{v}) &= \sum_{s=1}^{\ell} \begin{cases} 0 & \text{if } u_s = v_s \wedge u_s \neq 0 \wedge v_s \neq 0 \\ 1 & \text{if } u_s \neq v_s \wedge u_s \neq 0 \wedge v_s \neq 0 \\ 1/2 & \text{if } u_s = 0 \vee v_s = 0 \end{cases} \\
&= \sum_{s=1}^{\ell} \frac{1 - u_s v_s}{2} \\
&= \frac{\ell - \mathbf{u} \cdot \mathbf{v}}{2}.
\end{aligned}$$

Our analysis then depends on the minimum distance ρ between pairs of distinct rows:

$$\rho = \min\{\Delta(\mathbf{M}(r_1), \mathbf{M}(r_2)) : r_1 \neq r_2\}. \tag{12.6}$$

For example, for the one-against-all code, $\rho = 2$. For the all-pairs code, $\rho = \binom{k}{2} - 1)/2 + 1$, since every two rows r_1, r_2 have exactly one component with opposite signs ($\mathbf{M}(r_1, s) = -\mathbf{M}(r_2, s)$ and $\mathbf{M}(r_1, s) \neq 0$) and for the rest at least one component of the two is 0 ($\mathbf{M}(r_1, s) = 0$ or $\mathbf{M}(r_2, s) = 0$). For a random matrix with components chosen uniformly over either $\{-1, +1\}$ or $\{-1, 0, +1\}$, the *expected* value of $\Delta(\mathbf{M}(r_1), \mathbf{M}(r_2))$ for any distinct pair of rows is exactly $\ell/2$.

Intuitively, the larger ρ, the more likely it is that decoding will "correct" for errors made by individual hypotheses. This was Dietterich and Bakiri's [13] insight in suggesting the use of output codes with error-correcting properties. This intuition is reflected in the analysis in which a larger value of ρ gives a better upper bound on the training error. In particular, Theorem 1 states that the training error is at most ℓ/ρ times worse than the average binary loss of the combined hypotheses (after scaling the loss by $L(0)$). For the one-against-all matrix, $\ell/\rho = \ell/2 = k/2$ which can be large if the number of classes is large. On the other hand, for the all-pairs matrix or for a random matrix, ℓ/ρ is close to the constant 2, independent of k.

We overview first the analysis of loss-based decoding. An analysis of Hamming decoding will follow as a corollary. Concerning the loss L, the analysis assumes only

that

$$\frac{L(z) + L(-z)}{2} \geq L(0) > 0 \tag{12.7}$$

for all $z \in \mathbb{R}$. Note that this property holds if L is convex, although convexity is by no means a necessary condition. Note also that all of the loss functions in Section 12.2 satisfy this property. The property is illustrated in Figure 12.1 for four of the loss functions discussed in that section.

Theorem 1 *Let ε be the average binary loss (as defined in (12.3)) of hypotheses $f_1, \ldots, f_\ell$ on a given training set $(x_1, y_1), \ldots, (x_m, y_m)$ with respect to the coding matrix $\mathbf{M} \in \{-1, 0, +1\}^{k \times \ell}$ and loss L, where k is the cardinality of the label set $\mathcal{Y}$. Let ρ be as in (12.6). Assume that L satisfies (12.7) for all $z \in \mathbb{R}$. Then the training error using loss-based decoding is at most*

$$\frac{\ell\varepsilon}{\rho L(0)}.$$

As a corollary, we a similar but weaker theorem for Hamming decoding holds. Note that we use a different assumption about the loss function L, but one that also holds for all of the loss functions described in Section 12.2.

Corollary 1 *Let $f_1, \ldots, f_\ell$ be a set of hypotheses on a training set $(x_1, y_1), \ldots, (x_m, y_m)$, and let $\mathbf{M} \in \{-1, 0, +1\}^{k \times \ell}$ be a coding matrix where k is the cardinality of the label set $\mathcal{Y}$. Let ρ be as in (12.6). Then the training error using Hamming decoding is at most*

$$\frac{1}{\rho m} \sum_{i=1}^{m} \sum_{s=1}^{\ell} \left(1 - \operatorname{sign}(M(y_i, s) f_s(x_i))\right). \tag{12.8}$$

Moreover, if L is a loss function satisfying $L(z) \geq L(0) > 0$ for $z < 0$ and ε is the average binary loss with respect to this loss function, then the training error using Hamming decoding is at most

$$\frac{2\ell\varepsilon}{\rho L(0)}. \tag{12.9}$$

Theorem 1 and Corollary 1 are broad generalizations of similar results proved by Schapire and Singer [27] in a much more specialized setting involving only AdaBoost. Also, Corollary 1 generalizes some of the results of Guruswami and Sahai [17] that bound the multiclass training error in terms of the training (misclassification) error rates of the binary classifiers.

The bounds of Theorem 1 and Corollary 1 depend implicitly on the fraction of zero entries in the matrix. Intuitively, the more zeros there are, the more examples that are ignored and the harder it should be to drive down the training error. At an extreme, if $\mathbf{M}$ is all zeros, then ρ is fairly large ($\ell/2$) but learning certainly should not be possible. To make this dependence explicit, let

$$T = \{(i, s) : M(y_i, s) = 0\}$$

be the set of pairs i, s inducing examples that are ignored during learning. Let $q = |T|/(m\ell)$ be the fraction of ignored pairs. Let ε be the average binary loss *restricted* to the pairs not ignored during training:

$$\varepsilon = \frac{1}{|T^c|} \sum_{(i,s)\in T^c} L(M(y_i, s) f_s(x_i))$$

where $T^c = \{(i, s) : M(y_i, s) \neq 0\}$. Then the bound in Theorem 1 can be rewritten

$$\begin{aligned} &\frac{\ell}{\rho L(0)} \frac{1}{m\ell} \left(\sum_{(i,s)\in T} L(0) + \sum_{(i,s)\notin T} L(M(y_i, s) f_s(x_i)) \right) \\ &= \frac{\ell}{\rho} \left(q + (1-q) \frac{\varepsilon}{L(0)} \right). \end{aligned}$$

Similarly, let ϵ be the fraction of misclassification errors made on T^c:

$$\epsilon = \frac{1}{|T^c|} \sum_{(i,s)\in T^c} [\![M(y_i, s) \neq \mathrm{sign}(f_s(x_i))]\!].$$

The first part of Corollary 1 implies that the training error using Hamming decoding is upper bounded by

$$\frac{\ell}{\rho}(q + 2(1-q)\epsilon).$$

We see from these bounds that there are many trade-offs in the design of the coding matrix M. On the one hand, we want the rows to be far apart so that ρ will be large, and we also want there to be few non-zero entries so that q will be small. On the other hand, attempting to make ρ large and q small may produce binary problems that are difficult to learn, yielding large (restricted) average binary loss.

12.5 Finding Good Output Codes

The initial work of Allwein et al. was concerned with the formal question of how to use output codes and what are the formal properties of codes in terms of empirical loss and generalization of classification with output codes. A natural question that arise is what code to use for a given classification problem or, alternatively, how to build good general or even task-specific output codes. Crammer and Singer [8] were first to address this issue. They discussed the computational complexity of output code design. Specifically, they showed that prove that given a set of l binary classifiers $\overline{h}(x)$, finding a code matrix which minimizes the empirical loss $\epsilon_S(M, \overline{h})$ is NP-complete. We now review this result.

Given a sample $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ and a set of classifiers $\overline{h}$, let us denote by $\widetilde{S}$ the evaluation of $\overline{h}(\cdot)$ on the sample S, that is $\widetilde{S} = \{(\overline{h}_1, y_1), \ldots, (\overline{h}_m, y_m)\}$, where $\overline{h}_i \stackrel{\text{def}}{=} \overline{h}(x_i)$. We now show that even when $k = 2$ and $K(\overline{u}, \overline{v}) = \overline{u} \cdot \overline{v}$ the problem is NP-complete. (Clearly, the problem remains NPC for $k > 2$). Following the notation of previous sections, the output code matrix is composed of two rows $\overline{M}_1$ and $\overline{M}_2$ and

the predicted class for instance x_i is $H(x_i) = \arg\max_{r=1,2}\{\overline{M}_r \cdot \overline{h}_i\}$. For the simplicity, we assume that both the code M and the hypotheses' values $\overline{h}_i$ are over the set $\{0,1\}$ (instead of $\{-1,+1\}$). This assumption does not change the problem as it possible to show that the same proof technique can be used with hypotheses whose outputs are in $\{\pm 1\}$.

Theorem 2 *The following decision problem is NP-complete.*
`Input`: *A natural number q, a labelled sample $\widetilde{S} = \{(\overline{h}_1, y_1), \ldots, (\overline{h}_m, y_m)\}$, where $y_i \in \{1,2\}$, and $\overline{h}_i \in \{0,1\}^l$.*
`Question`: *Does there exist a matrix $M \in \{0,1\}^{2\times l}$ such that the classifier $H(x)$ based on an output code M makes at most q mistakes on $\widetilde{S}$.*

The intractability result above raised triggered research on alternatives to discrete output codes. In [8], Crammer and Singer also described the notion of output codes. This construction was also used in [11] to devise a scheme that learns a good code in tandem to the construction of the binary classifiers. The paper's starting point is the inherent decoupling of the learning process from the class representation problem employed by ECOC. This decoupling is both a blessing and a curse. On one hand it offers great flexibility and modularity, on the other hand, the resulting binary learning problems might be unnatural and therefore potentially difficult. In [11] an alternative approach was described and analyze. This work ties the learning problem with the class representation problem. To sidestep the intractability barrier, the scheme of Dekel and Singer perceives the set of binary classifiers as an embedding of the instance space and the code matrix as an embedding of the label set into a common space. In this common space each instance is assigned the label from which its divergence is smallest. To construct these embeddings, the notion of probabilistic output codes is introduced. Then, an algorithm that constructs the label and instance embeddings such that the resulting classifier achieves a small empirical error is provided. The result is a paradigm that includes ECOC as a special case.

The algorithm, termed Bunching, alternates between two steps. One step improves the embedding of the instance space into the common space while keeping the embedding of the label set fixed. This step is analogous to the learning stage of the ECOC technique, where a set of binary classifiers are learned with respect to a predefined code. The second step complements the first by updating the label embedding while keeping the instance embedding fixed. The two alternating steps resemble the steps performed by the EM algorithm [12] and by Alternating Minimization [10]. The techniques we use in the design and analysis of the Bunching algorithm also build on recent results in classification learning using Bregman divergences [21, 5].

12.6 Conclusions

Error correcting output codes (ECOC) provide a simple and powerful framework for solving multiclass prediction problems using binary classification learning algorithms. The formal and experimental results in the past five years spurred quite a few open problems and directions for future research. First, we still lack a more complete theory of generalization error for ECOC. A second important issue is whether the decoding

method and the error bounds can be improved for the case of highly correlated binary classifiers. Output codes may also be used in other complex decision problems. In particular, it might be possible to cast the problem of hierarchical classification as a special instance of output codes with tree metrics. Last, but not least, ECOC uses elementary constructions from coding theory. Problems in source and channel coding has been the focus of many researchers for over 50 years. We believe that both learning theory and applications in machine learning can greatly benefit by further exploitation of analysis tools and algorithms in coding theory.

Bibliography

[1] E.L. Allwein, R.E. Schapire, and Y. Singer, Reducing multiclass to binary: A unifying approach for margin classifiers, *Journal of Machine Learning Research* 1 (2000) 113–141.

[2] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, Wadsworth & Brooks (1984).

[3] L. Breiman, Arcing the edge, Technical Report 486, Statistics Department, University of California at Berkeley (1997).

[4] L. Breiman, Prediction games and arcing classifiers, Technical Report 504, Statistics Department, University of California at Berkeley (1997).

[5] M. Collins, R. E. Schapire, and Y. Singer, Logistic regression, adaboost and bregman distances, *Machine Learning* **47** (2002) 253–285.

[6] C. Cortes and V. Vapnik, Support-vector networks, *Machine Learning* **20**(3) (1995) 273–297.

[7] K. Crammer and Y. Singer, Improved output coding for classification using continuous relaxation, In *Advances in Neural Information Processing Systems 13* (2000).

[8] K. Crammer and Y. Singer, On the learnability and design of output codes for multiclass problems, In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory* (2000).

[9] K. Crammer and Y. Singer, Ultraconservative online algorithms for multiclass problems, In *Proceedings of the Fourteenth Annual Conference on Computational Learning Theory* (2001).

[10] I. Csiszár and G. Tusnády, Information geometry and alternating minimization procedures, *Statistics and Decisions, Supplement Issue* **1** (1984) 205–237.

[11] O. Dekel and Y. Singer, Multiclass learning by probabilistic embeddings, In *Advances in Neural Information Processing Systems 15* (2002).

[12] A.P. Dempster, N.M. Laird, and D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society, Ser. B* **39** (1977) 1–38.

[13] T. G. Dietterich and G. Bakiri, Solving multiclass learning problems via error-correcting output codes, *Journal of Artificial Intelligence Research* **2** (1995) 263–286.

[14] Y. Freund, An adaptive version of the boost by majority algorithm, In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory* (1999).

[15] Y. Freund and R. E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, *Journal of Computer and System Sciences* **55**(1) (1997) 119–139.

[16] J. Friedman, T. Hastie, and R. Tibshirani, Additive logistic regression: a statistical view of boosting, *Annals of Statistics* **28**(2) (2000) 337–374.

[17] V. Guruswami and A. Sahai, Multiclass learning, boosting, and error-correcting codes, In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory* (1999).

[18] T. Hastie and R. Tibshirani, Classification by pairwise coupling, *The Annals of Statistics* **26**(1) (1998) 451–471.

[19] K.-U. Höffgen and H.-U. Simon, Robust trainability of single neurons, In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, Pittsburgh, Pennsylvania (1992) 428–439.

[20] M. Kearns and Y. Mansour, On the boosting ability of top-down decision tree learning algorithms, In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing* (1996).

[21] J. D. Lafferty, Additive models, boosting and inference for generalized divergences, In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory* (1999).

[22] L. Mason, J. Baxter, P. Bartlett, and M. Frean, Functional gradient techniques for combining hypotheses. In *Advances in Large Margin Classifiers*, MIT Press (1999).

[23] S. Della Pietra, V. Della Pietra, and J. Lafferty, Inducing features of random fields, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **5** (1997) 179–190.

[24] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann (1993).

[25] G. Rätsch, T.Onoda, and K.-R. Müller, Regularizing adaboost, In *Advances in Neural Information Processing Systems 12* (1998).

[26] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation, In David E. Rumelhart and James L. McClelland, editors, *Parallel Distributed Processing – Explorations in the Microstructure of Cognition*, chapter 8, MIT Press (1986) 318–362.

[27] R. E. Schapire and Y. Singer, Improved boosting algorithms using confidence-rated predictions, *Machine Learning* **37**(3) (1999) 1–40.

[28] B. Schölkopf, A. Smola, R. Williamson, and P. Bartlett, New support vector algorithms, Technical Report NC2-TR-1998-053, NeuroColt2 (1998).

[29] V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer (1995).

Chapter 13

Bayesian Regression and Classification

Christopher M. Bishop and Michael E. Tipping

Abstract. In recent years Bayesian methods have become widespread in many domains including computer vision, signal processing, information retrieval and genome data analysis. The availability of fast computers allows the required computations to be performed in reasonable time, and thereby makes the benefits of a Bayesian treatment accessible to an ever broadening range of applications. In this tutorial we give an overview of the Bayesian approach to pattern recognition in the context of simple regression and classification problems. We then describe in detail a specific Bayesian model for regression and classification called the *Relevance Vector Machine*. This overcomes many of the limitations of the widely used Support Vector Machine, while retaining the highly desirable property of sparseness.

13.1 Introduction

Although Bayesian methods have been studied for many years, it is only recently that their practical application has become truly widespread. This is due in large part to the relatively high computational overhead of performing the marginalizations (integrations and summations) which lie at the heart of the Bayesian paradigm. For this reason more traditional approaches, based on point estimation of parameters, have typically been the method of choice. However, the widespread availability of fast computers allows Bayesian computations to be performed in reasonable time for an increasingly wide spectrum of real world applications. Furthermore, the development of Markov chain Monte Carlo techniques, and more recently of deterministic approximation schemes such as variational inference, have greatly extended the range of models amenable to a Bayesian treatment.

13.1.1 Least squares regression

In this tutorial we consider the relatively simple, but widely studied, problems of regression and classification for independent, identically distributed (i.i.d.) data. Consider a data set of examples of input vectors $\{\mathbf{x}_n\}_{n=1}^N$ along with corresponding targets $\mathbf{t} = \{t_n\}_{n=1}^N$. Note that, for notational simplicity, we shall consider a single target variable, but that the extension of the methods discussed in this paper to multiple target variables is straightforward. For regression, we generally assume that the targets are some noisy realization of an underlying functional relationship $y(\mathbf{x})$ that we wish to estimate so that

$$t_n = y(\mathbf{x}_n; \mathbf{w}) + \epsilon_n \tag{13.1}$$

where ϵ is an additive noise process in which the values ϵ_n are i.i.d., and $\mathbf{w}$ is a vector of adjustable parameters or 'weights'.

One interesting class of candidate functions for $y(\mathbf{x}; \mathbf{w})$ is given by

$$y(\mathbf{x}; \mathbf{w}) = \sum_{i=1}^{M} w_i \phi_i(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}), \tag{13.2}$$

which represents a linearly-weighted sum of M nonlinear fixed basis functions denoted by $\boldsymbol{\phi}(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots, \phi_M(\mathbf{x}))^{\mathrm{T}}$. Models the type (13.2) are known as *linear* models since the function $y(\mathbf{x}; \mathbf{w})$ is a linear function of the parameters $\mathbf{w} = (w_1, w_2, \ldots, w_M)^{\mathrm{T}}$. However, in general the function itself is non-linear, and indeed can be very flexible if M is relatively large.

Classical (non-Bayesian) techniques use some form of 'estimator' to determine a specific value for the parameter vector $\mathbf{w}$. One of the simplest examples is the sum-of-squares error function defined by

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} |y(\mathbf{x}_n; \mathbf{w}) - t_n|^2 \tag{13.3}$$

where the factor of $1/2$ is included for later convenience. Minimizing this error function with respect to $\mathbf{w}$ leads to an estimate $\mathbf{w}^*$ which can be used to make predictions for new values of $\mathbf{x}$ by evaluating $y(\mathbf{x}; \mathbf{w}^*)$.

In the case of classification problems, the function $y(\mathbf{x}; \mathbf{w})$ is transformed using an appropriate non-linearity, such as a logistic sigmoid for 2-class problems or a softmax (normalized exponential) for multi-class problems. The corresponding error function is given by the cross-entropy [3].

A well-known problem with error function minimization is that complex and flexible models can 'over-fit' the training data, leading to poor generalization. Indeed, when the number of parameters equals the number of data points, the least squares solution for a model of the form (13.2) can achieve a perfect fit to the training data while having very poor generalization to new data points. This behaviour is characterized by value of the parameters w_i which have large positive and negative values finely tuned to the individual noisy data points. The corresponding function $y(\mathbf{x}; \mathbf{w})$ typically exhibits strong oscillations as a function of $\mathbf{x}$. Whilst over-fitting can be avoided by limiting the complexity of the model, this too can lead to poor generalization if the model is insufficiently flexible to capture the underlying behaviour of the data set. However, we often have to work with data sets of limited size and yet we wish to be able to use flexible models many adjustable parameters. We shall see that the phenomenon of over-fitting is a pathological property of point estimation, and that by adopting a Bayesian viewpoint we can apply complex models to small data sets without encountering problems of over-fitting.

13.1.2 Regularization

One classical (non-Bayesian) technique for reducing over-fitting is that of regularization in which a penalty term $\Omega(\mathbf{w})$ is added to the error function to give

$$\widetilde{E}(\mathbf{w}) = E(\mathbf{w}) + \lambda\Omega(\mathbf{w}) \tag{13.4}$$

where $\Omega(\mathbf{w})$ discourages over-fitting, for example by penalizing large values for the weight parameters w_i. The parameter λ controls the trade-off between fitting the data by reducing $E(\mathbf{w})$ and smoothing the function $y(\mathbf{x}; \mathbf{w})$ as a function of $\mathbf{x}$ by reducing $\Omega(\mathbf{w})$. A common choice of regularizer is given by the sum of the squares of the weight parameters, so that

$$\Omega(\mathbf{w}) = \frac{\|\mathbf{w}\|^2}{2}. \tag{13.5}$$

The value of the regularization coefficient is typically set by holding back some data from the training set and optimizing λ by minimizing the value of the un-regularized error function $E(\mathbf{w})$ evaluated with respect to the held out data. For small data sets this procedure may be refined to give the cross-validation technique which makes more efficient use of limited data [3].

13.1.3 Probabilistic models

We can motivate the regularized least-squares framework from a probabilistic viewpoint as follows. The observed target values t_n are assumed to have been generated from the underlying function $y(\mathbf{x}; \mathbf{w})$ by the addition of independent Gaussian noise, so that in

(13.1) the noise values ϵ_n are normally distributed with zero mean and variance σ^2 so that

$$p(\epsilon|\sigma^2) = \mathcal{N}(\epsilon|0,\sigma^2) \tag{13.6}$$

$$= \left(\frac{1}{2\pi\sigma^2}\right)^{1/2} \exp\left\{-\frac{1}{2\sigma^2}\epsilon^2\right\} \tag{13.7}$$

where the notation $\mathcal{N}(\epsilon|\mu,\sigma^2)$ specifies a Gaussian distribution over ϵ with mean μ and variance σ^2. Variables such as μ and σ^2 are sometimes called *hyperparameters* since they control the distribution over parameters. From (13.1) and (13.7) it follows that the conditional distribution of the target variable given the input variable and the weight parameters is again a Gaussian distribution

$$p(t|\mathbf{x},\mathbf{w},\sigma^2) = \left(\frac{1}{2\pi\sigma^2}\right)^{1/2} \exp\left\{-\frac{1}{2\sigma^2}|y(\mathbf{x};\mathbf{w}) - t|^2\right\}. \tag{13.8}$$

Note that the distribution for t is conditioned on the value of $\mathbf{x}$. We are not interested in modelling the distribution of $\mathbf{x}$ and so from now on we shall omit $\mathbf{x}$ from the conditioning list in order to keep the notation compact. Since the data points are independent, the joint probability of the whole data set, given $\mathbf{w}$ and β, is given by the product over all data points of the conditional distribution (13.8) evaluated at the observed data values

$$L(\mathbf{w}) = p(\mathbf{t}|\mathbf{w},\sigma^2) = \left(\frac{1}{2\pi\sigma^2}\right)^{N/2} \exp\left\{-\frac{1}{2\sigma^2}\sum_{n=1}^{N}|y(\mathbf{x}_n;\mathbf{w}) - t_n|^2\right\}. \tag{13.9}$$

When viewed as a function of $\mathbf{w}$ this is called the *likelihood function.*

One technique from classical statistics for estimating $\mathbf{w}$ is called *maximum likelihood* and involves setting $\mathbf{w}$ to the value which maximizes the likelihood function. For convenience we can instead minimize the negative logarithm of the likelihood function (since '$-\ln$' is a monotonically decreasing function) given by

$$-\ln L(\mathbf{w}) = \frac{N}{2}\ln\sigma^2 + \frac{N}{2}\ln(2\pi) + \frac{1}{2\sigma^2}\sum_{n=1}^{N}|y(\mathbf{x}_n;\mathbf{w}) - t_n|^2. \tag{13.10}$$

Note that minimizing $\ln L(\mathbf{w})$ in (13.10) with respect to $\mathbf{w}$ is equivalent to minimizing the sum of squares error function (13.3). We denote the resulting value of $\mathbf{w}$ by $\mathbf{w}_{\mathrm{ML}}$. Similarly we can minimize (13.10) with respect to β with the result

$$\frac{1}{\beta_{\mathrm{ML}}} = \frac{1}{N}\sum_{n=1}^{N}|y(\mathbf{x}_n;\mathbf{w}_{\mathrm{ML}}) - t_n|^2. \tag{13.11}$$

This provides us with an estimate of the noise level associated with the data under the assumed model.

13.1.4 Bayesian regression

We have seen that a classical treatment of our regression problem seeks a point estimate of the unknown parameter vector $\mathbf{w}$. By contrast, in a Bayesian approach we characterize the uncertainty in $\mathbf{w}$ through a probability distribution $p(\mathbf{w})$. Observations of data points modify this distribution by virtue of Bayes' theorem, with the effect of the data being mediated through the likelihood function.

Specifically we define a prior distribution $p(\mathbf{w})$ which expresses our uncertainty in $\mathbf{w}$ taking account of all information aside from the data itself, and which, without loss of generality, can be written in the form

$$p(\mathbf{w}|\alpha) \propto \exp\{-\alpha\Omega(\mathbf{w})\} \tag{13.12}$$

where α can again be regarded as a hyperparameter. As a specific example we might choose a Gaussian distribution for $p(\mathbf{w}|\alpha)$ of the form

$$p(\mathbf{w}|\alpha) = \left(\frac{\alpha}{2\pi}\right)^{M/2} \exp\left\{-\frac{\alpha}{2}\|\mathbf{w}\|^2\right\}. \tag{13.13}$$

We can now use Bayes' theorem to express the posterior distribution for $\mathbf{w}$ as the product of the prior distribution and the likelihood function

$$p(\mathbf{w}|\mathbf{t}, \alpha, \sigma^2) \propto p(\mathbf{w}|\alpha)L(\mathbf{w}) \tag{13.14}$$

where, as before, $L(\mathbf{w}) = p(\mathbf{t}|\mathbf{w}, \sigma^2)$.

In a Bayesian treatment we make predictions by integrating with respect to the posterior distribution of $\mathbf{w}$, and we discuss this in detail shortly. For the moment, let us suppose that we wish to use the posterior distribution to find a point estimate for $\mathbf{w}$, and that we choose to do this by finding the value of $\mathbf{w}$ which maximizes the posterior distribution, or equivalently which minimizes the negative logarithm of the distribution. Taking the negative log of the right hand side of (13.14) and using (13.12) and (13.9) we see that maximizing the log of the posterior distribution is equivalent to minimizing

$$\frac{1}{2\sigma^2}\sum_{n=1}^{N}|y(\mathbf{x}_n; \mathbf{w}) - t_n|^2 + \frac{\alpha}{2}\Omega(\mathbf{w}) \tag{13.15}$$

which represents a specific example of the regularized error function given by (13.4) in which $E(\mathbf{w})$ is proportional to the sum-of-squares error function (13.3).

Thus we see that there are very close similarities between this Bayesian viewpoint and the conventional one based on error function minimization and regularization, since the latter can be obtained as a specific approximation to the Bayesian approach. However, there is also a key distinction which is that in a Bayesian treatment we make predictions by *integrating* over the distribution of model parameters $\mathbf{w}$, rather than by using a specific estimated value of $\mathbf{w}$. On the one hand such integrations may often be analytically intractable and require either sophisticated Markov chain Monte Carlo methods, or more recent deterministic schemes such as variational techniques, to approximate them. On the other hand the integration implied by the Bayesian framework overcomes the issue of over-fitting (by averaging over many different possible solutions) and typically results in improved predictive capability.

Specifically, if we are given a new value of $\mathbf{x}$ then the predictive distribution for t is obtained from the sum and product rules of probability by marginalizing over $\mathbf{w}$

$$p(t|\mathbf{t}, \alpha, \beta) = \int p(\mathbf{w}|\mathbf{t}, \alpha, \sigma^2) p(t|\mathbf{w}, \sigma^2)\, d\mathbf{w}. \tag{13.16}$$

So far we have said little about the treatment of the hyperparameters α and σ^2. In most applications, suitable values for these will not be known in advance (although in some cases the noise level σ^2 may be known) and so a Bayesian treatment will introduce prior distributions over these quantities, and then eliminate them from the problem by marginalization. We shall see in Section 13.3 that an appropriate choice of prior distribution can lead to some powerful properties for the resulting model, including sparsity of the basis function representation. First, however, we review briefly a popular model for regression and classification based on point estimates of parameters, which also exhibits sparsity.

13.2 Support Vector Machines

One specific instantiation of the model given by (13.2) is the *support vector machine* (SVM) [5, 17, 12] which, although not usually defined explicitly in this form, ultimately makes predictions based on the function

$$y(\mathbf{x}; \mathbf{w}) = \sum_{i=1}^{N} w_i K(\mathbf{x}, \mathbf{x}_i) + w_0. \tag{13.17}$$

Here $\phi_i(\mathbf{x}) = K(\mathbf{x}, \mathbf{x}_i)$ is a *kernel* function, effectively defining one basis function for each example in the training set. Note that we consider here a directly parameterized kernel-based model while for SVM models this form emerges from the 'dual' formulation. Throughout this chapter no primal-dual interpretations are made. All reasoning is done in terms of parameterized models.

One key feature of the SVM is that, in the classification case, its target function attempts to minimize a measure of error on the training set while simultaneously maximizing the 'margin' between the two classes (in the feature space implicitly defined by the kernel). This is an effective mechanism for avoiding over-fitting, which leads to good generalization, and which furthermore results in a sparse model dependent only on a subset of kernel functions, namely those associated with specific training examples $\mathbf{x}_n$ (the *support vectors*) that lie either on the margin or on the 'wrong' side of it. State-of-the-art results have been reported on many tasks where the SVM has been applied.

However, despite its success, we can identify a number of significant and practical disadvantages of the support vector learning methodology:

- Standard SVMs make unnecessarily liberal use of basis functions since the number of support vectors required typically grows linearly with the size of the training set.

- In general, predictions are not *probabilistic* (although approximate Bayesian inference schemes for the SVM have been discussed in [7, 13]). In regression the SVM outputs a point estimate, and in classification, a 'hard' binary decision. For many real world applications, as distinct from algorithm bench-marking, we require the conditional distribution $p(t|\mathbf{x})$ of targets given inputs rather than just a point prediction. Such a distribution expresses our uncertainty in the prediction and offers numerous advantages [3] such as optimal rejection, flexible and optimal decision making, fusion of outputs with other sources of probabilistic information, and so on.

- It is necessary to estimate the error/margin trade-off parameter 'C' (and in regression, the insensitivity parameter 'ϵ' too). This generally entails a cross-validation procedure, which is wasteful both of data and computation.

- The kernel function $K(\mathbf{x}, \mathbf{x}_i)$ must satisfy Mercer's condition.

Nevertheless, the twin properties of accuracy and sparsity make the SVM a very attractive model. We have already discussed how a Bayesian approach to modelling can naturally deal with complexity control and avoid over-fitting. Here we show that in addition, through a judicious choice of prior over $\mathbf{w}$, we can obtain models that are also highly sparse (typically much more so than the SVM) and at the same time also overcome all the above limitations.

13.3 The Relevance Vector Machine

While we stress that the framework we are about to describe can be applied to general models of the type (13.2) (*i.e.* to arbitrary sets of basis functions), we now focus on a model we term the *relevance vector machine*, or RVM [14, 15], which is a Bayesian framework for regression and classification with analogous sparsity properties to the support vector machine. We adopt a fully probabilistic framework and introduce a prior over the model weights governed by a set of hyperparameters, one associated with each weight, whose most probable values are iteratively estimated from the data. Sparsity is achieved because the posterior distributions of many of the weights are sharply (indeed infinitely) peaked around zero. We term those training vectors associated with the remaining non-zero weights 'relevance' vectors, in deference to the principle of *automatic relevance determination* which motivates this approach [9, 10]. The most compelling feature of the RVM is that, while capable of generalization performance comparable to an equivalent SVM, the number of relevance vectors is, in most cases, dramatically smaller than the number of support vectors used by an SVM to solve the same problem. For the purposes of this presentation, we focus initially on the Bayesian *regression* model and associated inference procedures, and then summarize the modifications required in the case of classification.

13.3.1 Model specification

Given a data set of input-target pairs $\{\mathbf{x}_n, t_n\}_{n=1}^N$ we assume that the targets are samples from a model with additive noise, as described by (13.1), with a noise process

given by a zero-mean Gaussian with variance σ^2, so that

$$p(t_n|\mathbf{x}) = \mathcal{N}(t_n|y(\mathbf{x}_n;\mathbf{w}),\sigma^2). \tag{13.18}$$

The function $y(\mathbf{x};\mathbf{w})$ is as defined in (13.17) for the SVM where we identify our general basis functions with the kernel as parameterized by the training vectors: $\phi_i(\mathbf{x}) \equiv K(\mathbf{x},\mathbf{x}_i)$. Due to the assumption of independence of the t_n, the likelihood of the complete data set can be written as

$$p(\mathbf{t}|\mathbf{w},\sigma^2) = (2\pi\sigma^2)^{-N/2}\exp\left\{-\frac{1}{2\sigma^2}\|\mathbf{t}-\boldsymbol{\Phi}\mathbf{w}\|^2\right\}, \tag{13.19}$$

where the $N \times (N+1)$ matrix $\boldsymbol{\Phi} = [\boldsymbol{\phi}(\mathbf{x}_1),\boldsymbol{\phi}(\mathbf{x}_2),\ldots,\boldsymbol{\phi}(\mathbf{x}_N)]^{\mathrm{T}}$ is called the *design* matrix, $\boldsymbol{\phi}(\mathbf{x}_n) = [1,K(\mathbf{x}_n,\mathbf{x}_1),K(\mathbf{x}_n,\mathbf{x}_2),\ldots,K(\mathbf{x}_n,\mathbf{x}_N)]^{\mathrm{T}}$, $\mathbf{t} = (t_1\ldots t_N)^{\mathrm{T}}$, and $\mathbf{w} = (w_0\ldots w_N)^{\mathrm{T}}$.

With as many parameters in the model as training examples, we would expect maximum-likelihood estimation of $\mathbf{w}$ and σ^2 from (13.19) to lead to severe over-fitting. In the SVM, this difficulty is effectively avoided by the inclusion of the 'margin' term. Here, instead, we adopt a Bayesian perspective, and introduce an explicit *prior* probability distribution over the parameters.

We encode a preference for smoother functions by using a Gaussian prior distribution over $\mathbf{w}$, as discussed earlier, but now modified through the introduction of a separate hyperparameter for each parameter in the model

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{i=0}^{N}\mathcal{N}(w_i|0,\alpha_i^{-1}), \tag{13.20}$$

with $\boldsymbol{\alpha}$ a vector of $N+1$ hyperparameters.

To complete the specification of this *hierarchical* prior, we must define hyperpriors over $\boldsymbol{\alpha}$, as well as over the final remaining parameter in the model, the noise variance σ^2. These quantities are examples of *scale* parameters, and suitable priors for these are given by Gamma distributions (see, *e.g.* [2]):

$$p(\boldsymbol{\alpha}) = \prod_{i=0}^{N}\mathrm{Gamma}(\alpha_i|a,b),$$
$$p(\beta) = \mathrm{Gamma}(\beta|c,d),$$

with $\beta \equiv \sigma^{-2}$ and where

$$\mathrm{Gamma}(\alpha|a,b) = \Gamma(a)^{-1}b^a\alpha^{a-1}e^{-b\alpha}, \tag{13.21}$$

in which $\Gamma(a) = \int_0^\infty t^{a-1}e^{-t}dt$, is the gamma function [1]. The Gamma prior becomes non-informative in the limit $a \to 0$, $b \to 0$. Since, in this limit, the hyperpriors become improper, we might fix their parameters to small values: *e.g.* $a = b = c = d = 10^{-4}$. However, by setting these parameters to zero, we obtain uniform hyperpriors (over a *logarithmic* scale). Since all scales are equally likely, a pleasing consequence of the use of these improper hyperpriors is that of scale-invariance: predictions are independent

of linear scaling of both t and the basis function outputs so, for example, results do not depend on the unit of measurement of the targets. The case of general Gamma priors for α and β is covered in more detail in [15] and [4], but from now on here we assume uniform scale priors with $a = b = c = d = 0$.

This choice of prior distributions is related to those used in *automatic relevance determination*, or ARD [9, 10]. Using such priors in a neural network, individual hyperparameters would typically control *groups* of weights, in particular those associated with each input dimension x. For inputs which have little value in predicting the outputs, the posterior distribution over the hyperparameters becomes concentrated at large values, thus effectively switching off such 'low relevance' inputs. This idea has also been applied to the input variables in 'Gaussian process' models [19].

Here, the assignment of an individual hyperparameter to each weight, or basis function, is the key feature of the sparse Bayesian framework, and is responsible ultimately for its sparsity properties. To introduce an additional $N + 1$ parameters to the model may seem counter-intuitive, since there is already one parameter per basis function (and therefore one parameter per data point for kernel functions centered on the data), but from a Bayesian perspective, provided we correctly integrate out all of these parameters, or can approximate such an integration sufficiently accurately, then having the number of parameters exceed the number of data points presents no particular difficulty either from a theoretical or from a practical point of view (see pp. 16–17, of [10]).

13.3.2 The effective prior

We may question why the choice of a Gaussian prior should express any preference for sparse models. In order to gain insight into the this effect we can integrate out the hyperparameters to discover the true identity of the prior over the weights. For a Gamma prior over the hyperparameters, it is possible to integrate out α_i independently for each weight, to obtain the marginal, or what might be considered the 'true', weight prior:

$$\begin{aligned} p(w_i) &= \int p(w_i|\alpha_i)p(\alpha_i)\, d\alpha_i, \\ &= \frac{b^a \Gamma(a + \frac{1}{2})}{(2\pi)^{\frac{1}{2}}\Gamma(a)} \, (b + w_i^2/2)^{-(a+\frac{1}{2})}, \end{aligned} \tag{13.22}$$

where $\Gamma(\cdot)$ is the gamma function as defined earlier. Equation (13.22) corresponds to the density of a Student-t distribution, and so the overall marginal weight prior is a product of independent Student-t distributions over the w_i. A visualization of this Student-t prior, alongside a Gaussian, is given in Figure 13.1. For the case of the uniform hyperprior, with $a = b = 0$, we obtain the improper prior $p(w_i) \propto 1/|w_i|$. Intuitively, this looks very much like a sparse prior since it is sharply peaked at zero like the popular Laplace prior $p(w_i) \propto \exp(-|w_i|)$, which has been previously utilized to obtain sparsity in Bayesian contexts [20]. The elegance of this approach therefore lies in the use of hierarchical modelling to obtain a prior over weights which encourages sparsity while still making use of fully conjugate exponential-family distributions throughout.

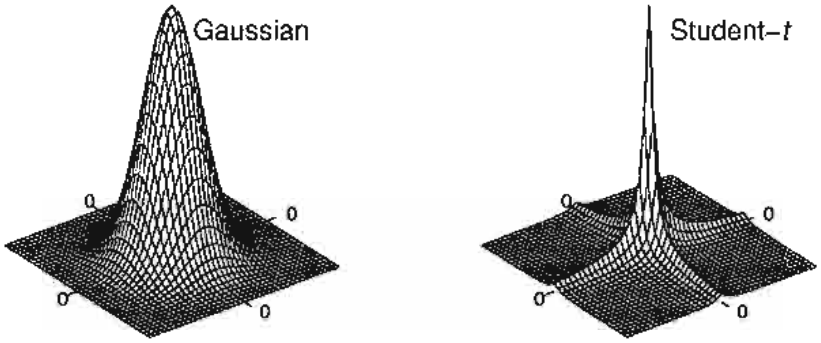


Figure 13.1: LEFT: an example Gaussian prior $p(\mathbf{w}|\boldsymbol{\alpha})$ in two dimensions. RIGHT: the prior $p(\mathbf{w})$, where the hyperparameters have been integrated out to give a product of Student-t distributions. Note that the probability mass is concentrated close to the origin, where both weights go to zero, and also along 'spines' where one or other of the two weights goes to zero.

Unfortunately, we cannot continue the Bayesian analysis down this route to compute $p(\mathbf{w}|\mathbf{t})$, since the marginal $p(\mathbf{w})$ is no longer Gaussian, and so the marginalization over $\mathbf{w}$ is no longer analytically tractable. Because of this, it is not convenient to work with the marginal prior directly and in the next section we take a different tack.

13.3.3 Inference

Having defined the prior, Bayesian inference proceeds by computing, from Bayes' rule, the posterior over all unknowns given the data:

$$p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2|\mathbf{t}) = \frac{p(\mathbf{t}|\mathbf{w}, \boldsymbol{\alpha}, \sigma^2)p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2)}{p(\mathbf{t})}. \tag{13.23}$$

Then, given a new test point, $\mathbf{x}_*$, predictions are made for the corresponding target t_*, in terms of the predictive distribution:

$$p(t_*|\mathbf{t}) = \int p(t_*|\mathbf{w}, \boldsymbol{\alpha}, \sigma^2)\; p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2|\mathbf{t})\; d\mathbf{w}\; d\boldsymbol{\alpha}\; d\sigma^2. \tag{13.24}$$

As is the case with many non-trivial Bayesian models, it is not possible to perform these computations in full analytically, and we must seek an effective approximation.

We cannot compute the posterior $p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2|\mathbf{t})$ in (13.23) directly since we cannot perform the normalizing integral on the right-hand-side, $p(\mathbf{t}) = \int p(\mathbf{t}|\mathbf{w}, \boldsymbol{\alpha}, \sigma^2)$ $p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2)\; d\mathbf{w}\; d\boldsymbol{\alpha}\; d\sigma^2$. Instead, we decompose the posterior as:

$$p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2|\mathbf{t}) = p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}, \sigma^2)p(\boldsymbol{\alpha}, \sigma^2|\mathbf{t}), \tag{13.25}$$

and note that we can compute analytically the posterior distribution over the weights since its normalizing integral, $p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2) = \int p(\mathbf{t}|\mathbf{w}, \sigma^2)\; p(\mathbf{w}|\boldsymbol{\alpha})\; d\mathbf{w}$, is a convolution of Gaussians. The posterior distribution over the weights is thus given by:

$$p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}, \sigma^2) = \frac{p(\mathbf{t}|\mathbf{w}, \sigma^2)p(\mathbf{w}|\boldsymbol{\alpha})}{p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2)}, \tag{13.26}$$

$$= (2\pi)^{-(N+1)/2}|\boldsymbol{\Sigma}|^{-1/2}\; \exp\left\{-\frac{1}{2}(\mathbf{w} - \boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{w} - \boldsymbol{\mu})\right\}, \tag{13.27}$$

where the posterior covariance and mean are respectively:

$$\Sigma = (\sigma^{-2}\Phi^{\mathrm{T}}\Phi + \mathbf{A})^{-1}, \tag{13.28}$$

$$\mu = \sigma^{-2}\Sigma\Phi^{\mathrm{T}}\mathrm{t}, \tag{13.29}$$

with $\mathbf{A} = \mathrm{diag}(\alpha_0, \alpha_1, \ldots, \alpha_N)$.

It is now necessary to make some form of approximation, and we do so by replacing the integration over the hyperparameters by point estimates involving their most probable posterior values. We do this on the basis that this point-estimate is representative of the posterior in the sense that functions generated utilizing the posterior mode values are close to those obtained by sampling from the full posterior distribution. It is important to realize that this does not necessitate that the entire mass of the posterior be accurately approximated by the delta-function. For predictive purposes, rather than requiring $p(\boldsymbol{\alpha}, \sigma^2|\mathbf{t}) \approx \delta(\boldsymbol{\alpha}_{\mathrm{MP}}, \sigma^2_{\mathrm{MP}})$, we only desire

$$\int p(t_*|\boldsymbol{\alpha}, \sigma^2) p(\boldsymbol{\alpha}, \sigma^2|\mathrm{t})\ d\boldsymbol{\alpha}\ d\sigma^2 \simeq p(t_*|\boldsymbol{\alpha}_{\mathrm{MP}}, \sigma^2_{\mathrm{MP}}) \tag{13.30}$$

to be a good approximation. This notion may be visualized by a thought experiment where we consider that we are utilizing two identical basis functions $\phi_i(\mathrm{x})$ and $\phi_j(\mathrm{x})$. It follows from (13.31) shortly that the mode of $p(\boldsymbol{\alpha}, \sigma^2|\mathrm{t})$ will not be unique, but will comprise an infinite 'ridge' where $\alpha_i^{-1} + \alpha_j^{-1}$ is some constant value. No delta-function can be considered to be a good approximation to the probability mass associated with this ridge, yet any point along it implies an identical predictive distribution and so (13.30) holds. Evidence from the experiments presented in this article and elsewhere suggests that this predictive approximation is very effective in general.

Relevance vector 'learning' thus becomes the search for the hyperparameter posterior mode, *i.e.* the maximization of $p(\boldsymbol{\alpha}, \sigma^2|\mathbf{t}) \propto p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2) p(\boldsymbol{\alpha}) p(\sigma^2)$ with respect to $\boldsymbol{\alpha}$ and β. For the case of uniform hyperpriors, we need only maximize the *marginal likelihood*, or equivalently its logarithm, $\ln p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2)$, which is computable and given by:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\alpha}) = \ln p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2) &= \ln \int_{-\infty}^{\infty} p(\mathbf{t}|\mathbf{w}, \sigma^2)\ p(\mathbf{w}|\boldsymbol{\alpha})\ d\mathbf{w}, \\ &= -\frac{1}{2}\left[N \ln 2\pi + \ln|\mathbf{C}| + \mathbf{t}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{t}\right], \end{aligned} \tag{13.31}$$

with

$$\mathbf{C} = \sigma^2\mathbf{I} + \Phi\mathbf{A}^{-1}\Phi^{\mathrm{T}}. \tag{13.32}$$

13.3.4 Making predictions

In practice, having maximized (13.31) (we consider this task shortly), we make predictions based on the posterior distribution over the weights, conditioned on the maximizing values $\boldsymbol{\alpha}_{\mathrm{MP}}$ and σ^2_{MP}. We can then compute the predictive distribution, from (13.24), for a new datum $\mathbf{x}_*$ using (13.27):

$$p(t_*|\mathbf{t}, \boldsymbol{\alpha}_{\mathrm{MP}}, \sigma^2_{\mathrm{MP}}) = \int p(t_*|\mathbf{w}, \sigma^2_{\mathrm{MP}}) p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}_{\mathrm{MP}}, \sigma^2_{\mathrm{MP}})\ d\mathbf{w}. \tag{13.33}$$

Since both terms in the integrand are Gaussian, this is readily computed, giving:

$$p(t_*|\mathbf{t}, \boldsymbol{\alpha}_{\mathrm{MP}}, \sigma^2_{\mathrm{MP}}) = \mathcal{N}(t_*|y_*, \sigma^2_*),$$

with

$$y_* = \boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_*), \tag{13.34}$$

$$\sigma^2_* = \sigma^2_{\mathrm{MP}} + \boldsymbol{\phi}(\mathbf{x}_*)^{\mathrm{T}}\boldsymbol{\Sigma}\boldsymbol{\phi}(\mathbf{x}_*). \tag{13.35}$$

So the predictive mean is intuitively $y(\mathbf{x}_*; \boldsymbol{\mu})$, or the basis functions weighted by the posterior mean weights. We will find that the maximizing values of many of the hyperparameters will be infinite, implying from (13.27) that the corresponding weights in $\mathbf{w}_{\mathrm{MP}}$ will be exactly zero and the predictor y_* is thus sparse.

13.3.5 Properties of the marginal likelihood

Values of $\boldsymbol{\alpha}$ (assume σ^2 is fixed for now) which maximize (13.31) cannot be jointly obtained in closed form. However, in [6] it was shown that we can maximize $\mathcal{L}(\boldsymbol{\alpha})$ with respect to a *single* hyperparameter α_i. To show this, we first straightforwardly decompose $\mathbf{C}$ in (13.32) as

$$\begin{aligned}\mathbf{C} &= \sigma^2\mathbf{I} + \sum_{m\neq i} \alpha_m^{-1}\boldsymbol{\phi}_m\boldsymbol{\phi}_m^{\mathrm{T}} + \alpha_i^{-1}\boldsymbol{\phi}_i\boldsymbol{\phi}_i^{\mathrm{T}},\\ &= \mathbf{C}_{-i} + \alpha_i^{-1}\boldsymbol{\phi}_i\boldsymbol{\phi}_i^{\mathrm{T}},\end{aligned} \tag{13.36}$$

where $\mathbf{C}_{-i}$ is $\mathbf{C}$ with the contribution of basis vector i removed and $\boldsymbol{\phi}_i$ is the i-th column of $\boldsymbol{\Phi}$. Established matrix determinant and inverse identities can then be employed to re-write $\mathcal{L}(\boldsymbol{\alpha})$ as:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\alpha}) &= -\frac{1}{2}\Big[N\ln(2\pi) + \ln|\mathbf{C}_{-i}| + \mathbf{t}^{\mathrm{T}}\mathbf{C}_{-i}^{-1}\mathbf{t}\\ &\qquad - \ln\alpha_i + \ln(\alpha_i + \boldsymbol{\phi}_i^{\mathrm{T}}\mathbf{C}_{-i}^{-1}\boldsymbol{\phi}_i) - \frac{(\boldsymbol{\phi}_i^{\mathrm{T}}\mathbf{C}_{-i}^{-1}\mathbf{t})^2}{\alpha_i + \boldsymbol{\phi}_i^{\mathrm{T}}\mathbf{C}_{-i}^{-1}\boldsymbol{\phi}_i}\Big],\\ &= \mathcal{L}(\boldsymbol{\alpha}_{-i}) + \frac{1}{2}\left[\ln\alpha_i - \ln(\alpha_i + s_i) + \frac{q_i^2}{\alpha_i + s_i}\right]\\ &= \mathcal{L}(\boldsymbol{\alpha}_{-i}) + \ell(\alpha_i),\end{aligned} \tag{13.37}$$

where for simplification of forthcoming expressions, we have defined:

$$s_i \triangleq \boldsymbol{\phi}_i^{\mathrm{T}}\mathbf{C}_{-i}^{-1}\boldsymbol{\phi}_i, \quad \text{and} \quad q_i \triangleq \boldsymbol{\phi}_i^{\mathrm{T}}\mathbf{C}_{-i}^{-1}\mathbf{t}. \tag{13.38}$$

The objective function has now been decomposed into $\mathcal{L}(\boldsymbol{\alpha}_{-i})$, the marginal likelihood with $\boldsymbol{\phi}_i$ excluded, and $\ell(\alpha_i)$, where terms in α_i are now conveniently isolated.

