OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
EIT
FAKULTÄT FÜR ELEKTROTECHNIK UND INFORMATIONSTECHNIK
INSTITUT FÜR ELEKTRONIK, SIGNALVERARBEITUNG UND KOMMUNIKATIONSTECHNIK (IESK)

Machine Learning with Lipschitz Classifiers

DISSERTATION

zur Erlangung des akademischen Grades
Doktoringenieur (Dr.-Ing.)

von

Dipl.-Ing.(FH) André Stuhlsatz

geb. am 06.05.1975 in Düsseldorf

genehmigt durch die
Fakultät für Elektrotechnik und Informationstechnik
der Otto-von-Guericke-Universität Magdeburg

Gutachter: Prof. Dr. rer. nat. Andreas Wendemuth
Prof. Dr.-Ing. Rolf Findeisen
Prof. Dr. rer. nat. Hans-Günter Meier

Promotionskolloquium am 21.06.2010

Für Moni

Danksagung

Ich möchte mich ganz herzlich und mit besonderer Hochachtung bei meinem Doktorvater Prof. Dr. Andreas Wendemuth und meinem Mentor Prof. Dr. Hans-Günter Meier bedanken. Die vorliegende Dissertation wäre ohne die Unterstützung von Prof. Dr. Andreas Wendemuth und dessen Vertrauen in mein Forschungsvorhaben nicht möglich gewesen. Gleiches gilt für Prof. Dr. Hans-Günter Meier, der mich immer motiviert hat diesen Weg zu gehen. Seine Leidenschaft für die Mathematik, das Bestreben zu einer präzisen und klaren Darstellung, aber auch sein unermüdlicher Einsatz für die Lehre und die Förderung talentierter Studenten, bleibt mir immer ein Vorbild.

Ich danke der Fachhochschule Düsseldorf für ihre Unterstützung durch ein erstmalig eingerichtetes Promotionsförderprogramm, das es mir ermöglicht hat, in Kooperation mit der Otto-von-Guericke Universität Magdeburg, an der Fachhochschule Düsseldorf forschen zu dürfen. Namentlich danke ich hierfür besonders folgenden Personen der Fachhochschule Düsseldorf: Herrn Prof. Dr. Hans-Joachim Krause, Herrn Prof. Dr.-Ing. Detmar Arlt, Herrn Prof. Dr.-Ing. Andreas Jahr und Herrn Prof. Dr. Harald Jacques. In diesem Zusammenhang möchte ich auch Herrn Prof. Dr.-Ing. Thomas Zielke für seinen Einsatz und seine Unterstützung danken.

Des Weiteren danke ich meinen Kollegen in Düsseldorf und Magdeburg. Besonders erwähnen möchte ich die tolle Zusammenarbeit und die vielen interessanten Diskussionen und Projekte mit Daniel Gaida und Jens Lippel. Mögen noch viele spannende Projekte folgen!

Mein liebster und innigster Dank gilt meiner Frau Moni und meiner Familie, die immer an mich glauben und mich in allem mental unterstützen.

This dissertation has been promoted by a graduate support program of the University of Applied Sciences Düsseldorf (FH D).

Zusammenfassung

Die Klassifikation von komplexen Mustern ist eine der beeindruckendsten kognitiven Leistungen des menschlichen Gehirns. Ein Mensch ist in der Lage innerhalb einer halben Sekunde ein komplexes Bild, zum Beispiel das von einer bekannten Person, zu erkennen und von anderen Objekten zu unterscheiden. Während das Gehirn zur Lösung dieser Aufgabe auf eine massive Parallelität und ein riesiges, hierarchisch organisiertes und auto-assoziatives Gedächtnis zurückgreifen kann, sind gewöhnliche Rechnerarchitekturen nur zu einer sequentiellen Verarbeitung von Informationen aus einem nicht-assoziativen Speicher fähig. Selbst moderne, parallel verarbeitende, Mehrprozessorsysteme erreichen bei weitem nicht die Leistungsfähigkeit unseres Gehirns. Trotzdem ist es heute möglich mit Hilfe von modernen statistischen und algorithmischen Lernverfahren komplexe und speicheraufwendige Mustererkennungsprobleme, wie zum Beispiel das Erkennen von handgeschriebenen Ziffern oder die Transkribierung gesprochener Sprache, zufriedenstellend auf einem Rechner zu lösen.

Eines der erfolgreichsten Verfahren der maschinellen Mustererkennung ist die sogenannte *Support Vector Machine* (SVM). Die SVM basiert auf dem Lernparadigma der strukturierten Risikominimierung, welches sich gegenüber empirisch motivierten Ansätzen auszeichnet, wenn nur wenig Daten zur Lösung des betrachteten Klassifikationsproblems vorliegen. Obwohl die SVM häufig sehr gute Erkennungsleistungen zeigt, stößt sie auch auf Grenzen ihrer Anwendbarkeit, zum Beispiel wenn spezielles Vorwissen genutzt werden soll. Insbesondere immer komplexer werdende Anwendungen erfordern eine hohe Anpassbarkeit des Klassifikationsverfahren an die konkrete Problemstellung. Auch in diesem Punkt ist die SVM auf Grund einer sehr eingeschränkten Auswahl an implementierbaren Klassifikationsfunktionen limitiert.

Das Ziel der vorliegenden Dissertation ist die Entwicklung von neuen Lernalgorithmen zur Musterklassifikation, die einerseits über die Grenzen der SVM hinausgehen, aber andererseits trotzdem auf den gleichen theoretischen Konzepten aufbauen, welche die herausragende Erkennungsleistung der SVM erklären. Es werden zwei neue Algorithmen basierend auf einer theoretischen Verallgemeinerung der SVM vorgestellt, und erstmalig für eine praktische Implementierung nutzbar gemacht. Im Gegensatz zur SVM erschließen die neuentwickelten Verfahren eine sehr viel größere Funktionenklasse zum Aufbau eines Klassifikators. Dies ist eine wichtige Voraussetzung für eine flexible Anpassung des Klassifikators an schwierige Klassifikationsaufgaben mit speziellen Anforderungen sowie der Integration von Vorwissen über das zu lösende Problem.

Der Weg zu implementierbaren Algorithmen führt in dieser Arbeit über verschiedene mathematische Reformulierungen des Ausgangsproblems. Ausgehend von einer theoretischen Verallgemeinerung der SVM resultiert ein im Allgemeinen sehr schwierig zu lösendes restringiertes Optimierungsproblem. In einem ersten Schritt wird dieses, bei Auswahl einer sehr großen Funktionenklasse, bestehend aus (Affin-)Linearkombinationen von mindestens einmal stetig differenzierbaren Funktionen, durch ein restringiertes Minimax-Problem reformuliert. Im nächsten Schritt wird dann das Minimax-Problem weiter in ein sogenanntes *semi-infinites Problem* (SIP) überführt. Es zeigt sich, dass diese spezielle mathematische Problemstellung geeignet ist, um mit bekannten Optimierungsmethoden eine Lösung des Ausgangsproblems für die betrachtete Funktionenklasse numerisch zu bestimmen. Um die Problemstruktur noch weiter auszunutzen, wird aus dem SIP ein gleichgestelltes duales Problem hergeleitet. Hierfür beweisen wir einen *Dualitätssatz* über die Gleichheit der Optimalwerte des dualen Problems und des Ausgangsproblems.

Zur Lösung des dualen Problems wird ein mehrstufiges iteratives Verfahren entwickelt, aus dem durch die Verfolgung unterschiedlicher Lösungsstrategien die vorgeschlagenen Algorithmen resultieren. Darüber hinaus werden alle für eine Softwareimplementierung notwendigen Teiloptimierungsverfahren jeder Stufe entwickelt. Hierzu gehören ein angepasstes *Innere-Punkt-Verfahren*, eine auf *Simulated Annealing* basierende Suchheuristik und ein spezielles *Gradientenabstiegsverfahren*. Außerdem werden Möglichkeiten zur Effizienzsteigerung für zukünftige Implementierungen aufgezeigt.

Neben dem Schwerpunkt der theoretischen Entwicklung von neuen Lernverfahren und deren praktischen Realisierungen wurden alle Algorithmen für den experimentellen Teil dieser Arbeit in der MATLAB® Programmierumgebung implementiert. Damit stehen diese auch für zukünftige Forschungsvorhaben zur Verfügung.

Erste Klassifikationsergebnisse mit den neuen Algorithmen werden im Vergleich zur SVM auf unterschiedlichen Datensätzen untersucht und bewertet. Als Testdaten wurden hierfür ein künstlich erstellter 2D-Datensatz, sowie zwei reale Anwendungsdatensätze verwendet. Im abschließenden Experiment wird beispielhaft ein Szenario betrachtet, auf das die SVM unzureichend anwendbar ist und das gerade hier die Einsatzfähigkeit der neuen Verfahren belegen soll.

Resultierend kann gesagt werden, dass die vorgestellten Lernverfahren in Standardanwendungen ähnlich gute Klassifikationsergebnisse wie die SVM auf den hier verwendeten Datensätzen erreichen. Der besondere Nutzen der neuen Verfahren zeigt sich theoretisch und experimentell jedoch in der Fähigkeit Klassifikationsprobleme mit Entscheidungsfunktionen zu lösen, die der SVM verschlossen sind. Dabei werden die zu Grunde liegenden Ideen übertragen,

welche die SVM gegenüber vielen anderen Verfahren, bezüglich Generalisierungsfähigkeit bei wenig Lerninformation, auszeichnet. Dies eröffnet erstmalig die Möglichkeit des Designs und der Nutzung von neuen Klassifikatoren, die bisher in einem als robust und generalisierungsfähig erwiesenen Grundkonzept wie der SVM nicht realisierbar sind.

Abstract

The classification of complex patterns is one of the most impressive cognitive achievements of the human brain. Humans have the ability to recognize a complex image, like for example that of a known person, and to distinguish it from other objects within half a second. While for a solution of this task the brain has access to a massive parallelism and a vast, hierarchically organized, and auto-associative memory, common computer architectures are just able to a sequential processing of information stored in a non auto-associative memory. Even modern, parallelly operating, multi-processor systems are far away from the performance of our brain. However, nowadays, it is possible to solve complex and memory extensive pattern recognition problems, like the recognition of handwritten digits or the transcription of speech, satisfactorily with a common computer by the use of modern statistical and algorithmic learning approaches.

One of the most successful pattern recognition methods is the so-called *Support Vector Machine* (SVM). The SVM is based on the learning paradigm of structural risk minimization, which outperforms empirical approaches if only few data is available for solving the considered classification problem. Although the SVM has proven very good recognition performances in many cases, the SVM also comes up with limitations, for example if specific a priori knowledge shall be used. In particular, the increasing complexity of applications requires a high adaptivity of the classification method to the specific problem. Also concerning this point, the SVM is limited due to a restricted variety of implementable classification functions.

The objective of the present thesis is the development of new learning algorithms for the classification of patterns, that on the one hand overcome the limitations of the SVM, but on the other hand are based on the same theoretical concepts facilitating the good performance of the SVM. Two new algorithms will be presented that are justified by a theoretical generalization of the SVM, and which will be utilized for the first time for a practical implementation. In contrast to the SVM, the new methods make accessible a much larger function class for constructing a classifier. This is an important prerequisite for flexible adaptation of the classifier to difficult classification tasks with particular requirements as well as for the integration of a priori knowledge about the problem at hand.

In this work, the way to implementable algorithms leads across different mathematical reformulations of the original problem. Starting with the theoretical generalization of the SVM, it results a restricted optimization problem

that is difficult to solve in general. In a first step, this problem is expressed in terms of a restricted minimax-problem by a modification of the suitable classification functions to a still very large function class consisting of (affine-)linear combinations of at least one-time continuously differentiable functions. In the next step, the minimax-problem is converted into a so-called *Semi-Infinite Problem* (SIP). It turns out, that this particular mathematical problem is appropriate in order to obtain a solution of the original problem for the considered function class using well-known optimization methods. To further exploit the problem structure, an equivalent dual problem is derived from the SIP. Therefore, we prove a *duality theorem* about the equality of the optimal values of the dual and the original problem.

For solving the dual problem, a multilevel iterative approach is developed from which the proposed algorithms follow by pursuing different solution strategies. Moreover, all sub-optimization methods of any stage necessary for an implementation in software are developed. Namely, these are an adapted *interior-point-method*, a *simulated annealing* based search heuristics and a particular *gradient decent approach*. Furthermore, options are depicted for an improvement of efficiency for future implementations.

Besides the emphasis on the theoretical development of new learning methods and their practical implementations, all algorithms were implemented in the MATLAB® programming environment for the experimental part of the present thesis. Hence, they are also available for further research purposes in future.

For the first time, classification results are explored and evaluated in comparison to the SVM on different data sets. As test data, an artificial 2d-dataset as well as two real-world datasets were used. In the concluding experiment, a scenario is prototypically considered to which the SVM is only inadequately applicable and which shall precisely prove the capability of the new methods in that case.

It follows, regarding the considered datasets, the proposed learning methods reach comparably good classification accuracy like the SVM in standard applications. Moreover, the particular benefit of the new methods is reflected theoretically and experimentally in the ability to solve classification problems using decision functions that are not accessible to SVMs. Thereby, the underlying ideas, which make the SVM excel compared to other approaches with respect to generalization performances in case of few available learning information, are adequately transported into the proposed new environment. This opens the way for a design and a use of new classifiers that have not been implementable in a robust and generalizing basic concept so far.

Contents

List of Abbreviations

ANN	Artificial Neural Network
ASR	Automatic Speech Recognition
CPD	Conditional Positive Definite
ERM	Empirical Risk Minimization
EV	Eigenvalue Decomposition
FERET	Facial Recognition Technology
HMM	Hidden-Markov-Model
i.i.d.	independently and identically-distributed
iff	if and only if
IP	Interior-Point
KKT	Karush-Kuhn-Tucker
LDA	Linear Discriminant Analysis
MFCQ	Mangasarian-Fromovitz Constraint Qualification
NIST	National Institute of Standards and Technology
OCR	Optical Character Recognition
OGI	Oregon Graduate Institute
PCQP	Primal Convex QP-Problem
QP	Quadratic Problem
RKHS	Reproducing Kernel Hilbert Space
SA	Simulated Annealing
SIP	Semi-Infinite Program
SRM	Structural Risk Minimization
SVM	Support Vector Machine
SV	Support Vector
USPS	U.S. Postal Service
VSV	Virtual Support Vector

List of Algorithms

List of Figures

List of Tables

List of Symbols

$\langle \cdot, \cdot \rangle$	scalar product or dot product
$\| \cdot \|_\infty$	supremum norm $\|f\|_\infty := \sup_{\boldsymbol{x} \in \mathcal{X}} \|f(\boldsymbol{x})\|$
$\succ 0$ ($\succeq 0$)	positive (semi-)definite
$0 \prec$ ($0 \preceq$)	negative (semi-)definite
$\overline{\mathcal{A}}$	closure of a space $\mathcal{A}$
$\mathbf{I}_n$	$n \times n$ identity matrix
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	multivariate Gaussian density with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$
$\mathbb{K}$	field $\mathbb{R}$ or $\mathbb{C}$
$\mathcal{X}$	data space
$\mathcal{Y}$	category space
$\mathcal{Z} := \mathcal{X} \times \mathcal{Y}$	space of training data
$\mathcal{P}$	power set
$\mathcal{F}$	decision function space
$\mathcal{M}$	metric space
$\mathcal{H}$	Hilbert space
$\mathcal{B}$	Banach space
$\mathcal{K}$	the set of all positive semi-definite matrices $\mathbf{K}(\boldsymbol{x})$ defined on $\mathcal{X}$, i.e. $\mathcal{K} := \left\{ 0 \preceq \mathbf{K}(\boldsymbol{x}) \in \mathbb{R}^{M \times M} \mid \boldsymbol{x} \in \mathcal{X} \right\}$.
$\mathcal{C}_b(\mathcal{X}, \mathbb{R})$	space of continuous and bounded functions $f : \mathcal{X} \to \mathbb{R}$
$\mathcal{C}(\mathcal{X}, \mathbb{R})$	space of continuous functions $f : \mathcal{X} \to \mathbb{R}$ defined on a compactum $\mathcal{X}$
$\mathcal{C}^{(n)}(\mathcal{X}, \mathbb{R})$	space of n-times continuously differentiable functions $f : \mathcal{D}_\mathcal{X} \to \mathbb{R}$, $\mathcal{X} \subset \mathcal{D}_\mathcal{X}$, $\mathcal{D}_\mathcal{X} \subseteq \mathbb{R}^m$ open
L_p	$L_p := \left\{ f : \mathcal{X} \to \mathbb{K} \mid (\int_\mathcal{X} \|f(\boldsymbol{x})\|^p d\mu(\boldsymbol{x}))^{1/p} < \infty \right\}$
$\mathcal{L}IP(\mathcal{M})$	space of Lipschitz continuous functions w.r.t a metric space $\mathcal{M}$ with bounded metric
$h \in \mathbb{N}$	VC-dimension
$\boldsymbol{z}_i \in \mathcal{Z}$	sample
$\mathcal{O}_l \subset \mathcal{Z}$	random sample $\{\boldsymbol{z}_1, ..., \boldsymbol{z}_l\}$
$G_\mathcal{F} : \mathbb{N} \to \mathbb{R}$	growth-function of a set of functions $\mathcal{F}$
$H_\mathcal{F} : \mathbb{N} \to \mathbb{R}$	entropy of a set of functions $\mathcal{F}$
$N_\mathcal{F} : \mathcal{P}(\mathcal{Z}) \to \mathbb{N}$	capacity of a set of functions $\mathcal{F}$
$R_l : \mathcal{F} \to \mathbb{R}$	empirical risk using a sample $\mathcal{O}_l$ of size l
$R : \mathcal{F} \to \mathbb{R}$	actual risk
$L_f^{0\|1} : \mathcal{Z} \to \{0, 1\}$	zero/one loss function
$L_f : \mathcal{Z} \to \mathbb{R}$	loss function

