Otto-von-Guericke Universität Magdeburg

Fakultät für Elektrotechnik
und Informationstechnik

Estimation of Nonparametric Probability Density Functions with Applications to Automatic Speech Recognition

Dissertation
zur Erlangung des akademischen Grades
Doktoringenieur (Dr.-Ing.)
von Dipl.-Ing. Martin Schafföner
geb. am 18. März 1978 in Burg
genehmigt durch die Fakultät für Elektrotechnik und Informationstechnik
der Otto-von-Guericke Universität Magdeburg

Gutachter: Prof. Dr. rer. nat. Andreas Wendemuth
Prof. Dr.-Ing. habil. Rüdiger Hoffmann

Promotionskolloqium am 28. November 2007

Für meine Eltern

Zusammenfassung

Im letzten Jahrzehnt hat ein neues Lernparadigma, die sog. Strukturelle Risikominimierung (SRM), viel Beachtung im Bereich des Maschinellen Lernens gefunden. Auf SRM basierende Lernmaschinen, wie z. B. Support Vector Machines (SVMs) oder Kernel Fisher Discriminants (KFDs) wurden überaus erfolgreich zur Lösung von Mustererkennungs- und Funktionsregressionsproblemen angewendet. Die Möglichkeit der SRM, gleichzeitig das Fehlerrisiko auf Trainingsdaten und die Komplexität der Lernmaschine zu minimieren, führt zu besserer Generalisierungsfähigkeit als konventionelle Empirische Risikominimierung (ERM), insbesondere bei kleinen Lernstichproben.

Die vorliegende Arbeit beschäftigt sich mit der Anwendung der SRM auf die Schätzung von Wahrscheinlichkeitsdichtefunktionen (WDFs). WDFs werden u. a. als Emissionswahrscheinlichkeiten bei der Modellierung von kontinuierlich-wertigen Mustersequenzen mit Hilfe von Hidden Markov Models (HMMs), z. B. in der automatischen Spracherkennung (ASE), benutzt.

Diese Dissertation untersucht und entwickelt Methoden zur effizienten Schätzung von dünn besetzten Kernel WDFs durch Regression der empirischen kumulativen Verteilungsfunktion (EKVF). Wir stellen ein neues Verfahren zur Bestimmung einer dünn besetzen Näherung der orthogonalen kleinste-Quadrate-Regressionslösung durch Vorwärtsauswahl relevanter Trainingspunkte vor. Das Verfahren beruht auf einer neuartigen speichereffizienten dünn besetzten Aktualisierung der orthogonalen Zerlegung der Problemmatrix. Die Methode wird auf Standardproblemen bis zu fünf Dimensionen untersucht, wobei sie deutlich verbesserte Leistungen im Vergleich mit traditionellen parametrischen Modellen wie Normal-Mischverteilungen (NMV) und ähnliche Leistung wie die theoretisch optimalen, vollbesetzten Parzen Windows WDF-Modelle zeigt.

Die Arbeit zeigt weiterhin, dass die Methode der EKVF-Regression wegen der Komplexität der EKVF bei hochdimensionalen Problemen, wie sie z. B. in der ASE erscheinen, allerdings nicht angewendet werden kann. Eine weitere Methode bestimmt daher WDFs durch Anwendung der Bayes'schen Regel auf Posterioriwahrscheinlichkeiten, die aus den Ausgaben von paarweisen Trennfunktionen wie SVMs oder KFDs kalibriert wurden. Dieser Ansatz wird in einem Monophon-HMM ASE-System auf der „Resource Management" Sprachdatenbank getestet, wobei traditionelle HMM-NMV-Modelle deutlich überboten werden, insbesondere auf beschränkten Stichproben. Die Experimente zeigen also eine signifikante Verbesserung der Generalisierungsfähigkeit durch die neuartigen WDF-Modelle.

Eine neuartige Programmbibliothek zur Durchführung der teilweise umfangreichen Experimente wird vorgestellt. Primäres Ziel wird auf schnelle Berechnungen,

Einfachheit in Bezug auf Ausdruck von Algorithmen und Funktionserweiterung sowie auf Flexibilität zur bestmöglichen Ausnutzung vorteilhafter Eigenschaften von Algorithmen gelegt. Die Bibliothek folgt einem objektorientierten Design und wurde in C++ implementiert. Zur Erhöhung der Produktivität ist die Bibliothek mit feinkörniger Ablaufüberwachung, einem objektorientierten Persistenzmodell, transparenter Fehlerbehandlung sowie Parallelisierung auf Computersystemen mit verteiltem Speicher (sog. Cluster) ausgestattet.

Abstract

During the last decade, a new learning paradigm called Structural Risk Minimization (SRM) derived from Statistical Learning Theory, has become widely studied in machine learning. Machines implementing SRM, e. g., Support Vector Machines (SVMs) and Kernel Fisher Discriminants (KFDs), have been very successfully used for solving pattern recognition and function regression problems. SRM's ability to simultaneously minimize the risk of error on training data and the complexity of a learning machine results in better generalization capability than plain Empirical Risk Minimization (ERM), especially if the amount of training data is limited.

The present work is devoted to applying SRM to the problem of probability density function (PDF) estimation. When modeling sequences of continuous-valued events using Hidden Markov Models (HMMs), e. g., automatic speech recognition (ASR), PDFs are used to model the emission probabilities of the HMMs' states.

This thesis investigates and develops methods to efficiently train sparse kernel PDF models by regression of the empirical cumulative distribution function (ECDF). A new method for obtaining a sparse approximation of the orthogonal least-squares regression solution by forward-selection of relevant samples is presented, where a novel memory-efficient thin update of the orthogonal decomposition is used. This method is evaluated on standard benchmark problems of up to five dimensions, showing superior performance to traditional parametric Gaussian Mixture Models (GMMs) and similar performance to the theoretically optimal, non-sparse Parzen windows PDF models.

However, it is found that this new method cannot be applied to the problem of estimating PDFs for ASR due to the complexity of the ECDF in high dimensions. Instead, posterior class probabilities calibrated from the outputs of binary discriminants such as SVMs or KFDs are turned into class-conditional PDFs using Bayes' rule. This approach is tested within a monophone HMM ASR system on the Resource Management task, outperforming traditional HMM-GMM systems significantly, especially on random limited samples which demonstrates the new models' improved generalization ability on small-sample problems.

In order to realize these large-scale experiments, a novel machine learning software library is presented. Primary focus is put on fast computations, simplicity both in terms of expressing algorithms and extending functionality, and flexibility in order to properly appreciate algorithms' properties and advantages. The software library follows an object-oriented design and has been implemented in C++. For productivity, the library is equipped with fine-grained tracing, an object-oriented persistence model, transparent error handling and parallelization on distributed-memory computer clusters.

Acknowledgments

I would like to express my gratefulness to my supervisor Prof. Andreas Wendemuth for his support in making this dissertation possible. My thanks also goes to my colleagues at the "Kognitive Systeme" chair in Magdeburg, especially to Edin Andelic, Marcel Katz, and Sven Krüger. I am deeply indebted to them for the valuable discussions we had.

Furthermore, I want to thank the federal state "Sachsen-Anhalt" for supporting this work through a grant. This work was also made possible through a grant by the "Friedrich-Naumann-Stiftung" with funds from the "Bundesministerium für Bildung und Forschung".

Last but not least, I wish to thank my family and my beloved wife Eva-Kristin for supporting me at all times. Without them, this work would not have been possible.

Contents

List of Figures

List of Tables

List of Symbols

$\boldsymbol{x}$	column-vector of unspecified dimension
$\boldsymbol{x}_{[N]}$	column-vector with N rows
$x_{(j)}$	the jth row (element) of a column vector $\boldsymbol{x}$
$\boldsymbol{X}$	matrix
$\boldsymbol{X}_{[N\times M]}$	matrix with N rows and M columns
$\boldsymbol{x}_{(j)}$	the jth column vector from a matrix $\boldsymbol{X}$
$x_{(i,j)}$	the element located in row i and column j of a matrix $\boldsymbol{X}$
$\vec{x}$	sequence of events $\langle x_1; x_2; \ldots; x_t \rangle$
$\boldsymbol{x}_i$	the ith event from a vector-valued sequence $\vec{\boldsymbol{x}}$

List of Abbreviations

ACML	*AMD* core math library
AD	Anderson-Darling
ALU	arithmetic-logic unit
ASR	automatic speech recognition
BLAS	Basic Linear Algebra Subprograms
CPU	central processing unit
ECDF	empirical cumulative distribution function
EM	Expectation-Maximization
ERM	Empirical Risk Minimization
ϵ-SVD	support vector probability density method using the ϵ-insensitive loss function
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
i. i. d.	independent and identically distributed
IPP	*Intel* Performance Primitives
KBM	Kernel Binary Machine
KFD	Kernel Fisher Discriminant
KKT	Karush-Kuhn-Tucker
KS	Kolmogorov-Smirnov
LAPACK	Linear Algebra Package
LOO	leave-one-out

LS-SVM	Least Squares Support Vector Machine
MFCC	Mel-frequence cepstral coefficient
MGS	modified Gram-Schmidt
MKL	*Intel* math kernel library
ML	Maximum Likelihood
MLP	Multi-Layer Perceptron
MPI	Message Passing Interface
NUMA	non-uniform memory access
OLS	orthogonal least squares
OROLS	order-recursive orthogonal least squares
OVO	one-vs.-one
OVR	one-vs.-rest
PDF	probability density function
PVM	Parallel Virtual Machine
QP	Quadratic Programming
RKHS	reproducing kernel Hilbert space
RM1	Resource Management
SMO	Sequential Minimal Optimization
SMP	symmetric multi-processor
SRM	Structural Risk Minimization
SVM	Support Vector Machine
UML	Unified Modeling Language
VC	Vapnik-Chervonenkis
XML	eXtendible Markup Language

1 Introduction

Machine Learning has made great progress in the last two decades especially with the development of Structural Risk Minimization (SRM). SRM is an inductive (learning) principle which, loosely speaking, at training time aims at simultaneously minimizing prediction errors and complexity, or in other words, the capacity of the learning machine. The controlled capacity of the machine is then expected to reduce prediction errors on unseen data compared to learning algorithms only minimizing the prediction error, i. e., the learning machine can generalize well to unobserved data. In chapter 2 we will discuss what we understand as learning, which conditions for learning to take place must be met, what generalization means more technically, and how generalizing machines can be obtained using the SRM inductive principle.

The method of SRM has mostly been studied in the pattern recognition or, in other words, pattern discrimination scenario, and in the function regression scenario. Popular discriminant methods implementing SRM are Support Vector Machines (SVMs) and Kernel Fisher Discriminants (KFDs), which will be revisited in chapter 3.

Automatic speech recognition (ASR) is, in general, quite a complex undertaking. The task is to deduce text from an observed segment of speech sound. This can be regarded as a kind of pattern recognition problem where using some observed pattern (the acoustic utterance) the corresponding class (the text) from an alphabet (all allowed texts) is to be chosen. In a naive approach, this problem could be solved using standard pattern recognition techniques which would assign the utterance to its text based on some statistical dependence the learning machine has deduced from training material.

However, the task of automatic speech recognition (ASR) to date is too complex to be solved by purely discriminating learning machines. Particular difficulties to be tackled are large vocabularies, and acoustic as well as temporal variabilities of utterances. Instead, one has to apply some prior knowledge about the problem of speech production in order to design a suitable learning machine. A well-established solution to these challenges is to follow an "analysis by synthesis" approach, namely dividing the task into language models, which capture structures of texts at the level of words, pronunciation models, which split words into phonetic units, and acoustic models, which ultimately model the acoustic realization of these sub-word phonetic units. The acoustic models commonly consist of Hidden Markov Models (HMMs) with continuous emission probability densities, typically weighted sums of multivariate normal densities (so-called Gaussian Mixture Models (GMMs)). Chapter 4 will revisit the topic in more depth.

Training of the parameters of the HMM-GMM acoustic models is usually based on Empirical Risk Minimization (ERM) which does not, in contrast to SRM, consider the complexity of the resulting learning machine. Modeling probability density functions (PDFs) from samples using the SRM inductive principle has not been studied much. This thesis will, therefore, consider methods for constructing emission PDF for HMM-based speech recognition systems using the promising SRM inductive principle.

In chapter 5 we will discuss an approach to directly derive sparse PDF models from data samples. Starting with the definition that a probability density function is a function whose integral is a distribution function, we will revisit how a reference distribution function, i. e., the empirical cumulative distribution function (ECDF), can be approximated by the integral of the desired PDF using SRM-based function regression. Existing approaches to this general idea suffer from a number of problems, most notably large training memory requirements. We will introduce a method which overcomes this limitation, especially for large training sample sets and relatively sparse approximations. However, we will also see the limits of PDF-training based on regression of the ECDF applied to PDF construction in the ASR scenario.

Chapter 6 will be concerned with the construction of PDFs for HMMs from standard binary discriminant machines such as SVM and KFD. This chapter discusses methods to extract probability measures from these binary machines and how to combine them into multi-class probability measures. Experiments on the Resource Management (RM1) speech recognition task support the idea and show improved generalization performance compared to conventional GMM-based emission PDFs.

Chapter 7 focuses on the software implementation of the various ideas necessary to build PDFs using SRM-based machines. The large memory and computation time requirements as well as the diversity of the investigated problem classes failed straightforward software solutions and consequently demanded a substantial software design in a way that equally appreciates common and divergent properties of different parts of the learning machines. We will present an object-oriented library of reusable software components with well-defined interfaces employing advanced techniques such as templatization, object persistence via serialization to XML files, efficient parallelization, and event logging, while still putting primary focus on speed and efficiency.

The main results of this thesis along with possible future research directions will be summarized in chapter 8.

2 Learning from Samples

Learning can be considered as the problem of finding a dependence using a (usually limited number of) observations (or samples). In this chapter we will first consider the general setting of a learning problem. We will define *risk* and *induction principles*. By studying (asymptotic) properties of the important *empirical risk minimization* inductive principle and defining concepts such as *capacity measures* for learning machines we will arrive at a better inductive principle, the *structural risk minimization* inductive principle. This chapter draws heavily from the fundamental ideas summarized in [Vapnik, 1998] and [Vapnik, 2000].

2.1 Function estimation

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

- A generator G of random vectors $\boldsymbol{x} \in \mathbb{R}^D$ drawn independently from a fixed, but unknown probability distribution $F(\boldsymbol{x})$,
- a supervisor S which assigns an output value $y \in \mathcal{Y}$ to each input vector $\boldsymbol{x}$ according to an also fixed, but unknown conditional distribution function $F(y|\boldsymbol{x})$,
- a learning machine implementing a set of functions $f(\boldsymbol{x}, \boldsymbol{\alpha})$, $\boldsymbol{\alpha} \in \mathcal{A}$, where $\mathcal{A}$ is a set of parameters (not necessarily vectors, but any abstract parameter).

Learning, then, is the problem of finding the function that best approximates the supervisor's output according to some criterion from the the given set of functions $f(\boldsymbol{x}, \boldsymbol{\alpha})$, $\boldsymbol{\alpha} \in \mathcal{A}$.

The joint probability distribution

$$F(\boldsymbol{x}, y) = F(y|\boldsymbol{x})F(\boldsymbol{x}) \tag{2.1}$$

is unknown in but the most trivial cases. Instead, only a limited number of N observations drawn according to this joint distribution function are known, making up a set of training samples:

$$\mathcal{Z} : \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\} \tag{2.2}$$

2.2 Risk minimization

For the purpose of choosing the best approximation of the supervisor's output one measures the loss (or, in other words, the cost of error), $L(y, f(\boldsymbol{x}, \boldsymbol{\alpha}))$. Following the well-known definition of risk which states that the risk of an event is the product of the cost of this event and the probability of occurrence of this event, the total risk of an approximation associated with a parameter vector $\boldsymbol{\alpha}$ can be expressed by integrating the loss (or cost) function over the entire probability space, resulting in the risk functional

$$R(\boldsymbol{\alpha}) = \int L(y, f(\boldsymbol{x}, \boldsymbol{\alpha}))\, \mathrm{d}\, F(y, \boldsymbol{x}) \tag{2.3}$$

The objective is to determine the function $f(\boldsymbol{x}, \boldsymbol{\alpha}_0)$ which minimizes $R(\boldsymbol{\alpha})$ over the set of functions $f(\boldsymbol{x}, \boldsymbol{\alpha})$, $\boldsymbol{\alpha} \in \mathcal{A}$. Only in trivial cases can this be done directly.

2.2.1 Induction principle

For convenience, the following shorthand notations are used: $\mathcal{Z} = \mathbb{R}^D \times \mathcal{Y}$ and $Q(\boldsymbol{z}, \boldsymbol{\alpha}) = L(y, f(\boldsymbol{x}, \boldsymbol{\alpha}))$. The risk (2.3) can then be rewritten as

$$R(\boldsymbol{\alpha}) = \int Q(\boldsymbol{z}, \boldsymbol{\alpha})\, \mathrm{d}\, F(\boldsymbol{z}) \tag{2.4}$$

If the risk functional (2.4) is to be minimized with an unknown distribution function $F(\boldsymbol{z})$, an *inductive principle* can be applied. The Empirical Risk Minimization (ERM) inductive principle (ERM principle) is quite well-known and works by replacing the true risk $R(\boldsymbol{\alpha})$ with the so-called empirical risk functional

$$R_{\mathrm{emp}}(\boldsymbol{\alpha}) = \frac{1}{N} \sum_{n=1}^{N} Q(\boldsymbol{z}_n, \boldsymbol{\alpha}) \tag{2.5}$$

which is constructed based on the training set (2.2). Notice how the integration in (2.4) has been replaced by summation over discrete events which are assumed (lacking further information) to be uniformly distributed.

The function $Q(\boldsymbol{z}, \hat{\boldsymbol{\alpha}})$ minimizing the actual risk is then approximated by the function $Q(\boldsymbol{z}, \boldsymbol{\alpha}_N)$ which minimizes the empirical risk (2.5).

An inductive principle (not limited to the ERM principle) defines a *learning process* if the learning machine employs this inductive principle when choosing the approximation given a set of observations.

A number of properties must be investigated for a learning process, i. e., conditions for consistency, rate of convergence, how to control the rate of convergence, and how to construct algorithms that can control the rate of convergence.

2.3 Consistency of a learning process

While we are ultimately interested in the performance of a learning process with limited data, it is, nevertheless, important to investigate consistency, i. e., the asymptotic performance of the learning process in the limit w. r. t. the size of the training sample because consistency guarantees that the the constructed theory is general and cannot theoretically be improved.

Definition 1. *The* ERM *principle is consistent for the set of functions* $Q(\boldsymbol{z}, \boldsymbol{\alpha})$, $\boldsymbol{\alpha} \in \mathcal{A}$, *and for the probability distribution* $F(\boldsymbol{z})$ *if the following two conditions hold:*

$$R(\boldsymbol{\alpha}_N) \xrightarrow[l\to\infty]{P} \inf_{\boldsymbol{\alpha}\in\mathcal{A}} R(\boldsymbol{\alpha}) \tag{2.6}$$

$$R_{\text{emp}}(\boldsymbol{\alpha}_N) \xrightarrow[l\to\infty]{P} \inf_{\boldsymbol{\alpha}\in\mathcal{A}} R(\boldsymbol{\alpha}) \tag{2.7}$$

It is, unfortunately, impossible to obtain conditions for consistency of the ERM inductive principle in terms of general characteristics of the set of functions and the probability measure based on the given classical definition (definition 1) because the definition includes cases of *trivial consistency*.

2.3.1 Trivial consistency

Trivial cases of consistency exist if the set of functions $Q(\boldsymbol{z}, \boldsymbol{\alpha})$, $\boldsymbol{\alpha} \in \mathcal{A}$, contains a minorizing function. This means that a theory of consistency based on definition 1 must determine whether a case of trivial consistency is possible, i. e., the specific functions of the given set must be considered individually. Therefore, a definition of consistency which excludes cases of trivial consistency must be used.

Definition 2. *An inductive principle is* non-trivially consistent *for the set* $Q(\boldsymbol{z}, \boldsymbol{\alpha})$, $\boldsymbol{\alpha} \in \mathcal{A}$, *and for the probability distribution* $F(\boldsymbol{z})$ *if for any non-empty subset* $\mathcal{A}(c)$, $c \in (-\infty, \infty)$, *of this set of functions, following*

$$\mathcal{A}(c) = \left\{ \boldsymbol{\alpha} : \int Q(\boldsymbol{z}, \boldsymbol{\alpha}) \, \mathrm{d}\, F(\boldsymbol{z}) > c,\ \boldsymbol{\alpha} \in \mathcal{A} \right\} \tag{2.8}$$

the convergence

$$\inf_{\boldsymbol{\alpha}\in\mathcal{A}(c)} R_{\text{emp}}(\boldsymbol{\alpha}) \xrightarrow[N\to\infty]{P} \inf_{\boldsymbol{\alpha}\in\mathcal{A}(c)} R(\boldsymbol{\alpha}) \tag{2.9}$$

is valid.

2.3.2 Uniform one-sided convergence

We now need to find conditions for non-trivial consistency, which leads to the key theorem of learning theory:

Theorem 1 (Vapnik-Chervonenkis). *Let $Q(\boldsymbol{z}, \boldsymbol{\alpha})$, $\boldsymbol{\alpha} \in \mathcal{A}$, be a set of functions satisfying*

$$A \leq \int Q(\boldsymbol{z}, \boldsymbol{\alpha}) \, \mathrm{d}\, F(\boldsymbol{z}) \leq B, \quad A \leq R(\boldsymbol{\alpha}) \leq B. \tag{2.10}$$

It is then necessary and sufficient for an inductive principle to be consistent that the empirical risk $R_{\mathrm{emp}}(\boldsymbol{\alpha})$ converges uniformly to the actual risk $R(\boldsymbol{\alpha})$ over the set $Q(\boldsymbol{z}, \boldsymbol{\alpha})$, $\boldsymbol{\alpha} \in \mathcal{A}$, such that

$$\lim_{l \to \infty} P \left\{ \sup_{\boldsymbol{\alpha} \in \mathcal{A}} \left(R(\boldsymbol{\alpha}) - R_{\mathrm{emp}}(\boldsymbol{\alpha}) \right) > \varepsilon \right\} = 0, \quad \forall \varepsilon > 0. \tag{2.11}$$

This type of uniform convergence is called *one-sided* convergence. Conceptually, this theorem asserts that consistency is necessarily and sufficiently determined by the "worst" function in the set $Q(\boldsymbol{z}, \boldsymbol{\alpha})$, $\boldsymbol{\alpha} \in \mathcal{A}$.

2.3.3 Uniform two-sided convergence

With theorem 1 the problem of consistency of the ERM principle (definition 1) was replaced by the problem of uniform one-sided convergence (2.11). If one is interested in necessary and sufficient conditions for uniform one-sided convergence, two stochastic processes, which are called *empirical processes*, must be considered.

The sequence of random variables

$$\begin{aligned} \xi^N &= \sup_{\boldsymbol{\alpha} \in \mathcal{A}} \left| R(\boldsymbol{\alpha}_N) - R_{\mathrm{emp}}(\boldsymbol{\alpha}_N) \right| \\ &= \sup_{\boldsymbol{\alpha} \in \mathcal{A}} \left| \int Q(\boldsymbol{z}, \boldsymbol{\alpha}) \, \mathrm{d}\, F(\boldsymbol{z}) - \frac{1}{N} \sum_{n=1}^{N} Q(\boldsymbol{z}_n, \boldsymbol{\alpha}) \right|, \quad N = 1, 2, \ldots \end{aligned} \tag{2.12}$$

is a *two-sided* empirical process because it depends both on the probability $F(\boldsymbol{z})$ and the set of functions $Q(\boldsymbol{z}, \boldsymbol{\alpha})$, $\boldsymbol{\alpha} \in \mathcal{A}$. For this empirical process to converge to zero in probability, the following must hold true:

$$\lim_{l \to \infty} P \left\{ \sup_{\boldsymbol{\alpha} \in \mathcal{A}} \left| \int Q(\boldsymbol{z}, \boldsymbol{\alpha}) \, \mathrm{d}\, F(\boldsymbol{z}) - \frac{1}{N} \sum_{n=1}^{N} Q(\boldsymbol{z}_n, \boldsymbol{\alpha}) \right| > \varepsilon \right\} = 0, \quad \forall \varepsilon > 0 \tag{2.13}$$

Together with the empirical process ξ^N we consider the *one-sided* empirical process given by the sequence of random variables

$$\xi_+^N = \sup_{\boldsymbol{\alpha} \in \mathcal{A}} \left(\int Q(\boldsymbol{z}, \boldsymbol{\alpha}) \, \mathrm{d}\, F(\boldsymbol{z}) - \frac{1}{N} \sum_{n=1}^{N} Q(\boldsymbol{z}_n, \boldsymbol{\alpha}) \right)_+, \quad N = 1, 2, \ldots \tag{2.14}$$

with

$$(u)_+ = \begin{cases} u & \text{if } u > 0, \\ 0 & \text{otherwise} \end{cases}$$

For ξ_+^N to converge to zero in probability, the following condition must hold true:

$$\lim_{l\to\infty} P\left\{ \sup_{\boldsymbol{\alpha}\in\mathcal{A}} \left(\int Q(\boldsymbol{z},\boldsymbol{\alpha})\,\mathrm{d}\,F(\boldsymbol{z}) - \frac{1}{N}\sum_{n=1}^{N} Q(\boldsymbol{z}_n,\boldsymbol{\alpha}) \right)_+ > \varepsilon \right\} = 0, \quad \forall\varepsilon > 0. \tag{2.15}$$

This condition is, according to theorem 1, necessary and sufficient for the ERM method to be consistent.

Using the law of large numbers in the functional space, uniform two-sided convergence will be used to construct conditions of uniform one-sided convergence. Necessary and sufficient conditions both for uniform one-sided convergence and uniform two-sided convergence are obtained using the *entropy of a set of functions* $Q(\boldsymbol{z},\boldsymbol{\alpha})$, $\boldsymbol{\alpha}\in\mathcal{A}$, on a sample of size N. In the derivation, we will limit ourselves to the set of indicator functions (functions taking only one of two values); a complete derivation for sets of real bounded functions can be found in [Vapnik, 2000].

2.3.4 Entropy of the set of indicator functions

Let $Q(\boldsymbol{z},\boldsymbol{\alpha})$, $\boldsymbol{\alpha}\in\mathcal{A}$, be a set of indicator functions. Consider a sample $\boldsymbol{z}_1,\ldots,\boldsymbol{z}_N$. The diversity of the set of functions $Q(\boldsymbol{z},\boldsymbol{\alpha})$, $\boldsymbol{\alpha}\in\mathcal{A}$, is characterized by the quantity $N^{\mathcal{A}}(\boldsymbol{z}_1,\ldots,\boldsymbol{z}_N)$ which evaluates the number of different possible separations of the given sample using the set of indicator functions. Then,

$$H^{\mathcal{A}}(\boldsymbol{z}_1,\ldots,\boldsymbol{z}_N) = \ln N^{\mathcal{A}}(\boldsymbol{z}_1,\ldots,\boldsymbol{z}_N) \tag{2.16}$$

is called the *random entropy*. The random entropy is a random variable because it was constructed using i.i.d. data. The expectation over the joint distribution function $F(\boldsymbol{z}_1,\ldots,\boldsymbol{z}_N)$

$$H^{\mathcal{A}}(N) = \mathrm{E}H^{\mathcal{A}}(\boldsymbol{z}_1,\ldots,\boldsymbol{z}_N) \tag{2.17}$$

is then called the Vapnik-Chervonenkis (VC) entropy of the set of indicator functions $Q(\boldsymbol{z},\boldsymbol{\alpha})$, $\boldsymbol{\alpha}\in\mathcal{A}$, on the sample size N.

2.3.5 Conditions for uniform one-sided convergence

Using the VC entropy, we can find the following condition for uniform two-sided convergence [Vapnik, 2000]:

Theorem 2. *Under some conditions of measurability on the set of indicator functions $Q(\boldsymbol{z}, \boldsymbol{\alpha})$, $\boldsymbol{\alpha} \in \mathcal{A}$, a necessary and sufficient condition for uniform two-sided convergence* (2.13) *is*

$$\lim_{N\to\infty} \frac{H^{\mathcal{A}}(N)}{N} = 0. \tag{2.18}$$

Uniform two-sided convergence of the ERM inductive principle can be described as

$$\begin{aligned} \lim_{l\to\infty} P\Bigg\{ \left[\sup_{\boldsymbol{\alpha}\in\mathcal{A}} (R(\boldsymbol{\alpha}) - R_{\text{emp}}(\boldsymbol{\alpha})) > \varepsilon\right] \\ \text{or } \left[\sup_{\boldsymbol{\alpha}\in\mathcal{A}} (R_{\text{emp}}(\boldsymbol{\alpha}) - R(\boldsymbol{\alpha})) > \varepsilon\right] \Bigg\} = 0 \quad \forall \varepsilon > 0 \end{aligned} \tag{2.19}$$

Condition (2.19) includes the case of uniform one-sided convergence (2.15). Therefore, it is a sufficient condition for consistency of the ERM method. Since we can neglect consistency with respect to maximizing the empirical risk when solving learning problems, the second condition on the left-hand side of (2.19) may be violated.

Therefore, we can finally describe conditions under which uniform one-sided convergence, i. e., for *minimizing* the empirical risk, takes place. Consider the set of real bounded functions $Q(\boldsymbol{z}, \boldsymbol{\alpha})$, $\boldsymbol{\alpha} \in \mathcal{A}$, and another set of functions $Q^*(\boldsymbol{z}, \boldsymbol{\alpha})$, $\boldsymbol{\alpha} \in \mathcal{A}^*$, such that

$$Q(\boldsymbol{z}, \boldsymbol{\alpha}) - Q^*(\boldsymbol{z}, \boldsymbol{\alpha}) \geq 0 \quad \forall z \tag{2.20}$$

$$\int (Q(\boldsymbol{z}, \boldsymbol{\alpha}) - Q^*(\boldsymbol{z}, \boldsymbol{\alpha}))\, \mathrm{d}\, F(z) \leq \delta. \tag{2.21}$$

Theorem 3. *In order for one-sided convergence of the ERM method to hold, it is necessary and sufficient that for any positive δ and η there exists a set of functions $Q^*(\boldsymbol{z}, \boldsymbol{\alpha})$, $\boldsymbol{\alpha} \in \mathcal{A}^*$, obeying* (2.20) *and* (2.21) *such that the following holds:*

$$\lim_{N\to\infty} \frac{H^{\mathcal{A}^*}(N)}{N} < \eta \tag{2.22}$$

Note that this condition is weaker than the (only sufficient) condition (2.18).

2.4 The rate of convergence

After having established *conditions* on the consistency of the ERM method, we will now be concerned with the *rate* of convergence.

According to theorem 2 we have a sufficient condition for uniform one-sided convergence, and theorem 3 extends this to necessary and sufficient conditions for uniform one-sided convergence. However, both of them do not evaluate how fast

the achieved risks $R(\boldsymbol{\alpha}_N)$ converges to the minimal risk $R(\boldsymbol{\alpha}_0)$. If the asymptotic rate of convergence follows

$$P\left\{(R(\boldsymbol{\alpha}_N) - R(\boldsymbol{\alpha}_0)) > \varepsilon\right\} < e^{-c\varepsilon^2 N}, \quad \varepsilon > 0 \tag{2.23}$$

for any $N > N_0$ and some constant c, the rate is said to be high.

Using $N^{\mathcal{A}}(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N)$, two new concepts useful for characterizing the rate of convergence can be constructed:

1. The *annealed VC entropy*

$$H^{\mathcal{A}}_{\text{ann}}(N) = \ln \mathrm{E} N^{\mathcal{A}}(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N) \tag{2.24}$$

2. The *growth function*

$$G^{\mathcal{A}}(N) = \ln \sup_{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N} N^{\mathcal{A}}(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N) \tag{2.25}$$

These concepts obey

$$H^{\mathcal{A}}(N) \leq H^{\mathcal{A}}_{\text{ann}}(N) \leq G^{\mathcal{A}}(N) \tag{2.26}$$

for any N.

It can be shown [Vapnik, 1998] that

$$\lim_{N \to \infty} \frac{H^{\mathcal{A}}_{\text{ann}}(N)}{N} = 0 \tag{2.27}$$

is a sufficient condition for a high rate of convergence. The problem with theorem 3 and equation (2.27) is that both describe conditions for the convergence and its rate for the ERM method if the probability measure $F(\boldsymbol{z})$ is known. It turns out that

$$\lim_{N \to \infty} \frac{G^{\mathcal{A}}(N)}{N} = 0 \tag{2.28}$$

is a necessary and sufficient condition for consistency of the ERM method and also a sufficient condition for a fast rate of convergence [Vapnik, 1998] for *any* probability measure $F(z)$.

2.4.1 Bounds on the rate of convergence

So far, we have established conditions for consistency of the ERM method via uniform one-sided convergence of the empirical risk to the true risk. Also, concepts for the rate of convergence of the ERM method have been introduced. Next, we will focus on bounds on the rate of convergence which may actually be computed.

Using the two capacity concepts (2.24) and (2.25) two bounds on the rate of convergence will be discussed, i. e., distribution-dependent bounds based on the

annealed entropy function (2.24) and distribution-independent bounds based on the growth function (2.25). Since these two bounds are non-constructive, a different measure for the capacity of a set of functions (the VC dimension), which is scalar computable for any set of functions, will be used to obtain constructive distribution-independent bounds [Vapnik, 1998]. Unless necessary otherwise, the derivations will be limited to $Q(\boldsymbol{z}, \boldsymbol{\alpha})$, $\boldsymbol{\alpha} \in \mathcal{A}$, from the set of indicator functions.

The following two bounds on the rate of uniform convergence are the basic inequalities in the theory of bounds [Vapnik, 1998].