Analysis of $\ell(\alpha_i)$ [6] shows that $\mathcal{L}(\boldsymbol{\alpha})$ has a unique maximum with respect to α_i:

$$\alpha_i = \frac{s_i^2}{q_i^2 - s_i}, \qquad \text{if } q_i^2 > s_i, \tag{13.39}$$

$$\alpha_i = \infty, \qquad \text{if } q_i^2 \leq s_i. \tag{13.40}$$

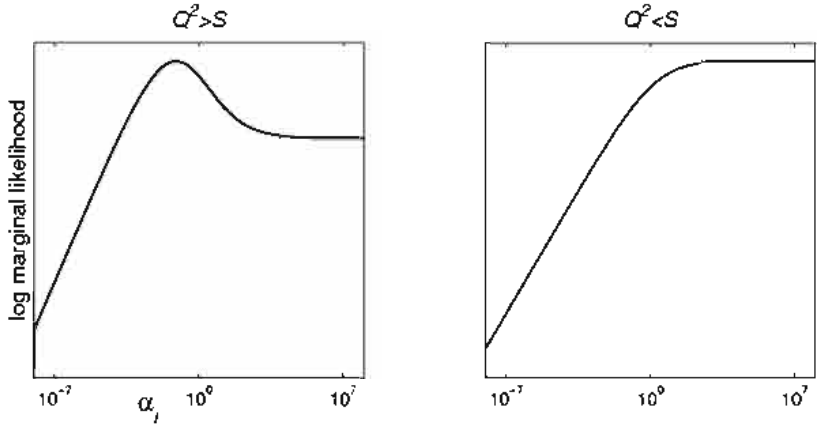


Figure 13.2: Example plots of $\ell(\alpha_i)$ against α_i (on a log scale) for $q^2 > s$ (left), showing the single maximum at finite α_i, and $q^2 < s$ (right), showing the maximum as $\alpha_i \to \infty$.

An example illustrating these two cases is given in Figure 13.2.

Thus sparsity arises from equation (13.40): if this condition holds, then the marginal likelihood is maximized when the individual basis (kernel) function $K(\mathbf{x}, \mathbf{x}_i)$ is removed from the model. Note, from (13.38), that q_i and s_i depend on all other ($m \neq i$) hyperparameters so the sparsity conditions for all $\boldsymbol{\alpha}$ mutually interact.

13.3.6 Hyperparameter optimization

For optimizing hyperparameters, a simple set of re-estimation formulae can be derived [14, 15], but a more recent, and much more efficient, approach is given in [16] which we briefly summarize here.

We start by computing the quantities s_i and q_i. In fact, it is easier to maintain and update values of

$$S_i = \boldsymbol{\phi}_i^{\mathsf{T}} \mathbf{C}^{-1} \boldsymbol{\phi}_i, \qquad Q_i = \boldsymbol{\phi}_i^{\mathsf{T}} \mathbf{C}^{-1} \mathbf{t}, \tag{13.41}$$

and from these it follows simply:

$$s_i = \frac{\alpha_i S_i}{\alpha_i - S_i}, \qquad q_i = \frac{\alpha_i Q_i}{\alpha_i - S_i}. \tag{13.42}$$

Note that when $\alpha_i = \infty$, $s_i = S_i$ and $q_i = Q_i$. In practice, then, it is convenient to utilise the Woodbury identity to obtain the quantities of interest:

$$S_i = \boldsymbol{\phi}_i^{\mathsf{T}} \mathbf{B} \boldsymbol{\phi}_i - \boldsymbol{\phi}_i^{\mathsf{T}} \mathbf{B} \boldsymbol{\Phi} \widehat{\boldsymbol{\Sigma}} \boldsymbol{\Phi}^{\mathsf{T}} \mathbf{B} \boldsymbol{\phi}_i, \tag{13.43}$$

$$Q_i = \boldsymbol{\phi}_i^{\mathsf{T}} \mathbf{B} \widehat{\mathbf{t}} - \boldsymbol{\phi}_i^{\mathsf{T}} \mathbf{B} \widehat{\boldsymbol{\mu}}, \tag{13.44}$$

where $\mathbf{B} \equiv \sigma^{-2}\mathbf{I}$, $\widehat{\boldsymbol{\Sigma}} \equiv \boldsymbol{\Sigma}$, $\widehat{\boldsymbol{\mu}} \equiv \boldsymbol{\mu}$ and $\widehat{\mathbf{t}} \equiv \mathbf{t}$ in the regression case, and for the classification case as explicitly defined in the next section.

Given these, consider individual basis functions (hyperparameters) in turn and note that the results (13.39) and (13.40) imply that for a given α_i:

- If $\boldsymbol{\phi}_i$ is 'in the model' (*i.e.* $\alpha_i < \infty$) yet $q_i^2 \leq s_i$, then $\boldsymbol{\phi}_i$ may be deleted (*i.e.* α_i set to ∞),

- If $\boldsymbol{\phi}_i$ is excluded from the model ($\alpha_i = \infty$) and $q_i^2 > s_i$, $\boldsymbol{\phi}_i$ may be 'added': *i.e.* α_i is set to the optimal finite value given by (13.39).

- If $\boldsymbol{\phi}_i$ is 'in the model' and $q_i^2 > s_i$ then α_i may be re-estimated.

All these actions guarantee to increase the marginal likelihood function, and we thus have a framework for making discrete changes to the model, by adding and deleting basis functions, in a principled probabilistic manner.

For the noise variance σ^2, we can derive a re-estimation equation which may be utilized concurrently with those of $\boldsymbol{\alpha}$ in order to infer the noise variance:

$$(\sigma^2)^{\text{new}} = \frac{\|\mathbf{t} - \boldsymbol{\Phi}\boldsymbol{\mu}\|^2}{N - M + \sum_i \alpha_i \Sigma_{ii}}. \tag{13.45}$$

Here, the 'N' in the denominator refers to the number of data examples and not the number of basis functions.

In terms of the efficiency of this procedure, note that SVM optimization does benefit from a convexity property whereas the marginal likelihood may be expected to be multi-modal (although it is generally observed that these modes lead to similar predictive functions). Furthermore, there exist some established SVM learning strategies that have been efficiently tuned for large-scale problems. The algorithm outlined here appears to necessitate some costly matrix inversions, but these matrices are in fact of the order of the number of basis functions actually in the model at any point. Since we may choose to initialise with an 'empty' model, and typically the converged model is very sparse, in practice the computational cost is very managable. Further details on the optimization procedure, including details of computing and updating s_i and q_i, as well as illustrative timing comparisons with the SVM, are given in [16].

13.3.7 Relevance vector machines for classification

Sparse Bayesian classification follows an essentially identical framework as described for regression above, but using a Bernoulli likelihood and a sigmoidal link function to account for the change in the target quantities. As a consequence, there is an additional approximation step in the algorithm.

Applying the logistic sigmoid link function $\sigma(y) = 1/(1 + e^{-y})$ to $y(\mathbf{x}; \mathbf{w})$ and, adopting the Bernoulli distribution for $P(t|\mathbf{x})$, we write the likelihood as:

$$P(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} \sigma\{y(\mathbf{x}_n; \mathbf{w})\}^{t_n} \left[1 - \sigma\{y(\mathbf{x}_n; \mathbf{w})\}\right]^{1-t_n}, \tag{13.46}$$

where, following from the probabilistic specification, the targets $t_n \in \{0, 1\}$.

Unlike the regression case, the weights cannot be integrated out analytically, precluding closed-form expressions for either the weight posterior $p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha})$ or the marginal likelihood $P(\mathbf{t}|\boldsymbol{\alpha})$. We thus utilize the Laplace approximation procedure, as used in [8]:

1. For the current values of $\boldsymbol{\alpha}$, the the mode of the posterior distribution is found iteratively to give the 'most probable' weights $\widehat{\boldsymbol{\mu}}$. Since $p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha}) \propto P(\mathbf{t}|\mathbf{w})p(\mathbf{w}|\boldsymbol{\alpha})$, this is equivalent to finding the maximum, over $\mathbf{w}$, of

$$\ln\{P(\mathbf{t}|\mathbf{w})p(\mathbf{w}|\boldsymbol{\alpha})\} = \sum_{n=1}^{N}[t_n \ln y_n + (1-t_n)\ln(1-y_n)] - \frac{1}{2}\mathbf{w}^{\mathsf{T}}\mathbf{A}\mathbf{w}, \tag{13.47}$$

with $y_n = \sigma\{y(\mathbf{x}_n;\mathbf{w})\}$. This is a standard procedure, since (13.47) is a penalized logistic log-likelihood function, and necessitates iterative maximization. We have used a second-order Newton method related to the 'iteratively-reweighted least-squares' algorithm to find $\widehat{\boldsymbol{\mu}}$.

2. Laplace's method is simply a quadratic approximation to the log-posterior around its mode. The quantity (13.47) is differentiated twice to give:

$$\nabla_{\mathbf{w}}\nabla_{\mathbf{w}}\ln p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha})\big|_{\widehat{\boldsymbol{\mu}}} = -(\boldsymbol{\Phi}^{\mathsf{T}}\mathbf{B}\boldsymbol{\Phi} + \mathbf{A}), \tag{13.48}$$

where $\mathbf{B} = \mathrm{diag}\beta_1, \beta_2, \ldots, \beta_N$ is a diagonal matrix with $\beta_n = \sigma\{y(\mathbf{x}_n)\}\,[1-\sigma\{y(\mathbf{x}_n)\}]$. This is then negated and inverted to give the covariance $\widehat{\boldsymbol{\Sigma}}$ for a Gaussian approximation to the posterior over weights centered at $\widehat{\boldsymbol{\mu}}$.

At the mode of $p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha})$, using (13.48) and the fact that $\nabla_{\mathbf{w}}\ln p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha})\big|_{\widehat{\boldsymbol{\mu}}} = 0$, we can see we have effectively locally 'linearized' the classification problem around $\widehat{\boldsymbol{\mu}}$ with

$$\widehat{\boldsymbol{\Sigma}} = (\boldsymbol{\Phi}^{\mathsf{T}}\mathbf{B}\boldsymbol{\Phi} + \mathbf{A})^{-1}, \tag{13.49}$$

$$\widehat{\boldsymbol{\mu}} = \widehat{\boldsymbol{\Sigma}}\boldsymbol{\Phi}^{\mathsf{T}}\mathbf{B}\widehat{\mathbf{t}}, \tag{13.50}$$

and

$$\widehat{\mathbf{t}} = \boldsymbol{\Phi}\widehat{\boldsymbol{\mu}} + \mathbf{B}^{-1}(\mathbf{t} - \sigma\{\boldsymbol{\Phi}\widehat{\boldsymbol{\mu}}\}). \tag{13.51}$$

These equations are equivalent to the solution to a generalized least squares problem. Compared with (13.29), it can be seen that the Laplace approximation effectively maps the classification problem to a regression one with targets $\widehat{\mathbf{t}}$ and data-dependent (heteroscedastic) noise, in which the inverse noise variance for ϵ_n is given by $\beta_n = \sigma\{y(\mathbf{x}_n)\}\,[1-\sigma\{y(\mathbf{x}_n)\}]$.

The quantities $\widehat{\boldsymbol{\Sigma}}$, $\widehat{\boldsymbol{\mu}}$ and $\widehat{\mathbf{t}}$ can be substituted into equations (13.43) and (13.44) in order to compute the quantities s_i and q_i which may then be exploited in the algorithm of Section 13.3.6 exactly as in the regression case.

13.4 The Relevance Vector Machine in Action

13.4.1 Illustrative synthetic data: regression

The function $\mathrm{sinc}(x) = \sin(x)/x$ has been a popular choice to illustrate support vector regression [18, 17], where in place of the classification margin, the *ϵ-insensitive region* is introduced, a 'tube' of $\pm\epsilon$ around the function within which errors are not penalized.

In this case, the support vectors lie on the edge of, or outside, this region. For example, using a univariate 'linear spline' kernel:

$$K(x_m, x_n) = 1 + x_m x_n + x_m x_n \min(x_m, x_n) - \frac{x_m + x_n}{2} \min(x_m, x_n)^2 + \frac{\min(x_m, x_n)^3}{3}, \tag{13.52}$$

and with $\epsilon = 0.01$, the approximation of $\mathrm{sinc}(x)$ based on 100 uniformly-spaced noise-free samples in $[-10, 10]$ utilizes 36 support vectors as shown in Figure 13.3 (left).

In the RVM, we model the same data with the same kernel (13.52), which is utilized to define a set of basis functions $\phi_n(x) = K(x, x_n)$, $n = 1 \ldots N$. Typically, we will be tackling problems where the target function has some additive noise component, whose variance is represented by σ^2. However, for the purposes of comparison with this "function approximation" SVM example, we model the sinc function with a relevance vector machine but *fix* the noise variance in this case at 0.01^2 and then re-estimate $\boldsymbol{\alpha}$ alone. This setting of the noise standard deviation to 0.01 is intended to be analogous, in an approximate sense, to the setting the ϵ-insensitivity to the same value in the SVM. Using this fixed σ, the RVM approximator is plotted in Figure 13.3 (right), and requires only 9 relevance vectors. The largest error is 0.0070, compared to 0.010 in the support vector case, and we have obtained the dual benefit of both increased accuracy and sparsity.

Figure 13.4 illustrates a case which is more representative of real data in which uniform noise (*i.e.* not corresponding to the RVM noise model) in $[-0.2, 0.2]$ is added to the targets. Again, a linear spline kernel was used. The trained RVM uses 6 relevance vectors, compared to 29 for the SVM. The root-mean-square (RMS) deviation from the true function for the RVM is 0.0245, while for the SVM it is 0.0291. Note that for the latter model, it was necessary to tune the parameters C and ϵ, in this case using 5-fold cross-validation. For the RVM, the analogues of these parameters (the α's and σ^2) are automatically estimated by the learning procedure.

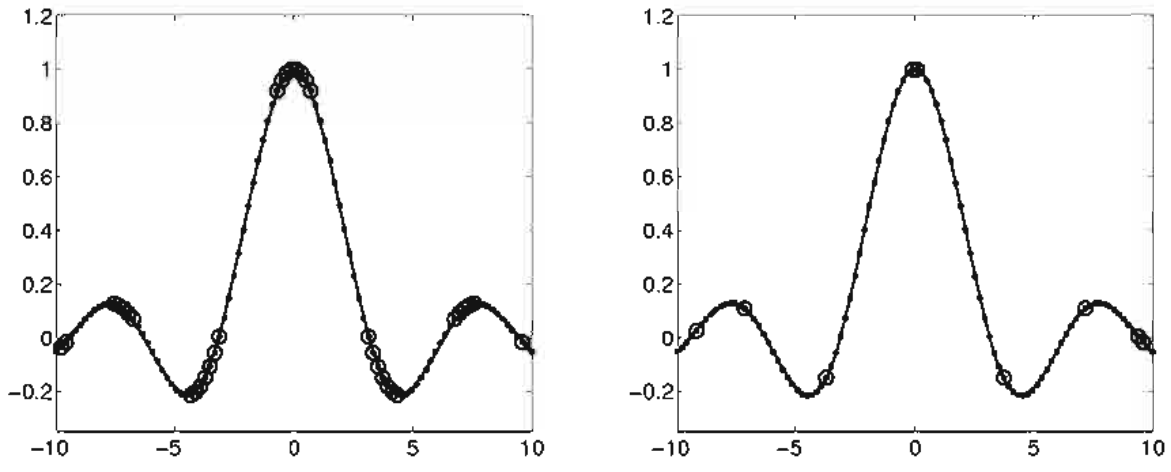


Figure 13.3: Support (left) and relevance (right) vector approximations to $\mathrm{sinc}(x)$ from 100 noise-free examples using 'linear spline' basis functions. The estimated functions are drawn as solid lines with support/relevance vectors shown circled.

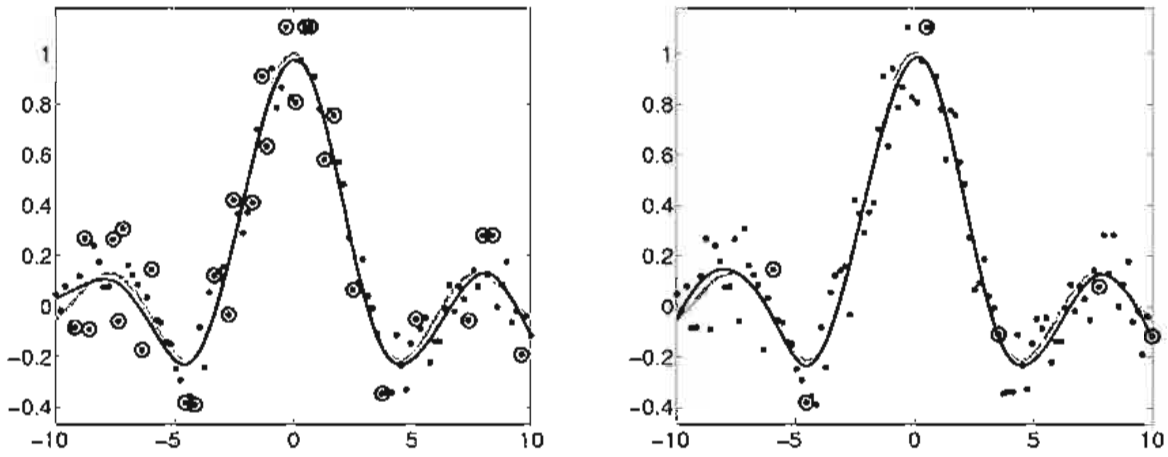


Figure 13.4: Support (left) and relevance (right) vector approximations to $\mathrm{sinc}(x)$, based on 100 noisy samples. The estimated functions are drawn as solid lines, the true function in grey, and support/relevance vectors are again shown circled.

13.4.2 Illustrative synthetic data: classification

We utilize artificially-generated data in two dimensions in order to illustrate graphically the selection of relevance vectors for classification. Both class 1 (denoted by '×') and class 2 (denoted by '•') were generated from mixtures of two Gaussians by [11], with the classes overlapping to the extent that the Bayes error is around 8%.

A relevance vector classifier is compared to its support vector counterpart, using a 'Gaussian' kernel which we define as

$$K(\mathbf{x}_m, \mathbf{x}_n) = \exp(-r^{-2}\|\mathbf{x}_m - \mathbf{x}_n\|^2), \tag{13.53}$$

with r the 'width' parameter, chosen here to be 0.5. A value of C for the SVM was selected using 5-fold cross-validation on the training set. The results for a 100-example training set (randomly chosen from Ripley's original 250) are given in Figure 13.5. The test error (from the associated 1000-example test set) for the RVM (9.3%) is slightly superior to the SVM (10.6%), but the remarkable feature of contrast is the complexity of the classifiers. The support vector machine utilizes 38 kernel functions compared to just 4 for the relevance vector method. This considerable difference in sparsity between the two methods is typical, as the later results on benchmark data sets support.

Of interest also is the fact that, unlike with the SVM, the relevance vectors are some distance from the decision boundary (in x-space), appearing more 'prototypical' or even 'anti-boundary' in character. A qualitative explanation for this phenomenon, discussed in more detail in [15], is that the output of a basis function centered on or near the decision boundary is an unreliable indicator of class membership (*i.e.* its output is poorly-aligned with the data set in t-space), and such basis functions are naturally penalized (deemed 'irrelevant') under the Bayesian framework. Of course, there is no implication that the utilization of either boundary-located or prototypically-located functions is 'correct' in any sense.

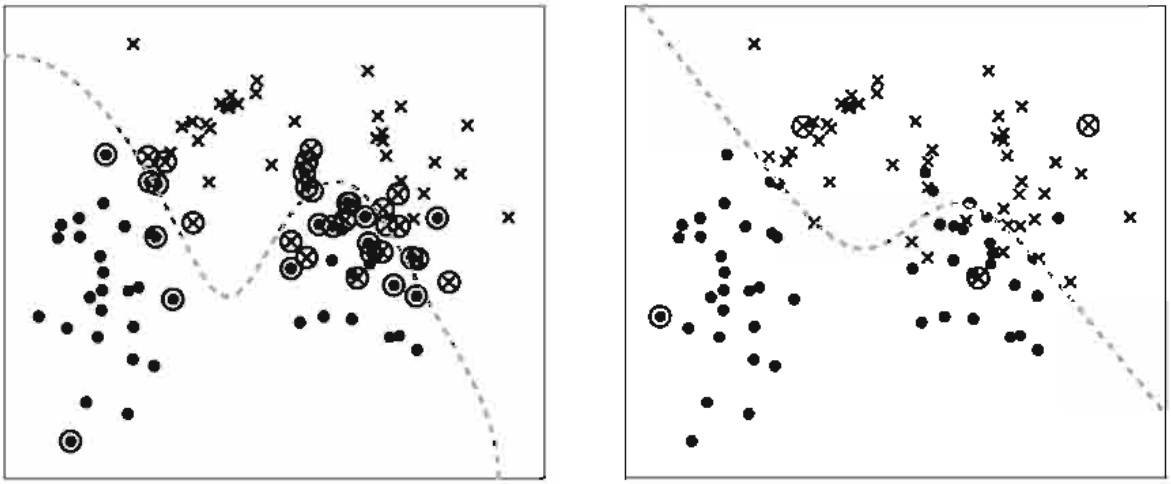


Figure 13.5: SVM (left) and RVM (right) classifiers on 100 examples from Ripley's Gaussian-mixture data set. The decision boundary is shown dashed, and relevance/support vectors are shown circled to emphasize the dramatic reduction in complexity of the RVM model.

13.4.3 Benchmark results

The following tables, taken from [15], summarize regression and classification performance of the relevance vector machine on some example benchmark data sets, comparing results for illustrative purposes with equivalent support vector machines. For each data set the number of training examples (N) and the number of input variables (d) are given in the tables. The prediction error obtained and the number of vectors (support or relevance) required, generally averaged over a number of repetitions, are then given for both models. By way of summary, the RVM statistics were also normalized by those of the SVM and the overall average is displayed. A Gaussian kernel was utilized and its input scale parameter chosen by 5-fold cross-validation.

			errors		*vectors*	
Regression Data set	N	d	**SVM**	**RVM**	**SVM**	**RVM**
Sinc (Gaussian noise)	100	1	0.0378	0.0326	45.2	6.7
Sinc (Uniform noise)	100	1	0.0215	0.0187	44.3	7.0
Friedman #2	240	4	4140	3505	110.3	6.9
Friedman #3	240	4	0.0202	0.0164	106.5	11.5
Boston Housing	481	13	8.04	7.46	142.8	39.0
Normalized Mean			1.00	**0.86**	1.00	**0.15**

			errors		*vectors*	
Classification Data set	N	d	**SVM**	**RVM**	**SVM**	**RVM**
Pima Diabetes	200	8	20.1%	19.6%	109	4
U.S.P.S.	7291	256	4.4%	5.1%	2540	316
Banana	400	2	10.9%	10.8%	135.2	11.4
Breast Cancer	200	9	26.9%	29.9%	116.7	6.3
Titanic	150	3	22.1%	23.0%	93.7	65.3
Waveform	400	21	10.3%	10.9%	146.4	14.6
German	700	20	22.6%	22.2%	411.2	12.5
Image	1300	18	3.0%	3.9 %	166.6	34.6
Normalized Mean			1.00	**1.08**	1.00	**0.17**

In summary, in this small number of experiments, the RVM exhibited 14% lower error than the SVM and utilized only 15% of the basis functions on average for regression. In classification, error was 8% greater on average, yet still only 17% of the basis functions were utilized.

13.5 Discussion

In this brief tutorial we have outlined some of the basic concepts of regression and classification from the Bayesian perspective. We have discussed in detail a specific Bayesian model called the Relevance Vector Machine, which leads to highly sparse solutions and having excellent generalization properties.

The treatment of the Relevance Vector Machine given here is not completely Bayesian since point estimates are made for the hyperparameters, whereas in a fully Bayesian treatment we should define hyperpriors over these hyperparameters, and then integrate out the hyperparameters in order to make predictions.

However, as we have already noted, it is not possible to integrate out all of the parameters and hyperparameters analytically. This problem can be addressed by using deterministic approximation schemes based *variational inference*, in which a factorized approximation to the full posterior distribution is used [4]. One consequence of this more complete treatment of the RVM is confirmation that the approach based on point estimates, as discussed in this tutorial, does indeed give a good approximation to a more complete Bayesian approach.

Bibliography

[1] M. Abromowitz and I. A. Stegun, *Handbook of Mathematical Functions*, Dover (1965).

[2] J. O. Berger, *Statistical decision theory and Bayesian analysis*, Springer, second edition (1985).

[3] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press (1995).

[4] C. M. Bishop and M. E. Tipping, Variational relevance vector machines, In Craig Boutilier and Moisés Goldszmidt, editors, *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann (2000) 46–53.

[5] B. Boser, I. Guyon, and V. N. Vapnik, A training algorithm for optimal margin classifiers, In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory* (1992) 144–152.

[6] A. C. Faul and M. E. Tipping, Analysis of sparse Bayesian learning, In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, MIT Press (2002) 383–389.

[7] J.T. Kwok, The evidence framework applied to support vector machines, *IEEE Transactions on Neural Networks* 10 (2000) 1018–1031.

[8] D.J.C. MacKay, The evidence framework applied to classification networks, *Neural Computation* 4(5) (1992) 720–736.

[9] D.J.C. MacKay, Bayesian methods for backpropagation networks, In E Domany, J L van Hemmen, and K Schulten, editors, *Models of Neural Networks III*, chapter 6, Springer (1994) 211–254.

[10] R. M. Neal, *Bayesian Learning for Neural Networks*, Springer (1996).

[11] B. D. Ripley, *Pattern Recognition and Neural Networks*, Cambridge University Press (1996).

[12] B. Schölkopf, C. J C Burges, and A. J Smola (editors), *Advances in Kernel Methods: Support Vector Learning*, MIT Press (1999).

[13] P. Sollich Probabilistic methods for support vector machines. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, MIT Press (2000) 349–355.

[14] M. E. Tipping, The Relevance Vector Machine, In Sara A Solla, Todd K Leen, and Klaus-Robert Müller, editors, *Advances in Neural Information Processing Systems 12*, MIT Press (2000) 652–658.

[15] M. E. Tipping, Sparse Bayesian learning and the relevance vector machine, *Journal of Machine Learning Research* 1 (2001) 211–244.

[16] M. E. Tipping and A. C. Faul, Fast marginal likelihood maximisation for sparse Bayesian models, In B. Frey and C. M. Bishop, editors, *Artificial Intelligence and Statistics* (2003).

[17] V. N. Vapnik, *Statistical Learning Theory*, Wiley (1998).

[18] V. N. Vapnik, S. E. Golowich, and A. J Smola, Support vector method for function approximation, regression estimation and signal processing, In Michael C Mozer, Michael I Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems 9*, MIT Press (1997).

[19] C. K. I. Williams, Prediction with gaussian processes: From linear regression to linear prediction and beyond, Technical report, NCRG, Aston University, Birmingham, U.K. (1997).

[20] P. M. Williams, Bayesian regularisation and pruning using a Laplace prior, *Neural Computation* **7**(1) (1995) 117–143.

Chapter 14

Bayesian Field Theory: from Likelihood Fields to Hyperfields

Jörg C. Lemm[1]

Abstract. Bayesian models are based on the combination of a likelihood model and a prior model. The nonparametric Bayesian models discussed in this chapter combine the likelihood models of density estimation, regression, and inverse quantum theory, expressed in terms of 'likelihood fields', with nonparametric prior models for such fields. Starting from Gaussian process priors, which, for example, are able to implement approximate symmetries for the likelihood fields, those are made more flexible by introducing hyperparameters and, generalized to a hyperparameter valued function, by hyperfields. In a joint Maximum A Posteriori approximation this results in coupled equations for likelihood fields and hyperfields. An example of a useful application of hyperfields is the adaption of the mean function of a Gaussian process prior.

[1]The author wants to thank J. Uhlig and A. Weiguny for fruitful discussions.

14.1 Introduction

It is well known that empirical learning can not be based on observational data alone but does always also depend on *a priori* information, explicitly or implicitly. For models with a small number of parameters, the essential part of *a priori* information being in that case the restriction of the model space, additional priors for the parameters are typically easily overwhelmed by the data if the number of observations becomes large enough. The situation is completely different for nonparametric approaches, where the number of data is too small to determine all the degrees of freedom of the model. Thus, in nonparametric modeling it is essential that all available *a priori* information is included in the model. To this end prior models are useful which are formulated explicitly in terms of the likelihood of interest. Most priors used for nonparametric models in practice are some kind of smoothness priors. In the following we will discuss some possibilities to construct priors more general than smoothness priors and present techniques to adapt general prior models to the available *a priori* knowledge. Before we discuss specific nonparametric likelihood and prior models we first have to define the basic quantities in empirical learning and their probabilistic relations.

14.2 The Bayesian framework

14.2.1 The basic probabilistic model

For empirical learning it is convenient to distinguish three groups of variables:

1. observable (visible) independent variables x representing 'inputs' or 'controlled causes',
2. observable (visible) dependent variables y representing 'outputs' or 'measured effects',
3. not directly observable (hidden, latent) variables ϕ describing the possible (pure) 'states of Nature' considered in the model under study.

The joint probability can be factorized according to

$$p(x, y, \phi) = p(y|x, \phi)\, p(\phi|x) p(x). \tag{14.1}$$

The first factor $p(y|x, \phi)$ describes the probability (or probability density for continuous y) of finding y given the visible variables are in state x under a specific model state ('state of Nature') ϕ. For given data, $D = (x, y)$, the factor $p(y|x, \phi)$ is also known as the *likelihood* of ϕ under D.

The hidden variables ϕ parameterize the space of possible states we are using to model the probabilistic input–output relation between the x and the y. In 'parametric' models the hidden variables ϕ represent a (in many practical cases relatively small) number of parameters, while in 'nonparameteric' models the hidden variables ϕ form a function or 'field', so that expressions like $p(\phi|x)$ represent a stochastic process [8, 11, 41, 37]. For example, in nonparametric density estimation the function values of the likelihood itself, i.e., all the numbers $\phi(x, y) = p(y|x, \phi)$ for all x and y, can be

considered as the primary degrees of freedom. A function $\phi(x, y)$ which determines the likelihood $p(y|x, \phi)$ will be called a *likelihood field*. Likelihood fields which are different from $\phi(x, y) = p(y|x, \phi)$ will be discussed below. Similar to nonparametric density estimation problems, we will speak of nonparametric regression if each value of the regression function $\phi(x) = \int dy\, y\, p(y|x, \phi)$ is considered as a primary degree of freedom. Another example is nonparametric inverse quantum theory where the function of interest can be a function $\phi(x) = v(x)$ describing a local quantum potential as function of a coordinate x. The primary degrees of freedom $\phi(x, y)$ or $\phi(x)$ can then be related by explicit prior information like a smoothness constraint. As the number of parameters in a parametric model can become large, for instance, in a neural network, and also nonparametric models must be discretized in some basis to perform numerical calculations, there is no sharp distinction between parametric and nonparametric models in practice. Nonparametric models as we will understand them, however, try to formulate the available *a priori* information not in terms of parameters, which are often difficult to interpret, but explicitly in terms of the function ϕ of interest, e.g., in terms of the likelihood values $p(y|x, \phi)$ in density estimation problems, or in terms of the regression function in regression problems [23].

In the following we will assume scenarios with n independent (tuples of) *training data*

$$D = \{(x_i, y_i)|1 \leq i \leq n\} = \{(x_D, y_D)\} \tag{14.2}$$

which are sampled according to the likelihood $p(y_i|x_i, \phi)$ and where the x_i and y_i can be vectors. The independent variables in the data, $x_D = \{x_i|1 \leq i \leq n\}$, may have been either fixed in advance or sampled under some given $p(x)$. We also assume, that $p(\phi|x)$ can be written as $p(\phi|D_0)$ with D_0 being that part of the independent visible variables which determines the prior $p(\phi|D_0)$ of the hidden variables ϕ. We will call D_0 the *a priori information* on ϕ and $p(\phi|D_0)$ the *prior* for ϕ. Under those assumptions, we end up with models of the form

$$p(x, y, \phi|D_0) = p(\phi|D_0)p(y_D|x_D, \phi)\, p(x_D) = p(\phi|D_0) \prod_{i=1}^{n} p(y_i|x_i, \phi)\, p(x_i), \tag{14.3}$$

or, conditional on the independent variable x,

$$p(y, \phi|x, D_0) = p(y_D|x_D, \phi)\, p(\phi|D_0) = \prod_{i=1}^{n} p(y_i|x_i, \phi)\, p(\phi|D_0). \tag{14.4}$$

14.2.2 Bayesian decision theory and predictive density

Empirical learning is used to make decisions. Hence, let us consider a set of possible actions a from which we want to choose an optimal one. In selecting an optimal action we want to use all the information we have at hand, i.e., training data D and *a priori* information D_0. In particular, we will be interested in approximation problems, like density estimation where the possible actions represent predictive densities $p(y|x, a)$ and our aim is to approximate the true state $p(y|x, \phi_{\text{true}})$, by an optimal posterior predictive density $p(y|x, D, D_0)$. To define optimality we select a loss function $l(x, y, a)$

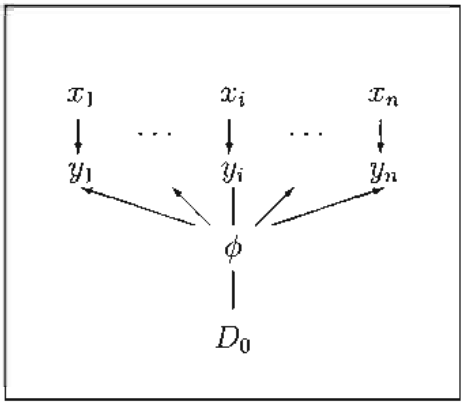


Figure 14.1: Graphical representation of the probabilistic model (14.4) with independent data, conditioned on the independent variables x_i and the unknown state ϕ.

which describes the *loss* suffered in situation x if y appears and action a has been selected. Then an optimal action (e.g., an optimal prediction) minimizes, for given data D and prior D_0, the *expected risk* [1, 35]

$$r(a, D, D_0) = \int dx\, dy\, p(x)\, p(y|x, D, D_0)\, l(x, y, a). \tag{14.5}$$

If we denote the expectation under the joint posterior predictive density $p(x, y|D, D_0)$ $= p(x)p(y|D, D_0)$ by $< \cdots >_{X,Y|D,D_0}$, we can write Eq. (14.5) as

$$r(a, D, D_0) =< l(x, y, a) >_{X,Y|D,D_0} . \tag{14.6}$$

We may remark that $p(x)$ is the density of x for the (test) data for which prediction is intended and not the possibly different $p(x)$ which has been used for sampling the training data D. The typical loss function for approximation problems is the log–loss

$$l(x, y, a) = -\ln p(y|x, a), \tag{14.7}$$

for which the expected risk becomes the (x–averaged) Kull–Back–Leibler distance between $p(y|x, a)$ and $p(y|x, D, D_0)$. As it is not difficult to show by using Jensen's inequality the Kullback–Leibler–distance is minimal for $p(y|x, a) = p(y|x, D, D_0)$. Thus, under log–loss the optimal model is the posterior predictive density

$$p(y|x, D, D_0) = \int d\phi\, p(y|x, \phi)\, p(\phi|D, D_0). \tag{14.8}$$

In nonparametric models where ϕ is a function the integral over ϕ stands for a functional integral. Such integrals are mathematically well defined for Gaussian processes ('generalized free Euclidean fields' or 'generalized Brownian motion' in the language of physics), but for more general processes ('(self–)interacting fields') a general mathematical definition of functional integrals is still an unsolved problem. One possible approach is to define a non–Gaussian functional integral by renormalization procedures

in perturbation theory [5] or on a lattice [30], another simpler possibility is to discretize the function ϕ in some appropriate basis so the functional integral becomes a standard (but typically still extremely highdimensional) integral. The remaining highdimensional integrals can either be approximated by Monte Carlo methods [16, 14, 12, 33, 19] or by the method of steepest descents (Laplace approximation), in this context known as Maximum A Posteriori (MAP) approximation [7, 4, 35, 12]. The latter assumes that

$$p(y|x, D, D_0) \approx p(y|x, \phi^*) \tag{14.9}$$

for the 'MAP solution' ϕ^* which maximizes the posterior

$$\phi^* = \text{argmax}_\phi p(\phi|D, D_0). \tag{14.10}$$

Hence, in a MAP approximation the integration over ϕ is replaced by maximization over ϕ. Eq. (14.9) is a good approximation if the posterior $p(\phi|D, D_0)$ can be well approximated by a single Gaussian and the factor $p(y|x, \phi)$ in Eq. (14.8) is slowly varying at the stationary point ϕ^* compared to the posterior. This is often expected to be the case if the number of data is sufficiently large. In principle one can also go beyond the MAP approximation by using the MAP solution as starting point for a perturbation series, for example, graphically expressed by Feynman diagrams [34, 17].

In Gaussian regression $p(y|x, a)$ is assumed to be of Gaussian form with fixed variance and only the mean or regression function $a(x) = < y >_{Y|x,a} = \int dy\, y\, p(y|x, a)$ is adapted. In linear (parametric) Gaussian regression the regression function is chosen as a linear function in x, i.e., $< y >_{Y|x,a} = a_0 + a_1 x$, while in nonparametric Gaussian regression for each x the value $< y >_{Y|x,a}$ is treated as a single degree of freedom so that the form of the regression function is not restricted. For Gaussian regression log–loss is equivalent to a squared error loss and it is straightforward to check that the regression function which minimizes the expected risk is the posterior regression function, i.e., the expectation of y under the posterior predictive density

$$a^*(x) = < y >_{Y|x,a^*} = < y >_{Y|x,D,D_0} = \int dy\, y\, p(y|x, D, D_0). \tag{14.11}$$

A typical loss function for classification problems (where a discrete dependent variable y represents the class variable) is the 0–1–loss

$$l(y, x, a) = -\delta_{y,a(x)} \tag{14.12}$$

where $\delta_{y,a(x)}$ stands for the Kronecker-δ which is equal to one if $y = a(x)$ and zero otherwise. In that case it turns out that the optimal $a(x)$ is a mode function $\max_y p(y|x, D, D_0)$ of the predictive density. Table 14.1 lists the basic functions which appear in Bayesian models.

14.2.3 Bayes' theorem: from prior and likelihood to the posterior

The short discussion of Bayesian decision theory in 14.2.2 showed that for density estimation, regression, as well as for classification learning can be based on the posterior

$p(\phi\|D_0)$	prior
$p(\phi\|D, D_0)$	posterior
$p(y\|x, \phi)$	likelihood, predictive density for a pure state
$p(y\|x, D_0)$	prior predictive density
$p(y\|x, D, D_0)$	posterior predictive density

Table 14.1: The basic probabilities (or densities, stochastic processes, respectively).

predictive density $p(y|x, D, D_0)$. Using $p(A) = \int dB\, p(A, B)$ for $A = \phi$ and $B = y$, $p(y|x, \phi, D, D_0) = p(y|x, \phi)$, and $p(\phi|x, D, D_0) = p(\phi|D, D_0)$ the posterior predictive density can be expressed as

$$p(y|x, D, D_0) = \int d\phi\, p(y|x, \phi) p(\phi|D, D_0), \tag{14.13}$$

where the density (or the stochastic process in nonparametric models) $p(\phi|D, D_0)$ is known as the *posterior*.

The posterior, which we are interested in, is linked to the prior and likelihood, which are assumed to be given, by Bayes' theorem:

$$p(H|D) = \frac{p(D|H)p(H)}{p(D)}. \tag{14.14}$$

Because $p(D)$ can not be zero for observed data, Eq. (14.14) is a direct consequence of the definition of conditional probabilities,

$$p(H, D) = p(H|D)p(D) = p(D|H)p(H). \tag{14.15}$$

In a Bayesian context D represents the available data and H stands for the hidden variables (hypotheses, theories, models) which in the nonparametric approaches we will study are described by functions or likelihood fields, like $\phi(x)$ in regression or $\phi(x, y)$ in general density estimation problems. The terms in Eq. (14.14) are commonly referred to as

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}, \tag{14.16}$$

where the evidence $p(D)$ is independent of H. Conditional on the independent training data $x_D = \{x_i | 1 \leq i \leq n\}$, denoting the *a priori* information explicitly by D_0, and writing ϕ for H Bayes' theorem (14.14) reads

$$p(\phi|y_D, x_D, D_0) = \frac{p(y_D|x_D, \phi)p(\phi|D_0)}{p(y_D|x_D, D_0)}. \tag{14.17}$$

According to Eq. (14.17) two ingredients are needed to calculate the posterior: a likelihood model describing the measurement process from which the data are obtained and a prior model implementing the available *a priori* information.

14.3 Likelihood models

Likelihood or data models define the different problem classes. In the following we will introduce the nonparametric likelihood models of density estimation, regression and inverse quantum theory.

14.3.1 Log–probabilities, energies, and density estimation

In practice it is often more convenient to work with log–probabilities (or log–densities, log-processes, respectively)

$$L = \ln p \tag{14.18}$$

instead of probabilities p. Indeed, for log–probabilities no non–negativity constraint is necessary because $p = \exp L$ is non–negative for arbitrary real L. Another advantage of log–probabilities is that products of probabilities become sums for log–probabilities,

$$p(A, B) = p(A)p(B) \Rightarrow L(A, B) = L(A) + L(B). \tag{14.19}$$

In limits where the number of probability factors approaches infinity the corresponding sums of log–probability terms become integrals, e.g., in the log–prior, which are more common objects than the infinite products which appear for probabilities. For example, according to Bayes' theorem (14.17) the *log–posterior*

$$L(\phi|D, D_0) = \ln p(\phi|D, D_0) \tag{14.20}$$

becomes up to a ϕ–independent constant the sum of the *log–prior*

$$L(\phi|D_0) = \ln p(\phi|D_0), \tag{14.21}$$

which in nonparametric approaches typically consists of an integral over x and/or y, and the *log–likelihood*

$$L(y_D|x_D, \phi) = \sum_i L(y_i|x_i, \phi) = \ln \prod_i p(y_i|x_i, \phi), \tag{14.22}$$

which, in contrast to the log–prior, is a finite sum over data points. We remark that the log–likelihood can be related to the *averaged conditional Kerridge inaccuracy* K_c,

$$L(y_D|x_D, \phi) = \sum_i L(y_i|x_i, \phi) = -nK_c[p_{\text{emp}}(x', y'), p(y'|x', \phi)], \tag{14.23}$$

in short $L = \sum_i L_i = -nK_c$, where the averaged conditional Kerridge inaccuracy

$$K_c[p_{\text{emp}}(x', y'), p(y'|x', \phi)] = -\int dx'\, p_{\text{emp}}(x') K[p_{\text{emp}}(y'|x'), p(y'|x', \phi)] \tag{14.24}$$

is the empirical expectation of the corresponding Kerridge inaccuracy of the conditional densities, obtained by integrating over x with $p_{\text{emp}}(x')$,

$$K[p_{\text{emp}}(y'|x'), p(y'|x', \phi)] = -\int dy'\, p_{\text{emp}}(y'|x') \ln p(y'|x', \phi), \tag{14.25}$$

where the empirical densities for data $D = \{(x_i, y_i)|1 \leq i \leq n\}$ are given by

$$p_{\text{emp}}(y|x_i) = \frac{p_{\text{emp}}(x_i, y)}{p_{\text{emp}}(x_i)}, \tag{14.26}$$

$$p_{\text{emp}}(x, y) = \frac{1}{n}\sum_{i=1}^{n} \delta(x - x_i)\delta(y - y_i), \tag{14.27}$$

$$p_{\text{emp}}(x) = \frac{1}{n}\sum_{i=1}^{n} \delta(x - x_i), \tag{14.28}$$

with $x_i \in x_D$ so that $p_{\text{emp}}(y|x_i)$ is defined. Denoting the likelihood vector by $P(\phi)$ with components $P(x, y; \phi) = p(y|x, \phi)$ and defining the data vector $N(x, y) = np_{\text{emp}}(x, y)$ we can also write for the likelihood, alternatively to Eq. (14.23),

$$L(y_D|x_D, \phi) = -\int dx\, dy \ln p(y|x, \phi) N(x, y) = -\langle \ln P(\phi)|\, N\rangle \tag{14.29}$$

using the bra-ket notation with respect to x and y, i.e., $\langle f|\, g\rangle = \int dx\, dy f(x, y) g(x, y)$.

Besides being non-negative, probabilities (or densities) have to be normalized to one. Like non-negativity, the normalization condition can be implemented in the parameterization of a probability. Indeed, if we parameterize a probability as follows

$$p(x) = \frac{\exp\left(-\beta E(x)\right)}{Z}, \tag{14.30}$$

where the normalization factor or *partition sum* Z is given by

$$Z = \int dx\, \exp\left(-\beta E(x)\right), \tag{14.31}$$

then $p(x)$ is normalized and non-negative for arbitrary real functions $E(x)$. In analogy to statistical physics, we will call the function E *energy*. Clearly, the random variable x in Eq. (14.30) can be replaced by any other random variable. Hence energies, for example, appear as *likelihood energy* $E(y|x, \phi)$, *prior energy* $E(\phi|D_0)$, or *posterior energy* $E(\phi|D, D_0)$. The factor β, which in statistical physics plays the role of an *inverse temperature*, can be useful for several technical purposes, for example, when calculating the moments of the random variable $E(x)$. We will set $\beta = 1$ in the following.

If searching for a maximum of $p(x)$ the x-independent normalization factor Z is irrelevant. That means, when maximizing a probability, e.g., a posterior $p(\phi|D, D_0)$, then when working with energies like $E(\phi|D, D_0)$ we do not have to implement non-negativity and normalization constraints explicitly and we do not have to calculate the normalization factor Z. This is especially useful if Z is an integral over a high dimensional ϕ-space.

Summarizing, the terms in Eq. (14.30) can be referred to as

$$\text{probability} = \frac{1}{\text{partition sum}} \exp\left(-\frac{\text{energy}}{\text{temperature}}\right). \tag{14.32}$$

Table 14.2 lists some possible choices for the likelihood field $\phi(x, y)$ in density estimation and the corresponding constraints the field has to obey.

field $\phi(x,y)$	likelihood $p(y\|x,\phi) =$	constraints	
likelihood	$\phi(x,y)$	norm	non–negativity
unnormalized likelihood	$\frac{\phi(x,y)}{\int dy\,\phi(x,y)}$	—	non–negativity
log–likelihood	$\exp(\phi(x,y))$	norm	—
likelihood energy	$\frac{\exp(-\beta\phi(x,y))}{\int dy\,\exp(-\beta\phi(x,y))}$	—	—
distribution function	$\frac{d\phi(x,y)}{dy}$	boundary	monotony

Table 14.2: Some choices of likelihood fields and the corresponding constraints.

14.3.2 Regression

In regression problems the only unknown parameter of the likelihood , i.e., the conditional density $p(y|x,\phi)$ is assumed to be the mean or regression function

$$\phi(x) = \int dy\, y\, p(y|x,\phi). \tag{14.33}$$

A common case is Gaussian regression where the likelihood is chosen as a Gaussian with fixed variance

$$p(y|x,\phi) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{|y-\phi(x)|^2}{2\sigma^2}\right). \tag{14.34}$$

An example of such a Gaussian likelihood is shown in Fig. 14.2.