$\mathbf{K} : \mathcal{X} \to \mathbb{R}^{M \times M}$	$\mathbf{K}(\boldsymbol{x})_{m,n} := \langle \nabla \Phi_m(\boldsymbol{x}), \nabla \Phi_n(\boldsymbol{x}) \rangle_2$
$\overline{\mathbf{K}} : [0,1] \to \mathbb{R}$	$\overline{\mathbf{K}}(\mu) := (1-\mu) \cdot \mathbf{K}_{(t)} + \mu \cdot \mathbf{K}^*$
$Q : \mathbb{R}^N \times \mathbb{R}^M \times \mathbb{R}^{M \times M} \to \mathbb{R}$	$Q(\boldsymbol{\alpha}, \boldsymbol{c}, \boldsymbol{K}) := \boldsymbol{\alpha}^T \mathbf{1}_N - \frac{1}{2} \boldsymbol{c}^T \mathbf{K} \boldsymbol{c}$
$G : \mathbb{R}^N \times [0,1] \to \mathbb{R}$	$G(\boldsymbol{\alpha}, \mu) := -Q(\boldsymbol{\alpha}, \boldsymbol{c}(\boldsymbol{\alpha}, \mu), \overline{\mathbf{K}}(\mu))$
$q : \mathbb{R}^{M \times M} \to \mathbb{R}$	$q(\mathbf{K}) := \max\limits_{(\boldsymbol{\alpha}, \boldsymbol{c}) \in \mathcal{S}_D(\mathbf{K})} Q(\boldsymbol{\alpha}, \boldsymbol{c}, \boldsymbol{K})$
$g : [0,1] \to \mathbb{R}$	$g(\mu) := -q(\overline{\mathbf{K}}(\mu))$

Chapter 1

Introduction and Motivation

"I believe that something drastic has happened in computer science and machine learning. Until recently, philosophy was based on the very simple idea that the world is simple. In machine learning, for the first time, we have examples where the world is not simple." - Vladimir N. Vapnik (2008)

Machine learning is concerned with the design and development of algorithms that allow computers to learn based on data. In the late 1950s, the scientific discipline of machine learning was in its infancy. Due to the available primitive computer technology at that time, most research was almost of theoretical nature. A first step was made with the development of the Perceptron [Rosenblatt 1958], the Pandemonium [Selfridge 1959] and the Adaline [Widrow 1962]. Due to the large number of research efforts in this area, a next stage was to leave the pattern classification setting as represented by Perceptrons and to concentrate in the development of associative memories based on the idea the brain stores information in the connections of millions of neurons. The first publication on associative memories was due to [Kohonen 1972], and 1982 so-called Hopfield Association Networks were introduced [Hopfield 1982]. In parallel, the experience with the early stages of neural networks spawned the new discipline of pattern recognition and led to the development of a decision-theoretic approach to machine learning. In the center of this approach was the supervised learning of discriminant functions from training data. One of the best known successful learning systems using generalized discriminant functions was a computer checkers [Samuel 1959]. Remarkably, the checkers program reached master-level performance via repeated training. The expeditious growth of pattern recognition methods utilizing statistical decision theory led to the excellent and well-known classical textbooks [Fukunaga 1972] and [Duda 1973]. In 1975 still inspired by the self-organizing ability of the brain, the Cognitron network was introduced as an extension of Perceptrons [Fukushima 1975]. The Cognitron was the first developed multilayer neural network. A leverage was the presentation of the Back-propagation algorithm for learning multilayer networks in 1986. The Back-popagation algorithm was found by three independent research groups [Rumelhart 1986], [Parker 1985], [LeCun 1985].

During 1960 and 1970 the theoretical branch of machine learning, namely the statistical learning theory, was developed. The concepts of quantifying learning machines for pattern recognition in terms of their separation ability were introduced in 1969. Based on these concepts, namely VC-Entropy and VC-Dimension, bounds for the rate of uniform convergence of empirical estimates of the risk of misclassification to the unknown true risk were found [Vapnik 1968], [Vapnik 1971]. These bounds made a novel learning concept possible. Thitherto, the empirical risk minimization principle was prevailing in pattern recognition, but the new learning paradigm called structural risk minimization inductive principle completed the development of learning theory [Vapnik 1974]. The statistical learning theory cumulated in conditions for the consistency of the empirical risk minimization in the end of the 1980s [Vapnik 1989].

In the 1980s, Vapnik and co-workers introduced the first learning machine implementing the structural risk minimization principle - the celebrated Support Vector Machine (SVM). At this time, the SVM was largely unnoticed, although justified by a strong theoretical background. The reason was the widespread belief that SVMs are neither suitable nor relevant for practical applications. This belief changed in the early 1990, because very good results using SVMs on a handwritten digit recognition task were reported (see e.g. [Vapnik 1999]). During the 1990, a great progress was made in the development of efficient SVM learning algorithms even for problems with millions of training data. Until now, many different classification problems have been solved successfully using SVMs. The success of SVMs is rooted in the statistical properties of a maximum margin separating hyperplane, as well as in its ability to efficiently learn a separating hyperplane in feature spaces of very high dimension via a mathematical technique, distantly called the "kernel-trick".

Nowadays, due to the SVM's superiority compared to other approaches, almost any well-known pattern recognition method is "kernelized", if possible, to profit from the "kernel-trick" as well. Examples are PCA, LDA, clustering or Bayesian methods. In order to apply any kernel-method, the kernel has to be chosen beforehand due to validation results or based on some knowledge of the problem at hand. As representer of the structure of a feature space, kernels have to satisfy mathematical constraints limiting any kernel-machine, in particular the SVM. Unfortunately, only a handful of functions are proved to be kernels, and thus are available for the SVM. On the other hand, whenever prior knowledge is present, it should be incorporated some how into the kernel such that the additional information is useful for the learning method. This is often not possible for the desired functions. For example, in DNA microarray analysis, similarity measures best suited to distinguish and emphasize specific

patterns of gene expressions can not be directly used to compose the decision function of SVMs. Similar, in applications where generative models are used to model varying features, like in speech recognition, information theoretic measures of density functions would be appropriate, but can not be employed in SVMs.

Using (dis-)similarities or any other non-kernel function in SVMs is only possible indirectly by unsatisfactory distortions of the original function such that it is turned into a valid kernel function. Moreover, if violating the assumption of a kernel function, a good generalization performance is not justified by statistical learning theory and a maximum margin concept anymore, nor is the SVM algorithm guaranteed to find a global solution of the classification problem at all. On the one hand, overcoming the limitation to kernel functions, and on the other hand, maintaining the generalizing properties rooted in a maximum margin classification, requires to translate the margin concept to more general feature spaces than assumed by the SVM framework. So far, only few publications address the issue of learning a separating hyperplane with a maximizing margin in more general spaces, theoretically, and almost no algorithms implementable in practice are discussed.

The main focus of the present thesis is to overcome the lack of practical algorithms for maximum margin classification beyond SVMs. We contribute to this goal with the development of new implementable maximum margin algorithms for learning separating hyperplanes in a very general setting. The new algorithms make accessible a larger class of decision functions for solving binary classification problems than it is the case for SVMs. Namely, the implementable decision functions are expanded to the space of finite affine-linear combinations of one-times continuously differentiable functions, of course including almost any kind of (dis-)similarity measure, kernel functions, polynomials, trigonometric functions and many other functions valuable for a given application.

Summary by Chapters

The present thesis is structured in eight chapters. We decided to start each chapter (except, introduction and conclusion) with a mini table of contents so that the reader keeps the overview, and we feel it is advantageous when revisiting several sections. Additionally, each chapter concludes with a short summary. This Chapter (1) reviews the history of machine learning and gives the motivation for our research. The following chapters are build on top of each other, but Chapter (2) and (3) can be skipped if the reader is familiar with the topics therein. We start with a general survey of pattern classification

in Chapter (2), in order to make the reader familiar with pattern classification and methods. The survey ends with an introduction of Support Vector Machines on a basic level focusing more on applications. The next Chapter (3) introduces statistical learning theory for pattern classification. Statistical learning theory is crucial for the argumentation that among separating hyperplanes the maximum margin hyperplanes are to prefer for classification. The chapter concludes with a detailed discussion on SVMs. Although SVMs have shown very good classification performances, they also have a shortcoming we discuss in Chapter (4). After accepting the limitations, we review research with the objective to overcome the drawback of SVMs, that is related work. Chapter (5) is dedicated to the main focus of this thesis - the development of new maximum margin algorithms for binary classifications beyond SVMs. We introduce the theoretical framework our research is based on. Then we show how to derive implementable algorithms from the theory. In Chapter (6), we extensively discuss the technical details for an implementation. Given different implementations of the new maximum margin algorithms, Chapter (7) presents results obtained by experiments. The thesis ends with Chapter (8) giving the conclusion, open issues and future work. Last but not least, in the Appendix the reader can find some important mathematical background, some proofs and additional information to some sections.

Chapter 2

Pattern Classification: A General Survey

Contents

The following chapter gives an overview on pattern classification in general. Starting with the classification of data using linear models, we introduce subsequently the most prominent non-linear generalizations, namely Artificial Neural Networks (ANNs) and the Support Vector Machines (SVMs). Because pattern recognition is a broad field, we just scratch the surface here in the context of the present thesis. For more details, we refer the reader to e.g. [Fukunaga 1990], [Duda 2000], [Hastie 2001] and [Bishop 2006].

2.1 Recognition Systems

A general recognition system can be thought of as a black-box, in which information about the observed reality is collected and processed in order to derive some decision. For example, imagine an e-mail spam filter. In this case,

the information about the reality is manifested in the words or word sequences of the e-mail under consideration. The spam filter (recognition system) has to produce a decision based on the observed words. A decision could be to move an e-mail into the recycle bin, if the e-mail is categorized as spam because it contains one of the most frequently occurring spam words like adviser, credit-card, blackjack, etc., or sequences of such words. Clearly, in order to make a careful decision, the spam filter has to extract the essential information (*features*) out of the whole e-mail, like e.g. signal words, frequency of words and/or dependencies to other words or even e-mails received in the past. Therefore, this part of a recognition system is called the *feature extractor*. Ideally, a feature extractor provides features that are invariant to irrelevant transformations of the input data. Using a set of extracted features, a second part of the overall system - the *classifier* - has to categorize the objects into predefined classes. In our spam filter example, these classes are defined to be spam or not spam. Another possibility could be two have a ranking of spam mails for different grades of vulnerability, which would result in more than two classes.

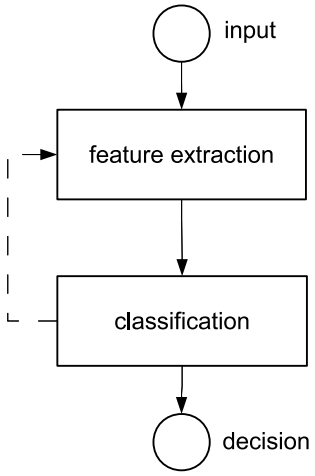


Figure 2.1: Sketch of a typical recognition system divided into its fundamental functionalities. The feature extraction provides distinguishing features, that are invariant to irrelevant transformations of the input. Given a set of features of an input object, the classification has to assign the object to a category in order to make a decision. Some systems also use some kind of feedback (dashed line) from the classification to refine the feature extraction.

Summarizing, a recognition system can be divided into two coarse functional parts - *feature extraction* and *classification* (Fig. 2.1). Each part has its own requirements and could itself further decomposed into meaningful sub-functionalities. Although very important, we do not further discuss the

feature extraction. We address exclusively the classification starting with the simplest one, namely the linear classification.

2.2 Linear Classifiers

Consider two set of feature vectors $\mathcal{X}_1, \mathcal{X}_2 \subset \mathbb{R}^m$ extracted from observed objects of two categories ω_1, ω_2. An example in two dimensions is plotted in figure (2.2). The figure shows two separable clouds of points that can be perfectly

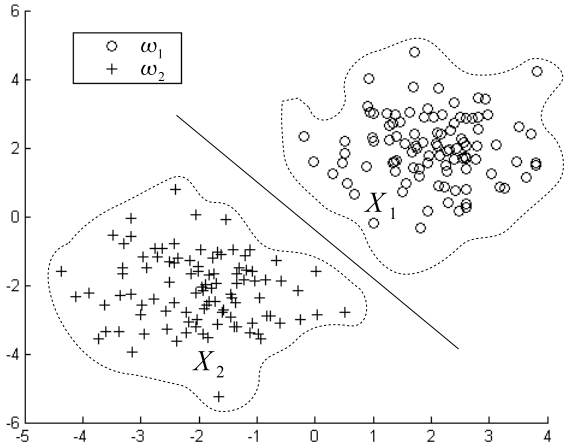


Figure 2.2: An example showing two clouds of points separated by a linear (solid line) and non-linear (dashed line) decision boundary.

classified using (linear) *decision boundaries* $\{\boldsymbol{x} \in \mathbb{R}^2 \ : \ g(\boldsymbol{x}) = 0\}$ implied by a (linear) *discriminant function* $g : \mathbb{R}^2 \to \mathbb{R}$. In general, classification is faced with the problem to seek for a discriminant function $g : \mathbb{R}^m \to \mathbb{R}$ implying a decision boundary which classifies with a low error rate not just the observed objects but also the unobserved prospective ones. For example, the dashed decision boundary in Figure (2.2) is more unlikely to classify new data points correctly than the solid one. The dashed boundary does not *generalize* well because of *overfitting* to the training data.

A statistical approach for selecting the best possible decision boundary is to assume a particular probability model $P(\omega_i|\boldsymbol{x}) \propto p(\boldsymbol{x}|\omega_i)P(\omega_i) =: g_i(\boldsymbol{x})$ for each class ω_i given a data point $\boldsymbol{x} \in \mathbb{R}^m$. The probability density $p(\boldsymbol{x}|\omega_i)$ is also called *likelihood*. With a predefined probability model, a data point is classified to the class with highest probability, i.e. $\boldsymbol{x}$ belongs to class ω_i if $g_i(\boldsymbol{x}) > g_j(\boldsymbol{x})\ \forall i \neq j$. This approach is motivated by the *Bayesian Decision Theory* and yields the lowest expected error provided the probability model

represents the true nature of data, there are equal costs of misclassification and assuming independently drawn data points. In particular, the assumption of a Gaussian model $p(\boldsymbol{x}|\omega_i) \propto \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma})$ with mean $\boldsymbol{\mu}_i \in \mathbb{R}^m$ and diagonal covariance matrix $\boldsymbol{\Sigma} := \sigma^2 \mathbf{I}_m$ leads to the (affine-)linear function

$$\ln\left(g_i(\boldsymbol{x})\right) = \boldsymbol{w}_i^T \boldsymbol{x} + b_i + c(\boldsymbol{x}) \tag{2.1}$$

with

$$\boldsymbol{w}_i := \frac{1}{\sigma^2}\boldsymbol{\mu}_i, \;\; b_i := -\frac{1}{2\sigma^2}\boldsymbol{\mu}_i^T \boldsymbol{\mu}_i + \ln\left(P(\omega_i)\right). \tag{2.2}$$

Note, $c(\boldsymbol{x})$ is independent of the parameters $(\boldsymbol{w}_i, b_i)$. Because $g_i(\boldsymbol{x}) > g_j(\boldsymbol{x}) \Leftrightarrow \ln\left(g_i(\boldsymbol{x})\right) > \ln\left(g_j(\boldsymbol{x})\right)$, the decision boundary between two classes ω_i and ω_j is implied by the (affine-)linear discriminant function

$$g(\boldsymbol{x}) = (\boldsymbol{w}_i - \boldsymbol{w}_j)^T \boldsymbol{x} + b_i - b_j. \tag{2.3}$$

Rewriting the boundary equation $g(\boldsymbol{x}) = 0$ as

$$\boldsymbol{w}^T(\boldsymbol{x} - \boldsymbol{x}_0) = 0 \tag{2.4}$$

with

$$\boldsymbol{w} := \boldsymbol{\mu}_i - \boldsymbol{\mu}_j, \tag{2.5}$$

$$\boldsymbol{x}_0 := \tfrac{1}{2}(\boldsymbol{\mu}_i + \boldsymbol{\mu}_j) - \tfrac{\sigma^2}{\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2} \ln\left(\frac{P(\omega_i)}{P(\omega_j)}\right)(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j) \tag{2.6}$$

shows, that the decision boundary is a hyperplane intersecting $\boldsymbol{x}_0$ perpendicular to the line going through the means, i.e. orthogonal to $\boldsymbol{w}$ (Fig. 2.3).

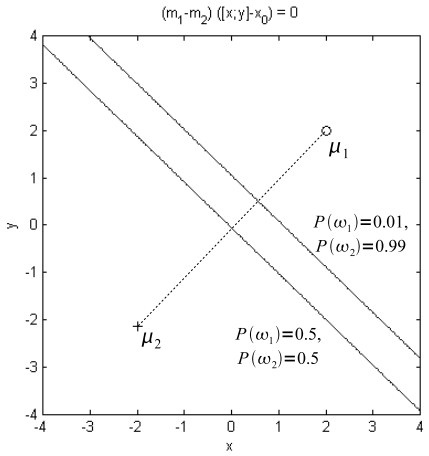


Figure 2.3: Separating hyperplanes implied by Bayesian linear discriminant functions. The hyperplanes are perpendicular to the line going through the means $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$ of each class ω_1, ω_2 shown in Fig. (2.2). If the prior probabilities are not equal, then the intersection point $\boldsymbol{x}_0$ shifts away from the most likely class.

In particular, if $P(\omega_i) = P(\omega_j)\ \forall i, j$ each decision boundary is the perpendicular bisector of the line between the corresponding means (Fig. 2.3) and the decision rule simplifies to the *minimum-distance classifier*: classify $\boldsymbol{x}$ to the class ω_i for which the distance $\|\boldsymbol{x} - \boldsymbol{\mu}_i\|$ is minimal. The minimum distance classifier can also be considered as a template matching method, where the means are prototypes of the different objects.

2.3 Learning of Discriminant Functions

So far, we assumed to know the true prior probabilities $P(\omega_i)$ and class-conditional probabilities $p(\boldsymbol{x}|\omega_i)$. Unfortunately, often one has just a vague knowledge of the true reality. Thus, the question arise how to estimate these probabilities appropriately, that means such that the decision boundary is as near as possible to the optimal one in the Bayesian sense? The answer to this question is in the heart of *statistical learning theory* (Chapter 3). Next, we review a very important approach for estimating parametric probability models, and in particular we introduce methods for learning discriminant functions directly from data. The latter is the basis for the development of very successful learning algorithms.