Theorem 4. *The following inequality holds true:*

$$P\left\{\sup_{\boldsymbol{\alpha}\in\mathcal{A}}\left|\int Q(\boldsymbol{z},\boldsymbol{\alpha})\,\mathrm{d}\,F(\boldsymbol{z}) - \frac{1}{N}\sum_{n=1}^{N} Q(\boldsymbol{z}_n,\boldsymbol{\alpha})\right| > \varepsilon\right\} \leq 4e^{\left(\frac{H_{\mathrm{ann}}^{\mathcal{A}}(2N)}{N}-\varepsilon^2\right)N} \tag{2.29}$$

Theorem 5. *The following inequality holds true:*

$$P\left\{\sup_{\boldsymbol{\alpha}\in\mathcal{A}}\frac{\int Q(\boldsymbol{z},\boldsymbol{\alpha})\,\mathrm{d}\,F(\boldsymbol{z}) - \frac{1}{N}\sum_{n=1}^{N} Q(\boldsymbol{z}_n,\boldsymbol{\alpha})}{\sqrt{\int Q(\boldsymbol{z},\boldsymbol{\alpha})\,\mathrm{d}\,F(\boldsymbol{z})}} > \varepsilon\right\} \leq 4e^{\left(\frac{H_{\mathrm{ann}}^{\mathcal{A}}(2N)}{N}-\frac{\varepsilon^2}{4}\right)N} \tag{2.30}$$

These two bounds are distribution-dependent since the given distribution function $F(\boldsymbol{z})$ was used for constructing the annealed VC entropy $H_{\mathrm{ann}}^{\mathcal{A}}(N)$. Recalling inequality (2.26), the following two bounds on the rate of convergence for any distribution function $F(\boldsymbol{z})$ can very simply be derived:

$$P\left\{\sup_{\boldsymbol{\alpha}\in\mathcal{A}}\left|\int Q(\boldsymbol{z},\boldsymbol{\alpha})\,\mathrm{d}\,F(\boldsymbol{z}) - \frac{1}{N}\sum_{n=1}^{N} Q(\boldsymbol{z}_n,\boldsymbol{\alpha})\right| > \varepsilon\right\} \leq 4e^{\left(\frac{G^{\mathcal{A}}(2N)}{N}-\varepsilon^2\right)N} \tag{2.31}$$

$$P\left\{\sup_{\boldsymbol{\alpha}\in\mathcal{A}}\frac{\int Q(\boldsymbol{z},\boldsymbol{\alpha})\,\mathrm{d}\,F(\boldsymbol{z}) - \frac{1}{N}\sum_{n=1}^{N} Q(\boldsymbol{z}_n,\boldsymbol{\alpha})}{\sqrt{\int Q(\boldsymbol{z},\boldsymbol{\alpha})\,\mathrm{d}\,F(\boldsymbol{z})}} > \varepsilon\right\} \leq 4e^{\left(\frac{G^{\mathcal{A}}(2N)}{N}-\frac{\varepsilon^2}{4}\right)N} \tag{2.32}$$

2.4.2 Bounds on the generalization ability of a learning machine

We can now derive bounds on the generalization performance of a learning machine by rearranging the bounds on the rate of convergence. Using the notation

$$\varepsilon = 4\frac{G^{\mathcal{A}}(2N) - \ln(\eta/4)}{N} \tag{2.33}$$

we can find from (2.32) that the inequality

$$R(\boldsymbol{\alpha}) \leq R_{\mathrm{emp}}(\boldsymbol{\alpha}) + \frac{\varepsilon}{2}\left(1 + \sqrt{1 + \frac{4R_{\mathrm{emp}}(\boldsymbol{\alpha})}{\varepsilon}}\right) \tag{2.34}$$

holds with probability at least $1-\eta$ for all functions $Q(\boldsymbol{z}, \boldsymbol{\alpha})$, $\boldsymbol{\alpha} \in \mathcal{A}$, (including the one that minimizes R_{emp}). It was also found that the inequality

$$R(\boldsymbol{\alpha}_N) - \inf_{\boldsymbol{\alpha} \in \mathcal{A}} R(\boldsymbol{\alpha}) \leq \sqrt{\frac{-\ln \eta}{2N}} + \frac{\varepsilon}{2}\left(1 + \sqrt{1 + \frac{4}{\varepsilon}}\right) \tag{2.35}$$

holds with probability at least $1-2\eta$ for the function $Q(\boldsymbol{z}, \boldsymbol{\alpha}_N)$ minimizing the empirical risk.

However, the bounds (2.34) and (2.35) should be regarded as conceptual. It would be necessary to find a way to evaluate the growth function $G^{\mathcal{A}}(2N)$ for the given set of functions $Q(\boldsymbol{z}, \boldsymbol{\alpha})$, $\boldsymbol{\alpha} \in \mathcal{A}$. Constructive bounds using the *VC dimension* of a set of functions $Q(\boldsymbol{z}, \boldsymbol{\alpha})$, $\boldsymbol{\alpha} \in \mathcal{A}$, were developed instead [Vapnik, 2000].

2.4.3 The VC dimension of a set of functions

In order to define the VC dimension, we will first examine properties of the growth function used in obtaining generalization bounds for learning machines.

Theorem 6. *Any growth function either satisfies*

$$G^{\mathcal{A}}(N) = N \ln 2 \tag{2.36}$$

or is bounded by

$$G^{\mathcal{A}}(N) \leq h\left(\ln \frac{N}{h} + 1\right) \tag{2.37}$$

where h is an integer such that for $N = h$

$$G^{\mathcal{A}}(h) = h \ln 2 \tag{2.38}$$

$$G^{\mathcal{A}}(h) < (h+1) \ln 2 \tag{2.39}$$

To speak illustratively, the growth function is either linear (2.36) or bounded by a logarithmic function (2.37).

Using theorem 6, we can define the VC dimension.

Definition 3. *The VC dimension of a set of functions $Q(\boldsymbol{z}, \boldsymbol{\alpha})$, $\boldsymbol{\alpha} \in \mathcal{A}$, is infinite if the growth function for this set of functions is linear.*

The VC dimension of a set of functions $Q(\boldsymbol{z}, \boldsymbol{\alpha})$, $\boldsymbol{\alpha} \in \mathcal{A}$, is finite and equal to h if the corresponding growth function is bounded by a logarithmic function with coefficient h.

Using this definition, inequality (2.26) can be extended to

$$H^{\mathcal{A}}(N) \leq H^{\mathcal{A}}_{\text{ann}}(N) \leq G^{\mathcal{A}}(N) \leq \frac{h\left(\ln \frac{N}{h} + 1\right)}{N} \quad (l > h) \tag{2.40}$$

A finite VC dimension is, therefore, a sufficient condition for consistency of the ERM method applied to a learning machine implementing a set of indicator functions, independently of the probability measure $F(\boldsymbol{z})$. Moreover, a finite VC dimension also guarantees a fast rate of convergence.

Geometrically speaking one can say that the VC dimension of a set of indicator functions $Q(\boldsymbol{z}, \boldsymbol{\alpha})$, $\boldsymbol{\alpha} \in \mathcal{A}$, is the maximum number h of vectors that can be separated into two classes in all 2^h possible ways using functions from the set. The VC dimension is infinite if for any n there exists a set of n vectors that can be shattered by functions from the set.

Example 1. *Consider a set of* linear *indicator functions*

$$Q(\boldsymbol{z}, \boldsymbol{\alpha}) = \Theta\left\{\boldsymbol{\alpha}^\top \boldsymbol{z} + \alpha_0\right\}$$

in an Ddimensional coordinate space. Then the VC dimension of this set of functions is $h = D+1$, i. e., one can separate at most $D+1$ vectors using this set of functions.

Example 2. *Consider the set of functions*

$$f(x, \alpha) = \theta(\sin \alpha x), \quad \alpha \in \mathbb{R}.$$

The VC dimension of this set of functions is infinite since by choosing an appropriate parameter α one can approximate values of any function bounded by $[-1, 1]$ using $\sin \alpha x$.

From the examples we conclude that the VC dimension of a set of functions is not necessarily related to the number of parameters, though it is in the case of linear indicator functions.

2.4.4 Constructive distribution-independent bounds

If one considers functions with finite VC dimension h, with (2.37) from theorem 6 we can use the expression

$$\varepsilon = 4\frac{h\left(\ln \frac{2N}{h} + 1\right) - \ln \frac{\eta}{4}}{N} \tag{2.41}$$

for equations (2.34) and (2.35) if $N > h$. In the special case where the set of functions $Q(\boldsymbol{z}, \boldsymbol{\alpha})$, $\boldsymbol{\alpha} \in \mathcal{A}$, contains a finite number of A elements, the following expression can be used instead [Vapnik, 1998]:

$$\varepsilon = 2\frac{\ln A - \ln \eta}{N} \tag{2.42}$$

2.5 Controlling the generalization ability of a learning machine

So far, we have only been concerned with the ERM inductive principle. This inductive principle is especially well suited to the case of a large sample size N. This becomes clear when evaluating (2.34) and (2.35) with expressions (2.41) or (2.42). When N/h is large, ε tends to zero, which makes the second summand on the right-hand side of inequality (2.34), the term associated with the capacity of the learning machine, (respectively the second summand on the right-hand side of (2.35)) diminish. The actual risk is then close to the empirical risk; in turn, a small empirical risk guarantees a small value of the (expected) risk.

But if N/h is small (usually one considers $N/h < 20$), a small $R_{\text{emp}}(\boldsymbol{\alpha}_N)$ does not guarantee a small actual risk. This means that in order to minimize the actual risk $R(\boldsymbol{\alpha})$ the right-hand side of inequality (2.34) (or, respectively, (2.35)) must be minimized over both the empirical risk and the machine capacity term simultaneously. Consequently, the VC dimension of the set of functions $Q(\boldsymbol{z}, \boldsymbol{\alpha})$, $\boldsymbol{\alpha} \in \mathcal{A}$, must be made a controlling variable. This is achieved by SRM.

Let us consider the set $\mathcal{S}$ of functions $Q(\boldsymbol{z}, \boldsymbol{\alpha})$, $\boldsymbol{\alpha} \in \mathcal{A}$, with a *structure* of nested subsets $\mathcal{S}_k = \{Q(\boldsymbol{z}, \boldsymbol{\alpha}),\ \boldsymbol{\alpha} \in \mathcal{A}_k\}$ such that

$$\mathcal{S}_1 \subset \mathcal{S}_2 \subset \cdots \subset \mathcal{S}_n \cdots \tag{2.43}$$

where the VC dimension h_k of each set $\mathcal{S}_k$ is finite, i. e.,

$$h_1 \leq h_2 \leq \cdots \leq h_n \ldots \tag{2.44}$$

Given a set of observations $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N$, the SRM inductive principle selects the function $Q(\boldsymbol{z}, \boldsymbol{\alpha}_N^k)$ which minimizes the empirical risk in the subset $\mathcal{S}_k$ for which the guaranteed risk, as given by the right-hand side of (2.34), is minimal, i. e., the SRM principle weighs quality of approximation against complexity of the approximating function.

2.5.1 Examples

In general, the SRM principle can be applied in many different ways. Conventional feed-forward neural networks are excellent examples of this. For instance, a neural network where the number of units in the hidden layer are increased monotonically defines a structure on the sets of implementable functions. Another example can be found by considering neural networks where the weight-decay procedure is applied at training time. By continuously decreasing the maximum magnitude of the weight vectors a structure is defined by the sets of implementable functions. In chapter 4, further examples are given, as appropriate.

2.5.2 Regularization as a means of controlling the capacity of a learning machine

Various other approaches at improving the performance of methods in mathematics and statistics have led to essentially the same idea as the SRM principle. One very famous example is the method of regularization [Tikhonov, 1963]. Consider the problem of finding a quasi-solution of the linear operator equation

$$\mathrm{A}f = F, \quad f \in \mathcal{M}, \tag{2.45}$$

when solving ill-posed problems. The linear operator A maps elements from the metric space $\mathcal{M} \subset \mathcal{E}_1$ with metric $\rho_{\mathcal{E}_1}$ to elements of the metric space $\mathcal{N} \subset \mathcal{E}_2$ with metric $\rho_{\mathcal{E}_2}$. Thikonov introduced a nonnegative semi-continuous functional $\Omega(f)$ with the following properties:

1. The domain of $\Omega(f)$ is $\mathcal{M}$,
2. the region for which

$$\mathcal{M}_j = \{f : \Omega(f) \leq d_j\}, \quad d_j > 0, \tag{2.46}$$

   holds is a compactum in the metric space $\mathcal{E}_1$,
3. The solution of (2.45) belongs to some $\mathcal{M}_i^*$ such that

$$\Omega(f) \leq d^* < \infty. \tag{2.47}$$

Thikonov proved that the functions f_γ minimizing the functionals

$$\Phi_\gamma(f) = \rho^2_{\mathcal{E}_2}(Af, F) + \gamma\Omega(f) \tag{2.48}$$

converge to the desired function as γ converges to zero. The proofs did not explicitly target capacity control. However, since any subset as in (2.46) is compact, it has bounded capacity. Therefore, a structure on the subsets M_j is described as in section 2.5.

Similar procedures and proofs were found by Ivanov [Ivanov, 1962] and Phillips [Phillips, 1962].

2.6 Summary

In this chapter, important points of Statistical Learning Theory have been revisited. Conditions for asymptotic convergence (consistency) as well as bounds on the rate have been established. Using these bounds, the structural risk minimization learning paradigm, which is especially useful in the case of small sample sizes, has been motivated, followed by examples of how to implement the SRM procedure. These important foundations will be used in the following chapters to analyze and improve procedures for learning probability density function models.

3 Pattern Recognition Machines implementing Structural Risk Minimization

In this chapter I will revisit two learning machines especially suited for pattern recognition where samples may belong to one of two possible labels. The extension to more than two labels (multi-label classification) will be discussed in section 6.2.

From section 2.5 it is known that in order to achieve a good learning result one has to minimize both the empirical risk and the confidence interval. One approach is to keep the VC dimension (and, therefore, the confidence interval) of a learning machine fixed and to then minimize the empirical risk achieved by this machine. A typical example of this approach are conventional Multi-Layer Perceptrons (MLPs) or ASR training procedures which require a-priori knowledge of the problem and the complexity of the learning machine.

A different approach is to keep the empirical risk fixed at a (small) value and to then minimize the confidence interval, i. e., the complexity of the machine. In this chapter, two such learning machines will be revisited, i. e., the SVM and the KFD.

3.1 Support Vector Machines

3.1.1 The optimal separating hyperplane

Consider a set of (training) samples $\mathcal{Z} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, $\boldsymbol{x} \in \mathbb{R}^D$, $y \in \{-1, 1\}$, which we will write in a more convenient matrix-vector notation as $\boldsymbol{X} = [\boldsymbol{x}_{(1)}, \ldots, \boldsymbol{x}_{(N)}]$ and $\boldsymbol{y} = [y_{(1)}, \ldots, y_{(N)}]^\top$. Suppose the data can be separated by a hyperplane

$$\boldsymbol{w}^\top \boldsymbol{x} - b = 0 \tag{3.1}$$

The *optimal separating hyperplane* $(\hat{\boldsymbol{w}}, \hat{b})$ separates the data without error while keeping the distance between the hyperplane and the closest vector maximal. In other words, the optimal separating hyperplane satisfies

$$\boldsymbol{w}^\top \boldsymbol{x}_{(n)} - b \geq 1, \qquad \text{if} \quad y_{(n)} = 1, \tag{3.2}$$

$$\boldsymbol{w}^\top \boldsymbol{x}_{(n)} - b \leq -1, \qquad \text{if} \quad y_{(n)} = -1, \tag{3.3}$$

or, more compactly,

$$y_{(n)}(\boldsymbol{w}^\top \boldsymbol{x}_{(n)} - b) \geq 1, \quad n = 1, \ldots, N, \tag{3.4}$$

while solving the primal optimization problem

$$\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2, \tag{3.5}$$

which leads to a quadratic optimization problem.

The solution to this optimization problem is given by the saddle point of the Lagrangian

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} - \sum_{n=1}^{N} \alpha_{(n)} \left[(\boldsymbol{w}^\top \boldsymbol{x}_{(n)} - b) y_{(n)} - 1\right] \tag{3.6}$$

with $\boldsymbol{\alpha}$ the Lagrangian multipliers. The Lagrangian of the optimization problem must be minimized with respect to $\boldsymbol{w}$ and b and maximized with respect to $\boldsymbol{\alpha} \geq \mathbf{0}$. This leads to the following Karush-Kuhn-Tucker (KKT) conditions of optimality:

$$\frac{\partial L(\hat{\boldsymbol{w}}, \hat{b}, \hat{\boldsymbol{\alpha}})}{\partial \boldsymbol{w}} = \hat{\boldsymbol{w}} - \sum_{n=1}^{N} y_{(n)} \hat{\alpha}_{(n)} \boldsymbol{x}_{(n)} = \mathbf{0} \tag{3.7}$$

$$\frac{\partial L(\hat{\boldsymbol{w}}, \hat{b}, \hat{\boldsymbol{\alpha}})}{\partial b} = \hat{\boldsymbol{w}}^\top \boldsymbol{y} = 0 \tag{3.8}$$

$$\boldsymbol{\alpha} \geq \mathbf{0} \tag{3.9}$$

$$\hat{\alpha}_{(n)}[(\hat{\boldsymbol{w}}^\top \boldsymbol{x}_{(n)} - \hat{b}) y_{(n)} - 1] = 0 \qquad n = 1, \ldots, N. \tag{3.10}$$

By inspecting (3.7) we conclude that the optimal separating hyperplane has an expansion in (or, in other words, is a linear combination of) the vectors of the training set. Furthermore, according to (3.9), only some coefficient $\hat{\alpha}_{(n)}$ are nonzero; the corresponding vectors $\boldsymbol{x}_{(n)}$ are called *support vectors*. The sparse solution is a result of the SRM inductive principle employed for SVM training, i. e., the empirical risk is minimized (zero in this case), while the complexity (the VC dimension of the SVM) is also minimized by including only the strictly necessary samples, i. e., the samples *supporting* the optimal separating hyperplane, in the solution. However, we cannot *expect* the SVM to be a sparse model; depending on the problem, the SVM solution may have as little as 10% sparsity [Mika et al., 1999].

Substituting (3.7) and (3.10) into the Lagrangian (3.6) leads to the dual optimization problem

$$\max_{\boldsymbol{\alpha}} \mathbf{1}^\top \boldsymbol{\alpha} - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_{(n)} \alpha_{(m)} y_{(n)} y_{(m)} \boldsymbol{x}_{(n)}^\top \boldsymbol{x}_{(m)} \tag{3.11}$$

which needs to be solved subject to constraints (3.8) and (3.9).

3.1.2 The non-separable case

If the data are not linearly separable, an optimal hyperplane (which is the hyperplane committing as little as possible error) may be constructed by introducing non-negative *slack variables* $\boldsymbol{\xi}$. The generalized optimal hyperplane [Cortes and Vapnik, 1995] is constructed by solving the optimization problem

$$\min_{\boldsymbol{w},\boldsymbol{\xi}} \frac{1}{2}\boldsymbol{w}^\top\boldsymbol{w} + C\mathbf{1}^\top\boldsymbol{\xi} \tag{3.12}$$

subject to

$$y_{(n)}(\boldsymbol{w}^\top\boldsymbol{x}_{(n)} - b) \geq 1 - \xi_{(n)} \qquad n = 1,\ldots,N \tag{3.13}$$

$$\boldsymbol{\xi} \geq \mathbf{0} \tag{3.14}$$

where $C > 0$ is a given parameter determining the trade-off between the capacity (first term in (3.12)) and the accuracy (second term in (3.12)) of the model.

As before, the Lagrangian of (3.12) is constructed by adding constraints (3.13) and (3.14) with corresponding coefficients $\boldsymbol{\alpha}$, $\boldsymbol{\nu}$

$$L(\boldsymbol{w},\boldsymbol{\xi},b,\boldsymbol{\alpha},\boldsymbol{\nu}) = \frac{1}{2}\boldsymbol{w}^\top\boldsymbol{w} + C\mathbf{1}^\top\boldsymbol{\xi} - \sum_{n=1}^{N}\alpha_{(n)}[(\boldsymbol{w}^\top\boldsymbol{x}_{(n)} - b)y_{(n)} - 1 + \xi_{(n)}] - \boldsymbol{\nu}^\top\boldsymbol{\xi} \tag{3.15}$$

whose optimizing saddle-point is obtained by minimizing with respect to $\boldsymbol{w}$, $\boldsymbol{\xi}$ and b and simultaneously maximizing with respect to $\boldsymbol{\alpha}$ and $\boldsymbol{\nu}$. Before continuing the discussion, let us rewrite (3.15) using the short-hand notation $\boldsymbol{s} : s_{(n)} = \alpha_{(n)}y_{(n)}$:

$$L(\boldsymbol{w},\boldsymbol{\xi},b,\boldsymbol{\alpha},\boldsymbol{\nu}) = \frac{1}{2}\boldsymbol{w}^\top\boldsymbol{w} + C\mathbf{1}^\top\boldsymbol{\xi} - \boldsymbol{s}^\top\boldsymbol{X}^\top\boldsymbol{w} + b\mathbf{1}^\top\boldsymbol{s} + \mathbf{1}^\top\boldsymbol{\alpha} - \boldsymbol{\xi}^\top\boldsymbol{\alpha} - \boldsymbol{\nu}^\top\boldsymbol{\xi} \tag{3.16}$$

Thus, the corresponding KKT conditions are

$$\frac{\partial L(\hat{\boldsymbol{w}},\hat{\boldsymbol{\xi}},\hat{b},\hat{\boldsymbol{\alpha}},\hat{\boldsymbol{\nu}})}{\partial \boldsymbol{w}} = \hat{\boldsymbol{w}} - \boldsymbol{X}^\top\hat{\boldsymbol{s}} = \mathbf{0} \tag{3.17}$$

$$\frac{\partial L(\hat{\boldsymbol{w}},\hat{\boldsymbol{\xi}},\hat{b},\hat{\boldsymbol{\alpha}},\hat{\boldsymbol{\nu}})}{\partial \boldsymbol{\xi}} = C\mathbf{1} - \hat{\boldsymbol{\alpha}} - \hat{\boldsymbol{\nu}} = \mathbf{0} \tag{3.18}$$

$$\frac{\partial L(\hat{\boldsymbol{w}},\hat{\boldsymbol{\xi}},\hat{b},\hat{\boldsymbol{\alpha}},\hat{\boldsymbol{\nu}})}{\partial b} = \hat{\boldsymbol{w}}^\top\boldsymbol{y} = 0 \tag{3.19}$$

$$\hat{\alpha}_{(n)}[(\hat{\boldsymbol{w}}^\top\boldsymbol{x}_{(n)} - \hat{b})y_{(n)} - 1 + \xi_{(n)}] = 0 \qquad n = 1,\ldots,N \tag{3.20}$$

$$\boldsymbol{\alpha} \geq \mathbf{0} \tag{3.21}$$

$$\boldsymbol{\nu} \geq \mathbf{0} \tag{3.22}$$

From (3.18), (3.21) and (3.22) we can conclude that

$$\mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1} \tag{3.23}$$

By substituting (3.17), (3.18) and (3.20) into the Lagrangian (3.15) we arrive at the same dual optimization problem as in the separable case (cf. section 3.1.1)

$$\max_{\boldsymbol{\alpha}} \mathbf{1}^\top \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{s}^\top \boldsymbol{X}^\top \boldsymbol{X} \boldsymbol{s} \tag{3.24}$$

only with different constraint (3.23).

3.2 Adding Kernels

The learning machine introduced in section 3.1.2 is also called the *linear support-vector machine* because it applies the linear inner product of the samples $\boldsymbol{x}$ in the matrix $\boldsymbol{X}^\top \boldsymbol{X}$ in (3.24). As a way of solving the non-separability of the data in its *input space* $\mathbb{R}^D$ it may be possible to use a non-linear mapping $\boldsymbol{\phi} : \mathbb{R}^D \to \mathcal{F}$ where $\mathcal{F}$ is the so-called *feature space* with large, and possibly infinite, dimensionality F.

Now the separation problem (3.12) is to be solved in the feature space $\mathcal{F}$. The resulting generalized optimal hyperplane is then given by $(\hat{\boldsymbol{w}}_{[F]}, b)$:

$$\min_{\boldsymbol{w},\boldsymbol{\xi}} \frac{1}{2} \boldsymbol{w}^\top \boldsymbol{w} + C \mathbf{1}^\top \boldsymbol{\xi} \tag{3.25}$$

subject to

$$\boldsymbol{y}^\top (\boldsymbol{\Phi}^\top(\boldsymbol{X}) \boldsymbol{w} - b\mathbf{1}) \geq \mathbf{1} - \boldsymbol{\xi} \tag{3.26}$$

$$\boldsymbol{\xi} \geq \mathbf{0} \tag{3.27}$$

with $\boldsymbol{\Phi}(\boldsymbol{X}) = [\boldsymbol{\phi}(\boldsymbol{x}_{(1)}), \ldots, \boldsymbol{\phi}(\boldsymbol{x}_{(N)})]$. Now, similar in fashion to section 3.1.2 the Lagrangian of the optimization problem is constructed:

$$L(\boldsymbol{w}, \boldsymbol{\xi}, b, \boldsymbol{\alpha}, \boldsymbol{\nu}) = \frac{1}{2} \boldsymbol{w}^\top \boldsymbol{w} + C \mathbf{1}^\top \boldsymbol{\xi} - \boldsymbol{s}^\top \boldsymbol{\Phi}^\top(\boldsymbol{X}) \boldsymbol{w} + b \mathbf{1}^\top \boldsymbol{s} + \mathbf{1}^\top \boldsymbol{\alpha} - \boldsymbol{\xi}^\top \boldsymbol{\alpha} - \boldsymbol{\nu}^\top \boldsymbol{\xi} \tag{3.28}$$

The KKT conditions of optimality are:

$$\frac{\partial L(\hat{\boldsymbol{w}}, \hat{\boldsymbol{\xi}}, \hat{b}, \hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\nu}})}{\partial \boldsymbol{w}} = \hat{\boldsymbol{w}} - \boldsymbol{\Phi}(\boldsymbol{X}) \hat{\boldsymbol{s}} = \mathbf{0} \tag{3.29}$$

$$\frac{\partial L(\hat{\boldsymbol{w}}, \hat{\boldsymbol{\xi}}, \hat{b}, \hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\nu}})}{\partial \boldsymbol{\xi}} = C\mathbf{1} - \hat{\boldsymbol{\alpha}} - \hat{\boldsymbol{\nu}} = \mathbf{0} \tag{3.30}$$

$$\frac{\partial L(\hat{\boldsymbol{w}}, \hat{\boldsymbol{\xi}}, \hat{b}, \hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\nu}})}{\partial b} = \hat{\boldsymbol{w}}^\top \boldsymbol{y} = 0 \tag{3.31}$$

$$\hat{\alpha}_{(n)} \left[(\hat{\boldsymbol{w}}^\top \boldsymbol{\phi}(\boldsymbol{x}_{(n)}) - \hat{b}) y_{(n)} - 1 + \xi_{(n)} \right] = 0 \qquad n = 1, \ldots, N \tag{3.32}$$

$$\boldsymbol{\alpha} \geq \mathbf{0} \tag{3.33}$$

$$\boldsymbol{\nu} \geq \mathbf{0} \tag{3.34}$$

Note that (3.29) is also known as the Representer Theorem [Aronszajn, 1950]. Substituting (3.29) and (3.30) into the Lagrangian (3.28) leads to the dual optimization problem

$$\max_{\boldsymbol{\alpha}} \mathbf{1}^\top \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{s}^\top \boldsymbol{\Phi}^\top(\boldsymbol{X}) \boldsymbol{\Phi}(\boldsymbol{X}) \boldsymbol{s} \tag{3.35}$$

subject to

$$\mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1} \tag{3.36}$$

The decision rule is now also constructed in the feature space; by adapting (3.1) with (3.29) we find that the SVM predicts the label y of a sample $\boldsymbol{x}$ as

$$y = \operatorname{sgn}(f(\boldsymbol{x})) = \operatorname{sgn}(\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}) - b) = \operatorname{sgn}(\hat{\boldsymbol{s}}^\top \boldsymbol{\Phi}^\top(\boldsymbol{X}) \boldsymbol{\phi}(\boldsymbol{x}) - b) \tag{3.37}$$

3.2.1 The Kernel trick

In the previous section we basically substituted the scalar product of the samples in the original input space with scalar product of the samples' projections in the feature space. Except for the hope that a linear decision rule may be more easily constructed in the richer feature space, not much was won because the mapping $\boldsymbol{\phi}$ needs to computed which may be very computationally intractable.

If the feature space $\mathcal{F}$ is a reproducing kernel Hilbert space (RKHS) [Aronszajn, 1950], there exists for certain mappings $\boldsymbol{\phi}(\boldsymbol{x})$ a function $k(\boldsymbol{x}, \boldsymbol{y})$ such that

$$k(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{\phi}^\top(\boldsymbol{x}) \boldsymbol{\phi}(\boldsymbol{y}) \tag{3.38}$$

This means that the scalar products of the samples' projections in feature space can be computed directly without having to compute the mapping $\boldsymbol{\phi}(\boldsymbol{x})$ directly; in fact, the mapping does not even need to be known.

The function from (3.38) is called a *kernel function* which must satisfy the following conditions [Mercer, 1909, Schölkopf et al., 1999]:

Theorem 7 (Mercer). *If k is a continuous symmetric kernel of a positive integral operator T, i. e.,*

$$(Tf)(\boldsymbol{y}) = \int_{\mathcal{C}} k(\boldsymbol{x}, \boldsymbol{y}) f(\boldsymbol{x}) \,\mathrm{d}\,\boldsymbol{x} \tag{3.39}$$

with

$$\iint_{\mathcal{C} \times \mathcal{C}} k(\boldsymbol{x}, \boldsymbol{y}) f(\boldsymbol{x}) f(\boldsymbol{y}) \,\mathrm{d}\,\boldsymbol{x} \,\mathrm{d}\,\boldsymbol{y} \geq 0 \tag{3.40}$$

for all $f \in L_2(\mathcal{C})$ with $\mathcal{C}$ a compact subset of $\mathbb{R}^D$, it can be expanded in uniformly and absolutely convergent series in terms of T's eigenfunctions ψ_j and positive eigenvalues λ_j:

$$k(\boldsymbol{x}, \boldsymbol{y}) = \sum_{j=1}^{\infty} \lambda_j \psi_j(\boldsymbol{x}) \psi_j(\boldsymbol{y}) \tag{3.41}$$

Mercer kernels give rise to a positive symmetric matrix $\boldsymbol{K}_{[N\times N]}$ with $K_{(i,j)} = k(\boldsymbol{x}_{(i)}, \boldsymbol{x}_{(j)})$, which we will from now on call the *kernel matrix*. The kernel matrix is a Gram matrix, i. e., the following holds:

$$\boldsymbol{a}^\top \boldsymbol{K}\boldsymbol{a} = \left\| \sum_{n=1}^{N} a_{(n)} \boldsymbol{\phi}(\boldsymbol{x}_{(n)}) \right\|^2 \geq 0 \quad \forall \boldsymbol{a} \in \mathbb{R}^N \tag{3.42}$$

Popular examples of Mercer kernels are

- the polynomial kernel

$$k_{\text{poly}}(\boldsymbol{x}, \boldsymbol{y}) = (a\boldsymbol{x}^\top \boldsymbol{y} + b)^c \tag{3.43}$$

  with parameters $a, b \in \mathbb{R}$ and $c \in \mathbb{N}$,

- the Gaussian kernel

$$k_{\text{gauss}}(\boldsymbol{x}, \boldsymbol{y}) = e^{\frac{-\|\boldsymbol{x}-\boldsymbol{y}\|^2}{2\sigma^2}}, \quad \sigma \in \mathbb{R}^+ \tag{3.44}$$

- the sigmoid kernel

$$k_{\text{sigm}}(\boldsymbol{x}, \boldsymbol{y}) = \tanh(a\boldsymbol{x}^\top \boldsymbol{y} + b) \tag{3.45}$$

  for suitable parameters $a, b \in \mathbb{R}$.

With this in mind we can rewrite the dual optimization problem (3.35) in a computationally more tractable form as

$$\max_{\boldsymbol{\alpha}} \mathbf{1}^\top \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{s}^\top \boldsymbol{K}\boldsymbol{s} \tag{3.46}$$

subject to (3.36). Also, with $\boldsymbol{k}_{[N]}, k_{(n)} = k(\boldsymbol{x}, \boldsymbol{x}_{(n)})$, we can rewrite the prediction rule (3.37) as

$$y = \text{sgn}(f(\boldsymbol{x})) = \text{sgn}(\hat{\boldsymbol{s}}^\top \boldsymbol{k} - b) \tag{3.47}$$

3.3 Solving the Quadratic Programming problem

The SVM training formulation is quite appealing because it implements the SRM inductive principle. Also, since the design matrix $\boldsymbol{K}$ is positive semi-definite and the constraints are linear, the resulting Quadratic Programming (QP) problem is convex which in turn means that any solution found by the optimization procedure is also a global minimum. (Compare this with, e. g., the traditional back-propagation training of feed-forward neural networks which leads to local minima only.) The QP problem can be solved using the Bunch-Kaufman-algorithm [Bunch and Kaufman, 1977] or interior point methods such as *LOQO* [Vanderbei, 1999]; general properties of QP problems and related algorithms are discussed in [Boyd and Vandenberghe, 2004] or [Fletcher, 1987].

3.4 Reducing the memory requirements of SVM training

The SVM training formulation has advantages and disadvantages. Among the advantages we find the structural risk minimization inductive principle and the fact that the resulting optimization problem is convex due to the fact that the design matrix $\boldsymbol{K}$ is positive semi-definite. However, exactly this matrix is the disadvantage of the optimization problem as the size of the matrix is $N \times N$ which may for real-world problems lead to matrices that do not fit into a computer's memory. Therefore, considerable research effort has been put into making SVM training on real-world problems possible, which resulted in exact and heuristic-approximate solutions of which a few will be reviewed in the next sections.

3.4.1 Chunking

One of the first exact SVM training procedures involving reduced optimization problems was published in [Boser et al., 1992] and became known under the name "chunking" whose main idea is summarized in algorithm 1.

Algorithm 1 Chunking SVM training procedure

choose a random subset $\mathcal{Z}_s$, initialize $\boldsymbol{\alpha} = \mathbf{0}$
repeat
  solve the optimization problem on $\mathcal{Z}_s$
  check for samples from $\mathcal{Z}_v \subseteq \mathcal{Z} \setminus \mathcal{Z}_s$ violating the optimality
  augment the reduced training set with the violating samples: $\mathcal{Z}_s = \mathcal{Z}_s \cup \mathcal{Z}_v$
until $\mathcal{Z}_v = \emptyset$

Thus, the name "chunking" results from the subsets $\mathcal{Z}_s$ on which training is carried out. In [Osuna et al., 1997b] it was formally proven that the algorithm converges to the same solution as the original non-chunking approach. The final training subset will contain all the support vectors and typically a number of non-support vectors, i. e., the chunking approach reduces the memory requirements at the expense of additional computation for repeatedly solving QP problems. Variations of the algorithm, e. g., those deleting non-support vectors from $\mathcal{Z}_s$, are also known. However, since the QP problems scale at least with the number of support vectors, even chunking may not be applicable to all data problems.