Regression models, however, can not only be formulated with a Gaussian likelihood as in Eq. (14.34) but also with different likelihood models. For example, in classical inverse problems one assumes a likelihood of the form

$$p(y|x,\phi) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{|y-(\mathbf{F}\phi)(x)|^2}{2\sigma^2}\right). \tag{14.35}$$

In that case data $D = \{(x_i,y_i)|1 \le i \le n\}$ are sampled for $(\mathbf{F}\phi)(x)$ and not for $\phi(x)$. The operator $\mathbf{F}$ often represents an 'instrument or apparatus function'. If it is linear, then

$$(\mathbf{F}\phi)(x) = \int dx'\, \mathbf{F}(x,x')\phi(x'). \tag{14.36}$$

A regression model with a Poisson likelihood

$$p(y|x,\phi) = \frac{[\phi(x)]^y}{y!}\, e^{-\phi(x)}, \quad y \in N,\, \phi > 0 \tag{14.37}$$

can be useful for counting events (y counts with mean $\phi(x)$ at x). Similarly , a binomial likelihood

$$p(y|x,n;\phi) = \binom{n(x)}{y} [\phi(x)]^y [1-\phi(x)]^{n(x)-y}, \quad y \in N,\, y \le n \in N,\, 0 \le \phi \le 1, \tag{14.38}$$

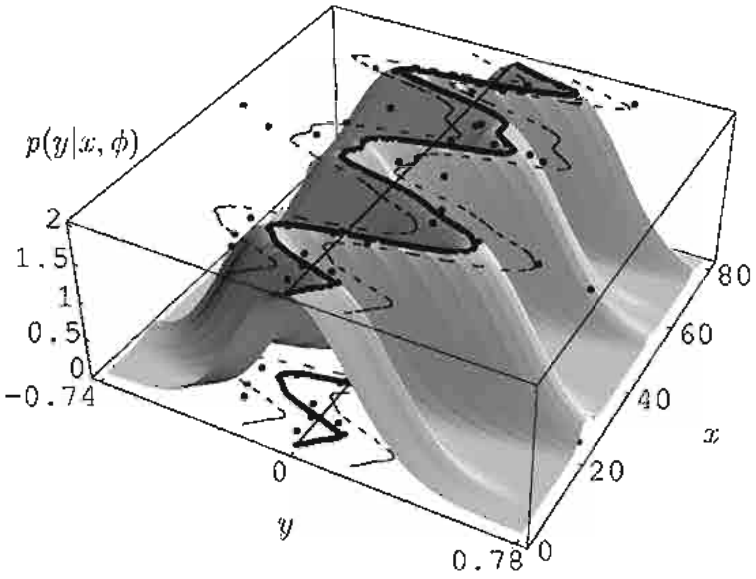


Figure 14.2: Example of a Gaussian likelihood $p(y|x,\phi)$ as in Eq. (14.34) where the regression function $\phi(x) = \int dy\, y\, p(y|x,\phi)$ (thick line) is sampled from a prior process $p(\phi|D_0)$ which favors smooth and approximately periodic functions. (The corresponding prior covariance is shown in Fig. 14.4.) The black dots represent data points $D = \{(x_i, y_i)|1 \leq i \leq n\}$ sampled from the given likelihood, the dashed line shows the $\pm\sigma$-range around mean function $\phi(x)$.

is appropriate if at x from $n(x)$ binary events a number of y 'ones' and $n(x) - y$ 'zeros' have been observed, $\phi(x)$ being the probability of 'one' at x. Finally we may remark that support vector machines for regression are formally equivalent to an regression model with ϵ-insensitive likelihood energy $E_\epsilon(y, \phi(x)) = \Theta(|y-\phi(x)|-\epsilon)(|y-\phi(x)|-\epsilon)$, $\Theta(x)$ denoting the step function and ϵ a constant [38, 42, 36].

14.3.3 Inverse quantum theory

As an example of a specific application of nonparametric Bayesian methods we will give a short introduction to the Bayesian approach to inverse quantum theory. In inverse quantum theory a quantum system is determined by measurements. A typical example, is the reconstruction of a quantum potential (which determines the force acting on a particle) from a finite number of position measurements. Such problems are based on the specific likelihood model of quantum theory: Measuring observable $\widehat{X}$ for a quantum system in a state described by density operator ρ, e.g., depending on an unknown potential ϕ, the probability to obtain value y is given by [40, 29]

$$p(y|\widehat{X}, \phi) = \mathrm{Tr}(\Pi_{\widehat{X},y}\, \rho(\phi)), \tag{14.39}$$

where Tr stands for the trace and $\Pi_{\widehat{X},y}$ is the projector on the space of eigenfunctions of operator $\widehat{X}$ with eigenvalue y. For a system in a state described by wave function

ψ the density operator is given by

$$\rho = \frac{|\psi\rangle\langle\psi|}{\langle\psi|\psi\rangle}, \tag{14.40}$$

for a system at (physical) temperature $1/\beta$ the density operator of the corresponding (canonical) ensemble is of the form (setting Boltzmann's constant equal to 1) [24]

$$\rho = \frac{\exp(-\beta H(\phi))}{\mathrm{Tr}\exp(-\beta H(\phi))}, \tag{14.41}$$

where H stands for the Hamiltonian of the system which depends on the field ϕ (and which may represent, for example, a potential). Similarly, the likelihood for quantum time series data $y(t_i)$ for an observable $\widehat{X}$ measured at times t_i, $1 \leq i \leq n$, reads [22]

$$p(y(t_i)|\widehat{X}, y(t_{i-1}), \phi) = |\langle y_i|U_i|y_{i-1}\rangle|^2 \tag{14.42}$$

where $|y_i\rangle$ stands for the eigenfunction of $\widehat{X}$ with eigenvalue $y(t_i)$ and (setting $\hbar = 1$)

$$U_i = \exp(-i(t_i - t_{i-1})H) \tag{14.43}$$

is the time evolution operator of a quantum system with time–independent Hamiltonian H. An example of the likelihood of a particle moving in a quantum potential is shown in Fig. 14.3.

In nonparametric Bayesian inverse quantum theory the likelihood model of quantum mechanics is combined with a prior model for the field ϕ. Being interested in the reconstruction of a quantum potential, the field $\phi(x)$ may represent the diagonal elements $v(x)$ of a local potential (where this time x stands not for an independent variable but for the coordinates of a quantum particle). The prior model for ϕ then implements available *a priori* information on $v(x)$, like smoothness or an approximate periodicity, the possibility of certain distortions or a specific fractal structure. Such nonparametric prior models, useful for density estimation, regression, and inverse quantum theory, will be discussed in the next section.

14.4 Prior models

For nonparametric models, where the number of degrees of freedom (represented by the field values $\phi(x)$ or $\phi(x, y)$) is large compared to the number n of data, learning requires the combination of the likelihood model with an appropriate prior model.

14.4.1 Gaussian prior factors and approximate symmetries

The simplest but already very flexible nonparametric prior models are Gaussian process priors [41, 43, 33, 18]

$$p(\phi|D_0) = \left(\det\frac{\mathrm{K}}{2\pi}\right)^{\frac{1}{2}} e^{-\frac{1}{2}\langle\phi-t|\mathrm{K}|\phi-t\rangle} \propto e^{-E(\phi|D_0)}, \tag{14.44}$$

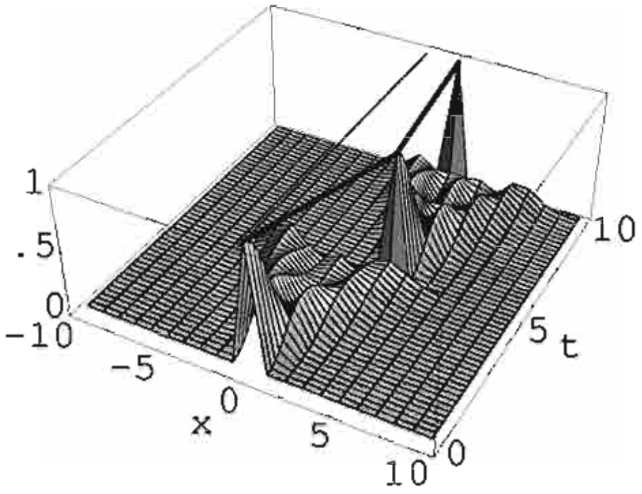


Figure 14.3: Quantum time series data: Time evolution of a particle observed at times $t_1 = 5$ and $t_2 = 10$ at positions x_i in a given potential $v_{\text{true}}(x)$ unknown to the learner. (The potential $v_{\text{true}}(x)$ is shown on the right hand side of Fig. 14.7.) The figure shows the likelihood $p(x(t_i)|\widehat{X}, x(t_{i-1}), v_{\text{true}})$ of Eq. (14.42), where the dependent variable y is represented by the measured coordinates $x(t_i)$, the potential function is chosen as field $\phi(x)$, and the Hamiltonian in (14.43) is the sum of a kinetic energy term and the local potential. In time–dependent inverse quantum theory the aim is to reconstruct the potential by combining the likelihood of such times series data with a (nonparametric) prior on $v(x)$.

where

$$\langle \phi - t \,|\mathbf{K}|\, \phi - t \rangle = \int dx\, dx'\, dy\, dy'\; (\phi(x,y) - t(x,y))\, \mathbf{K}(x,y;x',y')\, (\phi(x',y') - t(x',y')) \tag{14.45}$$

with $\mathbf{K}$ being the real symmetric positive semi-definite inverse covariance operator of the Gaussian process and $t(x,y)$ (or $t(x)$ for regression) the mean function of the process. As the $\phi(x,y)$ with maximal prior is given by the mean function, $t(x,y)$ will also be called a reference function or 'prior template' for the field ϕ.

A prior of the form (14.44) implements the *a priori* knowledge that the field ϕ is expected to be similar to the mean function t measured in a distance defined by the inverse covariance $\mathbf{K}$. Similarly, the *a priori* knowledge that the field ϕ is expected to be similar to a reference field t_1 OR to a reference field t_2, (measured in a distance being defined the by two inverse covariances $\mathbf{K}_1$ and $\mathbf{K}_2$) can be implemented by using a sum of two Gaussian prior components of the form (14.44). Generalized to more than two components this yields a *Gaussian mixture prior* $p(\phi|D_0) = \sum_k a_k p_k(\phi)$, with $\sum_k a_k = 1$, mean functions t_k and inverse covariances $\mathbf{K}_k$. In particular if the $\mathbf{K}_k$ are all equal, then such mixture models are not much more difficult to solve than models with a single Gaussian prior [20, 21]. An application of a mixture of Gaussian priors is presented in Fig. 14.6.

We remark that a product of Gaussian likelihood factors (14.34) for regression problems can be expressed in a form similar to that of the prior (14.44): The sum of mean squared error terms appearing in the exponent of the likelihood product can be

written [20]

$$\sum_{i=1}^{n} \frac{|y_i - \phi(x_i)|^2}{2\sigma^2} = \frac{1}{2} \langle \phi - t_D \,|\mathbf{K}_D|\, \phi - t_D \rangle + \frac{n}{2} V^D \tag{14.46}$$

with ϕ–independent data variance V_D, data mean vector $t_D = \mathbf{K}_D^{-1} \sum_{i=1}^{n} \mathbf{K}_i t_i$, inverse data covariance $\mathbf{K}_D = \sum_{i=1}^{n} \mathbf{K}_i$, where $t_i(x) = y_i$ and $\mathbf{K}_i(x, x') = \delta(x - x')\delta(x - x_i)/\sigma^2$.

The inverse prior covariance $\mathbf{K}$ can be chosen to implement approximate invariance of the field ϕ under an operation $\mathbf{S}$. Indeed, if we choose in (14.44)

$$\mathbf{K} = (\mathbf{I} - \mathbf{S})^T (\mathbf{I} - \mathbf{S}), \tag{14.47}$$

$\mathbf{I}$ denoting the identity and $\mathbf{S}^T$ the transpose of $\mathbf{S}$, then fields which fulfill $\phi = \mathbf{S}\phi$, i.e., which are invariant under $\mathbf{S}$, maximize the prior (14.44). Hence, a Gaussian prior (14.44) with an inverse covariance of the form (14.47) generates functions which are approximately invariant under $\mathbf{S}$. For example, if $\mathbf{S}$ is chosen as the translation of ϕ by θ units in x–direction, i.e., $\mathbf{S}_\theta^x \phi(x, y) = \phi(x - \theta, y)$, then the invariant fields are periodic in x and $\mathbf{K} = (\mathbf{I} - \mathbf{S}_\theta^x)^T (\mathbf{I} - \mathbf{S}_\theta^x)$ implements approximate periodicity in x–direction. A regression function sampled from a prior which favors smooth and approximate periodic functions is shown in Fig. 14.2.

Similarly, approximate invariance under infinitesimal transformations $\mathbf{s}$, generating a Lie group $\mathbf{S}(\theta) = \exp(\theta \mathbf{s})$ with parameter θ, can be implemented by inverse covariances

$$\mathbf{K} = \mathbf{s}^T \mathbf{s}. \tag{14.48}$$

A typical example is approximate invariance under infinitesimal translations ('smoothness'), say with respect to a K–dimensional vector x with components x_k, for which $\mathbf{s}_k = \partial/\partial x_k$ so that $\mathbf{K} = \sum_{k=1}^{K} \mathbf{s}_k^T \mathbf{s}_k$ becomes (under appropriate boundary conditions) the negative Laplacian $\mathbf{K} = -\Delta = -\sum_k \partial^2/\partial x_k^2$ (being of the form of kinetic energy terms in Euclidean field theory). In statistics one typically uses inverse covariances which also include higher order derivatives and which result in 'smoother', i.e., more times differentiable fields ϕ^* as MAP solutions. One such example is the Radial Basis Function (RBF) inverse prior covariance [15]

$$\mathbf{K} = \lambda \sum_{k=0}^{\infty} \frac{1}{k!} \left(-\frac{\sigma_0^2 \Delta}{2} \right)^k = \lambda \exp\left(-\frac{\sigma_0^2 \Delta}{2} \right), \tag{14.49}$$

with parameters λ and σ_0.

For density estimation with likelihood fields where the non–negativity and normalization constraints are not fulfilled automatically, those hard constraints have to be added to the Gaussian prior of Eq. (14.44), resulting in

$$p(\phi|D_0, \text{norm, non–neg.}) \propto \prod_x \delta\Big(\int dy\, p(y|x, \phi) - 1 \Big) \prod_{xy} \Theta\Big(p(y|x, \phi) \Big) p(\phi|D_0) \tag{14.50}$$

where the δ–functions ensure normalization for all x and the step functions Θ non–negativity of the likelihood for all x and y. The δ–function can be transformed into a

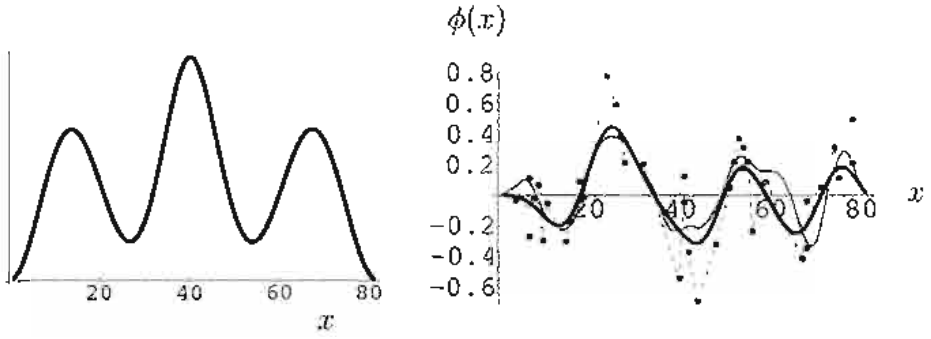


Figure 14.4: Left hand side: Prior covariance $\mathbf{K}^{-1}(x, x_0)$ (shown for fixed $x_0 = 40$ on mesh with 80 points) used to generate the sample data in Fig. 14.2. Right hand side: Regression function (thick line) reconstructed from the data points shown in Fig. 14.2 and using the prior covariance shown on the left. Also shown are: the data points (dots), the true regression function (thin line), and a piecewise linear interpolation (dashed line).

Lagrange multiplier term for the log-prior, yielding

$$\begin{aligned} E(\phi) &= -\langle \ln P(\phi) |\, N \rangle + \frac{1}{2} \langle \phi - t \,|\mathbf{K}|\, \phi - t \rangle + \langle P(\phi) |\, \Lambda_X \rangle \\ &= -\langle \ln P(\phi) |\, N \rangle + \frac{1}{2} \langle \phi \,|\mathbf{K}|\, \phi \rangle - \langle J |\, \phi \rangle + \langle P(\phi) |\, \Lambda_X \rangle + c, \end{aligned} \tag{14.51}$$

with a *Lagrange multiplier function* $\Lambda_X(x, y) = \Lambda_X(x)$ which is determined by the normalization condition. A similar Lagrange multiplier term can be introduced for the non-negativity condition, but calculating the posterior predictive density in MAP approximation (see Eq. (14.10)) non–negativity terms are in many practical cases not necessary. Indeed, because the likelihood is by definition larger than zero at all data points, the field ϕ^* with maximal posterior can typically also not become smaller than zero between data points if we assume some common smoothness prior. Technically speaking, the non–negativity constraint is for smoothness priors typically not active at the stationary point ϕ^*. In the language of physics the term $J = \mathbf{K}t$ in Eq. (14.51) represents an *external field* coupling to $\phi(x, y)$, similar, for example, to a magnetic field. A non–zero field J leads to a non–zero expectation of ϕ in the no–data case. The ϕ–independent c stands for the term $\frac{1}{2} \langle t \,|\mathbf{K}|\, t \rangle$, which is for invertible $\mathbf{K}$ equal to $\frac{1}{2} \langle J \,|\mathbf{K}^{-1}|\, J \rangle$, and can be skipped when minimizing $E(\phi)$ with respect to ϕ.

A MAP solution ϕ^*, which maximizes the posterior $p(\phi|D, D_0)$ or, equivalently, minimizes the energy $E(\phi)$ can be found by setting the functional (Fréchet) derivative of the functional $E(\phi)$ in (14.51) with respect to the function $\phi(x, y)$ to zero, i.e., by solving the stationarity equations $\delta E(\phi)/\delta\phi(x, y) = 0$ for all x and y. The stationarity equations for ϕ resulting from (14.51) can easily be expressed in vector notation

$$0 = \mathbf{P}'(\phi)\mathbf{P}^{-1}(\phi)N - \mathbf{P}'(\phi)\Lambda_X - \mathbf{K}\,(\phi - t)\,, \tag{14.52}$$

where the Lagrange multiplier function follows from the normalization constraints over y for all x yielding, if $\mathbf{P}\mathbf{P}'^{-1}$ is invertible and $\Lambda_X \neq 0$,

$$\Lambda_X = \mathbf{I}_X \left(N - \mathbf{P}\mathbf{P}'^{-1}\mathbf{K}\,(\phi - t) \right), \tag{14.53}$$

with identity on X, $\mathbf{I}_X(x,y;x',y') = \delta(x-x')$ and diagonal likelihood operator

$$\mathbf{P}(x,y;x',y';\phi) = \delta(x-x')\delta(y-y')P(x,y;\phi), \tag{14.54}$$

with Jacobian

$$\mathbf{P}'(x,y;x',y';\phi) = \frac{\delta P(x',y';\phi)}{\delta\phi(x,y)}. \tag{14.55}$$

Choosing the regression function as field the stationarity equation (14.52) is linear for (nonparametric, nonlinear) regression problems provided the likelihood model and the prior model are Gaussian. In general, however, the stationarity equation (14.52) is nonlinear and has to be solved by iteration.

14.4.2 Hyperparameters and hyperfields

Hyperparameters and the boosting of parametric models

Hyperparameters are parameters of the prior [1, 2, 12, 6, 28], like the 'regularization factor' λ and the width σ_0 in Eq. (14.49). Introducing hyperparameters into a model means decomposing the prior according to

$$p(\phi|D_0) = \int d\theta\; p(\phi|\theta, D_0)p(\theta|D_0) \tag{14.56}$$

denoting hyperparameters collectively by θ. In many practical cases the integral over θ in Eq. (14.56) can not be calculated analytically; like the integral over ϕ in Eq. (14.8) it then has to be calculated numerically, e.g., by Monte Carlo methods, or in Maximum A Posteriori Approximation. In a joint MAP approximation the integral over ϕ in Eq. (14.8) and the integral over θ in Eq. (14.56) are calculated simultaneously in Maximum A Posteriori Approximation, resulting in coupled stationarity equations for ϕ and θ.

Hyperparameters can be used to adapt the mean function or the covariance of a Gaussian process prior. An example of the latter is the Automatic Relevance Detection by MacKay and Neal [32, 27, 26, 20]. Hyperparameters for the mean functions of a Gaussian process have been used, for example, in image completion tasks (see Fig. 14.5 and Fig. 14.6). The adaption of the mean function is technically simpler then changing the covariance because changing the mean of a Gaussian process does not change the normalization constant but changing the covariance normally does. Solving the coupled stationarity equations for θ and ϕ in a joint MAP approximation for adapting the mean function of a Gaussian prior can be interpreted as a *nonparametric boosting of a parametric model* [20] where during iteration first a parametric model $t(x,y:\theta)$ is optimized with respect to the (hyper)parameters θ and then the optimal parametric solution $t(x,y;\theta^*)$ is used as a mean function for the Gaussian prior for ϕ. Fig. 14.7 shows as an example the reconstruction of a quantum potential $v(x)$ using as reference potential a parametric solution $t(x;\theta^*)$ with a θ^* which is obtained in maximum likelihood approximation (i.e., in a MAP approximation with uniform (hyper)prior for θ).

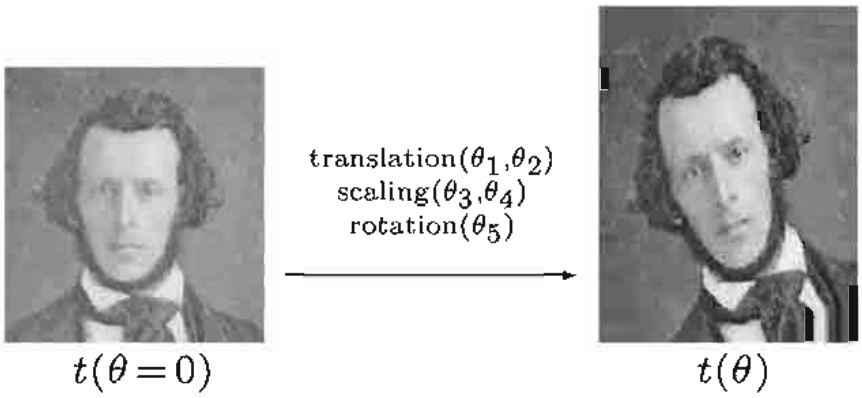


Figure 14.5: Example of a mean function $t(x,\theta)$ with a five–dimensional hyperparameter vector (and a two–dimensional x) used for a Gaussian prior as in Eq. (14.44) with prior energy $\frac{1}{2}\,\langle\phi - t\,|\mathrm{K}|\,\phi - t\rangle$. Such (graylevel) mean functions $t(x,\theta)$ have been used in image completion tasks (see Fig. 14.6) where the (graylevel) function $\phi(x)$ represents an image which has to be reconstructed from pixels sampled with Gaussian noise from the original image [20, 21]. The five dimensional hyperparameter vector θ which includes two translation, one rotation, and two scaling parameters makes the mean function t an adaptable template for the image ϕ. In a joint MAP approximation the function $\phi(x)$ and the hyperparameters θ are optimized simultaneously.

Local hyperfields and filtered differences

Under *local hyperfields* $\theta(x)$ we understand a collection of hyperparameters indexed by the visible variables x [20, 25]. It is straightforward to consider local hyperfields $\theta(x,y)$ depending on both, x and y, but for the sake of simplicity we restrict to fields $\theta(x)$, having in mind, for example, regression problems or the reconstruction of a quantum potential $v(x)$. Similarly, nonlocal hyperfields can be introduced depending, for example, on more than one x–value. Local hyperfields are useful to adapt the mean function or the inverse covariance of a Gaussian process locally. To show this, we decompose a real–symmetric, positive (semi–)definite inverse covariance $\mathrm{K}(x,x')$ (defining for example a regression problem) into square roots,

$$\mathrm{K} = \mathrm{W}^T\mathrm{W}, \tag{14.57}$$

i.e.,

$$\mathrm{K}(x,x') = \int dx''\; \mathrm{W}^T(x,x'')\mathrm{W}(x'',x'). \tag{14.58}$$

We will call the real square roots W *filter operators* and define the corresponding *filtered differences*

$$\omega(x) = \int dx'\; \mathrm{W}(x,x')[\phi(x') - t(x')]. \tag{14.59}$$

Using Eq. (14.59) the negative logarithm of a Gaussian prior like in Eq. (14.44) with mean function $t(x)$ and inverse covariance $\mathrm{K}(x,x')$ becomes, up to a ϕ–independent

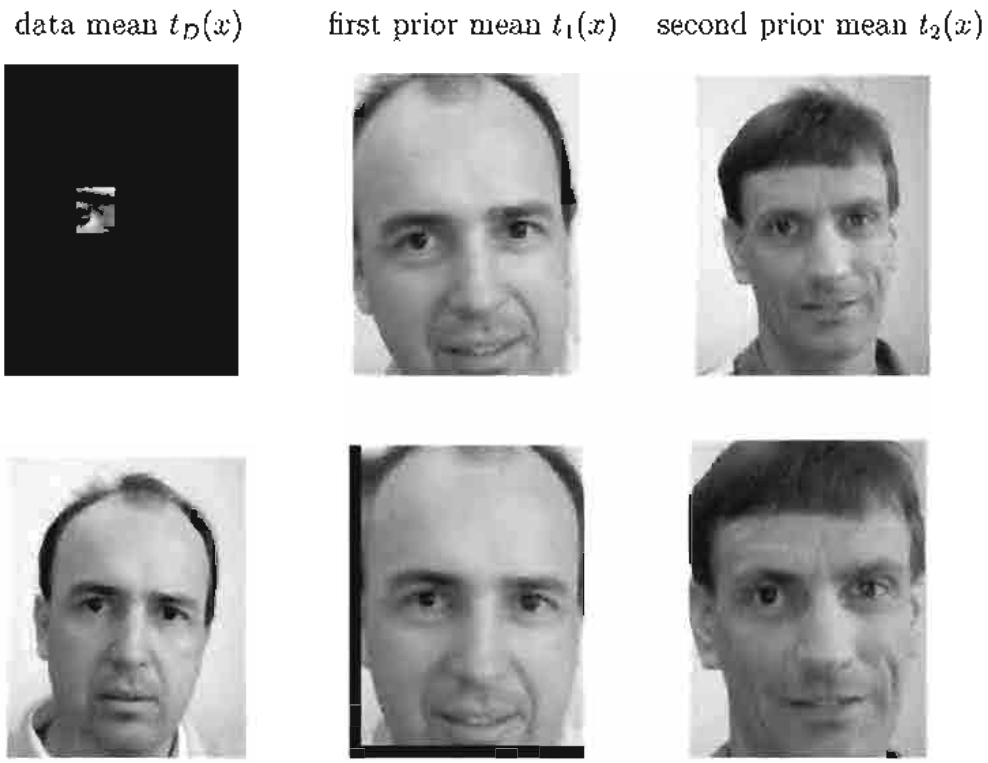


Figure 14.6: Image completion with a Gaussian mixture prior as an example of a regression problem [20, 21]. The data (first row on the left, for the definition of t_D see 14.46) have been sampled from the original image $\phi_{\text{true}}(x)$ (second row on the left) with Gaussian noise. The prior consists of a sum of two Gaussian priors $p(\phi|D_0) = 0.5p_1(\phi)+0.5p_2(\phi)$ both components with a negative Laplacian covariance but with two different mean functions (image templates) $t_1(x)$ and $t_2(x)$ as shown in the first row. Both mean functions t_i have been made flexible using the five dimensional hyperparameter vector introduced in Fig. 14.5. Shown is a situation where the mixture prior and also the posterior possesses two local maxima ('low temperature case'). The two regression functions $\phi_1(x)$ and $\phi_2(x)$ representing the local maxima of the posterior are shown in the second row, optimally scaled, shifted and rotated. The global maximum, which is $\phi_1(x)$, is the MAP reconstruction of the true regression function ϕ_{true}.

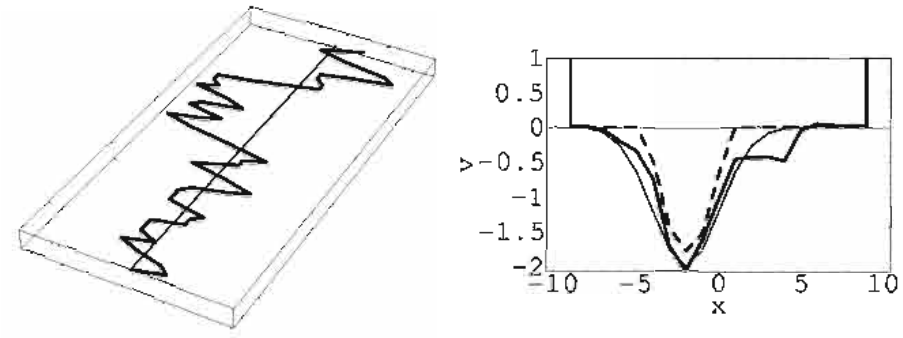


Figure 14.7: Left hand side: Shown is a sample path of a quantum particle, obtained from 50 coordinate measurements at fixed time intervals analogous to Fig. 14.3. Right hand side: An example of the numerical reconstruction of a quantum potential from the 50 coordinate measurements with likelihood as in (14.42) and a Gaussian prior on $v(x)$ similar to Eq. (14.49). (For more details see [22].) Shown are the true potential $v_{\text{true}}(x)$ (thin line), the best parametric approximation used as reference potential $t(x)$ (dashed line), and the reconstructed potential $\phi(x) = v(x)$ (thick line).

normalization term,

$$\begin{aligned} \frac{1}{2}\langle \phi - t \,|\mathbf{K}|\, \phi - t \rangle &= \frac{1}{2}\int dx\, dx'\, dx'' \,[\phi(x) - t(x)]\mathbf{W}^T(x,x') \\ &\qquad \times\ \mathbf{W}(x',x'')[\phi(x'') - t(x'')] \\ &= \frac{1}{2}\int dx\, |\omega(x)|^2. \end{aligned} \tag{14.60}$$

At this point we can introduce a local hyperfield by replacing the filtered difference $\omega(x)$ by a local mixture of two alternative filtered differences

$$\omega(x;\theta) = [1 - \theta(x)]\,\omega_1(x) + \theta(x)\,\omega_2(x), \tag{14.61}$$

with the real local hyperfield $\theta(x) \in [0,1]$ controlling the mixture of the two ω_i. Such a hyperfield $\theta(x)$ can adapt the prior by selecting locally the best mixture of the two filtered differences ω_i. Similarly, additional hyperfields can be introduced to mix more then two filtered differences.

One possibility to transform an unrestricted real hyperfield $-\infty \leq \widetilde{\theta}(x) \leq \infty$ into a bounded real hyperfield $\theta(x) \in [0,1]$ is given by

$$\theta(x) = \sigma(\widetilde{\theta}(x) - \vartheta), \tag{14.62}$$

with a threshold parameter ϑ and the sigmoidal transformation

$$\sigma(x) = \frac{1}{1 + \exp(-2\nu x)} = \frac{1}{2}(\tanh(\nu x) + 1). \tag{14.63}$$

In the limit $\nu \to \infty$ where the sigmoid in Eq. (14.63) approaches a step function we obtain a binary local hyperfield $\theta(x) \in \{0,1\}$. In contrast to a 'soft mixing' with real

functions with $0 \leq \theta(x) \leq 1$, a binary local hyperfield implements a 'hard switching' between alternative filtered differences ω_i.

For a prior depending on hyperfields

$$p(\phi|\theta) \propto \exp\left(-E(\phi|\theta)\right) \tag{14.64}$$

the prior energy $E(\phi|\theta)$ can be written

$$\begin{aligned} E(\phi|\theta) &= \frac{1}{2}\int dx |\omega(x;\theta)|^2 + \ln Z_\phi(\theta) \\ &= \frac{1}{2}\int dx \Big|[1-\theta(x)]\omega_1(x) + \theta(x)\omega_2(x)\Big|^2 + \ln Z_\phi(\theta). \end{aligned} \tag{14.65}$$

The normalization factor

$$Z_\phi(\theta) = \int d\phi \, \exp\left(-\frac{1}{2}\int dx |\omega(x;\theta)|^2\right) \tag{14.66}$$

can depend on θ, if the filters $\mathbf{W}_i$ of the ω_i differ. If depending on θ the normalization factor $Z_\phi(\theta)$ has to be included when integrating over θ or solving for the optimal θ in Maximum A Posteriori Approximation. For binary θ mixed terms proportional to $\omega_1\omega_2$ vanish in Eq. (14.65), because for binary θ we have $\theta(1-\theta) = 0$. In that case we can write

$$E(\phi|\theta) = \frac{1}{2}\int dx \left([1-\theta(x)]|\omega_1(x)|^2 + \theta(x)|\omega_2(x)|^2\right) + \ln Z_\phi(\theta). \tag{14.67}$$

Mixing reference functions

A filtered difference $\omega(x;\theta)$ as in Eq. (14.61) can be obtained, for instance, by local mixing or switching between two alternative reference functions $t_1(x')$ and $t_2(x')$

$$t_x(x';\theta) = [1-\theta(x)]\, t_1(x') + \theta(x)\, t_2(x'), \tag{14.68}$$

where the local reference functions $t_x(x';\theta)$ are functions of x' and x: To obtain a filtered difference $\omega(x;\theta)$ at position x, one needs the reference function t_x for all x' for which the corresponding $\mathbf{W}(x,x')$ is nonzero,

$$\omega(x;\theta) = \int dx' \, \mathbf{W}(x,x')[\phi(x') - t_x(x';\theta)]. \tag{14.69}$$

Thus, for every local filtered difference the whole template function $t_x(x';\theta)$, rather than individual function values $t(x,\theta)$, have to be adapted.

In contrast to the local reference functions $t_x(x')$ used in Eq. (14.68) one global reference function $t(x')$ can be adapted locally using

$$t(x';\theta) = [1-\theta(x')]\, t_1(x') + \theta(x')\, t_2(x'). \tag{14.70}$$

On the other hand, working with different reference functions $t_{1,x}(x')$, $t_{2,x}(x')$ for different x generalizes Eq. (14.68) to

$$t_x(x';\theta) = [1-\theta(x)]\, t_{1,x}(x') + \theta(x)\, t_{2,x}(x'). \tag{14.71}$$

It is possible to use a nonlocal prior (i.e., a prior with $\mathbf{K} \neq \mathbf{I}$, for example, when choosing as $\mathbf{K}$ a differential operator because smooth functions are expected) and still avoid local template functions $t_x(x', \theta)$. This can be achieved by taking the product of a Gaussian prior with $\mathbf{K} = \mathbf{I}$ and a second Gaussian prior with a nondiagonal inverse covariance $\mathbf{K}_2$ and a zero (or fixed) reference function, yielding for the combined prior energy

$$E(\phi|\theta) = \frac{1}{2} \langle \phi - \widetilde{t}(\theta) | \phi - \widetilde{t}(\theta) \rangle + \frac{1}{2} \langle \phi | \mathbf{K}_2 | \phi \rangle \tag{14.72}$$

$$= \frac{1}{2} \Big(\langle \phi - t(\theta) | \mathbf{K} | \phi - t(\theta) \rangle + \langle \widetilde{t}(\theta) | \mathbf{I} - \mathbf{K}^{-1} | \widetilde{t}(\theta) \rangle \Big), \tag{14.73}$$

The second term in Eq. (14.73) is independent of ϕ, the effective template $t(\theta)$ is given by

$$t(\theta) = \mathbf{K}^{-1} \widetilde{t}(\theta), \tag{14.74}$$

and the effective inverse covariance $\mathbf{K}$ by

$$\mathbf{K} = \mathbf{I} + \mathbf{K}_2. \tag{14.75}$$

Choosing as inverse prior covariance $\mathbf{K}$ a differential operator, the effective $t(\theta)$ becomes a smoothed version of $\widetilde{t}(\theta)$.

Mixing filter operators

Similar to Eq. (14.68) a mixed filtered difference $\omega(x; \theta)$ can be obtained by mixing locally two alternative filter operators $\mathbf{W}_1(x, x')$ and $\mathbf{W}_2(x, x')$

$$\mathbf{W}(x, x'; \theta) = [1 - \theta(x)] \mathbf{W}_1(x, x') + \theta(x) \mathbf{W}_2(x, x'). \tag{14.76}$$

Introducing

$$\mathbf{K}_x(\theta) = W_x(\theta) W_x^T(\theta) \tag{14.77}$$

with vector $W_x(\theta) = \mathbf{W}(x, \cdot\,; \theta)$, we obtain from Eq. (14.76) for binary $\theta(x)$ as inverse covariance

$$\mathbf{K}(\theta) = \int dx\, \mathbf{K}_x(\theta) = \int dx\, W_x(\theta) W_x^T(\theta)$$

$$= \int dx \left([1 - \theta(x)] W_{1,x} W_{1,x}^T + \theta(x) W_{2,x} W_{2,x}^T \right). \tag{14.78}$$

14.4.3 Hyperpriors for hyperfields

Working with hyperfields typically requires non–uniform *hyperpriors* $p(\theta)$. Indeed, allowing completely unrestricted functions t and operators $\mathbf{W}$ just eliminates the corresponding prior term. In nonparametric hyperfield models, where functions like $\theta(x)$ are not restricted to some parametric family, hyperpriors for hyperfields are stochastic processes, like priors for the functions $\phi(x)$ or $\phi(x, y)$. Such hyperpriors can, for instance, favor smooth hyperfields $\theta(x)$ analogous to a smoothness prior for a function ϕ. In analogy to (Euclidean) field theory in physics the part $E(\phi|\theta)$ may be interpreted

as the 'interaction' between the fields ϕ and θ while the hyperprior describes a 'free' hyperfield including possible 'self–interactions' of the hyperfield.

As a smoothness prior for a real $\theta(x)$, for example, one can use a Laplacian prior with *hyperprior energy*

$$E(\theta) = -\frac{\tau}{2} \langle \theta | \Delta | \theta \rangle . \tag{14.79}$$

where τ is the analogue of a regularization constant. Parameters of the hyperprior like τ in Eq. (14.80) can be treated as higher level hyperparameters.

Another kind of 'smoothness' is implemented by the non–Gaussian hyperprior

$$p(\theta) \propto \exp\left(-\frac{\tau}{2} \int dx\, C_\theta(x)\right), \tag{14.80}$$

where

$$C_\theta(x) = \sigma\left(\left(\frac{\partial\theta}{\partial x}\right)^2 - \vartheta_\theta\right) \tag{14.81}$$

with a sigmoid function $\sigma(x)$ like in (14.63) and some constant τ. For $\nu \to \infty$, where the sigmoid approaches a step function, $C_\theta(x)$ becomes 0 at locations where the square of the first derivative is smaller than a certain threshold $0 \leq \vartheta_\theta < \infty$, and 1 otherwise. Similarly, one can penalize the number $N_d(\theta)$ of discontinuities, where $(\partial\theta/\partial x)^2 = \infty$, choosing

$$p(\theta) \propto \exp\left(-\frac{\tau}{2} N_d(\theta)\right). \tag{14.82}$$

In the case of a binary field this means counting the number of times the field changes its value.

Eq. (14.81) can be generalized to

$$C_\theta(x) = \sigma\left(|\omega_\theta(x)|^2 - \vartheta_\theta\right), \tag{14.83}$$

with filtered difference

$$\omega_\theta(x) = \int dx'\, \mathbf{W}_\theta(x, x')[\theta(x') - t_\theta(x')], \tag{14.84}$$

like in Eq. (14.59). The reference function $t_\theta(x')$ in Eq. (14.84) gives the expected form for the hyperfield $\theta(x)$, while the filter operator $\mathbf{W}_\theta$ defines the measure which is used to measure the distance of hyperfields $\theta(x)$ from the reference $t_\theta(x')$.

14.4.4 Auxiliary fields

When working with hyperfields one introduces additional degrees of freedom which influence the prior for ϕ. Integrating over the hyperfields $\theta(x)$ to obtain the predictive density in a full Bayesian approach would leave us with a prior $p(\phi)$ which is non–Gaussian in ϕ, even if all conditional priors $p(\phi|\theta)$ are Gaussian in ϕ. Similarly, when solving the problem in Maximum A Posteriori Approximation, the stationarity equation for ϕ (linear for Gaussian $p(\phi|\theta)$ for given θ) and the nonlinear stationarity equation for θ are coupled. Eliminating θ from the set of coupled equations leaves us with a

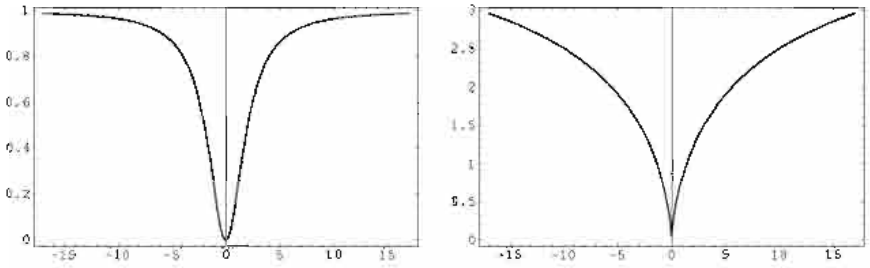


Figure 14.8: Two 'cup' functions of the form $\psi(x) = a(1.0 - 1/(1 + (|x - x_0|/b)^\gamma))$ [45] as example of a non–quadratic function $\psi(x)$ in Eq. (14.85). Left hand side: Winkler's cup function [44] ($a = 5$, $b = 10$, $\gamma = 0.7$, $x_0 = 0$). Right hand side: Function with cusp ($a = 1$, $b = 3$, $\gamma = 2$, $x_0 = 0$).

nonlinear equation in ϕ. Hence, instead of introducing hyperfields as additional degrees of freedom one may try as well to directly formulate a non-Gaussian prior for ϕ.

One possibility to obtain a non–Gaussian prior from a Gaussian prior is to use as prior energy instead of the square ω^2 a non–quadratic function $\psi(\omega)$ of the filtered difference, corresponding to a prior

$$p(\phi|D_0) \propto \exp\left(-\int dx\, \psi(\omega(x))\right) \tag{14.85}$$

where for density estimation problems x can be replaced by the pair (x, y). Typical choices are 'cup' functions with flat tails for which one large step is cheaper than many small ones (see Fig. 14.8). Such non–Gaussian priors are, for example, used to deal with discontinuities in images [14, 3, 31, 13, 46, 45] or in the identification of 'outliers' like the separation of background and signal in experimental spectra [39, 9, 10].

Another possibility to construct non–Gaussian priors is the introduction of *auxiliary fields* $B(x;\phi)$, or more general $B(x, y;\phi)$, whose function values are not independent variables but are defined as functionals of ϕ. (To simplify notation we will denote $B(x;\phi)$ by $B(x)$ or $B(\phi)$, depending on the context.) Similar to hyperfields $\theta(x)$ auxiliary fields can be used to select locally the best adapted filtered difference from a set of alternative ω_i, each of the form of Eq. (14.59). As an example in analogy to Eq. (14.83) consider an auxiliary field

$$B(x) = \sigma\left(u(x) - \vartheta\right), \tag{14.86}$$

with

$$u(x) = |\omega_1(x)|^2 - |\omega_2(x)|^2, \tag{14.87}$$

ϑ representing a threshold, and $\sigma(x)$ a sigmoidal function like in (14.63). The ω_i are filtered differences defined in terms of ϕ analogous to Eq. (14.59). In contrast to a hyperfield the auxiliary field $B(x)$ in Eq. (14.86) does not introduce new degrees of freedom because the ω_i in Eq. (14.87) are defined in terms of ϕ. Note that if $\omega_i(x)$ is nonlocal with respect to $\phi(x)$ then also $B(x)$ is nonlocal, meaning that a value $B(x)$ depends on more than one $\phi(x)$–value. For a negative Laplacian prior in one–dimension, i.e.,

$$\mathbf{K}(x, x') = -\delta(x, x')\frac{\partial^2}{\partial x^2} \tag{14.88}$$

(the δ-function is usually skipped from the notation) Eq. (14.86) becomes (for appropriate boundary conditions)

$$B(x) = \sigma\left(\left|\frac{\partial(\phi - t_1)}{\partial x}\right|^2 - \left|\frac{\partial(\phi - t_2)}{\partial x}\right|^2 - \vartheta\right). \tag{14.89}$$

While auxiliary fields $B(x)$ are directly determined by ϕ, hyperfields $\theta(x)$ are indirectly related to ϕ, for instance through the stationarity equations of a Maximum A Posteriori Approximation or through integration over $\theta(x)$ in a full Bayesian approach.

Auxiliary fields can be used similarly to hyperfields. They can help to adapt reference functions t or filter operators $\mathbf{W}$. For instance, to switch between two filtered differences one can use a binary $B(x)$

$$|\omega(x;B)|^2 = [1 - B(x)]|\omega_1(x)|^2 + B(x)|\omega_2(x)|^2, \tag{14.90}$$

yielding a prior energy of a form similar to Eq. (14.67)

$$E(\phi) = \frac{1}{2}\int dx\ \left([1 - B(x)]|\omega_1(x)|^2 + B(x)|\omega_2(x)|^2\right). \tag{14.91}$$

Resembling the role of hyperpriors $p(\theta)$, additional prior factors

$$p(B(\phi)) \propto \exp(-E_B(\phi)), \tag{14.92}$$

depending on ϕ only over $B(\phi)$, can be introduced. For example, if $N_d(B)$ counts the number of discontinuities of $B(x)$, the number of switchings is restricted by choosing

$$E_B(\phi) = \frac{\tau}{2}N_d(B). \tag{14.93}$$

For real $B(x)$ one can use terms of the form

$$E_B(\phi) = \frac{\tau}{2}\int dx|\omega_B(x)|^2 \tag{14.94}$$

where

$$\omega_B(x) = \int dx'\,\mathbf{W}_B(x,x')[B(x') - t_B(x')] \tag{14.95}$$

is a filtered difference of B with some filter operator $\mathbf{W}_B$ and template t_B. Non-quadratic energies as in (14.83) become, now written for B,

$$E_B(\phi) = \frac{\tau}{2}\int dx\, C_B(x), \tag{14.96}$$

with

$$C_B(x) = \sigma\left(|\omega_B(x)|^2 - \vartheta_B\right). \tag{14.97}$$

The normalization factor

$$Z = \int d\phi\, \exp\left(-E(\phi) - E_B(\phi)\right) \tag{14.98}$$

for a prior

$$p(\phi) \propto \exp\left(-E(\phi) - E_B(\phi)\right), \tag{14.99}$$

is by definition independent of ϕ and can thus be skipped for calculations in Maximum A Posteriori Approximation.

We have now encountered two methods for constructing prior models which may result in quite similar looking expressions. For instance, combining prior energy (14.91) with (14.93) for a binary auxiliary field (14.86) results in a prior

$$p(\phi) \propto \exp\left(-\frac{1}{2}\int dx\, \left([1-B(x)]|\omega_1(x)|^2 + B(x)|\omega_2(x)|^2\right) - \frac{\tau}{2}N_d(B)\right). \tag{14.100}$$

A similar looking prior model with a binary hyperfield can be obtained by combining a conditional Gaussian prior (14.67) with the hyperprior (14.82)

$$p(\phi,\theta) = p(\phi|\theta)p(\theta) \propto \tag{14.101}$$

$$\exp\left(-\frac{1}{2}\int dx\, \left[(1-\theta(x))|\omega_1(x)|^2 + \theta(x)|\omega_2(x)|^2\right] - \frac{\tau}{2}N_d(\theta) - \ln Z_\phi(\theta)\right).$$

We can now compare the two models: Working with hyperfields means working with conditional priors $p(\phi|\theta)$, so that normalization factors which are in general θ–dependent. Therefore the normalization factors have to be included for calculations in Maximum A Posteriori Approximations. This is not the case if we are working with auxiliary fields. Hence, in general MAP solutions for B, $N_d(B)$, and C_B, are different from the MAP solutions for θ, $N_d(\theta)$, and C_θ. If, however, the filtered differences ω_i in Eq. (14.101) differ only in their reference functions $t_i(x)$, then the normalization term can be skipped. The two MAP equations are then indeed equivalent for $\theta(x) = \Theta\left(|\omega_1(x)|^2 - |\omega_2(x)|^2\right)$, if the threshold vanishes $\vartheta = 0$ and a hyperprior or additional p_B is absent, i.e., if $p(\theta) \propto 1$ and $p(B) \propto 1$. Furthermore it is also easily seen that in that case $\theta(x) = \Theta\left(|\omega_1(x)|^2 - |\omega_2(x)|^2\right)$ is a selfconsistent solution for θ for every given ϕ. Fig. 14.9 shows two reconstructions of a quantum potential, one using a hyperfield and another one using an auxiliary field.

14.5 Summary

In nonparametric Bayesian models, where the number of the degrees of freedom of the likelihood is much larger than the number of available data points, the quality of learning does depend essentially on the implemented *a priori* information. Starting from Gaussian process priors we have discussed several methods to implement available *a priori* information explicitly in terms of the likelihood, for example, by choosing a specific prior covariance which corresponds to some approximate symmetry of the likelihood or by adapting the mean function of a prior process using hyperfields. Those techniques for constructing nonparametric priors can be used for many different likelihood models, including density estimation, regression and inverse quantum theory.