2.3.1 Maximum Likelihood Method

A prominent statistical approach for the estimation of parametric probability models is the so-called *maximum likelihood* method: Given N_i independently drawn samples $\mathcal{X}_{N_i} := (\boldsymbol{x}_1^i, \ldots, \boldsymbol{x}_{N_i}^i)$ for each class ω_i, and some parametric form of the class-conditionals $p(\boldsymbol{x}_n^i|\omega_i) = p(\boldsymbol{x}_n^i|\boldsymbol{\lambda}_i)$ with parameters $\boldsymbol{\lambda}_i \in \boldsymbol{\Lambda}$, the *likelihood* of the observations of class ω_i reads as

$$\mathcal{L}(\boldsymbol{\lambda}_i) := p(\mathcal{X}_{N_i}|\boldsymbol{\lambda}_i) = \prod_{n=1}^{N_i} p(\boldsymbol{x}_n^i|\boldsymbol{\lambda}_i). \tag{2.7}$$

A maximum likelihood estimate $\boldsymbol{\lambda}_i^*$ is then given by

$$p(\mathcal{X}_{N_i}|\boldsymbol{\lambda}_i^*) = \max\left\{\mathcal{L}(\boldsymbol{\lambda}_i)\ :\ \boldsymbol{\lambda}_i \in \boldsymbol{\Lambda}\right\}. \tag{2.8}$$

In the Gaussian case $p(\boldsymbol{x}_n^i|\boldsymbol{\lambda}_i) \propto \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ with mean $\boldsymbol{\mu}_i$ and covariance matrix $\boldsymbol{\Sigma}_i$, the maximum likelihood estimates are easily derived using differential calculus and yield:

$$\boldsymbol{\mu}_i^* = \frac{1}{N_i}\sum_{n=1}^{N_i} \boldsymbol{x}_n^i \tag{2.9}$$

$$\boldsymbol{\Sigma}_i^* = \frac{1}{N_i}\sum_{n=1}^{N_i} (\boldsymbol{x}_n^i - \boldsymbol{\mu}_i^*)(\boldsymbol{x}_n^i - \boldsymbol{\mu}_i^*)^T. \tag{2.10}$$

The maximum likelihood estimates equal the empirical estimators of mean and covariance well-known from statistics. Note, assuming a class-common covariance matrix $\boldsymbol{\Sigma}_i = \boldsymbol{\Sigma} \forall i$, the corresponding discriminant function is again (affine-)linear (2.3), but with parameters $\boldsymbol{w}_i = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i$ and $b_i := -\frac{1}{2}\boldsymbol{\mu}_i^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i + \ln(P(\omega_i))$. Thus, the induced hyperplane is in general not perpendicular to the line through the means anymore.

The maximum likelihood approach also applies to the prior class probabilities. For this purpose define a random variable

$$y_i(\boldsymbol{x}) := \begin{cases} 1 & \text{if } \omega(\boldsymbol{x}) = \omega_i \\ 0 & \text{else} \end{cases} \tag{2.11}$$

with the sample's class $\omega(\boldsymbol{x})$. Let be $\lambda_i := P(\omega_i)$, then because it holds

$$P(y_i(\boldsymbol{x}) = 1|\lambda_i) + P(y_i(\boldsymbol{x}) = 0|\lambda_i) = 1, \tag{2.12}$$

the likelihood of the parameter λ_i using a complete observation $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ with $N = \sum_{i=1}^{c} N_i$ and $N_i := card\{1 \leq j \leq N : \omega(\boldsymbol{x}_j) = \omega_i\}$, i.e the number of samples $\boldsymbol{x}_j$ of class ω_i , is given by

$$\mathcal{L}(\lambda_i) := P((y_i(\boldsymbol{x}_n))_{n=1}^N|\lambda_i) = \prod_{n=1}^{N} \lambda_i^{y_i(\boldsymbol{x}_n)}(1-\lambda_i)^{1-y_i(\boldsymbol{x}_n)}. \tag{2.13}$$

The distribution $P((y_i(\boldsymbol{x}_n))_{n=1}^N|\lambda_i)$ is known as *Bernoulli* distribution. It is easy to see, the corresponding maximum likelihood estimate is $P^*(\omega_i) = \lambda_i^* = N_i/N$. Strongly related to the maximum likelihood method is *Bayesian Parameter Estimation*, in which a additional parametric prior on the parameters is considered, e.g. see [Duda 2000].

2.3.2 Least-Squares Method

Another approach to derive discriminant functions is not to assume some particular form of the underlying probability densities but instead to directly specify the discriminant function itself. After defining the parametric from of the discriminant function, the parameters are fitted to the training data. That means for (affine-)linear discriminant functions

$$g_i(\boldsymbol{x}) := \boldsymbol{w}_i^T\boldsymbol{x} + b_i \tag{2.14}$$

the weights $\boldsymbol{w}$ and the bias b are optimized with respect to a specific criterion function J. A criterion function J measures typically the classification error on a given training sample $\mathcal{X}_l := (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_l)$, $l \in \mathbb{N}$, $\boldsymbol{x}_j \in \mathcal{X} \subseteq \mathbb{R}^m$, $1 \leq j \leq l$.

A well-known approach for solving a c-class problem using linear discriminant functions is the *method of least squares*. For this purpose, one encodes the class $\omega(\boldsymbol{x}_n)$ of a training example $\boldsymbol{x}_n$ using a target vector $\boldsymbol{y}_n := (0, \ldots, y_{n,k}, \ldots, 0)^T \in \mathbb{R}^c$ with the k-th component set to $y_{n,k} = 1$, if $\omega(\boldsymbol{x}_n)$ equals the class ω_k. The remaining components are set to zero. The criterion function J of the method of least-squares is the *mean squared error*

$$J(\mathbf{A}) := \sum_{n=1}^{l} \|\boldsymbol{y}_n - \boldsymbol{g}(\boldsymbol{x}_n)\|^2 \tag{2.15}$$

with discriminant function

$$\boldsymbol{g}(\boldsymbol{x}_n) := \mathbf{A}\hat{\boldsymbol{x}}_n = \left(\boldsymbol{\alpha}_1^T \hat{\boldsymbol{x}}_n, \ldots, \boldsymbol{\alpha}_c^T \hat{\boldsymbol{x}}_n\right)^T = (g_1(\boldsymbol{x}_n), \ldots, g_c(\boldsymbol{x}_n))^T \tag{2.16}$$

where $\boldsymbol{\alpha}_i^T := (\boldsymbol{w}_i^T, b_i)$ is the i-th row of $\mathbf{A}$ and $\hat{\boldsymbol{x}}_n := (\boldsymbol{x}_n^T, 1)^T$ is an augmented feature vector. The associated parameter optimization can be stated as:

$$\min_{\mathbf{A}} J(\mathbf{A}) = \min_{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_c} \sum_{n=1}^{l} \sum_{k=1}^{c} (y_{n,k} - \boldsymbol{\alpha}_k^T \hat{\boldsymbol{x}}_n)^2 \tag{2.17}$$

If $\hat{\mathbf{X}} := (\hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_l) \in \mathbb{R}^{(m+1)\times l}$ is of full rank, then the well-known solution of this multivariate regression is

$$\mathbf{A}^* = \mathbf{Y}\hat{\mathbf{X}}^T(\hat{\mathbf{X}}\hat{\mathbf{X}}^T)^{-1} = \mathbf{Y}\hat{\mathbf{X}}^+. \tag{2.18}$$

The matrix $\hat{\mathbf{X}}^+$ denotes the right-pseudo inverse of $\hat{\mathbf{X}}$, and $\mathbf{Y} := (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_l) \in \mathbb{R}^{c\times l}$ is called target coding matrix. Using the optimized parameters $\mathbf{A}^*$ a new sample can be classified simply by

$$\omega(\boldsymbol{x}) = \omega_{\arg\max_k \left\{(\boldsymbol{\alpha}_k^*)^T \cdot (\boldsymbol{x}^T, 1)^T\right\}}. \tag{2.19}$$

The least-squares approach is attractive, because it yields a closed-form solution. However, the resulting discriminant function suffers from severe problems.

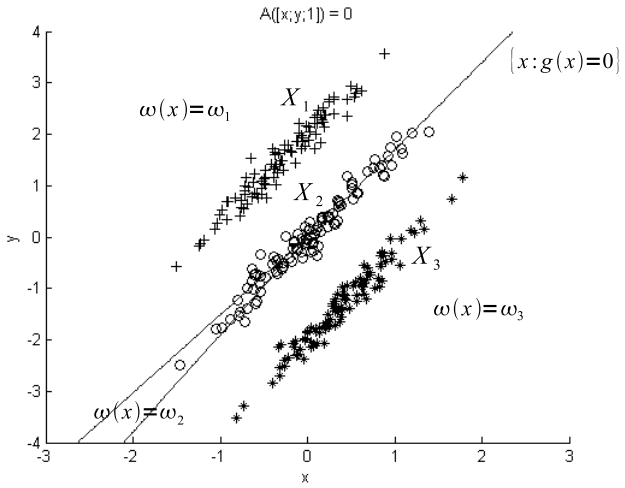


Figure 2.4: A negative example of a solution found by the method of least-squares. The plot shows points $\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3$ of three-classes $\omega_1, \omega_2, \omega_3$. The decision boundary $\{\boldsymbol{x} : \boldsymbol{g}(\boldsymbol{x}) = \mathbf{0}\}$ obtained by the method of least-squares strongly failures in classifying points from class ω_2.

A negative example of a three-class problem solved by the method of least-squares is shown in figure (2.4). In the general case of multi-class problems, one has to take into account all inequalities $g_i(\boldsymbol{x}) > g_j(\boldsymbol{x})\ \forall i \neq j$ in order to classify a sample $\boldsymbol{x}$ into a class ω_i. It follows for a three-class problem a decision boundary that divide the feature space into three regions associated to each class. Unfortunately, the multi-class decision from the method of least-squares shown in figure (2.4) classifies poorly the class ω_2. The failure of least-squares can be explained via its correspondence to a maximum likelihood estimation under the assumption of unimodal Gaussian class conditional probabilities (cf. 2.2). Obviously, the distribution of binary target vectors is far away from a unimodal Gaussian. To circumvent this problem, one approach is to use another criterion function J. For example the minimization of the *cross-entropy function*

$$J(\mathbf{A}) := -\sum_{n=1}^{l}\sum_{k=1}^{c} y_{n,k} \ln\left(\sigma(\boldsymbol{\alpha}_k^T \hat{\boldsymbol{x}}_n)\right) \tag{2.20}$$

with $\sigma(g_k) := \exp(g_k)/\sum_{i=1}^{c}\exp(g_i)$, which yields the so-called multi-class *logistic regression* approach. In opposite to least-squares, the logistic regression approach introduces nonlinearities in order to model more appropriate conditional probabilities. In Figure (2.5) the same three-class problem is shown as in Figure (2.4), but solved via logistic regression.

Very similar in essence, is a direct generalization of the linear function (2.14) by a nonlinear function $g_i(\boldsymbol{x}) := \boldsymbol{w}_i^T \boldsymbol{\Phi}(\boldsymbol{x}) + b_i$ with some non-linear

mapping $\mathbf{\Phi} : \mathcal{X} \to \mathbb{R}^p$. This idea is implemented for example in Artificial Neural Networks, presented in Section (2.4.1).

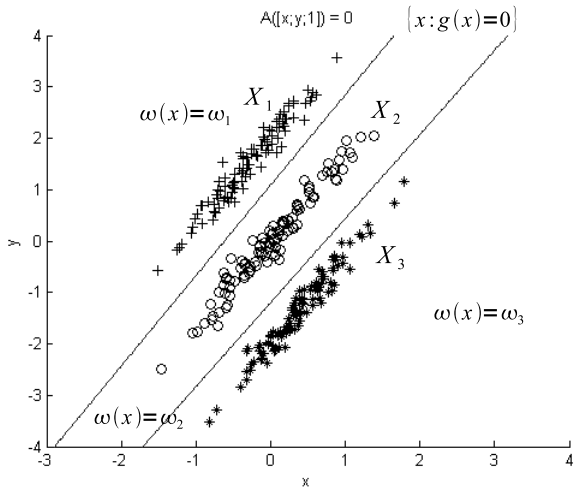


Figure 2.5: A classification found by the logistic regression for the same three-class problem as shown in Figure (2.4). The plot shows points $\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3$ of three-classes $\omega_1, \omega_2, \omega_3$. In opposite to the least-squares method, the decision boundary $\{\boldsymbol{x} : \boldsymbol{g}(\boldsymbol{x}) = \mathbf{0}\}$ obtained by the logistic regression approach perfectly classifies all examples.

2.3.3 Perceptron Algorithm

A further important procedure for constructing linear decision boundaries is that of finding a separating hyperplane. Separating hyperplanes provide the basis for Support Vector Machines, introduced in Section (2.4.3). Figure (2.6) shows a toy example of two classes ω_1, ω_2 of artificial feature vectors separated by differently learnt hyperplanes H_1, H_2 in $\mathbb{R}^2$. In this example, the least-squares solution H_1 produces errors, but the hyperplane H_2 found by an alternative approach, namely the *Perceptron Algorithm*, classifies all training points perfectly.

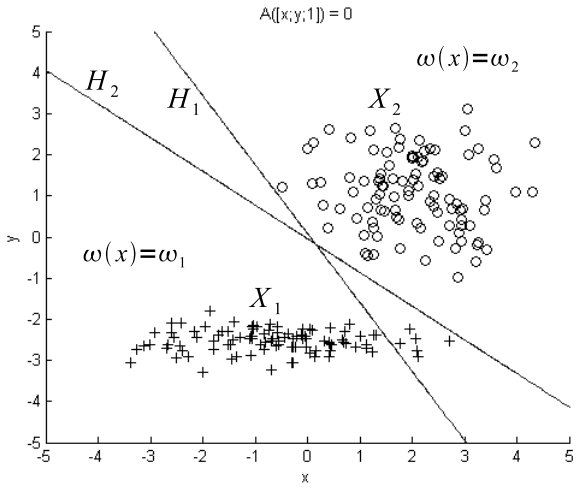


Figure 2.6: A toy example showing two differently learnt hyperplanes. Hyperplane H_1 is learnt using the least-squares method and hyperplane H_2 is found by the Perceptron Algorithm. While the solution H_1 is sub-optimal, the solution H_2 classifies all points perfectly.

The Perceptron Algorithm tries to find an separating hyperplane that minimizes the criterion function

$$J(\boldsymbol{w}, b) := -\sum_{i \in \mathcal{M}} y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) \tag{2.21}$$

where the set $\mathcal{M}$ comprises all indices of misclassified feature vectors.

Because a hyperplane is defined by $H := \left\{\boldsymbol{x} \in \mathcal{X} \ : \ \boldsymbol{w}^T\boldsymbol{x} + b = 0\right\}$ the vector $\boldsymbol{w}/\|\boldsymbol{w}\|$ is orthonormal to H (Fig. 2.7). Thus, the signed distance between any point $\boldsymbol{x}$ and its projection to the line intersecting $\boldsymbol{x}_0 \in H$ orthogonal to H is given by $\langle \boldsymbol{x} - \boldsymbol{x}_0, \boldsymbol{w}/\|\boldsymbol{w}\| \rangle$. Because it holds $\boldsymbol{w}^T\boldsymbol{x}_0 + b = 0$, it is easy to see that the signed distance of any point $\boldsymbol{x} \in \mathcal{X}$ to H is given by

$$\langle \boldsymbol{x} - \boldsymbol{x}_0, \boldsymbol{w}/\|\boldsymbol{w}\| \rangle = \frac{1}{\|\boldsymbol{w}\|}(\boldsymbol{w}^T\boldsymbol{x} + b). \tag{2.22}$$

The sign of $\boldsymbol{w}^T\boldsymbol{x} + b$ depends on the side of the hyperplane the vector $\boldsymbol{x}$ is lying. In particular, the hyperplane has the distance $|b|/\|\boldsymbol{w}\|$ from the origin.

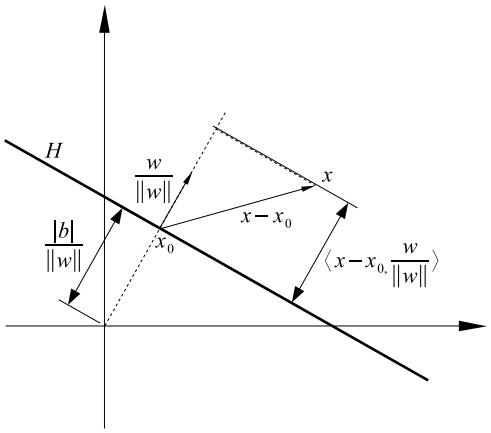


Figure 2.7: A linear hyperplane in $\mathbb{R}^2$.

In case of a binary classification, suppose the j-th sample's class $\omega(\boldsymbol{x}_j) = \omega_1$ is encoded by a label $y_j = 1$, or respectively $\omega(\boldsymbol{x}_j) = \omega_2$ by $y_j = -1$. Defining $\boldsymbol{w}^T\boldsymbol{x}_j + b > 0$ for all $\boldsymbol{x}_j$ of class ω_1, and $\boldsymbol{w}^T\boldsymbol{x}_j + b < 0$ for all $\boldsymbol{x}_j$ of class ω_2, the criterion function (2.21) is non-negative and proportional to the sum of distances of the misclassified points to the decision boundary. An analytic minimization of (2.21) is not possible in closed form. Therefore, a local solution $(\boldsymbol{w}^*, \boldsymbol{b}^*)$ is usually iterated via a gradient descent with step-size $\nu > 0$:

Algorithm 2.3.1 Perceptron Algorithm [Rosenblatt 1958]

Require: Training set $\emptyset \neq \mathcal{O}_N \subset \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, start values $(\boldsymbol{w}^{(0)}, b^{(0)})$, step-size $\nu > 0$, accuracy $\epsilon > 0$

Ensure: $\exists(\boldsymbol{x}_j, y_j), (\boldsymbol{x}_i, y_i) \in \mathcal{O}_N \ : \ y_j = -1 \wedge y_i = 1$, both classes are linearly separable

$t \leftarrow 0$

while $\left(\left\|\sum_{i\in\mathcal{M}^{(t)}} y_i\boldsymbol{x}_i\right\| > \epsilon\right) \bigwedge \left(\left\|\sum_{i\in\mathcal{M}^{(t)}} y_i\right\| > \epsilon\right)$ **do**

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \nu\nabla_{\boldsymbol{w}^{(t)}} J(\boldsymbol{w}^{(t)}, b^{(t)}) = \boldsymbol{w}^{(t)} + \nu \sum_{i\in\mathcal{M}^{(t)}} y_i\boldsymbol{x}_i \tag{2.23}$$

$$b^{(t+1)} = b^{(t)} - \nu\nabla_{b^{(t)}} J(\boldsymbol{w}^{(t)}, b^{(t)}) = b^{(t)} + \nu \sum_{i\in\mathcal{M}^{(t)}} y_i. \tag{2.24}$$

$t \leftarrow t + 1$

end while

It can be shown, that the gradient descent converges to a separating hyperplane, if the classes are linearly separable (e.g. [Duda 2000]). In the non-

separable case, ordinary Perceptron learning completely fails. However, modified Perceptron algorithms exist which still produce meaningful results. Fast algorithms reaching local optima are given by [Wendemuth 1995]. A Global solution can be achieved by the *Pocket algorithm* [Gallant 1990] albeit at the cost of an unknown and possibly extremely slow rate of convergence.