3.4.2 Decomposition

Instead of solving QP problems of the size of the number of support vectors, a strategy suggested in [Osuna et al., 1997a] *decomposes* the problem into a smaller

sub-problems where only some components from the solution $\boldsymbol{\alpha}$ and corresponding constraints are considered. The algorithm in its nature is similar to the active set strategy proposed by [Gill et al., 1993]. The basic procedure of the decomposition method is summarized in algorithm 2.

Algorithm 2 Decomposition SVM training procedure

while optimality conditions are violated **do**
  split the training set $\mathcal{Z}$ into sets $\mathcal{Z}_B$ corresponding to b free variables and $\mathcal{Z}_N$ corresponding to $l = N - b$ fixed variables
  decompose the QP problem (3.35) and solve it on $\mathcal{Z}_B$
end while

Assuming properly arranged matrices and vectors

$$\boldsymbol{\alpha} = \begin{pmatrix} \boldsymbol{\alpha}_B \\ \boldsymbol{\alpha}_N \end{pmatrix} \quad \boldsymbol{y} = \begin{pmatrix} \boldsymbol{y}_B \\ \boldsymbol{y}_N \end{pmatrix} \quad \boldsymbol{K} = \begin{pmatrix} \boldsymbol{K}_{BB} & \boldsymbol{K}_{BN} \\ \boldsymbol{K}_{NB} & \boldsymbol{K}_{NN} \end{pmatrix} \tag{3.48}$$

and observing that $\boldsymbol{K}_{BN} = \boldsymbol{K}_{NB}^\top$ due to symmetricity of $\boldsymbol{K}$, we can write the optimization problem as

$$\max_{\boldsymbol{\alpha}} = -\frac{1}{2}\boldsymbol{\alpha}_B^\top \boldsymbol{K}_{BB}\boldsymbol{\alpha}_B - \boldsymbol{\alpha}_B^\top \boldsymbol{K}_{BN}\boldsymbol{\alpha}_N - \frac{1}{2}\boldsymbol{\alpha}_N^\top \boldsymbol{K}_{NN}\boldsymbol{\alpha}_N + \mathbf{1}^\top \boldsymbol{\alpha}_B + \mathbf{1}^\top \boldsymbol{\alpha}_N \tag{3.49}$$

subject to

$$\boldsymbol{\alpha}_B^\top \boldsymbol{y}_B + \boldsymbol{\alpha}_N^\top \boldsymbol{y}_N = 0 \tag{3.50}$$

$$\mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1} \tag{3.51}$$

The terms $\frac{1}{2}\boldsymbol{\alpha}_N^\top \boldsymbol{K}_{NN}\boldsymbol{\alpha}_N$ and $\mathbf{1}^\top \boldsymbol{\alpha}_N$ are constant and can, therefore, be omitted without changing the solution of (3.49). The new, smaller optimization problem can now be solved using standard methods, e. g., those mentioned in section 3.3. Fast progress on the complete optimization problem depends on the selection of a good working set. Also, a number of additional measures must be taken to arrive at an efficient implementation, e. g., caching of parts of $\boldsymbol{K}$. The decomposition method is implemented, e. g., in *SVMLight* [Joachims, 1998].

3.4.3 Sequential Minimal Optimization

In [Platt, 1998] the decomposition method from section 3.4.2 is taken to the extreme by restricting the working set $\mathcal{Z}_B$ to its minimal size of only two variables which is due to the sum constraint (3.50). Therefore, this algorithm was given the name *Sequential Minimal Optimization (SMO)*. The same issues as in section 3.4.2 apply, i. e., strategy of working-set selection, caching of entries $\boldsymbol{K}_{(i,j)}$, caching of gradients,

and so on. However, the great advantage of this algorithm is that the resulting small optimization problem on $\mathcal{Z}_B$ can be solved analytically which results in a great speedup. The SMO procedure is implemented, e. g., in the *Torch* software library [Collobert and Bengio, 2001].

3.5 Kernel Fisher Discriminant

In section 3.1.2 and onwards, we considered the case of non-separable data, i. e., where the learning machine was allowed to commit (some) errors denoted by the slack variables $\boldsymbol{\xi}$, using the L_1-norm of the error vector in the primal optimization problem (3.12). Consider the following modifications: replace the inequality constraint in (3.26) with an equality constraint and use the L_2-norm of the error vector in the primal optimization problem:

$$\min_{\boldsymbol{\alpha},b} \frac{1}{2}\boldsymbol{\alpha}^\top \boldsymbol{K}\boldsymbol{\alpha} + C\boldsymbol{\xi}^\top\boldsymbol{\xi} \tag{3.52}$$

subject to

$$\boldsymbol{K}\boldsymbol{\alpha} + b\mathbf{1} = \boldsymbol{y} + \boldsymbol{\xi} \tag{3.53}$$

$$\boldsymbol{\alpha} \geq \mathbf{0} \tag{3.54}$$

This procedure may be regarded as a least-squares regression of the samples $\boldsymbol{X}$ onto their corresponding labels $\boldsymbol{y}$. The method became known as the Least Squares Support Vector Machine (LS-SVM) [Suykens et al., 2002].

So far, the introduction of slack variables to the SVM training procedure in section 3.1.2 and the substitution of constraints and error norm in this section had been motivated from a rather technical point of view. More formally, the choice of loss function (SVM: one-sided error with L_1-norm, KFD: two-sided error with L_2-norm) depends on the distribution of the classification error. If the exact error model was known a-priori, a suitable loss function could be chosen (including, but not limited to, the SVM or KFD loss functions) [Smola and Schölkopf, 2003]. In most cases, the error model is unknown, so a suitable loss function must be determined experimentally.

A method similar to the LS-SVM was derived by modifying the traditional *Fisher discriminant* [Fisher, 1936]. By substituting the linear scalar products with scalar products in a Hilbert feature space $\mathcal{F}$ using Mercer kernels (cf. section 3.2.1), the KFD was introduced [Mika, 2002]. The KFD's primal optimization problem is the following:

$$\min_{\boldsymbol{\alpha},b} \frac{1}{2}\boldsymbol{\xi}^\top\boldsymbol{\xi} + CR(\boldsymbol{\alpha}) \tag{3.55}$$

subject to

$$\boldsymbol{K}\boldsymbol{\alpha} + b\mathbf{1} = \boldsymbol{y} + \boldsymbol{\xi} \tag{3.56}$$

$$\mathbf{1}_{\pm}^{\top}\boldsymbol{K}_{\pm}\boldsymbol{\alpha} + b = \pm 1 \tag{3.57}$$

$$\boldsymbol{\alpha} \geq \mathbf{0} \tag{3.58}$$

where $\boldsymbol{K}_{\pm}$ are two matrices containing only the rows of $\boldsymbol{K}$ corresponding to positive and negative classes' samples, and $\mathbf{1}_{\pm}$ vectors of appropriate length containing only ones. $R(\boldsymbol{\alpha})$ is a properly chosen regularization functional. Typically, $R(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^{\top}\boldsymbol{K}\boldsymbol{\alpha}$ is used (as with LS-SVM or Vapnik's SVMs), but in [Mika, 2002] it was shown that one can also use the simpler form $R(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^{\top}\boldsymbol{\alpha}$. The major difference between LS-SVMs and KFDs are the two additional constraints (3.57) which force the average of the projections of individual classes' samples to equal its label.

In contrast to Vapnik's SVM, the LS-SVM does not produce a sparse model by itself, i. e., in general one cannot expect any component $\alpha_{(n)}$ to vanish. This increases the learning model's complexity and the required resources to predict the label y of a sample $\boldsymbol{x}$ via

$$y = \operatorname{sgn}(\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}_{(n)})^{\top}\boldsymbol{\alpha} + b) \tag{3.59}$$

It is, therefore, desirable to produce a sparse approximation of the complete KFD solution in order to achieve a lower complexity (VC dimension) and better generalization capability. The non-sparsity of the KFD solution should not, however, be interpreted as a disadvantage w. r. t. to the SVM solution as we mentioned that the SVM solution does not guarantee sparsity for all problems.

3.5.1 KFD Training Procedures

Greedy-Sparse Kernel Fisher Discriminant

As mentioned in section 3.5 the solutions $\boldsymbol{\alpha}$ of KFDs are not sparse, but a sparse approximation $\tilde{\boldsymbol{\alpha}}$ would be desirable to make the model more robust (by limiting its capacity) and to speed up prediction of test samples' labels. Technically, this means finding an optimal $\tilde{\boldsymbol{\alpha}}$ with at most M non-zero elements by using an L_0-regularizer or a non-linear constraint. This is impossible because there exist $\binom{N}{M}$ such solutions.

In [Mika et al., 2001] an algorithm was presented which efficiently computes a nearly optimal sparse approximation $\tilde{\boldsymbol{\alpha}}$. The procedure will be briefly revisited below.

First, we introduce the following shorthand notations:

$$\boldsymbol{a} = \begin{pmatrix} b \\ \boldsymbol{\alpha} \end{pmatrix} \tag{3.60}$$

$$\boldsymbol{c} = \begin{pmatrix} N_+ - N_- \\ \boldsymbol{K}^\top \boldsymbol{y} \end{pmatrix} \tag{3.61}$$

$$\boldsymbol{A}_\pm = \begin{pmatrix} N_\pm \\ \boldsymbol{K}^\top \mathbf{1}_\pm \end{pmatrix} \tag{3.62}$$

$$\boldsymbol{H} = \begin{pmatrix} N & \mathbf{1}^\top \boldsymbol{K} \\ \boldsymbol{K}^\top \mathbf{1} & \boldsymbol{K}^\top \boldsymbol{K} + C\boldsymbol{I} \end{pmatrix} \tag{3.63}$$

where $N_\pm$ denotes the number of samples in each label ± 1, respectively. The optimization problem can then be written as

$$\min_{\boldsymbol{a}} \frac{1}{2} \boldsymbol{a}^\top \boldsymbol{H} \boldsymbol{a} - \boldsymbol{c}^\top \boldsymbol{a} \tag{3.64}$$

subject to

$$\boldsymbol{A}_+^\top \boldsymbol{a} - N_+ = 0 \tag{3.65}$$

$$\boldsymbol{A}_-^\top \boldsymbol{a} - N_- = 0 \tag{3.66}$$

As usual, the constraints are added to the optimization problem with Lagrangian multipliers $\boldsymbol{\lambda}_\pm$:

$$L(\boldsymbol{a}, \lambda_+, \lambda_-) = \frac{1}{2} \boldsymbol{a}^\top \boldsymbol{H} \boldsymbol{a} - \boldsymbol{c}^\top \boldsymbol{a} + \lambda_+ (\boldsymbol{d}_+^\top \boldsymbol{a} - N_+) + \lambda_- (\boldsymbol{d}_-^\top \boldsymbol{a} - N_-) \tag{3.67}$$

The KKT conditions of optimality are obtained by taking the partial derivatives of the Lagrangian (3.67) with respect to the primal variables $\boldsymbol{a}$:

$$\frac{\partial L(\hat{\boldsymbol{a}}, \hat{\lambda}_+, \hat{\lambda}_-)}{\partial \boldsymbol{a}} = \boldsymbol{H} \hat{\boldsymbol{a}} - \boldsymbol{c} + \hat{\lambda}_+ \boldsymbol{d}_+ + \hat{\lambda}_- \boldsymbol{d}_- = \mathbf{0} \tag{3.68}$$

which may be solved for $\hat{\boldsymbol{a}}$:

$$\hat{\boldsymbol{a}} = \boldsymbol{H}^{-1} (c - (\hat{\lambda}_+ \boldsymbol{d}_+ + \hat{\lambda}_- \boldsymbol{d}_-)) \tag{3.69}$$

Substituting (3.69) into the Lagrangian (3.67) yields the dual optimization problem

$$\max_{\boldsymbol{\lambda}_\pm} -\frac{1}{2} \boldsymbol{\lambda}_\pm \begin{pmatrix} \boldsymbol{d}_+^\top \boldsymbol{H}^{-1} \boldsymbol{d}_+ & \boldsymbol{d}_+^\top \boldsymbol{H}^{-1} \boldsymbol{d}_- \\ \boldsymbol{d}_-^\top \boldsymbol{H}^{-1} \boldsymbol{d}_+ & \boldsymbol{d}_-^\top \boldsymbol{H}^{-1} \boldsymbol{d}_- \end{pmatrix} \boldsymbol{\lambda}_\pm + \begin{pmatrix} -N_+ + \boldsymbol{c}^\top \boldsymbol{H}^{-1} \boldsymbol{d}_+ \\ N_- + \boldsymbol{c}^\top \boldsymbol{H}^{-1} \boldsymbol{d}_- \end{pmatrix}^\top \boldsymbol{\lambda}_\pm - \frac{1}{2} \boldsymbol{c}^\top \boldsymbol{H}^{-1} \boldsymbol{c} \tag{3.70}$$

Since this is an unconstrained optimization problem it may be solved analytically by simply setting the derivative of (3.70) with respect to $\boldsymbol{\lambda}_{\pm}$ to zero:

$$-\begin{pmatrix} \boldsymbol{d}_{+}^{\top}\boldsymbol{H}^{-1}\boldsymbol{d}_{+} & \boldsymbol{d}_{+}^{\top}\boldsymbol{H}^{-1}\boldsymbol{d}_{-} \\ \boldsymbol{d}_{-}^{\top}\boldsymbol{H}^{-1}\boldsymbol{d}_{+} & \boldsymbol{d}_{-}^{\top}\boldsymbol{H}^{-1}\boldsymbol{d}_{-} \end{pmatrix} \hat{\boldsymbol{\lambda}}_{\pm} + \begin{pmatrix} -N_{+} + \boldsymbol{c}^{\top}\boldsymbol{H}^{-1}\boldsymbol{d}_{+} \\ N_{-} + \boldsymbol{c}^{\top}\boldsymbol{H}^{-1}\boldsymbol{d}_{-} \end{pmatrix} = \mathbf{0} \tag{3.71}$$

which is easily solved for $\hat{\boldsymbol{\lambda}}_{\pm}$.

Recall that one is actually interested in a sparse approximation $\tilde{\boldsymbol{\alpha}} \approx \hat{\boldsymbol{\alpha}}$ where most $\tilde{\alpha}_{(n)}$ are zero. If the $\tilde{\boldsymbol{\alpha}}$ is constructed in a forward-selection manner, i. e., the number of non-zero entries in $\tilde{\boldsymbol{\alpha}}$ is increased one at a time, the inverse matrix $\boldsymbol{H}^{-1}$ may be efficiently computed by making use of the Sherman-Morrison-Woodbury rank-one update formula [Horn and Johnson, 1985, Golub and van Loan, 1996] due to block-matrix structure of $\boldsymbol{H}$. The forward-selection procedure is stopped if the dual objective from (3.70) does not decrease significantly for a number of iterations.

To approximately and quickly solve the combinatorial problem mentioned above, [Mika et al., 2001] employs a probabilistic speed-up originally suggested in [Smola and Schölkopf, 2000] which, in essence, says that in order to find an approximation which with probability 0.95 is within the best 0.05 of all estimates, a random sample of $\kappa = 59$ is sufficient.

4 Automatic Speech Recognition revisited

In this chapter I will revisit properties of the speech recognition problem and its solution depending on various parameters (size of vocabulary, isolated vs. continuous speech, speaker-dependent vs. speaker-independent).

4.1 Human speech production

Humans use speech naturally to communicate a text carrying a certain meaning to another human (or a machine, which is our field of interest). The text is usually a concatenation of words governed by some structure (syntax) while the meaning of the text (semantics) is not only controlled by its constituting words but also by other factors such as prosody, emotion, and contextual information. With a certain intention in mind, the human brain instructs its speech production engine consisting of glottis, vocal tract, and lips to code this intention into acoustic waves. The communication partner has to capture these acoustic waves with his or her ears and to decode text and meaning from them.

The process of coding a text into acoustic waves is not fixed but rather subject to great variability. This variability is caused by varying fundamental frequencies, varying speed of the utterance, dialects, and "fillers" such as hesitations, coughs and so on.

Besides the acoustic variability of speech, the structure of the underlying text is also not fixed. In practice, the number of possible grammatically correct sentences is infinite, but unlike written text, spoken sentences often do not follow the correct grammatical syntax. However, humans use prior knowledge to anticipate certain words or phrases from the preceding context. This is something automatic speech recognition makes use of.

Thus, speech production can be regarded as a stochastic process. However, the process is relatively stationary within a few 10 ms because muscles limit the rate of change in the body parts involved in acoustic speech production [Fant, 1960].

4.2 Automatic speech recognition in general

In automatic speech recognition the task is to deduce the most likely spoken text (sequence of words) $\hat{\vec{w}}$ from a given acoustic utterance u. This can be regarded as a pattern recognition problem where a pattern (the acoustic utterance) is to be assigned to a certain class label, i. e., the text. Unless the domain of the task is very limited, e. g., single-word command recognition, the task cannot be accomplished by simple comparison of the utterance in question with prototype utterances. This becomes clear if we only consider an also very limited task of recognizing ten-digit numbers, such as US phone numbers, which would already result in 10^{10} possible hypotheses.

Instead, we use a divide-and-conquer approach to tackle the complexity of the problem. First, by making use of the short-term stationarity of acoustic speech, using methods of digital signal processing we extract short-term features from the properly digitizes speech. These features should capture properties of the speech which are relevant (useful) for speech recognition while discarding irrelevant, especially speaker-dependent, properties such as fundamental frequency. Common feature extraction procedure are Mel-frequence cepstral coefficients (MFCCs) or Perceptual Linear Prediction, often augmented by temporal context (first and second order temporal differences). Feature extraction is a field of research in its own; for further details, see, e. g., [Wendemuth, 2004], [Rabiner and Juang, 1993], and [Junqua and Haton, 1996]. The result of the feature extraction process is a stream $\vec{\boldsymbol{x}} = \langle \boldsymbol{x}_1; \boldsymbol{x}_2; \ldots; \boldsymbol{x}_T \rangle$ describing the original utterance u. We will use this sequence of feature vectors to make the recognition decision.

Instead of directly classifying the utterance u, we derive the most likely text $\hat{\vec{w}}$ using the Maximum-A-Posteriori method with Bayes' rule:

$$\hat{\vec{w}} = \operatorname*{argmax}_{\vec{w}} P(\vec{w}|\vec{\boldsymbol{x}}) = \operatorname*{argmax}_{\vec{w}} \frac{P(\vec{\boldsymbol{x}}|\vec{w})P(\vec{w})}{P(\vec{\boldsymbol{x}})} = \operatorname*{argmax}_{\vec{w}} P(\vec{\boldsymbol{x}}|\vec{w})P(\vec{w}) \tag{4.1}$$

In short, $P(\vec{\boldsymbol{x}}|\vec{w})$ is called the acoustic model which describes the coding of the intended text into acoustic speech, and $P(\vec{w})$ is called the language model describing the plausibility of a hypotheses text. Figure 4.1 gives a schematic overview of the architecture of an automatic speech recognition system.

In the following sections (4.1) will be refined for the case of large-vocabulary continuous speech recognition.

4.3 The language model

In (4.1), $P(\vec{w})$ models the probability of the word sequence $\vec{w}$. Since it is impossible to compute and store the probabilities for each complete text, approximations whose parameters may be estimated from sample data are used. Most commonly, m-gram

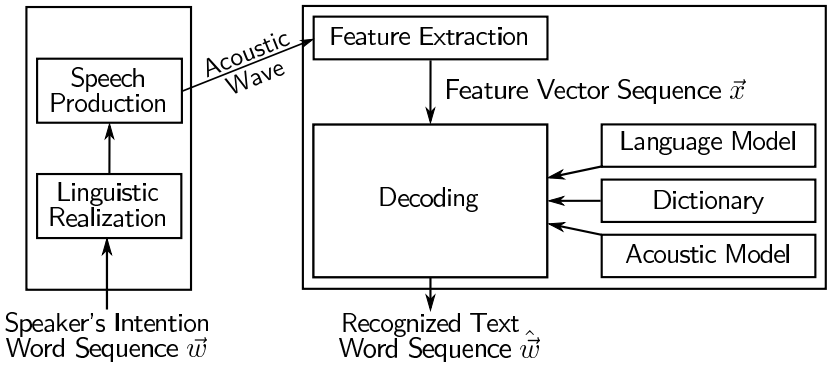


Figure 4.1: Schematic design of an automatic speech recognition system

models are used and estimated from large corpora of sample text. They ignore word contexts larger than $m-1$. Thus, the probability of a word sequence $\vec{w}$ of length N would be

$$P(\vec{w}) = \prod_{n=1}^{N} P(w_n | \langle w_{n-1}; \ldots; w_{n-m+1} \rangle) \tag{4.2}$$

From a grammatical point of view this is not really true because there is context of larger length in real sentences. However, this would require grammatical analysis and construction of texts which, as already mentioned, is not always very helpful with speech recognition. Instead, m-gram models focus on a purely statistical analysis of possible texts.

Usually, the following m-gram models are considered:

- zerograms: context length $m = 0$ (no context), all words have equal probability,
- unigrams: context length $m = 1$, words' individual probabilities are estimated,
- bigrams: context length $m = 2$, the co-occurrence of two words is estimated,
- trigrams: context length $m = 3$, the co-occurrence of three words is estimated.

For special linguistic constructs specific higher-order m-gram models may be used.

4.4 From words to acoustics

If we consider (4.1) again, we called $P(\vec{\boldsymbol{x}}|\vec{w})$ the acoustic model, i. e., it describes the probability of the human generating a certain acoustic realization $\vec{\boldsymbol{x}}$ of a word sequence $\vec{w}$. If we consider that each of the words w_n can be assigned a segment $\vec{\boldsymbol{x}}_{w_n} = \langle \boldsymbol{x}_{t_n^{\mathrm{start}}}; \ldots; \boldsymbol{x}_{t_n^{\mathrm{end}}} \rangle$ with different possible starting indices t_n^{start} and finishing

indices t_n^{end} such that $t_{n+1}^{\text{start}} = t_n^{\text{end}} + 1$ and $t_1^{\text{start}} = 1$ and $t_N^{\text{end}} = N$, the acoustic model can be expanded to

$$P(\vec{\boldsymbol{x}}|\vec{w}) = \sum_{\forall\ \text{seg}(\vec{\boldsymbol{x}},\vec{w})} \prod_{n=1}^{N} P(\vec{\boldsymbol{x}}_{w_n}|w_n) \tag{4.3}$$

where the sum runs over all possible segmentations $\text{seg}(\vec{\boldsymbol{x}}, \vec{w}_{[N]}) = \langle \vec{\boldsymbol{x}}_{w_1}; \ldots; \vec{\boldsymbol{x}}_{w_N} \rangle$ of the utterance $\vec{\boldsymbol{x}}$ into words $\vec{w}$. It should be noted that (4.3) discards co-articulation between words by assuming the word-level acoustic models to be statistically independent. This shortcoming will be remedied in a later section.

4.5 Sub-word phonetic modeling — the pronunciation dictionary

In the previous section we split the acoustic model for a complete sequence of words into distinct acoustic models for each individual word. In practice, however, having acoustic models at the word level has several disadvantages:

- Statistical unreliability: For the estimation of word-level acoustic models large amounts of training data would be necessary.
- Memory and computational complexity are too large.
- Low flexibility: Recognition of words outside the pre-defined vocabulary or addition of user-defined words to the dictionary (especially in speaker-independent systems) is impossible.
- Low modularity: Word-level acoustic models do not handle co-articulation between adjacent words well.

So, instead of acoustically modeling entire words, we borrow from linguists, i. e., we employ some prior knowledge.

The acoustic realization of a word results in a series of sounds with practically unlimited variability, depending on, e. g., the intonation, gender of the speaker, or age. However, it is common understanding that the acoustic realization of certain vowels and consonants results in reproducibly similar sounds. On the other hand it is known that the human ear can only distinguish a limited number of different sound classes, called phones. In the same way written words are composed of a limited number of letters, spoken words can be identified to consist of an also limited number of phonemes, i. e., classes of phones with the same function or meaning. Phonemes are purely theoretical constructs, a set of 40–50 phonemes is considered sufficient to describe Western languages.

This is very convenient for ASR systems because instead of different acoustic samples for each word only a number of acoustic samples for each phoneme is

```
coasts          k ow s            0.2
coasts          k ow s s          0.3
coasts          k ow s t s        0.5
humphrey        hh ah m f r iy    0.7
humphrey        hh ah m p f r iy  0.3
```

Figure 4.2: Excerpt from a dictionary listing different phonetic transcription of a word with associated probabilities.

required. Therefore, for each word in the vocabulary one or more entries in a pronunciation dictionary are defined, mapping a word w to a sequence of phonemes $\vec{h} = \langle h_1; \ldots; h_M \rangle$ from the set of phonemes $\mathcal{H}$ while possibly also assigning different probabilities $P(\vec{h}|w)$ to each pronunciation, see figure 4.2. Thus, the word-level acoustic model from (4.3) can be expanded to

$$P(\vec{\boldsymbol{x}}, \vec{h}|w) = \sum_{\forall\, \mathrm{seg}(\vec{\boldsymbol{x}}, \vec{h})} P(\vec{h}_{[M]}|w) \prod_{m=1}^{M} P(\vec{\boldsymbol{x}}_{h_m}|h_m) \tag{4.4}$$

where similar to (4.3) the sum runs over all possible temporal segmentation of the utterance segment $\vec{\boldsymbol{x}}$ into phonemes h. This model also assumes statistical independence between different phonemes, thus also ignoring dependence of a phoneme's acoustical realization on the current word as expressed in (4.4) by assuming that $P(\vec{\boldsymbol{x}}_{h_m}|h_m, w) = P(\vec{\boldsymbol{x}}_{h_m}|h_m)$, which is not true in real speech. This issue will be taken care of in section 4.7.1.

4.6 Hidden Markov phoneme models

By modeling entire speech utterances through concatenation of its building blocks, i. e., phonemes, we are now able to handle the complex speech recognition problem with relative ease. From (4.4) we know that we now only need to determine reference acoustic models of the phonemes. Of course, the general problem of great variability in speech still applies when modeling phonemes. Some of the variability, especially spectral variability, should have been eliminated by the feature extraction process mentioned in section 4.2. However, some spectral and all of the temporal variability are still left to be solved.

Phonemes are usually described using Hidden Markov Models (HMMs). An HMM is a finite state automaton whose state transitions and outputs are determined stochastically. More formally, it can be described by

- a finite set of states

$$\mathcal{S} = \{s_1, \ldots, s_Q\} \tag{4.5}$$

- transition probabilities
$$a_{(i,j)} = P(s_j|s_i) \tag{4.6}$$
which can be conveniently written in a matrix $\boldsymbol{A}_{[Q\times Q]}$,
- a set $\mathcal{X}$ of possible emission symbols $\boldsymbol{x}$ (discrete or continuous),
- emission (production) probabilities
$$b_{(j,l)} = P(\boldsymbol{x}_l|s_j) \tag{4.7}$$

This means that the HMM produces an observable sequence of random symbols $\vec{\boldsymbol{x}}$ with $\boldsymbol{x}_t \in \mathcal{X}$ while traversing an unobservable random sequence of states $\vec{s}$ with $s_t \in \mathcal{S}$. This property can be used in automatic speech recognition where an observable sequence of random symbols (the feature vectors $\boldsymbol{x}$) are produced by an unobserved sequence of states, i. e., word, phoneme, or even sub-phoneme states. This matching of the speech production model with the structure of the HMM can, similar to the motivation of the pronunciation dictionary, be regarded as prior knowledge.

Using HMMs for modeling phonemes has the clear advantage that the same HMM can be used to match different possible segmentations of a word-level utterance segment into phoneme-level utterance segments by simply letting it traverse the desired length of a state sequence.

For automatic speech recognition, only first-order HMMs are used:

$$P(s_t|\langle s_{t-1};\ldots;s_0\rangle) = P(s_t|s_{t-1}) \tag{4.8}$$
$$P(\boldsymbol{x}_t|\langle s_t;\ldots;s_0\rangle) = P(\boldsymbol{x}_t|s_t) \tag{4.9}$$

which has already been assumed in (4.6) and (4.7), respectively. Figure 4.3 illustrates a simple example of a left-to-right 3-state HMM.

Thus, we can once more do an expansion which results in a refinement of the phoneme-level acoustic model:

$$P(\vec{\boldsymbol{x}}_h, \vec{s}|h) = \sum_{\forall \vec{s}\in\mathcal{S}^T} P(\vec{\boldsymbol{x}}_h|\vec{s},h)P(\vec{s}|h) = \sum_{\forall \vec{s}\in\mathcal{S}^T}\prod_{t=1}^{T} P(\boldsymbol{x}_t|s_t,h)P(s_t|s_{t-1},h) \tag{4.10}$$

4.6.1 Gaussian Mixture Models for emission probabilities

Typically, the emission probabilities are either discrete (feature vectors $\boldsymbol{x}$ have been pre-binned and the emission probability is computed for the relevant bin only — also known as vector quantization) or continuous in which case probability density functions must be used instead. These continuous density functions are commonly modeled using weighted sums of normal densities, so-called Gaussian Mixture

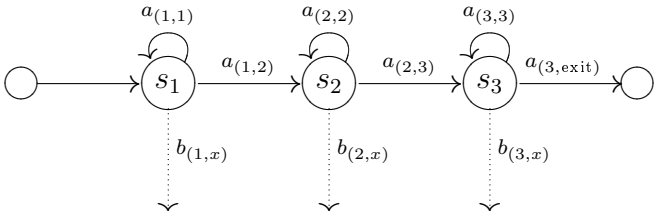


Figure 4.3: Schematic illustration of a first-order Hidden Markov Model

Models (GMMs).

$$p(\boldsymbol{x}|s) = \sum_{k=1}^{K_s} w_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\Sigma}_{s,k}, \boldsymbol{\mu}_{s,k}), \quad \sum_{k=1}^{K_s} w_k = 1 \tag{4.11}$$

The use of density values in place of discrete probability values must be justified. If we considered continuous-valued feature vectors $\boldsymbol{x}$, then $P(\boldsymbol{x} = \boldsymbol{x}_{\text{fixed}}|\dots) = 0$. If the probability density function exists, the probability $P(\boldsymbol{x} = \boldsymbol{x}_{\text{fixed}}|\dots)$ may be computed:

$$P(\boldsymbol{x} = \boldsymbol{x}_{\text{fixed}}|\dots) = \lim_{\delta\boldsymbol{x}\to 0} \int_{\boldsymbol{x}_{\text{fixed}}-\delta\boldsymbol{x}}^{\boldsymbol{x}_{\text{fixed}}+\delta\boldsymbol{x}} p(\boldsymbol{x}|\dots)\,\mathrm{d}\,\boldsymbol{x} = 0 \tag{4.12}$$

which we knew before. If, however, we plug this formula into Bayes' rule (4.1) for both the acoustic model and the feature vectors' prior probabilities, we may apply L'Hospital's rule:

$$\frac{P(\boldsymbol{x} = \boldsymbol{x}_{\text{fixed}}|\dots)}{P(\boldsymbol{x} = \boldsymbol{x}_{\text{fixed}})} = \lim_{\delta\boldsymbol{x}\to 0} \frac{\int_{\boldsymbol{x}_{\text{fixed}}-\delta\boldsymbol{x}}^{\boldsymbol{x}_{\text{fixed}}+\delta\boldsymbol{x}} p(\boldsymbol{t}|\dots)\,\mathrm{d}\,\boldsymbol{t}}{\int_{\boldsymbol{x}_{\text{fixed}}-\delta\boldsymbol{x}}^{\boldsymbol{x}_{\text{fixed}}+\delta\boldsymbol{x}} p(\boldsymbol{t})\,\mathrm{d}\,\boldsymbol{t}} = \frac{p(\boldsymbol{x}_{\text{fixed}}|\dots)}{p(\boldsymbol{x}_{\text{fixed}})} \tag{4.13}$$

where the term $p(\boldsymbol{x}_{\text{fixed}})$ may be omitted in the case of MAP-rule classification. Consequently, discrete probability and continuous probability density values may be used as appropriate and available.

4.6.2 Alternative models for emission probabilities

The predominant model for HMMs' emission probabilities in speech recognition are the aforementioned GMMs. Naturally, GMMs are not the only existing models for estimating probabilities. In speech recognition, Artificial Neural Networks (specifically, Multi-Layer Perceptrons with sigmoid output functions) were successfully used for estimating HMMs emission probabilities, outperforming conventional GMM models [Bourlard and Morgan, 1998, Trentin and Gori, 2001]. In chapters 5 and 6, other models, employing Structural Risk Minimization during training, will be introduced and investigated.

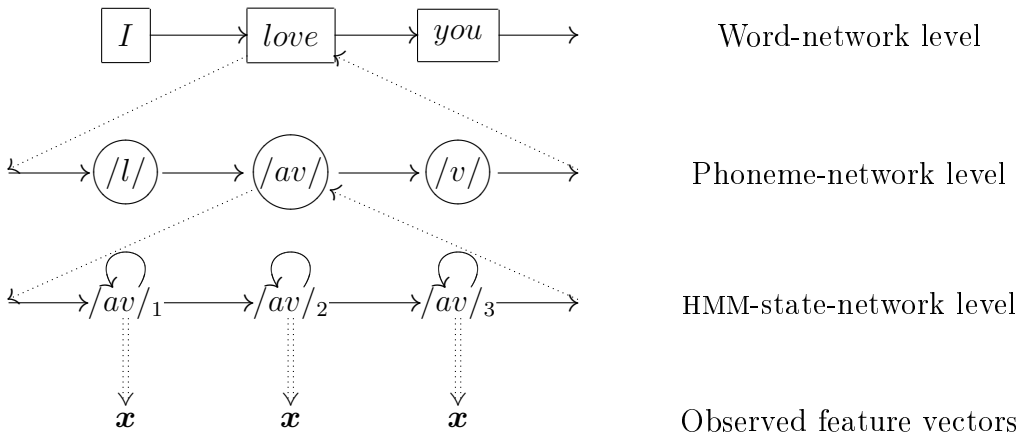


Figure 4.4: Schematic illustration of the hierarchical decomposition of the speech recognition problem

4.7 Combining language model, pronunciation dictionary, and phoneme models

By recursively substituting (4.10), (4.4), (4.3), and (4.2) into (4.1) we can grasp how the ASR problem is tackled by a divide-and-conquer strategy. The probability of a hypothesis sentence $\vec{w}$ being the source of the observed utterance u coded as the sequence $\vec{\boldsymbol{x}}$ of feature vectors $\boldsymbol{x}$ can, thus, be computed. The hypothesis $\vec{w}$ is generated from a grammar which describes all possible sentences of the domain. For limited-domain tasks, e. g., voice dialing of ten-digit phone numbers, this grammar can be quite simple, but it can be almost arbitrarily complex for large-vocabulary systems, i. e., a loop of all words from the vocabulary is used. This hypothesis sentence $\vec{w}$ is assigned a prior probability through the language model. To compute the acoustic probability of the hypothesis $\vec{w}$ it is hierarchically expanded into possible sequences $\vec{h}$ of phonemes where each is assigned a conditional probability through the dictionary, cf. (4.4). Each of these possible phoneme sequences $\vec{h}$ is in turn expanded into sequences of HMM-states $\vec{s}$, each with a certain individual probability computed from the transition probabilities. Each of the observed feature vectors $\boldsymbol{x}_t$ is finally matched against each hypothesis state s_t by computing the state's emission probability $P(\boldsymbol{x}_t|s_t)$. So, for the full probability $P(\vec{w}|\vec{\boldsymbol{x}})$ the sum over all possible expansions of the hypothesis $\vec{w}$ must be computed, which is illustrated in figure 4.4.