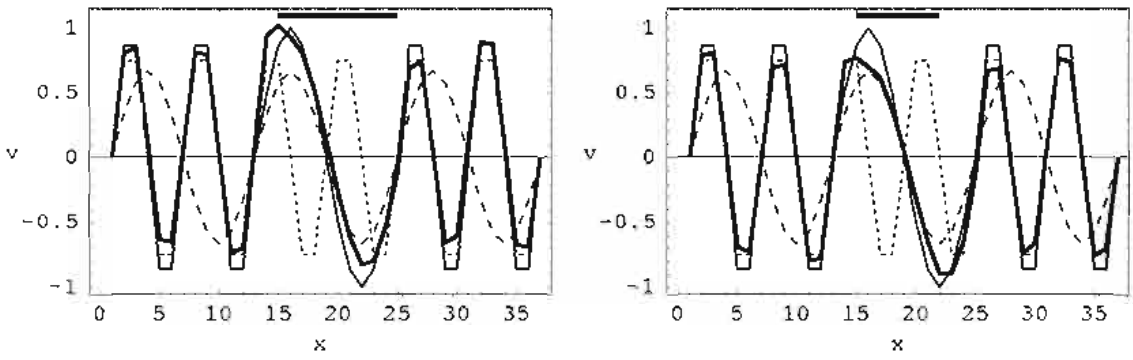


Figure 14.9: Example of the reconstruction of a potential $v(x)$ from coordinate measurements in a quantum system at finite temperature with likelihood (14.41) and with a binary auxiliary field (left figure) or a binary hyperfield (right figure), respectively, switching locally between two alternative mean functions (or reference potentials) $t_1(x) = v_1(x)$ and $t_2(x) = v_2(x)$ of a nonparametric Gaussian smoothness prior [25]. In both figures the switching between the two reference potentials v_1, v_2 (induced by the binary hyperfield or the binary auxiliary field, respectively) is indicated by a thick black line above the potentials. Left hand side: Reconstruction with a binary auxiliary field defined in terms of $\phi(x) = v(x)$ as step function $B(x) = \Theta\left(|\omega_1(x)|^2 - |\omega_2(x)|^2\right)$, (which corresponds to Eq. (14.86) with $\vartheta = 0$ in the limit where the sigmoid approaches a step function), using a prior like in (14.91) switching between two alternative filtered differences and a penalty on the number of jumps of the auxiliary field. Right hand side: Reconstruction with a binary hyperfield which switches locally between the two nonzero reference potentials v_1, v_2 and a penalty on the number of jumps of the hyperfield as hyperprior.

Bibliography

[1] J. O. Berger, *Statistical Decision Theory and Bayesian Analysis*, Springer–Verlag, New York, 2nd edition (1985).

[2] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford (1995).

[3] A. Blake and A. Zisserman, *Visual Reconstruction*, MIT Press, Cambridge, MA (1987).

[4] N. Bleistein and N. Handelsman, *Asymptotic Expansions of Integrals*, Dover, New York (1986). Originally published in 1975 by Holt, Rinehart, and Winston, New York.

[5] J. Cardy, *Scaling and Renormalization in Statistical Physics*, Cambridge University Press, Cambridge (1996).

[6] B. P. Carlin and T. A. Louis, *Bayes and Empirical Bayes Methods for Data Analysis*, volume 69 of *Monographs on Statistics and Applied Probability*, Chapman & Hall/CRC, Boca Raton (1996).

[7] N. G. de Bruijn, *Asymptotic Methods in Analysis*, Dover, New York (1981). Originally published by North–Holland Publishing Co., 1958, Amsterdam.

[8] J. L. Doob, *Stochastic Processes*, Wiley, New York, 1953. New edition published (1990).

[9] V. Dose and W. von der Linden, Outlier tolerant parameter estimation, In W. von der Linden, V. Dose, R. Fischer, and R. Preuss, editors, *Maximum Entropy and Bayesian Methods*, Dordrecht, Kluwer Academic Publishers (1999)

[10] R. Fischer, K. M. Hanson, V. Dose, and W. von der Linden, Background estimation in experimental spectra, *Phys. Rev. E* **61** (2000) 1152.

[11] C. W. Gardiner, *Handbook of Stochastic Methods*, Springer–Verlag, Berlin, 2nd edition (1990).

[12] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian Data Analysis*, Chapman & Hall, New York (1995).

[13] D. Geman and G. Reynolds, Constraint restoration and the recover of discontinuities, *IEEE Trans. on Pattern Analysis and Machine Intelligence* **14** (1992) 367–383.

[14] S. Geman and D. Geman, Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, *IEEE Trans. on Pattern Analysis and Machine Intelligence* **6** (1984) 721–741.

[15] F. Girosi, M. Jones, and T. Poggio, Regularization theory and neural network architectures, *Neural Computation* **7**(2) (1995) 219–269.

[16] J. M. Hammersley and D. C. Handscomb, *Monte Carlo Methods*, volume 3 of *Monographs on Statistics and Applied Probability*, Chapman & Hall, London (1964).

[17] J. Honerkamp, *Stochastic Dynamical Systems. Concepts, Numerical Methods, Data Analysis*, VCH Verlag, New York (1994).

[18] J. Honerkamp, *Statistical Physics*, Springer–Verlag, New York (1998).

[19] D. P. Landau and K. Binder, *A Guide to Monte Carlo Simulations in Statistical Physics*. Cambridge University Press, Cambridge (2000).

[20] J. C. Lemm, Bayesian field theory, Technical Report MS-TP1-99-1, Universität Münster (1999). arXiv:physics/9912005.

[21] J. C. Lemm, Mixtures of Gaussian process priors, In *ICANN 99. Proceedings of the 9th International Conference on Artificial Neural Networks, Edinburgh, UK, September 7–10, 1999*, London, IEEE Conference Publication (1999)

[22] J. C. Lemm, Inverse time–dependent quantum mechanics. *Phys. Lett. A* **276**(1–4) (2000) 19–24. arXiv:quant-ph/0002010.

[23] J. C. Lemm, Bayesian field theory, The Johns Hopkins University Press, Baltimore, MD (to appear 2003).

[24] J. C. Lemm and J. Uhlig, Bayesian inverse quantum theory, *Few–Body Systems* **29** (2000) 25–52. arXiv:quant-ph/0006027.

[25] J. C. Lemm, J. Uhlig, and A. Weiguny, Bayesian reconstruction of approximately periodic potentials for quantum systems at finite temperatures, *The European Physical Journal B* **20**(3) (2001) 349–366. MS-TP1-00-4, arXiv:quant-ph/0005122.

[26] D. J. C. MacKay, Bayesian methods for backpropagation networks, In E. Domany, J. L. van Hemmen, and K. Schulten, editors, *Models of Neural Networks III*, Springer–Verlag, New York (1996) 211–254.

[27] D. J. C. MacKay, Bayesian non–linear modeling for the prediction competition, In G. Heidbreder, editor, *Maximum Entropy and Bayesian Methods, Santa Barbara 1993*, Dordrecht, Kluwer Academic Publishers (1996) 221–234.

[28] D. J. C. MacKay, Comparison of approximate methods for handling hyperparameters, *Neural Computation* **11** (1999) 1035–1068.

[29] A. Messiah, *Quantum Mechanics*, North–Holland, Amsterdam (1961).

[30] I. Montvay and G. Münster, *Quantum Fields on a Lattice*, Cambridge University Press, Cambridge (1994).

[31] D. Mumford and J. Shah, Optimal approximations by piecewise smooth functions and associated variational problems, *Comm. Pure Applied Math.* **42** (1989) 577–684.

[32] R. M. Neal, *Bayesian Learning for Neural Networks*, Springer–Verlag, New York (1996).

[33] R. M. Neal, Monte Carlo implementation of Gaussian process models for Bayesian regression and classification, Technical Report 9702, Department of Statistics, University of Toronto, Canada (1997).

[34] J. W. Negele and H. Orland, *Quantum Many–Particle Systems*, volume 68 of *Frontiers in Physics Series*. Addison–Wesley, Redwood City, CA (1988).

[35] C. P. Robert, *The Bayesian Choice*, Springer–Verlag, New York (1994).

[36] B. Schölkopf and A. Smola, *Learning with Kernels*, MIT Press, Cambridge, MA (2001).

[37] N. G. van Kampen, *Stochastic Processes in Physics and Chemistry*, North–Holland, Amsterdam (1992).

[38] V. N. Vapnik, *Statistical Learning Theory*, Wiley, New York (1998).

[39] W. von der Linden, V. Dose, J. Padayachee, and V. Prozesky. Signal and background separation, *Phys. Rev. E* **59** (1999) 6527–6534.

[40] J. von Neumann, *Mathematical Foundations of Quantum Mechanics*, Princeton University Press, Princeton, NJ (1955).

[41] G. Wahba, *Spline Models for Observational Data*, SIAM, Philadelphia (1990).

[42] G. Wahba, Support vector machines, reproducing kernel Hilbert spaces and the randomized GACV, In B. Schölkopf, C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods: Support Vector Machines*, MIT Press, Cambridge, MA (1998) 69–88.

[43] C. K. I. Williams and C. E. Rasmussen, Gaussian processes for regression, In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, Cambridge, MA, MIT Press (1996) 514–520.

[44] G. Winkler, *Image Analysis, Random Fields, and Dynamic Monte Carlo Methods*, Springer–Verlag, Berlin (1995).

[45] S. C. Zhu and D. Mumford, Prior learning and Gibbs reaction–diffusion, *IEEE Trans. on Pattern Analysis and Machine Intelligence* **19**(11) (1997) 1236–1250.

[46] S. C. Zhu, Y. N. Wu, and D. Mumford, Minimax entropy principle and its application to texture modeling, *Neural Computation* **9**(8) (1997) 1627–1660.

Chapter 15

Bayesian Smoothing and Information Geometry

Rudolf Kulhavý[1]

Abstract. Local, cased-based modeling offers a natural way of capturing the complex behavior of data. As such, it has been a subject of intensive research in computational statistics, machine learning and system identification. Also, it has been applied successfully to numerous problems in different fields. Yet, the very concept of smoothing continues to be perceived as somewhat heuristic. The purpose of this chapter is to help understand better the connection of smoothing algorithms to Bayesian statistics and to present a natural geometry of local modeling.

[1]The author's research has been supported in part by the Grant Agency of the Czech Republic through Grant 102/01/0021. The support is gratefully acknowledged.

15.1 Introduction

Local modeling is an intuitively appealing paradigm of learning from data. At its root is the observation (supported by the everyday life experience) that one does not need to build a global model in order to predict response in a particular case. In fact, one can even improve prediction when minimizing the local error only, using a simpler model, even though estimated from less data.

Local models capture easily a nonlinear behavior, cope with bias problems at boundaries and in regions of high curvature, handle naturally multiple-mode data, adapt to changes in the data behavior, need only a fraction of historical data to work, scale up well to huge datasets, can be estimated using closed-form algorithms, and, last but not least, are easy to understand and interpret.

Such attractive properties do not come for free. Rather than arriving at a single globally valid model, the user ends up with multiple *ad hoc* models of varying quality, depending largely on the amount of data available for particular cases. Local modeling requires all historical data available on demand, involves a database-intensive step of retrieving relevant data, needs data organized properly for quick retrieval, requires careful tuning of bandwidth parameters for optimum data fit, and may suffer of the curse of dimensionality. The steady increase in the computer and database performance is removing some of the earlier technical hurdles, yet large-scale applications of local modeling are still far from routine.

As many good ideas in science and engineering, local modeling has been long a recurrent concept, discovered or rediscovered more or less independently by different research communities. In *computational statistics*, local modeling has been studied within the frameworks of data smoothing, local fitting, locally weighted regression and classification, kernel-based methods, and non-parametric estimation [6, 7, 11]. In *machine learning*, it has been known as local learning, case-based reasoning, example-based reasoning, memory-based learning, instance-based learning, or lazy learning [1, 4, 3]. In *system identification*, local modeling has been explicitly present in the concepts of just-in-time learning and on-demand modeling [10, 20].

In spite of much research attention and practical usage and solid understanding of the structure of smoothing algorithms [21], many conceptual questions remain. Is data smoothing a technique or method? Can the smoothing formulae be derived from a more general principle? What Bayesian interpretation can be given to the local modeling? Can one build a local modeling theory without referring to an underlying global model? What sort of geometry does the local modeling give rise to?

In the following, we try to shed some light on some of these questions. In particular, we demonstrate that the smoothing algorithms can be derived by approximation of the Bayesian estimation, with a specific choice of prior distribution over a set of local, cased-based models. In addition, we show a natural geometry of local modeling, based on measuring information carried by the empirical density relative to the model-based conditional densities.

15.2 Problem Statement

Our focus will be on fitting a model to the historical data. This is a problem narrower in scope than data modeling in large, which includes the data preprocessing and model selection tasks.

Data Transforms. The data definition and model selection proceeds typically in three steps.

First, we recognize directly manipulated inputs u_k to the underlying system at time $k = 1, \ldots, N$ and distinguish them from the outputs y_k of the system at the same time. The outputs represent the response of the system to the past history of data $y^{k-1} = (y_{k-1}, \ldots, y_1)$ and $u^k = (u_k, \ldots, u_1)$. At this stage, we consider a dynamic system that can be described through the (stochastic) functional relationship

$$y_k = F(u^k, y^{k-1}), \quad k = 1, \ldots, N\,.$$

Second, we introduce a vector of auxiliary variables x_k, which capture the system dynamics. A popular time-series model defines x_k-entries through the time-lagged values $u_k, u_{k-1}, \ldots$ and $y_{k-1}, y_{k-2}, \ldots$. Other functions of the data history u^k, y^{k-1}, such as time aggregates or dynamically filtered values, can be used as well. The objective of the second phase is to turn the original dynamic system into a static (still stochastic) one

$$y_k = f(x_k), \quad k = 1, \ldots, N\,.$$

Third, the data vector x can be mapped onto a feature vector $\phi_k = \phi(x_k)$, possibly of much higher dimension. The purpose of this step is to come up with a simpler, parametric representation of the map $f(\cdot)$. A particular example of such a parametric model is the linear regression, for scalar response y_k,

$$y_k = \theta^T \phi_k + \varepsilon_k\,.$$

Here θ stands for the vector of regression coefficients and ε_k accounts for the unpredictable component of the model.

Data Set. We assume that the data is available in a table composed of x_k and y_k values for $k = 1, \ldots, N$

$$\left|\begin{array}{ccc|ccc} x_{1,1} & \cdots & x_{1,m} & y_{1,1} & \cdots & y_{1,n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,m} & y_{N,1} & \cdots & y_{N,n} \end{array}\right|\,.$$

The x-entries represent predictors or explanatory variables whereas y-entries stand for responses or target variables. In general, the predictors and responses can be continuous or discrete or mixed (e.g., when forecasting the energy load and price tariff as a function of time of day and day of week).

We use the term *case* when speaking of a specific value of the vector x.

Conventions. In order to cover the cases of continuous and discrete random variables with a single notation, we consider each random variable Z with values in $\mathcal{Z}$ and probability distribution P on a measurable space $(\mathcal{Z}, \mathbf{Z})$ described by the density (Radon-Nikodym derivative) of P with respect to a dominating (Lebesgue or counting or product) measure μ

$$p(z) = \frac{P(\mathrm{d}z)}{\mu(\mathrm{d}z)} .$$

In the sequel, we do not mention the measurable space and the probability distribution. When using the same symbol for the densities p and dominating measures μ, we always include the argument to identify them uniquely. The reader not familiar with the measure theory can simply replace $\int p(z)\,\mu(\mathrm{d}z)$ for continuous and discrete variables z with ordinary integration and summation, respectively.

Objective. The chapter adopts a statistical perspective of learning. The data composed of responses observed under particular cases

$$y_1|x_1, \ldots, y_N|x_N$$

is supposed to be a sample from the conditional density

$$p(y|x, \theta)$$

where θ stands for the unknown parameters.

The objective is to estimate the conditional density from the sample

$$y_1|x_1, \ldots, y_N|x_N \rightarrow \widehat{p}_N(y|x) .$$

15.3 Probability-Based Inference

The essence of the Bayesian approach to parameter estimation and response prediction is the symmetrical treatment of stochastic data and uncertain parameters. Both stochastic and uncertain quantities are dealt with as random variables.

Joint Density. The starting point for derivation of the conditional density of the unknown parameters is to express the joint density $p(y^N, x^N, \theta)$ of data and parameters in terms of model assumptions. A recursive application of the chain rule makes it possible to decompose the joint density as follows

$$p(y^N, x^N, \theta) = \prod_{k=1}^{N} p(y_k|x_k, y^{k-1}, x^{k-1}, \theta)\, p(x_k|y^{k-1}, x^{k-1}, \theta)\, p(\theta) .$$

By the model assumption, the response y_k at any time k depends on the past data only through the current predictor x_k, i.e.,

$$p(y_k|x_k, y^{k-1}, x^{k-1}, \theta) = p(y_k|x_k, \theta) .$$

The joint density thus simplifies to

$$p(y^N, x^N, \theta) = \prod_{k=1}^{N} p(y_k|x_k, \theta)\, p(x_k|y^{k-1}, x^{k-1}, \theta)\, p(\theta)\,.$$

Furthermore, we assume that the predictor x_k at any time k is independent of θ given the past data

$$p(x_k|y^{k-1}, x^{k-1}, \theta) = p(x_k|y^{k-1}, x^{k-1})\,.$$

Under this assumption, introduced as *natural conditions of control* in [18], the joint density takes the form

$$p(y^N, x^N, \theta) = \prod_{k=1}^{N} p(y_k|x_k, \theta)\, p(x_k|y^{k-1}, x^{k-1})\, p(\theta)\,. \tag{15.1}$$

Posterior Density. Now, let us apply the chain rule in the other direction

$$p(y^N, x^N, \theta) = p(\theta|y^N, x^N)\, p(y^N, x^N)\,. \tag{15.2}$$

Combining the expressions (15.1) and (15.2), we obtain

$$p(\theta|y^N, x^N)\, p(y^N, x^N) = \prod_{k=1}^{N} p(y_k|x_k, \theta)\, p(x_k|y^{k-1}, x^{k-1})\, p(\theta)\,.$$

¿From this, the *posterior* density of the random variable Θ conditioned on y^N, x^N follows easily

$$p(\theta|y^N, x^N) \propto p(\theta) \prod_{k=1}^{N} p(y_k|x_k, \theta)\,.$$

Here $\propto$ stands for proportionality, i.e., equality up to a normalizing constant.

After introducing a short-cut notation

$$p_0(\theta) \triangleq p(\theta),\;\; p_N(\theta) \triangleq p(\theta|y^N, x^N),\;\; s_\theta(y|x) \triangleq p(y|x, \theta)\,,$$

we obtain the posterior density formula

$$p_N(\theta) \propto p_0(\theta) \prod_{k=1}^{N} s_\theta(y_k|x_k)\,. \tag{15.3}$$

Likelihood Function. The conditional density of the observed data taken as a function of the unknown parameter for given data is known as a *likelihood function*

$$l_N(\theta) = \prod_{k=1}^{N} s_\theta(y_k|x_k)\,. \tag{15.4}$$

Using the likelihood function, the posterior density can be rewritten in a compact form

$$p_N(\theta) \propto p_0(\theta)\, l_N(\theta)\,.$$

Predictive Density. The unknown parameter can be eliminated (integrated out) using probability calculus rules

$$p(y|x, y^N, x^N) = \int p(y|x,\theta)\, p(\theta|y^N, x^N)\, \mu(\mathrm{d}\theta)\,.$$

Using a simpler notation

$$s_N(y|x) \stackrel{\triangle}{=} p(y|x, y^N, x^N)\,,$$

we can rewrite the predictive density as follows

$$s_N(y|x) = \int s_\theta(y|x)\, p_N(\theta)\, \mu(\mathrm{d}\theta)\,. \tag{15.5}$$

Example. Consider a linear normal regression model

$$Y_k = \theta^T \phi(X_k) + E_k, \quad E_k \sim N(0, \sigma^2)\,.$$

The vector of regression coefficients θ is the unknown parameter of the model. The variance σ^2 is considered known for simplicity.

The conditional density of Y given $X = x$ is

$$s_\theta(y|x) = (2\pi\sigma^2)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2\sigma^2}(y - \theta^T\phi(x))^2\right\}\,.$$

The likelihood function (15.4) for a sample y^N, x^N takes the form

$$\begin{aligned} l_N(\theta) &= \prod_{k=1}^{N} (2\pi\sigma^2)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2\sigma^2}(y_k - \theta^T\phi(x_k))^2\right\} \\ &= (2\pi\sigma^2)^{-\frac{N}{2}} \exp\left\{-\frac{1}{2\sigma^2}\sum_{k=1}^{N}(y_k - \theta^T\phi(x_k))^2\right\}, \end{aligned}$$

which can be rewritten as

$$l_N(\theta) = (2\pi\sigma^2)^{-\frac{N}{2}} \exp\left\{-\frac{1}{2\sigma^2}\, N\, V_N\right\} \exp\left\{-\frac{1}{2\sigma^2}\, N\, (\theta - \widehat{\theta}_N)^T C_N\, (\theta - \widehat{\theta}_N)\right\}$$

with the statistics

$$\widehat{\theta}_N = C_N^{-1}\, \mathbf{E}_N(\Phi Y)\,, \tag{15.6}$$

$$V_N = \mathbf{E}_N(Y^2) - \mathbf{E}_N(Y\Phi^T)\, C_N^{-1}\, \mathbf{E}_N(\Phi Y)\,, \tag{15.7}$$

$$C_N = \mathbf{E}_N(\Phi\Phi^T)\,. \tag{15.8}$$

where $\Phi = \phi(X)$ and $\mathbf{E}_N(\cdot)$ denotes the empirical mean.

15.4 Information-Based Inference

We will show that the Bayesian inference implicitly measures the amount of information carried by the data relative to particular models parametrized by θ.

Empirical Density. For a given sample y^N, x^N, let us define the *joint empirical density* of (Y, X) as

$$r_N(y,x) = \frac{1}{N}\sum_{k=1}^{N} \delta(y - y_k, x - x_k)$$

where $\delta(y - y_k, x - x_k)$ stands for the density of a point-mass distribution at (y_k, x_k), with the properties

$$\delta(y - y_k, x - x_k) = 0 \text{ if } y \neq y_k \text{ or } x \neq x_k$$

and

$$\iint f(y,x)\,\delta(y - y_k, x - x_k)\,\mu(\mathrm{d}y)\,\mu(\mathrm{d}x) = f(y_k, x_k)$$

for all integrable f.

Let $\mathcal{X}_N \subset \mathcal{X}$ be the set of all distinct cases observed in the sample $x_1, \ldots, x_N$. We denote the empirical and model-based densities of Y for a particular case $x \in \mathcal{X}_N$ as

$$r_{N,x}(y) \triangleq r_N(y|x), \quad s_{\theta,x}(y) \triangleq s_\theta(y|x)\,.$$

Note that if the vector X includes continuous random variables, the probability of observing a perfectly identical case once again is theoretically zero and practically very low. The empirical density $r_{N,x}(y)$ is thus typically composed of a single δ-function. This does not affect the validity of the results presented in this and next sections, although it makes them perhaps less intuitive compared with discrete or discretized variables. The drastical lack of data just exhibits that when facing infinitely many (or just too many) cases, we cannot learn the response to one particular case without considering responses to other, related cases. More on this in Section 15.7.

Kerridge Inaccuracy. With the above notation, we can define the *Kerridge inaccuracy* [12] of conditional densities as

$$I(r_{N,x} : s_{\theta,x}) = \int r_{N,x}(y)\,\log\,\frac{1}{s_{\theta,x}(y)}\,\mu(\mathrm{d}y)\,.$$

The empirical expectation of the inaccuracy of conditional densities yields the *conditional Kerridge inaccuracy*

$$\begin{aligned} I(r_N : s_\theta) &= \mathbf{E}_N\, I(r_{N,x} : s_{\theta,x}) \\ &= \int r_N(x)\, I(r_{N,x} : s_{\theta,x})\,\mu(\mathrm{d}x) \\ &= \frac{1}{N}\sum_{x\in\mathcal{X}_N} N_x\, I(r_{N,x} : s_{\theta,x}) \\ &= \frac{1}{N}\sum_{k=1}^{N} I(r_{N,x_k} : s_{\theta,x_k})\,. \end{aligned} \tag{15.9}$$

Bayesian Inference Revisited. Using Kerridge inaccuracy and assuming $s(y|x) > 0$ for all (y, x), we can rewrite the likelihood function as

$$\begin{aligned}\prod_{k=1}^{N} s_\theta(y_k|x_k) &= \exp\Big(-\sum_{k=1}^{N} \log \frac{1}{s_\theta(y_k|x_k)}\Big) \\ &= \exp\Big(-N \iint r_N(y,x) \log \frac{1}{s_\theta(y|x)}\, \mu(\mathrm{d}y)\, \mu(\mathrm{d}x)\Big) \\ &= \exp\big(-N\, I(r_N : s_\theta)\big) \\ &= \exp\big(-N\, \mathbf{E}_N\, I(r_{N,X} : s_{\theta,X})\big)\ .\end{aligned}$$

With this expression, the posterior density becomes

$$p_N(\theta) \propto p_0(\theta)\, \exp\big(-N\, I(r_N : s_\theta)\big) \tag{15.10}$$

or, alternatively,

$$p_N(\theta) \propto p_0(\theta)\, \exp\big(-N\, \mathbf{E}_N\, I(r_{N,X} : s_{\theta,X})\big)\ . \tag{15.11}$$

Example. Let us assume the general regression model with a normally distributed additive noise

$$Y = f(X) + E, \quad E \sim N(0, \sigma^2)\ .$$

The sampling density for the model is

$$s_f(y|x) = (2\pi\sigma^2)^{\frac{1}{2}}\, \exp\Big\{-\frac{1}{2\sigma^2}\,(y - f(x))^2\Big\}\ .$$

The conditional inaccuracy relative the model is

$$I(r_N : s_\theta) = \frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}\,\frac{1}{N}\sum_{k=1}^{N}(y_k - f(x_k))^2\ .$$

Note it is a linear transform of the *empirical risk functional* in [22] and the *empirical error of* f in [9].

Prior Knowledge. We have seen that the likelihood function can be written as

$$l_N(\theta) = \exp\big(-N\, I(r_N : s_\theta)\big)\ .$$

Let us choose the prior density of Θ in the same form

$$p_0(\theta) = \exp\big(-\nu_0\, I(\rho_0 : s_\theta)\big) \tag{15.12}$$

where $\rho_0(y, x)$ denotes the prior density of (Y, X) and ν_0 stands for the "number of data" ρ_0 is built on, i.e., the degree of belief in ρ_0. The form (15.12) can be seen as a *generalized conjugate prior*. Indeed, the posterior density derived from such prior preserves its form

$$p_N(\theta) = \exp\big(-\nu_N\, I(\rho_N : s_\theta)\big)$$

while the statistics ν_N and ρ_N are updated as follows

$$\begin{aligned} \nu_N &= \nu_0 + N\,, \\ \rho_N(y,x) &= \frac{\nu_0}{\nu_0+N}\,\rho_0(y,x) + \frac{N}{\nu_0+N}\,r_N(y,x)\,. \end{aligned}$$

The prior can be built using *virtual* data provided by experts, generated by simulation models, created by controlled replication of existing data, etc. The virtual data can be combined from various sources, weighted according to the relevance and reliability of source, and turned into the prior density ρ_0 and the degree of belief ν_0.

Big Picture. The concepts introduced so far can be given the following interpretation. The model is a collection of densities of Y parametrized by the parameters θ *and* the case x. For each x observed in the data, one can define the empirical distribution of Y. The essence of modeling is in fitting of the empirical densities with model-based densities for each observed case (see Fig. 15.1). In Bayesian inference, the goodness of fit is expressed through the conditional (i.e., average) Kerridge inaccuracy (15.9). The impact of data depends on the local density of x-points (cf. Fig. 15.2).

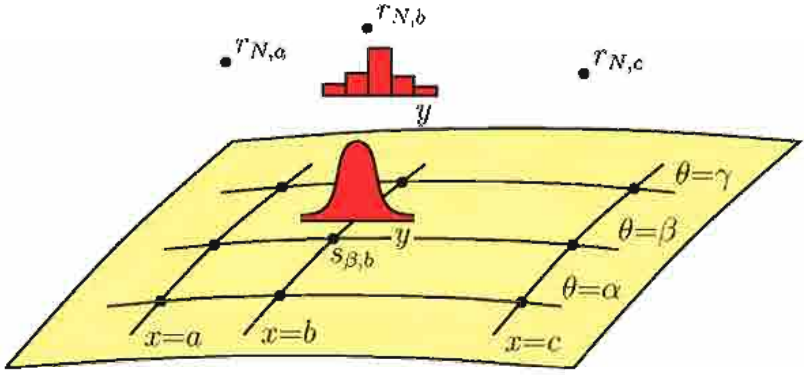


Figure 15.1: A statistical manifold of model- and case-based densities that approximate the empirical densities for the observed cases.

Note that the empirical distribution is discrete, concentrated on a finite number of points, but Kerridge inaccuracy is well defined even for continuous model distributions. Compare it with Kullback-Leibler divergence, which is infinite in this case.

To make the picture more intuitive for continuous cases x, one can consider a smoothened version of the conditional empirical density, e.g., by taking cross-section of a multivariate histogram of (Y, X).

The reader is referred to [13] for more discussion on various interpretations of Kerridge inaccuracy and its relationship to Kullback-Leibler divergence [17] and Shannon entropy [19].

15.5 Single-Case Geometry

We start exploring the geometry of Bayesian inference by analyzing the single case $X = x$ first. We assume to have observed responses $(y_1, \ldots, y_{N_x})$ under this particular

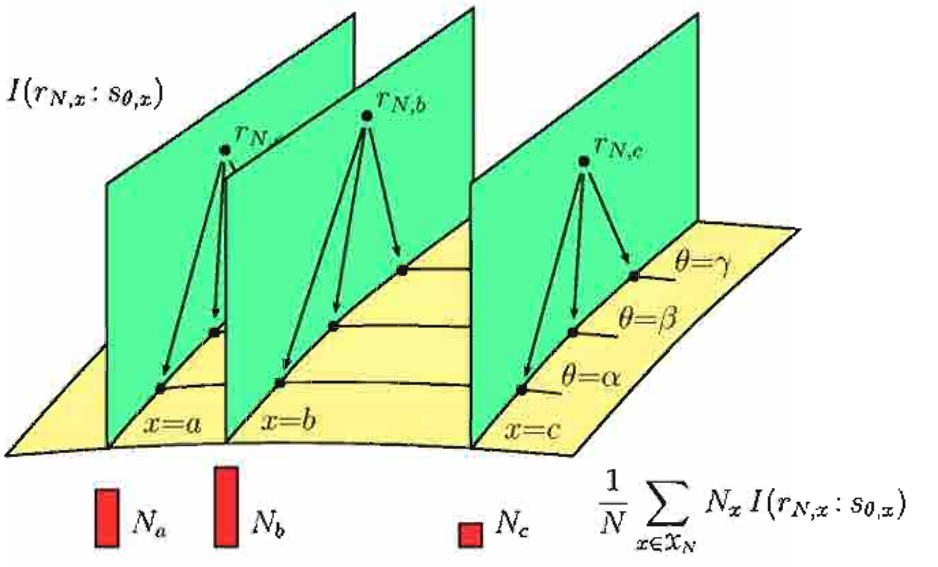


Figure 15.2: The overall goodness of fit is given by the empirical mean of Kerridge inaccuracy of case-based empirical densities relative to the corresponding model densities.

case. Note again (cf. discussion in the previous section) that the number of available samples generally decreases with the increasing number of distinct cases x, but this fact does not affect the following analysis. Our objective is to fit the empirical density $r_{N,x}$ with a case-based sampling density $s_{\theta,x}$.

We will omit the subscript x for a while to simplify the notation.

Exponential Family. Consider an exponential family $\mathcal{S}_h$ composed of densities

$$s_\lambda(y) = s_0(y)\, \exp\left(\lambda^T h(y) - \psi(\lambda)\right)$$

where s_0 is a fixed density (the family origin), θ is a vector parameter, h is a vector canonical statistic and $\psi(\lambda)$ is logarithm of the normalizing divisor

$$\psi(\lambda) = \log \int s_0(y)\, \exp\left(\lambda^T h(y)\right)\, \mu(\mathrm{d}y)\,.$$

The parameter λ of the family depends on the model and a particular case, $\lambda = \lambda(\theta, x)$.

h-Projection. We define a h-projection $s_{\widehat{\lambda}}(y)$ of $r_N(y)$ onto $\mathcal{S}_h$ by the equality

$$\int s_{\widehat{\lambda}}(y)\, h(y)\, \mu(\mathrm{d}y) = \int r_N(y)\, h(y)\, \mu(\mathrm{d}y)\,.$$

Note this is a necessary condition for $\widehat{\lambda}$ to minimize $I(r_N : s_\lambda)$

$$0 \;=\; \nabla_\lambda\, I(r_N : s_{\widehat{\lambda}}) = \int r_N(y)\, h(y)\, \mu(\mathrm{d}y) - \int s_{\widehat{\lambda}}(y)\, h(y)\, \mu(\mathrm{d}y)\,.$$

The expectation of $h(Y)$ with respect to r_N amounts to its sample average

$$\int r_N(y)\, h(y)\, \mu(\mathrm{d}y) = \frac{1}{N}\sum_{k=1}^{N} h(y_k) \triangleq \overline{h}_N\,.$$

We introduce the set of all densities with the same h-projection as the empirical density has

$$\mathcal{R}_N = \left\{ \text{density } r(y) \colon \int r(y)\, h(y)\, \mu(\mathrm{d}y) = \overline{h}_N \right\} .$$

Pythagorean Relation. Let $\mathcal{S}_h$ be exponential and $s_{\widehat{\lambda}}$ be a h-projection of r_N onto $\mathcal{S}_h$. Then, for every $s_\lambda \in \mathcal{S}_h$ and every $r \in \mathcal{R}_N$, it holds

$$I(r : s_\lambda) = I(r : s_{\widehat{\lambda}}) + D(s_{\widehat{\lambda}} \| s_\lambda) \tag{15.13}$$

where

$$D(s \| s') = \int s(y)\, \log \frac{s(y)}{s'(y)}\, \mu(\mathrm{d}y)$$

is Kullback-Leibler divergence of $s(y)$ relative to $s'(y)$.

The identity (15.13) follows directly by definitions of Kerridge inaccuracy and Kullback-Leibler divergence

$$\begin{aligned} &\int \left[r(y) - s_{\widehat{\lambda}}(y) \right] \left[\log s_{\widehat{\lambda}}(y) - \log s_\lambda(y) \right] \mu(\mathrm{d}y) \\ &= \int \left[r(y) - s_{\widehat{\lambda}}(y) \right] (\widehat{\lambda} - \lambda)^T h(y)\, \mu(\mathrm{d}y) \\ &= (\widehat{\lambda} - \lambda)^T \int \left[r(y) - s_{\widehat{\lambda}}(y) \right]^T h(y)\, \mu(\mathrm{d}y) = 0 . \end{aligned} \tag{15.14}$$

The relation (15.13) can be viewed as a version of Pythagorean-like theorem. It allows us to decompose the Kerridge inaccuracy of the empirical density r_N relative to an exponential density s_λ with the canonical statistic $h(y)$ into sum of two terms - the Kerridge inaccuracy of r_N relative to the h-projection $s_{\widehat{\lambda}}$ plus the Kullback-Leibler divergence of $s_{\widehat{\lambda}}$ relative to s_λ (cf. Fig. 15.3).

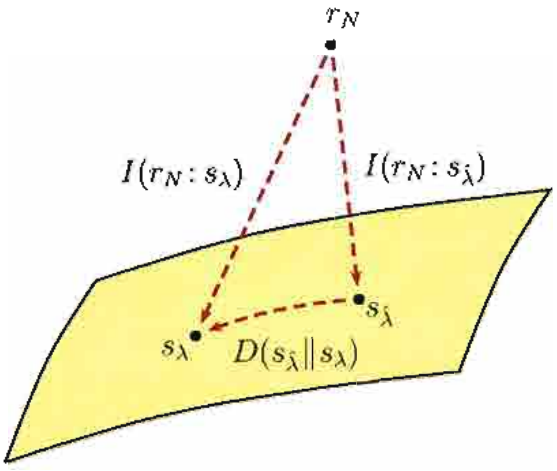


Figure 15.3: The Pythagorean relation for the h-projection $s_{\widehat{\lambda}}$ of the empirical density r_N onto an exponential family $\mathcal{S}_h$.

Differential-Geometric View. The condition (15.14) suggests that the Pythagorean relationship can be rewritten as

$$\int \left[r(y) - s_{\hat{\lambda}}(y)\right] \left[\log s_{\hat{\lambda}}(y) - \log s_{\lambda}(y)\right] \mu(\mathrm{d}y) = 0\,.$$

This can be taken as a definition of *orthogonal projection of* r_N *onto* $\mathcal{S}_h$ or, from a dual viewpoint, *orthogonal projection of* s_λ *onto* $\mathcal{R}_N$.

In contrast to the orthogonality of vectors in the Euclidean space, the appearance of logarithm makes the above condition asymmetric in $r(y)$ and $s(y)$. Consequently, in the space of probability distributions there is no straightforward analogy of the "natural" inner product as we know it from the Euclidean space.

We can, however, rewrite the condition as follows

$$\int s_{\hat{\lambda}}(y) \frac{\partial}{\partial \mu} \log\Big[\mu\, r_N(y) + (1-\mu)\, s_{\hat{\lambda}}(y)\Big]\Big|_{\mu=0} \cdot \frac{\partial}{\partial \lambda} \log\Big[s_0(y)\, \exp\big(\lambda\, h(y) - \psi(\lambda)\big)\Big]\Big|_{\lambda=\hat{\lambda}} \mathrm{d}y = 0\,.$$

This definition gives rise to a specific kind of Riemannian geometry on a differentiable manifold of probability distributions. The underlying metric tensor is closely related to the *Fisher information matrix*. In contrast to the classical Riemannian-geometric picture, two dual affine connections need to be considered at the same time in order to explain the asymmetry of the geometry. In these connections, *exponential and mixture families* of probability distributions act as analogy of hyperplanes in the Euclidean case.

Elaboration of this view is beyond the scope of the chapter. The interested reader is referred for details to [5, 8, 2, 13, 23].

Dual Optimization Tasks. It follows directly from the Pythagorean relation (15.13) that the projection $s_{\hat{\lambda}}$ is a solution to two dual optimization tasks (cf. Fig. 15.4).

Maximum Likelihood Estimate: For every $r \in \mathcal{R}_N$

$$I(r : s_{\hat{\lambda}}) = \min_{\lambda} I(r : s_\lambda)\,.$$

Note that replacing of r_N with ρ_N yields the *maximum a posteriori probability* estimate.

Maximum Entropy Estimate: For every $s_\lambda \in \mathcal{S}_h$,

$$D(s_{\hat{\lambda}} \| s_\lambda) = \min_{r \in \mathcal{R}_N} D(r \| s_\lambda)\,.$$

Note that minimizing Kullback-Leibler divergence is – up to a term relative to s_λ – equivalent to maximizing Shannon entropy. This explains the *maximum entropy* label given to this task.

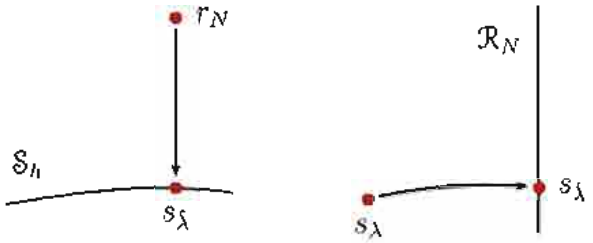


Figure 15.4: The h-projection $s_{\widehat{\lambda}}$ of the empirical density r_N onto an exponential family $\mathcal{S}_h$ is a solution to the maximum likelihood and maximum (relative) entropy tasks.

Inverse Problem. At the beginning of this section, we have said that the parameter λ of the exponential family $\mathcal{S}_h$ depends on the model θ and a particular case x, i.e., $\lambda = \lambda(\theta, x)$. Given the projection $\widehat{\lambda}_N$, we can define the estimate $\widehat{\theta}_{N,x}$ of θ as a solution to the equation

$$\lambda(\theta, x) = \widehat{\lambda}_N$$

for a given x. Typically, $\widehat{\theta}_{N,x}$ is not unique. For any solution $\widehat{\theta}_{N,x}$, the Pythagorean relation for conditional densities $s_{\theta,x}(y)$ reads

$$I(r_{N,x} \colon s_{\theta,x}) = I(r_{N,x} \colon s_{\widehat{\theta}_{N,x},x}) + D(s_{\widehat{\theta}_{N,x},x} \| s_{\theta,x}) \,. \tag{15.15}$$

15.6 Average-Case Geometry

We will consider now all observed cases x jointly. Our objective is to fit a single model s_θ to the data so that the average inaccuracy is minimized.

Exponential Family. Consider an *exponential family* $\mathcal{S}_h$ composed of densities

$$s_\theta(y|x) = s_0(y|x) \, \exp\left(\theta^T h(y,x) - \psi(\theta,x)\right)$$

where s_0 is a fixed density, θ is a vector parameter, h is a vector canonical statistic and ψ is logarithm of the normalizing divisor

$$\psi(\theta,x) = \log \int s_0(y|x) \, \exp\left(\theta^T h(y,x)\right) \, \mu(\mathrm{d}y) \,.$$

h-Projection. We define a h-projection of $r_N(y,x)$ onto $\mathcal{S}_h$ by the equality

$$\int r_N(x) \int s_{\widehat{\theta}}(y|x) \, h(y,x) \, \mu(\mathrm{d}y) \, \mu(\mathrm{d}x) = \iint r_N(y,x) \, h(y,x) \, \mu(\mathrm{d}y) \, \mu(\mathrm{d}x) \,.$$

This is a necessary condition for $\widehat{\theta}$ to minimize $I(r_N \colon s_\theta)$

$$0 = \nabla_\theta \, I(r_N \colon s_{\widehat{\theta}}) \,.$$

Pythagorean Relation. Let $\mathcal{S}_h$ be exponential and $s_{\widehat{\theta}}$ be a h-projection of r_N onto $\mathcal{S}_h$. Then, for every $s_\theta \in \mathcal{S}_h$, the following Pythagorean relation holds

$$I(r_N : s_\theta) = I(r_N : s_{\widehat{\theta}}) + \mathbf{E}_N\, D(s_{\widehat{\theta}} \| s_\theta)\,.$$

The relation can be alternatively written as

$$\mathbf{E}_N\, I(r_{N,X} : s_{\theta,X}) = \mathbf{E}_N\, I(r_{N,X} : s_{\widehat{\theta},X}) + \mathbf{E}_N\, D(s_{\widehat{\theta},X} \| s_{\theta,X})\,.$$

Compare this *global estimation* formula with the *local estimation* formula (15.15)

$$I(r_{N,x} : s_{\theta,x}) = I(r_{N,x} : s_{\widehat{\theta},x}) + D(s_{\widehat{\theta},x} \| s_{\theta,x})\,.$$

Clearly, the former is just the expected version of the latter.

The global model found through the sample average of Kerridge inaccuracy is clearly a tradeoff. The use of all available data sets N to its maximum, thus reducing the total uncertainty of estimation. On the other hand, unless a single model with constant θ explains well the data behavior for all cases x, the global error expressed through the case-averaged inaccuracy typically increases.

Example. The conditional inaccuracy for the linear normal regression model takes the form

$$I(r_N : s_\theta) = \frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}V_N + \frac{1}{2\sigma^2}(\theta - \widehat{\theta}_N)^T C_N (\theta - \widehat{\theta}_N)$$

with the statistics $\widehat{\theta}_N$, V_N and C_N introduced earlier through (15.6)–(15.8).

The posterior expectation $\mathcal{E}_N(\cdot)$ of the conditional inaccuracy follows after some algebraic manipulations

$$\mathcal{E}_N\, I(r_N : s_\theta) = \frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{2}\frac{V_N}{\sigma^2} + \frac{1}{2}\frac{\dim\theta}{N}$$

assuming that the prior is flat and the posterior covariance is positive definite.

The formula combines all ingredients of the modeling task - the sum of residuals squared V_N, the model variance σ^2, the model complexity $\dim\theta$, and the sample size N. It suggests that the model performance can be tuned up by balancing the coherence of data ("use *only* relevant data"), sample size ("use *all* relevant data") and model complexity ("strive for the *simplest* model").

15.7 Similar-Case Modeling

In Sections 15.5 and 15.6, we have considered two extreme approaches to modeling – building of a strictly local model for the case-specific data and fitting of a global model to all the data. In this section, we show how one can smoothly move between the extremes.

Local Models. Assume a set of local models $\mathcal{M} = \{\theta_x : x \in \overline{\mathcal{X}}\}$ with x coming from a *finite* set $\overline{\mathcal{X}}$ such that $\mathcal{X}_N \subset \overline{\mathcal{X}} \subset \mathcal{X}$.

The posterior density over the model set $\mathcal{M}$ follows by the standard probability calculus rules

$$p_N(\{\theta_x\}) \propto p_0(\{\theta_x\}) \prod_{k=1}^{N} s_{\theta_{x_k}}(y_k|x_k) .$$

Using the Kronecker delta

$$\delta_{a,b} = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{if } a \neq b \end{cases} ,$$

the local likelihood $l_{N,x}(\theta_x)$ at $X = x$ and the corresponding number of data N_x can be defined as

$$l_{N,x}(\theta_x) \;=\; \prod_{k=1}^{N} s_{\theta_x}(y_k|x_k)^{\delta_{x,x_k}} , \quad N_x = \sum_{k=1}^{N} \delta_{x,x_k} .$$

The *joint posterior density* of all models can be rewritten in terms of the local likelihoods as

$$p_N(\{\theta_x\}) \propto p_0(\{\theta_x\}) \prod_{x} l_{N,x}(\theta_x) .$$

The *marginal posterior density* for the local model at $X = \xi$ is obtained from the joint posterior density by integrating out all θ_x for $x \neq \xi$

$$p_N(\theta_\xi) \propto \int \ldots \int p_0(\{\theta_x\}) \prod_{x} l_{N,x}(\theta_x) \prod_{x \neq \xi} \mu(\mathrm{d}\theta_x) . \tag{15.16}$$

Multiple-Model Prior. It is the choice of the prior density $p_0(\{\theta_x\})$ in (15.16) that determines to which level information accumulated about θ_x, $x \neq \xi$ is used to reduce the uncertainty of θ_ξ. Let us consider three basic options:

1. All the data is fitted with a single model.
2. The data is fitted separately for each condition x.
3. The selected data, for x close to a given ξ, is fitted with a local model.

Single Global Model. If we choose the prior density concentrated on a constant

$$p_0(\{\theta_x\}) \propto p_0(\theta) \prod_{x} \delta(\theta_x - \theta) ,$$

the posterior density combines all the data

$$p_N(\theta) \;\propto\; p_0(\theta) \prod_{x} l_{N,x}(\theta)^{N_x} \propto p_0(\theta)\, l_N(\theta) .$$

Multiple Strictly Local Models. If we choose the prior density in the product form

$$p_0(\{\theta_x\}) \propto \prod_x p_0(\theta_x)\,,$$

the posterior density uses only the local data

$$p_N(\theta_\xi) \;\propto\; p_0(\theta_\xi)\, l_{N,\xi}(\theta_\xi)\,.$$

If there is no data for $X = \xi$, the posterior coincides with the prior

$$N_\xi = 0 \;\Rightarrow\; p_N(\theta_\xi) = p_0(\theta_\xi)\,,$$

i.e., we do not learn any way from the other data available.

Statistically Dependent Models. Let us choose the prior density as a mixture

$$p_0(\{\theta_x\}) \propto \sum_\xi \pi(\xi)\, p_0(\{\theta_x\}|\xi)$$

where for each ξ the parameters $\{\theta_x : x \neq \xi\}$ are *conditionally independent* given θ_ξ

$$p_0(\{\theta_x\}|\xi) = p_0(\theta_\xi) \prod_{x\neq\xi} p_0(\theta_x|\theta_\xi)\,.$$

The posterior density of θ_ξ is then

$$p_N(\theta_\xi) \propto p_0(\theta_\xi)\, l_{N,\xi}(\theta_\xi) \prod_{x\neq\xi} \int p_0(\theta_x|\theta_\xi)\, l_{N,x}(\theta_x)\, \mu(\mathrm{d}\theta_x)\,. \tag{15.17}$$

Let us analyze the last option in more detail.

Cross-Model Likelihood. In the formula (15.17), information available about θ_x for $x \neq \xi$ affects estimation of θ_ξ through the cross-model likelihood factor

$$l_{N,x}(\theta_\xi) = \int p_0(\theta_x|\theta_\xi)\, l_{N,x}(\theta_x)\, \mu(\mathrm{d}\theta_x)\,. \tag{15.18}$$

Consider two extreme instances of cross-model dependence:

A. θ_x is identical (coincides) with θ_ξ,

$$p_0^{\mathbf{A}}(\theta_x|\theta_\xi) = \delta(\theta_x - \theta_\xi) \;\Rightarrow\; l_{N,x}^{\mathbf{A}}(\theta_\xi) = l_{N,x}(\theta_\xi)\,.$$

B. θ_x is independent of θ_ξ,

$$p_0^{\mathbf{B}}(\theta_x|\theta_\xi) = p_0(\theta_x) \;\Rightarrow\; l_{N,x}^{\mathbf{B}}(\theta_\xi) = \text{const.}$$

Now, rather than calculating (15.18) directly, we can approximate it by smoothing between the extremes **A** and **B**

$$l^w_{N,x}(\theta_\xi) = c' \left[l^{\mathbf{A}}_{N,x}(\theta_\xi)\right]^{w(x,\xi)} \left[l^{\mathbf{B}}_{N,x}(\theta_\xi)\right]^{1-w(x,\xi)}$$

with respective weights

$$w(x,\xi) \quad \text{and} \quad 1 - w(x,\xi)$$

satisfying

$$0 \le w(x,\xi) \le w(\xi,\xi) = 1\,.$$

Since $l^{\mathbf{B}}_{N,x}(\theta_\xi)$ is a constant independent of θ_ξ, the cross-model likelihood is approximated through

$$l^w_{N,x}(\theta_\xi) = c\left[l_{N,x}(\theta_\xi)\right]^{w(x,\xi)}. \tag{15.19}$$

Example. Consider the linear normal regression model with the local likelihood in the form

$$l_{N,x}(\theta_x) = c_1 \exp\left\{-\frac{1}{2}(\theta_x - \widehat{\theta}_x)^T P_x^{-1}(\theta_x - \widehat{\theta}_x)\right\}.$$

Let us define the dependence of θ_x on θ_ξ explicitly via the stochastic equation

$$\theta_x = \theta_\xi + v, \;\; v \sim N(0, Q)\,.$$

The cross-model likelihood (15.18) results after some algebraic manipulations

$$\int p_0(\theta_x|\theta_\xi)\, l_{N,x}(\theta_x)\, \mu(\mathrm{d}\theta_x) = c_2 \exp\left\{-\frac{1}{2}(\theta_\xi - \widehat{\theta}_x)^T (P_x + Q)^{-1}(\theta_\xi - \widehat{\theta}_x)\right\}.$$

Compare the result with the approximate expression (15.19)

$$l^w_{N,x}(\theta_\xi) = c_3\left[l_{N,x}(\theta_\xi)\right]^w = c_4 \exp\left\{-\frac{1}{2}(\theta_\xi - \widehat{\theta}_x)^T w\, P_x^{-1}(\theta_\xi - \widehat{\theta}_x)\right\}.$$

Both Q and w depend here on (x,ξ).