As Figure (2.6) let imagine, there are infinitely many separating hyperplanes perfectly classifying the feature vectors. To which of them the Perceptron Algorithm will end up depends on the starting point $(\boldsymbol{w}^{(0)}, b^{(0)})$. Another issue is concerning the generalization performance of the discriminant function. Although all training points are correctly classified, this must not be true for unseen data points. An elegant solution to avoid arbitrariness is to impose additional constraints to the separating hyperplane. Instead of requiring a correct classification, i.e. $y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) > 0 \; \forall i$, one could for example impose alternatively the constraints $y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq \rho > 0 \; \forall i$. The constant ρ is called *margin*, because it defines a minimum distance that all points have to maintain to the separating hyperplane in order to satisfy the constraints. Intuitively, for increasing ρ the hyperplane is more and more shifted to a stable position in the middle of the solution region. Such a stable hyperplane is more likely to classify new examples correctly. This is exactly the idea behind an optimal separating hyperplane implemented by the Support Vector Machine, as we will see in Section (2.4.3). There is also a Perceptron algorithm for achieving a stable hyperplane in the linearly separable case, called *Perceptron of Optimal Stability* [Krauth 1987]. However, this algorithm fails completely in the nonlinearly separable case. But it was generalized by [Wendemuth 1995], producing (globally) maximum margins for the linearly separable case, and (locally) maximum (negative) margins for the nonlinearly separable case without having any knowledge about separability or nonseparability in advance.

2.4 Nonlinear Classifiers

The linear classifiers reviewed so far, rely on some monotone transformation like the logarithm of the considered posterior probabilities $P(\omega_i|\boldsymbol{x})$ yielding a linear function in $\boldsymbol{x}$. Such a transformation into linear functionals is often not possible for complex data distributions of real-world problems. In the next sections, we review Artificial Neural Networks and in particular Support Vector Machines that both go beyond linear models.

2.4.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) realize generalizations of linear discriminant functions $g_i(\boldsymbol{x})$ defined in Section (2.2), namely nonlinear functions $g_i(\boldsymbol{x}) = f_i(\boldsymbol{w}_i^T \boldsymbol{\Phi}(\boldsymbol{x}) + b_i)$ with some non-linear mapping $\boldsymbol{\Phi} : \mathbb{R}^d \to \mathbb{R}^p$. In particular, ANNs implement continuously differentiable *output-layer* functions $f_1, \ldots, f_c : \mathbb{R} \to \mathbb{R}$ composed of *input-layer* functions $f_1^0, \ldots, f_d^0 : \mathbb{R} \to \mathbb{R}$ and *hidden-layer* functions $f_1^r, \ldots, f_{N_r}^r : \mathbb{R} \to \mathbb{R}$ with $1 \leq r \leq L$, $N_0 := d$, $N_L := p$, i.e. $\forall 1 \leq i \leq c$

$$g_i(\boldsymbol{x}) = f_i \left(\sum_{j=1}^{N_L} w_{ij} \Phi_j(\boldsymbol{x}) + b_i \right) \tag{2.25}$$

and $\forall 1 \leq j \leq N_L$

$$\Phi_j(\boldsymbol{x}) := f_j^L \left(\sum_{k=1}^{N_{L-1}} w_{jk}^L f_k^{L-1} \left(\ldots f_l^1 \left(\sum_{m=1}^{N_0} w_{lm}^1 f_m^0(x_m) + b_l^1 \right) \ldots \right) + b_j^L \right). \tag{2.26}$$

The discriminant function of ANNs has a biological analogy in the neural network of the brain (Fig. 2.8).

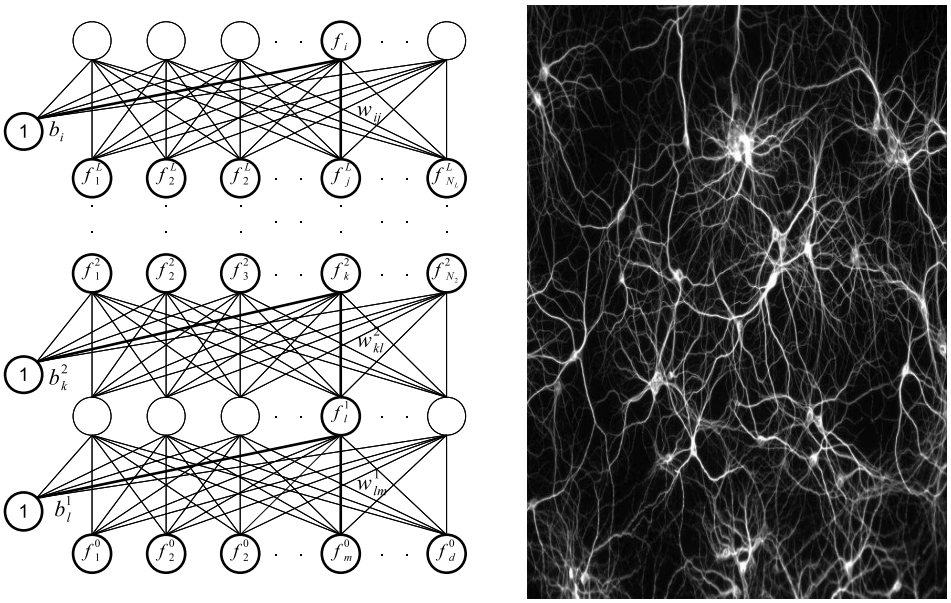


Figure 2.8: An artificial neural network (picture on the left) is a technical abstraction of the neural network of the brain (picture on the right [Koninck 2007], 2007 copyright by Paul De Koninck. All rights reserved.)

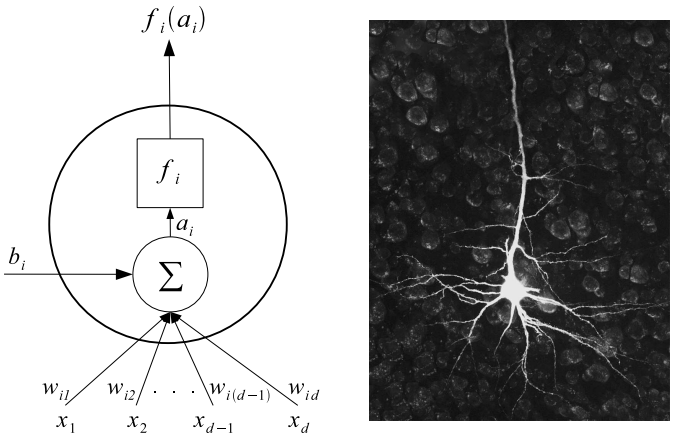
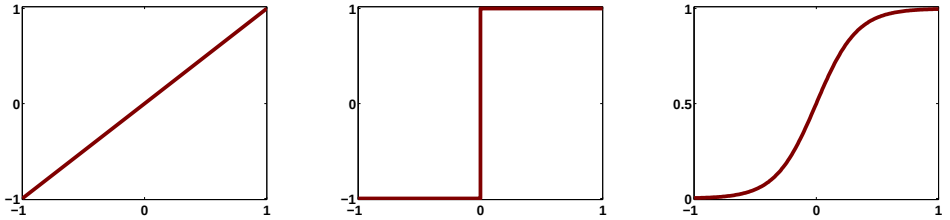


Figure 2.9: A single neuron. (picture on the right [Schrader 2010], 2010 copyright by Tulane University. All rights reserved.)

A neural network is build up of single neurons (Fig. 2.9). Each neuron i receives signals x_j from other neurons j with different strength w_{ij}, which are summed to yield the *activation* $a_i := \sum_j w_{ij}x_j + b_i$ of the neuron. The threshold b_i is some basis-potential of the neuron. The neuron reacts immediately with a response $f_i(a_i)$ modeled via the *activation function* f_i. Typical activation functions used in ANNs are the linear function $f_i(a_i) = a_i$, the threshold function $f_i(a_i) = \begin{cases} 1 & \text{if } a_i \geq 0 \\ 0 & \text{if } a_i < 0 \end{cases}$ or the sigmoid function $f_i(a_i) = 1/\left(1 + \exp(-a_i)\right)$ (Fig. 2.10).



(a) linear function (b) threshold function (c) sigmoid function

Figure 2.10: Typical activation functions used in Artificial Neural Networks.

ANNs can be used for regression and classification as well. For classification the output function f_i in (2.25) is usually chosen to be linear or the *softmax* function $\sigma(a_i) = \exp(a_i)/\sum_j \exp(a_j)$.

The weights and biases of an ANN are fitted to the training data $\mathcal{O}_N =$

$\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, $N \in \mathbb{N}$ by minimizing the sum-of-squared errors

$$J(\boldsymbol{\Theta}) := \sum_{n=1}^{N} \overbrace{\|\boldsymbol{y}_n - \boldsymbol{g}(\boldsymbol{x}_n)\|^2}^{=:J_n(\boldsymbol{\Theta})} \tag{2.27}$$

between the network's outputs $\boldsymbol{g}(\boldsymbol{x}_n) := (g_1(\boldsymbol{x}_n), \ldots, g_c(\boldsymbol{x}_n))^T$ and a target vector $\boldsymbol{y}_n \in \mathbb{R}^c$ associated to a training vector $\boldsymbol{x}_n \in \mathbb{R}^d$ via gradient descent. The complete set of weights and biases is denoted by $\boldsymbol{\Theta}$. For classification, the target vectors $\boldsymbol{y}_n$ are coded with the k-th component set to $y_{n,k} = 1$, if $\omega(\boldsymbol{x}_n)$ equals the class ω_k, while the remaining components are set to zero, i.e. $\boldsymbol{y}_n = (0, \ldots, y_{n,k}, \ldots, 0)^T$. However, the cross-entropy function (cf. 2.20) is often a more appropriate criterion in a classification setting than the sum-of-squared errors.

2.4.2 Back-propagation Algorithm

The gradient descent approach for neural networks is called *back-propagation* [Rumelhart 1986], because the compositional form of the model enables to compute the gradients by using the chain rule for differentiation in a forward and backward sweep over the network. Denoting $a_k^r(\boldsymbol{x}_n), 1 \leq r \leq L$ (respectively a_k) the activation of a neuron k in the r-th *hidden-layer* (respectively *output-layer*) given a training vector $\boldsymbol{x}_n$, the output-layer updates at the $(t+1)$-st iteration reads as

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} - \nu \frac{\partial J(\boldsymbol{\Theta}^{(t)})}{\partial w_{ij}^{(t)}} = w_{ij}^{(t)} - \nu \sum_{n=1}^{N} \delta_{i,n}^{(t)} \cdot f_j^L(a_j^{L,(t)}(\boldsymbol{x}_n)) \tag{2.28}$$

$$b_i^{(t+1)} = b_i^{(t)} - \nu \frac{\partial J(\boldsymbol{\Theta}^{(t)})}{\partial b_i^{(t)}} = b_i^{(t)} - \nu \sum_{n=1}^{N} \delta_{i,n}^{(t)} \tag{2.29}$$

with

$$\delta_{i,n}^{(t)} := \frac{\partial J_n(\boldsymbol{\Theta}^{(t)})}{\partial g_i} \cdot (f_i)'(a_i^{(t)}(\boldsymbol{x}_n)). \tag{2.30}$$

Similar, the input- and hidden-layer updates ($1 \leq r \leq L$) are given by

$$w_{jk}^{r,(t+1)} = w_{jk}^{r,(t)} - \nu \frac{\partial J(\boldsymbol{\Theta}^{(t)})}{\partial w_{jk}^{r,(t)}} \tag{2.31}$$

$$= w_{jk}^{r,(t)} - \nu \sum_{n=1}^{N} \delta_{j,n}^{r,(t)} \cdot f_k^{r-1}(a_k^{r-1,(t)}(\boldsymbol{x}_n)) \tag{2.32}$$

$$b_j^{r,(t+1)} = b_j^{r,(t)} - \nu \frac{\partial J(\boldsymbol{\Theta}^{(t)})}{\partial b_j^{r,(t)}} = b_j^{r,(t)} - \nu \sum_{n=1}^{N} \delta_{j,n}^{r,(t)}. \tag{2.33}$$

with

$$\delta_{j,n}^{r,(t)} := (f_j^r)'(a_j^{r,(t)}(\boldsymbol{x}_n)) \cdot \sum_{i=1}^{N_{r+1}} w_{ij}^{r+1,(t)} \delta_{i,n}^{r+1,(t)}, \tag{2.34}$$

$$\text{and} \qquad N_{L+1} := c, w_{ij}^{L+1,(t)} := w_{ij}^{(t)}, \delta_{i,n}^{L+1,(t)} := \delta_{i,n}^{(t)}, a_k^{0,(t)}(\boldsymbol{x}_n) := \boldsymbol{x}_n.$$

We see from the update equations that first all activations and neuron outputs of the network have to be computed (forward sweep). Second, the activations and neuron outputs are used to compute error-terms δ from each layer back to the predecessor layer in order to adjust the network parameters (backward sweep). Hence, the backward sweep is sometimes called backward error propagation. The back-propagation algorithm is summarized in (Alg. 2.4.1).

Algorithm 2.4.1 Back-propagation Algorithm [Rumelhart 1986]

Require: Training sample $\emptyset \neq \mathcal{O}_N \subset \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, start value $\boldsymbol{\Theta}^{(0)}$, step-size $\nu > 0$, accuracy $\epsilon > 0$

Ensure: $\exists(\boldsymbol{x}_j, y_j), (\boldsymbol{x}_i, y_i) \in \mathcal{O}_N \ : \ y_j = -1 \wedge y_i = 1$

$t \leftarrow 0$

while $\left\|\nabla J(\boldsymbol{\Theta}^{(t)})\right\| > \epsilon$ **do**

for all $1 \leq n \leq N$ **do**

{Forward Sweep:}

Compute $\forall 1 \leq r \leq L, 1 \leq k \leq N_r$ the activations $a_k^{r,(t)}(\boldsymbol{x}_n)$.

Compute $\forall 1 \leq i \leq c$ the activations $a_i^{(t)}(\boldsymbol{x}_n)$.

{Backward Sweep:}

Compute $\forall 1 \leq i \leq c$ the errors $\delta_{i,n}^{(t)}$.

Compute $\forall L \geq r \geq 1, 1 \leq j \leq N_r$ the errors $\delta_{j,n}^{r,(t)}$.

{Updates:}

Compute $\forall 1 \leq i \leq c, \forall 1 \leq j \leq N_L$ the weights $w_{ij}^{(t+1)} = w_{ij}^{(t)} - \nu \frac{\partial J(\boldsymbol{\Theta}^{(t)})}{\partial w_{ij}^{(t)}}$ and biases $b_i^{(t+1)} = b_i^{(t)} - \nu \frac{\partial J(\boldsymbol{\Theta}^{(t)})}{\partial b_i^{(t)}}$.

Compute $\forall 1 \leq r \leq L, 1 \leq j \leq N_r, 1 \leq k \leq N_{r-1}$ the weights $w_{jk}^{r,(t+1)} = w_{jk}^{r,(t)} - \nu \frac{\partial J(\boldsymbol{\Theta}^{(t)})}{\partial w_{jk}^{r,(t)}}$ and biases $b_j^{r,(t+1)} = b_j^{r,(t)} - \nu \frac{\partial J(\boldsymbol{\Theta}^{(t)})}{\partial b_j{}^{r,(t)}}$.

end for

$t \leftarrow t + 1$

end while

At first glance, ANNs seem to be very attractive because they potentially learn any complicated function (if the number of layers exceeds three and the number of neurons are sufficiently large) and the back-propagation algorithm is easy to implement. But training of ANNs is a challenge, because

the optimization of the network parameters is highly nonlinear and unstable. Moreover, large ANNs are overparametrized and thus tend to overfitting. A feasible solution depends mainly on the topology of the network and the starting values. Usually, the topology is often selected by experiments, and the starting values are randomly initialized leading to poor local solutions in particular for networks with many hidden-layers and thousands of free parameters.

2.4.3 Support Vector Machines

Although *Support Vector Machines* (SVMs) [Vapnik 1998], [Vapnik 1999] are in essence linear classifiers as presented in Section (2.2), they implement nonlinear discriminant functions. In contrast to linear classifiers, the SVM's hyperplane is learnt in a space $\mathcal{H}$ (*feature space*) of high dimension in which the data has been implicitly mapped. In this way, highly overlapping samples of two categories, which can not be separated through a linear decision boundary in the lower dimensional data space $\mathcal{X}$, are more likely to be linear classifiable in the feature space $\mathcal{H}$. Fortunately, the nonlinear feature map $\mathbf{\Phi} : \mathcal{X} \to \mathcal{H}$ must not be carried out explicitly. A nonlinear decision function in $\mathcal{X}$ results from the implicit inverse transformation $\mathbf{\Phi}^{-1}$ of the hyperplane (Fig. 2.11).

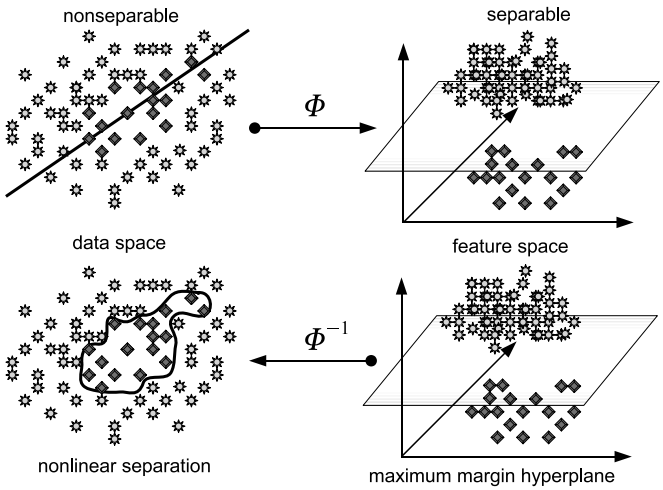


Figure 2.11: Nonlinear decision boundary obtained by a feature map $\mathbf{\Phi}$.