To summarize, the main building blocks of contemporary ASR systems are the language model, the phonetic dictionary, and the sub-word acoustic hidden Markov model (HMM) typically using mixtures of normal densities (GMMs) as emission probabilities.

4.7.1 Context-dependent sub-word models

The full expansion of a hypothesis sentence $\vec{w}$ into sub-word phonemes now makes it possible to remedy the assumption of statistical independence of the pronunciation of words which neglected inter-word co-articulation, cf. section 4.4, and, in turn, the assumption of statistical independence of the acoustic realization of different phonemes which neglected intra-word co-articulation, cf. section 4.5. Instead of modeling individual phonemes, (additional) artificial models for phonemes in the temporal context of other phonemes can be generated and trained. Most commonly, only the phonemes immediately preceding and following a phoneme are considered; the resulting constructs are consequently called *triphones*, whereas individual phoneme models are called *monophones* to distinguish them from the context-dependent variants. If we consider 50 monophones, this would result in $50^3 = 125000$ triphones. Fortunately, only about 1500 of these possible combinations really occur in actual speech, which drastically reduces the complexity. Additionally, approaches such as *generalized triphones* [Lee and Reddy, 1988] and others have been used to further decrease the complexity of the problem. In the context of learning theory introduced in chapter 2, this reduction of the number of context-dependent models can be regarded as a kind of capacity control which is guided by prior knowledge about the speech recognition problem.

4.7.2 Minimizing number of parameters and improving robustness

If we consider an acoustic model consisting of 1500 triphone HMMs, each having 3 emitting states, with each state using a GMM with, e. g., 8 mixture components, we would have to determine the parameters of 36000 normal densities. Usually, this is not possible because the amount of training data is too limited. If viewed in the perspective of learning theory, the capacity of the acoustic model is too large. A number of approaches are used to empirically limit this capacity, including, but not limited too,

- considering only diagonal covariance matrices in the GMMs' constituting normal densities,
- tying the parameters of (parts of) phonetically similar HMMs, especially in the context of triphone HMMs,
- introducing lower bounds on the values of the weights and entries of the covariance matrices in each GMM,
- obeying upper bounds on the number of components in each GMM,

- constructing "artificial" phonetic units called *senones* [Hwang and Huang, 1992].

Of course, the decision on the tying and bounding of parameters must be guided by prior knowledge and experimental validation.

4.8 Viterbi decoding

The word-net resulting from the grammar can be depicted as a lattice where each feature vector $\boldsymbol{x}_t$ can be assigned to any word. The lattice can be expanded to the monophone or HMM-state level where, again, each $\boldsymbol{x}_t$ can be assigned to any monophone or HMM-state. Technically, to compute the full acoustic likelihood $P(\vec{w}, \vec{\boldsymbol{x}})$ of a word sequence $\vec{w}$, the individual acoustic likelihoods of *all* possible paths through the lattice, which resemble the word sequence $\vec{w}$, must be computed and accumulated. Of course, this accumulation must be carried out for each allowed hypothesis word sequence $\vec{w}$. If done naïvely, this is computationally very intractable. However, the method of Dynamic Programming [Bellmann, 1957] can be used to reduce the computational complexity, resulting in the well-known Viterbi algorithm [Viterbi, 1967]. Also, a method call *beam searching* is commonly employed to further speed up computations [Greer et al., 1982]. Instead of considering all possible hypotheses until the end of the utterance, at time t those hypotheses whose acoustic likelihoods fall below a user-defined threshold from the so far observed maximum likelihood are no longer considered, i. e., pruned. Experiments have shown that this results in a very fair trade-off between recognition accuracy and runtime requirements.

4.8.1 Forced alignment

Instead of matching an utterance with a free-form grammar, Viterbi decoding may also be used on the word-level transcription of a pre-recorded utterance. This process is known as *forced alignment*. Due to the limited single-path word net, alternatives are limited to different time-warpings, i. e., assignments of feature vectors to words, phonemes, HMMs, or HMM states. The path with the highest likelihood is then used to construct temporal boundaries on a word, phoneme, or even HMM-state level.

Time alignment are especially useful for supervised learning of other components of an ASR system, e. g., feature preprocessors based on Linear Discriminant Analysis [Haeb-Umbach and Ney, 1992, Schafföner et al., 2003] or acoustic models incorporating binary discriminants which will be introduced in chapter 6.

4.9 Training

The parameters of the HMM-GMM-system are conventionally determined by maximizing the log-likelihood of the observed utterance

$$L = \log P(\vec{\boldsymbol{x}}|\boldsymbol{\alpha}) = \log \sum_{\forall \vec{s} \in \mathcal{S}^T} P(\vec{\boldsymbol{x}}, \vec{s}|\boldsymbol{\alpha}) \tag{4.14}$$

for the training sequence $\vec{\boldsymbol{x}} = \langle \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T \rangle$ and all admissible expansions of the reference word sequence into HMM-states $\vec{s} = \langle s_1, \ldots, s_T \rangle$. The parameter set of the HMM-GMM-system, consisting of the state-transition and symbol-emission probabilities, is denoted by $\boldsymbol{\alpha}$.

The likelihood is iteratively maximized approximately by updating the HMM-GMM-system's parameters using the Baum-Welch procedure [Rabiner and Juang, 1993, Wendemuth, 2004]. The GMMs are usually initialized with only one normal density component, and the parameters are trained with a few iterations of the Baum-Welch-procedure. The single normal densities are then split into a mixture of two normal densities along the direction of largest variance. Again, the two-component GMMs are trained with some iterations of the Baum-Welch-procedure. Then, the number of GMM components is increased by splitting the normal density component corresponding to the largest weight w_c from (4.11). The alternative splitting/updating procedure is carried out obeying the restrictions mentioned in section 4.7.2. Looking at this training procedure from the learning theory perspective, the act of limiting the number of Baum-Welch-iterations for each number of GMM-components can also be viewed as limiting the capacity of the HMM-GMM-system in order to avoid over-fitting to the training data.

It is known, however, that the ML-estimation may converge into a local optimum which may not necessarily be a global optimum. Also, ML estimation does not consider the risk of misclassification. This may lead to recognition accuracies well below the possible optimum. Therefore, alternative training procedures have been suggested. These training methods commonly run under the name *discriminative training methods*, amongst which Maximum Mutual Information [Normandin, 1991]. For further reading, e. g., [Schlüter and Macherey, 1998, Schlüter, 2000, Macherey et al., 2005], [Woodland and Povey, 2002], or [Kaiser et al., 2002] may be of interest.

5 Sparse Kernel Probability Density Modeling using Structural Risk Minimization

As mentioned in section 4.6.2, probability density functions are used within HMMs when modeling sequences of continuous-valued events. Traditionally, PDFs trained using an ERM inductive principle have been used, e. g., GMMs or MLPs with suitable transfer functions. In this chapter we will investigate how kernel PDF models can be trained using the SRM inductive principle. We will revisit a procedure introduced in [Vapnik and Mukherjee, 2000] and show how this procedure is limited from an algorithmic and an experimental point of view. Learning from the limitations of this method, we will introduce a novel SRM-based training procedure for sparse kernel-PDF models which is more robust and more efficient.

5.1 Kernel probability density estimation by regression

A probability density $p(\boldsymbol{x})$ is defined as the solution of

$$\int_{-\infty}^{\boldsymbol{x}} p(\boldsymbol{t})\,\mathrm{d}\,\boldsymbol{t} = F(\boldsymbol{x}) \tag{5.1}$$

subject to the constraints

$$\int_{-\infty}^{\infty} p(\boldsymbol{t})\,\mathrm{d}\,\boldsymbol{t} = 1 \tag{5.2}$$

$$p(\boldsymbol{x}) \geq 0 \tag{5.3}$$

where $F(\boldsymbol{x})$ is the probability distribution function. Since $F(\boldsymbol{x})$ is unknown, (5.1) must be solved using an approximation $F_e(\boldsymbol{x})$, the empirical cumulative distribution function (ECDF) (cf. section 5.2).

Given an i. i. d. sample data set $\boldsymbol{X} = \{\boldsymbol{x}_{(1)}, \ldots, \boldsymbol{x}_{(N)}\}$ with $\boldsymbol{x} \in \mathbb{R}^D$ drawn from the unknown distribution $F(\boldsymbol{x})$, we wish to estimate the unknown density $p(\boldsymbol{x})$ such that

$$p(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_{(n)} k(\boldsymbol{x}, \boldsymbol{x}_{(n)}) \tag{5.4}$$

subject to the constraints

$$\boldsymbol{\alpha} \geq \mathbf{0} \tag{5.5}$$

$$\mathbf{1}^\top \boldsymbol{\alpha} = 1 \tag{5.6}$$

using a kernel $k(\boldsymbol{x}, \boldsymbol{y})$ with the properties

$$k(\boldsymbol{x}, \boldsymbol{y}) \geq 0 \tag{5.7}$$

$$\int_{-\infty}^{x} k(\boldsymbol{t}, \boldsymbol{y}) \,\mathrm{d}\, \boldsymbol{t} \geq 0 \tag{5.8}$$

$$\int_{-\infty}^{\infty} k(\boldsymbol{t}, \boldsymbol{y}) \,\mathrm{d}\, \boldsymbol{t} = 1 \tag{5.9}$$

to satisfy the constraints from (5.2) and (5.3). Examples of kernels satisfying these constraints are

- the normalized Gaussian kernel

$$k_{\mathrm{ngauss}}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{(2\pi\sigma^2)^{D/2}} \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{y}||^2}{2\sigma^2}\right) \tag{5.10}$$

- the Epanechnikov kernel

$$k_{\mathrm{epa}}(\boldsymbol{x}, \boldsymbol{y}) = \frac{3}{4}(1 - ||\boldsymbol{x} - \boldsymbol{y}||^2/\sigma^2)_+ \tag{5.11}$$

- the triangle kernel

$$k_{\mathrm{tri}}(\boldsymbol{x}, \boldsymbol{y}) = (1 - ||\boldsymbol{x} - \boldsymbol{y}||/\sigma)_+ \tag{5.12}$$

with $\sigma > 0$.

A possible solution of (5.4) is provided by the well known method of Parzen windows [Parzen, 1962] with $\alpha_{(n)} = 1/N, n = 1 \ldots N$, and any kernel (window function) satisfying constraints (5.7)–(5.9). This method is known to perform very well, however, it relies on all the problem's samples to characterize the solution. Therefore, we strive to obtain a solution with some or most $\alpha_{(n)} = 0$, i. e., a sparse approximation of the Parzen windows solution.

If the parameters of the kernel k are considered fixed, the density model is completely characterized by the weight vector $\boldsymbol{\alpha}$. With (5.1), kernel density estimation can then be posed as the regression modeling problem

$$F_e(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_{(n)} q(\boldsymbol{x}, \boldsymbol{x}_{(n)}) + \varepsilon(\boldsymbol{x}) \tag{5.13}$$

with

$$q(\boldsymbol{x}, \boldsymbol{x}_{(n)}) = \int_{-\infty}^{\boldsymbol{x}} k(\boldsymbol{t}, \boldsymbol{x}_{(n)}) \,\mathrm{d}\, \boldsymbol{t} \tag{5.14}$$

and $\varepsilon(\boldsymbol{x})$ the modeling error at $\boldsymbol{x}$ and F_e the problem's empirical cumulative distribution function, subject to constraints (5.5) and (5.6).

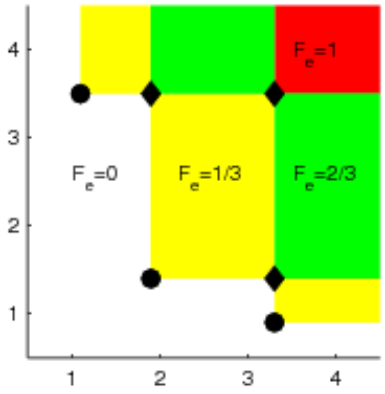


Figure 5.1: Construction of a two-dimensional example ECDF. Circles denote original problem samples, diamonds denote additional ECDF jumps, colored areas denote value of the ECDF.

5.2 The empirical cumulative distribution function

Given that the true distribution function $F(\boldsymbol{x})$ is unknown in (5.1), we resort to an estimation. We use the empirical cumulative distribution function (ECDF) $F_e(\boldsymbol{x})$ as it is the lowest variance unbiased estimator of the true distribution function [Scott, 1992]. The ECDF is determined solely by the samples $\boldsymbol{x}_{(n)}, n = 1 \dots N$, such that

$$F_e(\boldsymbol{x}) = \frac{\sum_{n=1}^{N} \Theta(\boldsymbol{x}, \boldsymbol{x}_{(n)})}{N} \tag{5.15}$$

where $\Theta(\boldsymbol{x}, \boldsymbol{y})$ denotes point dominance

$$\Theta(\boldsymbol{x}, \boldsymbol{y}) = \{\boldsymbol{x} \succeq \boldsymbol{y}\} = \begin{cases} 1, & \text{if } x_{(d)} \geq y_{(d)} \quad \forall d = 1 \dots D \\ 0, & \text{else.} \end{cases} \tag{5.16}$$

This means that $F_e(\boldsymbol{x})$ is a function which is given non-parametrically [Scott, 1992]. The ECDF has a staircase shape with N jumps in the one-dimensional case. In figure 5.1 an admittedly degenerate two-dimensional problem with $N = 3$ problem sample points is shown where the corresponding ECDF has a much more complex shape with $6 > (N = 3)$ jumps.

From (5.15) it is not clear at which points the regression fit in (5.13) is to be evaluated. In the one-dimensional case, one usually samples F_e at the problem's sample points $\boldsymbol{x}_{(n)}$ for convenience, since that's where the jumps of the function are located. In the multi-dimensional case, sampling F_e at the problem's sample points does not capture the complexity of the ECDF. Even worse, in high dimensions a situation can occur where none of the samples dominates any other sample (as in figure 5.1), i. e., sampling F_e at the problem sample points yields only the one

function value $1/N$ instead of the entire range $[0, 1]$. Therefore, the ECDF has to be sampled at additional locations.

One possibility is to explicitly compute all the points at which jumps in the ECDF occur, as these points are located on a grid defined by the problem's sample points. This computation can be quite involving, but in [Fonseca, 2002] an algorithm is proposed which solves the problem in $\mathcal{O}(Dk)$ time where k is the sample-dependent number of jumps of the ECDF. Since in high dimensions the number k of output points may also be very large, one may resort to only using a feasible number of randomly picked points from this algorithm. This approach then will not capture the full complexity of the ECDF but at least use meaningful points which generate the entire range $[0, 1]$ of the ECDF.

5.3 Support-vector regression of the ECDF using the ϵ-insensitive loss funtion

Consider that suitable locations $\boldsymbol{c}_{(s)},\ s = 1 \dots S$, for checking the regression model (5.13) have been identified as in section 5.2, we rewrite (5.13) in a more convenient fashion:

$$\boldsymbol{F}_e = \boldsymbol{Q}\boldsymbol{\alpha} + \boldsymbol{\varepsilon} \tag{5.17}$$

with $F_{e(s)} = F_e(\boldsymbol{c}_{(s)})$ and $q_{(s,k)} = q(\boldsymbol{c}_{(s)}, \boldsymbol{x}_{(k)})$, $\varepsilon_{(s)} = \varepsilon(\boldsymbol{c}_{(s)})$. If we accept only some fixed constant error at each $\boldsymbol{c}_{(s)}$, we can find a solution using a support-vector regression approach [Vapnik and Mukherjee, 2000]:

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}}\, \boldsymbol{\alpha}^\top \boldsymbol{K}\boldsymbol{\alpha} \tag{5.18}$$

subject to

$$\|\boldsymbol{Q}\boldsymbol{\alpha} - \boldsymbol{F}_e\| \leq \varepsilon_N \mathbf{1} \tag{5.19}$$

$$\boldsymbol{\alpha} \geq \mathbf{0} \tag{5.20}$$

$$\mathbf{1}^\top \boldsymbol{\alpha} = 1 \tag{5.21}$$

The optimization problem (5.18) states that the norm of the weight vector in a RKHS is to be minimized subject to constraints (5.19) and (5.20). The approach is very similar to finding the optimal separating hyperplane for pattern recognition considered in section 3.1.1. Since an ϵ-insensitive loss function is used, this method will be called ϵ-support vector densities (ϵ-SVD).

The maximum allowed error ε_N can, in the univariate case, be obtained from the Kolmogorov-Smirnov distribution [Stephens, 1974]

$$P\left(\sup_x |F(x) - F_e(x)| < \varepsilon/\sqrt{N}\right) = 1 - 2\sum_{k=1}^{\infty}(-1)^{k-1} e^{-2\varepsilon^2 k^2} \tag{5.22}$$

However, in multivariate cases it is still unclear how such a distribution-independent goodness-of-fit statistic can be constructed [Paramasamy, 1992, Cabaña and Cabaña, 1997, Justel et al., 1997, Loudin and Miettinen, 2003]. Vapnik and Mukherjee suggest to evaluate appropriate values of ε_N by simulations, but they neither describe the exact procedure nor whether these values can be applied independently of the distribution [Vapnik and Mukherjee, 2000], which leaves the general usefulness of this method for higher-dimensional problems questionable.

The above algorithm has the important advantage that if the maximum allowed error ε_N is known, there is no parameter to be manually set as the kernel parameter, e. g., the standard deviation of the normalized Gaussian kernel, is simply set to the largest value fulfilling constraint (5.19). However, our experiments, which will be discussed in section 5.5, showed that this rule may result in under-estimating the kernel width leading to over-fitting of the resulting model.

5.3.1 Adding slack variables

It may, therefore, be desirable to increase the kernel width beyond the limits set by constraint (5.19). This may be acceptable as we can imagine that the sample set $\boldsymbol{X}$ used for construction of the ECDF may contain noisy or outlier samples. To handle this case, the optimization problem must be augmented with slack variables, leading to

$$\min_{\boldsymbol{\alpha},\boldsymbol{\xi}_+,\boldsymbol{\xi}_-} \boldsymbol{\alpha}^\top \boldsymbol{K}\boldsymbol{\alpha} + C\mathbf{1}^\top(\boldsymbol{\xi}_+ + \boldsymbol{\xi}_-) \tag{5.23}$$

subject to

$$\boldsymbol{F}_e - \boldsymbol{Q}\boldsymbol{\alpha} \leq \varepsilon_N\mathbf{1} + \boldsymbol{\xi}_+ \tag{5.24}$$
$$\boldsymbol{Q}\boldsymbol{\alpha} - \boldsymbol{F}_e \leq \varepsilon_N\mathbf{1} + \boldsymbol{\xi}_- \tag{5.25}$$
$$\boldsymbol{\alpha}, \boldsymbol{\xi}_+, \boldsymbol{\xi}_- \geq \mathbf{0} \tag{5.26}$$
$$\mathbf{1}^\top\boldsymbol{\alpha} = 1 \tag{5.27}$$

where C defines the trade-off between smoothness of the solution and committed error, as usual.

However, this problem cannot be solved efficiently. A decomposition similar to the ones mentioned in section 3.4 is impossible due to the coupling of the primal variables $\boldsymbol{\alpha}$ by constraints (5.24) and (5.25). On the other hand, general-purpose QP solvers like the ones introduced in section 3.3 must, due to the slack variables being primal variables, consider (5.23) a $3N \times 3N$ optimization problem which cannot be solved efficiently for but the smallest real-world problems.

5.4 Least-squares regression of the ECDF

Let us re-consider the formulation (5.17). If an orthogonal decomposition of $\boldsymbol{Q} = \boldsymbol{W}\boldsymbol{A}$ with $\boldsymbol{w}_{(i)}^{\top}\boldsymbol{w}_{(j)} = 0$, $i \neq j$, is assumed, (5.17) can be written as

$$\boldsymbol{F}_e = \boldsymbol{W}\boldsymbol{g} + \boldsymbol{\varepsilon} \tag{5.28}$$

with

$$\boldsymbol{g} = \boldsymbol{A}\boldsymbol{\alpha} \tag{5.29}$$

the weights in the orthogonal space $\boldsymbol{W}$. Similar to the motivation of the LS-SVM, we can replace the inequality constraint (5.19) with an equality constraint and use a square loss function. The optimal weight vector $\hat{\boldsymbol{g}}$ can be obtained as the solution of the least squares problem with local regularization

$$\hat{\boldsymbol{g}} = \underset{\boldsymbol{g}}{\operatorname{argmin}}\, \boldsymbol{\varepsilon}^{\top}\boldsymbol{\varepsilon} + \sum_{n=1}^{N} \lambda_{(n)} g_{(n)}^2 \tag{5.30}$$

where $\boldsymbol{\lambda}$ is the regularization parameter vector. This vector is optimized based on the Bayesian evidence procedure [MacKay, 1992]. Briefly, the update of the regularization parameter $\lambda_{(m)}$ works as follows:

$$\lambda_{(m)}^{\text{new}} = \frac{\gamma_{(m)}\boldsymbol{\varepsilon}^{\top}\boldsymbol{\varepsilon}}{(N - \mathbf{1}^{\top}\boldsymbol{\gamma})g_{(m)}^2} \tag{5.31}$$

where

$$\gamma_{(m)} = \frac{\boldsymbol{w}_{(m)}^{\top}\boldsymbol{w}_{(m)}}{\lambda_{(m)}^{\text{old}} + \boldsymbol{w}_{(m)}^{\top}\boldsymbol{w}_{(m)}} \tag{5.32}$$

Details of the derivation of the formulas can be found in [Chen et al., 2004].

Inserting (5.28) into (5.30) yields

$$\hat{\boldsymbol{g}} = \underset{\boldsymbol{g}}{\operatorname{argmin}}\, \boldsymbol{g}^{\top}\boldsymbol{W}^{\top}\boldsymbol{W}\boldsymbol{g} + \sum_{n=1}^{N} \lambda_{(n)} g_{(n)}^2 - 2\boldsymbol{F}_e^{\top}\boldsymbol{W}\boldsymbol{g} \tag{5.33}$$

which reveals the merit of the orthogonal decomposition that the components of $\boldsymbol{g}$ can be optimized independently of each other which lends itself to forward selection if a sparse approximation of (5.33) is intended.

5.4.1 Memory-efficient orthogonal decomposition for forward-selection

In [Chen et al., 2004] the modified Gram-Schmidt (MGS) procedure is proposed for the orthogonalization of $\boldsymbol{Q}$. However, since we are interested in a sparse approximation of (5.33) via forward selection, in the mth selection iteration we need to

orthogonalize $N - m$ columns of $\boldsymbol{Q}$ onto the previously selected m columns in $\boldsymbol{W}$. This implies that the complete matrix $\boldsymbol{Q}$ must be known and kept in memory. The memory complexity of the MGS algorithm is thus roughly $\mathcal{O}(SN + mN)$ for the complete $\boldsymbol{Q}$ matrix (including the orthogonalized parts here referred to as $\boldsymbol{W}$) and the m complete rows of $\boldsymbol{A}$.

Therefore, we propose a more efficient algorithm which computes both $\boldsymbol{W}$ and $\boldsymbol{A}$ column-by-column without the need to keep unused columns of these matrices in memory. This algorithm, which is based on a rank-update of the pseudo-inverse, has already successfully been employed for computing discriminants [Andelic et al., 2007] and is extended here to the case of sparse kernel PDF estimation.

In the mth forward selection step we consider the following partitioning of the reduced $\boldsymbol{Q}$ matrix and corresponding $\boldsymbol{\alpha}$

$$\boldsymbol{Q}_m = \left[\boldsymbol{Q}_{m-1}\boldsymbol{q}_{(m)}\right] \tag{5.34}$$

$$\boldsymbol{\alpha}_m = \left[\boldsymbol{\alpha}_{m-1}\alpha_{(m)}\right]^\top \tag{5.35}$$

such that the square loss from (5.30) becomes

$$L(\boldsymbol{\alpha}_{m-1}, \alpha_{(m)}) = \|\boldsymbol{Q}_{m-1}\boldsymbol{\alpha}_{m-1} - (\boldsymbol{F}_e - \boldsymbol{q}_{(m)}\alpha_{(m)})\|^2 \tag{5.36}$$

The minimum of (5.36) is given by

$$\hat{\boldsymbol{\alpha}}_{m-1} = \boldsymbol{Q}_{m-1}^\dagger(\boldsymbol{F}_e - \boldsymbol{q}_{(m)}\alpha_{(m)}) \tag{5.37}$$

with $\boldsymbol{Q}_{m-1}^\dagger$ the pseudo-inverse of $\boldsymbol{Q}$. This yields after insertion into (5.36)

$$L(\alpha_{(m)}) = \|(\boldsymbol{I} - \boldsymbol{Q}_{m-1}\boldsymbol{Q}_{m-1}^\dagger)\boldsymbol{q}_{(m)}\alpha_{(m)} - (\boldsymbol{I} - \boldsymbol{Q}_{m-1}\boldsymbol{Q}_{m-1}^\dagger)\boldsymbol{F}_e\|^2 \tag{5.38}$$

with $\boldsymbol{I}$ the identity matrix of appropriate size.

The minimum of (5.38) is reached at

$$\alpha_{(m)} = \boldsymbol{w}_{(m)}^\dagger(\boldsymbol{I} - \boldsymbol{Q}_{m-1}\boldsymbol{Q}_{m-1}^\dagger)\boldsymbol{F}_e \tag{5.39}$$

Considering that the pseudo-inverse of $\boldsymbol{w}$ is given by

$$\boldsymbol{w}_{(m)}^\dagger = \frac{\boldsymbol{w}_{(m)}^\top}{\|\boldsymbol{w}_{(m)}\|^2} \tag{5.40}$$

(5.39) may be written as

$$\begin{aligned}\alpha_{(m)} &= \frac{\boldsymbol{w}_{(m)}^\top(\boldsymbol{I} - \boldsymbol{Q}_{m-1}\boldsymbol{Q}_{m-1}^\dagger)\boldsymbol{F}_e}{\|\ \boldsymbol{w}_{(m)}\|^2} \\ &= \frac{\boldsymbol{q}_{(m)}^\top(\boldsymbol{I} - \boldsymbol{Q}_{m-1}\boldsymbol{Q}_{m-1}^\dagger)^\top(\boldsymbol{I} - \boldsymbol{Q}_{m-1}\boldsymbol{Q}_{m-1}^\dagger)\boldsymbol{F}_e}{\|\boldsymbol{w}_{(m)}\|^2}\end{aligned} \tag{5.41}$$

The matrix

$$\boldsymbol{P}_m = \boldsymbol{I} - \boldsymbol{Q}_{m-1}\boldsymbol{Q}_{m-1}^{\dagger} \tag{5.42}$$

is an orthogonal projection matrix which implies it being symmetric and idempotent. Thus, (5.41) can be simplified as

$$\alpha_{(m)} = \boldsymbol{w}_{(m)}^{\dagger}\boldsymbol{F}_e \tag{5.43}$$

Combining (5.43) with (5.37), the weight vector $\hat{\boldsymbol{\alpha}}_m$ may be updated as

$$\hat{\boldsymbol{\alpha}}_m = \begin{bmatrix} \hat{\boldsymbol{\alpha}}_{m-1} \\ \hat{\alpha}_{(m)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q}_{m-1}^{\dagger} - \boldsymbol{Q}_{m-1}^{\dagger}\boldsymbol{q}_{(m)}\boldsymbol{w}_{(m)}^{\dagger} \\ \boldsymbol{w}_{(m)}^{\dagger} \end{bmatrix} \boldsymbol{F}_e \tag{5.44}$$

yielding the update

$$\boldsymbol{Q}_m^{\dagger} = \begin{bmatrix} \boldsymbol{Q}_{m-1}^{\dagger} - \boldsymbol{Q}_{m-1}^{\dagger}\boldsymbol{q}_{(m)}\boldsymbol{w}_{(m)}^{\dagger} \\ \boldsymbol{w}_{(m)}^{\dagger} \end{bmatrix} \tag{5.45}$$

of the current pseudo-inverse.

Because every projection $\boldsymbol{w}_{(m)} = \boldsymbol{P}_m\boldsymbol{q}_{(m)}$ lies in a subspace orthogonal to $\boldsymbol{Q}_{m-1}$, it follows directly that $\boldsymbol{w}_{(i)}^{\top}\boldsymbol{w}_{(j)} = 0, i \neq j$. Therefore, the orthogonal decomposition of $\boldsymbol{Q}_m$ can be updated as

$$\boldsymbol{W}_m = [\boldsymbol{W}_{m-1}\boldsymbol{w}_{(m)}] \tag{5.46}$$

$$\boldsymbol{A}_m = \begin{bmatrix} \boldsymbol{A}_{m-1}(\boldsymbol{W}_m^{\top}\boldsymbol{W}_m)^{-1}\boldsymbol{W}_m^{\top}\boldsymbol{q}_{(m)} \\ \boldsymbol{0}_{m-1}^{\top} \end{bmatrix} \tag{5.47}$$

for which we call our algorithm order-recursive orthogonal least squares (OROLS). The inversion of $(\boldsymbol{W}_m^{\top}\boldsymbol{W}_m)$ is trivial since it is diagonal. It is important to monitor the condition number of $\boldsymbol{W}_m$ for it increases as the number m of admitted columns grows, so to ensure numerical stability a termination threshold on the condition number must be defined.

So the complete algorithm is as follows:

1. Define the set of $\mathcal{M} = \{1, \ldots, N\}$ of all possible sample indices, the set $\mathcal{S} = \emptyset$ of already admitted sample indices, the current solution index $m = 1$

2. For each $n \in \mathcal{M}\backslash\mathcal{S}$, do the following
   a) Compute the corresponding updates of $\boldsymbol{W}_m$ and $\boldsymbol{A}_m$ via (5.46) and (5.47)
   b) If the condition number $\boldsymbol{w}_{(m)}^{\top}\boldsymbol{w}_{(m)}$ violates a threshold, continue with the next n
   c) Solve the LS-problem from (5.33) while optimizing the regularization parameter $\lambda_{(m)}$

d) Reconstruct the original weights $\boldsymbol{\alpha}_m = \boldsymbol{A}_m^{-1}\boldsymbol{g}_m$. If they contain any negative elements (which would violate the constraint from (5.5)), continue with the next n

e) Determine the fitness f_n of the current sample by computing its leave-one-out (LOO) test error [Chen et al., 2004]

3. Admit the optimal sample $\hat{n} = \operatorname{argmin}_n f_n$ into $\mathcal{S}$, re-compute the corresponding updates of $\boldsymbol{W}_m$ and $\boldsymbol{A}_m$ and continue with the next m if $f_{\hat{n}}$ decreases the previous m-iteration's LOO score, otherwise terminate

4. Normalize the original weights to meet (5.6)

The memory requirement for this algorithm is $\mathcal{O}(3Sm + m^2/2)$ for the candidate $\boldsymbol{W}_m$, $\boldsymbol{A}_m$ and $\boldsymbol{Q}_m^\dagger$matrices needed in step 2 and the old $\boldsymbol{Q}_{m-1}^\dagger$ matrix which needs to be saved until the optimal candidate has been chosen in step 3. Since our algorithm construct the solution in a forward-selection manner, it requires less runtime memory if $m < N/3$ (which we usually strive for). Also, since we can restrict the maximum size of the solution and consequently the memory requirements, we may now approximately solve PDF-estimation problems which may be too large for the MGS algorithm introduced in [Chen et al., 2004] whose memory requirements are $\mathcal{O}(SN + mN)$, or for the support-vector method presented in [Vapnik and Mukherjee, 2000] whose memory requirements are $\mathcal{O}(SN + N^2)$ in the strict ϵ-insensitive case.

5.5 Experiments

To assess the performance of our proposed algorithm, experiments on three problems were performed. We compared the proposed method with classical density estimation procedures, i. e., Gaussian Mixture Models (GMMs) and Parzen windows, and the ϵ-SVD method without slack variables. Throughout all experiments, the normalized Gaussian kernel (5.10) with integral

$$\begin{aligned} q(\boldsymbol{x}, \boldsymbol{y}) &= \int_{-\infty}^{\boldsymbol{x}} k_{\text{ngauss}}(\boldsymbol{t}, \boldsymbol{y}) \,\mathrm{d}\,\boldsymbol{t} \\ &= \prod_{d=1}^{D} \left(1 + \operatorname{erf}\left(\frac{\|x_{(d)} - y_{(d)}\|}{\sqrt{2}\sigma}\right)\right) \end{aligned} \tag{5.48}$$

was used. This kernel was chosen because it additionally satisfies Mercer's conditions [Mercer, 1909] which enables comparison of our method with the ϵ-SVD procedure. It was assumed that all kernels in a model share the same covariance σ^2.

GMMs with diagonal covariance matrices were trained using the Expectation-Maximization (EM) algorithm. Initialization of the EM algorithm was performed using k-means clustering, which in turn had been randomly seeded. ϵ-SVD models and the proposed OROLS density models were trained using naive ECDF sampling (ECDF only sampled at the problem's sample points) and enhanced ECDF sampling (ECDF sampled at additional jump points, cf. section 5.2) schemes. For the ϵ-SVD method no model selection was required as all the parameters including the kernel width are set automatically. Model selection procedures for the other methods are discussed individually for each data set.