The resulting formulae for the update of the covariance matrix of θ_ξ are related similarly as the Kalman filter-like *linear forgetting* $P_x + Q$, $\quad Q > 0$ and *exponential forgetting* $\frac{1}{w} P_x$, $\quad 0 \le w \le 1$. (cf. [16, 15]).

Continuous and Discrete Predictors. When dealing with continuous predictor variables, a popular approach is to define the weights $w(x,\xi)$ via a suitable *kernel function*

$$K_h(x,\xi)$$

the shape of which can be fine-tuned by bandwidth parameters h. The weight on the model at x relative to ξ thus depends on the Euclidean distance of x from ξ. To put it other way, one makes us of the topology of the predictor space to infer on the similarity of respective models.

This approach does not work for discrete (categorical) predictors. Consider, e.g., days of week. Lacking any natural embedding of the respective values into a Euclidean space, the weight such as $w(\text{Tuesday}, \text{Friday})$ can be derived only from the "strength" of statistical dependence of the respective models expressed through the density $p_0(\theta_{\text{Tuesday}}|\theta_{\text{Friday}})$.

Posterior Density. After substituting the approximate expression (15.19) of the cross-model likelihoods (15.18) in the posterior density (15.17), we obtain a locally weighted formula

$$\begin{aligned} p_N(\theta_\xi) &\propto p_0(\theta_\xi)\, l_{N,\xi}(\theta_\xi) \prod_{x\neq\xi} [l_{N,x}(\theta_\xi)]^{w(x,\xi)} \\ &\propto p_0(\theta_\xi) \prod_{x\in\mathcal{X}_N} [l_{N,x}(\theta_\xi)]^{w(x,\xi)} \\ &\propto p_0(\theta_\xi) \prod_{k=1}^{N} \left[s_{\theta_\xi}(y_k|x_k)\right]^{w(x_k,\xi)} . \end{aligned} \tag{15.20}$$

15.8 Locally Weighted Geometry

The locally weighted Bayesian estimation developed in the previous section can be given an intuitive geometric interpretation again.

Empirical Density. For a given sample y^N, x^N and a fixed "query" point ξ, we define the *effective number of data* and the *marginal empirical density* of X as

$$\begin{aligned} N_\xi^w &= \sum_{k=1}^{N} w(x_k,\xi)\,, \\ r_{N_\xi^w}(x) &= \frac{1}{N_\xi^w} \sum_{k=1}^{N} w(x_k,\xi)\,\delta(x-x_k)\,. \end{aligned}$$

The *joint empirical density* of (Y,X) combines the original conditional density $r_N(y|x)$ and the weighted marginal density $r_{N_\xi^w}(x)$

$$r_{N_\xi^w}(y,x) = r_N(y|x)\, r_{N_\xi^w}(x)\,.$$

Kerridge Inaccuracy. The empirical expectation of the inaccuracy of conditional densities yields the *conditional Kerridge inaccuracy*

$$\begin{aligned} I(r_{N_\xi^w} : s_\theta) &= E_{N_\xi^w}\, I(r_{N,x} : s_{\theta,x}) \\ &= \int r_{N_\xi^w}(x)\, I(r_{N,x} : s_{\theta,x})\, \mu(\mathrm{d}x) \\ &= \frac{1}{N_\xi^w} \sum_{x\in\mathcal{X}_N} w(x,\xi)\, N_x\, I(r_{N,x} : s_{\theta,x}) \\ &= \frac{1}{N_\xi^w} \sum_{k=1}^{N} w(x_k,\xi)\, I(r_{N,x_k} : s_{\theta,x_k})\,. \end{aligned}$$

Likelihood Function. The resulting likelihood function can be rewritten as

$$\begin{aligned}\prod_{k=1}^{N} s_\theta(y_k|x_k)^{w(x_k,\xi)} &= \exp\Big(-\sum_{k=1}^{N} w(x_k,\xi)\,\log\frac{1}{s_\theta(y_k|x_k)}\Big)\\ &= \exp\Big(-N_\xi^w \iint r_{N_\xi^w}(y,x)\,\log\frac{1}{s_\theta(y|x)}\,\mu(\mathrm{d}y)\,\mu(\mathrm{d}x)\Big)\\ &= \exp\big(-N_\xi^w\, I(r_{N_\xi^w}:s_\theta)\big)\\ &= \exp\big(-N_\xi^w\, E_{N_\xi^w}\, I(r_{N,X}:s_{\theta,X})\big)\end{aligned}$$

Posterior Density. The posterior density (15.20) can thus be expressed in the following compact form

$$p_N(\theta) \propto p_0(\theta)\,\exp\Big(-N_\xi^w\, I(r_{N_\xi^w}:s_\theta)\Big)\ .$$

Where appropriate, the prior density $p_0(\theta)$ can be chosen in the generalized conjugate form (15.12).

Big Picture. In locally weighted estimation, we deliberately modify the empirical density of the observed cases so as to use information learnt about θ_ξ *only* at cases x "relevant" or "similar" to a given case ξ. The weighting in the empirical density results in the same weighting of conditional Kerridge inaccuracy. The practical effect of relevance weighting is in making the model $s_{\widehat{\theta}_N^w}$ fit better the local data at cases x "close to" ξ (cf. Fig. 15.5).

The weight put on the case x reflect the level of statistical dependence of θ_x on θ_ξ. "If I knew θ_ξ, how much would that affect my knowledge of θ_x at $x \neq \xi$?" The perfect dependence (identity) implies weight 1 while independence means weight 0.

15.9 Concluding Remarks

The chapter has analyzed local modeling using the information geometry of Bayesian inference for conditional probabilities.

Bayesian Smoothing

- We have shown that nonparametric regression can be derived as a special case of Bayesian inference over a set of local, case-based models with a properly chosen prior density linking the models.
- The result can be approximated as a weighted Bayesian inference with weights on the local likelihoods being proportional to the "strength" of statistical cross-model dependence.
- The practical advantage of the chosen approximation is that one needs to retrieve from the history only data that are assigned positive weights.

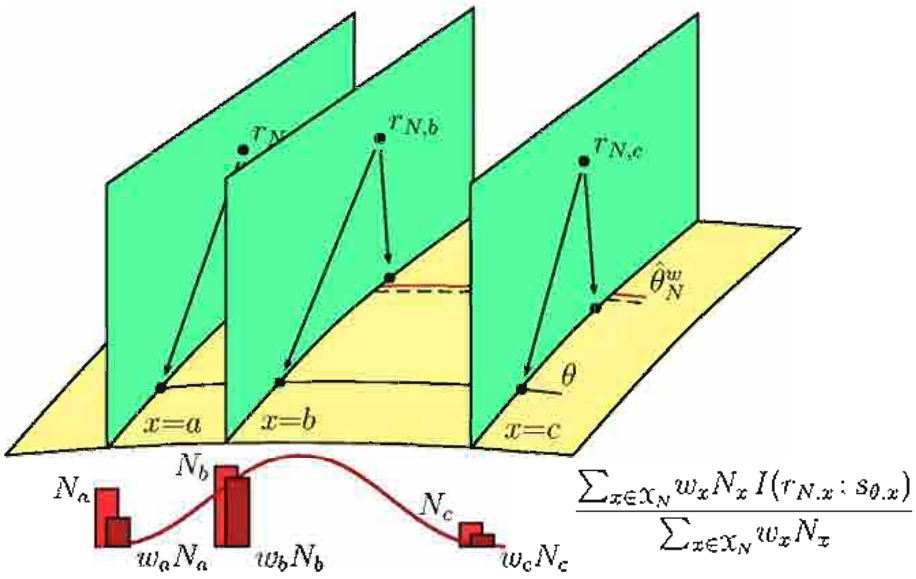


Figure 15.5: The goodness of fit in locally weighted Bayesian inference is given by the weighted empirical mean of Kerridge inaccuracy of case-based empirical densities relative to the corresponding model densities.

Why Information Geometry?

- We can view statistical inference as approximation of the empirical density rather than estimation of a hypothetical "true" density.
- Kerridge inaccuracy provides us with a generalized empirical error, which changes consistently with the underlying model family.
- We can elicit prior knowledge via the virtual data and capture the knowledge in the prior density of data using rigorous statistical methods.
- We can fine-tune the model by analyzing the orthogonal projection "trace" of conditional empirical (possibly smoothened) distributions onto the model manifold.
- The resulting "big picture" provides a natural departure point for design of approximations to the optimal but intractable solutions.

Bibliography

[1] D.W. Aha, D. Kibler and M.K. Albert, Instance-based learning algorithms, Machine Learning **6** (1991) 37–66.

[2] S. Amari, Differential-Geometrical Methods in Statistics, Vol. 28 of Lecture Notes in Statistics, Springer-Verlag, Berlin (1985).

[3] C.G. Atkeson, S.A. Schaal and A.W. Moore, Locally weighted learning, AI Review **11** (1997) 11–73.

[4] L. Bottou and V. Vapnik, Local learning algorithms, Neural Computation 4 (1992) 888–900.

[5] N.N. Chentsov, Statistical Decision Rules and Optimal Inference (in Russian), Nauka, Moscow (1972). English translation in Translations of Mathematical Monographs 53, Amer. Math. Soc., Providence, RI (1982).

[6] W.S. Cleveland, Robust locally weighted regression and smoothing scatterplots, J. Amer. Statist. Assoc. **74** (1979) 829–836.

[7] W.S. Cleveland and S.J. Devlin. Locally-weighted regression: an approach to regression analysis by local fitting, J. Amer. Statist. Assoc. **83** (1988) 596–610.

[8] I. Csiszár, I-divergence geometry of probability distributions and minimization problems, Ann. Probab. **3** (1975) 146–158.

[9] F. Cucker and S. Smale, On the mathematical foundations of learning, Bull. Amer. Math. Soc. **39** (2002) 1–49.

[10] G. Cybenko, Just-in-time learning and estimation, in S. Bittanti and G. Picci (eds.), Identification, Adaptation, Learning, 423–434, NATO ASI Series, Springer-Verlag (1996).

[11] W. Härdle, Applied Non-parametric Regression, Cambridge University Press (1990).

[12] D.F. Kerridge, Inaccuracy and inference, J. Roy. Statist. Soc. Ser. B **23** (1961) 284–294.

[13] R. Kulhavý, Recursive Nonlinear Estimation: A Geometric Approach. Vol. 216 of Lecture Notes in Control and Information Science, Springer-Verlag, London (1996).

[14] R. Kulhavý and P. Ivanova, Memory-based prediction in control and optimisation, in Proc. 14th World Congress of IFAC, Beijing, PRC Vol. H. (1999) 289–294.

[15] R. Kulhavý and F.J. Kraus, On duality of regularized exponential and linear forgetting, Automatica **32** (1996) 1403–1415.

[16] R. Kulhavý and M.B. Zarrop, On a general concept of forgetting, Int. J. Control **58** (1993) 905–924.

[17] S. Kullback and R.A. Leibler, On information and sufficiency, Ann. Math. Statist. **22** (1951) 79–86.

[18] V. Peterka, Bayesian approach to system identification, in P. Eykhoff (Ed.), Trends and Progress in System Identification. Pergamon Press, Elsmford, NY (1981) 239–304.

[19] C.E. Shannon, A mathematical theory of communication, Bell System Tech. J. **26** (1948) 379–423, 623–656.

[20] A. Stenman, Model on demand: algorithms, analysis and applications, Ph.D. Thesis No. 571, Dept. of EE, Linköping University (1999).

[21] D.M. Titterington, Common structure of smoothing techniques in statistics. Intern. Stat. Review **53** (1985) 141–170.

[22] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer, New York (1995).

[23] H. Zhu and R. Rohwer, Information geometry, Bayesian inference, ideal estimates and error decomposition, Technical Report No. 98-06-045, Santa Fe Institute (1998).

Chapter 16

Nonparametric Prediction

László Györfi and Dominik Schäfer

Abstract. In this chapter we consider the prediction of stationary time series for various loss functions: squared loss (as it arises in the regression problem), $0-1$ loss (pattern recognition) and log utility (portfolio selection). The focus is on the construction of universal prediction rules, which are consistent for all possible stationary processes. Such rules can be obtained by combining elementary rules (experts) in a data dependent way.

16.1 Introduction

The problem of prediction of stationary time series arises in numerous fields. It is particularly desirable to construct **universal prediction rules** for the next output of a stationary time series given past observations. These are prediction rules which are asymptotically optimal, but do not require a priori knowledge about the underlying distribution of the time series (Haussler, Kivinen, Warmuth [18], Merhav, Feder [21]).

Depending upon the context of the prediction problem different loss functions are appropriate. The three most important loss functions are the squared loss (for real valued time series, i.e., the regression problem), the $0-1$ loss (for time series taking on values in a finite set, i.e., pattern recognition) and logarithmic utility (for time series of asset returns in portfolio selection).

Prediction rules that are asymptotically optimal can be constructed by combining elementary rules (experts) in a data dependent way. The key idea is simple: Roughly speaking, the worse an expert predicted in the past, the less credible he is, i.e., the less weight he is assigned in current decision taking (Cesa-Bianchi et al. [7], Littlestone, Warmuth [20], Vovk [26], [27], [28], Weinberger, Merhav and Feder [29]). The main purpose of this chapter is to present universal prediction rules with data dependent combination of experts in the three prototypical fields of regression, of pattern recognition and of portfolio selection.

16.2 Prediction for Squared Error

This section is devoted to the problem of sequential prediction of a real valued sequence. Let $y_1, y_2, \ldots$ be a sequence of real numbers, and let $x_1, x_2, \ldots$ be a sequence of d-dimensional vectors. At each time instant $i = 1, 2, \ldots$, the predictor is asked to guess the value of the next outcome y_i with knowledge of the past $(x_1, \ldots, x_i, y_1, \ldots y_{i-1}) = (x_1^i, y_1^{i-1})$. Thus, the predictor's estimate, at time i, is based on the value of (x_1^i, y_1^{i-1}). Formally, the strategy of the predictor is a sequence $g = \{g_i\}_{i=1}^{\infty}$ of decision functions, and the prediction formed at time i is $g_i(x_1^i, y_1^{i-1})$. After n time instants, the *normalized cumulative prediction error* on the string x_1^n, y_1^n is

$$L_n(g) = \frac{1}{n}\sum_{i=1}^{n}(g_i(x_1^i, y_1^{i-1}) - y_i)^2.$$

The main aim is to make $L_n(g)$ small (Haussler, Kivinen, Warmuth [18], Merhav, Feder [21]).

One possible means of prediction is to combine several predictors which will be called experts. Assume there are K experts: $h^{(1)}, \ldots, h^{(K)}$ and the prediction error $L_n(h^{(k)})$ of expert k is available from observation. At time instant $n+1$ we combine the experts according to their past performances. For this, a probability distribution on the set of experts is generated, where a good expert has relatively large weight, then the average of the experts' predictions is taken with respect to this distribution

(Cesa-Bianchi et al. [7], Littlestone, Warmuth [20], Vovk [26], Weinberger, Merhav and Feder [29]).

A "static" variant of this problem is regression estimation. Let Y be a real valued random variable and let X be a d dimensional random vector (observation). The aim of regression analysis is to approximate Y for given X, i.e., to find a function g such that $g(X)$ is "close" to Y. In particular, regression analysis aims to minimize the mean squared error

$$\min_g \mathbf{E}\{(g(X) - Y)^2\}.$$

It is well known that the solution of this minimization problem is given by the regression function

$$m(x) = \mathbf{E}\{Y|X = x\},$$

since for any function g

$$\mathbf{E}\{(g(X) - Y)^2\} = \mathbf{E}\{(m(X) - Y)^2\} + \mathbf{E}\{(m(X) - g(X))^2\}.$$

The second term on the right hand side is the L_2 error of g and will be denoted by $J(g)$:

$$J(g) = \mathbf{E}\{(m(X) - g(X))^2\}.$$

Obviously, the mean square error is close to its minimum if the L_2 error $J(g)$ is close to 0.

For the regression estimation problem we are given data

$$D_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$$

which are i.i.d. copies of (X, Y). On the basis of this data, we want to construct estimates of $m(x)$ of the form

$$m_n(x) = m_n(x, D_n)$$

such that $J(m_n)$ is small, i.e., m_n tends to m for all distributions of (X, Y) with $\mathbf{E}\{Y^2\} < \infty$ (cf. Györfi, Kohler, Krzyżak, Walk [14]).

Stone [24] showed that there are universally consistent regression estimates. He considered local averaging estimates:

$$m_n(x) = \sum_{i=1}^{n} W_{ni}(x; X_1, \ldots, X_n)Y_i,$$

where the weights W_{ni} are usually nonnegative and sum up to 1, moreover W_{ni} is "large", if x and X_i are "close" to each other, otherwise W_{ni} is "small". Common local averaging estimators comprise nearest neighbor, partitioning and kernel estimators.

For the k **nearest neighbor estimate**, $W_{ni}(x; X_1, \ldots, X_n) = 1/k$, if X_i is one the k nearest neighbors of x from $X_1, \ldots, X_n$, otherwise $W_{ni} = 0$. If

$$k_n \to \infty, \quad k_n/n \to 0,$$

then there are various consistency results.

For the **partitioning estimate** we are given a partition $\mathcal{P}_n = \{A_{n,1}, A_{n,2} \dots\}$ of $\mathcal{R}^d$, and set

$$m_n(x) = \frac{\sum_{i=1}^n Y_i K_n(x, X_i)}{\sum_{i=1}^n K_n(x, X_i)},$$

where $K_n(x,u) = \sum_{j=1}^\infty I_{[x \in A_{n,j}, u \in A_{n,j}]}$ (I_A is the indicator of the set A).

The **kernel estimate** is given by

$$m_n(x) = \frac{\sum_{i=1}^n Y_i K_h(x - X_i)}{\sum_{i=1}^n K_h(x - X_i)},$$

where $h = h_n > 0$ is the bandwidth and K is an integrable function, called kernel, and $K_h(x) = K(x/h)$.

The other important concept for estimating regression functions is the least squares principle. It is based on the simple idea to estimate the L_2 risk of f

$$\mathbf{E}\left\{(f(X) - Y)^2\right\}$$

by the empirical L_2 risk

$$\frac{1}{n} \sum_{j=1}^n |f(X_j) - Y_j|^2, \tag{16.1}$$

and to choose as a regression function estimate a function which minimizes the empirical L_2 risk. More precisely, for **least squares estimates** one first chooses a "suitable" class of functions $\mathcal{F}_n$ (maybe depending on the data, but at least depending on the sample size n) and then selects a function from this class which minimizes the empirical L_2 risk, i.e. one defines the estimate m_n by

$$m_n \in \mathcal{F}_n \quad \text{with} \quad \frac{1}{n} \sum_{j=1}^n |m_n(X_j) - Y_j|^2 = \min_{f \in \mathcal{F}_n} \frac{1}{n} \sum_{j=1}^n |f(X_j) - Y_j|^2. \tag{16.2}$$

The class of candidate functions grows as the sample-size n grows. Examples of possible choices of the set $\mathcal{F}_n$ are sets of piecewise polynomials with respect to a partition $\mathcal{P}_n$ of $\mathcal{R}^d$, or sets of smooth piecewise polynomials (splines).

The other framework in which the need for universal prediction arises is the case of time series where the data $D_n = \{(X_1, Y_1), \dots, (X_n, Y_n)\}$ are **dependent**. Here we assume long-range dependence, i.e., we assume that the data form a stationary and ergodic process with unknown autocovariance structure.

For given n, the problem is the following minimization:

$$\min_g \mathbf{E}\{(g(X_{n+1}, D_n) - Y_{n+1})^2\}.$$

From this one easily verifies that the best predictor is the conditional expectation

$$\mathbf{E}\{Y_{n+1} | X_{n+1}, D_n\}.$$

This, however, cannot be learned from data, i.e., there is no prediction sequence with

$$\lim_{n\to\infty}(g_n(X_{n+1},D_n)-\mathbf{E}\{Y_{n+1}|X_{n+1},D_n\})=0$$

a.s. for all stationary and ergodic sequence (cf., e.g., Györfi et al. [17]).

In general, our aim is to achieve the optimum

$$L^*=\lim_{n\to\infty}\min_g \mathbf{E}\{(g(X_{n+1},D_n)-Y_{n+1})^2\},$$

which again is impossible. However, there are universal Cesáro consistent prediction sequence g_n, i.e.,

$$\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}(g_i(X_{i+1},D_i)-Y_{i+1})^2=L^*$$

a.s. for all stationary and ergodic sequence. Such prediction sequences are called **universally consistent**. We show how to construct universally consistent predictors by combination of predictor experts.

One of the main ingredients of the construction is the following lemma, whose proof is a straightforward extension of standard arguments in prediction theory of individual sequences, see, for example, Kivinen and Warmuth [19], Singer and Feder [23].

Lemma 1 *Let $\widetilde{h}_1,\widetilde{h}_2,\ldots$ be a sequence of prediction strategies (experts), and let $\{q_k\}$ be a probability distribution on the set of positive integers. Assume that $\widetilde{h}_t(y_1^{t-1})\in[-B,B]$ and $y_1^n\in[-B,B]^n$. Define*

$$w_{t,k}=q_k e^{-(t-1)L_{t-1}(\widetilde{h}_k)/c}$$

with $c\geq 8B^2$, and the experts' weights by

$$v_{t,k}=\frac{w_{t,k}}{\sum_{i=1}^{\infty}w_{t,i}}.$$

Then the prediction strategy $\widetilde{g}$ defined by

$$\widetilde{g}_t(y_1^{t-1})=\sum_{k=1}^{\infty}v_{t,k}\widetilde{h}_k(y_1^{t-1})\qquad(t=1,2,\ldots)$$

has the property that for every $n\geq 1$,

$$L_n(\widetilde{g})\leq\inf_k\left(L_n(\widetilde{h}_k)-\frac{c\ln q_k}{n}\right).$$

Here $-\ln 0$ is treated as ∞.

We return to the problem of stationary and ergodic data $(X_1,Y_1),\ldots,(X_n,Y_n)$. Assume that $|Y_0|\leq B$. The elementary predictors (experts) will be denoted by $h^{(k,\ell)}$, $k,\ell=1,2,\ldots$. Each of the $h^{(k,\ell)}$ works as follows: Let G_ℓ be a quantizer of $\mathcal{R}^d$ and H_ℓ

be a quantizer of $\mathcal{R}$. For given k, ℓ, let I_n be the set of time instants $k < i < n$, for which a match of the k-length quantized sequences

$$G_\ell(x_{i-k}^i) = G_\ell(x_{n-k}^n)$$

and

$$H_\ell(y_{i-k}^{i-1}) = H_\ell(y_{n-k}^{n-1})$$

occurs. Then the prediction of expert $h^{(k,\ell)}$ is the average of the y_i's for which $i \in I_n$:

$$h_n^{(k,\ell)}(x_1^n, y_1^{n-1}) := \frac{\sum_{i \in I_n} y_i}{|I_n|}.$$

These elementary predictors are not universally consistent since for small k the bias tends to be large and for large k the variance grows considerably because of the few matchings. The same is true for the quantizers. The problem is how to choose k, ℓ in a data dependent way such as to obtain a universally consistent predictor. The solution is the combination of experts.

The combination of predictors can be derived according to the previous lemma. Let $\{q_{k,\ell}\}$ be a probability distribution on the set of all pairs (k, ℓ) of positive integers, and for $c = 8B^2$ put

$$w_{t,k,\ell} = q_{k,\ell} e^{-(t-1)L_{t-1}(h^{(k,\ell)})/c}$$

and

$$v_{t,k,\ell} = \frac{w_{t,k,\ell}}{\sum_{i,j=1}^{\infty} w_{t,i,j}}.$$

Then for the combined prediction rule

$$g_t(x_1^t, y_1^{t-1}) = \sum_{k,\ell=1}^{\infty} v_{t,k,\ell} h^{(k,\ell)}(x_1^t, y_1^{t-1})$$

the following universal consistency result holds:

Theorem 1 *(Györfi, Lugosi [15]). If the quantizers G_ℓ and H_ℓ "are asymptotically fine", and $\mathbf{P}\{Y_i \in [-B, B]\} = 1$, then the combined predictor g is universally consistent.*

16.3 Prediction for $0-1$ Loss: Pattern Recognition

In pattern recognition y_i takes on values in the finite set $\{1, 2, \ldots M\}$. At time instant i the classifier (predictor) decides on y_i based on the past observation (x_1^i, y_1^{i-1}).

After n rounds the empirical error for x_1^n, y_1^n is

$$L_n(g) = \frac{1}{n} \sum_{i=1}^{n} I_{\{g(x_1^i, y_1^{i-1}) \neq y_i\}}.$$

The natural loss is given by the $0-1$ loss, and $L_n(g)$ is the relative frequency of errors.

In the "static" version of pattern recognition the random variable Y takes on values in $\{1, 2, \ldots M\}$, and based on the random observation vector X one has to decide on Y. The decision rule (classifier) is defined by a decision function

$$g : \mathcal{R}^d \to \{1, 2, \ldots M\}.$$

The classifier has an error probability

$$L(g) = \mathbf{P}\{g(X) \neq Y\}.$$

As is well known, the error probability is minimized by the Bayes decision,

$$g^*(x) = i, \quad \text{if } \mathbf{P}\{Y = i | X = x\} = \max_j \mathbf{P}\{Y = j | X = x\}.$$

In pattern recognition we want to approach the Bayes decision if data $D_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ are given, which are i.i.d. copies of (X, Y). It is of considerable interest to find a pattern recognition rule

$$g_n(x) = g_n(x, D_n)$$

such that

$$L(g_n) = \mathbf{P}\{g_n(X) \neq Y | D_n\}$$

is close to $L(g^*)$ for all possible distributions of (X, Y). Similarly to the regression estimation problem, this may be achieved (cf. Devroye, Györfi and Lugosi [12]).

Clearly, this should be generalized to the case of dependent data $D_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$, where the data form a stationary and ergodic process. For given n, the problem is the following minimization:

$$\min_g \mathbf{P}\{g(X_{n+1}, D_n) \neq Y_{n+1}\},$$

which –as in the general regression estimation case– cannot be learned from data. Nor can there be a strategy achieving the optimum

$$R^* = \lim_{n \to \infty} \min_g \mathbf{P}\{g(X_{n+1}, D_n) \neq Y_{n+1}\}.$$

However, there are universal Cesáro consistent classifier sequences $g = \{g_n\}$, i.e., for the notation

$$L_n(g) = \frac{1}{n} \sum_{i=1}^{n} I_{\{g_i(X_{i+1}, D_i) \neq Y_{i+1}\}}$$

there exists g such that

$$\lim_{n \to \infty} L_n(g) = R^*$$

a.s. for all stationary and ergodic sequence. Such classifier sequences are called **universally consistent**. Györfi, Lugosi and Morvai [16] have constructed universally consistent classifiers by randomized combination of classifiers (experts).

The main ingredient of the proof is a beautiful result of Cesa-Bianchi et al. [7]. It states that, given a set of N experts, and a sequence of fixed length n, there exists a randomized predictor whose number of mistakes is not greater than that of the best classifier plus $\sqrt{(n/2)\ln N}$ for *all possible* sequences y_1^n. The simpler algorithm and statement cited below is due to Cesa-Bianchi [6]:

Lemma 2 *Let $\widetilde{h}^{(1)}, \ldots, \widetilde{h}^{(N)}$ be a finite collection of classifier strategies (experts). The classifier strategy $\widetilde{g}$ is defined by*

$$\widetilde{g}_t(y_1^{t-1}, x_1^t, u) = \begin{cases} 0 & \text{if } u > \dfrac{\sum_{k=1}^N I_{\left\{\widetilde{h}^{(k)}(y_1^{t-1}, x_1^t)=1\right\}} \widetilde{w}_t(k)}{\sum_{k=1}^N \widetilde{w}_t(k)} \\ 1 & \text{otherwise,} \end{cases}$$

$(t = 1, 2, \ldots, n)$, where for all $k = 1, \ldots, N$ and $t > 1$

$$\widetilde{w}_1(k) = 1 \quad \text{and} \quad \widetilde{w}_t(k) = e^{-\sqrt{8 \ln N/n} L_{t-1}(\widetilde{h}^{(k)})}.$$

Let $U_1, U_2, \ldots$ be i.i.d. uniformly distributed random variables on $[0, 1]$. Then at time moment t the randomized classification is

$$\widetilde{g}_t(y_1^{t-1}, x_1^t, U_t)$$

and for any $y_1^n \in \{0, 1\}^n$ and $x_1^n \in \mathcal{R}^{nd}$

$$\mathbf{E} L_n(\widetilde{g}) \leq \min_{k=1,\ldots,N} L_n(\widetilde{h}^{(k)}) + \sqrt{\frac{\ln N}{2n}}.$$

16.4 Prediction for Log Utility: Portfolio Selection

Consider investment in the stock market. We follow Breiman [5], Algoet and Cover [3], Cover [9] and Cover and Thomas [11]. The market consists of d stocks, and during one investment period (e.g., one day), it will be described by a return vector $x = (x^{(1)}, \ldots x^{(d)})$, where the j-th component $x^{(j)}$ is the factor by which capital invested in stock j grows during the market period. The investor is allowed to diversify his capital at the beginning of each day of trading according to a portfolio vector $b = (b^{(1)}, \ldots b^{(d)})$, the j-th component $b^{(j)}$ of which gives the proportion of the investor's capital invested in stock j. Assume that the portfolio vector $b = (b^{(1)}, \ldots b^{(d)})$ is a probability distribution, i.e. consumption of capital and short selling of stocks are excluded. If S_0 denotes the initial capital, then at the end of the day the investor will be left with a wealth of

$$S_1 = S_0 \sum_{j=1}^d b^{(j)} x^{(j)} = S_0(b, x),$$

where $(\cdot, \cdot)$ stands for the inner product.

For long term investment, assume the investor starts with an initial capital S_0 and let x_i be the return vector on day i. If $b = b_1$ is the portfolio vector the investor chooses for the first day of trading, he will accumulate a wealth of

$$S_1 = S_0 \cdot (b_1, x_1)$$

by the end of this day. For the second day, S_1 becomes his new initial capital and the portfolio vector for day two, b_2, may depend on x_1: $b_2 = b(x_1)$. Then

$$S_2 = S_0 \cdot (b_1, x_1) \cdot (b_2, x_2) = S_0 \cdot (b, x_1) \cdot (b(x_1), x_2).$$

In general, after the nth day of trading using a nonanticipating portfolio strategy $b_i = b(x_1^{i-1})$ $(i = 1, ..., n)$ the investor achieves

$$S_n = S_0 \prod_{i=1}^{n} (b(x_1^{i-1}), x_i) = S_0 e^{\sum_{i=1}^{n} \log(b(x_1^{i-1}), x_i)} = S_0 e^{nW_n(B)}.$$

The portfolio strategy $B = \{b(x_1^{i-1})\}$ is a sequence of functions, the quality of which is characterized by the average growth rate

$$W_n(B) = \frac{1}{n} \sum_{i=1}^{n} \log(b(x_1^{i-1}), x_i).$$

Obviously, the maximization of $S_n = S_n(B)$ and the maximization of $W_n(B)$ are equivalent.

Throughout, we assume that $x_1, x_2, \ldots$ are realizations of the random vectors X_1, X_2, ... drawn from the vector valued stationary and ergodic process $\{X_n\}_{-\infty}^{\infty}$ (note that by Kolmogorov's Theorem any stationary and ergodic process $\{X_n\}_1^{\infty}$ can be extended to a bi-infinite stationary process on some probability space $(\Omega, \mathcal{F}, \mathbf{P})$, such that ergodicity holds for both, $n \to \infty$ and $n \to -\infty$).

The fundamental limits for investment are delineated by results of Algoet and Cover [3], Algoet [1, 2], who showed that the so called log-optimum portfolio $B^* = \{b^*(\cdot)\}$ is the best possible choice. More precisely, on day n let $b^*(\cdot)$ be such that

$$\mathbf{E}\{\log(b^*(X_1^{n-1}), X_n) | X_1^{n-1}\} = \mathbf{E}\{\max_{b(\cdot)} \log(b(X_1^{n-1}), X_n) | X_1^{n-1}\}.$$

If $S_n^* = S_n(B^*)$ denotes the capital after day n achieved by a log-optimum portfolio strategy B^*, then for any portfolio strategy B with capital $S_n = S_n(B)$ and for any stationary ergodic process $\{X_n\}_{-\infty}^{\infty}$,

$$\limsup_{n \to \infty} \frac{1}{n} \log \frac{S_n}{S_n^*} \leq 0 \quad \text{almost surely}$$

and

$$\lim_{n \to \infty} \frac{1}{n} \log S_n^* = W^* \quad \text{almost surely,}$$

where

$$W^* = \mathbf{E}\left\{\max_{b(\cdot)} \mathbf{E}\{\log(b(X_{-\infty}^{-1}), X_0) | X_{-\infty}^{-1}\}\right\}$$

is the maximal growth rate of any portfolio.

These limit relations give rise to the following definition:

Definition 1 *A portfolio strategy B is called* **universal with respect to a class $\mathcal{C}$ of stationary and ergodic processes** $\{X_n\}_{-\infty}^{\infty}$, *if for each process in the class,*

$$\lim_{n \to \infty} \frac{1}{n} \log S_n(B) = W^* \quad \textit{almost surely.}$$

Universal strategies asymptotically achieve the best possible growth rate for all ergodic processes in the class. Algoet [1] introduced two portfolio strategies, and proved that the more complicated one is universal. The purpose of this section is to prove the universality of a strategy B similar to his second portfolio.

B is constructed as follows. We first define an infinite array of elementary portfolios $H^{(k,\ell)} = \{h^{(k,\ell)}(\cdot)\}$, $k, \ell = 1, 2, \ldots$ To this end, let $\mathcal{P}_\ell = \{A_{\ell,j}, j = 1, 2, \ldots, m_\ell\}$ be a sequence of finite partitions of the feature space $\mathcal{R}^d$, and let G_ℓ be the corresponding quantizer:

$$G_\ell(x) = j, \text{ if } x \in A_{\ell,j}.$$

With some abuse of notation, for any n and $x_1^n \in \mathcal{R}^{dn}$, we write $G_\ell(x_1^n)$ for the sequence $G_\ell(x_1), \ldots, G_\ell(x_n)$. Now, fix positive integers k, ℓ, and for each k-long string s of positive integers, define the partitioning portfolio

$$b^{(k,\ell)}(x_1^{n-1}, s) = \arg\max_b \prod_{\{k<i<n: G_\ell(x_{i-k}^{i-1})=s\}} (b, x_i), \qquad n > k+1,$$

if the product is nonvoid, and uniform b otherwise. If the product is nonvoid then

$$b^{(k,\ell)}(x_1^{n-1}, s) = \arg\max_b \frac{\sum_{\{k<i<n: G_\ell(x_{i-k}^{i-1})=s\}} \log(b, x_i)}{\left|\{k < i < n : G_\ell(x_{i-k}^{i-1}) = s\}\right|}, \qquad n > k+1.$$

From this we define the elementary portfolio $h^{(k,\ell)}$ by

$$h^{(k,\ell)}(x_1^{n-1}) = b^{(k,\ell)}(x_1^{n-1}, G_\ell(x_{n-k}^{n-1})), \qquad n = 1, 2, \ldots$$

That is, $h_n^{(k,\ell)}$ quantizes the sequence x_1^{n-1} according to the partition $\mathcal{P}_\ell$, and browses through all past appearances of the last seen quantized string $G_\ell(x_{n-k}^{n-1})$ of length k. Then it designs a fixed portfolio vector according to the returns on the days following the occurrence of the string.

Finally, let $\{q_{k,\ell}\}$ be a probability distribution on the set of all pairs (k, ℓ) of positive integers such that for all k, ℓ, $q_{k,\ell} > 0$. The strategy B then arises from weighing the elementary portfolio strategies $H^{(k,\ell)}$ according to their past performances and $\{q_{k,\ell}\}$:

$$b(x_1^{n-1}) := \frac{\sum_{k,\ell} q_{kl} S_{n-1}(H^{(k,\ell)}) h^{(k,\ell)}(x_1^{n-1})}{\sum_{k,\ell} q_{kl} S_{n-1}(H^{(k,\ell)})},$$

where $S_n(H^{(k,\ell)})$ is the capital accumulated after n days when using the portfolio strategy $H^{(k,\ell)}$ with initial capital S_0. Thus, after day n, the investor's capital becomes

$$S_n(B) = \sum_{k,\ell} q_{k,\ell} S_n(H^{(k,\ell)}).$$

The strategy B asymptotically achieves the best possible growth rate of wealth:

Theorem 2 *Assume that*

(a) the sequence of partitions is nested, that is, any cell of $\mathcal{P}_{\ell+1}$ is a subset of a cell of $\mathcal{P}_\ell$, $\ell = 1, 2, \ldots$;

(b) if $diam(A) = \sup_{x,y\in A} \|x - y\|$ *denotes the diameter of a set, then for any sphere* S *centered at the origin*

$$\lim_{\ell\to\infty} \max_{j: A_{\ell,j}\cap S\neq\emptyset} diam(A_{\ell,j}) = 0 \ .$$

Then the portfolio scheme B *defined above is universal with respect to the class of all ergodic processes such that* $\mathbf{E}\{|\log X^{(j)}|\} < \infty$, *for* $j = 1, 2, \ldots d$.

The first tool in the proof of Theorem 2 is known as Breiman's generalized ergodic theorem [4, 5], see also Algoet [2].

Lemma 3 (Breiman, [4]). *Let* $Z = \{Z_i\}_{-\infty}^{\infty}$ *be a stationary and ergodic process. Let* T *denote the left shift operator, shifting any sequence* $\{\ldots, z_{-1}, z_0, z_1, \ldots\}$ *one digit to the left. Let* f_i *be a sequence of real-valued functions such that for some function* f, $f_i(Z) \to f(Z)$ *almost surely. Assume that* $\mathbf{E}\sup_i |f_i(Z)| < \infty$. *Then*

$$\lim_{t\to\infty} \frac{1}{n} \sum_{i=1}^{n} f_i(T^i Z) = \mathbf{E} f(Z) \qquad a.s.$$

The second tool is a theorem due to Algoet and Cover ([3], Theorems 3 and 4).

Theorem 3 (Algoet and Cover, [3]).
(a) Let $\mathbf{Q}_{n\in\mathcal{N}\cup\{\infty\}}$ *be a family of regular probability distributions on* $(0,\infty)^d$ *such that* $\mathbf{E}\{|\log U_n^{(j)}|\} < \infty$ *for any coordinate of a return vector* $U_n = (U_n^{(1)}, \ldots, U_n^{(d)})$ *distributed according to* $\mathbf{Q}_n$. *In addition, let* $B^*(\mathbf{Q}_n)$ *be the set of all log-optimal portfolios w.r.t.* $\mathbf{Q}_n$, *i.e. of all portfolios* b *that attain* $\max_b \mathbf{E}\{\log(b, U_n)\}$. *Consider an arbitrary sequence* $b_n \in B^*(\mathbf{Q}_n)$. *If*

$$\mathbf{Q}_n \to \mathbf{Q}_\infty \quad \textit{weakly as } n \to \infty$$

then, for $\mathbf{Q}_\infty$*-almost all* u,

$$(b_n, u) \to (b^*, u) \quad (n \to \infty)$$

where the right hand side is constant as b^* *ranges over* $B^*(\mathbf{Q}_\infty)$.

(b) Let X *be a return vector on a probability space* $(\Omega, \mathcal{F}, \mathbf{P})$ *satisfying* $\mathbf{E}\{|\log X^{(j)}|\} < \infty$. *If* $\mathcal{F}_k$ *is an increasing sequence of sub-*σ*-fields of* $\mathcal{F}$,

$$\mathcal{F}_k \nearrow \mathcal{F}_\infty \subseteq \mathcal{F},$$

then

$$\mathbf{E}\left\{\max_{b\ \mathcal{F}_k-measurable} \mathbf{E}[\log(b, X)|\mathcal{F}_k]\right\} \nearrow \mathbf{E}\left\{\max_{b\ \mathcal{F}_\infty-measurable} \mathbf{E}[\log(b, X)|\mathcal{F}_\infty]\right\}$$

as $k \to \infty$.

PROOF OF THEOREM 2. We have to prove that

$$\liminf_{n\to\infty} W_n(B) = \liminf_{n\to\infty} \frac{1}{n}\log S_n(B) \geq W^* \quad \text{a.s.}$$

W.l.o.g. we may assume $S_0 = 1$, so that

$$\begin{aligned} W_n(B) &= \frac{1}{n}\log S_n(B) \\ &= \frac{1}{n}\log\left(\sum_{k,\ell} q_{k,\ell} S_n(H^{(k,\ell)})\right) \\ &\geq \frac{1}{n}\log\left(\sup_{k,\ell} q_{k,\ell} S_n(H^{(k,\ell)})\right) \\ &= \frac{1}{n}\sup_{k,\ell}\left(\log q_{k,\ell} + \log S_n(H^{(k,\ell)})\right) \\ &= \sup_{k,\ell}\left(W_n(H^{(k,\ell)}) + \frac{\log q_{k,\ell}}{n}\right). \end{aligned}$$

Thus

$$\begin{aligned} \liminf_{n\to\infty} W_n(H) &\geq \liminf_{n\to\infty}\sup_{k,\ell}\left(W_n(H^{(k,\ell)}) + \frac{\log q_{k,\ell}}{n}\right) \\ &\geq \sup_{k,\ell}\liminf_{n\to\infty}\left(W_n(H^{(k,\ell)}) + \frac{\log q_{k,\ell}}{n}\right) \\ &\geq \sup_{k,\ell}\liminf_{n\to\infty} W_n(H^{(k,\ell)}). \end{aligned} \tag{16.3}$$

In order to evaluate the lim inf on the right hand side we investigate the performance of the $b^{(k,\ell)}(\cdot,\cdot)$ on the stationary and ergodic sequence $X_0, X_{-1}, X_{-2}, \ldots$ First let k, ℓ and s be fixed. $\mathbf{P}_{j,s}^{(k,\ell)}$ denotes the (random) measure concentrated on $\{X_i : 1-j+k \leq i \leq 0, G_\ell(X_{i-k}^{i-1}) = s\}$ with

$$\mathbf{P}_{j,s}^{(k,\ell)}(A) := \frac{\sum_{i:1-j+k\leq i\leq 0, G_\ell(X_{i-k}^{i-1})=s} I_A(X_i)}{|\{i : 1-j+k \leq i \leq 0, G_\ell(X_{i-k}^{i-1}) = s\}|}.$$

If the above set of X_i's is void, then let $\mathbf{P}_{j,s}^{(k,\ell)} := \delta_{(1,\ldots,1)}$ be the probability measure concentrated on $(1,\ldots,1)$.

Observe that for all s with probability one

$$\mathbf{P}_{j,s}^{(k,\ell)} \to \begin{cases} \mathbf{P}_{X_0|G_\ell(X_{-k}^{-1})=s} & \text{if } \mathbf{P}(G_\ell(X_{-k}^{-1}) = s) > 0, \\ \delta_{(1,\ldots,1)} & \text{if } \mathbf{P}(G_\ell(X_{-k}^{-1}) = s) = 0 \end{cases} \tag{16.4}$$

weakly as $j \to \infty$. Indeed, let f be a bounded continuous function. By the ergodic theorem:

$$\begin{aligned} \int f(x)\mathbf{P}_{j,s}^{(k,\ell)}(dx) &= \frac{\frac{1}{|1-j+k|}\sum_{i:1-j+k\leq i\leq 0, G_\ell(X_{i-k}^{i-1})=s} f(X_i)}{\frac{1}{|1-j+k|}|\{i : 1-j+k \leq i \leq 0, G_\ell(X_{i-k}^{i-1}) = s\}|} \\ &\to \frac{\mathbf{E}\{f(X_0) I_{\{G_\ell(X_{-k}^{-1})=s\}}\}}{\mathbf{P}\{G_\ell(X_{-k}^{-1}) = s\}} \\ &= \mathbf{E}\{f(X_0)|G_\ell(X_{-k}^{-1}) = s\} \\ &= \int f(x)\mathbf{P}_{X_0|G_\ell(X_{-k}^{-1})=s}(dx) \quad \text{a.s.}, \end{aligned}$$

if $\mathbf{P}(G_\ell(X_{-k}^{-1}) = s) > 0$. If $\mathbf{P}(G_\ell(X_{-k}^{-1}) = s) = 0$, then with probability one $\mathbf{P}_{j,s}^{(k,\ell)}$ is concentrated on $(1,...,1)$ for all j, and

$$\int f(x)\mathbf{P}_{j,s}^{(k,\ell)}(dx) \quad = \quad f(1,...,1).$$

By definition, $b^{(k,\ell)}(X_{1-j}^{-1}, s)$ is a log-optimal portfolio w.r.t. $\mathbf{P}_{j,s}^{(k,\ell)}$. Let $b_{k,\ell}^*(s)$ be a log-optimal portfolio w.r.t. the limit distribution of $\mathbf{P}_{j,s}^{(k,\ell)}$. Then, using Theorem 3(a), we infer from (16.4) that as j tends to infinity the almost surely the following convergence holds:

$$(b^{(k,\ell)}(X_{1-j}^{-1}, s), x_0) \to (b_{k,\ell}^*(s), x_0))$$

for $\mathbf{P}_{X_0|G_\ell(X_{-k}^{-1})=s}$- and hence $\mathbf{P}_{X_0}$-almost all values of x_0. Here and in the following we exploit the fact that there are only finitely many values of s to be considered. In particular, we obtain

$$(b^{(k,\ell)}(X_{1-j}^{-1}, G_\ell(X_{-k}^{-1})), X_0) \to (b_{k,\ell}^*(G_\ell(X_{-k}^{-1})), X_0) \quad \text{a.s.} \tag{16.5}$$

as $j \to \infty$.