The training algorithm for the SVM's discriminant function

$$g(\boldsymbol{x}) := \boldsymbol{w}^T \mathbf{\Phi}(\boldsymbol{x}) + b \tag{2.35}$$

is based on the Perceptron Algorithm with margin $\rho > 0$ for the linearly separable case (Sec. 2.3.3). In particular, the SVM tries to find an optimal

separating hyperplane that classifies all training patterns correctly with a maximum margin (Fig. 2.12).

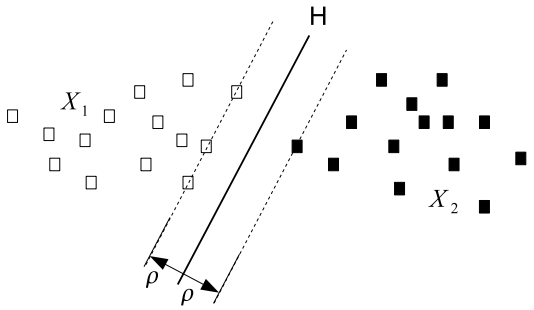


Figure 2.12: Optimal separating hyperplane.

A *maximum margin hyperplane* is justified to have better statistical properties than any other possible linear separation (Chapter 3). The SVM can also be thought of as a particularly trained neural network (Sec. 2.4.1) with one hidden layer representing the transform $\mathbf{\Phi}$ and a linear input- and output-layer (Fig. 2.13). In contrast to neural networks, the data transformation is implicitly performed by SVMs.

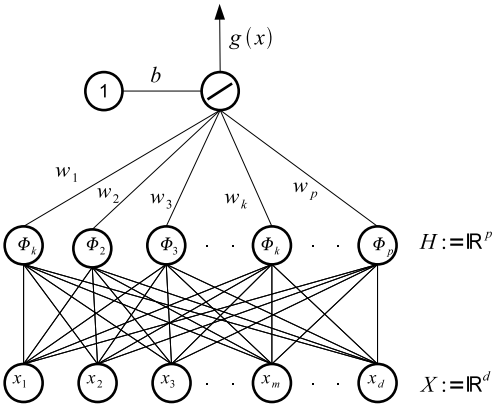


Figure 2.13: SVM viewed as feed-forward ANN.

Because of its particular importance for this thesis, in Chapter (3) much more details on Support Vector Machines are revisited. Therefore, in the next section we go on with a few application where SVMs have been successfully applied.

2.4.4 Some Applications of SVMs

Due to the good generalization properties and the efficient computation, SVMs have been successfully applied to many binary- as well as multi-category clas-

sification problems appearing in different application contexts. Even on tasks where the linear methods or ANNs are worse or inapplicable.

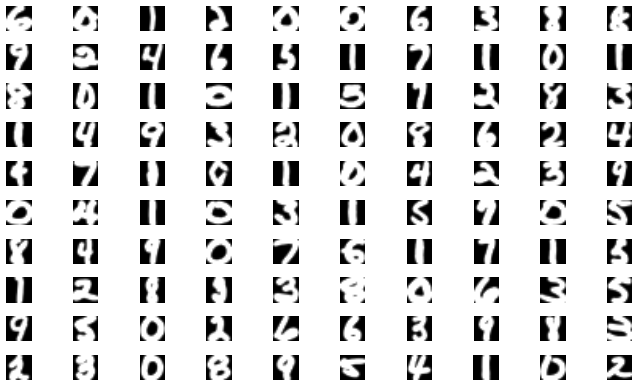


Figure 2.14: Visualization of one hundred examples from the U.S. Postal Service database [Wang 1988].

The most prominent application of SVMs is surely the task of handwritten digit recognition [Vapnik 1999]. First experiments were performed using the *U.S. Postal Service* (USPS) database [Wang 1988]. It contains 7,300 training patterns and 2,000 test patterns of handwritten digits extracted from real-life zip codes. Each pattern has a resolution of 16 x 16 pixels (Fig. 2.14). The human raw error rate for this data set is about 2.5% [Bromley 1991]. The best reported result before SVMs showed up was 5.0% reached with a five layer ANN using task specific adapted recpetive fields [LeCun 1989]. In comparision, the SVM's misclassification rate of the test patterns is about 4.0% [Vapnik 1999], which is one of the best reported results[1].

Another classical handwritten digit task is given by the NIST database, which is a benchmark database provided by the *U.S. National Institute of Standards and Technology* (NIST). This database consists of 60,000 training patterns and 10,000 test patterns. The characters have a resolution of 20 x 20 pixels. For the NIST data the best reported result is 0.7% error using a very special convolution network that is highly adapted to the handwirtten digit task [LeCun 1998]. Not far away from the best, a standard SVM reached a comparable performance of 1.1% test error [Vapnik 1999].

A popular benchmark for text categorization is the Reuters-22173 text corpus. Usually, text categorization is a high-dimensional classification problem with many classes. For example the Reuters corpus was collected from 21,450

[1] 3.3% error using a local learning approch, 2.7% error using tangent distance matching

					SVM (poly) degree d =					SVM (rbf) width γ =			
	Bayes	Rocchio	C4.5	k-NN	1	2	3	4	5	0.6	0.8	1.0	1.2
earn	95.9	96.1	96.1	97.3	98.2	98.4	**98.5**	98.4	98.3	**98.5**	98.5	98.4	98.3
acq	91.5	92.1	85.3	92.0	92.6	94.6	**95.2**	95.2	95.3	95.0	95.3	95.3	**95.4**
money-fx	62.9	67.6	69.4	78.2	66.9	72.5	75.4	74.9	**76.2**	74.0	75.4	**76.3**	75.9
grain	72.5	79.5	89.1	82.2	91.3	93.1	**92.4**	91.3	89.9	**93.1**	91.9	91.9	90.6
crude	81.0	81.5	75.5	85.7	86.0	87.3	88.6	**88.9**	87.8	**88.9**	89.0	88.9	88.2
trade	50.0	77.4	59.2	77.4	69.2	75.5	76.6	77.3	**77.1**	76.9	78.0	**77.8**	76.8
interest	58.0	72.5	49.1	74.0	69.8	63.3	67.9	73.1	**76.2**	74.4	75.0	**76.2**	76.1
ship	78.7	83.1	80.9	79.2	82.0	85.4	86.0	**86.5**	86.0	**85.4**	86.5	87.6	87.1
wheat	60.6	79.4	85.5	76.6	83.1	84.5	85.2	**85.9**	83.8	**85.2**	85.9	85.9	85.9
corn	47.3	62.2	87.7	77.9	86.0	86.5	85.3	**85.7**	83.9	**85.1**	85.7	85.7	84.5
microavg.	**72.0**	**79.9**	**79.4**	**82.3**	84.2	85.1	85.9	86.2	85.9	86.4	86.5	86.3	86.2
					combined: **86.0**					combined: **86.4**			

Figure 2.15: SVM accuracies on ten most frequent Reuters categories. (*Courtesy of [Joachims 1998].*)

news stories from 1997, and is partitioned and indexed into 135 different categories. The dimensionality of the raw data space is 10^4 containing the word frequencies within a document. Even for document classification improved results have been reported using SVMs in comparison to other approaches (Fig. 2.15, [Joachims 1998]).

A very interesting application of Support Vector Machines is from the domain of biometrics, namely the recognition of faces. Face recognition is a field of active research and recent results combining SVMs with different linear feature extraction methods have shown the potential of SVMs to improve the overall performance [Mazanec 2008]. The experiments were performed on the *Facial Recognition Technology* (FERET) database. This benchmark corpus consists of 14,051 eight-bit grayscale images of human heads with views ranging from frontal to left and right profiles (Fig. 2.16).

The experiments reported in [Mazanec 2008] indicate that very good results are possible combining *Linear Discriminant Analysis* (LDA) and Support Vector Machines (Fig. 2.17).

Although SVMs are static classifiers by nature, a interesting attempt is to combine SVMs with *Hidden-Markov-Models* (HMM) for the classification of time-varying data appearing in particular in *Automatic Speech Recognition* (ASR), speaker verification or speaker identification.

Substituting statistical models by SVMs for modeling the acoustic of speech and using HMMs for capturing the time dependencies is a straightforward way to benefit from the generalization performance of SVMs also in speech recognition. Because SVMs are static with respect to the length of feature vectors as well as dependencies in time, Hidden-Markov-Models are necessary to provide the temporal information. For example using the time alignment from a HMM-Viterbi decoder enables to pre-process the feature stream from spoken

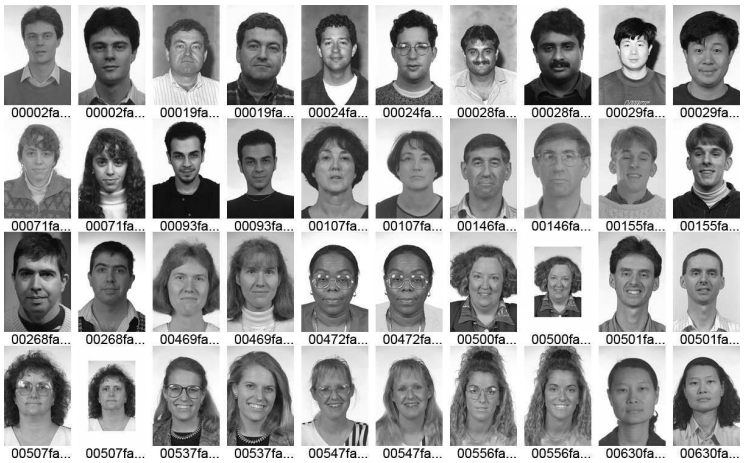


Figure 2.16: Examples from the FERET training-set. (*Courtesy of [Mazanec 2008].*)

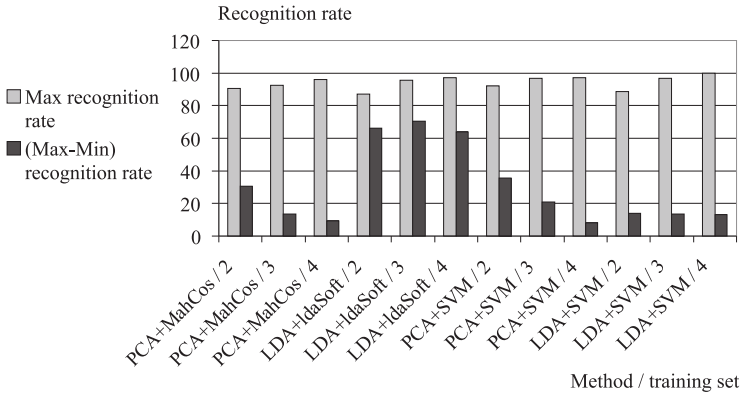


Figure 2.17: Face recognition results on the FERET database using different combinations of SVMs and feature extractors. (*Courtesy of [Mazanec 2008].*)

utterances (Fig. 2.18) to feature vectors of constant length for an adjacent SVM classification [Ganapathiraju 2000].

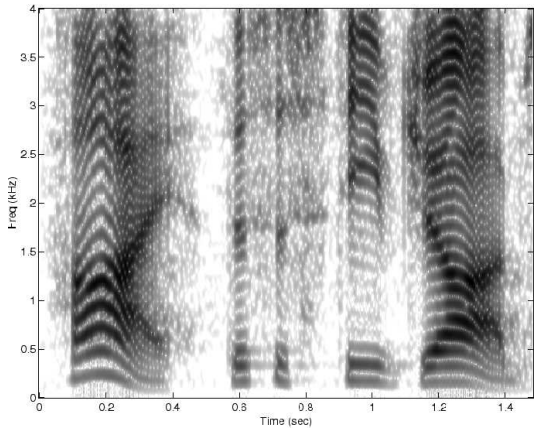


Figure 2.18: Visualization of a feature stream of some spoken utterance.

In [Ganapathiraju 2000] a first hybrid system was proposed and tested on the *Oregon Graduate Institute* (OGI) Alphadigit corpus which is a telephone speech database similar to the SWITCHBOARD corpus. The OGI task consists of spoken six word strings out of a vocabulary of 36 words. Using this dataset, the SVM/HMM approach reached an improvement of 10% relative to the HMM baseline system.

Motivated by the former result, it has been extensively studied how SVM can be applied in ASR and what value of improvement can be expected due to a substitution of the Gaussian mixture models usually used in HMMs to model the acoustic of speech, e.g. [Campbell 2003], [Gurban 2005], [Liu 2007].

A offline-framework for using SVMs and HMMs side by side was proposed in [Stuhlsatz 2008a], [Stuhlsatz 2007a] and is based on the work of [Ganapathiraju 2000]. The framework uses HMMs for the time-alignment and SVMs for an offline re-scoring of N-best lists as well as phoneme lattices produced by the HMM-Viterbi decoder. Experiments using N-best list re-scoring were performed on a phoneme recognition database, namely the *DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus* with N-best list re-scoring [Stuhlsatz 2003]. The TIMIT corpus was spoken by 630 speaker from eight regions of the United States. The training-set consists of about 150,000 training examples while the test-set is composed of about 7,000 test patterns. A total of 50 different SVMs (one for each phoneme in the vocabulary) were evaluated for this recognition task. After training and transforming the SVM classifier's outputs to probabilities, different N-best lists were re-scored first and then all hypotheses were re-ranked. The best hypothesis selected from the new list was then used as prediction (Fig. 2.19). This ap-

proach could reduce the phoneme misclassification rate about 2.64% compared to the HMM baseline [Stuhlsatz 2003].

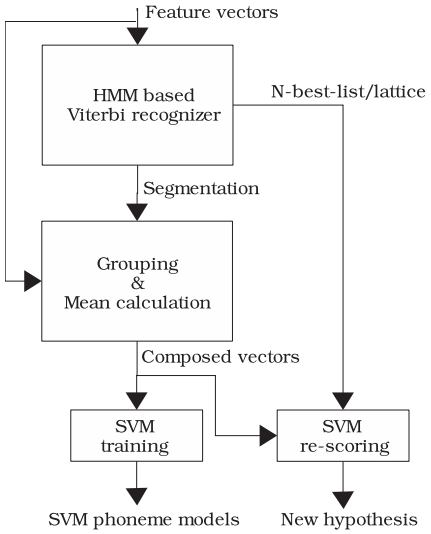


Figure 2.19: Sketch of the hybrid HMM/SVM architecture.

A further improvement was obtained due to a SVM re-scoring of recognition lattices produced by the decoder instead of N-best list re-scoring. Using this method, a relative error decrease of 7.72% was reported on the TIMIT database, and on the *Wallstreet Journal Cambridge* (WSJCAM0) corpus, which constists of about 170,000 phonemes for training and about 38,000 for testing, an error decrease of 12.8% was obtained [Stuhlsatz 2006]. The performance of a HMM/SVM hybrid architecture has also been investigated on the *DARPA RM1* database. For these experiments refined estimates of emission probabilities from the SVM outputs and a more online integration of SVMs into the recognition process were proposed [Krüger 2005a], [Krüger 2005b], [Andelic 2006], [Krüger 2006], [Krüger 2007].

Summary

In this chapter, a general overview of pattern classification and methods are given. Without doubt, the SVM is a *state-of-the-art* classifier in pattern recognition for solving many different classification problems with good performance and manageable computational costs. In particular, the SVM has shown very good generalization properties. Because the list of successful applications of SVMs seems almost endless, we refer the reader to the vast number of available articles for further reading (e.g. [Osuna 1997], [Brown 1999], [Lee 2000], [Kim 2002], [Ghent 2005], [Iplikci 2006] and references therein).

However, it is important to note that classification is extremely data and therefore application dependent. Moreover, there is an inherent lack of superiority of any classifier in advance. That means, one can not decide to prefer a classifier without any prior knowledge of the nature of the classification problem. Thus, some problems might be solved very well (sometimes better than SVMs) using for example Artificial Neural Networks or other (nonlinear) classification methods.

In the following chapter, more theoretical background is presented for the purpose to understand why SVMs perform better on many tasks than other pattern recognition methods. It turns out that the maximum margin concept is the key to the generalization ability of SVMs and thus the maximum margin will be crucial throughout this thesis.

Chapter 3

Statistical Learning Theory and SVMs

Contents

Statistical learning theory [Vapnik 1998], [Vapnik 1999], [Vapnik 2006], the fundamental theory of machine learning and pattern classification, deals mainly with the question whether a learning method, that utilizes the *Empirical Risk Minimization* (ERM) principle, converges to the best possible risk and how fast convergence takes place.

Although the theory is very general covering regression, density estimation and classification problems the present chapter will concentrate on classification only, because it is the focus of this thesis.

The results obtained from statistical learning theory justify the development of learning algorithms like the SVM that implement decision functions

maximizing a minimal distance to the given data (recall Chapter 2). Moreover, we will see that a maximum margin hyperplane is crucial for a good generalization performance.

We start our discussion with an illustrative example (Sec. 3.1). The example leads immediately to the question, which constraints are necessary and sufficient for successful learning from data (Sec. 3.2). For this purpose some measure of the capacity of the functions used for classification is needed (Sec. 3.3.3). Given this measure a new learning paradigm is introduced (Sec. 3.4) and searching for a maximum margin hyperplane approximates this principle (Sec. 3.5). At the end (Sec. 3.6), we revisit the SVM again but with more technical details than in Section (2.4.3).

3.1 Illustration Example

Consider the following simple pattern recognition scenario: We are faced with the problem of classifying a data point $\boldsymbol{x} \in \mathcal{X}$ of some data space $\mathcal{X}$ into a category $y \in \mathcal{Y}$. The category space $\mathcal{Y}$ considered here consists of two possible realizations $\{-1, 1\} =: \mathcal{Y}$ (binary classification) for the two possible outcomes that $\boldsymbol{x}$ belongs to one of two classes ω_1, ω_2.

Neither $\boldsymbol{x}$ nor y are known in advance, however we can treat them as random variables with joint density $p : \mathcal{X} \times \mathcal{Y} \to [0, \infty)$, $\int p(\boldsymbol{x}, y)\, d\boldsymbol{x}\, dy = 1$. Unfortunately, we do not know the true density p, but we assume to have available some set of observations $\mathcal{O}_l := \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_l, y_l)\} \subset (\mathcal{X} \times \mathcal{Y})$, $l \in \mathbb{N}$, where the samples $(\boldsymbol{x}_i, y_i)$ are generated from p. In Figure (3.1) an example of nine points is shown.