Artificial 2D-Problem

The first one is an artificial problem already considered in [Vapnik and Mukherjee, 2000] were 100 training sets of 60 samples each and a test set of 10000 samples are generated from the following density

$$\begin{aligned} p(\boldsymbol{x}) =& \frac{1}{4\pi}\exp\left(-\frac{(x_{(1)}-2)^2+(x_{(2)}-2)^2}{2}\right) \\ &+ \frac{0.35}{8}\exp(-0.7|x_{(1)}+2|-0.5|x_{(2)}+2|) \end{aligned} \tag{5.49}$$

As a baseline, GMMs with 2 and 4 centers were trained as described above. Parzen windows setup was straightforward. Next, ϵ-SVD models were trained using both naive ECDF sampling and enhanced ECDF sampling. Finally, the proposed algorithm was used to estimate another set of density models, again using both incarnations of ECDF sampling. For all the 7 sets of 100 density models the L_1-norm of the test errors $\boldsymbol{e}$ were computed on the 10000 sample test set. Selection of the suitable kernel width for Parzen windows and the OROLS models was performed by selecting the kernel width which produces the lowest average $L_1(\boldsymbol{e})$ over all 100 training sets.

Figure 5.2 compares the achievements of the different algorithms showing clearly that the proposed algorithm performs superior to all other algorithms if used with enhanced ECDF sampling and at least comparable with the GMMs if used with naive ECDF sampling. It can also be seen clearly that the proposed algorithm provides better results than the ϵ-SVD method, regardless of the ECDF sampling used. Table 5.1 summarizes some details of the trained models. It also supports the superiority of our proposed algorithm: While it can be observed that, in general, using enhanced ECDF sampling results in a larger number m of samples in the solution, this number does not increase as much for the OROLS algorithm as for the ϵ-SVD algorithm, yet much more accurate models are achieved using the OROLS algorithm. Furthermore, it can be seen that as, on average, the OROLS algorithm selects about 22% of the problem samples using the naive ECDF, and about 29% of the problem samples using the enhanced ECDF, the computation of the solution using OROLS is less

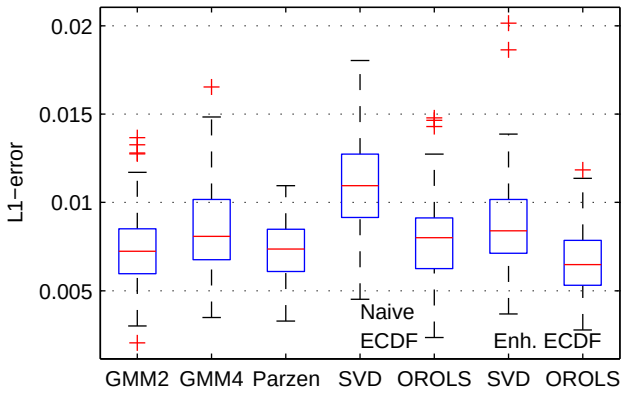


Figure 5.2: Comparison of the L_1-norm of the errors committed by different density estimation algorithms on the artificial 2D-problem. The boxplot is showing median, 0.25/0.75 and 0.05/0.95 quantiles as well as outliers.

Model	**ECDF**	**Kernel Width**	**# centers**
Parzen	—	0.6	100
ϵ-SVD	Naive	1.34($\pm$0.27)	13.1($\pm$3.7)
	Enhanced	0.82($\pm$0.26)	26.1($\pm$9.1)
OROLS	Naive	1.0	13.4($\pm$2.3)
	Enhanced	1.0	17.6($\pm$3.3)

Table 5.1: Details of the models for the artificial 2D-problem.

memory-consuming than ϵ-SVD and MGS-OLS based density estimation procedures, cf. section 5.4.1.

Ripley data set

This second experiment is based on a data set considered in [Ripley, 1996]. It consists of two labels in two dimensions with 125 training samples and 500 testing samples each. We trained GMMs with 2 and 4 components, Parzen windows, and ϵ-SVD and OROLS models, each of the latter two with naive and enhanced ECDF sampling. We then assessed the classification error on the test set. As in the previous experiment, no parameter needed to be manually tuned for the ϵ-SVD algorithm, while the kernel width for Parzen windows and OROLS models were set according to test error. Assignment of a test sample $\boldsymbol{x}$ to a label $y \in \{-1, +1\}$ was

Model	ECDF	σ	# centers	err.
GMM2	—	—	2 / 2	9.1%
GMM4	—	—	4 / 4	9.2%
Parzen	—	0.28	125 / 125	8.1%
ϵ-SVD	Naive	0.157	17 / 12	10.0%
	Enhanced	0.075	77 / 51	9.8%
OROLS	Naive	0.17	13 / 6	8.9%
	Enhanced	0.25	7 / 7	8.5%

Table 5.2: Details and classification error rates of the density models on the Ripley data set. The number of solution centers is reported individually for each label.

performed using maximum a-posteriori classification with Bayes' rule:

$$\begin{aligned} \hat{y} = \operatorname*{argmax}_y P(y|\boldsymbol{x}) &= \operatorname*{argmax}_y \frac{p(\boldsymbol{x}|y)P(y)}{p(\boldsymbol{x})} \\ &= \operatorname*{argmax}_y p(\boldsymbol{x}|y)P(y) \end{aligned} \tag{5.50}$$

where, in this case, $P(y = +1) = P(y = -1) = 0.5$.

The results of the experiment are displayed in table 5.2. It can easily be seen that our proposed OROLS algorithm has a lower test error than the ϵ-SVD algorithm while needing considerably fewer samples in the solution. On the other hand, it can be observed that using enhanced ECDF sampling for the training of the models improves the test error regardless of the training algorithm. In [Tipping, 2001] it is reported that the theoretical Bayes error rate of this problem is about 8%, while the paper reports error rates of 10.6% for standard discriminatory SVM with Gaussian kernel and 38 solution vectors, and an error rate of 9.3% for the Relevance Vector Machine with Gaussian kernel and four solution vectors. Compared to the previous artificial 2D-problem, the advantage of the OROLS algorithm with respect to memory requirements is even more striking: In the worst case, 13 samples are selected, which is about 1/10 of the samples — much less than the rough 1/3 of the samples which would make competing ϵ-SVD and MGS-OLS algorithms break even with the OROLS algorithm.

Thyroid data set

The thyroid disease data set is part of the UCI repository of machine learning problems [Asuncion and Newman, 2007]. It was thoroughly studied, e. g., in [Mika et al., 1999]. It is a two-label problem with 5-dimensional feature vectors. For our experiments, the problem setting from [Mika et al., 1999] was used, where 100 realizations of training sets with 140 samples each and test sets of 75 samples each

had been generated. The 140 training samples of each training set consisted of roughly 40 positively labeled and 100 negatively labeled samples.

In contrast to the previous experiments, model selection, i. e., selection of the number of components of the GMMs and the kernel width for Parzen windows and OROLS models, was performed using thorough cross-validation similar to the procedure described in [Mika et al., 1999]. Models with varying parameters were trained on each of the first 5 training sets. The models' performance was assessed on the 5 cross-validation test sets which were concatenations of the 4 respective unused training sets. The parameters for the complete evaluation of the 100 realizations were then determined by averaging the parameters leading to the best results on the cross-validation sets.

The use of the enhanced ECDF sampling scheme was modified for this experiment. As stated in section 5.2, the number of jumps of the ECDF is problem-dependent, but usually much larger than the number of problem samples. For the problem at hand, the about 100 negatively labeled training samples would have generated an ECDF with roughly 1.5 million jumps. Instead of using all of these points, we conducted experiments where a maximum of 5, 50, and 500 random points was chosen from each level of the ECDF; we denote these experiments with Enh-5, Enh-50, and Enh-500, respectively.

Classification of the test samples was also performed using (5.50) with the labels' priors estimated from the relative frequency of the labels in the training set. Figure 5.3 compares GMMs with 4 components and Parzen windows with ϵ-SVD and OROLS models trained using different ECDF sampling schemes, as described above. It can be seen that the accuracy of the OROLS models improves as the number of ECDF samples increases, which supports the concept that naive sampling as well as under-sampling the enhanced ECDF significantly degrades the performance of the so-estimated density models. If we consider a maximum of 500 samples per ECDF level sufficient then it shows that the sparse OROLS kernel density models greatly outperform the best GMM models; however, the accuracy of the Parzen windows model, obtained at the cost of using all samples, is not met. Furthermore, OROLS models again outperform ϵ-SVD models in all cases while producing sparser models. Through the obtained sparsity the OROLS algorithm takes advantage of its lower memory requirements compared to ϵ-SVD as less than one third of the samples is used for the solutions.

It might be surprising at first to see the performance of the ϵ-SVD decrease (after an initial improvement over naive ECDF sampling) with increasing number of ECDF samples, especially in the Enh-500 setting. This can be attributed to the automatic tuning of the kernel width. The kernel width selected is the largest feasible, however, it still results in overfitted ϵ-SVD models.

Our algorithm also compares favorably to other methods, even if performance is not en par with discriminative models such as SVM (classification error $4.8 \pm 2.2\%$) or kernel fisher discriminant (classification error $4.2 \pm 2.1\%$) [Mika et al., 1999]. On the

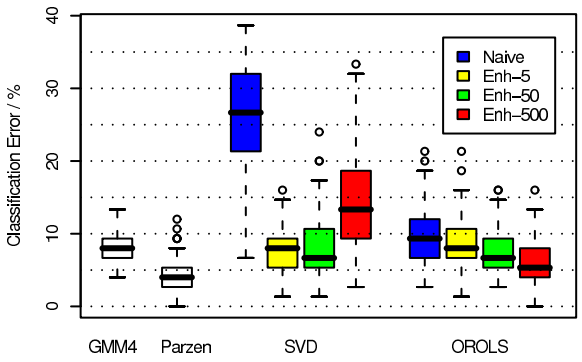


Figure 5.3: Classification errors on the Thyroid data set performed by different probability density models using different samplings from the ECDF for training.

Model	ECDF	Kernel Width	# centers
GMM4	—	—	4 / 4
Parzen	—	0.27	≈ 100 / ≈ 40
ϵ-SVD	Naive	2.85 ± 1.51	43.5 / 9.2
	Enh-5	0.067 ± 0.015	35.2 / 34.6
	Enh-50	0.052 ± 0.014	52.3 / 38.0
	Enh-500	0.031 ± 0.01	81.6 / 40.0
OROLS	Naive	0.17	12.9 / 10.6
	Enh-5	0.07	16.1 / 12.9
	Enh-50	0.07	17.7 / 13.4
	Enh-500	0.07	18.1 / 14.6

Table 5.3: Details of the kernel density models on the Thyroid data set. The number of centers is reported individually for each label.

other hand, the well-known C4.5 algorithm performs much worse for classification with an error of about 10.2% [Webb, 1996].

5.6 Discussion

We have presented an algorithm which constructs sparse probability density models from training sample in a memory-efficient and accurate way. It outperforms standard, conventional models like EM-trained Gaussian mixture models while still producing sparse models. We have also shown the importance of sensible application of the ECDF in the training process in that a sampling at points additional to the original problem samples impressively improves the accuracy of the so-trained models.

However, the methodology of estimating kernel PDF models by regression onto the ECDF seems to be limited to rather low-dimensional problems like the ones considered in the experiments section. For high-dimensional PDF-estimation problems like in automatic speech recognition, the method does not seem to be suitable due to the complexity of the resulting ECDFs. Methods which either choose sensitive points or regions for the regression of the ECDF or which entirely avoid sampling the ECDF by "implicitly" regressing the ECDF would be needed.

6 Probability Density Function Models for Automatic Speech Recognition using Binary Discriminant Functions

In the previous chapter we have investigated and developed methods for training sparse kernel PDF models using the SRM inductive principle for regression of the ECDF. The methods showed promising results on problems of low dimensionality but are difficult, or even impossible, to apply to problems of greater dimensionality, like ASR.

Loosely speaking, PDFs are used to estimate where a label-conditional sample generator is located in the sample space $\mathbb{R}^D$. In this chapter we will investigate how nonlinear binary discriminant functions based on the SRM inductive principle, e. g., SVMs or KFDs revisited in chapter 3, can be used to characterize the location of a label in the sample space. By extracting estimates of posterior label probabilities from the discriminant functions we will by applying Bayes' rule arrive at another family of PDF models. These novel PDF estimates will then be compared against conventional GMMs on a common ASR problem.

6.1 Turning discriminants into probabilities

Assuming a binary problem setting which is solved using a discriminant function such as SVM or KFD, this section discusses procedures and algorithms to relate the output of the discriminant to label-conditional (emission) probabilities or to a-posteriori label probabilities.

6.1.1 Modeling the projections' probability density functions

Discriminants, such as Support Vector Machines (SVMs) or Kernel Fisher Discriminants (KFDs), may be viewed as the projection of the D-dimensional samples $\boldsymbol{x}$ into a 1-dimensional subspace where separation of the two labels' samples is optimal with respect to the discriminants' criteria. Probability density functions can, in general,

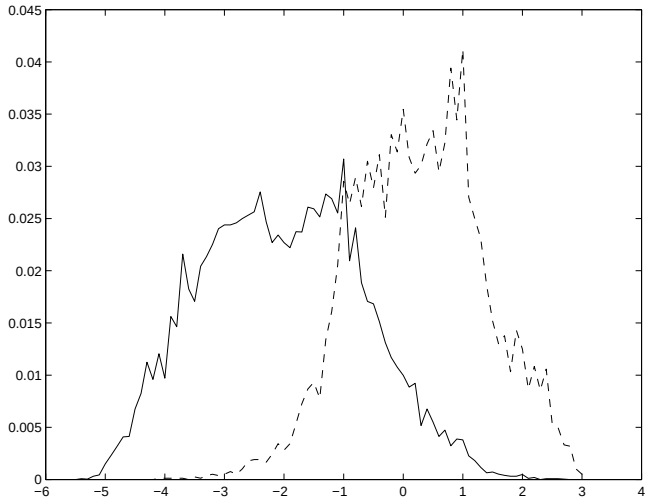


Figure 6.1: Sample histogram for $p(f(\boldsymbol{x})|y = \pm 1)$ of a linear SVM [Platt, 2000]. The solid line is $p(f(\boldsymbol{x})|y = -1)$, while the dashed line is $p(f(\boldsymbol{x})|y = 1)$. Notice the approximately exponential densities in the region of the overlap around -1.

be more easily and more robustly solved if the dimensionality of the problem space is decreased.

In the case of projections created using KFDs, the assumption is that the projections $f(\boldsymbol{x})$ of each label's samples follow a normal distribution $P(f(\boldsymbol{x})|y = \pm 1) = N(f(\boldsymbol{x})|\mu_\pm, \sigma_\pm)$ with means $\mu_\pm$ and standard-deviations $\sigma_\pm$ [Mika et al., 2001]. In the special case that due to constraints (3.57) the means of the distributions are known as $\mu_\pm = \pm 1$, all that remains is to estimate the normal projection densities' standard-deviations from data, e. g., the same data on which the KFD was trained, or a separate sample set.

6.1.2 Modeling posterior label probabilities

If the projections are obtained using SVMs it is not easy to model the projections' density functions as they do not seem to exhibit such appealing properties as KFDs' projections [Platt, 2000]. Instead of modeling the individual projections' PDFs it is also possible to model the posterior label probability $P(y = \pm 1|\boldsymbol{x}) = P(y = \pm 1|f(\boldsymbol{x}))$ of a sample $\boldsymbol{x}$ belonging to either of the two labels ± 1. This can be accomplished by replacing the sgn function in (3.37) (or, equivalently, in (3.47), (3.59)), which had been used to extract the sign of the projection as a label, with a suitable real-valued transfer function with value range $[0, 1]$.

In [Platt, 2000] it was (visually) observed that the projections of the two labels' samples exhibit exponential densities in the region of the overlap (cf. figure 6.1). If

we consider

$$p(f(\boldsymbol{x})|y=\pm 1) = e^{-a_\pm f(\boldsymbol{x})}, \quad a_\pm \in \mathbb{R}^+ \tag{6.1}$$

the posterior label probability $P(y=1|\boldsymbol{x})$ can be computed using Bayes' rule and marginalization:

$$\begin{aligned} P(y=1|\boldsymbol{x}) &= \frac{P(f(\boldsymbol{x})|y=1)P(y=1)}{P(f(\boldsymbol{x}))} \\ &= \frac{p(f(\boldsymbol{x})|y=1)P(y=1)}{p(f(\boldsymbol{x})|y=1)P(y=1)+p(f(\boldsymbol{x})|y=-1)P(y=-1)} \end{aligned} \tag{6.2}$$

recalling from (4.13) that the quotient $P(a)/P(b)$ may be substituted with $p(a)/p(b)$ in the case of continuous PDFs. Inserting (6.1) into (6.2) leads to

$$\begin{aligned} P(y=1|\boldsymbol{x}) &= \frac{e^{-a_+ f(\boldsymbol{x})}P(y=1)}{e^{-a_+ f(\boldsymbol{x})}P(y=1)+-a_- f(\boldsymbol{x})P(y=-1)} \\ &= \frac{e^{-a_+ f(\boldsymbol{x})+\ln P(y=1)}}{e^{-a_+ f(\boldsymbol{x})+\ln P(y=1)}+e^{-a_- f(\boldsymbol{x})+\ln P(y=-1)}} \\ &= \frac{1}{1+e^{(a_- -a_+)f(\boldsymbol{x})+\ln P(y=+1)-\ln P(y=-1)}} \\ &= \frac{1}{1+e^{A_+ f(\boldsymbol{x})+B_+}} \end{aligned} \tag{6.3}$$

with $A_+ = a_- - a_+$ and $B_+ = \ln P(y=+1) - \ln P(y=-1)$. For $P(y=-1|\boldsymbol{x})$ it follows

$$P(y=-1|\boldsymbol{x}) = 1 - P(y=1|\boldsymbol{x}) = \frac{1}{1+e^{A_- f(\boldsymbol{x})+B_-}} = \frac{1}{1+e^{-A_+ f(\boldsymbol{x})-B_+}} \tag{6.4}$$

In essence, what remains is to determine suitable parameters A and B. A procedure for this training based on maximizing a cross-entropy function is also proposed in [Platt, 2000]. An improved version of the algorithm with better numerical stability is presented in [Lin et al., 2003].

6.2 Pairwise coupling of binary classifiers

While section 6.1 discussed how to turn the projections of binary discriminants into probability measures (either label-conditional production probabilities or posterior label probabilities) it still needs to be clarified how the binary regimes are to be constructed and how they can be used together with the methods discussed in section 6.1 to estimate probability measures in a multi-label regime.

6.2.1 Simple strategy: one-versus-rest

A straight-forward approach is to discriminate each label against the union of the remaining labels which is called "one-versus-rest" (OVR). Using the OVR approach, there are C discriminants to be trained. The results obtained by applying probability moderation techniques as in sections 6.1.1 or 6.1.2 are easy to interpret as the original multi-label problem was not partitioned at all. However, a drawback of the OVR scheme is that every discriminant is trained on the complete training sample set, which may be large in automatic speech recognition, and that the training set may be badly im-balanced if there is a large number of labels involved. Nevertheless, the OVR scheme enjoys some popularity in practice [Rifkin and Klautau, 2004].

6.2.2 Smaller subproblems: one-versus-one

Another popular approach is to construct binary problems for each pair of two elementary labels. This method is called "one-versus-one" (OVO). Applying techniques for probability moderation as mentioned in section 6.1, estimates r_{ij} of the pairwise posterior label probabilities $P_{ij} = P(y = y_i | y \in \{y_i, y_j\}, f(\boldsymbol{x})))$ can be obtained. These pairwise estimates must be combined into a multi-class posterior label probability.

Voting

A very simple strategy to estimate the multi-label posterior probability of a label y_i from estimates of the pairwise posterior probabilities r_{ij} is to use the heuristic voting rule from [Knerr et al., 1990]:

$$P(y_i|\boldsymbol{x}) = \frac{2}{C(C-1)} \sum_{j=1, j\neq i}^{C} \Theta(r_{ij} - r_{ji}) \tag{6.5}$$

While this rule may seem very appealing due to its simplicity, the estimates of the multi-label probability obtained actually exhibit large errors [Wu et al., 2004]. We only consider the voting rule for completeness.

An exact formula

Taking the estimating r_{ij} as proxies for the true P_{ij}, Price et al. derived an exact formula for combining binary posterior probabilities resulting from the OVO binary regime into the posterior class probabilities of the generating multi-label problem [Price et al., 1995]. They assumed that each pattern belongs to only one class:

$$P\left(\bigcup_{i=1}^{C} y_i \,\middle|\, f(\boldsymbol{x})\right) = 1 \tag{6.6}$$

Denoting the union of the events y_i and y_j as $y_{ij} = (y = y_i \cup y = y_j)$ it follows that for any j

$$P\left(\bigcup_{i=1}^{C} y_i \middle| f(\boldsymbol{x})\right) = P\left(\bigcup_{i=1, i\neq j}^{C} y_{ij} \middle| f(\boldsymbol{x})\right) = 1 \tag{6.7}$$

Then, with the closed-form expression for the union of C events y_i

$$\begin{aligned} P\left(\bigcup_{i=1}^{C} y_i\right) = &\sum_{i=1}^{C} P(y_i) + \ldots + (-1)^{k-1} \sum_{i_1<\ldots<i_k}^{C} P(y_{i_1} \wedge \ldots \wedge y_{i_k}) \\ &+ \ldots + (-1)^{C-1} P(y_1 \wedge \ldots \wedge y_C) \end{aligned} \tag{6.8}$$

it follows from (6.7) that

$$\sum_{i=1, i\neq j}^{C} P(y_{ij}|f(\boldsymbol{x})) - (C-2)P(y_j|f(\boldsymbol{x})) = 1 \tag{6.9}$$

Using

$$P_{ij} = P(y_i|y_{ij}, f(\boldsymbol{x})) = \frac{P(y_i, y_{ij}, f(\boldsymbol{x}))}{P(y_{ij}, f(\boldsymbol{x}))} = \frac{P(y_i|f(\boldsymbol{x}))}{P(y_{ij}|f(\boldsymbol{x}))} \tag{6.10}$$

one finally obtains the unifying formula

$$P(y_i|f(\boldsymbol{x})) = \frac{1}{\sum_{j=1, j\neq i}^{C} \frac{1}{P_{ij}} - (C-2)} \tag{6.11}$$

Minimization of Kullback-Leibler divergence

Hastie and Tibshirani observed that for a set of estimates r_{ij} there does not actually exist a set of probabilities $P(y_i|f(\boldsymbol{x}))$ that is compatible with the estimates r_{ij} [Hastie and Tibshirani, 1998]. Consider the setting in the Bradley-Terry model for paired comparisons [Bradley and Terry, 1952, Bradley, 1954, 1955]:

$$P(y_i|y_i \vee y_j) = \frac{P(y_i)}{P(y_i) + P(y_j)} \tag{6.12}$$

$$\sum_{i=1}^{C} P(y_i) = 1 \tag{6.13}$$

This model requires that $C-1$ free parameters satisfy $C(C-1)/2$ constraints from (6.12) which will not yield a solution in general (see [Hastie and Tibshirani, 1998] for an example). Hastie and Tibshirani solve the problem by minimizing a Kullback-Leibler distance criterion to find approximations $\hat{r}_{ij} = \hat{P}(y_i)/(\hat{P}(y_i) + \hat{P}(y_j))$.

Assume that for each $i \neq j$ there are N_{ij} training observations with correspondingly estimated conditional probabilities r_{ij}. Rewriting (6.12) yields

$$\log r_{ij} = \log(P(y_i)) - \log(P(y_i) + P(y_j)) \tag{6.14}$$

The closeness criterion is the average weighted Kullback-Leibler distance between $\hat{r}_{ij}$ and r_{ij}:

$$l(\boldsymbol{P}) = \sum_{i<j} N_{ij} \left[r_{ij} \log \frac{r_{ij}}{\hat{r}_{ij}} + (1 - r_{ij}) \log \frac{1 - r_{ij}}{1 - \hat{r}_{ij}} \right] \tag{6.15}$$

which needs to be minimized with respect to $\boldsymbol{P}$. Therefore, the gradient equations are:

$$\sum_{j \neq i} N_{ij} \hat{r}_{ij} = \sum_{j \neq i} N_{ij} r_{ij} \quad i = 1, \ldots, C \tag{6.16}$$

which are used to iteratively compute the $\hat{P}$.

A simple approximation of the solution can be obtained by

$$\tilde{P}_i = \frac{2 \sum_{j \neq i} r_{ij}}{C(C-1)} \tag{6.17}$$

which can also be used to initialize the iterative procedure. Algorithm 3 summarizes the pairwise probability coupling procedure.

Algorithm 3 Pairwise coupling of probabilities according to [Hastie and Tibshirani, 1998]

Initialize $\hat{P}_i$ (e. g., according to (6.17))
repeat
  compute $\hat{r}_{ij}$
  $\forall i = 1, \ldots, C$: $\hat{P}_i \leftarrow \hat{p}_i \frac{\sum_{j \neq i} N_{ij} r_{ij}}{\sum_{j \neq i} N_{ij} \hat{r}_{ij}}$
  normalize: $\hat{P}_i \leftarrow \hat{P}_i / \sum_{i=1}^{C} \hat{P}_i$
until convergence

6.3 Setup of the hybrid system

After having discussed how to construct nonlinear binary discriminants in chapter 3, how to obtain pairwise posterior label probabilities in section 6.1, and how to turn these estimates into posterior class probabilities of a multi-label setting in section 6.2, we will see in this section how this stack of concepts can be used to construct a novel method of modeling HMMs' state-conditional emission probabilities. The method is applied to an ASR environment and evaluated on the *Resource Management (RM1)* corpus [Price et al., 1988].

6.3.1 Definition of labels

Since for the kernel binary classifiers such as SVMs or KFDs strict labels are required, we considered each state of each phoneme's HMM as one label. For the RM1 task, a set of 48 monophone HMMs with 3 states each and one short-pause HMM with only one state was used, resulting in a total of 145 state labels.

The generally more sophisticated method of using triphones was not considered in our experiments. Some considerations will be made in section 6.8.1.

6.3.2 Definition of samples

For our investigations we used the high-dimensional feature vectors produced by the feature extraction methods described in section 4.1 as the problem samples. Specifically, 12 Mel-frequence cepstral coefficients (MFCCs) and the energy component were augmented with first- and second-order temporal differences to form a 39-dimensional input space. The speech feature vectors were extracted from the speech signal Fourier transform using a Hamming window of 25ms duration and a shift of 10ms, resulting in 100 acoustic samples per second of speech.

The conversion of the acoustic data using these parameters results in a total of 987649 feature vectors for the 72 speaker independent training set (henceforth referred to as *72_indtr*).

6.3.3 Software

We used the *Hidden Markov Toolkit (HTK)* [Young et al., 2002] for feature extraction, training of conventional HMM-GMM models, and decoding for testing purposes. SVMs were trained using *SVMTorch* [Collobert and Bengio, 2001]. For KFD training, the greedy-sparse training procedure described in section 3.5.1 was used; the algorithm was implemented using a newly developed high-performance machine learning software library which will be described in chapter 7. Finally, the functionality of both SVM- and KFD-based emission PDF systems was provided to HTK using an external plug-in library to be described in section 7.6.

6.4 Training of emission PDFs based on kernel binary machines

Algorithm 4 summarizes the steps involved in training the new emission PDF models based on SVMs or KFDs, which we will now collectively call kernel binary machines Kernel Binary Machines (KBMs). Each training step needs a number of parameters which will influence the performance of the resulting KBM-based PDFs. The choice of systems or parameters in each step will be considered independently, i. e., we use

a forward-optimization procedure where in each training step systems or parameters will be selected based on a certain criterion. The systems and parameters are then kept fixed for the subsequent training steps.

Therefore, we will discuss the procedure and influence of each training step separately. The evaluation of some system or parameter choices was carried out by measuring the performance of the KBM-PDF model in the full ASR system. The testing procedure will be discussed later in section 6.5; however, we will consider it given for the time being.

Algorithm 4 How to train an ASR system employing binary discriminants as building blocks of the acoustic model

- Train conventional GMM-HMM-based ASR-system
- Perform forced HMM-state-level time alignment of the training data using above GMM-HMM system
- (Optionally) Pre-process data, save pre-processing parameters
- Train binary discriminants (SVM or KFD) in OVR or OVO fashion
- Estimate labels' prior probabilities, possibly on separate data set
- Train probabilistic outputs of binary discriminants (either class-conditional projection densities or pairwise posterior class-probabilities), possibly on separate data set

6.4.1 Initial ASR-system and time-alignment of training data

In order to train these KBMs we need to have labeled data on the frame level, i. e., each frame from the training set must be assigned one of the labels constructed in section 6.3.1. This is achieved by first training an initial ASR system using GMMs for HMM-state emission probabilities. This initial ASR system is then used to compute a time-alignment of the HMM-states (section 4.8.1). Using the time-stamps of a label contained in the resulting label files, corresponding speech samples can be extracted from the parameter files. The parameterized speech samples together with their corresponding labels are then merged into a file suitable for use with SVM or KFD training tools.

In [Schafföner et al., 2003] it was shown that the time-alignment is not strongly influenced by the quality of the initial ASR system, i. e., "wrong" decisions with respect to the time-borders of HMM-states are very unlikely due to the recognition network being restricted to the known text of the sample sentences. Therefore, it is not necessary to use the best available ASR-system to construct the time-alignment as long as the ASR-system is not extremely under- or over-fitted.

6.4.2 Optional pre-processing of the training data

When examining the examples of commonly used kernels from (3.43), (3.44), (3.45) it can be seen that a conventional dot-product in the input space is used to "reduce" the vectors to a scalar. This exposes the computation of the kernel functions to a scaling problem, i. e., if, on average, the magnitudes of the components of the vectors differ greatly, the vectors' components with the largest average magnitude will largely determine the value of the kernel function while the influence of the components with a smaller expected magnitudes will be suppressed. Therefore, data need to pre-processed in order to evenly distribute the influence of each feature-vector component. The utility of preprocessing the data has also been described in [Burges, 1998].

There exists a number of possible methods for preprocessing the data to circumvent the scaling problem:

- Shifting to zero mean and subsequent scaling to unit variance of each component. Mean and variance can be reliably estimated given the large number of samples in speech recognition problems. However, some (mild) assumption about the distribution of the components' values are made in this case.

- Similar in fashion to the previous procedure: Shifting to zero mean, conducting a Principal Components Analysis (PCA) and transforming data with the PCA's orthonormal matrix. Here also, some (mild) assumption about the distribution of the data is made.

- Determine smallest and largest value of each component, shift value by mean of both, and scale to $[-1, 1]$. This method does not make any assumptions about the distribution of the components' values. This method can be made more statistically robust by estimating the minimum and maximum of each component at the 0.05 and 0.95 quantiles of the distributions, respectively.

Here, we only investigate the first method mentioned above. To limit the effort of experiments, only the results using SVMs in the OVO binary regime are considered here. Without pre-scaling the feature vectors, the best recognition rate obtained was 91.3% accuracy, whereas pre-scaling the feature vectors to component-wise zero-mean and unit-variance boosted the recognition rate to 94.6% accuracy, a relative error reduction of 38%. Since the pre-scaling step is so cheap and has such great influence on the recognition accuracy, this step was mandatory in all further experiments assessing the framework.

6.4.3 Choice of binary regimes

Using the (preprocessed) feature vectors KBMs are trained in either an OVO or OVR fashion. Table 6.1 compares these two binary regimes using prior knowledge. From

Property	**OVR**	**OVO**
# KBMs	**C**	$C(C-1)/2$
avg. training set sizes	N	**2N/C**
avg. training set balance	N/C vs. $N - N/C$	**N/C** vs. **N/C**
expected cost of recall	**small (1 KBM/label)**	large ($C-1$ KBMs/label)

Table 6.1: A-priori comparison of the OVR and OVO binary regimes. More appealing properties are typeset in boldface.

the table it is not clear which method should be preferred a-priori. The OVR regime has the advantage that multi-label posterior probabilities are directly available without the need for further processing as described in section 6.2.2. However, OVR has the disadvantage of requiring the samples of all labels for training, and that the training procedure is further complicated by a great imbalance of the training samples — on average, N/C samples of the one (positive) class in question are discriminated against $N - N/C$ samples of the (negative) pool of the remaining labels [Shin and Cho, 2003]. Additionally, the resulting binary problems may become prohibitively large and, therefore, impractical to be trained efficiently.

The competing OVO regime relaxes some of the problems of the OVR regime. The training sets are more balanced with, on average, N/C samples for each of the two labels involved in one discriminant. Also, the overall size of the binary problems is reduced to an average of $2N/C$ which seems more manageable in practice. The drawback of the OVO regime is that the number of discriminants grows to $C(C-1)/2$, and that an additional processing step for combining pairwise posterior label probabilities into the multi-label posterior probability is needed.

In [Krüger et al., 2005b] we compared the influence of the choice of binary regime on the recognition rate. As with the preprocessing step, the experiments were only carried out using SVMs as binary discriminants. A number of limitations were put onto the training procedure. First, when applying the OVR binary regime, it was not possible to handle the entire training set, as expected. Instead, a random subset of 1/5 and 1/10 was used for comparing the binary regimes. Secondly, there is, strictly speaking, one set of "optimal" hyper-parameters (kernel parameters, regularization tradeoff parameter) for *each* kernel machine; however, the parameters were tied for all machines to keep training and parameter selection efforts in a manageable range, see section 6.4.4.

Table 6.2 summarizes the data of the comparison. As can be seen, the binary regime does not dramatically influence the achievable accuracy, though the OVO regime performs slightly better in both cases where comparison with OVR was possible. Furthermore, the OVO regime extends the limit of manageable training sample sizes. Therefore, for further experiments only the OVO binary regime was considered.

	OVR	OVO
using 1/10 of training set	89.34%	89.73%
using 1/5 of the training set	92.07%	92.19%
using full training set	—	94.10%

Table 6.2: Comparison of the influence of the binary regime on the recognition rates [Krüger et al., 2005b].

6.4.4 Choice of hyper-parameters of kernel binary discriminants

For both SVM and KFD training, there are a number of hyper-parameters which must be set prior to training. These are

- for SVM and KFD:
  - the type of kernel,
  - the kernel parameters (i.e., , the variance in the case of the Gaussian kernel, the integer exponent in the case of the polynomial kernel),
  - the regularization (capacity control) parameter,
- for KFD only:
  - the maximum number of terms in the sparse approximation.

Throughout all experiments, the Gaussian kernel (3.44) was used as it was observed that the choice of kernel type is not the most influencing part when building a KBM [Burges, 1998]. There exist a number of methods to set the kernel parameters and the regularization parameter (semi-)automatically [Tsang et al., 2005, Schölkopf and Smola, 2001, Centeno and Lawrence, 2006], and there are also measures to evaluate the optimal number of terms in the sparse approximation of the KFD, e. g., the leave-one-out score [Chen et al., 2003] or the minimum description length score [Hansen and Yu, 2001].