We are now in a position to apply Lemma 3. For $x = (...,x_{-1},x_0,x_1,...)$, set

$$f_i(x) := \log(h^{(k,\ell)}(x_{1-i}^{-1}), X_0) = \log(b^{(k,\ell)}(x_{1-i}^{-1}, G_\ell(X_{-k}^{-1})), X_0).$$

Note that

$$f_i(X) = |\log(h^{(k,\ell)}(X_{1-i}^{-1}), X_0)| \le \sum_{j=1}^{d} |\log X_0^{(j)}|,$$

the right hand side of which has finite expectation, and

$$f_i(X) \to (b_{k,\ell}^*(G_\ell(X_{-k}^{-1})), X_0) \quad \text{a.s. as } i \to \infty$$

from (16.5). As $n \to \infty$, Lemma 3 yields

$$\begin{aligned} W_n(H^{(k,\ell)}) &= \frac{1}{n}\sum_{i=1}^{n} \log(h^{(k,\ell)}(X_1^{i-1}), X_i) \\ &\to \mathbf{E}\{\log(b_{k,\ell}^*(G_\ell(X_{-k}^{-1})), X_0)\} \\ &= \mathbf{E}\{\max_{b(\cdot)} \mathbf{E}\{\log(b(G_\ell(X_{-k}^{-1})), X_0)|G_\ell(X_{-k}^{-1})\}\} \\ &= \epsilon_{k,\ell} \quad \text{a.s.} \end{aligned}$$

Therefore, by virtue of (16.3)

$$\liminf_{n\to\infty} W_n(B) \ge \sup_{k,\ell} \epsilon_{k,\ell} \qquad \text{a.s.}$$

Since the partitions $\mathcal{P}_\ell$ are nested, we have $\sigma(G_\ell(X_{-k}^{-1})) \subseteq \sigma(G_{\ell'}(X_{-k'}^{-1}))$ for all $\ell' \ge \ell, k' \ge k$, and the sequence

$$\begin{aligned} &\max_{b(\cdot)} \mathbf{E}\{\log(b(G_\ell(X_{-k}^{-1})), X_0)|G_\ell(X_{-k}^{-1})\} \\ &\quad = \max_{b \text{ is } \sigma(G_\ell(X_{-k}^{-1}))-\text{measurable}} \mathbf{E}\{\log(b, X_0)|G_\ell(X_{-k}^{-1})\} \end{aligned}$$

becomes a sub-martingale indexed by the pair (k, ℓ). This sequence is bounded by

$$\max_{b(\cdot)} \mathbf{E}\{\log(b(X_{-\infty}^{-1}), X_0))|X_{-\infty}^{-1}\},$$

which has finite expectation. The sub-martingale convergence theorem (see, e.g., Stout (1974) implies that this sub-martingale is convergent a.s., and $\sup_{k,\ell} \epsilon_{k,\ell}$ is finite. In particular, by the submartingale property, $\epsilon_{k,\ell}$ is a bounded double sequence increasing in k and ℓ, so that

$$\sup_{k,\ell} \epsilon_{k,\ell} = \lim_{k\to\infty} \lim_{\ell\to\infty} \epsilon_{k,\ell}.$$

Assumption (b) for the sequence of partitions implies that for fixed k

$$\sigma(G_\ell(X_{-k}^{-1})) \nearrow \sigma(X_{-k}^{-1})$$

as $\ell \to \infty$. Hence, by Theorem 3(b)

$$\begin{aligned}\lim_{\ell\to\infty} \epsilon_{k,\ell} &= \lim_{l\to\infty} \mathbf{E}\left\{\max_{b \text{ is } \sigma(G_\ell(X_{-k}^{-1}))-\text{measurable}} \mathbf{E}\{\log(b, X_0)|G_\ell(X_{-k}^{-1})\}\right\} \\ &= \mathbf{E}\left\{\max_{b \text{ is } \sigma(X_{-k}^{-1})-\text{measurable}} \mathbf{E}\{\log(b, X_0)|X_{-k}^{-1}\}\right\}.\end{aligned}$$

Applying Theorem 3(b) again with

$$\sigma(X_{-k}^{-1}) \nearrow \sigma(X_{-\infty}^{-1}) \quad \text{as } k \to \infty$$

finally yields

$$\begin{aligned}\sup_{k,\ell} \epsilon_{k,\ell} &= \lim_{k\to\infty} \mathbf{E}\left\{\max_{b \text{ is } \sigma(X_{-k}^{-1})-\text{measurable}} \mathbf{E}\{\log(b, X_0)|X_{-k}^{-1}\}\right\} \\ &= \mathbf{E}\left\{\max_{b \text{ is } \sigma(X_{-\infty}^{-1})-\text{measurable}} \mathbf{E}\{\log(b, X_0)|X_{-\infty}^{-1}\}\right\} \\ &= \mathbf{E}\left\{\max_{b(\cdot)} \mathbf{E}\{\log(b(X_{-\infty}^{-1}), X_0)|X_{-\infty}^{-1}\}\right\} \\ &= W^*\end{aligned}$$

and the proof of the theorem is finished. □

Bibliography

[1] P. Algoet, Universal schemes for prediction, gambling, and portfolio selection, *Annals of Probability* **20** (1992) 901–941.

[2] P. Algoet, The strong law of large numbers for sequential decisions under uncertainty, *IEEE Transactions on Information Theory* **40** (1994) 609–634.

[3] P. Algoet, T.M. Cover, Asymptotic optimality asymptotic equipartition properties of log-optimum investments, *Annals of Probability* **16** (1998) 876–898.

[4] L. Breiman, The individual ergodic theorem of information theory, *Annals of Mathematical Statistics* **31** (1957) 809–811. Correction. *Annals of Mathematical Statistics* **31** (1960) 809–810.

[5] L. Breiman, Optimal gambling systems for favorable games, *Proc. Fourth Berkeley Symp. Math. Statist. Prob.*, Univ. California Press, Berkeley 1 (1961) 65–78.

[6] N. Cesa-Bianchi, Analysis of two gradient-based algorithms for on-line regression, in *Proc. 10th Ann. Conf. Computational Learning Theory*, New York ACM Press (1997) 163–170.

[7] N. Cesa-Bianchi, Y. Freund, D.P. Helmbold, D. Haussler, R. Schapire, M.K. Warmuth, How to use expert advice, *Journal of the ACM* **44**(3) (1997) 427–485.

[8] Y.S. Chow, Local convergence of martingales and the law of large numbers, *Annals of Mathematical Statistics* **36** (1965) 552–558.

[9] T. Cover, Universal Portfolios, *Mathematical Finance* **1** (1991) 1–29.

[10] T. Cover, E. Ordentlich, Universal Portfolios with Side Information, *IEEE Transactions on Information Theory* **42** (1996) 348–363.

[11] T. Cover, J. Thomas, *Elements of Information Theory*, John Wiley and Sons (1991).

[12] L. Devroye, L. Györfi, G. Lugosi, *A Probabilistic Theory of Pattern Recognition*, Springer Verlag (1996).

[13] M. Feder, N. Merhav, M. Gutman, Universal prediction of individual sequences, *IEEE Transactions on Information Theory* **38** (1992) 1258–1270.

[14] L. Györfi, M. Kohler, A. Krzyżak, H. Walk, *A Distribution-Free Theory of Nonparametric Regression*, Springer Verlag (2002).

[15] L. Györfi, G. Lugosi, Strategies for sequential prediction of stationary time series, in *Modeling Uncertainty: An Examination of its Theory, Methods and Applications*, M. Dror, P. L'Ecuyer, F. Szidarovszky (Eds.), Kluwer Academic Publisher (2001).

[16] L. Györfi, G. Lugosi. G. Morvai, A simple randomized algorithm for consistent sequential prediction of ergodic time series, *IEEE Transactions on Information Theory* **45** (1999) 2642–2650.

[17] L. Györfi, G. Morvai, S. Yakowitz, Limits to consistent on-line forecasting for ergodic time series, *IEEE Transactions on Information Theory* **44** (1998) 886–892.

[18] D. Haussler, J. Kivinen, M. Warmuth, Sequential Prediction of Individual Sequences Under General Loss Functions, *IEEE Transactions on Information Theory* **44** (1998) 1906–1925.

[19] J. Kivinen, M.K. Warmuth, Averaging expert predictions, In H. U. Simon P. Fischer, editor, *Computational Learning Theory: Proceedings of the Fourth European Conference, EuroCOLT'99*, Springer, Berlin. Lecture Notes in Artificial Intelligence 1572 (1999) 153–167.

[20] N. Littlestone, M.K. Warmuth, The weighted majority algorithm, *Information and Computation* **108** (1994) 212–261.

[21] N. Merhav, M. Feder, Universal prediction, *IEEE Transactions on Information Theory* **44** (1998) 2124–2147.

[22] M. Opper, D. Haussler, Worst Case Prediction over Sequences under Log Loss, In: *The Mathematics of Information Coding, Extraction, and Distribution*, Springer Verlag (1997).

[23] A. Singer, M. Feder, Universal linear prediction by model order weighting, *IEEE Transactions on Signal Processing* **47** (1999) 2685–2699.

[24] C.J. Stone, Consistent nonparametric regression, *Annals of Statistics* **5** (1977) 595–645.

[25] W.F. Stout, *Almost sure convergence*, Academic Press, New York (1974).

[26] V.G. Vovk, Aggregating strategies, In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, Association of Computing Machinery, New York (1990) 372–383.

[27] V.G. Vovk, A Game of Prediction with Expert Advice, *Journal of Computer and System Sciences* **56**(2) (1998) 153–173.

[28] V.G. Vovk, Competitive on-line statistics, *International Statistical Review* **69**(2) (2001) 213–248.

[29] M. Weinberger, N. Merhav, M. Feder, Optimal Sequential Probability Assignment for Individual Sequences, *IEEE Transactions on Information Theory* **40** (1994) 384–396.

Chapter 17

Recent Advances in Statistical Learning Theory

M. Vidyasagar

Abstract. In this chapter, we discuss several advances in the area of statistical learning theory (SLT). The basic problem formulations are given, the "classical" known results are summarized, recently derived results are stated, and some applications are indicated to learning with inputs generated by a Markov chain.

17.1 Introduction

In this chapter, we discuss several advances in the area of statistical learning theory (SLT). The basic problem formulations are given, the "classical" known results are summarized, recently derived results are stated, and some applications are indicated to learning with inputs generated by a Markov chain.

Throughout the chapter, the principal reference is [18] and the references cited therein.

17.2 Problem Formulations

In this section, we state the two main problem classes that are studied in this chapter. These two problems are also the foundations of SLT.

17.2.1 Uniform convergence of empirical means

The problem of uniform convergence of empirical means has its origins in attempts to extend the classical Glivenko-Cantelli Lemma (see [5]) to more general situations. See the Notes and References in [17, 18], Chapter 3 for a description of the historical evolution of this problem.

Suppose X is a set, $\mathcal{S}$ is a σ-algebra of subsets of X, and that P is a probability measure on $(X, \mathcal{S})$. Thus $(X, \mathcal{S})$ is a measurable space and $(X, \mathcal{S}, P)$ is a probability space. For notational convenience, let X^∞ denote the infinite Cartesian product $\prod_{i=1}^\infty X$, and let $\mathcal{S}^\infty$ and P^∞ denote the corresponding product σ-algebra and product measure, respectively. Suppose that $\mathcal{F}$ is a family of functions mapping X into $[0,1]$ such that every function $f \in \mathcal{F}$ is measurable with respect to $(X, \mathcal{S})$.[1] Finally, suppose $\{\mathcal{X}_i\}_{i \geq 1}$ is an i.i.d. stochastic process assuming values in X, with the law P. Let $\mathbf{x} := (x_i) \in X^\infty$ denote a sample path of this stochastic process. For each function $f \in \mathcal{F}$, define the quantity

$$\widehat{E}_m(f; \mathbf{x}) := \frac{1}{m} \sum_{i=1}^m f(x_i),$$

and call it the **empirical mean** of the function f after m samples, based on the sample sequence $\mathbf{x}$. In some sense, the quantity $\widehat{E}_m(f; \mathbf{x})$ is an approximation to the "true mean"

$$E(f, P) := \int_X f(x)\, P(dx)$$

of the function f with respect to the probability measure P. The classical "law of large numbers" states that as the number of samples $m \to \infty$, the empirical mean $\widehat{E}_m(f; \mathbf{x})$ converges almost surely to the true mean $E(f, P)$. However, in SLT the interest is on so-called "finite time estimates," which tell us just how far the empirical mean is from

[1] Actually there is nothing special about the interval $[0,1]$, and it can be replaced by any *compact* interval. However, some technical difficulties arise if the range of the functions in $\mathcal{F}$ is unbounded.

the true mean for *finite* values of m. One such result is the well-known **Hoeffding's inequality**, which states that

$$P^\infty\{\mathbf{x} : |\widehat{E}_m(f;\mathbf{x}) - E(f,P)| > \epsilon\} \leq 2\exp(-2m\epsilon^2),\ \forall m, \epsilon.$$

Thus, after m samples have been drawn, it can be said with confidence $1 - 2e^{-2m\epsilon^2}$ that $|\widehat{E}_m(f;\mathbf{x}) - E(f,P)| \leq \epsilon$.

Given the family $\mathcal{F}$, let us define the quantity

$$q(m,\epsilon,P) := P^\infty\{\mathbf{x} \in X^\infty : \sup_{f\in\mathcal{F}} |\widehat{E}_m(f;\mathbf{x}) - E(f,P)| > \epsilon\}.$$

Thus, after m samples have been drawn, it can be said with confidence $1 - q(m,\epsilon,P)$ that *every* empirical mean $\widehat{E}_m(f;\mathbf{x})$ is within ϵ of the corresponding true mean $E(f,P)$.

Definition 1 *The pair $(\mathcal{F},P)$ is said to have the property of* **uniform convergence of empirical means (UCEM)** *if $q(m,\epsilon,P) \to 0$ as $m \to \infty$.*

The above definition can be extended readily to the case where the underlying probability measure P is not fixed ahead of time, but is known only to belong to a specified family $\mathcal{P}$ of probability measures on $(X,\mathcal{S})$. In this case we define

$$\begin{aligned}\overline{q}(m,\epsilon,\mathcal{P}) &:= \sup_{P\in\mathcal{P}} q(m,\epsilon,P) \\ &= \sup_{P\in\mathcal{P}} P^\infty\{\mathbf{x} \in X^\infty : \sup_{f\in\mathcal{F}} |\widehat{E}_m(f;\mathbf{x}) - E(f,P)| > \epsilon\}\end{aligned}$$

Definition 2 *The pair $(\mathcal{F},\mathcal{P})$ is said to have the property of* **uniform convergence of empirical means, uniformly in probability (UCEMUP)** *if $\overline{q}(m,\epsilon,\mathcal{P}) \to 0$ as $m \to \infty$.*

If the family $\mathcal{F}$ is finite, it follows readily from Hoeffding's inequality that

$$q(m,\epsilon,P) \leq 2|\mathcal{F}|\exp(-2m\epsilon^2),\ \forall m, \epsilon.$$

Hence every *finite* family automatically has the UCEM property. Moreover, if $\mathcal{P}$ is an arbitrary family of probability measures on $(X,\mathcal{S})$, it again follows from the above that

$$\overline{q}(m,\epsilon,\mathcal{P}) \leq 2|\mathcal{F}|\exp(-2m\epsilon^2),\ \forall m, \epsilon.$$

Hence every pair $(\mathcal{F},\mathcal{P})$ has the UCEMUP property if $\mathcal{F}$ is a finite family, for every $\mathcal{P}$. However, in case $\mathcal{F}$ is infinite, there is no *a priori* reason to assume that $(\mathcal{F},\mathcal{P})$ has the UCEMUP property. One of the principal aims of SLT is to derive necessary and sufficient conditions for a given pair $(\mathcal{F},\mathcal{P})$ to have the UCEMUP property, and in case this property holds, to derive explicit upper bounds for the quantity $\overline{q}(m,\epsilon,\mathcal{P})$.

17.2.2 Probably approximately correct learning

Over the years, researchers in machine learning have tried and rejected several mathematical models of how an abstract machine "learns" and "generalizes." Until the late 1970's, inductive learning theory was in vogue. In inductive learning, a machine is shown examples of a concept, say recognizing handwritten numerals. After a finite number of training inputs, the machine is then expected to generalize *perfectly*, that is, produce the correct output on *all* future queries. By the late 1970's there were far too many negative results about inductive learning; in effect, these results showed that inductive learning was possible only in very simple situations, where the concept that the machine is attempting to learn had a finite description.

A major breakthrough came in 1984 with the enunciation of the more relaxed notion of "probably approximately correct" (PAC) learning. In PAC learning, a machine is shown labelled training inputs generated at random. After a finite number of training samples, the machine is then presented with another randomly chosen test input. The efficacy of the learning process is judged by the probability that the machine produces the correct output on the randomly chosen test input. In contrast to inductive learning, in PAC learning the machine is *not* expected to produce a correct output on *all* test inputs, just *most* test inputs. As the training process proceeds, the probability that the machine produces the correct output on a random test input *approaches*, but never *equals*, one. As shown below, this seemingly simple relaxtion of what the machine is expected to do results in a spate of positive results, in contrast to the negative results of inductive learning theory.

The problem of PAC learning can be mathematically stated as follows: As above, suppose X is a set, $\mathcal{S}$ is a σ-algebra of subsets of X, $\mathcal{P}$ is a family of probability measures on $(X, \mathcal{S})$, and $\mathcal{F}$ is a family of measurable functions mapping X into $[0,1]$. Of particular interest is the situation where every function in $\mathcal{F}$ maps X into $\{0,1\}$, not just $[0,1]$. If every function in $\mathcal{F}$ is binary-valued, we say that $\mathcal{F}$ is a **concept class**, whereas in general we refer to $\mathcal{F}$ as a function class or a function family.

Learning proceeds as follows: The learner knows both $\mathcal{P}$ and $\mathcal{F}$. A probability measure $P \in \mathcal{P}$ is chosen; it may or may not be made known to the learner. An i.i.d. stochastic process $\{\mathcal{X}_i\}_{i \geq 0}$ is set up with the law P; this is the training sequence. A fixed but unknown function $f \in \mathcal{F}$, called the **target function**, is also chosen. Let $\mathbf{x} := \{x_1, x_2, \ldots\}$ denote a realization (or sample path) of the training sequence. For each input x_i, an "oracle" produces the value $f(x_i)$ of the unknown target function at the current training input. Thus, after m samples have been presented to the oracle, data available to the learner is the set of m "labelled samples"

$$[(x_1, f(x_1)), \ldots, (x_m, f(x_m))].$$

The objective of the learning process is to use this information to construct an approximation to the unknown function f; this approximation is called the "hypothesis". The procedure for constructing the hypothesis from the data is called the "algorithm." For the purposes of the present discussion, the "algorithm" is just an indexed family of maps $\{A_m\}_{m \geq 1}$ where

$$A_m : (X \times [0,1])^m \to \mathcal{F}.$$

The entity

$$A_m[(x_1, f(x_1)), \ldots, (x_m, f(x_m))] =: h_m(f; \mathbf{x}) \in \mathcal{F}$$

is called the **hypothesis** produced by the algorithm when the target function is f and the sample sequence is $\mathbf{x}$.

The hope is that, as the number of training samples $m \to \infty$, the hypothesis $h_m(f; \mathbf{x})$ "approaches" the true but unknown target function f. To formalize the notion of convergence, we need to have a metric on $\mathcal{F}$. In fact we use the metric induced by the norm on $L_1(X, \mathcal{S}, P)$.[2] Thus, if $f, g \in \mathcal{F}$ we define

$$d_P(f, g) := \int_X |f(y) - g(y)| \, P(dy).$$

With this definition, the quantity $d_P[f, h_m(f; \mathbf{x})]$ is called the **generalization error**. The generalization error has a very natural interpretation. Let $h_m(f; \mathbf{x})$ denote the hypothesis constructed by the algorithm after m training inputs, when the target function is f and the training sequence is $\mathbf{x}$. Now suppose a new input y, called the testing input, is generated at random according to the probability law P. The objective of machine learning is to predict the value of the unknown function $f(y)$ at this testing input. If the learner predicts that $f(y)$ equals $h_m(y)$, then the absolute prediction error is $|f(y) - h_m(y)|$. This quantity itself is random, since the testing input y is random. Thus the quantity $d_P[f, h_m(f; \mathbf{x})]$ is precisely the *expected value* of the absolute prediction error.

Next, the quantity

$$r(m, \epsilon) := \sup_{P \in \mathcal{P}} \sup_{f \in \mathcal{F}} P^\infty\{\mathbf{x} \in X^\infty : d_P[f, h_m(f; \mathbf{x})] > \epsilon\}$$

is called the **learning rate**. In defining the learning rate, we first look at the probability that the generalization error exceeds a prespecified accuracy level ϵ. Since f is unknown, we then take the supremum with respect to $f \in \mathcal{F}$, so that we get the worst-case estimate of this probability. Finally, in case the underlying probability measure P (that generates the training sample sequence) is itself unknown, we also take the supremum with respect to $P \in \mathcal{P}$. As a direct consequence of the definition of the learning rate, it follows that after m labelled training samples have been input to the learning algorithm, it can be said with confidence $1 - r(m, \epsilon)$ that the generalization error is less than ϵ, irrespective of what f is or P is.

Definition 3 *A learning algorithm $\{A_m\}$ is said to be* **probably approximately correct (PAC)** *to accuracy ϵ if $r(m, \epsilon) \to 0$ as $m \to \infty$, and to be* **PAC** *if $r(m, \epsilon) \to 0$ as $m \to \infty$ for each $\epsilon > 0$. The pair $(\mathcal{F}, \mathcal{P})$ is said to be* **PAC learnable** *if there exists a PAC algorithm.*

Suppose an algorithm is PAC, so that $r(m, \epsilon) \to 0$ as $m \to \infty$. Given a confidence level δ, the smallest number $m_0(\epsilon, \delta)$ with the property that

$$r(m, \epsilon) \leq \delta \ \ \forall m \geq m_0(\epsilon, \delta)$$

[2]Note that, since every function in $\mathcal{F}$ is bounded, it is also absolutely integrable.

is called the **sample complexity**. In general, it is virtually impossible to compute the sample complexity exactly. Instead, one can obtain upper bounds for $m_0(\epsilon, \delta)$. In other words, we can often compute a number $m_0(\epsilon, \delta)$ such that the above inequality holds, and through abuse of language, refer to it as the sample complexity. With this slightly incorrect usage, we can say with confidence $1 - \delta$ that the generalization error is no larger than ϵ provided we have at least $m_0(\epsilon, \delta)$ samples.

17.3 Summary of "Classical" Results

In this section we review some of the "classical" results in UCEM theory and PAC learning theory. The next section discusses more recent results.

The discussion in the present section is divided into two parts. In the first subsection, we study the so-called "fixed-distribution" case, in which the probability measure P that generates the sample sequence is known ahead of time. In the second subsection, we study the so-called "distribution-free" case, in which the probability measure P could be arbitrary, or equivalently, the family $\mathcal{P}$ of probability measures to which P is known to belong equals $\mathcal{P}^*$, the set of *all* probability measures on $(X, \mathcal{S})$. These are thus the two "extreme" cases. In both cases, the UCEM and PAC learning problems are fairly well-understood, which is one reason for focusing on them.

17.3.1 Fixed distribution case

Suppose P is a known fixed probability measure on $(X, \mathcal{S})$. In this case, a *necessary and sufficient* condition for the UCEM property to hold is known. A *sufficient* condition for PAC learnability is known, which is also *necessary* (and therefore necessary and sufficient) in case $\mathcal{F}$ is a concept class.

Let us begin with the UCEM problem. Given an m-tuple $\mathbf{x}_m = (x_1, \ldots, x_m)$, define

$$\mathbf{f}(\mathbf{x}_m) := [f(x_1) \ldots f(x_m)] \in [0,1]^m, \ \forall f \in \mathcal{F},$$

$$\mathcal{F}_m(\mathbf{x}) := \{\mathbf{f}(x_m) : f \in \mathcal{F}\} \subseteq [0,1]^m.$$

Let $\|\cdot\|_\infty$ denote the ℓ_∞-norm on $\mathbb{R}^m$, and let $L(\epsilon, \mathcal{F}_m(\mathbf{x}), \|\cdot\|_\infty)$ denote the *external covering number* of the set $\mathcal{F}_m(\mathbf{x})$ to accuracy ϵ, with respect to the ℓ_∞-norm. Thus $L(\epsilon, \mathcal{F}_m(\mathbf{x}), \|\cdot\|_\infty)$ denotes the smallest number of closed balls of radius ϵ needed to cover the set $\mathcal{F}_m(\mathbf{x})$, where the centers of the balls need not belong to $\mathcal{F}_m(\mathbf{x})$ and the radius is measured with respect to the ℓ_∞-norm. Finally, let lg denote the binary logarithm. With this notation, we can state the following theorem [15, 16, 14]:

Theorem 1 *The pair $(\mathcal{F}, P)$ has the UCEM property if and only if*

$$\lim_{m \to \infty} \frac{E\{\lg[L(\epsilon, \mathcal{F}_m(\mathbf{x}), \|\cdot\|_\infty)], P\}}{m} = 0.$$

In the case of PAC learning, we change notation slightly. Let $N(e, \mathcal{F}, d_P)$ denote the covering number (*not* external covering number) of $\mathcal{F}$ with respect to the metric d_P to accuracy ϵ. The distinction between N and L is that in the case of N, the centers

of the closed balls of radius ϵ that cover $\mathcal{F}$ are themselves required to belong to $\mathcal{F}$, whereas there is no such requirement in the case of L. Also, note that we are now covering the entire function class $\mathcal{F}$ instead of a subset of $[0,1]^m$ as earlier.

We now introduce the so-called "minimal empirical risk (MER) algorithm." Suppose that, for some ϵ, the covering number $N(\epsilon/2, \mathcal{F}, d_P)$ is finite. Given an accuracy ϵ, choose a minimal $\epsilon/2$-cover (not an ϵ-cover) of $\mathcal{F}$, and denote it by $\{g_1, \ldots, g_N\}$. Thus the finite collection $\{g_1, \ldots, g_N\}$ has the following properties:

1. $g_i \in \mathcal{F}$ for each i.
2. For each $f \in \mathcal{F}$, there exists an index j such that $d_P(f, g_j) < \epsilon/2$.
3. There is no collection of cardinality less than N that satisfies the above two properties.

Given a sample sequence $\mathbf{x}$ and an integer m, the MER algorithm is as follows: Choose a minimal $\epsilon/2$-cover $\{g_1, \ldots, g_N\}$ as above. For each index i, compute the corresponding empirical error as follows:

$$\widehat{J}_i := \frac{1}{m} \sum_{j=1}^{m} |f(x_j) - g_i(x_j)|.$$

Note that the quantity $f(x_j)$ is known for each j as the output of the oracle. Thus the quantity $\widehat{J}_i$ can be computed on the basis of the available data. Observe that $\widehat{J}_i$ is just an empirical approximation to the quantity $d_P(f, g_i)$ based on the first m samples. Now choose the hypothesis h_m as the (or a) g_i such that $\widehat{J}_i$ is minimal.

With this notation we can state the following theorem [14, 1]:

Theorem 2 *Suppose $N(\epsilon/2, \mathcal{F}, d_P)$ is finite for some ϵ. Then the MER algorithm is PAC to accuracy ϵ. Moreover,*

$$r(m, \epsilon) \leq N \exp(-m\epsilon^2/8)$$

if $\mathcal{F}$ is a function class, and

$$r(m, \epsilon) \leq N \exp(-m\epsilon/32)$$

if $\mathcal{F}$ is a concept class. Thus, if $\mathcal{F}$ is a function class, it is enough to use at least

$$m_0(\epsilon, \delta) = \frac{8}{\epsilon^2} \ln \frac{N}{\delta}$$

samples, while if $\mathcal{F}$ is a concept class, it is enough to use at least

$$m_0(\epsilon, \delta) = \frac{32}{\epsilon} \ln \frac{N}{\delta}$$

samples.

The above result shows only that the MER algorithm is PAC *to a specified accuracy* ϵ. However, it is possible to modify the above algorithm so that it is PAC (to all accuracies).

The previous theorem gives only *sufficient* conditions for PAC learnability. The next result, giving a *necessary* condition in case of concept classes, is due to [1].

Theorem 3 *Suppose $\mathcal{F}$ is a concept class; then the finiteness of $N(e, \mathcal{F}, d_P)$ for every ϵ is both necessary and sufficient for $(\mathcal{F}, P)$ to be PAC learnable.*

17.3.2 Distribution-free case

In the distribution-free case, it is assumed that $\mathcal{P}$, the set of probability measures to which P is known to belong, in fact equals $\mathcal{P}^*$, the set of *all* probability measures on $(X, \mathcal{S})$. In short, the underlying probability measure P could be anything. In this case, a central role is played by a concept called the Vapnik-Chervonenkis (VC)-dimension, which is an integer that measures the "richness" of a concept class. With due care, the notion can also be extended to function classes.

Definition 4 *Suppose $\mathcal{F}$ is a concept class defined over a set X. A set $S = \{x_1, \ldots, x_n\} \subseteq X$ is said to be* **shattered** *by the family $\mathcal{F}$ if, for each vector $\mathbf{e} \in \{0,1\}^n$, there exists a corresponding function $f_{\mathbf{e}} \in \mathcal{F}$ such that $f_{\mathbf{e}}(x_i) = e_i$. The* **Vapnik-Chervonenkis (VC)-dimension** *of the family $\mathcal{F}$ is the largest integer n such that there exists a set S of cardinality n that is shattered by $\mathcal{F}$, and is denoted by VC-dim($\mathcal{F}$).*

Thus the set S is shattered by $\mathcal{F}$ if each of the 2^n classifications of the elements in S is realized by some function in $\mathcal{F}$. Obviously, if S is shattered by $\mathcal{F}$, so is every subset of S. Hence, as the integer n becomes larger, it becomes more and more difficult to find sets of cardinality n that are shattered by $\mathcal{F}$. If a stage comes where no set of cardinality greater than n is shattered by $\mathcal{F}$, then $\mathcal{F}$ has finite VC-dimension.

To extend the notion to function classes, we introduce the so-called "Heaviside" function $\eta(\cdot) : \mathbb{R} \to \mathbb{R}$, defined by

$$\eta(s) := \begin{cases} 1 & \text{if } s \geq 0, \\ 0 & \text{if } s < 0. \end{cases}$$

Definition 5 *Suppose $\mathcal{F}$ is a function family mapping X into $[0,1]$. For each function $f \in \mathcal{F}$, define an associated function $\overline{f} : X \times [0,1] \to \{0,1\}$ by $\overline{f}(x,c) := \eta[f(x) - c]$. Define $\overline{\mathcal{F}} = \{\overline{f} : f \in \mathcal{F}\}$. The* **P-dimension** *of $\mathcal{F}$ is defined as the VC-dimension of $\overline{\mathcal{F}}$, and is denoted by P-dim($\mathcal{F}$).*

We begin with a discussion of the UCEMUP property. The result below is found in [14].

Theorem 4 *Suppose $\mathcal{F}$ is a concept class, and let $\mathcal{P} = \mathcal{P}^*$. Then the pair $(\mathcal{F}, \mathcal{P}^*)$ has the UCEMUP property if and only if VC-dim($\mathcal{F}$) is finite. Moreover, we have*

$$\overline{q}(m, \epsilon, \mathcal{P}^*) \leq 4 \left(\frac{2em}{d} \right)^d \exp(-m\epsilon^2/8), \ \forall m, \epsilon.$$

The result below is found in [17], Chapter 5, and is a refinement of a corresponding result in [7].

Theorem 5 *Suppose $\mathcal{F}$ is a function class, and let $\mathcal{P} = \mathcal{P}^*$. Then the pair $(\mathcal{F}, \mathcal{P}^*)$ has the UCEMUP property if P-dim($\mathcal{F}$) is finite. Moreover, we have*

$$\overline{q}(m, \epsilon, \mathcal{P}^*) \leq 8 \left(\frac{16e}{\epsilon} \ln \frac{16e}{\epsilon} \right)^d \exp(-m\epsilon^2/32), \ \forall m, \epsilon < e/(4 \lg e) \approx 0.47.$$

However, the finiteness of P-dim($\mathcal{F}$) is not necessary in general for the pair $(\mathcal{F}, \mathcal{P}^*)$ to have the UCEMUP property.

Next we discuss the problem of PAC learning a family $\mathcal{F}$ when $\mathcal{P} = \mathcal{P}^*$. In this case, a central role is played by the notion of a 'consistent' algorithm. An algorithm is said to be **consistent** if

$$h_m(f; \mathrm{x})(x_i) = f(x_i),\ i = 1, \ldots, m,\ \forall \mathrm{x}, \mathrm{f}.$$

In plain English, a consistent algorithm is one that always produces a hypothesis that matches the data.

Theorem 6 below, taken from [2], established a close connection between PAC learning and the VC dimension. It was extended to function families in [7], and the slight refinement given here is found in [17].

Theorem 6 *Suppose $\mathcal{F}$ is a concept class and that VC-dim($\mathcal{F}$) $< \infty$. Let $\{A_m\}$ be a consistent algorithm. Then*

$$r(m, \epsilon) \leq 2 \left(\frac{2em}{d} \right)^d 2^{-m\epsilon/2}.$$

To ensure that $r(m, \epsilon) \leq \delta$, it is enough to take

$$m_0(\epsilon, \delta) = \max \left\{ \frac{8d}{\epsilon} \lg \frac{8e}{\epsilon}, \frac{4}{\epsilon} \lg \frac{2}{\delta} \right\}$$

samples.

Theorem 7 *Suppose $\mathcal{F}$ is a function family and that P-dim($\mathcal{F}$) $< \infty$. Let $\{A_m\}$ be a consistent algorithm. Then*

$$r(m, \epsilon) \leq 8 \left(\frac{32e}{\epsilon} \ln \frac{32e}{\epsilon} \right)^d \exp(-m\epsilon/32).$$

To ensure that $r(m, \epsilon) \leq \delta$, it is enough to take

$$m_0(\epsilon, \delta) = \frac{32}{\epsilon} \left[d \left(\ln \frac{32e}{\epsilon} \ln \ln \frac{32e}{\epsilon} \right) + \ln \frac{8}{\delta} \right]$$

samples.

17.4 Recent Advances

17.4.1 Intermediate families of probability measures

In the previous section, we have seen results that address that two 'extreme' situations, namely: the fixed distribution case, and the distribution-free case. In this section, we discuss the case where the underlying family of probability measures $\mathcal{P}$ is neither of the two extreme cases.

We begin with the UCEM problem. The main result below is proved in [17].

Theorem 8 *Suppose all symbols are as in Theorem 1. Let $\mathcal{P}$ be an arbitrary family of probability measures on $(X, \mathcal{S})$. Then the pair $(\mathcal{F}, \mathcal{P})$ has the UCEMUP property if and only if*

$$\sup_{P \in \mathcal{P}} \lim_{m \to \infty} \frac{E\{\lg[L(\epsilon, \mathcal{F}_m(\mathbf{x}), \|\cdot\|_\infty)], P\}}{m} = 0.$$

Next let us study the PAC learning problem. For this purpose, we use the total variation metric on the set of probability measures on $(X, \mathcal{S})$. Suppose P, Q are two probability measures on $(X, \mathcal{S})$. Then the **total variation metric** $\rho(P, Q)$ is defined by

$$\rho(P, Q) := \sup_{A \in \mathcal{S}} |P(A) - Q(A)|.$$

Since ρ is a metric on $\mathcal{P}^*$, it also induces a corresponding topology on $\mathcal{P}^*$. We address two situations, namely: when $\mathcal{P}$ is a compact set in this topology, and when $\mathcal{P}$ has an interior point in this metric.

The results presented below are from [10].

Theorem 9 *Suppose $\mathcal{P}$ is a compact family of probability measures with respect to the total variation metric. Suppose $\mathcal{F}$ is a function family. Then the pair $(\mathcal{F}, \mathcal{P})$ is PAC learnable if*

$$\sup_{P \in \mathcal{P}} N(\epsilon, \mathcal{F}, d_P) < \infty, \ \forall \epsilon.$$

If $\mathcal{F}$ is a concept class, the condition is also necessary.

Theorem 10 *Suppose $\mathcal{P}$ has an interior point with respect to the total variation metric. Suppose $\mathcal{F}$ is a concept class. Then the pair $(\mathcal{F}, \mathcal{P})$ is PAC learnable if and only if VC-dim$(\mathcal{F}) < \infty$.*

17.4.2 Learning with prior information

Let us recall the definition of the learning rate function in the fixed distribution case. We have

$$r(m, \epsilon) := \sup_{f \in \mathcal{F}} P^\infty\{\mathbf{x} \in X^\infty : d_P[f, h_m(f; \mathbf{x})] > \epsilon\}.$$

This definition is 'asymmetric' in the samples $\mathbf{x} \in X^\infty$ and the target function $f \in \mathcal{F}$. Specifically, an algorithm is still allowed to fail 'occasionally' for some samples $\mathbf{x}$, but is expected to work at a uniform rate for *every* unknown function f. It is reasonable therefore to consider an alternate formulation of learning in which there is a prior probability distribution Q on the set $\mathcal{F}$, and it is known that the unknown target function f is distributed according to Q. Let us define the quantity

$$s(m, \epsilon) := (Q \times P^\infty)\{(f, \mathbf{x}) : d_P[f, h_m(f; \mathbf{x})] > \epsilon\}.$$

Thus, after m labelled samples have been drawn, it can be stated with confidence $1 - s(m, \epsilon)$ that the generalization error is no larger than ϵ. However, in contrast with the conventional definition of PAC learning as exemplified by the definition of $r(m, \epsilon)$, in the present case the poor performance of the algorithm (in the sense that the generalization error exceeds ϵ) can be caused *either* by an unrepresentative set of samples $\mathbf{x}$ *or* by a difficult target function f. This formulation might be called a 'fully stochastic' formulation.

Definition 6 *The triplet $(\mathcal{F}, P, Q)$ is said to be* **learnable with prior information (WPI)** *if there exists an algorithm such that $s(m, \epsilon) \to 0$ as $m \to \infty$.*

It is clear that PAC learnability implies learnability WPI. It would be interesting to see how much weaker learnability WPI is compared to PAC learnability. In this study, a central role is played by a notion called dispersability. Recall that a 'partition' π of the set $\mathcal{F}$ is a collection of sets $\pi = \{\mathcal{G}_i\}_{i=1}^r$ such that $\mathcal{G}_i \cap \mathcal{G}_j = \emptyset$ and $\cup_{i=1}^r \mathcal{G}_i = \mathcal{F}$. Here r is called the cardinality of the partition.

Definition 7 *Given the family $\mathcal{F}$ with the probability measure Q, suppose $\pi = \{\mathcal{G}_i\}_{i=1}^r$ is a partition of $\mathcal{F}$ of cardinality r. Then the* **dispersion** *of the partition π is defined as*

$$\text{disp}(P^{\infty}) := \sum_{i=1}^{r} \inf_{f \in \mathcal{F}} \int_{\mathcal{G}_i} d_P(g, f)\, Q(dg).$$

The triplet $(\mathcal{F}, P, Q)$ is said to be **dispersable** *if*

$$\lim_{r \to \infty} \inf_{|\pi| = r} \text{disp}(\pi) = 0.$$

Note that the quantity $\int_{\mathcal{G}_i} d_P(g, f)\, Q(dg)$ measures the 'spread' of the set $\mathcal{G}_i$ with respect to the probability measures P and Q. Thus the quantity $\text{disp}(\pi)$ measures how small the total spread of $\mathcal{F}$ is under the partition π. The set $\mathcal{F}$ is 'dispersable' if the dispersion of $\mathcal{F}$ can be made arbitrarily small by choosing partitions of arbitrarily large cardinality.

Now we can state the main result on learning WPI. This result is due to [3].

Theorem 11 *The triplet $(\mathcal{F}, P, Q)$ is learnable WPI if it is dispersable. In case $\mathcal{F}$ is a concept class, the dispersability of $(\mathcal{F}, P, Q)$ is also necessary for learnability WPI.*

There is a corresponding version of dispersability for the distribution-free case that provides a sufficient condition for distribution-free learnability WPI, and is also necessary in case $\mathcal{F}$ is a concept class; see [3] for details.

The discussion is concluded with the following result, which shows that learnability WPI is automatically guaranteed whenever the set X is separable.

Theorem 12 *Suppose X is a separable metric space and that $\mathcal{S}$ is the associated Borel σ-algebra. Let P be an arbitrary probability measure on $(X, \mathcal{S})$. Suppose $\mathcal{F} \subseteq [0,1]^X$ be a collection of measurable functions and let Q be an arbitrary probability measure on $\mathcal{F}$. Then $(\mathcal{F}, P, Q)$ is dispersable.*

17.5 Learning with Dependent Inputs

17.5.1 Problem formulations

Until now we have studied the learning problem under the assumption that the training sample sequence $\mathbf{x}$ is the sample path of an i.i.d. sequence. This assumption allows us to use standard inequalities such as Hoeffding's inequality in estimating the learning

rate in various situations. However, the assumption that the training samples are i.i.d. effectively restricts the PAC learning formulation to the case where the function f to be learnt is a 'static' function, in the sense that the current oracle output f_i depends only on the current input x_i. However, there are many situations in control and system theory in which the current output f_i depends on the entire past input $x_i, x_{i-1}, \ldots$. It is desirable to have a version of PAC learning that caters to this more general situation.

Hereafter, let X^∞ denote the *doubly infinite* cartesian product $\prod_{i=-\infty}^{\infty} X$, and let $\mathcal{S}^\infty$ denote the corresponding product σ-algebra associated with $\mathcal{S}$. Suppose $\tilde{P}$ is a stationary probability measure on the product space $(X^\infty, \mathcal{S}^\infty)$, and suppose $\{\mathcal{X}_t\}_{t=-\infty}^{\infty}$ is a stochastic process with the law $\tilde{P}$. Note that, in contrast to the previous notation, we are now assuming that the sample sequence is a *two-sided* stochastic process. This is done to avoid some technicalities associated with one-sided stochastic processes. Moreover, it is well-known that any one-sided stochastic process can be embedded within a two-sided process. However, it is assumed that the learning process has a definite starting time, which is denoted as $t = 0$. Let $\tilde{P}_0$ denote the one-dimensional marginal law of $\tilde{P}$. Thus each of the coordinate random variables $\mathcal{X}_t$ has the law $\tilde{P}_0$. In case the stochastic process is i.i.d., we have $\tilde{P} = (\tilde{P}_0)^\infty$. But this need not be so in the general case.

It is a straight-forward matter to extend the definitions of the UCEMUP property and PAC learning to the case where the input samples are not necessarily i.i.d. Specifically, all one has to do is to replace the probability measure P^∞ by $\tilde{P}$ in the definitions of $q(m, \epsilon, P)$ and $r(m, \epsilon)$. Thus we have

$$q_{\text{mixing}}(m, \epsilon, \tilde{P}) := \tilde{P}\{\mathbf{x} \in X^\infty : \sup_{f \in \mathcal{F}} |\hat{E}_m(f; \mathbf{x}) - E(f, \tilde{P}_0)| > \epsilon\},$$

$$r_{\text{mixing}}(m, \epsilon) := \sup_{f \in \mathcal{F}} \tilde{P}\{\mathbf{x} \in X^\infty : d_{\tilde{P}_0}[f, h_m(f; \mathbf{x})] > \epsilon\}.$$

We shall also use the symbols $q_{\text{iid}}(m, \epsilon, \tilde{P}_0)$ and $r_{\text{iid}}(m, \epsilon)$ defined in the obvious manner, namely:

$$q_{\text{iid}}(m, \epsilon, \tilde{P}_0) := (\tilde{P}_0)^\infty\{\mathbf{x} \in X^\infty : \sup_{f \in \mathcal{F}} |\hat{E}_m(f; \mathbf{x}) - E(f, \tilde{P}_0)| > \epsilon\},$$

$$r_{\text{iid}}(m, \epsilon) := \sup_{f \in \mathcal{F}} (\tilde{P}_0)^\infty\{\mathbf{x} \in X^\infty : d_{\tilde{P}_0}[f, h_m(f; \mathbf{x})] > \epsilon\}.$$

In other words, $q_{\text{iid}}(m, \epsilon, \tilde{P}_0)$ is what $q_{\text{mixing}}(m, \epsilon, \tilde{P})$ would be if the samples were i.i.d., and similarly for $r_{\text{mixing}}(m, \epsilon)$.

The above notation addresses the situation when there is just one known fixed probability measure $\tilde{P}$. If $\tilde{P}$ itself is unknown and is known only to belong to some family $\tilde{\mathcal{P}}$, then all one has to do is to take the supremum of both these quantities with respect to $\tilde{P} \in \tilde{\mathcal{P}}$.

17.5.2 Definition of β-mixing

The main approach to studying the problems of UCEM and PAC learning in the case of dependent inputs is to assume that the stochastic process generating the sample

sequence is 'nearly' i.i.d., specifically, that the dependence of $\mathcal{X}_{t+k}$ on $\mathcal{X}_t$ approaches zero as the temporal separation $k \to \infty$. In the literature, there are several ways of defining the dependence coefficient between two random variables. Among these, the α-mixing coefficient, the β-mixing coefficient, and the ϕ-mixing coefficient are among the most widely used. For the present purposes, it turns out that the β-mixing coefficient is most appropriate.

Given the stochastic process $\{\mathcal{X}_t\}$, let $\Sigma_{-\infty}^0$ denote the σ-algebra generated by the random variables $\mathcal{X}_t, t \leq 0$, and let Σ_k^∞ denote the σ-algebra generated by the random variables $\mathcal{X}_t, t \geq k$. Finally, let $\overline{\Sigma}_1^{k-1}$ denote the σ-algebra generated by $\Sigma_{-\infty}^0$ and Σ_k^∞. Note that $\overline{\Sigma}_1^{k-1}$ is precisely the σ-algebra generated by all $\mathcal{X}_t$ *except* $\mathcal{X}_1, \ldots, \mathcal{X}_{k-1}$. This is the meaning of the notation. Given the probability measure $\widetilde{P}$, there exists a unique probability measure, which is denoted here by $\tau_0(\widetilde{P})$, such that the following properties hold:

1. The joint law of $\mathcal{X}_t, t \leq 0$ is the same under both $\widetilde{P}$ and $\tau_0(\widetilde{P})$.
2. The joint law of $\mathcal{X}_t, t \geq k$ is the same under both $\widetilde{P}$ and $\tau_0(\widetilde{P})$.
3. Under $\tau_0(\widetilde{P})$, the variables $\mathcal{X}_t, t \geq 1$ are independent of the variables $\mathcal{X}_t, t \leq 0$.

Now we can define the β-mixing coefficient.

Definition 8 *The* β-**mixing coefficient** *of the stochastic process* $\{\mathcal{X}_t\}$ *is defined as*

$$\sup_{A \in \overline{\Sigma}_1^{k-1}} |\tau_0(\widetilde{P})(A) - \widetilde{P}(A)|.$$

The stochastic process $\{\mathcal{X}_t\}$ *is said to be* β-**mixing** *if* $\beta(k) \to 0$ *as* $k \to \infty$, *and is said to be* **geometrically** β-**mixing** *if there exists a constant* $\lambda < 1$ *such that* $\beta(k) = O(\lambda^k)$.

A very useful inequality for β-mixing stochastic processes is given next.

Lemma 1 *Suppose* $f : X^\infty \to [0, 1]$ *and that* f *depends only on* $\mathcal{X}_0, \mathcal{X}_k, \ldots, \mathcal{X}_{lk}$. *Then*

$$|E(f, \widetilde{P}) - E(f, (\widetilde{P}_0)^\infty)| \leq l\beta(k).$$

The point of the lemma is as follows: Suppose we compute the expected value of $f : X^\infty \to [0, 1]$ assuming that the random variables $\mathcal{X}_t$ are all independent. Then the maximum error we make in doing so is no larger than $l\beta(k)$.

17.5.3 UCEM and PAC learning with β-mixing inputs

In this subsection we present some results on the UCEM and PAC learning problems when the i.i.d. input sequence is replaced by a β-mixing input sequence.

The first result concerns the UCEM property. It is taken from [8] and generalizes an earlier result from [13].

Theorem 13 *Suppose $\{\mathcal{X}_t\}$ is a β-mixing stationary stochastic process with the law $\widetilde{P}$, and let $\widetilde{P}_0$ denote the one-dimensional marginal of $\widetilde{P}$. Suppose that the pair $(\mathcal{F}, (\widetilde{P}_0)^\infty)$ has the UCEM property. Then the pair $(\mathcal{F}, \widetilde{P})$ also has the UCEM property. Moreover, choose a sequence of integers $\{k_m\}$ such that $k_m \leq m$ $\forall m$, and define $l_m := \lfloor m/k_m \rfloor$. Then*

$$q_{\text{mixing}}(m, \epsilon, \widetilde{P}) \leq k_m \{\max\{q_{\text{iid}}(l_m + 1, \epsilon, \widetilde{P}_0), q_{\text{iid}}(l_m, \epsilon, \widetilde{P}_0)\} + m\beta(k_m).$$

To discuss PAC learning with a β-mixing input sequence, we first introduce an auxiliary notion, called a quasi sub-additive algorithm. Suppose $\{A_m\}$ is an algorithm. Suppose we are given a training input sequence $\{x_1, x_2, \ldots\}$; equivalently, suppose we are given a sample path of a doubly infinite stochastic process, and that learning starts at time $t = 0$. After m time instants, we can form the hypothesis

$$h_m(f; \mathbf{x}) := A_m[(x_1, f(x_1)), \ldots, (x_m, f(x_m))].$$

Now suppose $k \leq m$ is arbitrary. Define $l = \lfloor m/k \rfloor$, and let $r = m - lk$. Define the k sets

$$I_1 := \{1, k+1, 2k+1, \ldots, lk+1\}, \ldots, I_r := \{r, k+r, 2k+r, \ldots, lk+r\},$$

$$I_{r+1} := \{r+1, k+r+1, \ldots, (l-1)k+r+1\}, \ldots, I_k := \{k, 2k, \ldots, lk\}.$$

Note that the sets I_i are pairwise disjoint, and that their union is precisely $\{1, \ldots, m\}$. Now let us run the algorithm on each of these subsample sequences, as follows:

$$g_{m,1}(f; \mathbf{x}) := A_{|I_1|}[(x_1, f(x_1)), \ldots, (x_{lk+1}, f(x_{lk+1})],$$

and so on. In general, we have

$$g_{m,i}(f; \mathbf{x}) := A_{|I_i|}[(x_i, f(x_i)), i \in I_i].$$

Thus $g_{m,i}(f; \mathbf{x})$ is the hypothesis generated by using only a subset of the labelled samples. Now the algorithm is said to be **quasi sub-additive** if it is the case that

$$d_P[f, h_m(f; \mathbf{x})] \leq \sum_{i=1}^{k} \frac{|I_i|}{m} d_P[f, g_{m,i}(f; \mathbf{x})].$$

This complicated-looking inequality has a simple interpretation. If we run the learning algorithm on all m labelled samples, the resulting generalization error is $d_P[f, h_m(f; \mathbf{x})]$. On the other hand, suppose we divide the m labelled samples into k blocks of nearly equal length. If k divides m (so that $r = 0$), then all k blocks have equal length. Otherwise r blocks will have one more element than the rest. Let us run the same learning algorithm on each of these k blocks. Then the resulting generalization error is $d_P[f, g_{m,i}(f; \mathbf{x})]$ for block i. Now the quasi sub-additivity inequality states simply that the generalization error that we get by using *all* the labelled samples is no larger than the average of the k generalization errors we get by using only roughly m/k subsamples. Since one would naturally expect that the generalization error decreases as more and more samples are used, this assumption appears to be very natural. In fact, it is shown below that in some natural situations such as consistent learnability, the assumption does hold.