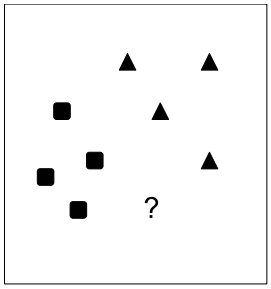


Figure 3.1: An example of two categories (triangles and squares). One data point has unknown class-membership (marked with ?).

The objective is to develop an algorithm that solves the classification problem for the given data points such that the misclassification will be as small as possible. For this purpose, suppose a look-up table is stored (Tab. 3.1), which enables (for a fixed l) to assign all possible classifications to a set of

rule	$(y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y)$
$\boldsymbol{f}^1$	$(+1,+1,+1,+1,+1,+1,+1,+1,+1)$
$\boldsymbol{f}^2$	$(+1,+1,+1,+1,+1,+1,+1,+1,-1)$
$\boldsymbol{f}^3$	$(+1,+1,+1,+1,+1,+1,+1,-1,+1)$
$\vdots$	$\vdots$
$\boldsymbol{f}^{emp,1}$	$(+1,+1,+1,+1,-1,-1,-1,-1,+1)$
$\boldsymbol{f}^{emp,2}$	$(+1,+1,+1,+1,-1,-1,-1,-1,-1)$
$\vdots$	$\vdots$
$\boldsymbol{f}^{2^9}$	$(-1,-1,-1,-1,-1,-1,-1,-1,-1)$

Table 3.1: Example of a look-up table of all decision rules assigning one out of 2^9 possible classifications to a set of nine points.

points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_l\} \cup \{\boldsymbol{x}\}$. Further, a *loss function* $L_f : (\mathcal{X} \times \mathcal{Y}) \to \mathbb{R}$ specifies how much an erroneous classification will hurt us when applying a decision rule $\boldsymbol{f}^i := (y_1, \ldots, y_l, y)$ with $y_j := f^i(\boldsymbol{x}_j)$ and $y := f^i(\boldsymbol{x})$ be the outputs of a decision function $f^i : \mathcal{X} \to \mathcal{Y}$. Hence, each decision rule $\boldsymbol{f}^i$ comprises a possible class assignment to a set of points. Obviously, for $l+1$ different elements one has to implement a total of $N_{\mathcal{F}} = 2^{(l+1)}$ different rules.

In order to obtain the lowest loss on average, the algorithm should select automatically a rule $\boldsymbol{f}^*$ from the look-up table such that the expected risk

$$R(f) := \int L_f(\boldsymbol{x}, y) p(\boldsymbol{x}, y) \; d\boldsymbol{x} \; dy \tag{3.1}$$

will be minimized, i.e. $R(f^*) = \min\{R(f^i) : f^i\}$.

Because $R(f)$ can not be determined due to a lack of p, one could try to select the best rule $\boldsymbol{f}^*$ based on an empirical estimate

$$R_l(f) := \frac{1}{l} \sum_{i=1}^{l} L_f(\boldsymbol{x}_i, y_i) \tag{3.2}$$

computed by the use of the observation $\mathcal{O}_l$.

As shown in table (3.1), there are two rules $\boldsymbol{f}^{emp,1}$ and $\boldsymbol{f}^{emp,2}$ perfectly classifying the observation $\mathcal{O}_l$, i.e. $R_l(f^{emp,1}) = R_l(f^{emp,2}) = 0$. Hence, minimizing the empirical risk (3.2) gives no useful information about the unknown category. So, how to choose $\boldsymbol{f}^* \in \{\boldsymbol{f}^{emp,1}, \boldsymbol{f}^{emp,1}\}$?

Consider an appropriate choice of loss L_f which is in case of a binary

classification problem the 0/1-loss function

$$L_f(\boldsymbol{x}, y) = L_f^{0|1}(\boldsymbol{x}, y) := \begin{cases} 0 & if \;\; y = f(\boldsymbol{x}) \\ 1 & if \;\; y \neq f(\boldsymbol{x}) \end{cases} . \tag{3.3}$$

Assuming a sample $(\boldsymbol{x}, y)$ is independently and identically-distributed (i.i.d.) according to a distribution p, it follows that L_f itself is an i.i.d. random variable. Therefore, when using the 0/1-loss function (3.3), the risk (3.1) equals the probability of an incorrect classification

$$\begin{aligned} P_{err}(f^*) &= \int p(\boldsymbol{x}) \underbrace{\left(\sum_{y \in \{-1,+1\}} L_{f^*}^{0|1}(\boldsymbol{x}, y) \; P(y|\boldsymbol{x}) \right)}_{=: P_{err}(f^*|\boldsymbol{x})} d\boldsymbol{x} \\ &= \int p(\boldsymbol{x}) P_{err}(f^*|\boldsymbol{x}) \; d\boldsymbol{x} \end{aligned} \tag{3.4}$$

and the empirical risk (3.2) equals the relative frequency $\hat{P}_{err}(f)$ of assigning an observed sample to a wrong class. The conditional $P_{err}(f^*|\boldsymbol{x})$ is called the *Bayes-error*.

From (3.4) we see, that in order to minimize the overall misclassification, f^* has to be selected such that $P_{err}(f^*|\boldsymbol{x})$ is minimal, i.e. the algorithm should select the rule $\boldsymbol{f}^* = \boldsymbol{f}^{emp,1}$ if $P(y = +1|\boldsymbol{x}) \geq P(y = -1|\boldsymbol{x})$, and $\boldsymbol{f}^* = \boldsymbol{f}^{emp,2}$ otherwise (*Bayes decision rule*). Clearly, to apply the Bayes decision rule in practice, one has to estimate the class posteriors $P(y = +1|\boldsymbol{x})$ and $P(y = -1|\boldsymbol{x})$ from the given observations shifting the function estimation problem to probability estimation (cf. Sec. 2.3.1).

3.2 Uniform Convergence

Despite the simplicity of the former example, we can get some important insights from it:

1. Although we are able to minimize the empirical error R_l down to zero, it does not imply that the true risk R will also be zero, because it depends on the Bayes-error.
2. It may happen, even though the empirical risk is minimized at f^* using l observations, e.g. $R_l(f^*) = 0$, that the expected loss $R(f^*)$ is not at the lowest possible (*Bayes-risk*). For example imagine the case, when our program selects always by chance out of $\left\{ \boldsymbol{f}^{emp,1}, \boldsymbol{f}^{emp,2} \right\}$ even when the true but unknown probabilities $P(y = +1|\boldsymbol{x})$ and $P(y = -1|\boldsymbol{x})$ are not equal.

3. Empirical risk minimization is *ill-posed*, because usually there are many (equivalent) minimizer of R_l from which one has to choose somehow. The expected risk R depends strongly on the chosen minimizer.

At this point, the question of *consistency* of an empirical learning method arises. In a formal sense that means for a set of decision functions $\mathcal{F}$ and given a set of i.i.d. observations $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_l\}$, $\boldsymbol{z}_i := (\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, it holds

$$\forall \epsilon > 0 : \lim_{l \to \infty} P(|R(f^l) - \inf_{f \in \mathcal{F}} R(f)| > \epsilon) = 0 \tag{3.5}$$

$$\forall \epsilon > 0 : \lim_{l \to \infty} P(|R_l(f^l) - \inf_{f \in \mathcal{F}} R(f)| > \epsilon) = 0, \tag{3.6}$$

where f^l denotes the minimizer of the empirical risk R_l (3.2) using an observation of size l.

Theorem 3.2.1 (The Key Theorem of Learning Theory, [Vapnik 1998])**.** *Let L_f, $f \in \mathcal{F}$, be a set of loss functions that satisfies*

$$A \leq R(f) \leq B. \tag{3.7}$$

Then for the ERM method to be consistent, it is necessary and sufficient that the empirical risk R_l converges uniformly to the actual risk R in the following sense:

$$\lim_{l \to \infty} P\left\{\sup_{f \in \mathcal{F}} (R(f) - R_l(f)) > \epsilon\right\} = 0 \ \forall \epsilon > 0 \tag{3.8}$$

Proof. See [Vapnik 1998][pp. 89]. ∎

Thus, *one-sided convergence* is equivalent to consistency of the ERM principle. Note, the loss function L_f has to be taken into account when applying this theorem.

3.3 Rate of Convergence for the 0/1-Loss Function

3.3.1 Finite Case

For the simple case of a finite set of 0/1-loss functions $\{L_f^{0|1} : f \in \mathcal{F}\}$ of size $N_{\mathcal{F}} < \infty$, we now derive when uniform convergence takes place and how fast the ERM method converges.

Define the event $\mathcal{A}_i$ associated to a function $L_{f_i}^{0|1}$ to be all pairs $\boldsymbol{z} := (\boldsymbol{x}, y) \in \mathcal{X} \times \mathcal{Y}$ with $L_{f_i}^{0|1}(\boldsymbol{x}, y) = 1$, i.e. $\mathcal{A}_i := \{\boldsymbol{z} \in \mathcal{X} \times \mathcal{Y} : L_{f_i}^{0|1}(\boldsymbol{z}) = 1\}$.

As already mentioned, the risks R_l and R can be considered as the frequency $R_l(f) = \hat{P}_{err}(f)$ of assigning an observed sample to a wrong class and

the probability of an incorrect classification $R(f) = P_{err}(f)$. By the *law of large numbers*, for each event the frequency converges to the true probability as the number of trials increases indefinitely.

Thus, applying *Hoeffding's inequality* (cf. App. A.1.1) yields for each event $\mathcal{A}_i$ an exponential rate of convergence for $l \to \infty$ and any $\epsilon > 0$:

$$P\left(|P_{err}(f_i) - \hat{P}_{err}(f_i)| > \epsilon\right) \leq 2\exp\left(-2\epsilon^2 l\right). \tag{3.9}$$

It follows

$$\begin{aligned} P\left(\max_{1\leq i\leq N_{\mathcal{F}}}\left(|P_{err}(f_i) - \hat{P}_{err}(f_i)|\right) > \epsilon\right) &\leq \sum_{i=1}^{N_{\mathcal{F}}} P\left(|P_{err}(f_i) - \hat{P}_{err}(f_i)| > \epsilon\right) \\ &\leq 2N_{\mathcal{F}} \cdot \exp\left(-2\epsilon^2 l\right) \\ &= \underbrace{2 \cdot \exp\left(\left(\frac{\ln(N_{\mathcal{F}})}{l} - 2\epsilon^2\right) \cdot l\right)}_{=:\nu} \end{aligned} \tag{3.10}$$

that in turn implies (two-sided) uniform convergence takes place for a finite function class if $\lim_{l\to\infty} \ln(N_{\mathcal{F}})/l = 0$:

$$\lim_{l\to\infty} P\left(\max_{1\leq i\leq N_{\mathcal{F}}}\left(|P_{err}(f_i) - \hat{P}_{err}(f_i)|\right) > \epsilon\right) = 0. \tag{3.11}$$

By definition of $\nu > 0$ (Eq. 3.10), one obtains the relation

$$\epsilon = \sqrt{\frac{\ln(N_{\mathcal{F}}) - \ln(\nu/2)}{2l}}. \tag{3.12}$$

Therefore, it holds at least with probability $1 - \nu$ for all $f_i \in \mathcal{F}$, $1 \leq i \leq N_{\mathcal{F}}$ (including the minimizer f^l of the emprirical risk):

$$R_l(f_i) - \epsilon \leq R(f_i) \leq R_l(f_i) + \epsilon. \tag{3.13}$$

Remember the example in Section (3.1) of selecting a rule $\boldsymbol{f}^*$ with $R_l(f^*) = 0$ from a look-up table out of $N_{\mathcal{F}} = 2^9$ rules using a observation of size $l = 8$. Applying the upper bound (Eq. 3.13), the true risk of misclassification is not greater than 85% almost sure ($\nu = 0.01$). This bound is not very tight, because in our example selecting a optimal rule by chance yields a probability of missclassification of 50%. A tighter bound is obtainable for a set of 0/1-loss functions when taking into account, that multiple equivalent minimizer of R_l solving the problem perfectly are in the set $\mathcal{F}$ (cf. [Vapnik 2006][pp. 144]). In general, the upper bound can further refined considering one-sided uniform convergence instead of two-sided uniform convergence (cf. [Vapnik 2006][pp. 146]).

3.3.2 Infinite Case

From the rate of convergence for a finite number $N_{\mathcal{F}}$ of functions (Eq. 3.10), it follows that uniform convergence takes place, if for any $\epsilon > 0$ holds

$$\frac{ln(N_{\mathcal{F}})}{l} \underset{l\to\infty}{\longrightarrow} 0. \tag{3.14}$$

The basic idea in case of an infinite set $\mathcal{F}$ is to substitute $N_{\mathcal{F}}$ with a similar quantity describing the *capacity* of $\mathcal{F}$ and then to prove uniform convergence using this quantity. The capacity assesses a function class, e.g. the space of all linear decision functions, with respect to its flexibility in classifying a finite set of samples. This measure of flexibility is not to be confused with the number of free parameters of a function, e.g. the number of coefficients of a polynomial. One can imagine that an infinite function class may contain many functions that classify a set of samples in the same way. Thus, a meaningful concept of capacity should not depend on equivalent functions. We will see the appropriate concept of capacity introduced by [Vapnik 1998] results in a condition like (Eq. 3.14).

For this purpose, a more refined measure of capacity is given using the effective number of functions in the class $\mathcal{F}$ with respect to the events $\mathcal{A}_f := \{\boldsymbol{z} \in \mathcal{X} \times \mathcal{Y} : L_f^{0|1}(\boldsymbol{z}) = 1\}$ given a random sample $\mathcal{O}_l := \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_l\}$.

Consider a binary vector $\boldsymbol{g}(f) := \left(L_f^{0|1}(\boldsymbol{z}_1), \ldots, L_f^{0|1}(\boldsymbol{z}_l)\right) \in \{0,1\}^l$. Each vector is a subset of vertices of a hypercube (Fig. 3.2) and represents the different subsets of O_l that are also subsets of the events $\mathcal{A}_f, f \in \mathcal{F}$. The point $(0, \ldots, 0)$ represents the case that no subset is contained in one of the events $\mathcal{A}_f$.

In other words, each vector $\boldsymbol{g}(f)$ is a symbolization of an equivalence class

$$[f] := \left\{\tilde{f} \in \mathcal{F} : \tilde{f} \equiv f\right\} \tag{3.15}$$

with the relation

$$\forall f_1, f_2 \in \mathcal{F} : f_1 \equiv f_2 \Leftrightarrow \forall \boldsymbol{z}_i \in \mathcal{O}_l : L_{f_1}^{0|1}(\boldsymbol{z}_i) = L_{f_2}^{0|1}(\boldsymbol{z}_i). \tag{3.16}$$

That means two functions $f_1, f_2 \in \mathcal{F}$ are equivalent, if they produce the same decomposition (induced by $L_f^{0|1}$) of a random sample $\mathcal{O}_l$ in two disjoint sets.

Obviously, the number of equivalence classes $[f]$ on $\mathcal{F}$, denoted $N_{\mathcal{F}}(\mathcal{O}_l)$, equals the number of subsets of $\mathcal{O}_l$ induced by the events $\mathcal{A}_f, f \in \mathcal{F}$. Thus, no more than $N_{\mathcal{F}}(\mathcal{O}_l) \leq 2^l$ equivalence classes are possible (cf. Fig. 3.2), although the set of functions $\mathcal{F}$ is infinite.

Because $N_{\mathcal{F}}(\mathcal{O}_l)$ is a random variable, one may define the *entropy*

$$H_{\mathcal{F}}(l) := E\left\{\ln\left(N_{\mathcal{F}}(\mathcal{O}_l)\right)\right\} \tag{3.17}$$

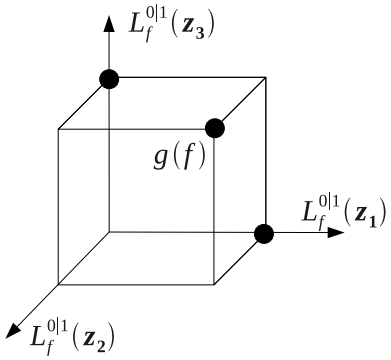


Figure 3.2: Sketch showing the space of distinguishable subsets (bold dots) of the sample $\mathcal{O}_l$, which are also subsets of the events $\mathcal{A}_f$ over the set of functions $\mathcal{F}$.

where E means the expectation with respect to the density $p(\mathcal{O}_l)$.

Defining this quantity one can show:

Theorem 3.3.1. *In order that (two-sided) uniform convergence over a set of indicator functions $L_f, f \in \mathcal{F}$, takes place, it is necessary and sufficient that the condition*

$$\lim_{l\to\infty} \frac{H_{\mathcal{F}}(l)}{l} = 0 \tag{3.18}$$

is satisfied.

Proof. See [Vapnik 1998],[Vapnik 2006]. ∎

The entropy $H_{\mathcal{F}}(l)$ depends on the sample size as well as on the function class. Unfortunately, it also depends on the distribution of the sample, which is not known in advance.

Applying *Jensen's inequality* (cf. App. A.1.2), we obtain

$$H_{\mathcal{F}}(l) \leq \ln\left(E\left\{N_{\mathcal{F}}(\mathcal{O}_l)\right\}\right) \leq \ln\left(\max_{\mathcal{O}_l}\left\{N_{\mathcal{F}}(\mathcal{O}_l)\right\}\right), \tag{3.19}$$

where

$$G_{\mathcal{F}}(l) := \ln\left(\max_{\mathcal{O}_l}\left\{N_{\mathcal{F}}(\mathcal{O}_l)\right\}\right) \tag{3.20}$$

is called the *Growth-Function*.

Using the Growth-Function, Theorem 3.3.1 can be restated:

Theorem 3.3.2. *In order that (two-sided) uniform convergence over a set of indicator functions $L_f, f \in \mathcal{F}$ is true, it is necessary and sufficient that the condition*

$$\lim_{l\to\infty} \frac{G_{\mathcal{F}}(l)}{l} = 0 \tag{3.21}$$

is satisfied.

The Growth-Function is distribution free, but not easy to compute. Thus, a easy to compute approximation of the Growth-Function would be convenient.

3.3.3 VC-Dimension

Suppose the set $\mathcal{F}$ would be as rich such that $N_{\mathcal{F}}(\mathcal{O}_l) = 2^l$. Then, it follows $G_{\mathcal{F}}(l) = l \cdot \ln(2)$, and thus the violation of the condition (Eq. 3.21).