In the KFD case, the relative improvements of the objective values of the 5 most recent selection steps were checked against a threshold value. Forward selection was stopped if all of these relative improvements had fallen below this threshold.

In order to get a feeling for the influence of these parameters on the performance of the speech recognition system as a whole, a conventional cross-validation procedure using the development test set was followed. The fully estimated emission PDF model (including the probabilistic outputs and prior label probabilities to be described in section 6.4.5) was plugged into the HMM, and the recognition performance of this entire system was measured on the development dataset. Still, tuning two or three parameters on this much data in a grid-search fashion would have been prohibitively costly, so procedures as described in algorithms 5 and 6 were adopted.

Algorithm 5 Tuning procedure for SVMs' hyper-parameters

initialize $C = 200$ {default value of *SVMTorch* software}
initialize range of kernel parameters and regularization coefficients
repeat
  for all kernel parameters in the given range with appropriate discretization **do**
    train SVMs with common kernel parameter and constant prior regularization parameter
    measure recognition performance on a development dataset
  end for
  set kernel parameter fixed at optimal value
  for all regularization coefficients in the given range with appropriate discretization **do**
    train SVMs with common regularization coefficients and constant prior kernel parameter
    measure recognition performance on a development dataset
  end for
  narrow parameter ranges
until convergence

6.4.5 Estimation of probabilistic outputs of binary kernel discriminants and prior label probabilities

After the kernel binary discriminants have been trained, it is necessary to estimate the parameters of the probabilistic output procedures (either normal probability densities or the sigmoid label-posterior function). Since this step is independent of the discriminant training procedures, one may use a separate data set in order to decrease over-fitting onto the training data. This, however, was not pursued for the experiments conducted here; instead, the training set was re-used.

For the projection normal probability densities used with KFDs, the means and variances were estimated using the standard averaging estimators

$$\mu_{\pm} = \frac{1}{N} \sum_{n=1}^{N} f(\boldsymbol{x}|y = \pm 1) \tag{6.18}$$

$$\sigma_{\pm}^2 = \frac{1}{N-1} \sum_{n=1}^{N} (f(\boldsymbol{x}|y = \pm 1) - \mu_{\pm})^2. \tag{6.19}$$

The two parameters of the sigmoid function used with SVMs were estimated using the numerically more stable algorithm from [Lin et al., 2003] instead of the original one found in [Platt, 2000].

Algorithm 6 Tuning procedure for KFDs' hyper-parameters

initialize maximum size m of solutions to a relatively small value
initialize range of kernel parameters and regularization coefficients
repeat
  for all kernel parameters in the given range with appropriate discretization **do**
    train KFDs with common settings
    measure recognition performance on a development dataset
  end for
  set kernel parameter fixed at optimal value
  for all regularization coefficients in the given range with appropriate discretization **do**
    train KFDs with common settings
    measure recognition performance on a development dataset
  end for
  narrow parameter ranges
until convergence
keep kernel parameter fixed
repeat
  scale maximum size m by a factor g and starting regularization coefficient by $1/g$
  choose range of regularization coefficients
  for all regularization coefficients in the given range with appropriate discretization **do**
    train KFDs with common settings
    measure recognition performance on a development dataset
  end for
until convergence

Prior probabilities for the multi-label and respective binary settings were estimated by counting the number of occurrences of a label relative to the number of samples. Prior probabilities were computed and stored even if they were not strictly needed for the speech recognition problem (cf. algorithm 7).

6.5 Testing the hybrid speech recognition system

Algorithm 7 schematically describes the course of computations when obtaining label-conditional PDF values (or proxies thereof) in the case of automatic speech recognition. Fortunately, there are no parameters within the new acoustic models to be tuned for recognition. However, there are a few heuristic parameters inside the speech decoder to be adjusted, most notably the word-insertion probability and the language model scale. In order to not complicate things further, to decrease experimental effort and to keep conditions as similar as possible for comparison with conventional GMM-based emission PDF models, the same values as determined optimal for GMM-based PDF models, i. e., a word insertion log probability of 10.5 and a language model scale factor of 5 [Schulze, 2005], were used for the proposed KBM-based PDF models.

6.6 Comparing the generalization performance of GMM- and KBM-based PDF models

As mentioned before, applying the SRM inductive principle (through the use of, e. g., SVMs or KFDs) instead of the ERM inductive principle employed for training GMMs with the Maximum-Likelihood cost function is expected to increase the generalization performance especially in the case of small training set sizes. Preliminary results indicating the validity of the above assumption were first published in [Andelic et al., 2004, 2005, Krüger et al., 2005a,b] with some partially incomplete experiments.

A thorough investigation was performed in [Schafföner et al., 2006]. The experiments were carried out on the Resource Management (RM1) speech recognition task [Price et al., 1988]. The RM1 task consists of about 2.8 hours of transcribed continuous speech training material, and an evaluation recognition problem containing about 1000 words. The speech corpus is accompanied by a development test set for tuning parameters of the decoding process.

To assess the performance of different acoustic models depending on the training sample sizes, random samples of 90, 180, 360, 720, and 1440 sentences were taken from the original complete training set, corresponding to 1/32, 1/16, 1/8, 1/4 and 1/2 of the complete 2880 sentences. The random sampling was repeated 10 times

Algorithm 7 How to test an ASR system employing binary discriminants as building blocks of the acoustic model

For each sample $\boldsymbol{x}$ and hypothesis label y presented by the recognizer:
(Optionally) Pre-process sample with previously saved parameters
if OVR **then**
  if projection densities **then**
    directly compute the value of projection density of the y-label as the emission probability
  else {binary posterior label probabilities}
    compute posterior probability of label y
    compute emission probability using Bayes' rule and the label's prior probability
  end if
else {OVO}
  if projection densities **then**
    compute the value of the projection densities of using all the binary discriminants involved with label y
    compute pairwise/binary posterior probability using Bayes' rule and the labels' prior probabilities in the binary setting
  else {binary posterior class-probabilities}
    directly compute the pairwise/binary posterior label probability
  end if
  merge all the binary posterior label probabilities into the multi-label posterior probability using (6.11)
  compute emission probability using Bayes' rule and the label's prior probability
end if

for each sample size in order to gain some statistical significance for the experiments to be conducted, resulting in a total of 51 training scenarios.

The reference baseline system using GMM-based acoustic model was trained with different numbers of GMM component using the 51 training sets. For more thorough comparison, monophone and triphone GMM acoustic models were considered. Then the recognition accuracy was measured on the *OCT89* development dataset using the resulting acoustic models, and the GMMs determined optimal on the development test set were then used to measure the recognition accuracy on the separate *FEB89* evaluation dataset. Table 6.3 lists the number of GMM-components together with the achieved evaluation recognition rates.

HMM	Subset	Recognition Accuracy			avg. # mixt.
		min	avg	max	
Mono	Full		**93.8**		16
	1/2	91.45	**92.36**	92.78	15.1
	1/4	89.26	**90.46**	91.25	14.1
	1/8	84.26	**86.15**	87.62	7.6
	1/16	78.45	**80.34**	82.66	4.5
	1/32	70.40	**71.91**	74.00	2.9
XwrdTri	Full		**96.8**		8
	1/2	94.61	**95.19**	95.70	6.7
	1/4	91.92	**92.94**	93.71	6.3
	1/8	85.86	**88.23**	89.57	6.3
	1/16	78.64	**81.78**	83.83	4.2
	1/32	68.22	**70.77**	72.51	2.7

Table 6.3: Baseline results on the *FEB89* evaluation dataset using the complete and randomly-sampled training set for monophone and crossword-triphone HMMs with GMMs as emission PDFs.

The competing KBM-based PDF models were only trained for monophone HMMs, as described in section 6.3.1. To keep experimental effort in a reasonable range, the SVM- and KFD-based acoustic models were only trained on the 1/32, 1/8 and the full training data sets. Additionally, the optimal hyper-parameters were determined on only one of the 1/32 and 1/8 training data sets, and then re-used for the other training data sets of the respective size. The *OCT89*-optimal GMM models were used to compute a state-time-alignment of the training data, and data were preprocessed with parameters estimated separately from each training selection. The transition probability matrices were kept fixed for all PDF models within each training dataset. For comparing the achievable recognition rates, this procedure may have even put the KBM-based acoustic models into a slightly disadvantageous position as the GMM-based acoustic models' number of GMM-components had been tuned individually for

KBM	Selection	Kernel Stddev.	Regul.	Max. solution size
SVM	Full	9.2	200	—
	1/8	10.5	20	—
	1/32	11.2	5	—
KFD	Full	6.2	0.8	100
	1/8	6.5	2	400
	1/32	7.0	5	600

Table 6.4: Parameters of SVMs and KFDs in the emission PDF models.

each training data set. The parameters of the SVMs and KFDs are listed in table 6.4.

Figure 6.2 gives a first impression of the performance of KBM-based vs. GMM-based PDF models. The first conclusion to be drawn is that in the monophone setting KBM-based PDF models outperform GMM-based PDF models regardless of the size of training data sets. Secondly, for small sample sizes, as in the 1/32 and 1/8 settings, the simple monophone HMMs using KBM-based emission probabilities even outperform the triphone HMM-GMM models which are otherwise almost always superior to monophone HMMs. Only in the full training set scenario can the triphone HMM-GMM models beat HMM-KBM models. Comparing the influence of the training set sizes it can be clearly seen that the convergence behavior of KBM-based emission probabilities is much better than that of GMM-based emission probabilities. The tendency of the KFD-based emission probabilities to perform worse than the SVM-based emission probabilities will be investigated in section 6.7. Tables 6.5 and 6.6 give more details about the recognition performance using KFD- and SVM-based emission probabilities.

Model	Subset	Recognition Accuracy		
		min	avg	max
MonoKFD	Full		**94.21**	
	1/8	88.79	**89.41**	90.33
	1/32	76.65	**78.56**	80.48
MonoSVM	Full		**94.58**	
	1/8	89.38	**89.94**	90.98
	1/32	78.72	**81.13**	82.74

Table 6.5: Recognition results for monophone HMMs using KBM-based emission PDFs

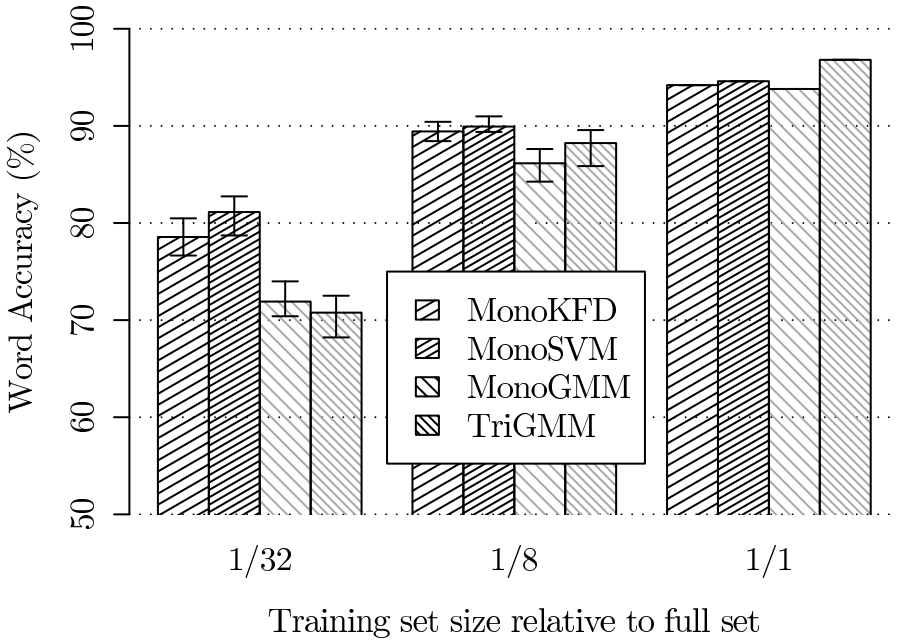


Figure 6.2: Different PDF models vs. different training set sizes

Subset size	MonoKFD	MonoSVM
1/1	6.5%	12.9%
1/8	23.5%	27.4%
1/32	23.7%	32.8%

Table 6.6: Average relative gains in Word Error Rate using KBM-based PDFs compared to GMMs

6.7 Properties of the KBM-based acoustic models

In order to better understand the differences in performance of the KBM-based PDF models depending on the choice of SVM or KFD, we investigated some second-order characteristics of the systems, i. e., sparsity of the models and quality of projections of the KFD.

6.7.1 Achieved sparsity

While the most important results are the superior recognition accuracy rates discussed in section 6.6, the sparsity of the underlying SVMs or KFDs can be used to characterize how difficult (or, in other words, demanding in terms of necessary samples) it was for the systems to produce accurate PDF values. For comparison we will discuss one experiment from the 1/8 series of trainings. We counted the number of

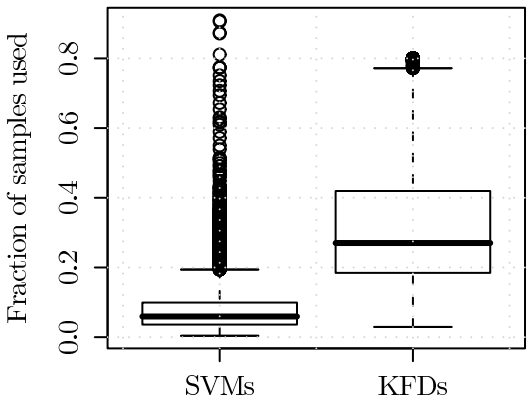


Figure 6.3: Boxplot comparison of the fraction of samples used in KFDs and SVMs

solution vectors in each of the 10440 KBMs and related it to the original number of training samples. Figure 6.3 shows the results. First of all, we can see that SVMs generate quite sparse models, with the median at 5.9%, a mean of 8.1%, and three quarters of the SVMs below 10% of support vectors.

It is not surprising that KFDs are less sparse than SVMs since KFDs are not sparse by nature. We had, however, been employing the greedy-sparse KFD approximation algorithm. The problem here is that the maximum number of samples to select was equal for all KFDs to be trained even though the size of the training sets differed greatly among all KFDs. In order to enforce at least some sparsity, an upper bound on the fraction of selectable samples had been set to 0.8. Still, even though we needed fairly non-sparse KFDs to reach acceptable recognition accuracy rates, KFDs were almost always performing worse than their SVM counterparts. Other sources do not support the idea that sparse KFD approximations generally need to be more dense than corresponding SVMs in order to achieve similar or even better results in the case of pattern recognition [Mika et al., 1999, 2001, Mika, 2002], so another possible cause for the performance loss of KFDs with respect to SVMs will be investigated in the next section. At any rate, better selection termination criteria than the ones mentioned in section 3.5.1 should be investigated in this scenario.

6.7.2 Projections generated by KFDs

As mentioned in section 6.1.1 we assumed that the projections generated by KFDs are normally distributed for each of the two labels. In [Mika et al., 2001] this assumption was supported by visual inspection of the projections' histogram plots. Using the KFD systems constructed in section 6.6, a more reliable validation was conducted using goodness-of-fit tests. A goodness-of-fit test computes a test statistic which indicates how well a sample matches a hypothesis distribution. The test rejects the hypothesis with a certain confidence if the computed test statistic becomes

significant, i. e., it exceeds a threshold corresponding to the significance level.

A very important goodness-of-fit test has already been mentioned in section 5.3, i. e., the Kolmogorov-Smirnov (KS) test. The KS test compares a hypothesis distribution F_H with the ECDF of the data. Assuming a sorted sample $\boldsymbol{x}$ of N samples, the KS test statistic is defined as

$$D = \max_{1 \leq n \leq N} \left(F_H(x_{(n)}) - \frac{n-1}{N}, \frac{n}{N} - F_H(x_{(n)}) \right) \tag{6.20}$$

The KS test has some limitations. First of all, it tends to be more sensitive near the center of the distribution. Secondly, and more importantly, the hypothesis distribution must be fully specified, i. e., if parameters are estimated from the data, the critical region and threshold statistic values are not valid. As we are aiming to determine if the data (the KFD label-projections) follow a normal distribution whose parameters had been estimated from the same data, the KS test cannot be used here.

Therefore, we use the Anderson-Darling (AD) goodness-of-fit test [Stephens, 1974] instead of the KS test. It gives more weight to the tails of the distribution than does the KS test. Also, the AD test may be applied to testing hypothesis distributions whose parameters had been estimated from the data. Given a (sorted) sample $\boldsymbol{x}$ of N samples, the AD test statistic is defined as

$$A^2 = -N - S \tag{6.21}$$

with

$$S = \sum_{n=1}^{N} \frac{2n-1}{N} \left[\ln F_H(x_{(n)}) + \ln(1 - F_H(x_{(N+1-n)})) \right] \tag{6.22}$$

The test statistics A^2 must be adjusted by a sample-size dependent factor, resulting in the "adjusted A^2 statistic"

$$A^{2*} = A^2 \left(1 + \frac{0.75}{N} + \frac{2.25}{N^2} \right) \tag{6.23}$$

The null hypothesis F_H is rejected with 5% significance if, in the case of the null hypothesis being a normal distribution, the A^{2*} exceeds 0.752.

We picked the set of KFDs from one of the experiments conducted on the 1/8 training for experimental investigations. As the set consists of 10,440 binary KFDs, we had 20,880 projection distributions to compute and test for normality. Figure 6.4 shows the distribution of the AD test scores obtained on each of the projections by testing the fit of the distribution of the data with the hypothesis normal distributions, whose parameters had before been computed using the data. We can see clearly that with a significance of 5%, close to 80% of the projections are not normally distributed. The median of the sample, an estimate of the expected value

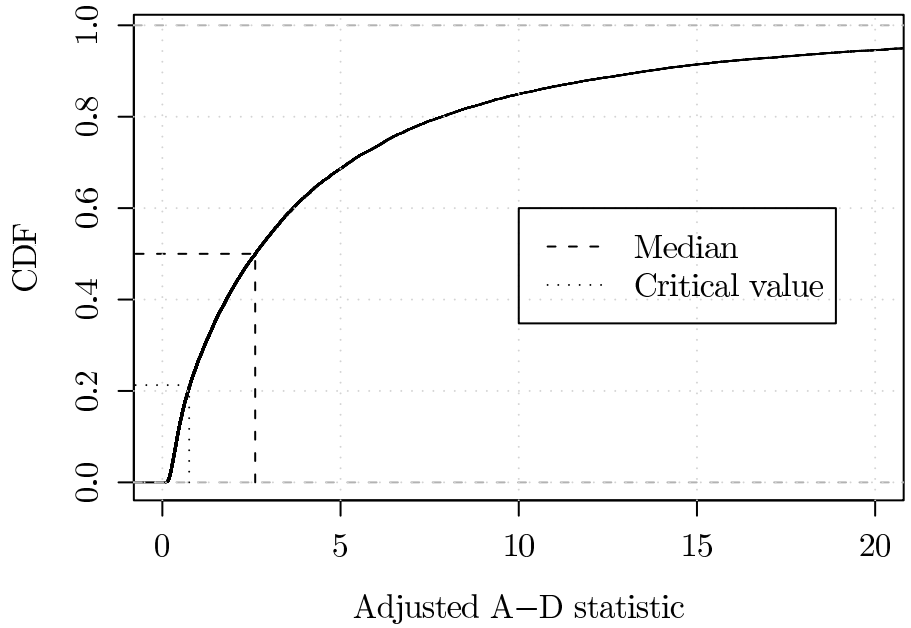


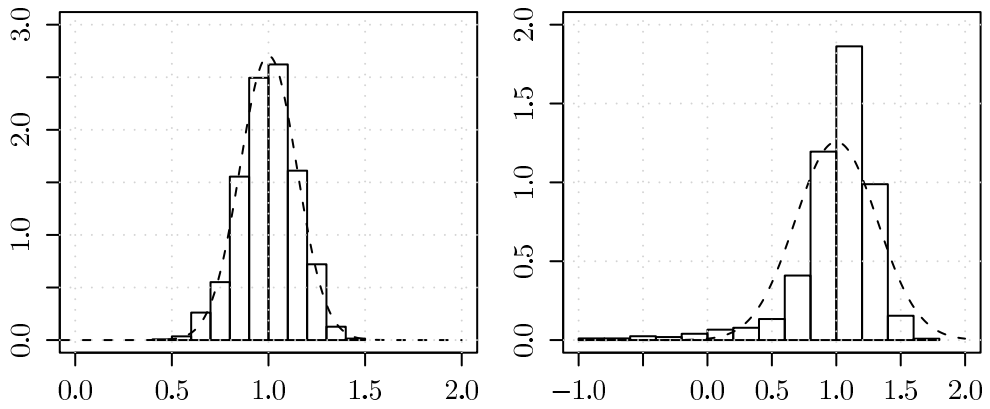
Figure 6.4: Distribution of Anderson-Darling test for normality statistic on a set of OVO-KFD-based data projections

of the true distribution, is about 2.6, which is much larger than the AD test's critical value. Figure 6.5 shows two examples of data projections, one whose normal-distribution hypothesis was accepted and one whose normal-distribution hypothesis was rejected.

We may, therefore, conclude that the assumption of normal projection densities is almost always violated. This may be one of the reasons why in section 6.6 the KFD-based PDF models were performing worse than SVM-based PDF models. Additionally, the violation of the normality assumption may have been the reason for the KFD-based PDF models requiring quite dense sample selections for acceptable recognition rates.

6.8 Discussion and open issues

In this chapter we have discussed how conditional PDF models in a multi-label scenario can be accurately estimated from binary discriminant functions. We have demonstrated the viability of the method using a prototypical application in ASR where great success in exceeding the performance of GMM-PDF models in otherwise same environments was achieved. We have also seen that greedy-sparse KFDs may be less suited for this endeavor as issues like setting the sparsity threshold of each sparse KFD, the assumption of normally distributed label-conditional projections, and possibly also the probabilistic greedy forward-selection scheme make



(a) Hypothesis accepted with $A^{2*} = 0.64$ (b) Hypothesis rejected with $A^{2*} = 16$

Figure 6.5: Two examples of label-conditional data projections computed by KFDs

KFDs models difficult to handle in the environment of ASR.

However, there remain a number of unanswered questions and unsolved problems. In the following, we will discuss some of them and present possible directions towards solutions.

6.8.1 Extension to triphone HMMs

One of the most striking limitations of the proposed system is the current limitation to monophone ASR systems. This limitation has practical rather than theoretical reasons. If we consider a tied-state cross-word triphone system, we would have, applying the label definition from section 6.3.1, around 4500 labels to discriminate. The OVR binary regime is still ruled out because the size of the individual problems is still N, but training set imbalance is even more severe. The OVO binary regime would result in extremely small binary problems, which might not be possible to sparsify, but the number of binary problems would be prohibitively large.

To overcome this problem, methods with a better, and possibly user-definable, tradeoff between number and size of binary problems should be applied instead of the extreme cases OVR and OVO. To this end, systems based on error-correcting output codes [Dietterich and Bakiri, 1995, Allwein et al., 2000] or on generalized Bradley-Terry models [Zadrozny, 2002, Huang et al., 2006], which allow arbitrary code matrices rather than fixed ones defined by OVR or OVO, should be investigated. Both approaches are a blend of OVR and OVO in that they allow arbitrary grouping of problem labels into discrimination labels, so that each problem label may, if considered at all, belong to one of the two discrimination labels. In [Zadrozny, 2002] it is reported that a number of pairwise comparisons as low as $15 \log_2 C$ may be sufficient to accurately model a multi-label problem. These approaches may not

only enable tackling PDF-modeling in the triphone-HMM scenario, but it may also be used to relax the system requirement in the monophone-HMM case.

6.8.2 Speaker adaptation

In section 6.6 we demonstrated the improved small-sample generalization performance of an HMM-ASR-system using KBM-based PDF models. However, conventional HMM-GMM ASR-systems use a technique called speaker adaptation where the parameters of the GMMs are adapted from the speaker-independent model to a speaker-dependent model. Methods for speaker-adaptation include Maximum-A-Posteriori (MAP) [Gauvain and Lee, 1994] and Maximum Likelihood Linear Regression (MLLR) [Leggetter and Woodland, 1995], which are applied either in a separate adaptation session or online during live recognition.

The proposed KBM-based PDF models are, so far, missing a speaker-adaptation feature. The tunable parts of the KBM-based PDF models would be the parameters of the probabilistic interpreters or the weight vector of the KBMs. Since adapting the weight vector may also mean promoting a sample weight of zero (a non-support vector) to a non-zero value (a support vector) or including new samples from the adaptation data, techniques like incremental support vector learning [Cauwenberghs and Poggio, 2001, Laskov et al., 2006] should be investigated.

7 Dedicated Software Solutions

In the previous two chapters the main focus of attention was put on algorithms and procedures for modeling class-conditional probability density functions. Experimental results were presented showing the validity of the proposed probability density function models, yet the effort of applying the new models to real-world data problems was neglected so far. Therefore, we will collect some figures showing how computationally challenging the proposed models are on large real-world problems such as automatic speech recognition. With these figures in mind we will construct a few examples showing that commonly used general-purpose software such as *MATLAB* cannot efficiently be used for conducting large-scale machine learning experiments, arguing why and how a novel machine learning library with a better tradeoff between versatility and performance was constructed.

7.1 Problem setting

In chapter 5 we constructed PDF models by regression of the models' distribution function onto the empirical cumulative distribution function using structural risk minimization. We observed in section 5.2 that this procedure may result in a large number of ECDF jumps if the dimensionality of the problem is large. As the loss between the problem's ECDF and the model's distribution function is measured at these jumps, the resulting design matrix becomes very large as the number of rows is determined by the number of (selected) ECDF jumps. In section 5.4.1 we presented an algorithm which can be used to obtain a sparse approximation by forward-selection and a thin update of the orthogonal decomposition of the design matrix. However, we were not concerned with the issues arising in implementing this algorithm in software.

The greedy-sparse KFD training procedure, which was introduced in section 3.5.1, was used in chapter 6 to construct sparse binary discriminant models. It has a few points in common with the sparse kernel PDF trainer mentioned before in that a sparse approximation of a regularized least squares regression solution is constructed using forward-selection of relevant samples, though a different selection criterion is used.

In the case of OVR discrimination, the QP problems were even larger than those considered for sparse kernel PDF estimation (roughly $1,000,000 \times 1,000,000$), which was too large even for the machine learning software library to be developed. How-

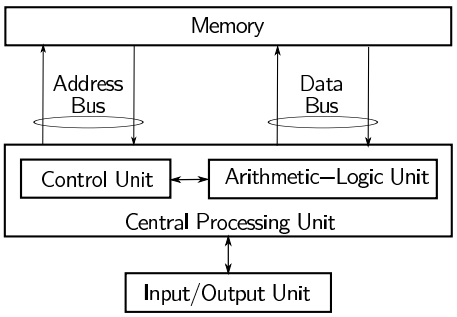


Figure 7.1: Von-Neumann computer architecture.

ever, in the case of OVO discrimination, the size of the QP problems was in a range that had a chance of being handled: On average, the QP problems were $N/C \times N/C \approx 7,000 \times 7,000$. While in the KFD-case the greedy forward-selection procedure drastically reduced the runtime-complexity on the algorithmic level, there were still, due to the large number of classes, 10,440 individual discriminants to be computed for each parameter set.

But regardless of the methodology and procedure used, there were vast amounts of kernel evaluations to be performed. Let us, as before, consider the OVO training in the ASR environment, resulting in 10,440 KBMs. Assuming 1000 selected samples per KFD, an average problem size of $7,000 \times 7,000$, a greedy selection procedure testing $\kappa = 59$ candidate samples in each iteration, we arrive at roughly $4.3 \cdot 10^{12}$ kernel evaluations as an upper bound if kernel evaluations are not cached, and about $500 \cdot 10^9$ kernel evaluations as a lower bound if kernel matrices can be cached completely.

It becomes clear that conducting the experiments in the environments just described was a challenging task in itself. Modern computers provide great computational power to the user, thus enabling large-scale experiments as described above. However, in order to make optimal use of the available computational power, some attention must be paid to the architecture of the computer system.

7.2 Architecture and limitations of modern computer systems

7.2.1 The von-Neumann computer architecture

Modern computers regularly follow the von-Neumann-design [von Neumann, 1945], figure 7.1. It consists of five main components:

- the arithmetic-logic unit (ALU) performing computational and boolean operations,

- the control unit interpreting commands of the program and controling the flow of operations,
- the memory storing programs and data,
- the I/O-unit controling the input and output of data, e. g., with the user through keyboard and terminal, with network interface cards or with external storage systems,
- the bus-system connecting the previous four components with each other.

The ALU and the control unit are commonly aggregated in the central processing unit (CPU).

The separation of CPU and memory has led to the so-called *von-Neumann-bottleneck*, i. e., if the CPU is to perform little processing on large amounts of data it is continuously forced to wait for data to be transferred to or from memory. This effect has been worsened by the fact that CPU speed and memory size have grown much faster than the transfer speed of the bus.

7.2.2 Modern CPUs

The performance penalty caused by the von-Neumann-bottleneck is reduced by a cache, i. e., a small area of high-speed memory, between CPU and main memory. Often, there will be a hierarchy of caches, e. g., a relatively small, but very fast level-one (L1) cache, a moderately sized and moderately fast level-two (L2) cache, and recently also a large but not so fast level-three (L3) cache. Additionally, branch-prediction algorithms implemented in the CPU's control unit aid in keeping the caches filled with data the CPU is likely to need in the near future.

Contemporary CPUs provide specific command sets for specialized tasks, among the most well-known being Intel's *MultiMedia eXtensions (MMX)*, the *Streaming SIMD Extensions (SSE)* and its enhancements *SSE2* and *SSE3*, or AMD's *3DNow!* extensions. These extensions enable parallel and streaming execution especially of floating point commands. Efficient use of these extensions depends even more on best use of caches, possibly by pre-fetching chunks from main memory.

In section 7.7 we will see how software libraries tailored towards specific CPUs can be used to boost the performance of users' programs.

7.2.3 Symmetric Multi-Processor machines

During the last decade the use of two or more CPUs has become more and more popular even in mainstream PCs, with the latest development being two or four CPUs integrated on single chip (multi-core processors). Computers following the

von-Neumann design containing more than one CPU are called symmetric multiprocessor (SMP) machines. They are, therefore, often also referred to as shared memory systems. While these computers offer a relatively cheap way of providing more nominal computing power, they suffer even more from the von-Neumann bottleneck due to all processors having to share the same bus for access to main memory. Additionally, programs must be prevented from being shifted between processors by the operating system to avoid causing cache coherency issues.

A technology called *non-uniform memory access (NUMA)*, implemented, e. g., in AMD's *Opteron* family of CPUs, allows each CPU to have a separate direct bus to an assigned part of main memory, effectively doubling the possible memory throughput. A separate high-speed bus between CPUs allows a process running on one CPU to access the memory managed by any other CPU at the expense of additional latency. However, if NUMA is combined with good planning and management of memory access patterns, programs making extensive use of main memory will benefit significantly from the NUMA architecture.

7.2.4 Distributed-Memory architectures

Another way to increase nominal computational power is to aggregate individual computer systems, e. g., in a *Beowulf* cluster [Sterling et al., 1999]. Figure 7.2 shows a typical setup of such a computer cluster. By combining relatively cheap standard computers, e. g., conventional PCs, this has become a widely accepted alternative to large-scale SMP systems. Due to the nature of separate computers being used in the aggregation, these systems are commonly referred to as distributed-memory systems. Of course, SMP systems can be used as the nodes of the cluster.

While the nominal computational power of a cluster may be large, additional measures must be taken to make this capacity available to users' programs. Especially if integrated computations are to be carried out involving multiple nodes of the cluster, instead of the von-Neumann-bottleneck within each node, the interconnect of the nodes may become the limiting factor in the performance of the entire system. To reduce this problem, high-speed low-latency interconnects like *Myrinet* or *Infiniband* have been developed as alternatives to the rather high-latency Ethernet.

Regardless of the hardware used for node interconnects, software libraries and standards must be available in order to enable user programs to be split into separate, cooperating processes located on different machines. A commonly used approach is the message passing paradigm which is used in standards like *Parallel Virtual Machine (PVM)* [Geist et al., 1994] or the *Message Passing Interface (MPI)* [Snir et al., 1998, Gropp et al., 1999, Pacheco, 1997].

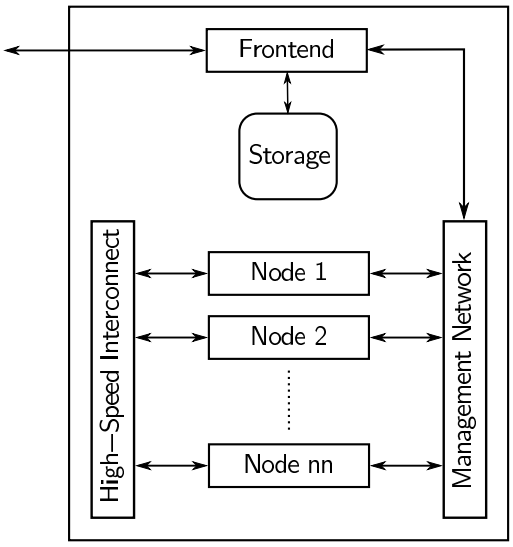


Figure 7.2: Schematic design of a PC cluster

7.3 Requirements on the software library

After considering the constraints summarized in sections 7.1 and 7.2, a number of formal requirements on the software library used in the experiments were developed. In the following, we will check the most important ones and directly argue for the selected solution.

7.3.1 Expressiveness, structure, and re-usability

In chapters 5 and 6 algorithms expressed in matrices and vectors were applied predominantly. Therefore, the software library was required to extend from elementary data types, e.g., individual floating point numbers, to aggregations such as matrices and vectors. These higher-order data types were required to manage meta-information as well, e.g., the number of rows or columns of a matrix. Furthermore, a seamless combination of similar data types, such as integer scalars with floating point matrices, was necessary.

Using the matrix and vectors aggregations, one should be able to make use of the structural advantages of algorithms such as the greedy-sparse KFD or the OROLS-based kernel PDF training procedures. Both of these algorithms frequently apply the idea of a rank update of a matrix, both symmetric or general. The library should honor and exploit computational advantages of, e.g., a symmetric over a general rank-one update.

Additionally, the software should be able to group similar ideas as formal concepts and apply a structure to the examples of these concepts. If we consider, for example,

the idea of the Mercer kernel introduced in section 3.2, we see that there exist a number of different kernel functions all expressing a dot-product in some RKHS. Furthermore, we can see that the standard linear kernel is effectively embedded in more complicated kernels such as the polynomial (3.43) or the tanh-kernel (3.45), making it possible to express these higher-order kernel functions as extensions or users of simpler kernel functions.