The main result for quasi sub-additive learning algorithms is given next. It is taken from [9].

Theorem 14 *Suppose $\{A_m\}$ is a quasi sub-additive learning algorithm, and is PAC for a pair $(\mathcal{F}, P)$. Let $r_{\text{iid}}(m, \epsilon)$ denote the learning rate of the algorithm when the training samples are i.i.d. Suppose now the same algorithm is run on a β-mixing input sequence. Then the algorithm continues to be PAC. Moreover, if we choose a sequence $\{k_m\}$ such that $k_m \leq m$ for all m and define $l_m := \lfloor m/k_m \rfloor$, we have that*

$$r_{\text{mixing}}(m, \epsilon) \leq k_m \max\{r_{\text{iid}}(l_m + 1, \epsilon), r_{\text{iid}}(l_m, \epsilon)\} + m\beta(k_m).$$

To illustrate a concrete application of this theorem, let us consider the problem of consistent learnability. A pair $(\mathcal{F}, P)$ is said to be **consistently learnable** if every consistent algorithm is PAC. Define the quantity

$$\widehat{d}_m(f, g; \mathbf{x}) := \frac{1}{m} \sum_{i=1}^{m} |f(x_i) - g(x_i)|,$$

and note that $\widehat{d}_m(f, g; \mathbf{x})$ denotes the empirical estimate of the distance $d_P(f, g)$ based on the first m samples in $\mathbf{x}$. Now define the quantity

$$\phi_{m,f}(\mathbf{x}) := \sup\{d_P(f, g) : g \in \mathcal{F} \text{ and } \widehat{d}_m(f, g; \mathbf{x}) = 0\},$$

It can be shown ([6] and also [18], Theorem 6.4) that the pair $(\mathcal{F}, P)$ is consistently learnable if and only if

$$\psi(m, \epsilon) := \sup_{f \in \mathcal{F}} P^{\infty}\{\mathbf{x} \in X^{\infty} : \phi_{m,f}(\mathbf{x}) > \epsilon\} \to 0 \text{ as } m \to \infty,$$

Let us now examine the behaviour of the stochastic process $\{\phi_{m,f}(\mathbf{x})\}$ for a fixed $f \in \mathcal{F}$. If $f(x_i) = g(x_i)$ for all i between 1 and m, it is also so for every subset of the x_i's. Hence if we denote $\mathbf{x}_i := \{x_j, j \in I_i\}$, it follows from the definition of ϕ that

$$\phi_{|I_i|,f}(\mathbf{x}_i) \geq \phi_{m,f}(\mathbf{x}), \ \forall i.$$

In particular, it is certainly the case that $\phi_{m,f}(\mathbf{x}) \leq \sum_{i=1}^{k} \frac{|I_i|}{m} \phi_{|I_i|,f}(\mathbf{x}_i)$. Hence, by mimicking the proof of Theorem 14 it can be shown that if $\{\phi_{m,f}(\mathbf{x})\}$ approaches zero with an i.i.d. input sequence, it also does so when $\{x_i\}$ is the sample path of a β-mixing stochastic process. Finally, the fact that the algorithm is consistent implies that

$$d_P[f, h_m(f; \mathbf{x})] \leq \phi_{m,f}(\mathbf{x}), \ \forall m, \mathbf{x}, f.$$

In turn this implies that the algorithm is PAC. (For a more careful argument, which is the same as the above in essence, see [9].)

17.6 Applications to Learning with Inputs Generated by a Markov Chain

To apply the results of the preceding section to a concrete situation, it is desirable to have some specific results that tell us when a stochastic process is β-mixing. One such very useful result is given here, based on some fundamental work in [11].

The problem of PAC learning with a training sequence generated by a Markov chain is studied in [4, 12]. Thus the present results are an improvement over the contents of these papers.

Throughout, we consider Markov chains described by the recursion relation

$$x_{t+1} = f(x_t, e_t), \tag{17.1}$$

where $x_t \in \mathbb{R}^k, e_t \in \mathbb{R}^m$ for some integers k, m, and $\{e_t\}$ is a stationary noise sequence. It is assumed that the following assumptions are satisfied:

A1. The function $f : \mathbb{R}^k \times \mathbb{R}^m \to \mathbb{R}^k$ is 'smooth,' i.e., is C^∞, and in addition, f is globally Lipschitz continuous. Thus there exist constants L and K such that

$$|f(x,u) - f(y,v)| \leq L|x-y| + K|u-v|. \tag{17.2}$$

A2. The noise sequence $\{e_t\}$ is i.i.d., has finite variance, and has a continuous multivariate density function $\phi(\cdot)$ that is positive in some neighbourhood Ω of the origin in $\mathbb{R}^m$.

A3. When $e_t = 0 \ \forall t$, the 'unforced' system $x_{t+1} = f(x_t, 0)$ is globally exponentially stable with the origin as the unique globally attractive equilibrium. This means that there exist constants M' and $\lambda < 1$ such that $|x_t| \leq M'|x_0|\lambda^t, \ \forall t \geq 1, \ \forall x_0$. By taking $M := \max\{M', 1\}$, one can write the above inequality as $|x_t| \leq M|x_0|\lambda^t, \ \forall t \geq 0, \ \forall x_0$.

A4. The associated deterministic control system $x_{t+1} = f(x_t, u_t)$ is 'globally forward accessible' from the origin with the control set Ω. In other words, for every $y \in \mathbb{R}^n$, there exist a time N and a control sequence $\{u_0, \ldots, u_{N-1}\} \subseteq \Omega$ such that, with $x_0 = 0$ we have $x_N = y$.

A5. The associated deterministic control system is 'locally controllable' to the origin with the control set Ω. This means that there exists a neighbourhood $\mathcal{B}$ of the origin in $\mathbb{R}^k$ such that, for every $y \in \mathcal{B}$ there exist a time N and a control sequence $\{u_0, \ldots, u_{N-1}\} \subseteq \Omega$ such that, with $x_0 = y$ we have $x_N = 0$.

Now we can state the main result.

Theorem 15 *Suppose assumptions A1 through A5 hold. Then the state sequence $\{x_t\}$ is geometrically β-mixing.*

17.7 Conclusions

In this chapter we have studied the problems of the uniform convergence of empirical means (UCEM) and of probably approximately correct (PAC) learning. We have derived the classical results, and indicated some extensions to the case of dependent inputs.

Bibliography

[1] G. M. Benedek and A. Itai, Learnability by fixed distributions, *Proc. First Workshop on Computational Learning Theory*, Morgan-Kaufmann, San Mateo, CA (1988) 80–90.

[2] A. Blumer, A. Ehrenfeucht, D. Haussler and M. Warmuth, Learnability and the Vapnik-Chervonenkis dimension, *J. ACM* **36**(4) (1989) 929–965.

[3] M. C. Campi and M. Vidyasagar, Learning with prior information, *IEEE Trans. Autom. Control* **46**(11) (2001) 1682–1695.

[4] D. Gamarnik, Extension of the PAC framework to finite and countable Markov chains, *Proc. Twelfth Annual Conf. on Computational Learning Theory* (1999).

[5] B. V. Gnedenko, *Theory of Probability*, 4th Ed., Chelsea, New York (1968).

[6] B. Hammer, Learning recursive data, *Math. of Control, Signals and Systems* **12**(1) (1999) 62–79.

[7] D. Haussler, Decision theoretic generalizations of the PAC model for neural net and other learning applications, *Information and Computation* **100** (1992) 78–150.

[8] R. L. Karandikar and M. Vidyasagar, Rates of uniform convergence of empirical means with mixing processes, *Stat. and Probab. Letters* (2003) to appear.

[9] R. L. Karandikar and M. Vidyasagar, Probably approximately correct learning with β-mixing input sequences, (manuscript under preparation).

[10] S. R. Kulkarni and M. Vidyasagar, Learning decision rules under a family of probability measures, *IEEE Trans. Info. Theory* **43**(1) (1997) 154–166.

[11] S. P. Meyn and R. L. Tweedie, *Markov Chains and Stochastic Stability*, Springer-Verlag, London (1993).

[12] R. Meir, Nonparametric time series prediction through adaptive model selection, *Machine Learning* **39**(1) (2000) 5–34.

[13] A. Nobel and A. Dembo, A note on uniform laws of averages for dependent processes, *Stat. & Probab. Letters* **17** (1993) 169–172.

[14] V. N. Vapnik, *Estimation of Dependences Based on Empirical Data*, Springer-Verlag (1982).

[15] V. N. Vapnik and A. Ya. Chervonenkis, On the uniform convergence of relative frequencies to their probabilities, *Theory of Probab. Appl.* **16**(2) (1971) 264–280.

[16] V. N. Vapnik and A. Ya. Chervonenkis, Necessary and and sufficient conditions for the uniform convergence of means to their expectations, *Theory of Probab. Appl.* **26**(3) (1981) 532-553.

[17] M. Vidyasagar, *A Theory of Learning and Generalization*, Springer-Verlag, London (1997).

[18] M. Vidyasagar, *Learning and Generalization*, (Second Edition), Springer-Verlag, London (2003).

Chapter 18

Neural Networks in Measurement Systems (an engineering view)

Gabor Horváth

Abstract. The goal of this chapter is to show that neural networks can play an important role in measurement systems. The chapter formulates relevant engineering questions; some of them are in close relation to the basic theoretical questions of neural computation, others are related only to practical applications. The chapter contrasts the theoretical results with the engineering questions.

18.1 Introduction

From an engineering point of view neural networks are efficient tools to solve different practical problems. Since the last years of the eighties many successful neural network-based applications have been developed in various application areas. It is enough to mention only a few well know examples as the NETtalk system [1] that converts English speech to phonemes, or ALVINN, a neural network that learns to steer a vehicle along a single lane of a highway [2], but one can mention the many recognition, modeling and control tasks e.g. [3]–[7] have been solved using different neural networks. Most of these and similar applications were developed using a practitioner's approach, which is somewhat different even from the classical way of engineering problem solving. The neural networks applied for these tasks were not designed in the classical sense. Instead, in the construction of the neural solutions the trial and error approach played a great role.

This way of the development of neural solutions was justified by the good performance of these applications. With neural networks rather complex problems could be solved much more easily than using classical engineering solutions. In spite of these successful applications more and more questions were arising which could not be answered using the practitioner's approach. What is the reason of this success? Why neural networks are so capable computing devices? Is there a well-defined way to design a neural network with given capability? Some general answers came soon, and more and more specific answers are coming from mathematics, where the results are formulated in theorems. The statements of theorems are important not only from a theoretical view, but also from the viewpoint of practice.

However, from an engineering point of view, some further questions need answers. How can we apply these results for practical applications? Can they be applied at all in practice? What conditions must be fulfilled for the validity of the results and what will happen if these conditions are not fulfilled or only partly fulfilled? A further problem from the engineering viewpoint is that, in solving a practical task, it is not enough if we can prove that a solution exists. We must realize this solution, we must implement the neural network and during implementation several further constraints must be satisfied; moreover these constraints are often in contradiction with those of the theoretical results.

The goal of this chapter is to formulate such questions in relation to measurement problems. First a short introduction is given about the measurement problems. It will be shown that measurement is closely related to modeling, so neural networks can play an important role in solving measurement problems as general modeling tools.

Next we review the most important practical questions about neural networks. What capabilities of neural networks can be utilised in measurement systems? How can we construct a neural network for a given task? How can we validate the neural solution? This section also gives an overview of the relevant theoretical results and contrasts these results and the engineering questions. Besides the classical neural architectures the chapter deals with some basic results of kernel methods and support vector machines as they give answers to many questions that cannot be answered using classical neural approaches.

At the end we deal with the nature of the knowledge in measurement problems, and

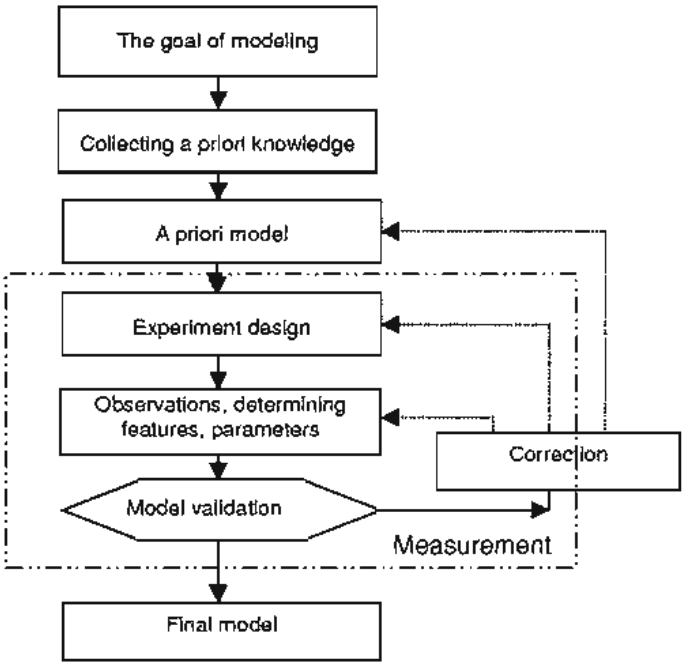


Figure 18.1: The basic steps of system modeling

with some special questions of implementation. This part overviews the most important constraints coming from implementation, and gives some possible ways, how to satisfy them at the expense of some compromise.

18.2 Measurement and Modeling

Measurement is the primary way of information collection. It is an empirical process to obtain experimental data (observations), and to extract knowledge from the observations. Measurement can be defined only in relation to modeling. It is always embedded in modeling, an autonomous phase of the whole modeling process. The relation of modeling and measurement is shown in Figure 18.1.

In every modeling task the following main steps can be distinguished:

- collection of prior information,
- selection of model set, model structure,
- experiment design and data collection,
- model parameter estimation,
- model validation.

In system modeling many different approaches can be applied depending on the available prior information, the goal of modeling, what part of the world has to be modeled and what aspects are to be considered, etc.

Model set selection means that the relation between inputs and outputs of a system is formulated in a general mathematical form. This mathematical form defines the structure of the model and defines a set of parameters, the values of which have to be determined using the observations. Model classes can be categorized in different ways depending on the aspects taken into consideration. Based on the *system characteristics* we can distinguish between

- static or dynamic,
- deterministic or stochastic,
- continuous-time or discrete-time,
- lumped parameter or distributed parameter,
- linear or non-linear,
- time invariant or time variant models.

All these differentiations are important for the further steps of the whole modeling process.

Independently from the previous aspect we can build *parametric* or *nonparametric* models. In parametric models a definite model structure is selected and only a limited number of parameters must be determined using observations. In many cases there is some physical insight about the system, we know what important parts of the system can be distinguished, how these parts are connected, so we know the structure of the model. In these cases physical models can be built. Physical models are typical parametric models. In nonparametric models there is no definite model structure and the system's behavior is described by the response of the system for special excitation signal. Nonparametric models can be built if we have less knowledge about the system. Typical nonparametric description of a (linear) system is the impulse response or the frequency characteristics.

Based on *prior information* we can speak about white box, grey box or black box models. When both the structure and the parameters of the model are completely known – complete physical knowledge is available – we have a *white box* model. White box models can be constructed from the prior information without the need of any observations. When the model construction is based only on observed data, we speak about input-output or behavioral model. An input-output model is often called empirical or *black box* model as the system to be modeled is considered as a black box, which is characterized with its input-output behavior without any detailed information about its structure. In black box modeling the model structure does not reflect the structure of the physical system, thus the elements of the model structure have no physical meaning. Instead, such model structure has to be chosen that is flexible enough to represent a large class of systems. The white box and the black box models represent extreme situations. In most of the modeling tasks we have certain physical information, however this is not complete (*grey box* modeling). We can construct a model, the structure of which – at least partly – reflects the available physical insight. The parameters of the model, however, are not known or only partly known, and they

must be estimated from observed data. The model will be fitted empirically using observations. In black box modeling – contrary to physical – a general model class has to be selected, where the model structure is not determined entirely by selecting the model class. We have to determine the size of the structure, the number of model parameters (e.g. in a polynomial model class the maximum order of the polynomial, etc.). In black box modeling both the model size (model complexity) and the numerical values of the parameters should be determined using the observations. In all cases we assume that the system to be modeled implements an $f : \mathbb{R}^n \rightarrow \mathbb{R}$ mapping, however the scalar output is used only for simplicity. The mapping that determines the relation between inputs and outputs of the system $y^*(i) = f(\mathbf{x}^*(i))$, is represented by a set of input-output measurement data $\{\mathbf{x}(i), y(i)\}_{i=1}^{\rho}$. The observation data usually differ from the inputs and outputs of the system, as neither $\mathbf{x}^*(i)$, nor $y^*(i)$ can be observed directly, the observations are burdened with measurement noise as shown in Figure 18.2. Here $\mathbf{n}_{m,x}(i)$ and $n_{m,y}(i)$ denotes the measurement noise at the input and output respectively. In many cases the input measurement noise can be neglected and the relation between the input and the output observations can be written as $y(i) = f(\mathbf{x}^*(i)) + n_{m,y}(i)$. This means that only an additive noise component can be found at the output observations. However, in most of the practical measurement problems both the input and the output observations are noisy, so $\mathbf{x} = \mathbf{x}^* + \mathbf{n}_{m,x}$ and $y = y^* + n_{m,y}$.

In modeling the mapping of the model f_M will approximate in some sense the mapping of the system. The model also implements an $\mathbb{R}^n \rightarrow \mathbb{R}$ mapping $y_M(i) = f_M(\mathbf{x}(i), \Theta)$, where y_M is the output of the model and Θ is its parameter vector. In this respect both physical and black box models can be considered as parametric ones, where first the model structure, then the model parameters must be determined. In this chapter we assume that the input-output mapping of the system to be modeled can be described by a continuous function $y^* = f(\mathbf{x}^*)$ where $f \in C$. From this point of view modeling is a function approximation problem: the input-output mapping of the model, $f_M(.)$ approximates the mapping of the system $f(.)$. This relationship can be expressed in several different forms, however, a general form can be described as a weighted sum of given basis functions $\{G_j(.)\}_{j=1}^{m}$:

$$y_M(i) = \sum_{j=1}^{m} \alpha_j G_j(\mathbf{x}(i)) \tag{18.1}$$

where the parameter vector is defined as $\Theta = [\alpha_1 \alpha_2 \ldots \alpha_m]^T$. There are many possible basis function sets, which can be applied successfully in black-box system modeling (nonlinear function approximation). For example we can form polynomial functions, when the mapping of the system is approximated by a polynomial, or we can use complex exponentials, which means, that the mapping of the system is approximated by a Fourier series. But a Taylor expansion, wavelets or Volterra series can also be applied [8], [9]. Among the black box structures neural networks play an important role too.

The selection between the possibilities is usually based on prior information about the system, or on some general (theoretical or practical) advantages or drawbacks of the different black box architectures. Having selected a basis function set two problems

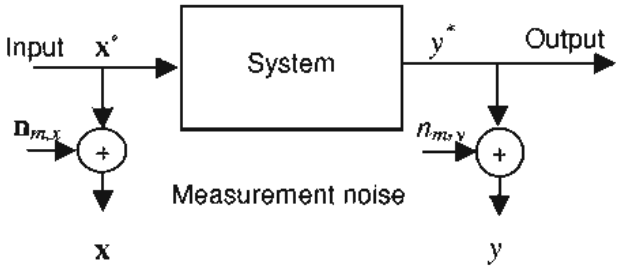


Figure 18.2: A general measurement setup

must be solved: (i) how many basis functions are required in this representation, and (ii) how the parameters of the model can be estimated. The first question belongs to the model selection problem, the selection of the size of the model, the second question is a parameter estimation problem. The answers to these questions can be divided into two groups. There are general solutions, which are valid for all black-box modeling approaches, and there are special results which apply only for a given black-box architecture. The general answers are related mainly to the model size problem, while for the parameter estimation task different methods have been developed for the different black-box architectures.

Experiment design. For collecting observations we have to design experiments, to design input signals, and measure the output signals as responses for the input ones. In the step of experiment design the circumstances of input-output data collection is determined, the excitation signals are designed. The construction of excitation signals depends on the prior knowledge about the system. E.g. different excitation signals have to be used for modeling a linear and a non-linear system; the excitation depends on whether the system is static or dynamic, deterministic or stochastic, etc. In non-linear system modeling the selection of excitation signals depends on the required validity range of the model. Different excitations can be used if model validity is required only for the neighborhood of an operating point or if such model is needed that reflects some important aspects of the system in many different operating points, etc.

In general we have to select input signals that will excite the system in such a way that the input-output data can be observed during the experiment carry enough knowledge about the system. In system modeling it is often required to design new and significantly modified experiments during the identification process, where the knowledge collected from the previous experiments are utilized. In many cases experiment design means to determine what signals can be measured at all, so this step depends largely on the practical modeling task. For getting observations from a physical system sensors are used, so in experiment design sensors have a great role. In many measurement (modeling) tasks the observation of physical quantities cannot be reached directly, only indirect ways or using the combination of several sensors signals can result in relevant observation data. There are modeling problems where there is no possibility to design excitation, we can only measure the input and output data in normal operating conditions. This situation may happen when experiment design would be too expensive

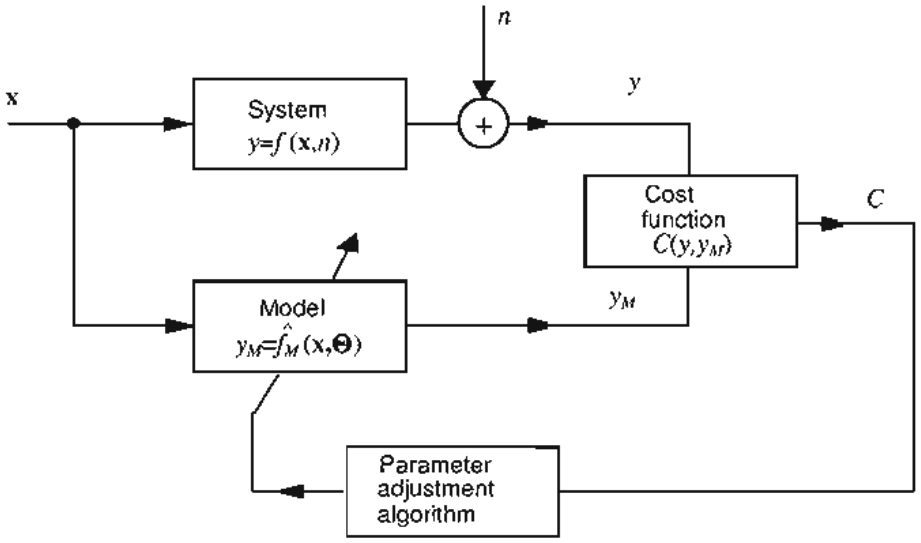


Figure 18.3: The parameter estimation process

(see e.g.[7]), or when the system to be modeled is an autonomous one, which operates without explicit input signals, etc. The general and special questions of experiment design are beyond the scope of this chapter, interested readers can consult relevant books, e.g. [10], [11].

Model parameter estimation. In parametric models the relation between inputs and outputs of a system is formulated in a mathematical form, where this mathematical form defines the structure of the model and defines a set of parameters. The values of the parameters have to be determined using the observations. There are well-developed methods, which give estimates for the numerical values of the parameters. These parameter estimation methods utilize different types of knowledge available about the system to be modeled. We may have prior information about the nature of the parameters to be determined (e.g. we may have physical knowledge about the possible range of certain parameters, we may have information if some parameters are deterministic ones or can be considered as random variables with known probability distribution, etc.), but the essential part of the knowledge used for parameter estimation is a set of measurement data, a set of observations about the system.

Parameter estimation is a way of adjusting the model parameters for fitting the observations according to some criterion or cost function. The parameter estimation process is shown in Figure 18.3. Depending on the cost function (which also may depend on the prior information about our system) we can speak e.g. about least square (LS) estimation, weighted least square (WLS) estimation, maximum likelihood (ML) estimation or Bayes estimation [12].

A cost function is a measure of the quality of the model, it is a function of the error between the model output y_M and the system output y:

$$C(\Theta) = C\left(y - y_M(\Theta, Z^P)\right) \tag{18.2}$$

where Z^P denotes the set of measured data pairs

$$Z^P = \{\mathbf{x}(i), y(i)\}_{i=1}^P. \tag{18.3}$$

If both model structure and model size are fixed, model parameters have to be estimated. For parameter estimation a cost function must be selected. The most common measure of discrepancy is the sum of squared errors,

$$C_{LS}(\Theta) = \frac{1}{2}\sum_{i=1}^{P}\left(y(i) - y_M(\Theta, i)\right)^2, \tag{18.4}$$

i.e. usually quadratic cost functions are used. Besides quadratic cost functions other ones can also be applied. The selection of cost function depends on the characteristics of measurement noise. For Gaussian additive output noise – which is a good model of the noise in most of the measurement problems – quadratic cost function is optimal, however for other noise characteristics *absolute* cost function, ϵ-*insensitive* function, etc. should be preferred. If the input measurement noise cannot be neglected the estimation can be based on the errors-in-variables (EIV) cost function [13], [14]:

$$C_{EIV}(\Theta) = \frac{1}{2}\sum_{i=1}^{P}\left[\frac{\left(y(i) - y_M(\Theta, i)\right)^2}{\sigma_{y,i}^2} + \frac{\left(x(i) - \widehat{x}^*(i)\right)^2}{\sigma_{x,i}^2}\right] \tag{18.5}$$

where $\widehat{x}^*(i)$ is an estimate of the true, but unknown input $x^*(i)$, and $\sigma_{x,i}^2$ and $\sigma_{y,i}^2$ are the input and output noise variances, respectively.

The parameter estimate based on the quadratic criterion function is the least square estimate:

$$\widehat{\Theta}_{LS} = \arg\min_{\Theta} C(\Theta). \tag{18.6}$$

The observations are noisy measurements, so if something is known about the statistical properties of the measurement noise some statistical estimation can be applied. One of the most common statistical estimation is maximum likelihood (ML) estimation, when we select the estimate, which makes the given observations most probable:

$$\widehat{\Theta}_{ML} = \arg\max_{\Theta} f\left(\mathbf{y}_P \mid \mathbf{x}_P, \Theta\right), \tag{18.7}$$

where $f\left(\mathbf{y}_P \mid \mathbf{x}_P, \Theta\right)$ denotes the conditional probability density function of the observations. If the parameter to be estimated is a random variable and if its probability density function is known, we can apply Bayes estimation. Although Bayes estimation has certain optimality property, it is rarely applied in practice because it requires more prior information than ML or LS estimations. Detailed discussion of parameter estimation methods, especially for linear dynamic systems, see e.g. [10], [11], [12], etc.

Model validation. The final step of system modeling is the *validation* of the model. For validation a proper criterion as a fitness of the model must be used. The choice of this criterion is extremely important as it determines a measure of the quality of the model. From the result of the validation we can decide whether or not the model is good enough for our purpose. If not, an iterative cycle of structure selection

(model class and model size selection), experiment design, parameter estimation and model evaluation must be repeated until a suitable representation is found; so system modeling is an iterative process.

For statistical estimations the estimate of a parameter is never a numerical value only, the estimate is a random variable and its statistical features should also be given. The most important features are the expected value and the variance or covariance matrix. An important quality feature if the estimate is biased or unbiased. If the expected value of the estimate equals to the true value of the parameter the estimate is unbiased, otherwise it is biased. The variance gives information about the dispersion of the estimate. The smaller is the variance, the better is the estimate. For certain statistical estimates there exists a lower bound of the variance. This bound given by the Cramer Rao inequality. The Cramer Rao bound is the key quantity to determine if the estimate is efficient (in this case the lower bound is reached) or not. It is proved that if efficient estimate exists, the ML estimate is efficient. Variance has a role in determining the confidence interval of the measurement result, which tells us how close the estimate is to the true value of the parameter. All these quality parameters depend on the number and quality (e.g. measurement noise characteristics) of observation data used for the estimation. In black-box modeling the characterization of the quality of the model can be given using observations. This problem is closely related to the validation of the neural models, so this problem will be discussed in the next section.

18.3 Neural Networks

Neural networks are distributed information processing systems made up of a great number of highly interconnected identical or similar simple processing units [15]. The processing units implement nonlinear mappings between their inputs and output $\varphi : \mathbb{R}^n \rightarrow \mathbb{R}$, where the mapping depends on the type of the processing unit. For perceptrons a ϕ activation function is used: $\varphi(x) = \phi(\mathbf{w}^T\mathbf{x})$, where $\phi(.)$ is usually a sigmoidal function, like logistic $\phi(s) = \frac{1}{1+e^{-s}}$ or the hyperbolic tangent function, both of which are continuous, bounded, strictly monotonically increasing and $\phi : \mathbb{R} \rightarrow [0,1]$ ($\lim_{s\rightarrow-\infty}\phi(s) = 0$ and $\lim_{s\rightarrow\infty}\phi(s) = 1$) or $\phi : \mathbb{R} \rightarrow [-1,1]$ ($\lim_{s\rightarrow-\infty}\phi(s) = -1$ and $\lim_{s\rightarrow\infty}\phi(s) = 1$) respectively. A multi-layer perceptron with sigmoidal processing units in its hidden layer and with linear output unit computes functions in the form of

$$f_{MLP}(\mathbf{x}, \mathbf{w}^{(1)}, \mathbf{w}^{(2)}) = \sum_{j=1}^{m} \mathbf{w}_j^{(2)} \phi\left(\mathbf{x}^T \mathbf{w}_j^{(1)}\right) = \sum_{j=1}^{m} \mathbf{w}_j^{(2)} \phi\left(\sum_{i=0}^{n} \mathbf{x}_i \mathbf{w}_{ij}^{(1)}\right) \tag{18.8}$$

Here $\mathbf{w}^{(1)} = [\mathbf{w}_1^{(1)}, \mathbf{w}_2^{(1)}, \ldots, \mathbf{w}_m^{(1)}]$ is the weight vector of the first computing layer (what is usually called hidden layer), where $\mathbf{w}_j^{(1)} \in \mathbb{R}^{n+1}, j = 1, 2, \ldots, m$ is the weight vector of the j^{th} hidden neuron, $x_0 = 1$, so $w_{0j}^{(1)}$ is a bias, and $\mathbf{w}^{(2)} = [w_1^{(2)}, w_2^{(2)}, \ldots, w_m^{(2)}]$ is the weight vector of the linear output layer.

For RBF networks radial basis functions $\phi : \mathbb{R} \rightarrow \mathbb{R}_+$ are used in the processing units, where $\mathbb{R}_+$ denotes the set of positive real numbers. The basis functions compute $\varphi(\mathbf{x}) = \phi\left(\frac{\|\mathbf{x}-\mathbf{c}\|}{\sigma}\right)$, where $\mathbf{c} \in \mathbb{R}^n$ is a center vector, σ is a width parameter of the

basis function and $\|.\|$ is a norm in $\mathbb{R}^n$. The most common radial basis function is the Gaussian function $\phi(s) = e^{-\frac{1}{2}s^2}$.

The mapping of an RBF network can be given as a weighted sum of basis functions:

$$f_{RBF}(\mathbf{x}, \mathbf{c}, \mathbf{w}) = \sum_{j=1}^{m} w_j \phi\left(\frac{\|\mathbf{x} - \mathbf{c}_j\|}{\sigma}\right) \tag{18.9}$$

where $\mathbf{w} = [w_1, w_2, \ldots, w_m]$ and $w_j \in \mathbb{R}$. Both MLPs and RBF network are feedforward neural networks.

In designing a neural model several questions must be answered. Perhaps the most important practical question concerning feedforward neural networks is about their *modeling capability*. Modeling capability decides if they can be used for system modeling as general black box architectures. The answer for this practical question came from mathematics, although it is an answer for a much more general theoretical question: what functions can be approximated by feedforward networks? The main result is that a one-hidden-layer feedforward MLP with sufficient number of hidden processing elements of sigmoidal type, and a single linear output neuron is capable of approximating any continuous function f: $\mathbb{R}^n \to \mathbb{R}$ to any desired accuracy.

There are several slightly different mathematical results formulating this *universal approximation* capability, the most important of which were developed by Hornik et al. [16], Cybenko [17], Funahashi [18], Leshno et al. [19], Blum and Li [20], etc. These results use different ways of proof. There are certain differences between the function classes into the functions to be approximated by a neural network belong, and there are different conditions for the activation functions. Here only the result of Cybenko is cited:

Let ϕ be any continuous sigmoid-type function, then given any continuous real-valued function f on $[0,1]^n$ or any other compact subset of $\mathbb{R}^n$ and $\epsilon > 0$, there exist vectors $\mathbf{w}^{(1)}$ and $\mathbf{w}^{(2)}$, and a parameterized function $f_{MLP}(\mathbf{x}, \mathbf{w}^{(1)}, \mathbf{w}^{(2)}) : [0,1]^n \to \mathbb{R}$ such that

$$\left|f(x) - f_{MLP}(\mathbf{x}, \mathbf{w}^{(1)}, \mathbf{w}^{(2)})\right| < \epsilon \text{ for all } \mathbf{x} \in [0,1]^n \tag{18.10}$$

where $f_{MLP}(\mathbf{x}, \mathbf{w}^{(1)}, \mathbf{w}^{(2)})$ is given by (18.8).

This and other similar theorems are existence theorems; they do not specify the number of hidden processing units and these results cannot be used for constructing a neural network for a given task. These theorems "only" prove that for a given task of system modeling, where the mapping of the system is continuous, one can find an MLP network. For RBF networks there are similar results. See e.g. Park and Sandberg [21], where the properties of the basis functions are exactly defined. The theoretical results of approximation capability are of fundamental importance to any applications. Although they cannot be applied directly in practice, the lack of these results would mean that neural networks cannot be used for approximating a large class of functions, they cannot be applied for modeling.

The next practical question is about the *size of a feedforward network*: how many hidden layers and how many neurons in the hidden layers are required for a given modeling task? The answer – at least partly – came again from mathematics. The mathematical results – also in a more general form – are related to the *complexity* of the network. Concerning the number of layers, the results of approximation capability

prove, that a single hidden layer is enough. However, some of the results concerning the network complexity apply networks with two hidden layers. Moreover – in practice too – there may be cases when simpler solution can be obtained if two-hidden-layer networks are used; an acceptable solution may be reached more easily or the overall complexity of the network may be smaller in spite of the two hidden layers.

For the number of hidden neurons the results are not so definite. The theoretical results derive certain relations between the complexity of a feedforward neural network and a bound for the error of approximation. For defining the approximation error some measure of discrepancy or cost function (also called loss function) must be defined. Using a cost function a risk functional can be defined as a measure of the model's fitness. If quadratic cost function is applied the risk functional will be:

$$R(\Theta) = \int (y - f_M(\mathbf{x}, \Theta))^2 p(\mathbf{x}, y) d\mathbf{x} dy, \tag{18.11}$$

where $p(\mathbf{x}, y)$ is the probability density function and Θ is the parameter (weight) vector of the network.

The complexity problem can be formulated in two ways: fixing the complexity (the number of hidden units), the question is the approximation error for a given class of functions; while if the allowed approximation error is fixed, the question is the number of hidden units required (or at least a bound on the number of hidden units). Barron [22] has an answer for the first question: he proves that for an MLP with one hidden layer of m sigmoidal neurons, the integrated squared error of approximation, integrating on a bounded subset of the n-dimensional input space is bounded by $2C^2/m$. More precisely: for every function f belonging to the class defined below, and for every sigmoidal function ϕ, every probability measure μ and every $m \geq 1$ there exists a linear combination of sigmoidal functions $f_{M,m}(\mathbf{x})$ of the form (18.8) such that

$$\int_{B_r} (f(\mathbf{x}) - f_{M,m}(\mathbf{x}))^2 \mu(d\mathbf{x}) \leq \frac{(2C)^2}{m} \tag{18.12}$$

where B_r is a compact subset of $\mathbb{R}^n$, $B_r = \{\mathbf{x} : \|\mathbf{x}\| \leq r\}$ with $r > 0$.

The theorem applies only for functions for which the Fourier transform exists, and where the first moment of the magnitude distribution of the Fourier transform C_f is bounded by C. Although the bound of the approximation error is $O(1/m)$. According to this result the order of magnitude of the number of neurons is independent of the number of input variables n. However, the input dimension can appear indirectly through the constant C_f. C_f which is a complexity measure of the function being approximated can be large and can grow exponentially with respect to n for some class of functions. Unfortunately in practical applications we have not enough information to determine C_f, so this result can be used directly neither for constructing neural networks, nor for estimating the order of magnitude of the number of hidden neurons.

Similar dimension-independent bounds for networks with more general activation functions were derived by Mhaskar and Micchelli [23]. The complexity of networks with two hidden layers were discussed for example by Kurkova [24] and Maiorov and Pinkus [25], but the complexity question is discussed by many other papers too (e.g. [26] and [27]). It should be mentioned that – from the point of view of practical applications –

these results are overly pessimistic. For example, according to the results of Kurkova [24] the required number of hidden neurons in the first hidden layer is $m_1 = nl(l+1)$, where $l \geq 2n+1$ if $n \geq 2$; and in the second one is $m_2 = l^2(l+1)^n$, where the curse of dimensionality can be seen easily (for example it would be impossible to realise a network with 10-dimension inputs, because of the required enormous number of processing units). Thus, although this result is constructive, it has only theoretical significance. On the other side an interesting aspect of the result of Kurkova, is that only the weights from the second hidden layer to the output layer depend on f, the others are fixed, which means that such a network can be trained more easily. The complexity bounds of Maiorov and Pinkus [25] are much more moderate, it needs only $3n$ and $6n+3$ hidden neurons in the first and the second hidden layer respectively. However, this is not a constructive result. The reason of the pessimistic theoretical results – at least partly – is, that they are too general, as they derive such bounds that are valid for *all* functions (including certain pathological ones) of a function class. So these theoretical results do not give satisfactory answers for the practical question, the bounds cannot be used directly in the construction of neural networks.

A further problem related to the complexity question is, that although the complexity results give bounds on the number of hidden neurons, the bounds refer to "hypothetical and ideal" networks. In the derivation of these results it is not taken into consideration that the ideal networks will only be estimated when they are trained using the observations. In practice we are interested not only in the theoretical possibilities of approximating a function with neural networks, but we have to consider, that this approximation is based on training examples, so even if a network can approximate a given function, the question is remaining: can we construct a network with prescribed accuracy using only the available limited number of observation data (which can be used for training the network)? This question is more complex, than the question of network complexity. From a practical point of view, perhaps this is the most important question as it connects modeling capability (which depends on certain complexity measure of the function to be approximated), the number of the training points P, model complexity m, the input dimension n and modeling (mean squared) error. The answer is related to the *generalization capability* of a network. The problem of generalization can be formulated in the following way: we use a neural network for approximating a function using a finite training set, however we want the trained network to give approximately good responses for such inputs, which were not used during training. The generalization capability, the model's fitness can be measured by the risk functional (18.11). However, the probability density function $p(\mathbf{x}, y)$ is unknown, we can use only the observations (training data) and instead of minimizing the risk functional the *empirical risk functional* can be minimized which is defined as the discrepancy (averaged squared error) between the system and the model outputs at the training points:

$$R_{emp}\left(\Theta \mid P\right) = \frac{1}{P}\sum_{i=1}^{P}\left(y_i - f_M(\mathbf{x}_i, \Theta)\right)^2. \tag{18.13}$$

The empirical risk, the average error at the training points, however, can be misleading. In neural network training a well-known phenomenon is overfitting, when the

network's response is getting better and better at the training points (the empirical risk is less and less), while the network loses its generalization ability, the response of the network in the points different from the training data became more erroneous. A network with too many free parameters, a too complex network is very prone to overfitting.

To obtain a proper complexity model several analytical model selection criteria have been proposed. They are based on the fact, that as the correct model complexity is not known a priori, it makes sense to postulate several models of different complexity. One intuitive approach would be to construct models of increasing complexity until the computed squared error reaches a minimum. However, as the expected error at the training points is not a good measure of the quality, it alone might not be sufficient to indicate when to terminate the search for the proper model complexity. Model complexity must be penalized to avoid using too complex model structures. Based on this approach several general analytical criteria were proposed. The most important ones are the Akaike Information Criteria (AIC) [28] and the Minimum Description Length (MDL) [29], which were developed for linear dynamic system modeling. Recently for MLPs a network information criterion (NIC) was proposed by Amari et al. [30], which was derived from AIC. The common feature of these criteria is that they have two terms: the first one depends on the approximation error for the training data (i.e. the empirical error), while the second is a penalty term. This penalty grows with the number of free parameters. Thus, if the model is too simple it will give a large value for the criterion because the residual training error is large, while a too complex model will have a large value for the criterion because the complexity term is large. The drawback of these criteria is that they need the construction of many different complexity models. Moreover, NIC is computationally rather expensive because inverses of matrices have to be computed [31].

To determine the proper model complexity, to get a network with good generalization capability, often crossvalidation is used. For crossvalidation we need a set of test data from the same problem that is not used in training, so crossvalidation computes an empirical risk too. However, the average empirical risk in this case is determined not in the training points, but in such testing points which were not used during training. The average error at the test data is a measure of the quality of the network. Although crossvalidation is a practical way of qualifying a network, it can be shown [32] that it is asymptotically equivalent to certain analytical model selection (complexity) criteria. Crossvalidation is especially useful if the number of the available data points is rather limited. In this case the leave-one-out version of crossvalidation can be used which utilizes efficiently the available data points. An important property of the leave-one-out crossvalidation is that it gives an asymptotically unbiased estimate of the expected risk. Unfortunately the computational complexity of crossvalidation can be extremely large for larger sets of data.

For obtaining a quality measure of a learning machine using the empirical risk a different answer is given by statistical learning theory (SLT). SLT has been developed by Vapnik and Chervonenkis and gives a strict theoretical basis of using empirical risk minimization (ERM) inductive principle [33]. Statistical learning theory gives necessary and sufficient conditions for consistency of the ERM inductive principle. The ERM principle is consistent if both the (unknown) true risk (expected risk) $R(\Theta^* \mid P)$

and the empirical risk $R_{emp}(\Theta^* \mid P)$ converge to the same limit $R(\Theta^\circ) = \min_\Theta R(\Theta)$ as the number of observations tends to infinity. Here Θ° denotes the true but unknown parameter vector. According to ERM principle the expected risk functional can be replaced by the empirical risk functional, and for reaching a solution with small expected risk a good strategy if we search for a solution with minimal empirical risk. Statistical learning theory deals not only with the conditions of the convergence, but also with the rate of convergence. It also deals with the question of generalization, moreover it suggest a new type of learning machine called support vector machine. In statistical learning theory the VC-dimension plays a key role. VC-dimension is basically defined for indicator functions, where the output of the function for any valid input can take only two values $y \in \{0, 1\}$, however this concept can be extended to real valued functions too. If the learning machine implements an indicator function it solves a two-class classification problem, while for general system modeling tasks the mapping of the learning machine estimates a real-valued function (a regression problem). VC-dimension of a set of indicator functions is defined as follows: a set of functions has VC dimension h if do exist h samples, but do not exist $h + 1$ samples that can be shattered by this set of functions. Shattering means, that the h samples can be separated in all 2^h possible ways using the elements of the function set. In other words VC-dimension is the maximum number of samples for which all possible binary labelings can be induced without error by a set of functions.

An important result of statistical learning theory is that the finiteness of the VC-dimension is the necessary and sufficient condition for consistency of the ERM principle. Moreover, finiteness of h implies fast convergence too. A further important result is that by the help of VC-dimension upper bounds of the true expected risk can be derived. These bounds depend on two expressions: the empirical risk functional and an expression of the VC-dimension. For binary classification case it is proved that with probability at least $1 - \eta$ the inequality for the expected risk holds true [33]:

$$R(\Theta) \leq R_{emp}(\Theta) + \frac{\nu}{2}\left(1 + \sqrt{1 + \frac{4R_{emp}(\Theta)}{\nu}}\right), \tag{18.14}$$

where

$$\nu(h) = 4\frac{h\left(\ln\frac{2P}{h} + 1\right) - \ln\eta}{P}. \tag{18.15}$$

The second term in the right hand side of (18.14) can be considered as confidence interval, which shows the "closeness" of the expected risk and the empirical risk. For real valued functions

$$R(\Theta) \leq \frac{R_{emp}(\Theta)}{1 - c\sqrt{\nu(h)}}, \tag{18.16}$$

where $\nu(h)$ is given in (18.15), and we can assume that – for practical regression problems – $c = 1$.

From these bounds it can be seen, that for small expected risk it is not enough if the empirical risk is small. We have to minimize simultaneously both the empirical risk and the VC-dimension. The solution – which can be obtained as a tradeoff between the quality of the approximation and the complexity of the approximating function – can be obtained following the structural risk minimization (SRM) principle. The SRM

principle provides a formal mechanism for choosing an optimal model complexity for a finite number of sample points [34]. Structural risk minimization has been originally suggested for classification problems, however it can be applied to any learning problems where the expected risk functional has to be minimized. In the implementation of SRM principle support vector machines have an important role.

18.4 Support Vector Machines

Support Vector Machines have been developed recently [34]. Originally it was worked out for linear two-class classification with margin, where margin means the minimal distance from the separating hyperplane to the closest data points. SVM learning machine seeks for an optimal separating hyperplane, where the margin is maximal. An important and unique feature of this approach is that the solution is based only on those data points, which are at the margin. These points are called support vectors. The linear SVM can be extended to nonlinear one when first the problem is transformed into a feature space using a set of nonlinear basis functions. In the feature space - which can be very high dimensional - the data points can be separated linearly. An important advantage of the SVM is that it is not necessary to implement this transformation and to determine the separating hyperplane in the possibly very-high dimensional feature space, instead a kernel representation can be used, where the solution is written as a weighted sum of the values of certain kernel function evaluated at the support vectors.

SVM was extended to handle regression tasks, so support vector machines can be used for function estimation. As in modeling this is the most common task, we will give a very short overview of support vector regression. Support vector regression is based again on a training data set $\{\mathbf{x}_i, y_i\}_{i=1}^{P}$, where $\mathbf{x}_i \in \mathbb{R}^n$ represents an n-dimensional input vector and $y_i \in \mathbb{R}$ is the corresponding scalar target output. In the classical Vapnik's support vector regression machine ϵ-insensitive loss function (L_ϵ) is used.

$$L_\epsilon(y) = \begin{cases} 0 & \text{for } |f(\mathbf{x}) - y| < \epsilon \\ |f(\mathbf{x}) - y| - \epsilon & \text{otherwise} \end{cases} \tag{18.17}$$

In this case approximation errors smaller than ϵ are ignored, while the larger ones are punished in a linear way.