Nevertheless, [Vapnik 1971] has shown, that any Growth-Function is linear, or there exists a maximum number $l = h$ such that the Growth-Function is bounded:

$$G_{\mathcal{F}}(l) \begin{cases} = l \cdot \ln(2) & \text{if } l \leq h \\ \leq h \left(\ln \left(\frac{l}{h} \right) + 1 \right) & \text{if } l > h \end{cases}, \tag{3.22}$$

(see Fig. 3.3).

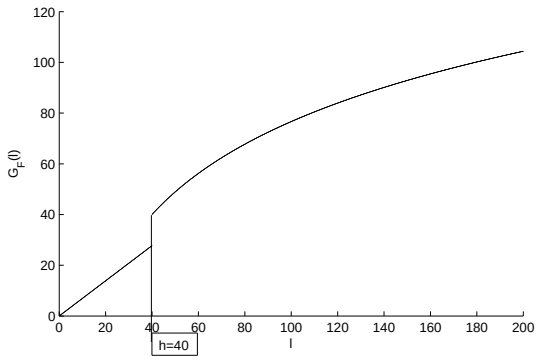


Figure 3.3: Graph bounding the Growth-Function, when successful learning is possible.

Definition 3.3.1 (VC-Dimension)**.** *The quantity h is called VC-Dimension, and it denotes the maximum number of points $\boldsymbol{z}_i \in \mathcal{X} \times \mathcal{Y}, 1 \leq i \leq h$ that can be decomposed in two subsets induced by a indicator function L_f, $f \in \mathcal{F}$, in 2^h ways.*

It is important to note, that the VC-Dimension of a set of functions is not to be confused with the term complexity for the number of free parameters of parameterized functions.

Applying a similar technique for bounding the rate of convergence as in the finite case (Eq. 3.10), one gets following (one-sided) rate of convergence if $\mathcal{F}$

is of infinite cardinality (cf. [Vapnik 2006] or [Schölkopf 2002][pp. 136-138]):

$$P\left(\sup_{f\in\mathcal{F}}\left(R(f)-R_l(f)\right)>\epsilon\right)\leq 4\cdot\exp\left(\ln\left(E\left\{N_{\mathcal{F}}(\mathcal{O}_{2l})\right\}\right)-\frac{\epsilon^2 l}{8}\right). \quad (3.23)$$

Using the inequalities (Eq. 3.19) and the property (Eq. 3.22) of the Growth-Function, it follows for $l > h$:

$$P\left(\sup_{f\in\mathcal{F}}\left(R(f)-R_l(f)\right)>\epsilon\right)\leq \underbrace{4\cdot\exp\left(h\left(\ln\left[\frac{2l}{h}\right]+1\right)-\frac{\epsilon^2 l}{8}\right)}_{=:\nu}, \quad (3.24)$$

and theorem (3.3.2) applies.

By definition of $\nu > 0$ (Eq. 3.24), one obtains the relation

$$\epsilon=\sqrt{\frac{8}{l}\left[h\left(\ln\left[\frac{2l}{h}\right]+1\right)-\ln\left(\frac{\nu}{4}\right)\right]}. \quad (3.25)$$

Therefore, it holds at least with probability $1-\nu$ for all $f \in \mathcal{F}$ (including the minimizer f^l of the empirical risk):

$$R(f)\leq R_l(f)+\epsilon. \quad (3.26)$$

3.4 Structural Risk Minimization

The risk bound for a set of functions $f \in \mathcal{F}$ (assuming a indicator loss function)

$$R(f)\leq R_l(f)+\Phi_h\left(\frac{l}{h}\right) \quad (3.27)$$

consists of two terms, namely the empirical risk R_l and the *confidence term*

$$\Phi_h(x):=\sqrt{\frac{8}{x}\left[\left(\ln(2x)+1\right)-\frac{1}{h}\ln\left(\frac{\nu}{4}\right)\right]},\ x>1. \quad (3.28)$$

If a large number l of samples is given, then the bound (Eq. 3.27) depends mainly on the empirical risk. But if the sample size is small compared to the VC-dimension, i.e. $1 < l/h$ is small, the influence of the confidence term is significant. Thus, to obtain a low actual risk R, one has to optimize both terms simultaneously.

An approach to find the minimizer $f^* \in \mathcal{F}$ of the right-hand side in (Eq. 3.27) is called *Structural Risk Minimization* (SRM) [Vapnik 1998].

SRM suggests to introduce a structure $\mathcal{S} := \bigcup_k \mathcal{F}_k, \mathcal{F}_k \subset \mathcal{F}$ such that

$$\mathcal{F}_1 \subset \mathcal{F}_2 \subset \cdots \subset \mathcal{F}_n \tag{3.29}$$

and the VC-dimension of each element of the structure $h(\mathcal{F}_k)$ satisfies

$$h(\mathcal{F}_1) \leq h(\mathcal{F}_2) \leq \cdots \leq h(\mathcal{F}_n). \tag{3.30}$$

Given such a structure $\mathcal{S}$ (Fig. 3.4) determine the function $f_k^l \in \mathcal{F}_k$ of the subset $\mathcal{F}_k$, which is the minimizer of R_l. Then select from the set of minimizing functions $\left\{f_1^l, \ldots, f_n^l\right\}$ a function f^* that minimizes the risk bound (Eq. 3.27).

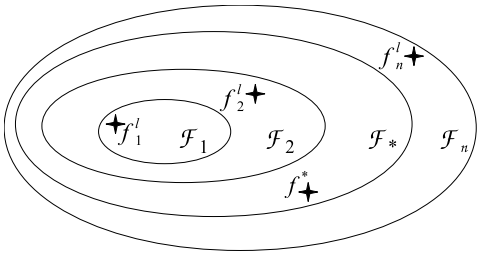


Figure 3.4: Nested subsets of $\mathcal{F}$ used for the SRM method.

In the following, it will be shown how to construct a nested sequence of subsets of the space of (affine-)linear decision functions, which is one of the most important spaces for constructing classification algorithms.

3.5 Maximum Margin Hyperplanes

Consider a set of linear parameterized decision functions:

$$\mathcal{F}_{lin} := \left\{ f : \mathcal{X} \to \mathbb{R} \;\middle|\; f(\boldsymbol{x}, c_1, \ldots, c_n, b) := \sum_{k=1}^{n} c_k \Phi_k(\boldsymbol{x}) + b, \; (c_1, \ldots, c_n, b) \in \mathbb{R}^{n+1} \right\} \tag{3.31}$$

with $\Phi_k : \mathcal{X} \to \mathbb{R}$.

Recall, the VC-dimension is defined as the maximum number h of points $\boldsymbol{x} \in \mathcal{X}$ that can be separated in two disjoint sets in all 2^h ways using indicator functions $L_f^{0|1}$ (Def. 3.3.1).

Obviously, for a set of linear functions $\mathcal{F}_{lin}$ (3.31) in $\mathbb{R}^n$ it holds $h = n+1$, cf. (Fig. 3.5), thus this is also true for the set of 0/1-loss functions $\{L_f^{0|1} : f \in \mathcal{F}_{lin}\}$ with respect to the parameter space.

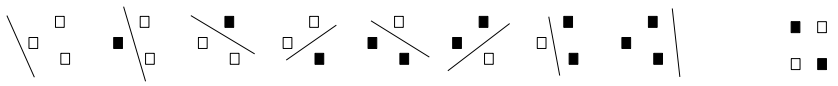


Figure 3.5: An example of separating hyperplanes in two dimensions. Note, $h = 3$ points can be linearly separated but four can not. In general, hyperplanes in $\mathbb{R}^n$ separate a maximum of $h = n+1$ points in all 2^h possible binary classifications.

In view of the definition of the Growth-function (3.20) and its properties (3.22), it follows for a sample $\mathcal{O}_l \subset X \times \mathcal{Y}$, $\mathcal{X} := \mathbb{R}^n$ of size $l > h$ that the number $N_{\mathcal{F}_{lin}}(\mathcal{O}_l)$ of equivalence classes $[f] \subset \mathcal{F}_{lin}$ (Def. 3.15) is bounded by

$$N_{\mathcal{F}_{lin}}(\mathcal{O}_l) \leq \exp(h) \left(\frac{l}{h}\right)^h, \; h = n+1. \tag{3.32}$$

An appropriate structure $\mathcal{S} = \bigcup_r \mathcal{F}_r \subseteq \mathcal{F}_{lin}$ can be constructed by

$$\mathcal{F}_1 := \emptyset \tag{3.33}$$

$$\mathcal{F}_r := \left\{ f \in \mathcal{F} : \rho_f / D > 1/\sqrt{r-1} \right\} \text{ for } 2 \leq r < n \tag{3.34}$$

$$\mathcal{F}_r := \mathcal{F} \text{ for } r \geq n, \tag{3.35}$$

where D is the diameter of the smallest sphere enclosing all vectors, and ρ_f denotes the distance between the convex hulls of vectors $\boldsymbol{x} \in \mathbb{R}^n$ separated by functions $f \in \mathcal{F}_r$ into two classes (Fig. 3.6). The smallest distance

$$\rho_r := \inf \{ \rho_f / D : f \in \mathcal{F}_r \} \tag{3.36}$$

$$= \min \{ \rho_f / D : [f] \in [\mathcal{F}_r] \}, \tag{3.37}$$

where $[\mathcal{F}_r]$ denotes the set of all equivalence classes with respect to $\mathcal{F}_r$, equals two times the maximum of the minimum distance, which is called the margin $\rho = \rho_r/2$, between a hyperplane H and all points.

For a structure $\mathcal{S}$ constructed in this way, the following important lemma holds:

Lemma 3.5.1. *The number $N_{\mathcal{F}_r}(\mathcal{O}_l)$ of equivalence classes in the structure element $\mathcal{F}_r \subseteq \mathcal{F}_{lin}$ (i.e., functions linear in their parameters, cf. (3.31)) is bounded by*

$$N_{\mathcal{F}_r}(\mathcal{O}_l) \leq exp(h_r) \left(\frac{l}{h_r}\right)^{h_r} \tag{3.38}$$

for $l > h_r$, where h_r is the VC-Dimension of $\mathcal{F}_r$ bounded by

$$h_r \leq \min \left\{ n, \left\lceil \frac{D^2}{\rho_r^2} \right\rceil \right\} + 1, \tag{3.39}$$

n is the dimensionality of the feature space, and $\lceil \cdot \rceil$ is the ceiling function.

Proof. See [Vapnik 2006][pp. 323]. ∎

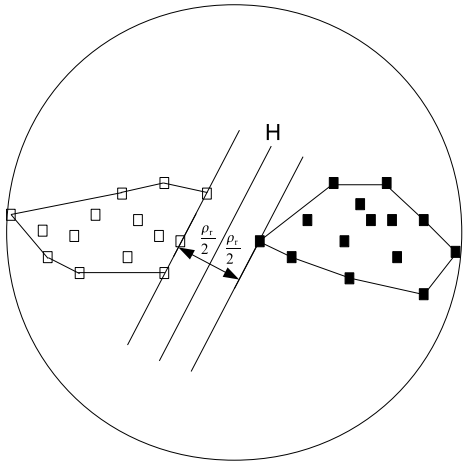


Figure 3.6: Example of a separation of a set of points in two subsets induced by a equivalence class $[f_r]$. The distance ρ_r between the convex hulls of the sets equals two times the maximum of the minimum distance (margin) between a hyperplane H and the two classes of points.

The result of Lemma (3.5.1) and the bound on the actual risk (3.13) propose the following procedure for selecting an optimal linear decision function $f^* \in \mathcal{F}$: choose f^* such that the empirical error is minimized and the minimum distance between a separating hyperplane induced by f^* to the convex hulls of two classes of points is maximized. In particular, Lemma (3.5.1) is important for explaining the generalization performance of Support Vector Machines revisited in the next section. However, it is important to note, the argument that SVMs implement the SRM principle is known to be flawed (for a discussion of this issue, see e.g. [Burges 1998]). This is because the structure $\mathcal{S}$ constructed by SVMs is based on the training data which is known in advance. And this contradicts the assumption of SRM, because the structure $\mathcal{S}$ must be constructed before the data arrives. Nevertheless, Lemma (3.5.1) strongly suggests that algorithms minimizing D^2/ρ_r^2 can be expected to give better generalization performance. In particular, one can show [Vapnik 1999] the expectation of a test error P_{err} using maximum margin hyperplanes associated to training sets of size $l-1$ can be bounded by $E(P_{err}) \leq E(D^2/\rho_r^2)/l$. This result gives further evidence in the argumentation for a maximum margin concept.

3.6 Support Vector Machines Revisited

As concluded from Lemma (3.5.1) of the preceding section, among all separating hyperplanes H_f induced by linear decision functions f classifying two

classes of points with empirical error $R_l(f) = 0$, the hyperplane H_{f^*} (respectively the function $f^* \in \mathcal{F}$) that maximizes a margin ρ has to be preferred in order to minimize the actual risk R. Moreover, the VC-Dimension h^* of a family of maximum margin separating hyperplanes can be much smaller than the VC-Dimension of non-restricted hyperplanes in $\mathbb{R}^n$, i.e. $h^* \leq h = n + 1$. It follows the probability that a maximum margin hyperplane missclassifies a test sample is bounded at least with probability $1 - \nu$ by

$$P_{err}(f^*) \leq \sqrt{\frac{8}{l}\left[h^*\left(\ln\left[\frac{2l}{h^*}\right] + 1\right) - \ln\left(\frac{\nu}{4}\right)\right]} \tag{3.40}$$

with VC-Dimension

$$h^* \leq \min\left\{n, \left\lceil\frac{D^2}{\rho^2}\right\rceil\right\} + 1 \tag{3.41}$$

where n is the dimension of the feature space, D is the radius of a data enclosing sphere, and ρ denotes the margin.

3.6.1 The Separable Case

Recall the Perceptron Algorithm introduced in Section (2.3.3) in case of linear separability. The signed distance between a point $\boldsymbol{x}$ and a separating hyperplane $f \in \mathcal{F}_{lin}$ with $f(\boldsymbol{x}) := \boldsymbol{w}^T\boldsymbol{\Phi}(\boldsymbol{x}) + b$ is given by

$$\rho = \frac{1}{\|\boldsymbol{w}\|}(\boldsymbol{w}^T\boldsymbol{\Phi}(\boldsymbol{x}) + b). \tag{3.42}$$

Now, consider a training set $\mathcal{O}_N$ containing points of two classes that are linear separable, i.e. there exist a margin $\rho > 0$ (For the non-linearly separable case refer to Section (3.6.2)). Then, the requirements that $R_N(f) = 0$, i.e. $\forall 1 \leq i \leq N : (\boldsymbol{w}^T\boldsymbol{\Phi}(\boldsymbol{x}_i)+b)\cdot y_i > 0$, as well as that all points have to maintain a distance of at least $\rho > 0$ to the hyperplane H_f can be written

$$\forall 1 \leq i \leq N \;:\; \frac{1}{\|\boldsymbol{w}\|}(\boldsymbol{w}^T\boldsymbol{\Phi}(\boldsymbol{x}_i) + b) \cdot y_i \geq \rho. \tag{3.43}$$

For any $\boldsymbol{w}$ and b satisfying these constraints, also any positively scaled multiple satisfies them. Hence the constraints can be arbitrarily rewritten with $\rho := \frac{1}{\|w\|}$ as

$$\forall 1 \leq i \leq N \;:\; (\boldsymbol{w}^T\boldsymbol{\Phi}(\boldsymbol{x}_i) + b) \cdot y_i \geq 1. \tag{3.44}$$

Due to this rescaling, the optimal hyperplane is in canonical form such that its margin is given by $\rho = 1/\|\boldsymbol{w}\|$ (Fig. 3.7).

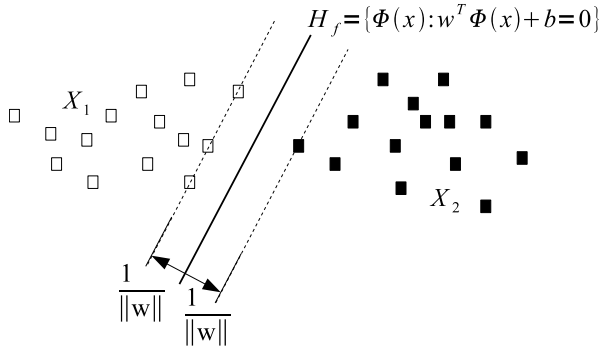


Figure 3.7: Canonical maximum margin hyperplane with four points on the margin boundary.