By grouping similar ideas into concepts, we have already aided in solving another requirement: Re-usability. When planning and implementing large-scale experiments such as those in chapter 6, small conceptual modifications should only cause evenly small modifications to the experimental setup, i. e., the software library.

7.3.2 Runtime efficiency

In the previous section we were concerned with what can be called programming efficiency. In section 7.1 it became clear, however, that the software library should focus on providing solutions as fast as possible. In section 7.2 we saw that the computing environment may vary greatly, so the software library should take best advantage of, e. g., different CPU types and generations as they are and become available. The software library should also provide ways to extend prototype serial software to production parallel software suitable for shared-memory or distributed-memory environments.

This strong call for runtime efficiency should not contradict the previous call for programming simplicity. The software library should rather hide and encapsulate the details of CPU and architecture used, which goes hand in hand with the requirement to separate dislike concepts and tasks.

7.3.3 Object serialization

The library was required to provide storage and retrieval of the parameters associated with the building blocks of the software. This concept is known as *serialization*. The serialization module should reflect the conceptual structure of the core math functionality to maintain re-usability of the components. If possible, the format should contain sufficient meta-information to enable a human expert to interpret the structure and parameters of the described system.

7.3.4 Tracing, logging, and error handling

Additionally, the library was required to include runtime tracing and logging functionality to catch errors or to peek at the computation being carried out. While it is clear that computation and logging steps must be intertwined to provide detailed and meaningful information, the details of the functionality should be separated

from the core math functionality. This approach aids in maintaining, extending, and possible replacing the logging functionality as users' needs are evolving.

Logging and tracing functionality was required to be flexible in configuration at runtime. Details to be configured included

- selection of sources (abstract or concrete concepts) to be traced,
- setting of the verbosity level of logging,
- choice of logging target (sinks), e. g., disk files, network daemons, databases, ...
- choice of transmission and storage formats.

The logging components were also required to gather information from parallel or distributed processes in a coordinated fashion and to incur very low performance penalty if tracing was disabled. An ancillary requirement was the availability of a software tool for easy and intuitive examination of the possibly large amounts of gathered information.

Furthermore, the software library was required to collect and record detailed information in the event of failures, e. g., memory allocation errors or numerical instability.

7.3.5 Integration with HTK

Finally, it was necessary to construct a software module to be integrated with HTK, the speech recognition engine used in chapter 6. As the explored concepts were new, no infrastructure dealing with kernels, binary discriminants, probability calibration and so on was provided with HTK. Consequently, we wished to use the same software library used in building the training tools for conducting evaluation experiments.

7.4 Design and Implementation

As the analysis of the requirements in the previous section suggests, an object-oriented design was taken for the software library. Objected-oriented analysis and design lend themselves to techniques such as grouping of similar concepts into a hierarchy of concrete classes implementing a concept described in an abstract base class, encapsulating and hiding implementation details such as dependencies on a certain CPU type, and separating orthogonal concepts such as computation and serialization into distinct class hierarchies.

The implementation was carried out using the C++ programming language, a language widely used in the field of engineering. Since the ASR engine HTK is implemented in C, the implementation of the software library in C++ enabled the

use of its components in HTK. By making use of a C++ language feature called templates it was possible to write the library ignorant of the precision of the floating point data types used, i. e., for the HTK module built for recognition purposes it was sufficient to use single precision floating point numbers (resulting in a great speedup), while the same code could be used for writing KFD or kernel-PDF training software which needed double precision floating point numbers to limit the effect of accumulating round-off errors.

The implementation does not aim at implementing the requirements literally, but a rather strong focus was put on making use of external libraries for particular purposes. Using the language features of C++ such as templatization and function overloading, lots of functionality was obtained by simply wrapping legacy software libraries into more convenient interfaces.

7.4.1 Algebra and numerics libraries for computations

In sections 7.3.1 and 7.3.2 we defined expressiveness, runtime efficiency and reusability as central targets of the software library. Most of the data were expected to be presented as vectors or matrices. Therefore, the proven *Basic Linear Algebra Subprograms (BLAS)* [Lawson et al., 1979, Dongarra et al., 2001] were elected for computations involving the use and manipulation of vectors and matrices. While a reference implementation of BLAS is available at http://www.netlib.org, it should rather be regarded as a standard defining interfaces. Originally written in and for the Fortran programming language (which does not limit its usability within other programming languages), an adaptation for the C/C++ programming languages is available under the name *CBLAS*. Since the BLAS is a de-facto standard in linear algebra, versions specifically tuned for specific CPUs are available, leveraging the similarities and differences mentioned in section 7.2.2. The libraries specifically target the von-Neumann-bottleneck by aiming at re-arranging computations and pre-fetching data from main memory into CPU caches in order to minimize idle CPU cycles. Examples of specialized implementations are the *Automatically Tuned Linear Algebra Subprograms (ATLAS)* [Whaley et al., 2001] and *GOTO BLAS* [Goto and van de Geijn, 2002], and most CPU manufacturers provide specific versions tuned to their family of CPUs, e. g., the *Intel* math kernel library (MKL) [Intel Corp., 2005] or the *AMD* core math library (ACML) [AMD, Inc., 2005]. BLAS provides functions for just about any type of vector or matrix manipulation, ranging from simple dot products of two vectors across rank-update functions exploiting symmetric or triangular shape to products of general matrices.

Another de-facto standard library used was the *Linear Algebra Package (LAPACK)* [Anderson et al., 1999], building upon BLAS. LAPACK can be used, e. g., for eigenvalue or singular value decompositions, triangularization and inversion of matrices, or for solving linear equation systems. A reference implementation is also available from http://www.netlib.org, but most specialized BLAS packages are accompanied

by a specific implementation of LAPACK.

Finally, the MKL and ACML both provided additional functions for quickly evaluating certain transcendental functions such as exp, log and various trigonometric functions on vector arguments, exploiting streaming technologies such as SSE mentioned in section 7.2.2. For Intel CPUs, another library called the *Intel* Performance Primitives (IPP) offering similar vectorized transcendental functions was available. Consequently, these libraries were used to obtain high-performance implementations of the Gaussian kernel (3.44).

As all of the above mentioned libraries target legacy programming languages such as C and Fortran, functions with the same functionality on different numeric precision are disambiguated by different names. Therefore, wrapper functions with common name were overloaded to hide the details of the numeric precision from our library. The wrapper functions were also used to turn the error codes returned by most LAPACK functions into C++ exception. This results in very clean user code if errors are not generally expected; however, due to a generic exception handling routine required in section 7.3.4, unexpected errors can still be trapped and recorded.

7.4.2 Memory management

A very close focus was taken at making the handling of memory as simple and fool-proof as possible. As already mentioned, most of the memory to be consumed was forecasted to be vectors and matrices. To wrap these concepts, the optArray template class was created. A UML diagram of the optArray class can be found in figure 7.3. The template can be instantiated with any data type to be stored in a matrix or vector. The template uses reference counting to automatically free allocated memory when it is no longer referenced by any object. If a chunk of memory is used by more than one optArray object, reference counting enables a copy-on-write procedure that postpones copying of data until an actual attempt at modifying the chunk of memory is made, which makes copying of optArray objects very cheap.

In contrast to common usage of C++, the optArray class provides a column-major view of a math matrix. This can be easily handled in C++, but it enables interfacing with numerical software libraries such as BLAS or LAPACK mentioned in the previous section. Also similar to Fortran, the entire memory consumed by a matrix object is allocated as one contiguous block of memory instead of a double-indexed C array. Contiguous memory is then divided into individual columns using a column stride.

The optArray template also provides convenience functions for setting or retrieving meta-properties such as column stride, or row and column count, for initializing memory with fixed values and for common computations such as coordinate-wise products or sums of columns which are not directly available in C++ or

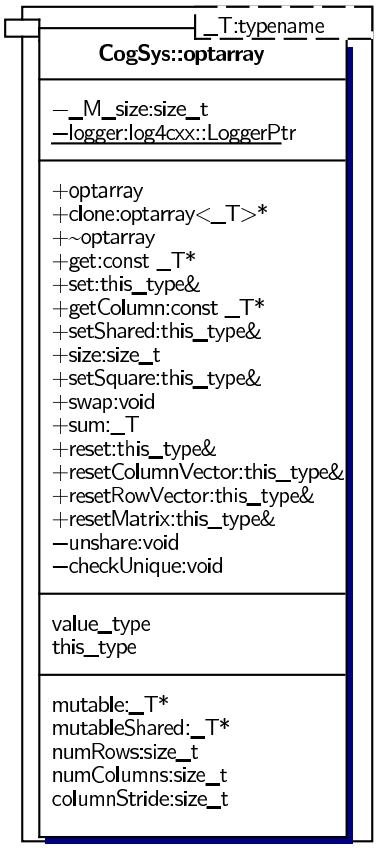


Figure 7.3: UML class diagram of the optArray software class

BLAS/LAPACK.

7.4.3 Core Components

In this section we will examine some major building blocks of the software library used throughout this work. We will use a bottom-up approach which follows the process of creating functional programs from the set of building blocks.

Kernels

Kernel function play a central role throughout this work. Therefore, they have been dedicated a separate class hierarchy. Figure 7.4 shows how the Mercer kernels (3.43), (3.44), (3.45), and (5.10) are implemented as classes specializing the abstract *Kernel* base class. While IntNormGaussKernel implementing (5.48) is not a Mercer kernel, its usage and interface is similar to those, therefore it is part of the Kernel class hierarchy. The Kernel interface provides member functions for computing the dot-product of single samples, a single sample with a vector of samples (resulting

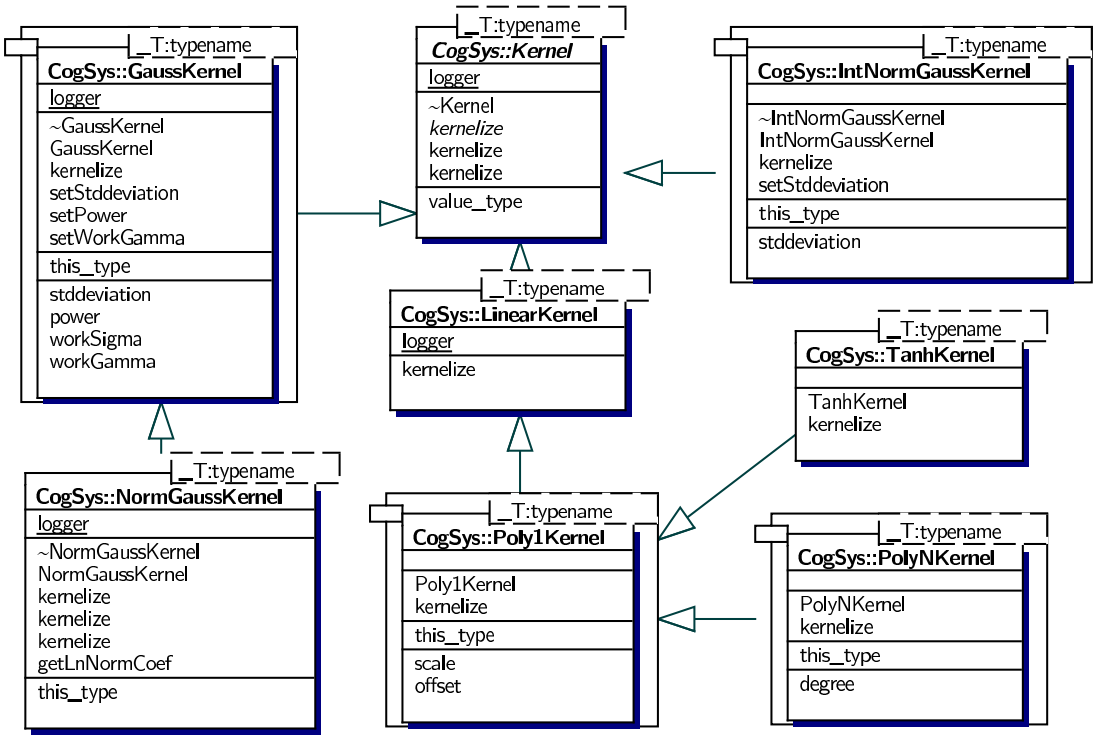


Figure 7.4: Schematic description of the kernel class hierarchy.

in a kernel vector), and for two vectors of samples (resulting in a kernel matrix). This approach avoids explicite loops in user code and enables the use of specialized library functions for computing the exp or tanh functions on vectorized arguments, as mentioned in section 7.4.1.

Machines

A machine is a very basic model of a problem to be solved, i. e., it may be a GMM, an HMM, a layer of an MLP, an entire MLP, or, as in our case, a kernel machine such as an SVM or a KFD. Here, we only needed an implementation of a kernel machine, as HMMs and GMMs were already gracefully handled within HTK, and MLPs were not of significant interest in this work. Figure 7.5 shows the interface of the Machine class hierarchy.

Probability moderators

In section 6.1 we had discussed two principle ways for computing probabilities from the output of a discriminant, either by modeling the densities of the resulting projections as normal densities with properly estimated parameters, or by calibrating the output into posterior label probabilities using a sigmoid function, also with

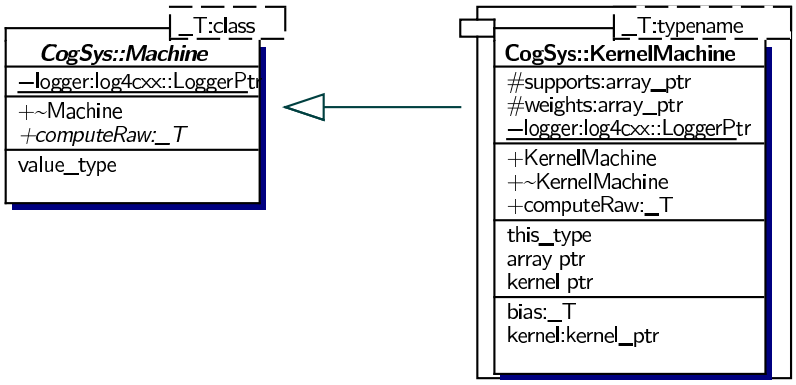


Figure 7.5: UML diagram of the machine class hierarchy

properly estimated parameters. Consequently, both methods are reflected in the ProbModerator class hierarchy shown in figure 7.6. In addition, we provide two proxy moderators which can be used to simply pass the output of a machine as already being a calibrated probability density or posterior label probability. While these dummy moderators do not perform any computation, they are used to unify different approaches (projection followed by probability calibration vs. direct probability estimation) in higher-order aggregations of machines.

Probability unifiers

The combination of probability estimates in the OVO binary regime had been discussed in section 6.2.2. Although we only tested the exact-formula method, the voting method and Kullback-Leibler-distance minimization have been implemented for completeness, see figure 7.7. Additionally, an averaging unifier was implemented to be used if parallel models of the same problem are available.

Classifiers

Finally, we need models integrating the aforementioned ideas of kernel machines, probability moderators, and probability set unifiers. These new models were intuitively called classifiers as they actually model the global problem of multi-label MAP classification, posterior label probability or label-conditional PDF computation.

Now that we have all the core components available, we can define a structure as used for modeling the PDFs in an OVO fashion in chapter 6. Figure 7.8 shows a typical example of this structure. Let us inspect this figure: We start with a classifier concept in the upper left corner, especially a single data stream classifier. This concept is specialized by a unifying classifier, which unifies probabilities drawn from an ensemble of other single data stream classifiers, machine classifiers in this

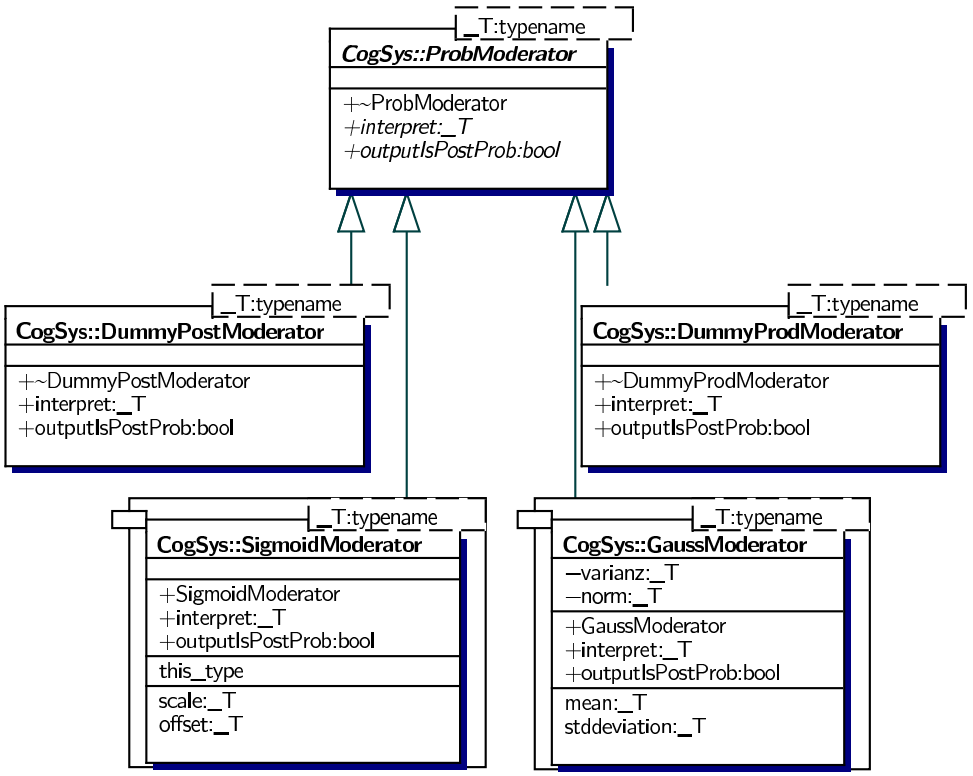


Figure 7.6: UML diagram of the ProbModerator class hierarchy

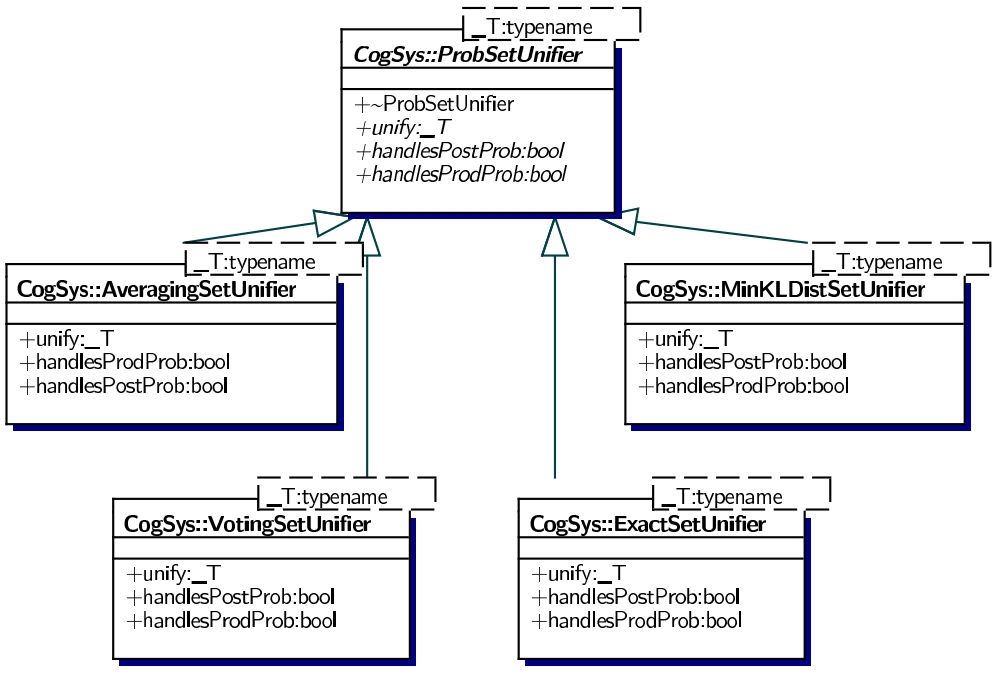


Figure 7.7: UML diagram of the ProbSetUnifier class hierarchy

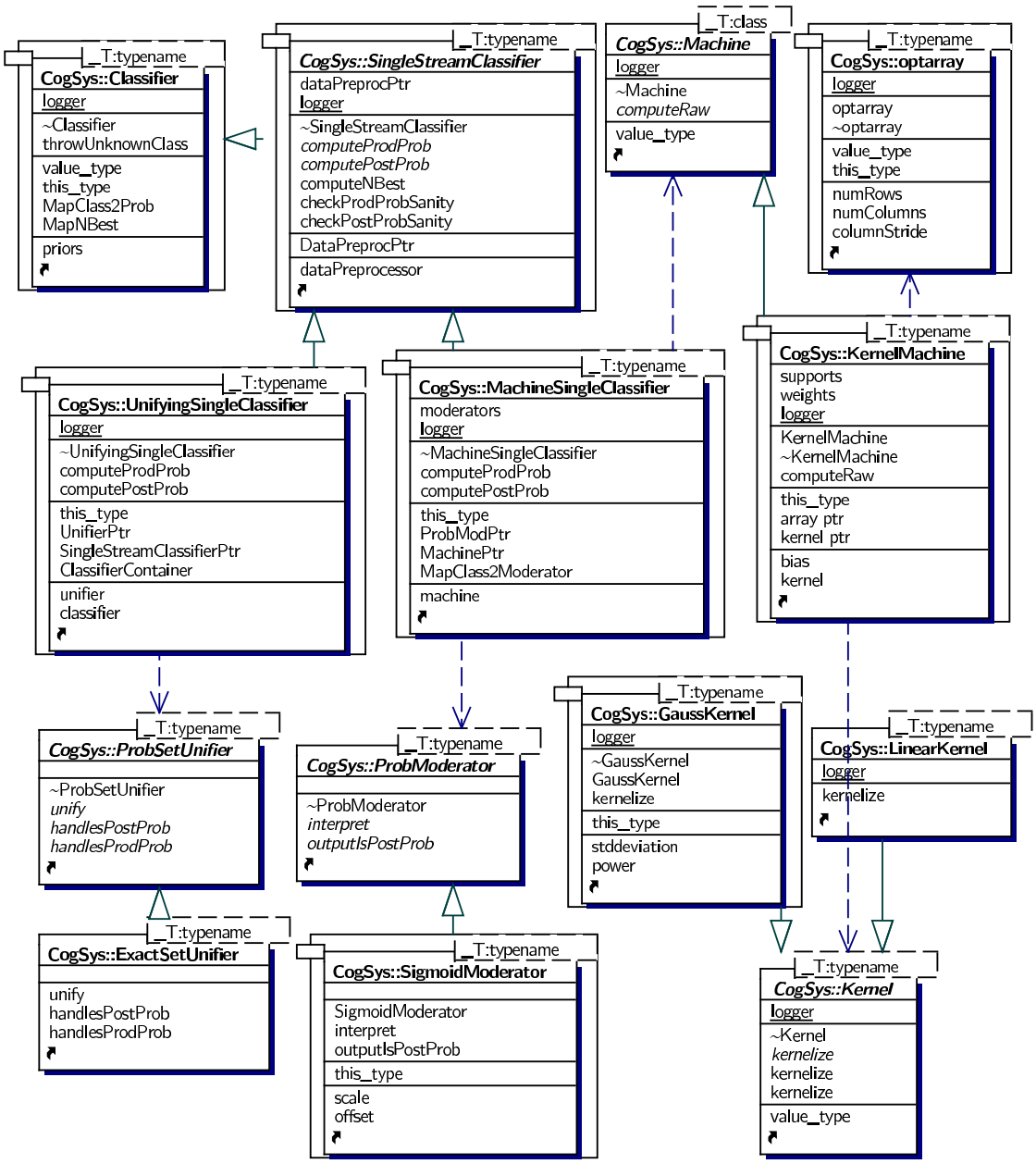


Figure 7.8: UML collaboration diagram showing the typical structure of an OVO model as used in chapter 6

case, using the method discussed in section 6.2.2. Each machine classifiers makes use of an abstract machine, a kernel machine in this case, and a set of probability moderator for each problem class it knows about, in this case, a sigmoid class-posterior probability moderator as discussed in section 6.1.2 is used. In turn, each kernel machine contains its matrix of support vectors and a corresponding weight vector, an abstract pointer to a Kernel object, a Gaussian kernel as in (3.44), and a linear kernel for multiplying the weight vector with the vector of kernel evaluations as in (3.47).

While the structure may look complicated at first, it can be concluded that by first aggregating abstract concepts which are later appropriated, the setup of experiments can be easily changed and reflected in the software library. After having solved the runtime efficiency problem in sections 7.4.1 and 7.4.2, we have now solved the problem of expressiveness and re-usability.

7.4.4 Utilities

Technically, everything that is really needed for performing experiments involving kernelized models is now available. However, there are a number of secondary functionalities to be implemented, i. e., those required by sections 7.3.3, 7.3.4, and 7.3.5.

Serialization

In section 7.3.3 we called for re-usable, component-based, human-readable storage of the structure and the parameters of a model, e. g., one like in figure 7.8. It was decided a-priori that the structure and parameters should be serialized as an *eXtendible Markup Language (XML)* document. XML has a number of advantages, including, but not limited to, the following:

- It is a markup language designed for storing hierarchically structured data in text form.
- It is designed to be relatively legible to humans.
- XML is an open standard, developed by the *World Wide Web Consortium*.
- Due to XML being standardized, software tools and libraries aiding in the design and implementation of software using XML are available.
- XML documents can be easily checked for correct syntax using a number of freely available software tools.
- Using an XML schema language such as *Document Type Definition (DTD)* or *XML Schema Definition (XSD)*, which describe constraints on the structure

and the content of an XML document, the semantic correctness of an XML document can be tested, for which free software tools are available, too.

In XML, data are enclosed within tags which describe the type of the data. The contents of the data type is usually found between the opening and closing tags, where the contents itself may of course be structured data as well. Refinements of the data type are usually given as attributes within the opening tag's braces. Figure 7.9 is a very simple example of how a structure similar to the one described in section 7.4.3 is represented in an XML document. The root element of the XML file describes a SingleStreamClassifier, but since SingleStreamClassifier is only an abstract base class, the type attribute specifies that the concrete implementation to be used is a UnifyingSingleStreamClassifier. The only other attribute "acc" instructs us that the template class was instantiated using double-precision floating point numbers. Consequently, if aggregations of objects are expressed only as pointers to objects of an abstract base class (a concept), the actual structure of the software can be specified in a text document rather than through source code manipulation by using the C++ polymorphism language feature.

Serialization was implemented using the *Simple XML Persistence (SXP)* software library. Figure 7.10 exemplifies the procedure for the Mercer kernels class hierarchy. An abstract base class SXPBase defines the interface of classes implementing persistence through SXP. Additional abstract classes, i. e., PolymorphicIPersistObj and PolymorphicIPersistTemplate, standardize interfaces to be followed by classes filling an abstract concept through runtime polymorphism and interfaces for serializable classes with templatized numeric precision. By deriving new software classes each inheriting from a core math class and the serialization interfaces, specialized software classes simultaneously handling core math functionality and persistence of parameters can be accomplished.

Tracing, logging, error handling

Tracing and logging was implemented using the *LOG4CXX* software library. LOG4CXX is a port of *Log4J*, an extensive logging library implemented in and for the Java programming language, to the C++ programming language. While only the core part of the port is complete, missing functionality can be complimented using tools from the Log4J original.

The machine learning software library was extended with logging functionality by instrumenting each software class with its own named instance of a logger. The loggers' names also reflect the concept group of the instrumented class so that entire class hierarchies may be targeted directly without having to name each implementation separately. As an example, the logger for the GaussKernel class is named "CogSys.Kernels.GaussKernel" showing that the GaussKernel software class belongs to the concept hierarchy of Kernels in the CogSys software library. Therefore, the

```
<?xml version="1.0" encoding="UTF-8"?>
<SingleStreamClassifier acc="double" type="unifying">
  <Prior id="testclass">1.0</Prior>
  <SingleStreamClassifier acc="double" type="machine" id="0">
    <Prior id="testclass">1</Prior>
    <Machine acc="double" type="Kernel">
      <Bias>-1.99526</Bias>
      <OptArray acc="double" columnStride="3" id="supports"
                numColumns="4" numRows="3" size="12">
        <column id="0">1 2 3</column>
        <column id="2">3 6 9</column>
        <column id="3">
             4 8 -12
        </column>   <column id="1">2 4 6</column>
      </OptArray>
      <OptArray acc="double" columnStride="4" id="weights"
                numColumns="1" numRows="4" size="4">
        <column id="0">1 2 3 4 </column>
      </OptArray>
      <Kernel acc="double" type="gauss">
        <stddeviation>1.375</stddeviation>
      </Kernel>
    </Machine>
    <ProbModerator acc="double" id="testclass" type="gauss">
      <mean>0</mean>
      <stddeviation>1</stddeviation>
    </ProbModerator>
  </SingleStreamClassifier>
  <ProbSetUnifier acc="double" type="exact"/>
</SingleStreamClassifier>
```

Figure 7.9: Example of an XML file used to store the structure and parameters of a classifier.

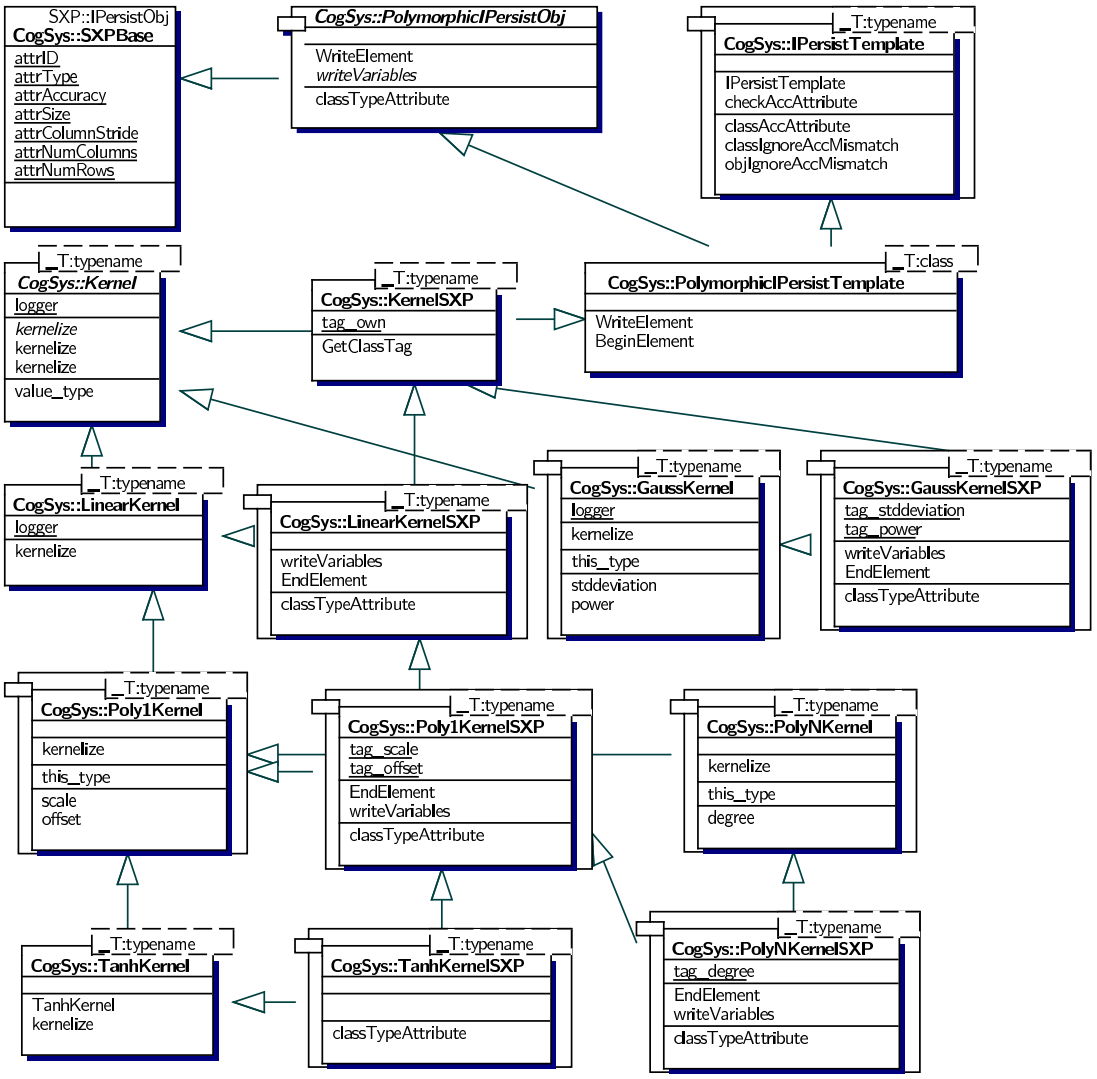


Figure 7.10: UML diagram of core math kernel classes and their persistence-enabled derivative classes

virtual logger name "CogSys.Kernels" will refer to all Kernel class implementations simultaneously.

Information to be saved are written to the logger, together with a tag denoting level of detail or severity. Using new convenience functions, tracing of computations by logging matrices and vectors is possible with close to no effort. If tracing of certain components is limited or disabled, virtually no performance hit is encountered even though the component in question may be fully instrumented with logging calls because message preparation and format conversion steps will only take place after having compared the log level of the message with the desired log level to be saved. LOG4CXX automatically adds ancillary information such as time, process and thread identifiers, or hostname to each message. Messages are kept in internal data structures before being send to a sink. There exist a number of possible message sinks including, but not limited to, disk files in various formats, e. g., plain text or XML, intermediate network daemons like syslog or Log4J's XML-collector, or SQL databases. The latter two are especially useful if tasks involving multiple processes possibly running on separate computers, e. g., on a distributed-memory computer cluster mentioned in section 7.2.4, are to be traced in a coordinated fashion. Subsequently, tracing logs may be examined through Log4J's interactive *Chainsaw* GUI program which allows for filtering the logs according to any desired criterion, e. g., messages' time-stamps, log level, or source components.

7.5 Driver applications

Using the components introduced in the previous section, driver applications for training the parameters of various parts of the PDF model system were developed, including, but not limited to, programs implementing QP training algorithms for ϵ-SVD, OROLS-SVD, and greedy-sparse KFD, and programs for training the parameters of probability calibration techniques discussed in section 6.1. The library was also used in cooperation with colleagues to implement algorithms and experimental setups published in [Andelic et al., 2004, 2005, Katz et al., 2006a,b, Andelic et al., 2006b,a, Katz et al., 2007a, Andelic et al., 2007].