Our goal again is to approximate a $y^* = f(\mathbf{x}^*)$ function, which represents the dependence of the output y^* on the input $\mathbf{x}^*$ of a system. The input vectors are projected into a higher dimensional feature space, using a set of nonlinear basis functions $\varphi(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}^m$. The dimensionality ($m$) of the new feature space is not defined, it follows from the method (it can even be infinite dimensional). The function is estimated by projecting the input data to a higher dimensional feature space as follows:

$$y = \sum_{j=0}^{m} w_j \varphi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) \ , \ \mathbf{w} = [w_0, w_1, \ldots, w_m]^T \ , \ \boldsymbol{\varphi} = [\varphi_0(\mathbf{x}), \varphi_1(\mathbf{x}), \ldots \varphi_m(\mathbf{x})]^T \tag{18.18}$$

The $\varphi_0(\mathbf{x})$ basis function is assumed to be 1, therefore w_0 represents a bias term b. The solution is obtained by minimizing the *empirical risk functional,* using the

ϵ-insensitive loss function:

$$R_{emp}[f] = \frac{1}{P}\sum_{i=1}^{P} L_\epsilon\left(f\left(\mathbf{x}_i, y_i\right)\right) \tag{18.19}$$

with the constraint of $\|\mathbf{w}\|^2 \leq c_0$ to keep $\|\mathbf{w}\|$ as small as possible (c_0 is a positive constant). To deal with training points outside the ϵ boundary, the $\{\xi_i\}_{i=1}^{P}$ and $\{\xi_i'\}_{i=1}^{P}$ slack variables are introduced:

$$\begin{aligned} y_i - \mathbf{w}^T\boldsymbol{\varphi}\left(\mathbf{x}_i\right) &\leq \epsilon + \xi_i, \\ \mathbf{w}^T\boldsymbol{\varphi}\left(\mathbf{x}_i\right) - y_i &\leq \epsilon + \xi_i', \\ \xi_i &\geq 0, \\ \xi_i' &\geq 0, \end{aligned} \tag{18.20}$$

with $i = 1, ..., P$. The slack variables are to describe the penalty for the training points lying outside the ϵ boundary. The measure of this cost is determined by the loss function. The solution can be obtained by minimizing the cost function:

$$F(\mathbf{w}, \xi, \xi') = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\left(\sum_{i-1}^{P}(\xi_i + \xi_i')\right), \tag{18.21}$$

subject to the constraints

$$\begin{aligned} y_i - \mathbf{w}^T\boldsymbol{\varphi}\left(\mathbf{x}_i\right) &\leq \epsilon + \xi_i,\ \xi_i \geq 0 \\ \mathbf{w}^T\boldsymbol{\varphi}\left(\mathbf{x}_i\right) - y_i &\leq \epsilon + \xi_i',\ \xi_i' \geq 0, \end{aligned} \tag{18.22}$$

with $i = 1, ..., P$. The first term in (18.21) stands for the minimization of $\|\mathbf{w}\|$, while the C constant is the trade-off parameter between this and the minimization of training data errors. This constrained optimization can be defined as a Lagrangian function:

$$\begin{aligned} J(\mathbf{w}, \sigma, \sigma', \alpha, \alpha', \gamma, \gamma') = C\sum_{i=1}^{P}(\sigma_i + \sigma_i') + \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{P}\alpha_i\left[\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i) - y_i + \epsilon + \xi_i\right] - \\ - \sum_{i=1}^{P}\alpha_i'\left[\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i) - y_i + \epsilon + \xi_i'\right] - \sum_{i=1}^{P}(\gamma_i\xi_i + \gamma_i'\xi_i') \end{aligned} \tag{18.23}$$

which can be solved more easily in its dual form. The primal problem deals with a convex cost function and linear constraints, therefore from this constrained optimisation problem a dual problem can be constructed. To do this the Karush–Kuhn–Tucker (KKT) conditions [35] are used. In the dual problem only the Lagrange multipliers the values of α and α' are unknown:

$$Q(\alpha, \alpha') = \sum_{i=1}^{P} y_i(\alpha_i + \alpha_i') - \epsilon\sum_{i=1}^{P}(\alpha_i + \alpha_i') - \frac{1}{2}\sum_{i=1}^{P}\sum_{j=1}^{P}(\alpha_i - \alpha_i')(\alpha_j - \alpha_j')K(\mathbf{x}_i, \mathbf{x}_j) \tag{18.24}$$

with constraints:

$$\sum_{i=1}^{P}(\alpha_i + \alpha_i') = 0,\ 0 \leq \alpha_i \leq C,\ 0 \leq \alpha_i' \leq C,\ i = 1, 2, \ldots, P \tag{18.25}$$

The dual problem can be solved using quadratic programming (QP), which results in the α_i and α_i' Lagrange multipliers. The bias term b follows from the KKT conditions [33]. Finally the response of the network for any input can be calculated as:

$$y = \sum_{i=1}^{P} (\alpha_i - \alpha_i') K(\mathbf{x}, \mathbf{x}_i) + b \tag{18.26}$$

where $K(\mathbf{x}, \mathbf{x}_i) = \boldsymbol{\varphi}^T(\mathbf{x})\boldsymbol{\varphi}(\mathbf{x}_i)$. It can be seen that for obtaining the response of the network only those training data are to be used, where $(\alpha_i - \alpha') \neq 0$. The corresponding indexes mark the input data points as support vectors, so the response of the learning machine depends only on the support vectors instead of depending on all training samples.

Besides its theoretical significance – from a practical point of view – support vector machines have some useful properties. Having selected the kernel function, the support vector approach solves "automatically" the model set selection (model size) problem; the solution is unique, as it is a solution of a convex optimisation problem. SVM is based on SRM, so the bounds of the expected risk give some information about its generalization capability. The construction of an SVM does not need an iterative training process, etc. The selection of kernel function is an important step of the construction of a support vector machine. The kernel function must fulfill certain conditions; for SVMs kernels satisfying the conditions of Mercer's theorem can be applied [34]. Most often Gaussian kernels are used, when the resulted SVM corresponds to an RBF network with Gaussian radial basis functions. As the SVM approach "automatically" solves the network complexity problem, the size of the hidden layer is obtained as the result of the QP procedure. Hidden neurons and support vectors correspond to each other, so the center problems of the RBF network is also solved, as the support vectors serve as the basis function centers.

Besides Gaussian kernels other kernels functions satisfying the Mercer conditions are applied in practice [35], however the selection or construction of kernels for given tasks – a new field within the theory of support vector machines – opens new possibilities [36].

Besides the advantages of SVMs – from a practical point of view – they have some drawbacks. An important practical question that is not entirely solved, is the selection of the kernel function parameters – for Gaussian kernels the width parameter σ – and the value of ϵ in the ϵ-insensitive loss function. As the kernel parameters have effect on the VC-dimension of the SVM, the generalization capability of the learning machine depends on the selection of these free parameters. Unfortunately the VC-dimension-based upper bounds are usually too pessimistic. An investigation about the practical application of VC-dimension based bounds shows, that the bound derived this way may be too conservative; up to an order of magnitude in SV classification [37]. Similar results are presented in [38] with the conclusion that these bounds provide an approximate, probably pessimistic guide to expected generalization error, and appear to be applicable only in certain circumstances as an initial aid to design. On the other hand, it was shown that VC-dimension helps to find the optimal value of the free kernel parameters [39]. A further problem with the VC-dimension based learning regression machine construction is that the VC-dimension may be infinite [40]. Recently several variants of VC-dimension (like fat-shattering dimension, covering number, entropy

number) were proposed for real valued functions, which may help to obtain significantly improved, tighter bounds [40], [41].

For an SVM the value of ϵ in the ϵ-insensitive loss function should also be selected. ϵ has an effect on the smoothness of the SVM's response and it affects the number of support vectors, so both the complexity and the generalization capability of the network depend on its value. There is also some connection between observation noise in the training data and the value of ϵ. Fixing the parameter ϵ can be useful if the desired accuracy of the approximation can be specified in advance. In some cases, however, we want the solution to be as accurate as possible. For this case an automatic way was proposed [42] to minimize ϵ, where the value of ϵ is built into the optimisation task. However, from a practical point of view perhaps the most serious problem with SVMs is the high algorithmic complexity and extensive memory requirements of the required quadratic programming in large-scale tasks. To overcome these problems, several modifications of the method have been proposed.

These algorithms are mostly iterative methods that decompose the large problem into smaller optimisation tasks [43]–[49]. These methods are commonly known as "chunking" algorithms, where the methods mainly differ in the way they determine the decomposed sub–problems. The traditional "chunking" may not reduce the problem enough, therefore different modifications have been proposed. The two main techniques are Osuna's algorithm and SMO [50]. Osuna et al. suggest maximizing the reduced QP sub-problems of a fixed size. The Sequential Minimal Optimisation (SMO) brakes up the large quadratic problem into a series of smallest possible QP problems, which are solved analytically [45]. These consist of only two Lagrange multipliers, which are jointly optimised at every iteration. Successive overrelaxation (SOR) has also been applied to large SVM problems [51].

A different solution for reducing the computational complexity is the use of the LS–SVM. The LS–SVM solves this problem by replacing the quadratic programming with a simple matrix inversion. The basic idea is similar to that of the Vapnik's SVMs. The main differences are that the inequality constraints of (21) are replaced by an equality one, and instead of using the ϵ-insensitive loss function quadratic loss function is applied which corresponds to a form of ridge regression [52], [53]. The optimisation problem and the constraint of LS-SVM are as follows:

$$\min_{\mathbf{w},b,e} F(\mathbf{w},e) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\frac{1}{2}\left(\sum_{i=1}^{P} e_i^2\right) \quad s.t.\ y_i = \mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i) + b + e_i, \tag{18.27}$$

for $i = 1, ..., P$ where e_i are error variables and b is a bias. From this equation, along with the conditions for optimality, one can construct the Lagrangian, which leads to the following solution:

$$\begin{bmatrix} 0 & \tilde{\mathbf{1}}^T \\ \tilde{\mathbf{1}} & \boldsymbol{\Omega} + C^{-1}\mathbf{I} \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix}, \mathbf{y}^T = [y_0, y_1, ..., y_P], \tilde{\mathbf{1}}^T = [1, \ldots, 1], \boldsymbol{\alpha}^T = [\alpha_0, \alpha_1, \ldots, \alpha_P], \tag{18.28}$$

where $\Omega_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j)$. The response of LS-SVM can be obtained again as a weighted sum of values of the kernel function:

$$f(x) = \sum_{i=1}^{P} \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b \tag{18.29}$$

One of the main drawbacks of the least-squares solution is that it is not sparse, meaning that it incorporates all training vectors in the resulting network. This loss of sparseness is very important, especially in the light of the equivalence between SVMs and sparse approximation [54], [55]. Practically this means that the corresponding net consists of – in its hidden layer – as many neurons, as the numbers of training vectors are used. In most real life situations the resulting networks are unnecessarily large.

Recently for eliminating this drawback different complexity reducing methods have been developed. One of them is a pruning method, which eliminates some training samples based on the sorted support vector spectrum [53], [56]. An other technique is to reduce the $\Omega + C^{-1}\mathbf{I}$ matrix finding a "basis" (the quote indicates, that this basis is only true under certain conditions) of this matrix [57]. This method is used for bringing the matrix to the reduced row echelon form [58].

18.5 The Nature of Knowledge, Prior Information

In black box modeling the primary knowledge that can be used for model building is a set of input-output data. So the first task of modeling is to build a proper database. One serious problem in real-world tasks is that in many cases the number of available data is limited and rather small. There are further difficulties: the data are noisy, we have to deal with the problem of missing data, and it is not a rare situation, that we have to work with distorted or false data [59].

The deficiency of the database can be compensated if prior information or additional knowledge about the system is available. In general, in model building it is important to use all information available at all. This means that the learning machine must be able to utilise different representations of knowledge: measurement data, non-numerical observations, symbolic knowledge (e.g. inference rules), etc.

Prior or additional information can be utilised in different ways. One way is to use prior knowledge to create virtual data. With virtual examples the database can be extended, and the larger database can be used in the construction of learning machine. There are some good examples of the application of this approach. One well-known example is ALVINN, the autonomous vehicle navigation system, which was trained using camera pictures acquired when a human driver was steering the car [2]. These acquired pictures, however, represent only the right way of navigation, as a human driver is good at keeping the vehicle in the lane. To get further training examples that represent non-standard situations, new "camera images" were generated using image transformations. The extended database contained many images of unusual situations, making it possible to achieve much better performance of the trained system. Similarly, some legal transformations of images are used to generate virtual examples in a face recognition problem [60].

There are cases, however, where it is better if the prior information can be built directly into the learning machine. This is the case when symbolic knowledge is available in the form of *if-then-else* rules. One efficient way of utilising symbolic knowledge is, if it is used to build the initial structure (and to determine the initial values of the parameters) of the learning machine, and this initial network is trained further using the observed training data. The explanation-based learning approach [61] or the

Knowledge-Based-Artificial Neural Network (KBANN) [62] and its variants are good examples of using this approach. For the application of this approach such learning machine can be used where its performance can be improved step-by-step. Incremental learning capability is also a feature that helps the application of this approach.

Incremental learning means that new knowledge (new examples) can be trained without forgetting the previous knowledge or when the new knowledge only modifies the behaviour of the machine. This capability is also important when the adaptation of the learning machine to the changing environment is required. In this respect learning machines that can be trained iteratively are much more appropriate than machines that use batch training. So MLP and especially the networks with local learning and generalization like RBF and CMAC (networks with local basis functions) are more suitable for incremental learning so far than SVM, even if recently a recursive version of the SVM learning was proposed making incremental and decremental learning possible [63].

18.6 Questions Concerning Implementation

Until now we have discussed many questions, which have both theoretical and practical importance. Although the main emphasis throughout this chapter is on the questions of practical importance, most of these questions can only be answered using theoretical results; these questions can be regarded as theoretical ones, which have direct connection to practice. However, in addition to these questions there are further ones, which are entirely related to the practical applications. These are the special questions of implementation.

In measurement systems these questions may be essentially important, as the answers tell us whether or not a learning machine can be applied for a given application. In some applications the constraints on the technological parameters can be so strict, that all other aspects are of secondary rank.

In a measurement system there is often a need for real-time operation, when the solution must fulfill certain time constraints both in the operation (recall) and in the training phase. The latter is especially important if the learning machine is applied in a changing environment, where the solution must follow these changes in real-time. For applications, where real-time adaptation is required, batch learning cannot be used, the requirement can be satisfied only if on-line recursive learning is possible. The algorithmic complexity of the learning process and the lack or the possibility of incremental learning are also important features from this respect. The time constraints are important in applications like real time control [64], detection of phenomena of very short duration [65], or real-time recognition of patterns [66], etc.

In measurement systems, especially in such applications where complex sensors systems are used [67], [68], besides high-speed operation low cost, small size solutions are required, where further constraints like limited power consumption must also be satisfied. In such cases only dedicated *embedded hardware* solutions can be applied, where the inherently parallel architecture of the classical neural networks must also be utilised.

For constructing a hardware neural network certain technological possibilities and

limitations must be considered. A hardware neural solution may be obtained using analog or digital circuits (there are also a few optical implementations [69], however, they cannot be regarded as common solutions). Both approaches have strengths and weaknesses. Analog circuits are orders of magnitudes smaller than digital ones, their operating speed may be much higher, however, their low noise immunity, low stability in time, and the fact that both multipliers and nonlinear activation functions can be implemented only approximately, may be an obstacle of their application. Using analog electronic technology on-line training, modification of the weight values cannot be easily implemented. On the contrary, digital circuits have the advantage of implementing precise, highly stable computational elements with easily adjustable parameters (e.g. weights), but they are not so good in operating speed, they require larger chip area (in a VLSI implementation). There are also mixed analog-digital solutions, which attempt to utilise the advantages of both technologies [70].

The technological problems of a digital implementation depend largely on the operating mode of a neural network. There are applications where there is no need for on-line training, pretrained networks can be used (e.g. some important recognition task). In these cases the main task is to implement a networks with high speed operation (recall), the training of the network is done off-line, usually using an other system and only the network with fixed architecture and weights is to be realised. For networks where high-speed real-time adaptive training is required, such learning machines and technologies can be used, where the training algorithm can also be implemented satisfying the complexity and time constraints.

In this respect depending on the network architecture and the learning algorithm great differences can be found. The classical MLP with its simple backpropagation learning algorithm uses rather simple training steps, however the number of training iterations can be very large. More complex training algorithms, like quickprop [71] or Levenberg-Marquardt [15] algorithms are not so easy to implement in hardware form. The training of support vector machines – the quadratic programming – is also a computational intensive task, so SVMs are not such good candidates for hardware realisation. It is true even if we take into consideration the more efficient versions of quadratic programming mentioned previously.

In digital implementations the most demanding elements of a classical neural architecture, like MLP or RBF are the multipliers and the nonlinear functions (activation functions or basis functions). In this respect the kernel-based support vector machines are rather similar (if we do not deal with the problem of QP). For getting efficient hardware solutions these architectural elements must be realised efficiently. An efficient multiplier architecture was proposed in [72], which can be used not only for implementing the matrix-vector multiplications required in great number in a parallel hardware architecture, but it can be used to implement efficiently the sigmoidal activation functions using B-spline approximation [73]. This solution applies a serial/parallel approach and bit-level optimization.

From the point of view of digital hardware realization the best is if the architecture does not need multipliers and nonlinear activation functions at all. In this respect one of the most promising neural network architecture is cerebellar model articulation controller (CMAC). CMAC is an associative memory type network, without multipliers and nonlinear activation functions [74], [75]. A further advantage of CMAC against

MLP, RBF and others is its extremely fast learning [76], which is a consequence of its architecture, its linear output layer and the fact that an optimal arrangement of training points can be found where only one iteration of training is enough. The price of its simple architecture is its moderate modeling and generalization capability [77], [78]. However, using a regularized version of its training algorithm the generalization error can be reduced significantly [79]. It can also be shown that binary CMAC and SVM with linear B-spline kernels are equivalent [80].

In hardware implementation – using either analog or digital techniques – a further problem must be considered, the problem of finite precision. The effects of limited precision of the weight values, the limited accuracy of multipliers and the nonlinear functions (activation functions or basis functions) and the finite precision of arithmetics have to be analysed in hardware implementations. Using analog circuits the equivalent word-length and the precision of arithmetics cannot be increased arbitrarily because of technological reasons. In digital implementations the precision is limited only by the computation time and/or the hardware complexity. In this respect the solution proposed in [72] has an advantage as the precision can be increased arbitrarily of course at the expense of operating speed. So according to the trade-off between speed and computational precision the designer can decide about the hardware construction. It should be mentioned, that CMAC has also advantage in this respect, as it needs only to store the weight values, where increasing the word length needs only a larger memory, moreover the computation required by CMAC is the addition of some weight values where the precision can be increased rather easily without serious hardware consequence.

18.7 Conclusions

The purpose of this chapter was to give an overview about some important practical questions we encounter when neural networks are used in measurement (modeling) tasks. Although these questions are related to practice, answers can be obtained only from theory. The theoretical results are of primary importance: as it is cited in Vapnik's book [34] "nothing is more practical than a good theory". However – at least in their present state – several of these results are far from being applicable in practice. The theoretical questions – and even more the answers – are too general, so usually they give too pessimistic results, which form obstacles to their direct applications in the construction of different neural solutions. However, the questions – and in some cases the answers - of new theoretical approaches are getting more and more useful for practice. In this respect statistical learning theory and structural risk minimization principle give entirely new possibilities. The latest achievements reached in the SLT-based kernel methods help to narrow the gap between theory and practice. However, the most critical problems with constraints of implementation still pose important challenges for the future.

Bibliography

[1] T.J. Sejnowski, and C. Rosenberg, Parallel Networks that Learn to Pronounce English Text, *Journal of Artificial Intelligence Research* **1** (1987) 145–168.

[2] D.A. Pomerleau, Neural Network Perception for Mobile Robot Guidance, Kluwer, Dordrecht, The Netherlands (1993).

[3] Y. LeCun, B. Boser and J.S. Denker, Backpropagation Applied to Handwritten Zip Code Recognition, *Neural Computation* **1** (1989) 541–551.

[4] S. Cho, Y. Cho and S. Yoon, Reliable roll force prediction in cold mill using multiple neural networks, *IEEE Trans. on Neural Networks* **8**(4) (1997) 874–882.

[5] G. Bloch, F. Sirou, V. Eustache, P. Fatrez, Neural Intelligent Control for a Steel Plant, *IEEE Trans. on Neural Networks* **8**(4) (1997) 910–918.

[6] P.J. Edwards, A.F. Murray, G. Papadopoulos, A.R. Wallace, J. Barnard, G. Smith, The Application of Neural Networks to the Papermaking Industry, *IEEE Trans. on Neural Networks* **10**(6) (1999) 1456–1464.

[7] P. Berényi, G. Horváth, B. Pataki, G. Strausz, Hybrid-Neural Modeling of a Complex Industrial Process, *Proc. IEEE Instrumentation and Measurement Technology Conference,* Budapest, Hungary, May 21–23, Vol. 3 (2001) 1424–1429.

[8] L. Ljung, Black-box Models from Input-Output Measurements, *Proc. IEEE Instrumentation and Measurement Technology Conference,* Budapest, Hungary, May 21–23, Vol. 1 (2001) 138–146.

[9] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P.Y. Glorennec, H. Hjalmarsson, A. Juditsky, Non-linear Black-box Modeling in System Identification: A Unified Overview, *Automatica* **31** (1995) 1691–1724.

[10] L. Ljung, System Identification - Theory for the User, Prentice-Hall, N.J. 2nd edition (1999).

[11] J. Schoukens, R. Pintelon, System Identification: a Frequency Domain Approach, IEEE Press, New York (2001).

[12] P. Eykhoff, System Identification, Parameter and State Estimation, Wiley, New York (1974).

[13] M. Deistler, Linear Dynamic Errors-in-Variables Models, *Journal of Applied Probability* **23** (1986) 23–39.

[14] J. Van Gorp, J. Schoukens, R. Pintelon, Learning Neural Networks with Noisy Inputs Using the Errors-in-Variables Approach, *IEEE Trans. on Neural Networks* 11(2) (2000) 402–414.

[15] S. Haykin, Neural Networks. A Comprehensive Foundation, Second Edition, Prentice Hall, N. J. (1999).

[16] K. Hornik, M. Stinchcombe, H. White, Multilayer Feedforward Networks are Universal Approximators, *Neural Networks* **2** (1989) 359–366.

[17] G. Cybenko, Approximation by Superpositions of a Sigmoidal Function, *Mathematics of Control, Signals and Systems* **3** (1989) 303–314.

[18] K.I. Funahashi, On the Approximate Realization of Continuous Mappings by Neural Networks, *Neural Networks* **2**(3) (1989) 183–192.

[19] M. Leshno, V.Y. Lin, A. Pinkus, S. Schocken, Multilayer Feed-forward Networks with a Nonpolynomial Activation Function can Approximate any Function, *Neural Networks* **6** (1993) 861–67.

[20] E.K. Blum, L.K. Li, Approximation Theory and Feedforward Networks, *Neural Networks* **4** (1991) 511–515.

[21] J. Park, I.W. Sandberg, Approximation and Radial-Basis-Function Networks, *Neural Computation* **5**(2) (1993) 305–316.

[22] A.R. Barron, Universal Approximation Bounds for Superposition of Sigmoidal Functions, *IEEE Trans. on Information Theory* **39**(3) (1993) 930–945.

[23] N.H. Mhaskar, C.A. Micchelli, Dimension Independent Bounds on the Degree of Approximation by Neural Networks, *IBM Journal of Research and Development* **38** (1994) 277–284.

[24] V. Kurková, Kolmogorov's Theorem and Multilayer Neural Networks, *Neural Networks* **5** (1992) 501–506.

[25] V. Maiorov, A. Pinkus, Lower Bounds for Approximation by MLP Neural Networks, *Neurocomputing* **25** (1999) 81–91.

[26] V. Kurková, P.C. Kainen, V. Kreinovich, Estimates of the number of hidden units and variation with respect to half-spaces, *Neural Networks* **10** (1997) 1061–1068.

[27] F. Carselli, A.C. Tsoi, Universal approximation using Feedforward Neural Networks: a Survey of some Existing Methods and Some New Results, *Neural Networks* **11**(1) (1998) 15–37.

[28] H. Akaike, Information Theory and an Extension of the Maximum Likelihood Principle, Second International Symposium on Information Theory, Akadémiai Kiadó, Budapest, (1972) 267–281.

[29] J. Rissanen, Modeling by Shortest Data Description, *Automatica* **14** (1978) 465–471.

[30] N. Murata, S. Yoshizawa, S.-I. Amari, Network Information Criterion – Determining the Number of Hidden Units for an Artificial Neural Network Model, *IEEE Trans. Neural Networks* 5(6) (1994) 865–871.

[31] G. te Brake, J.N. Kok, P.M.B. Vitányi, Model Selection for Neural Networks: Comparing MDL and NIC, NeuroCOLT, Neural Networks and Computational Learning Theory, Technical Report Series, NC-TR-95-021 Royal Holloway University of London (1995).

[32] B.D. Ripley, Pattern Recognition and Neural Networks, Cambridge University Press, Cambridge (1996).

[33] V.N. Vapnik, Statistical Learning Theory, Wiley, New York (1998).

[34] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer, New York (1995).

[35] V. Vapnik, S. Golowich, A. Smola, Support Vector Method for Function Approximation, Regression Estimation and Signal Processing, In Mozer, M., Jordan, M. and Petsche, T. (Eds.) Advances in Neural Information Processing Systems 9. Cambridge, Mass: MIT Press (1997) 281–287.

[36] B. Schölkopf, A. Smola, Learning with Kernels. Support Vector Machines, Regularization, Optimization and Beyond MIT Press, Cambridge, MA (2002).

[37] C.J.C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, *Knowledge Discovery and Data Mining* **2**(2) (1998) 121–167.

[38] S.B. Holden, M. Niranjan, On the Practical Applicability of VC Dimension Bounds, *Neural Computation* **7** (1995) 1265–1288.

[39] O. Chapelle, V. Vapnik, Model Selection for Support Vector Machines, S. Solla, T. Leen, K. R. Muller (Eds.), Advances in Neural Information Processing Systems Vol. 12, MIT Press, Cambridge, MA (2000) 230–237.

[40] A.J. Smola, B. Schölkopf, A Tutorial on Support Vector Regression, NeuroCOLT, Neural Networks and Computational Learning Theory, Technical Report Series, NC2-TR-1998-030 (1998).

[41] P.L. Bartlett, P. Long., R. Williamson, Fat-shattering and Learnability of Real-valued Functions, *Journal of Computer and System Sciences* **52**(3) (1996) 434–452.

[42] B. Schölkopf, P. Bartlett, A. Smola, R. Williamson, Support Vector Regression with Automatic Accuracy Control, in Kearns, M. S., Solla, S. A., and Cohn, D. A. (Eds.), Advances in Neural Information Processing Systems, Vol. 11, MIT Press, Cambridge, MA (1999) 330–336.

[43] C.J.C. Burges, B. Schölkopf, Improving the Accuracy and Speed of Support Vector Learning Machines, In M. Mozer, M. Jordan, and T. Petsche, (Eds.) Advances in Neural Information Processing Systems 9, Cambridge, MA, 1997, MIT Press (1997) 375–381.

[44] E. Osuna, R. Freund, F. Girosi, An Improved Training Algorithm for Support Vector Machines, In J. Principe, L. Giles, N. Morgan, and E. Wilson, (Eds.) Neural Networks for Signal Processing VII Proceedings of the 1997 IEEE Workshop, New York (1997) 276–285.

[45] J. Platt, Sequential Minimal Optimization: Fast Algorithm for Training Support Vector Machines, Microsoft Research Technical Report MSR–TR–98–14, April 21 (1998).

[46] P. Laskov, An Improved Decomposition Algorithm for Regression Support Vector Machines, In S.A. Solla, T.K. Leen, and K.R. Müller, (Eds.) Advances in Neural Information Processing Systems 12, MIT Press (2000) 484–490.

[47] S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, K.R.K. Murthy, Improvements to Platt's SMO Algorithm for SVM Classifier Design, Technical report, Dept of CSA, IISc, Bangalore, India (1999).

[48] T. Joachims. Making Large-Scale SVM Learning Practical, Advances in Kernel Methods – Support Vector Learning', MIT Press, Cambridge, USA (1998).

[49] M. Moser, M. Jordan, T. Petsche (Eds.), Improving the Accuracy and Speed of Support Vector Machines, Advances in Neural Information Processing Systems Vol. 9., MIT Press, Cambridge, MA (1997).

[50] E. Osuna, R. Freund, F. Girosi, Support Vector Machines: Training and Applications, Technical Report AIM-1602, MIT A.I. Lab. (1996).

[51] O.L. Mangasarian, D. Musicant, Successive Overrelaxation for Support Vector Machines, *IEEE Trans. on Neural Networks* **10** (1999) 1032–1037.

[52] J.A.K. Suykens, J. Vandewalle, Least Squares Support Vector Machine Classifiers, *Neural Processing Letters* **9**(3) (1999) 293–300.

[53] J.A.K. Suykens, L. Lukas, J. Vandewalle, Sparse Approximation Using Least Squares Support Vector Machines, *Proc. IEEE International Symposium on Circuits and Systems ISCAS 2000*, Vol. 2 (2000) 757–760.

[54] J.A.K. Suykens, Nonlinear modeling and Support Vector Machines, *Proc. IEEE Instrumentation and Measurement Technology Conference*, Budapest, Hungary, May 21–23, Vol. 1 (2001) 287–294.

[55] F. Girosi, An Equivalence Between Sparse Approximation and Support Vector Machines, *Neural Computation* **10**(6) (1998) 1455–1480.

[56] J.A.K. Suykens, J. De Brabanter, L. Lukas, J. Vandewalle, Weighted least squares support vector machines: robustness and sparse approximation, *Neurocomputing* **48** (2002) 85-105.

[57] J. Valyon, G. Horváth, Reducing the Complexity and Network Size of LS-SVM Solutions, submitted to *IEEE Trans. Neural Networks* (2002).

[58] "Numerical Recipes", Cambridge University Press, Books On-Line,
web: http://www.nr.com

[59] B. Pataki, G. Horváth, G. Strausz, Z. Talata, Inverse Neural Modeling of a Linz-Donawitz Steel Converter, *e & i Elektrotechnik und Informationstechnik* **117**(1) (2000) 13–17.

[60] P. Niyogi, F. Girosi, T. Poggio, Incorporating Prior Information in Machine Learning by Creating Virtual Examples, Technical report, MA 02139. MIT Center for Biological and Computational Learning, Cambridge, MA (1998).

[61] J.W. Shavlik, G.G. Towell, An Approach to Combining Explanation Based and Neural Learning Algorithms, *Connection Science* **1** (1989) 233–255.

[62] G.G. Towell, J.W. Shavlik, Knowledge-Based Artificial Neural Networks, *Artificial Intelligence* **70** (1994) 119–165.

[63] G. Cauwenberghs, T. Poggio, Incremental and Decremental Support Vector Machine Learning, In T. Leen, T. Dietterich, V. Tresp (Eds.), Advances in Neural Information Processing Systems 13, MIT Press (2001).

[64] J. Liu, M. Brooke, Fully Parallel On-chip Learning Hardware Neural Network for Real-Time Control, *Proc. IEEE International Symposium on Circuits and Sytems,* ISCAS 99, Orlando, FL (1999).

[65] P. Masa, K. Hoen, H. Wallinga, A High-Speed Analog Neural Processor, *IEEE Micro* (1994) 40–50.

[66] S. Knerr, L. Personnaz, G. Dreyfus, Handwritten Digit Recognition by Neural Networks with Single-Layer Training, *IEEE Trans. Neural Networks* **3** (1992) 962–969.

[67] J.W.M. Van Dam, B.J.A. Kröse, F.C.A. Groen, Adaptive Sensor Models, *Proc. of International Conference on Multisensor Fusion and Integration for Intelligent Systems,* Washington D.C. (1996) 705–712.

[68] J.W.M. Van Dam, Environment modeling of Mobile Robots: Neural Learning for Sensor Fusion, Ph.D. thesis, Universiteit van Amsterdam (1998).

[69] P.E. Keller, A.F. Gmitro, An Optical Neural Network Implemented with Fixed, Planar Holographic Interconnects, *Proceedings of the Neural Network Workshop for the Hanford Community,* Pacific Northwest National Laboratory, Richland, WA, USA (1994) 93–99.

[70] A. Schmid, Y. Leblebici, D. Mlynek, A Mixed Analog-Digital Artificial Neural Network Architecture with On-Chip Learning, *IEE Proceedings – Circuits, Devices and Systems* **146**(6) (1999) 345–349.

[71] S.E. Fahlman, Fast Learning Variations on Back-Propagation: an Empirical Study, *Proc. of the 1988 Connectionist Models Summer School,* Pittsburg, Morgan Kaufmann, San Mateo, Calif (1988) 38–51.

[72] T. Szabó, L. Antoni, G. Horváth, B. Fehér, Full-Parallel Digital Implementation for Pre-Trained Neural Networks, *Proc. IEEE International Joint Conference on Neural Networks*, IJCNN 2000, Como, Italy, Vol. 3 (2000) 85–90.

[73] T. Szabó, G. Horváth, An Efficient Hardware Implementation of Feed-forward Neural Networks, *Proc. International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, IEA/AIE 2001, Budapest, Lecture Notes in Artificial Intelligence, 2070 Springer (2001) 300–313.

[74] J.S. Albus, A New Approach to Manipulator Control: the Cerebellar Model Articulation Controller (CMAC), *Truns. ASME* (1975) 220–227.

[75] M. Brown, C.J. Harris, Neurofuzzy Adaptive modeling and Control, Prentice Hall, New York (1994).

[76] D.E. Thompson, S. Kwon, Neighborhood Sequential and Random Training Techniques for CMAC, *IEEE Trans. Neural Networks* **6** (1995) 196 –202.

[77] M. Brown, C.J. Harris, P. Parks, The Interpolation Capability of the Binary CMAC, *Neural Networks* **6**(3) (1993) 429–440.

[78] T. Szabó, G. Horváth, CMAC Neural Network with Improved Generalization Capability for System modeling, *Proc. IEEE Conference on Instrumentation and Measurement*, Anchorage, AK. Vol. 2 (2002) 1603–1608.

[79] T. Szabó, G. Horváth, Improving the Generalization Capability of the Binary CMAC, *Proc. International Joint Conference on Neural Networks*, IJCNN 2000. Como, Italy, Vol. 3 (2000) 85–90.

[80] G. Horváth, CMAC Neural Network as an SVM with B-Spline Kernel Functions, submitted to *IEEE Instrumentation and Measurement Conference*, Vail, Colorado (2003).

List of participants

Peter Antal
Budapest Univ. Technology & Economics
Dept. Measurement and Information Systems
Magyar tudósok körútja 2
H-1117 Budapest, Hungary
peter.antal@esat.kuleuven.ac.be

Valentin Arkov
Ufa State Aviation Technical University
12, K. Marx St.
UGATU, ASU Dept.
450000, Ufa, Russia
arkov@asu.ugatu.ac.ru

Marta Avalos
Heudiasyc Lab. Univ. Techn. Compiègne
Research Center Royallieu
BP 60649 / 60206 Compiègne
France
avalos@hds.utc.fr

Gökhan Bakir
Instit. of Robotics & Mechatr.
German Aerospace Center DLR
Oberpfaffenhofen P.O. Box 1116
D-82230 Wessling, Germany
goekhan.bakir@dlr.de

Annalisa Barla
Univ. Di Genova, DISI
Via Dodecaneso 35
16146 Genova
Italy
barla@disi.unige.it

Peter Bartlett
Univ. of California, Dept. of Statistics
367 Evans Hill # 3860
Berkeley, CA 94720-3860
USA
peter.bartlett@anu.edu.au

Mitra Basu
City College of New York
Electrical Engineering Dept.
140th St. and Convent Av.
New York, NY 10031, USA
basu@ccny.cuny.edu

Sankar Basu
National Science Foundation
CISE/CCR Division
4201 Wilson Blvd., Room 1145
Arlington, VA 22230, USA
sabasu@nsf.gov

Ildar Batyrshin
Kazan State Techn. Univ.
Inst. Informatics ans Applied Math.
K. Marx St. 68, Kazan 420015
Russia
batyr@emntu.kcn.ru

Mikhail Belkin
Univ. of Chicago, Dept. of Mathematics
5734 S. University Ave.
Chicago, IL 60637
USA
misha@math.uchicago.edu

Anton Belousov
ICB Inst. of Chem. and Biochem.
Sensor Research, Mendelstrasse 7
48149 Münster
Germany
a.belousov@icb-online.de

Kristin Bennett
Rensselaer Polytechnic Institute
Dept. Mathematical Sciences
Troy, New York, 12180-3590
USA
bennek@rpi.edu

Christopher Bishop
Microsoft Research Ltd.
7 J.J. Thomson Avenue
Cambridge CB3 OFB
U.K.
cmbishop@microsoft.com

Peter Bosman
Utrecht University
Inst. Information and Computing Science
P.O. Box 80.089 3508 TB Utrecht
Nederland
peter.bosman@cs.uu.nl

Sergey Butakov
Altai Academy of Economy and Law
Komsomolsky av 86
Barnaul 656032
Russia
swb@agtu.secna.ru

Marco Campi
University of Brescia
Dept. Electrical Engineering
Via Branze 38, 25123 Brescia
Italy
campi@ing.unibs.it

Stéphane Canu
INSA Rouen
BP 8 Place E. Blondel
76131 Mont. St. Agnon, Cedex
France
scanu@insa-rouen.fr

Yu-Han Chang
MIT
143 Albany St. # 3
Cambridge, MA 02139
USA
ychang@mit.edu

Gal Chechik
Interdiscipl. Center for Neural Comp.
Shprintzak Bldg.
Givat Ram, Jerusalem
Israel
ggal@cs.huji.ac.il

Ilkay Colakoglu
Electrical and Electronics Eng. Dept.
Middle East Technical University
06531 Ankara
Turkey
ilkay@metu.edu.tr

Nello Cristianini
Royal Holloway, University of London
Dept. Computer Science
Egham, Surrey
TW 20 OEX, UK
nello@support-vector.net

Lehel Csato
Neural Computing Res. Group
MB306, Aston Street
B4 7ET Birmingham
UK
csatol@aston.ac.uk

Bojana Dalbelo Basic
University of Zagreb
Fac. Electr. Engineering and Computing
Unska 3, HR 10000 Zagreb
Croatia
bojana.dalbelo@fer.hr

Jos De Brabanter
K.U. Leuven
Dept. Elektrotechniek, ESAT-SCD-SISTA
Kasteelpark Arenberg 10
3001 Heverlee, Belgium
jos.debrabanter@esat.kuleuven.ac.be

Luc Devroye
McGill University
School of Computer Science
Montreal, H3A 2K6
Canada
luc@cs.mcgill.ca

Hatice Dogan
Dokuz Eylul Univ.
Eng. Fac. Electrical & Electronics Dept.
35160 Buca, Izmir
Turkey
hatice.dogan@deu.edu.tr

Alexander Dolia
Dept. 504, Radioengineering Faculty
National Aerospace University
Chkalova 17, St., Kharkov, 61070
Ukraine
lukin@xai.kharkov.ua

Pinar Duygulu
Middle East Technical University
Dept. Computer Engineering
Ankara, TR 06531
Turkey
duygulu@ceng.metu.edu.tr

Mark Embrechts
Rensselaer Polytechnic Institute
Dept. Decision Sc. and Eng. Syst.
Troy, NY 12180
USA
embrem@rpi.edu

Farida Enikeeva
Moscow State University
Fac. Mechanics & Mathematics
Dept. Probability Theory
Moscow 119992, Russia
farida@shade.msu.ru

Zeki Erdem
TUBITAK, Marmara Research Center
Information Technologies Institute
Electronic Systems Group
POB 21, Gebze/Kocaeli, Turkey
zeki@btae.mam.gov.tr

Eleazar Eskin
Columbia University
450 CS building, 500 W 120th str.
New York City, NY 10027
USA
eeskin@cs.columbia.edu

Mario Figueiredo
Instituto Superior Tecnico
Instituto de Telecomunicacioes
1049-001 Lisboa
Portugal
mtf@lx.it.pt

Emanuele Franceschi
Univ. Genova, DISI
Via Dodecaneso 35
16146 Genova
Italy
emafranc@disi.unige.it

Laurentiu Frangu
Univ. "Dunarea de Jos" of Galati
St. Domneasca 47
6200 Galati
Romania
Laurentiu.Frangu@ugal.ro

Martin Giese
Univ. Clinic Tübingen
ARL, Dept. Cognitive Neurology
Spemannstr. 34, 72076 Tübingen
Germany
martin.giese@tuebingen.mpg.de

Arnulf Graf
Max Planck Institute Biol. Cybernetics
Spemannstrasse 38
72076 Tübingen
Germany
arnulf.graf@tuebingen.mpg.de

Laszlo Gyorfi
Budapest Univ. Techn. and Economics
Dept. Comp. sc. & inform. theory
Stoczek u.2.
Hungary
gyorfi@szit.bme.hu

Sandor Györi
Budapest Univ. Techn. and Economics
Goldmann Gy ter. 3 V2.104
1111 Budapest
Hungary
gyori@szit.bine.hu

Ugur Halici
Middle East Technical University
Electrical & Electronics Eng. Dept.
06531 Ankara
Turkey
halici@metu.edu.tr

Bart Hamers
K.U. Leuven
Dept. Elektrotechniek, ESAT-SCD-SISTA
Kasteelpark Arenberg 10
3001 Heverlee, Belgium
bart.hamers@esat.kuleuven.ac.be

Gábor Horváth
Budapest Univ. of Technology & Economics
Dept. Measurement and Information Systems
Magyar tudósok körútja 2
H-1117 Budapest, Hungary
horvath@mit.bme.hu

Dimitris Iakovidis
Dept. Informatics & Telecommunication
7 Mykonou, Zokrafou
15772 Athens
Greece
diakov@di.uoa.gr

Paul Kainen
Dept. of Math.
Georgetown University
Washington, DC 20057
USA
kainen@georgetown.edu

Andras Kocsor
Research group on Artificial Intelligence
Aradi Vertanuk tere 1
6720 Szeged
Hungary
kocsor@inf.u-szeged.hu

Jacob Kogan
Univ. Maryland, Baltimore Country
1000 Hilltop Circle
Baltimare, MD 21250
USA
kogan@math.umbc.edu

Adam Krzyzak
Department of Computer Science
Concordia University
1455 De Maisonneuve Blvd. West
Montreal, Canada H3G 1M8
krzyzak@cs.concordia.ca

Rudolf Kulhavy
Honeywell Prague Laboratory
Pod vodarenskou vezi 4
182 08 Prague 8
Czech Republic
rudolf.kulhavy@honeywell.com

Vera Kurkova
Academy of Sciences of the Czech Republic
Institute of Computer Science
Pod Vodarenskou vezi 2
182 07 Prague, Czech Republic
vera@cs.cas.cz

Gert Lanckriet
University of California, Berkeley
262M Cory Hall
Berkeley, CA, 94720
USA
gert@eecs.berkeley.edu

Jörg Lemm
Universität Münster
Institut für Theoretische Physik
Wilhelm-Klemm-Str. 9
48149 Münster, Germany
joerg.lemm@wgz-bank.de

Chuan Lu
K.U. Leuven
Dept. Elektrotechniek, ESAT-SCD-SISTA
Kasteelpark Arenberg 10
3001 Heverlee, Belgium
chuan.lu@esat.kuleuven.ac.be

Andriy Lutsyk
Institute of Physics and Mechanics
NAS of Ukraine
5 Naukova St., 79601 Lviv
Urkraine
lutsyk@ah.ipm.lviv.ua

Ana Madevska Bogdanova
Institute of Informatics
Fac. Natural Sciences and Mathematics
PO Box 162. 1000 Skopje
Macedonia
ana@ii.edu.mk

Michel Maignan
University of Lausanne, Geostatistics
Banque Cantonale de Geneve
BFSH 2, 1015 Lausanne
Switzerland
michel.maignan@img.unil.ch

Alexander Malyscheff
University of Oklahoma
School Industr. Eng. 202 W. Boyd
Room # 124, Norman, OK 73019
USA
alexm@ou.edu

Florian Markowetz
Max Planck Institute Molecular Genetics
Dept. Computational Molecular Biology
Ihnestrasse 63-73, 14195 Berlin
Germany
florian.markowetz@molgen.mpg.de

Santiago Marco
Universitat de Barcelona
Departament d'Electrònica
Marti Franquès 1
08028 Barcelona, Spain
santi@el.ub.es

Charles Micchelli
State University of New York
Dept. Mathematics & Statistics
University at Albany, SUNY
Albany, NY 12222, USA
cam@math.albany.edu

Yves Moreau
K.U. Leuven
Dept. Elektrotechniek, ESAT-SCD-SISTA
Kasteelpark Arenberg 10
3001 Heverlee, Belgium
yves.moreau@esat.kuleuven.ac.be

Antoine Naud
Nicholas Copernicus Univ.
Dept. Informatics
ul. Grudziadzka 5, 87-100 Torun
Poland
naud@phys.uni.torun.pl

Zidrina Pabarskaite
SCISM
South Bank University
London SE1 0AA
UK
zipa@softhome.net

Matteo Pardo
INFM & University of Brescia
V. Valotti 9
25133 Brescia
Italy
pardo@ing.unibs.it

Kristiaan Pelckmans
K.U. Leuven
Dept. Elektrotechniek, ESAT-SCD-SISTA
Kasteelpark Arenberg 10
3001 Heverlee, Belgium
kristiaan.pelckmans@esat.kuleuven.ac.be

Marta Pinter
Budapest Univ. Techn. and Economics
Dept. Comp. Sc. & Inform. Theory
Magyar tudósok körútja 2, H-1117 Budapest
Hungary
marti@szit.bme.hu

Tomaso Poggio
Massachusetts Inst. of Technology
Dept. Brain & Cognitive Sciences, AI Lab
45 Carleton Street, Cambridge MA 02142
USA
tp@ai.mit.edu

Massimiliano Pontil
Dipartimento di Ingegneria Informatica
Università di Siena
Via Roma 56, 53100 Siena
Italy
pontil@dii.unisi.it

Maria Prandini
University of Brescia
Dept. Electronics for Automation
Via Branze 38. 25123 Brescia
Italy
prandini@ing.unibs.it

Liva Ralaivola
Lab. D'Informatique de Paris 6
LIP6, Pole IA, Univ. Paris 6
8, rue du Capitaine Scott, 75015 Paris
France
liva.ralaivola@lip6.fr

Carlos Rodriguez
Suny at Albany Dept. Math.
1400 Washington Ave.
Albany, MY 12222
USA
carlos@math.albany.edu

Roman Rosipal
NASA Ames Research Center
Mail Stop 269-3
CA 94035
USA
rrosipal@mail.arc.nasa.gov

Cynthia Rudin
Princeton University, NEC
133 Old Farm Circle
Williamsville NY 14221
USA
crudin@princeton.edu

Rauf Sadykhov
Belarussian State University
Informatics & Radioelectronics
P; Brovkastreet 6, 220600 Minsk
Belarus
rsadykhov@gw.bsuir.unibel.by

Bernhard Schölkopf
Max Planck Institute for
Biological Cybernetics, Dept. Schölkopf
Spemannstrasse 3, 72076 Tübingen
Germany
bernhard.schoelkopf@tuebingen.mpg.de

Armin Shmilovici Leib
Ben-Gurion University
Dept. Information Systems
PO Box 653, 84105 Beer-Sheva
Israel
armin@bgumail.bgu.ac.il

Dmitriy Shutin
Graz University of Technology
Inst. f. Nachrichtentechnik
Inffeldgasse 12, 8010 Graz
Austria
dshutin@inw.tugraz.at

Yoram Singer
The Hebrew Univ., Givat-Ram campus
School of Computer Sc. & Eng.
Jerusalem 91904
Israel
singer@cs.huji.ac.il

Steve Smale
University of California
Dept. Mathematics
Berkeley, CA 94720-384
USA
smale@math.berkeley.edu

Masashi Sugiyama
Tokyo Inst. Technology
Dept. Computer Science 2-12-1
O-okayama Meguro-ku, Tokyo 152-8552
Japan
sugi@og.cs.titech.ac.jp

Johan Suykens
K.U. Leuven
Dept. Elektrotechniek, ESAT-SCD-SISTA
Kasteelpark Arenberg 10
3001 Heverlee, Belgium
johan.suykens@esat.kuleuven.ac.be

Peter Tino
The University of Birmingham
School of Computer Science
Edgbaston
Birmingham B15 2TT, UK
pxt@cs.bham.ac.uk

Theodore Trafalis
University of Oklahoma
School of Industrial Engineering
202 W. Boyd, Suite 124
Norman OK 73019, USA
ttrafalis@ou.edu

Karl Tuyls
VUB
Computational Modeling Lab
Building F, Campus Etterbeek
Belgium
ktuyls@vub.ac.be

Jozsef Valyon
Budapest Univ. Techn. and Economics
Dept. Measurement and Inf. Systems
Magyar tudósok körútja 2
H-1117 Budapest, Hungary
valyon@mit.bme.hu

Joos Vandewalle
K.U. Leuven
Dept. Elektrotechniek, ESAT-SCD-SISTA
Kasteelpark Arenberg 10
3001 Heverlee, Belgium
joos.vandewalle@esat.kuleuven.ac.be

Tony Van Gestel
K.U. Leuven
Dept. Elektrotechniek, ESAT-SCD-SISTA
Kasteelpark Arenberg 10
3001 Heverlee, Belgium
tony.vangestel@esat.kuleuven.ac.be

Sabine Van Huffel
K.U. Leuven
Dept. Elektrotechniek, ESAT-SCD-SISTA
Kasteelpark Arenberg 10
3001 Heverlee, Belgium
sabine.vanhuffel@esat.kuleuven.ac.be

Vladimir Vapnik
NEC Research
4 Independence Way
Princeton, NJ 08540
USA
vlad@research.nj.nec.com

Andrey Vasnev
New Economic School, Moscow
NES, Nakhimovsky Prospekt 47, Suite 1721
117418 Moscow
Russian Federation
avasnev@nes.ru

Jakob Verbeek
University of Amsterdam
Kruislaan 403
1098 SJ Amsterdam
Nederland
jverbeek@science.uva.nl

M. Vidyasagar
Tata Consultancy Services
6th floor, Khan Lateefkahn Estate
Fateh Maidan Road, Hyderabad 500 001
India
sagar@atc.tcs.co.in

Michael Werman
The Hebrew University
Computer Science, Givat Ram
Jerusalem
Israel
werman@cs.huji.ac.il

Slawo Wesolkowski
University of Waterloo
Systems Design Engineering
Waterloo, Ontario N2H 1K6
Canada
s.wesolkowski@ieee.org

Marco Wiering
University Utrecht
Inst. Information & Computer Science
Padualaan 14, 3508 TB Utrecht
Nederland
marco@cs.uu.nl

Atila Yilmaz
Hacettepe University
Electrical and Electronics Eng.
06532 Beytepe, Ankara
Turkey
ayilmaz@hacettepe.edu.tr

Gaetano Zanghirati
Univ. Di Ferrara, Dip. Matematica
Via Machiavelli, 35
44100 Ferrara
Italy
g.zanghirati@unife.it

Luca Zanni
Univ. Modena and Reggio Emilia
Dept. Mathematics
Via Campi 213/B, 41100 Modena
Italy
zanniluca@unimo.it

Onno Zoeter
SNN, University of Nijmegen
Geert Grooteplein 21
6525 EZ Nijmegen
Nederland
orzoeter@snn.kun.nl

Index