7.5.1 Parallelization

For computational jobs involving many similar tasks, e. g., training KFDs in an OVO binary regime, a simple-to-use framework for parallelizing these jobs to distributed-memory computer clusters was created. The idea was to assign each processor one of the many optimization tasks rather than using multiple processors for a single optimization task. Using a simple scheduling component which ranks the tasks by expected runtime, tasks are dynamically assigned to a processor in the cluster starting with the ones with largest expected runtime in order to heuristically sustain

a dense packing of the processors allocated to each job.

The MPI standard mentioned in section 7.2.4 was used for communications. The use of MPI in environments using highly structured data, e. g., the C++ software library described here, is generally quite cumbersome because the memory layout of each data structure, i. e., software class, must be described in terms of MPI commands in order to enable MPI to send the data structures in a coordinated fashion. This becomes even more difficult when dynamically sized memory, e. g., for matrices and vectors, or pointers to polymorphic objects are involved.

Fortunately, the proposed software library offers a simple solution to this problem. In section 7.4.4 a method of serializing data objects primarily into persistent files was developed. The same methodology can be used by serializing objects into memory buffers which, being simple character string objects, can easily be transfered between processes of a job using MPI. In essence, a sending process will pack the data into a memory buffer using the library's XML layout and send it to the expecting process. The receiving process is accepting the XML character string into another buffer from where the original data can be restored using, again, the library's serialization engine.

It is clear that this procedure incurs larger runtime latencies compared to using intrinsic MPI technology. However, the library user is relieved from the burden of dealing with the MPI standard by simply using the library's familiar serialization engine. Additionally, as the envisaged usage of this approach is coarse-grained parallelization, the performance penalty is expected to be negligible.

7.6 Integration in HTK

As stated in section 7.3.5, HTK was to be adapted to accept PDF models other than GMMs. HTK itself is unable to handle PDF models like the ones discussed in chapters 5 and 6 directly. Consequently, there were two possible options: Adding the functionality directly, making it a part of HTK, or defining and implementing interfaces in HTK in order to facilitate the use of arbitrary PDF models.

The latter approach was pursued because it is the more general one. HTK was adapted to accept two additional directives in the global section of an HMM file, i. e., the keyword *EXTPROBLIB* defining the name or path of an external program library to be used, and the keyword *EXTPROBCONF* accepting a free-from character string to be passed to the external program library as an argument at initialization time.

The external program library, called a *Dynamically Linkable Library (DLL)* on *Microsoft Windows* or a *Dynamically Shared Object (DSO)* on most *Unix* systems, is required to provide three functions with pre-defined signature, i. e., one function for initializing the program library using the aforementioned initialization string, one function for coordination shutdown, and the central function computing a PDF

value given a problem class and a feature vector. Any program library having these basic properties can be used as a plug-in PDF source in HTK, which provides great flexibility in the choice of the PDF models. Here, the external program library was constructed using components introduced in section 7.4.

7.7 Benchmarks and comparison

The development of this software library had been pushed by preliminary experiments implemented and run in MATLAB, a general-purpose mathematical programming environment. Even after careful profiling and tuning, the initial MATLAB implementation of the greedy-sparse KFD fell way behind our expectations in terms of runtime performance, which had me develop the proposed alternative in C++.

Two distinct causes of the bad runtime performance of the initial MATLAB implementation had been isolated: Slow computation of kernel function values and the fact that it was impossible to properly make use of algorithmic advantages like efficient rank-updates of matrices, in MATLAB.

While the C++ software library was being developed, exemplary experiments were conducted to verify that the assumption of being able to accelerate computations using C++ and high-performance libraries, i. e., those mentioned in section 7.4.1, was actually valid. The experiments are meant to be conceptual rather than complete, but the presented material will give striking evidence that our assumptions were correct.

First, the Mercer kernel hierarchy of software classes was developed and implemented, as it was fairly easy and a prerequisite to implementing training methods like greedy-sparse KFD. Table 7.1 compares the performance of a profiled and tuned MATLAB implementation of the Gaussian kernel with implementations in the proposed C++ library, which is also compared in terms of the external performance library employed. In the experiment, two sets of $10,000$ random 40-dimensional vector were kernelized with each other, resulting in a total of $100,000,000$ kernel values to be computed. Each set of sample vectors was stored in $40 \times 10,000$ matrices; all necessary memory was pre-allocated. The runtime for allocating and freeing memory as well as for loading the vectors from files was not included, resulting in the runtimes for bare-kernel computations. The C++ software library was compiled using full compiler optimization, which should, however, only influence the results obtained without external computational libraries. MATLAB was using the MKL implementation of the BLAS. All experiments were carried out on a PC containing a 2.5 GHz Pentium-4 CPU.

There are several conclusion to be drawn. The most striking one is that MATLAB may be well-suited for small-scale or prototypical applications but not for serious large-scale computations, at least if they involve Gaussian kernels. MATLAB non-performance computing Gaussian kernels may be negligible if additional

Language	Ext. Libs	Single Precision		Double Precision	
		Abs. Time	Rel. Penalty	Abs. Time	Rel. Penalty
MATLAB	MKL	550s	2955%	579s	2085%
C++	none	32.9s	83%	47.9s	81%
	MKL	27.2s	51%	40.0s	51%
	IPP	18.0s	—	26.5s	—

Table 7.1: Runtime comparison for different implementations of the Gaussian kernel, computing 100,000,000 kernel evaluations

computations of an algorithm dominate the evaluation of Gaussian kernels, but only if these additional computations may be carried out efficiently. However, if a so-trained model is to be applied during testing, the evaluation of kernel functions vastly dominates computation requirements, clearly prohibiting the use of MATLAB.

Another important point is that the runtime of the C++ software library benefits from limiting computations to single precision, if numerically possible. Regardless of the use of external libraries, the C++ software library shows about 30% reduction of runtime if using single precision only. MATLAB, however, cannot make use of single precision computations — the reduction in runtime is a mere 5%.

Lastly, if we limit our consideration to the C++ software library, we can experimentally validate our claim from section 7.4.1 that the use of external numerical libraries would greatly reduce runtime requirements. From the table it can be seen that compared to a plain-C++ implementation, the use of a tuned BLAS implementation results in a performance boost of roughly 15%, while the use of an even more sophisticated library (IPP) yields an acceleration of approximately 46%.

As a second example benchmark, the MATLAB implementation of the greedy-sparse KFD was compared against a C++ implementation using the proposed software library and the MKL-incarnation of the BLAS. Both implementations followed the procedure discussed in section 3.5.1. It turns out that the C++ implementation expressing the algorithm directly using appropriate functions from the BLAS runs about 3.5 times faster than the competing MATLAB implementation. As we already knew about the slowness of MATLAB computing Gaussian kernels, these times were not taken into account. Tracing the use of BLAS routines by MATLAB, it turned out that MATLAB had not taken advantage of the structural advantages of the greedy-sparse KFD training algorithm. Especially a rank-one update of a symmetric matrix, could not be expressed as such in MATLAB which had, instead, deduced a higher-order rank update of a general matrix, resulting in a rather large performance loss.

8 Conclusion

In this thesis we have investigated how nonparametric probability density functions can be constructed using learning machines based on the structural risk minimization approach. When estimating the parameters of a learning machine, structural risk minimization achieves better generalization ability than conventional empirical risk minimization by minimizing a bound on the expected risk on unobserved samples, especially if the training sample size if relatively small. To this end, a regularization functional is added to the learning machine's cost function in order to weigh empirical risk (error on training data) against capacity of the learning machine.

Our focus was the automatic speech recognition learning problem, where using prior knowledge the learning problem is split into language modeling, phonetic modeling, and acoustic modeling. Sub-word phonetic units are conventionally represented by Hidden-Markov-Models, i.e., finite state automata with unobservable states emitting observable symbols. Usually, parametric Gaussian Mixture Models (GMMs) trained using empirical risk minimization are used to model the Hidden Markov Model (HMM)-states' emission probability density functions (PDFs). Our ultimate aim was replacing GMMs with PDF models trained using Structural Risk Minimization (SRM) in order to obtain better performance of automatic speech recognition (ASR) systems if training data is limited.

After a thorough introduction of SRM in chapter 2, of pattern discrimination machines employing SRM in chapter 3, and the ASR learning problem in chapter 4, we investigated methods constructing sparse kernel PDFs using SRM-based regression of the empirical cumulative distribution function (ECDF) in chapter 5. We showed that previous approaches suffered from large memory requirements at training time, rendering the methods impossible to apply to real-world problems. To overcome this problem, we developed an algorithm where memory requirements scale with the size of the (sparse) solution rather than with the size of the training problem. This reduction of memory requirements was achieved by applying a novel thin update of the orthogonal decomposition of the training formulation in a forward sample-selection procedure. Initial experiments on relatively low-dimensional problems indicated that the new training algorithm can construct sparse kernel PDFs with similar accuracy as non-sparse optimal Parzen windows PDFs. However, it was found that the ECDF regression approach effectively cannot be applied to high-dimensional problems such as acoustic modeling in ASR as the complexity of the ECDF in high dimensions becomes prohibitively large.

In chapter 6 we followed our idea that PDFs can be constructed from binarily discriminating functions and Bayes' rule. We collected methods for turning the outputs of the discriminants into probability measures and for combining these pairwise probabilities measures into multi-class probability measures in the one-vs-one binary regime. After a suitable setting of samples and problem classes had been defined, we applied probabilistically interpreted Support Vector Machines (SVMs) and Kernel Fisher Discriminants (KFDs) to the Resource Management (RM1) speech recognition task, where the limitation of training data was simulated by ten-fold sub-sampling of the complete training data. Our prototypical system indicated that a monophone HMM system benefits greatly from applying the proposed discriminant-based PDF models by outperforming conventional GMM-based emission PDFs in all cases. We found that SVM-based PDFs continuously perform better than KFD-based PDFs and concluded that the greedy-sparse KFD training procedure requires further investigation and improvements in complex scenarios like ASR on large databases.

While our exemplary experiments showed very good results, there remain a number of open questions. For the proposed PDFsetup to be a truly viable alternative to GMMs our method must be extended to handle context-dependent phonetic units, e. g., triphones. As the presently used one-vs.-one (OVO) or one-vs.-rest (OVR) binary regimes are inadequate for this, method like error-correcting output codes and arbitrary code matrices have been identified from the literature and proposed for investigation. Additionally, adaptation of speaker-independent acoustic models to specific speakers or acoustic conditions is commonly used to improve speech recognition accuracy. For our proposed discriminant-based PDF models, adaptation techniques such as incremental learning and un-learning should, therefore, be studied.

Finally, we have explored techniques to implement the proposed systems in software in chapter 7. We have identified theoretical performance bottlenecks associated with hardware or software limitations. We proposed a novel object-oriented software component library which gathers similar tasks as specializations of abstract concepts and represents them as a hierarchy of derived software classes. A primary focus of attention was put on best runtime efficiency. To this end, we identified standardized BLAS and LAPACK software packages as solutions, because specialized implementations appreciating specific properties of certain CPUs are available. Experiments showed that BLAS/LAPACK based implementations significantly beat traditionally coded implementations both in programming and in runtime efficiency. The library is augmented with a set of utility functionality like component-based object serialization, transparent exception handling, and efficient event tracing and logging, which aids the user in exploring a learning task rather than a software engineering task.

Summarizing what has been said in this chapter, we can conclude that a fair amount of progress has been made with respect to applying SRM-based PDF models to a large-scale problem like automatic speech recognition. However, it was disap-

pointing to find a promising method like SRM-based sparse kernel PDF construction (chapter 5) inapplicable to high-dimensional problems like ASR. On the other hand, it was surprising to find a conceptually simple model like the discriminant-based PDFs provide such good results in chapter 6, even though the system looks complex in practice.

We conjecture that the philosophers' stone with respect to SRM-based PDF estimation has not yet been found...

Bibliography

Author's Publications

M. Schafföner, M. Katz, S. E. Krüger, and A. Wendemuth. Improved robustness of automatic speech recognition using a new class definition in linear discriminant analysis. In *Proceedings of the 8th European Conference on Speech Communication and Technology (EUROSPEECH)*, pages 2841–2844, 2003.

M. Schafföner, E. Andelic, M. Katz, S. E. Krüger, and A. Wendemuth. Kernel fisher discriminants as acoustic models in HMM-based speech recognition. In G. Kokkinakis, editor, *Proceedings of the 10th International Conference "Speech and Computer" (SPECOM)*, pages 349–352, 2005. ISBN 5-7452-0110-x.

M. Schafföner, E. Andelic, M. Katz, S. E. Krüger, and A. Wendemuth. Limited training data robust speech recognition using kernel-based acoustic models. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 1137–1140, 2006. ISBN 1-4244-0469-X.

M. Schafföner, E. Andelic, M. Katz, S. E. Krüger, and A. Wendemuth. Memory-efficient orthogonal least squares kernel density estimation using enhanced empirical cumulative distribution functions. In M. Meila and X. Shen, editors, *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 417–424, 2007. ISBN 0-9727358-2-8.

Co-author Publications

E. Andelic, M. Schafföner, S. E. Krüger, M. Katz, and A. Wendemuth. Iterative implementation of the kernel fisher discriminant for speech recognition. In *Proceedings of the Ninth International Conference "Speech and Computer" (SPECOM)*, pages 99–103, 2004. ISBN 5-7452-0110-x.

E. Andelic, M. Schafföner, S. E. Krüger, M. Katz, and A. Wendemuth. Acoustic modelling using kernel-based discriminants. In *Proceedings of the 10th International Conference "Speech and Computer" (SPECOM)*, pages 139–142, 2005. ISBN 5-7452-0110-x.

E. Andelic, M. Schafföner, M. Katz, S. E. Krüger, and A. Wendemuth. Kernel least squares models using updates of the pseudoinverse. *Neural Computation*, 18(12): 2928–2935, Dec. 2006a.

E. Andelic, M. Schafföner, M. Katz, S. E. Krüger, and A. Wendemuth. A hybrid HMM-based speech recognizer using kernel-based discriminants as acoustic models. In *Proceedings of the 18th International Conference on Pattern Recognition (ICPR)*, volume 2, pages 1158–1161, 2006b. ISBN 0-7695-2521-0.

E. Andelic, M. Schafföner, M. Katz, S. E. Krüger, and A. Wendemuth. Updates for nonlinear discriminants. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 660–665, 2007.

M. Katz, S. E. Krüger, M. Schafföner, E. Andelic, and A. Wendemuth. Kernel methods for discriminant analysis in speech recognition. In *Proceedings of the Ninth International Conference "Speech and Computer" (SPECOM)*, pages 95–98, 2004. ISBN 5-7452-0110-x.

M. Katz, M. Schafföner, E. Andelic, S. E. Krüger, and A. Wendemuth. Sparse kernel logistic regression for phoneme classification. In *Proceedings of the 10th International Conference "Speech and Computer" (SPECOM)*, pages 523–526, 2005. ISBN 5-7452-0110-x.

M. Katz, S. E. Krüger, M. Schafföner, E. Andelic, and A. Wendemuth. Speaker identification and verification using support vector machines and sparse kernel logistic regression. In *Advances in Machine Vision, Image Processing, and Pattern Analysis, International Workshop on Intelligent Computing in Pattern Analysis/Synthesis, IWICPAS 2006, Proceedings*, volume 4153 of *Lecture Notes in Computer Science*, pages 176–184, Berlin/Heidelberg, 2006a. Springer. ISBN 3-540-37597-X. doi: http://dx.doi.org/10.1007/11821045_19.

M. Katz, M. Schafföner, E. Andelic, S. E. Krüger, and A. Wendemuth. Sparse kernel logistic regression using incremental feature selection for text-independent speaker identification. In *IEEE Odyssey 2006 — The Speaker and Language Recognition Workshop, Proceedings*, pages 1–6, 2006b. ISBN 1-4244-0472-X. doi: http://dx.doi.org/10.1109/ODYSSEY.2006.248115.

M. Katz, E. Andelic, S. E. Krüger, M. Schafföner, and A. Wendemuth. Discriminative kernel classifiers in speaker recognition. In *33rd German Annual Conference on Acoustics (DAGA)*, 2007a.

M. Katz, M. Schafföner, S. E. Krüger, and A. Wendemuth. Score calibrating for speaker recognition based on support vector machines and gaussian mixture models. In *9th IASTED International Conference on Signal and Image Processing*, 2007b.

S. E. Krüger, S. Barth, M. Katz, M. Schafföner, E. Andelic, and A. Wendemuth. Free energy classification at various temperatures for speech recognition. In *Proceed-*

ings of the Ninth International Conference "Speech and Computer" (SPECOM), pages 104–107, 2004. ISBN 5-7452-0110-x.

S. E. Krüger, M. Schafföner, M. Katz, E. Andelic, and A. Wendemuth. Using support vector machines in a HMM-based speech recognition system. In *Proceedings of the 10th International Conference "Speech and Computer" (SPECOM)*, pages 329–332, 2005a. ISBN 5-7452-0110-x.

S. E. Krüger, M. Schafföner, M. Katz, E. Andelic, and A. Wendemuth. Speech recognition with support vector machines in a hybrid system. In *Proceedings of the 9th European Conference on Speech Communication and Technology (EUROSPEECH)*, pages 993–996. ISCA, 2005b.

S. E. Krüger, M. Schafföner, M. Karz, E. Andelic, and A. Wendemuth. Mixture of support vector machines for HMM based speech recognition. In *Proceedings of the 18th International Conference on Pattern Recognition (ICPR)*, volume 4, pages 326–329, 2006. ISBN 0-7695-2521-0.

S. E. Krüger, M. Schafföner, M. Katz, E. Andelic, and A. Wendemuth. Support vector machines as acoustic models in speech recognition. In *33rd German Annual Conference on Acoustics (DAGA)*, pages 1–4, 2007.

K. T. Mengistu, M. Schafföner, and A. Wendemuth. Gender recognition and adaptation for telephone-based spoken dialog system. In *18. Konferenz Elektronische Sprachsignalverarbeitung (ESSV)*, pages 1–8, 2007.

References

E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, Sept. 2000. ISSN 1533-7928.

AMD, Inc. *AMD Core Math Library, Reference Manual*, 2005.

E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. C. Sorensen. *LAPACK User's Guide*. SIAM, third edition, 1999. ISBN 0-89871-447-8.

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337–404, 1950.

A. Asuncion and D. J. Newman. UCI machine learning repository, 2007. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

R. E. Bellmann. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1957.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, New York, NY, USA, 1992. ACM Press. ISBN 0-89791-497-X. doi: http://doi.acm.org/10.1145/130385.130401.

H. Bourlard and N. Morgan. Hybrid HMM/ANN systems for speech recognition: Overview and new research directions. In C. L. Giles and M. Gori, editors, *Adaptive Processing of Sequences and Data Structures: International Summer School on Neural Networks*, pages 389–417. Springer, 1998. ISBN 3540643419.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004. ISBN 0-521-83378-7.

R. A. Bradley. Rank analysis of incomplete block designs. II. Additional tables for the method of paired comparisons. *Biometrika*, 41(3–4):502–537, 1954. doi: http://dx.doi.org/10.1093/biomet/41.3-4.502.

R. A. Bradley. Rank analysis of incomplete block designs. III. Some large-sample results on estimation and power for a method of paired comparisons. *Biometrika*, 42(3–4):450–470, 1955. doi: http://dx.doi.org/10.1093/biomet/42.3-4.450.

R. A. Bradley and M. E. Terry. Rank analysis of incomplete block designs. I. The method of paired comparisons. *Biometrika*, 39(3–4):324–345, 1952. doi: http://dx.doi.org/10.1093/biomet/39.3-4.324.

J. R. Bunch and L. Kaufman. Some stable methods for calculating inertia and solving symmetric linear systems. *Mathematics of Computation*, 31(137):163–179, Jan. 1977. ISSN 0025-5718. doi: http://dx.doi.org/10.2307/2005787.

C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, June 1998. ISSN 1384-5810.

A. Cabaña and E. M. Cabaña. Transformed empirical processes and modified kolmogorov-smirnov tests for multivariate distributions. *The Annals of Statistics*, 25(6):2388–2409, 1997. ISSN 0090-5364.

G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 409–415. MIT Press, Cambridge, MA, 2001. ISBN 0-262-12241-3.

T. P. Centeno and N. D. Lawrence. Optimising kernel parameters and regularisation coefficients for non-linear discriminant analysis. *Journal of Machine Learning Research*, 7:455–491, Feb. 2006.

S. Chen, X. Hong, and C. J. Harris. Sparse kernel regression modeling using combined locally regularized orthogonal least squares and d-optimality experimental design. *IEEE Transactions on Automatic Control*, 48(6):1029–1036, 2003. ISSN 0018-9286.

S. Chen, X. Hong, and C. J. Harris. Sparse kernel density construction using orthogonal forward regression with leave-one-out test score and local regularization. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 34 (4):1708–1717, Aug. 2004. ISSN 1083-4419.

R. Collobert and S. Bengio. SVMTorch: Support vector machines for large-scale regression problems. *The Journal of Machine Learning Research*, 1:143–160, 2001.

C. Cortes and V. N. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. ISSN 0885-6125. doi: http://dx.doi.org/10.1023/A:1022627411411.

T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

J. Dongarra, S. Hammarling, S. Blackford, and A. Lumsdaine. Basic Linear Algebra Subprograms Technical (BLAST) forum standard. Technical report, University of Tennessee, Knoxville, TN, 2001.

G. Fant. *Acoustic Theory of Speech Production*. Mouton, The Hague, 1960.

R. A. Fisher. The use of multiple measurements in taxonomic problems. In *Annals of Eugenics*, volume 7, pages 179–188. Cambridge University Press, 1936.

R. Fletcher. *Practical methods of optimization*. John Wiley & Sons, Inc., Chichester, UK, 1987. ISBN 0-471-91547-5.

C. M. Fonseca. Output-sensitive computation of the multivariate ecdf and related problems. In W. Härdle and B. Rönz, editors, *COMPSTAT 2002 - Proceedings in Computational Statistics*. Springer, 2002. ISBN 3-7908-1517-9.

J. L. Gauvain and C.-H. Lee. Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains. *IEEE Transactions on Speech and Audio Processing*, 2(2):291–298, Apr. 1994.

A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. S. Sunderam. *PVM : Parallel Virtual Machine — A Users' Guide and Tutorial for Networked Parallel Computing*. Scientific and Engineering Computing. MIT Press, Cambridge, MA, USA, 1994. ISBN 0-262-57108-0.

P. E. Gill, W. Murray, and M. H. Wright. *Practical optimization*. Academic Press, London, 10 edition, 1993. ISBN 0-12-283952-8.

G. H. Golub and C. F. van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, USA, 3rd edition, 1996. ISBN 0-8018-5414-8.

K. Goto and R. van de Geijn. On reducing TLB misses in matrix multiplication. Technical Report TR-2002-55, The University of Texas at Austin, Department of Computer Sciences, 2002.

K. Greer, B. Lowerre, and L. Wilcox. Acoustic pattern matching and beam searching. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 7, pages 1251–1254, 1982.

W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message-Passing Interface*. MIT Press, 2nd edition, 1999. ISBN 0-262-57134-X.

R. Haeb-Umbach and H. Ney. Linear discriminant analysis for large vocabulary speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 1992)*, volume 1, pages 13–16, 1992. doi: http://dx.doi.org/10.1109/ICASSP.1992.225984.

M. Hansen and B. Yu. Model selection and minimum description length principle. *Journal of the American Statistical Association*, 96(454):746–774, 2001.

T. Hastie and R. Tibshirani. Classification by pairwise coupling. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems 10*. MIT Press, Cambridge, MA, USA, June 1998. ISBN 0-262-10076-2.

R. A. Horn and C. R. Johnson. *Matrix analysis*. Cambridge University Press, Cambridge, UK, 1985. ISBN 0-521-38632-2.

T.-H. Huang, R. C. Weng, and C.-J. Lin. Generalized bradley-terry models and multi-class probability estimates. *Journal of Machine Learning Research*, 7:85–115, Jan. 2006.

M.-Y. Hwang and X. Huang. Subphonetic modeling for speech recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP). Proceedings.*, pages 174–179, 1992.

Intel Corp. *Intel Math Kernel Library, Reference Manual*, 2005.

V. K. Ivanov. On linear problems which are not well-posed. *Doklady Akademii Nauk USSR*, 145(2):270–272, 1962.

T. Joachims. Making large-scale svm learning practical. Technical Report 24, Universität Dortmund, 1998.

J.-C. Junqua and J.-P. Haton. *Robustness in Automatic Speech Recognition*. Kluwer Academic Publishers, Boston, MA, 1996.

A. Justel, D. Peña, and R. Zamar. A multivariate kolmogorov-smirnow test of goodness of fit. *Statistics & Probability Letters*, 35:251–259, 1997. ISSN 0167-7152.

J. Kaiser, B. Horvat, and Z. Kačič. Overall risk criterion estimation of hidden markov model parameters. *Speech Communication*, 38(3–4):383–398, Nov. 2002. doi: http://dx.doi.org/10.1016/S0167-6393(02)00009-2.

S. Knerr, L. Personnaz, and G. Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In F. Fogelman Soulié and J. Hérault, editors, *Proceedings of the NATO Advanced Research Workshop on Neurocomputing: Algorithms, Architectures, and Applications*, NATO Advanced Research Institute, series F: Computer and systems sciences, pages 41–50, Berlin, 1990. Springer. ISBN 0-387-53278-1.

P. Laskov, C. Gehl, S. Krüger, and K.-R. Müller. Incremental support vector learning: Analysis, implementation and applications. *Journal of Machine Learning Research*, 7:1909–1936, Sept. 2006.

C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh. Basic Linear Algebra Subprograms for Fortran usage. *ACM Transactions on Mathematical Software*, 5(3):308–323, Sept. 1979. ISSN 0098-3500. doi: http://doi.acm.org/10.1145/355841.355847.

K.-F. Lee and R. Reddy. *Automatic Speech Recognition: The Development of the Sphinx Recognition System*. Kluwer Academic Publishers, Norwell, MA, USA, 1988. ISBN 0-898-38296-3.

C. J. Leggetter and P. C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models. *Computer Speech and Language*, 9(2):171–185, 1995. ISSN 0885-2308.

H.-T. Lin, C.-J. Lin, and R. C. Weng. A note on Platt's probabilistic outputs for support vector machines. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 2003.

J. D. Loudin and H. E. Miettinen. A multivariate method for comparing N-dimensional distributions. In L. Lyons, R. Mount, and R. Reitmeyer, editors, *Proc. of the Conference on Statistical Problems in Particle Physics, Astrophysics and Cosmology (PHYSTAT)*, pages 207–210, 2003.

W. Macherey, L. Haferkamp, R. Schlüter, and H. Ney. Investigations on error minimizing training criteria for discriminative training in automatic speech recognition. In *Proceedings of the 9th European Conference on Speech Communication and Technology*, pages 2133–2136. ISCA, 2005.

D. J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992.

J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 209:415–446, 1909.

S. Mika. *Kernel Fisher Discriminants*. PhD thesis, Technische Universität Berlin, Berlin, Dec. 2002.

S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. *Neural Networks for Signal Processing*, 9:41–48, 1999. URL `http://ida.first.gmd.de/~mika/PS/MikRaeWesSchMue99.pdf`.

S. Mika, A. Smola, and B. Schölkopf. An improved training algorithm for kernel fisher discriminants. In T. Jaakkola and T. Richardson, editors, *Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics (AISTATS)*, pages 98–104, San Francisco, CA, USA, 2001. Society for Artificial Intelligence and Statistics, Morgan Kaufmann.

Y. Normandin. *Hidden Markov models, maximum mutual information estimation, and the speech recognition problem*. PhD thesis, McGill University, Montreal, Quebec, Canada, 1991.

E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, pages 276–285, 1997a.

E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. In *1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pages 130–136, Los Alamitos, CA, USA, 1997b. IEEE Computer Society. doi: http://doi.ieeecomputersociety.org/10.1109/CVPR.1997.609310.

P. S. Pacheco. *Parallel Programming with MPI*. Morgan Kaufmann, San Francisco, CA, 1997.

S. Paramasamy. On the multivariate kolmogorov-smirnov distribution. *Statistics & Probability Letters*, 15:149–155, 1992.

E. Parzen. On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

D. L. Phillips. A technique for the numerical solution of certain integral equations of the first kind. *Journal of the Association for Computing Machinery*, 9(1):84–97, 1962. ISSN 0004-5411. doi: http://doi.acm.org/10.1145/321105.321114.

J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, chapter 12, pages 185–208. MIT Press, Cambridge, MA, 1998. ISBN 0-262-19416-3.

J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In P. Bartlett, B. Schölkopf, D. Schuurmans, and A. Smola, editors, *Advances in Large-Margin Classifiers*, pages 61–74. MIT Press, Cambridge, MA, USA, Oct. 2000. ISBN 0-262-19448-1. URL `http://research.microsoft.com/~jplatt/abstracts/SVprob.html`.

D. Price, S. Knerr, L. Personnaz, and G. Dreyfus. Pairwise neural network classifiers with probabilistic outputs. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 1109–1116. MIT Press, Cambridge, MA, USA, 7 1995. ISBN 0-262-20104-6.

P. Price, W. M. Fisher, J. Bernstein, and D. S. Pallett. The DARPA 1000-word Resource Management database for continuous speech recognition. In *Proc. ICASSP*, volume 1, pages 651–654. IEEE, 1988.

L. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition.* Prentice Hall Signal Processing Series. Prentice Hall, Englewood Cliffs, NJ, USA, 1993. ISBN 0-13-015157-2.

R. Rifkin and A. Klautau. In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5:101–141, Dec. 2004. ISSN 1533-7928.

B. D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, Cambridge, UK, 1996. ISBN 0-521-46086-7.

R. Schlüter. *Investigations on Discriminative Training Criteria.* PhD thesis, Rheinisch-Westfälische Technische Hochschule, Aachen, Germany, Sept. 2000.

R. Schlüter and W. Macherey. Comparison of discriminative training criteria. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Proceedings*, volume 1, pages 493–496, 1998.

B. Schölkopf and A. J. Smola. *Learning with Kernels.* MIT Press, Cambridge, MA, USA, Dec. 2001. ISBN 0-262-19475-9.

B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors. *Advances in Kernel Methods: Support Vector Learning*. MIT Press, Cambridge, MA, 1999. ISBN 0-262-19416-3.

C. Schulze. Performance Tuning eines Spracherkennungssystems, Mar. 2005. Student project report, Otto-von-Guericke Universität Magdeburg.

D. W. Scott. *Multivariate density estimation: theory, practice, and visualization*. Wiley series in probability and mathematical statistics. John Wiley & Sons, Inc., 1992. ISBN 0-471-54770-0.

H. Shin and S. Cho. How to deal with large dataset, class imbalance and binary output in svm based response model. In *Proc. of the Korean Data Mining Conference*, pages 93–107, Dec. 2003.

A. J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pages 911–918, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1-55860-707-2.

A. J. Smola and B. Schölkopf. A tutorial on support vector regression. Technical Report 98-030, NeuroCOLT, Royal Holloway College, University of London, UK, 2003.

M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra. *MPI – The Complete Reference*, volume 1. MIT Press, Cambridge, MA, 2nd edition, 1998. ISBN 0-262-69215-5.

M. A. Stephens. EDF statistics for goodness of fit and some comparisons. *Journal of the American Statistical Association*, 69:730–737, 1974.

T. L. Sterling, J. Salmon, D. J. Becker, and D. F. Savarese. *How to build a Beowulf — A Guide to the Implementation and Application of PC Clusters*. Scientific and Engineering Computing. MIT Press, Cambridge, MA, USA, 1999. ISBN 0-262-69218-X.

J. A. K. Suykens, T. van Gestel, J. de Brabanter, B. de Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific Publishing Co. Pte. Ltd., Singapore, 2002. ISBN 9-812-38151-1.

A. N. Tikhonov. On solving ill-posed problems and method of regularization. *Doklady Akademii Nauk USSR*, 153:501–504, 1963.

M. E. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, June 2001.

E. Trentin and M. Gori. A survey of hybrid ANN/HMM models for automatic speech recognition. *Neurocomputing*, 37:91–126, 2001. ISSN 0925-2312.

I. V. Tsang, J. T. Kwok, and P.-M. Cheung. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6:363–392, Apr. 2005. ISSN 1533-7928. URL `http://portal.acm.org/citation.cfm?id=1058114{\&}coll=Portal{\&}dl=GUIDE{\&}CFID=50982226{\&}CFTOKEN=87107396`.

R. J. Vanderbei. LOQO: An interior point code for quadratic programming. *Optimization Methods and Software*, 11:451–484, 1999. ISSN 1055-6788.

V. N. Vapnik. *Statistical learning theory*. Adaptive and learning systems for signal processing, communications, and control. John Wiley & Sons, Inc., New York, NY, 1998. ISBN 0-471-03003-1.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Information Science and Statistics. Springer, Berlin, 2nd edition, 2000. ISBN 0-387-98780-0.

V. N. Vapnik and S. Mukherjee. Support vector method for multivariate density estimation. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 659–665. MIT Press, 2000. ISBN 0-262-19450-3.

A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967. ISSN 0018-9448.

J. von Neumann. First draft of a report on the EDVAC. Technical report, Moore School of Electrical Engineering, University of Pennsylvania, 1945.

G. I. Webb. Further experimental evidence against the utility of occam's razor. *Journal of Artificial Intelligence research*, 4:397–417, 1996.

A. Wendemuth. *Grundlagen der Stochastischen Sprachverarbeitung*. Oldenbourg, München, 2004. ISBN 3-486-27465-1.

R. C. Whaley, A. Petitet, and J. J. Dongarra. Automated empirical optimization of software and the ATLAS project. *Parallel Computing*, 27(1–2):3–35, 2001.

P. C. Woodland and D. Povey. Large scale discriminative training of hidden markov models for speech recognition. *Computer Speech & Language*, 16(1):25–47, Jan. 2002.

T.-F. Wu, C.-J. Lin, and R. Weng. Probability estimates for multi-class classification by pairwise coupling. *The Journal of Machine Learning Research*, 5:975–1005, Dec. 2004. ISSN 1533-7928.

S. Young, G. Evermann, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland. *The HTK Book*. Cambridge University Engineering Department, 2002.

B. Zadrozny. Reducing multiclass to binary by coupling probability estimates. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 1041–1048, Cambridge, MA, USA, 2002. MIT Press. ISBN 0-262-04208-8.