Because a maximization of $1/\|\boldsymbol{w}\|$ is conveniently equivalent to a minimization of $(1/2)\|\boldsymbol{w}\|^2$, it follows the optimization problem of the Support Vector Machine in case of two linearly separable classes:

Algorithm 3.6.1 Support Vector Machine (Hard-Margin, Primal Version) [Vapnik 1999]

Require: Training set $\emptyset \neq \mathcal{O}_N \subset \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, a mapping $\boldsymbol{\Phi} : \mathcal{X} \to \mathcal{H} \subseteq \mathbb{R}^n$.
Ensure: $\exists(\boldsymbol{x}_j, y_j), (\boldsymbol{x}_i, y_i) \in \mathcal{O}_N \ : \ y_j = -1 \wedge y_i = 1$, both classes are linearly separable in $\mathcal{H}$

$$\min_{(\boldsymbol{w},b)\in\mathbb{R}^{n+1}} \tfrac{1}{2}\|\boldsymbol{w}\|^2 \tag{3.45}$$

$$\text{s.t.} \ \ \forall 1 \leq i \leq N \ : \ (\boldsymbol{w}^T\boldsymbol{\Phi}(\boldsymbol{x}_i) + b) \cdot y_i - 1 \geq 0 \tag{3.46}$$

The SVM optimization problem can be solved using *Lagrange duality* (App. A.3.4). Define the *Lagrange function* $L : \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^N \to \mathbb{R}$ with

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) := \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{N} \alpha_i \left((\boldsymbol{w}^T\boldsymbol{\Phi}(\boldsymbol{x}_i) + b) \cdot y_i - 1 \right). \tag{3.47}$$

In order to determine the *dual function*

$$\mathcal{D}L(\boldsymbol{\alpha}) := \inf \left\{ L(\boldsymbol{w}, b, \boldsymbol{\alpha}) : (\boldsymbol{w}, b) \in \mathbb{R}^{n+1} \right\}, \tag{3.48}$$

the derivatives of the Lagrange function with respect to the parameters $(\boldsymbol{w}, b)$

have to be set to zero yielding:

$$\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{\Phi}(\boldsymbol{x}_i) \tag{3.49}$$

$$0 = \sum_{i=1}^{N} \alpha_i y_i. \tag{3.50}$$

The infimum in (3.48) is finite, if and only if the linear constraint (3.50) is satisfied, and it is uniquely attained for optimal weights given by equation (3.49). Thus, using (3.50) and (3.49) in (3.48) gives the dual function of the SVM for all $\alpha \in \mathbb{R}^N : \sum_{i=1}^{N} \alpha_i y_i = 0$:

$$\mathcal{D}L(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{\Phi}(\boldsymbol{x}_i), \boldsymbol{\Phi}(\boldsymbol{x}_j) \rangle_2. \tag{3.51}$$

The dual function is a concave (quadratic) function (App. A.3.2). Thus, in virtue of the strong duality theorem from convex optimization theory (App. A.3.4.2), the SVM algorithm can equivalently reformulated:

Algorithm 3.6.2 Support Vector Machine (Hard-Margin, Dual Version) [Vapnik 1999]

Require: Training set $\emptyset \neq \mathcal{O}_N \subset \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, a mapping $\boldsymbol{\Phi} : \mathcal{X} \to \mathcal{H} \subseteq \mathbb{R}^n$

Ensure: $\exists (\boldsymbol{x}_j, y_j), (\boldsymbol{x}_i, y_i) \in \mathcal{O}_N \ : \ y_j = -1 \wedge y_i = 1$, both classes are linearly separable in $\mathcal{H}$

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^N} \sum_{i=1}^{N} \alpha_i - \tfrac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{\Phi}(\boldsymbol{x}_i), \boldsymbol{\Phi}(\boldsymbol{x}_j) \rangle_2 \tag{3.52}$$

$$\text{s.t.} \quad \forall 1 \leq i \leq N \ : \ \alpha_i \geq 0 \tag{3.53}$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{3.54}$$

Due to the *Karush-Kuhn-Tucker optimality conditions* (App. A.3.3) the unique global solution $\boldsymbol{\alpha}^*$ of (3.52), (3.53), (3.54) must satisfies the equations

$$\forall 1 \leq i \leq N \ : \ \alpha_i^* \left((\boldsymbol{w}^T \boldsymbol{\Phi}(\boldsymbol{x}_i) + b) \cdot y_i - 1 \right) = 0. \tag{3.55}$$

From these, we can conclude for all $\alpha_i^* > 0$ it must hold $(\boldsymbol{w}^T \boldsymbol{\Phi}(\boldsymbol{x}_i) + b) y_i - 1 = 0$, i.e. the corresponding vectors $\boldsymbol{\Phi}(\boldsymbol{x}_i)$ are on the margin boundary. A vector for which α_i^* is strictly positive, is called *Support Vector* (SV). The Support Vectors are essential for constructing the maximum margin hyperplane (3.49),

because for vectors that are far away from the hyperplane and are classified on the correct side it holds $(\boldsymbol{w}^T\boldsymbol{\Phi}(\boldsymbol{x}_i)+b)y_i - 1 > 0$ and therefore $\alpha_i^* = 0$. In Figure (3.7) four Support Vectors are shown.

The decision function of the SVM reads as

$$g(\boldsymbol{x}) = \text{sign}\left(\sum_{i\ :\ \alpha_i^*>0} \alpha_i^* y_i \langle \boldsymbol{\Phi}(\boldsymbol{x}_i), \boldsymbol{\Phi}(\boldsymbol{x})\rangle_2 + b^*\right) \tag{3.56}$$

where the optimal bias b^* is usually determined from the Equations (3.55) using some Support Vectors. Although theoretically not necessary, but because of numerical reasons, often b^* is computed by averaging the solutions of (3.55) for more than one Support Vector. For example consider two Support Vectors $\boldsymbol{\Phi}(\boldsymbol{x}_i)$ with $y_i = 1, \alpha_i > 0$ and $\boldsymbol{\Phi}(\boldsymbol{x}_j)$ with $y_j = -1, \alpha_j > 0$. Taking the arithmetical mean of the solutions $b(\boldsymbol{\Phi}(\boldsymbol{x}_i))$ and $b(\boldsymbol{\Phi}(\boldsymbol{x}_j))$ obtained from the associated Equations (3.55) yields

$$b^* = -\frac{\sum\limits_{k\ :\ \alpha_k^*>0} \alpha_k^* y_k \langle \boldsymbol{\Phi}(\boldsymbol{x}_k), \boldsymbol{\Phi}(\boldsymbol{x}_j) + \boldsymbol{\Phi}(\boldsymbol{x}_i)\rangle_2}{2}. \tag{3.57}$$

3.6.2 The Nonseparable Case

So far, we assumed that the two classes of points of a training set $\mathcal{O}_N$ are linear separable. This is seldom the case for real-world data. Indeed, the classes almost always overlap in the feature space. In order to deal with the overlap, one still maximize the margin ρ, but one allows for some points to be on the wrong side of the margin.

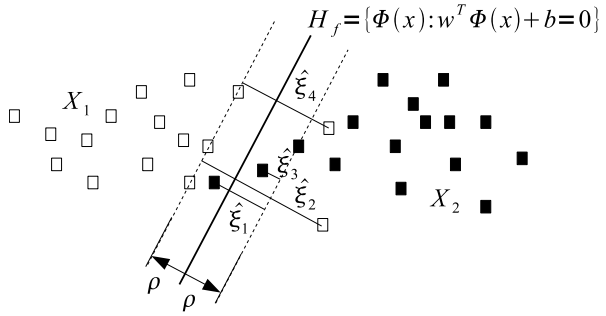


Figure 3.8: Maximum soft-margin hyperplane in case of overlapping classes. Four points are on the wrong side of the margin by an amount $\hat{\xi}_i = \rho\xi_i$.

For this purpose so-called *slack variables* $\boldsymbol{\xi} := (\xi_1, \ldots, \xi_N) \geq \mathbf{0}$ are introduced and the constraints (3.43) are modified to be

$$\forall 1 \leq i \leq N \ : \ \frac{1}{\|\boldsymbol{w}\|}(\boldsymbol{w}^T\boldsymbol{\Phi}(\boldsymbol{x}_i) + b) \cdot y_i \geq \rho(1 - \xi_i). \tag{3.58}$$

Thus the value of ξ_i is proportional to the distance a point $\boldsymbol{\Phi}(\boldsymbol{x}_i)$ is on the wrong side of the margin boundary (Fig. 3.8). If $\xi_i > 1$, then a misclassification occurs. Normalizing the constraints (3.58) with $\rho = 1/\|\boldsymbol{w}\|$ as in (3.43) yields

$$\forall 1 \leq i \leq N \ : \ (\boldsymbol{w}^T \boldsymbol{\Phi}(\boldsymbol{x}_i) + b) \cdot y_i \geq 1 - \xi_i. \tag{3.59}$$

In order to get the best compromise between minimizing the total amount of margin errors $\sum_{i=1}^N \xi_i$ and maximizing the margin size $(1/2)\|\boldsymbol{w}\|^2$, the objective is to minimize $(1/2)\|\boldsymbol{w}\|^2 + C\sum_{i=1}^N \xi_i$ with fixed $C > 0$. The constant C controls the trade-off between minimum margin errors and maximum margin size. Alternatively, a (local) optimal maximum margin hyperplane can found by e.g. minimizing the number of correctly classified samples [Wendemuth 1995].

Applying the Lagrangian dual approach in the same way as shown in case of the SVM hard-margin algorithm (3.6.2), the soft-margin version of the SVM algorithm can be obtained:

Algorithm 3.6.3 Support Vector Machine (Soft-Margin, Dual Version) [Vapnik 1999]

Require: Training set $\emptyset \neq \mathcal{O}_N \subset \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, a mapping $\boldsymbol{\Phi} : \mathcal{X} \to \mathcal{H} \subseteq \mathbb{R}^n$, trade-off paramater $C > 0$.

Ensure: $\exists (\boldsymbol{x}_j, y_j), (\boldsymbol{x}_i, y_i) \in \mathcal{O}_N \ : \ y_j = -1 \wedge y_i = 1$

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^N} \sum_{i=1}^N \alpha_i - \tfrac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \langle \boldsymbol{\Phi}(\boldsymbol{x}_i), \boldsymbol{\Phi}(\boldsymbol{x}_j) \rangle_2 \tag{3.60}$$

$$\text{s.t.} \ \ \forall 1 \leq i \leq N \ : \ 0 \leq \alpha_i \leq C \tag{3.61}$$

$$\sum_{i=1}^N \alpha_i y_i = 0 \tag{3.62}$$

The only difference compared to the hard-margin SVM algorithm is the box-constraint (3.61). The decision function reads as

$$g(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{i \ : \ \alpha_i^* > 0} \alpha_i^* y_i \langle \boldsymbol{\Phi}(\boldsymbol{x}_i), \boldsymbol{\Phi}(\boldsymbol{x}) \rangle_2 + b^* \right) \tag{3.63}$$

where b^* can be obtained from the Karush-Kuhn-Tucker conditions

$$\forall 1 \leq i \leq N \ : \ \alpha_i^* \left((\boldsymbol{w}^T \boldsymbol{\Phi}(\boldsymbol{x}_i) + b) \cdot y_i - 1 + \xi_i \right) = 0 \tag{3.64}$$

using Support Vectors with $0 < \alpha_i < C$ for which one can show that $\xi_i = 0$. Because of numerical reasons, often b^* is computed as in (3.57) by averaging

solutions of (3.64) with respect to b using more than one Support Vector. For example using two Support Vectors $\mathbf{\Phi}(\boldsymbol{x}_j), \mathbf{\Phi}(\boldsymbol{x}_i)$ with class labels $y_j = -1$ respectively $y_i = 1$ and $0 < \alpha_i, \alpha_j < C$ yields

$$b^* = -\frac{\sum\limits_{k \,:\, \alpha_k^* > 0} \alpha_k^* y_k \langle \mathbf{\Phi}(\boldsymbol{x}_k), \mathbf{\Phi}(\boldsymbol{x}_j) + \mathbf{\Phi}(\boldsymbol{x}_i) \rangle_2}{2}. \tag{3.65}$$

3.6.3 Mercer Kernel Hilbert Space

One can see from the SVM algorithms (3.6.2), (3.6.3) and the decision function (3.63), the data is only involved via Euclidean inner products $\langle \mathbf{\Phi}(\boldsymbol{x}), \mathbf{\Phi}(\boldsymbol{y}) \rangle_2$. The inner products are evaluated in a feature space $\mathcal{H}$ implied by some transformation $\mathbf{\Phi} : \mathcal{X} \to \mathcal{H}$. Thus, in order to obtain a nonlinear decision function in the original data space $\mathcal{X}$, one has to transform the data via $\mathbf{\Phi}$ into a high-dimensional feature space first. Then, the SVM learns a maximum margin hyperplane in the feature space using the inner products, while the implicit back-projection gives a nonlinear decision boundary in the raw data space (Fig. 2.11).

Because such a data transformation into spaces with very high dimension is computational intractable, one uses following theorem in order to implicitly evaluate the inner products:

Theorem 3.6.1 (Mercer's Condition, cf. [Courant 1953])**.** *To guarantee that a symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ from L_2 has an expansion*

$$k(\boldsymbol{x}, \boldsymbol{y}) := \sum_{k=1}^{N_{\mathcal{H}}} a_k \Psi_k(\boldsymbol{x}) \Psi_k(\boldsymbol{y}) \tag{3.66}$$

with positive eigenvalues $a_k > 0$ and eigenfunctions Ψ_k, i.e. k describes an inner product in some Hilbert space $\mathcal{H}$ of either finite or infinite dimension $N_{\mathcal{H}}$, it is necessary and sufficient that the condition

$$\int_{\mathcal{X}} \int_{\mathcal{X}} k(\boldsymbol{x}, \boldsymbol{y}) f(\boldsymbol{x}) f(\boldsymbol{y}) \; d\boldsymbol{x} \; d\boldsymbol{y} > 0 \tag{3.67}$$

holds for all $f \neq 0$ with

$$\int_{\mathcal{X}} f^2(\boldsymbol{x}) \; d\boldsymbol{x} < \infty. \tag{3.68}$$

Proof. See [Mercer 1909]. ∎

Note, it follows a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel, if for all $N \in \mathbb{N}$ and all $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathcal{X}$ the *kernel matrix* $\mathbf{K} \in \mathbb{R}^{N \times N}$ with elements $K_{ij} := k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is real, symmetric and positive definite.

Mercer's theorem states a correspondence of a function $k \in L_2$ (*kernel* function) and its associated (pre-)Hilbert space $\mathcal{H}$ (Def. A.2.3). This enables to implicitly evaluate the inner products $\langle \boldsymbol{\Phi}(\boldsymbol{x}), \boldsymbol{\Phi}(\boldsymbol{y}) \rangle_2 = k(\boldsymbol{x}, \boldsymbol{y})$ for transformations $\boldsymbol{\Phi}$ with components $\Phi_k(\boldsymbol{x}) := \sqrt{a_k}\Psi_k(\boldsymbol{x})$ without explicitly applying $\boldsymbol{\Phi}$. The correspondence of k and $\mathcal{H}$ is not unique, because for a given kernel there are usually many different feature spaces possible. For example consider the kernel $k(\boldsymbol{x}, \boldsymbol{y}) := \langle \boldsymbol{x}, \boldsymbol{y} \rangle^2$ that has following equivalent finite factorizations $k(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{\Phi}_1(\boldsymbol{x}), \boldsymbol{\Phi}_1(\boldsymbol{y}) \rangle_2 = \langle \boldsymbol{\Phi}_2(\boldsymbol{x}), \boldsymbol{\Phi}_2(\boldsymbol{y}) \rangle_2$ using

$$\boldsymbol{\Phi}_1(\boldsymbol{x}) := \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix} \tag{3.69}$$

$$\boldsymbol{\Phi}_2(\boldsymbol{x}) := \frac{1}{\sqrt{2}} \begin{pmatrix} x_1^2 - x_2^2 \\ 2x_1x_2 \\ x_1^2 + x_2^2 \end{pmatrix}. \tag{3.70}$$

Figure (3.9) visualizes the mappings under $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Phi}_2$ from $\mathbb{R}^2$ to $\mathbb{R}^3$.

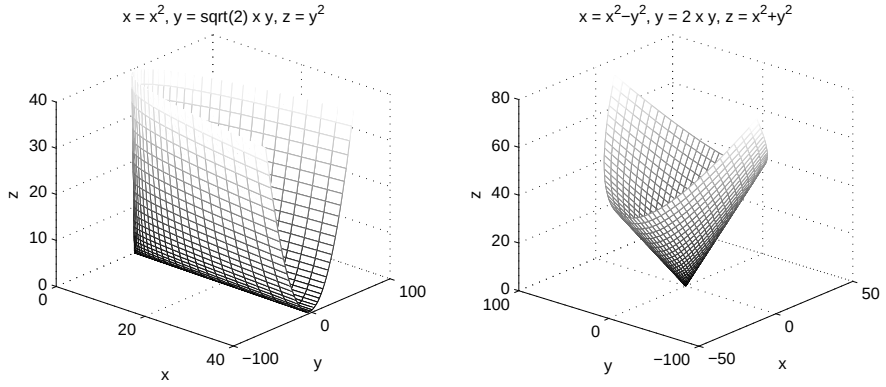
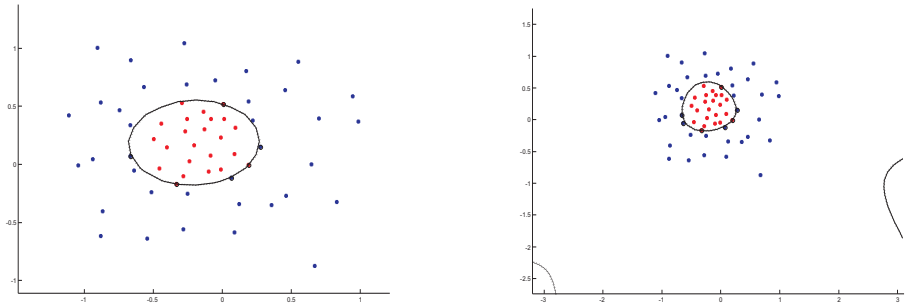


Figure 3.9: Two different Hilbert spaces embeddings represented by a kernel function.

Thus, in order to obtain nonlinear decision boundaries using SVMs (e.g. Fig. (3.10)) the inner products appearing in the Algorithms (3.6.2), (3.6.3) and the decision function (3.63) have to be substituted by appropriate kernel functions satisfying the Mercer's condition. Unfortunately, the Mercer's condition is proved only for a handful of kernels, summarized in Section (3.6.5). However, a remarkable property of kernel based SVMs is, that although actually learning a hyperplane in a space even of infinite dimension, a solution of



(a) Polynomial kernel SVM. (b) RBF-kernel SVM.

Figure 3.10: Nonlinear decision boundaries learnt by different SVMs.

the SVM is always built-up of a finite linear combination of kernel functions. This result is known as *Representer Theorem* [Kimeldorf 1971] (see Section (4.2.3), or refer to [Schölkopf 2002] and references therein).

In this section we noted, that many equivalent Hilbert spaces with respect to the kernel function exist. Because of its importance in analyzing SVM feature mappings, we additionally introduce in the next section the so-called Reproducing Kernel Hilbert Space (RKHS).

3.6.4 Reproducing Kernel Hilbert Space

Given a kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with non-empty and finite data space $\mathcal{X}$ (also called *index set*), one can define a feature map $\boldsymbol{\Phi} : \mathcal{X} \to \mathcal{H}$ using $\boldsymbol{\Phi}(\boldsymbol{x}) := k(\cdot, \boldsymbol{x})$. That means, the data space is mapped to the space

$$\mathcal{H} := span\{k(\cdot, \boldsymbol{x}) : \mathcal{X} \to \mathbb{R} \mid \boldsymbol{x} \in \mathcal{X}\}, \tag{3.71}$$

which is spanned by the kernel functions over $\mathcal{X}$. Thus, $\mathcal{H}$ is a linear space.

In order to turn the space $\mathcal{H}$ also into a Hilbert space (App. (A.2.3)), we have to endow the space with an appropriate inner product first and second we have to complete it[1]. For the former, consider two arbitrary sequences $(\hat{\boldsymbol{x}}_i)_{n=1}^N \in \mathcal{X}, N \in \mathbb{N}$ and $(\check{\boldsymbol{x}}_m)_{m=1}^M \in \mathcal{X}, M \in \mathbb{N}$ and the functions $g, f \in \mathcal{H}$ with

$$f(\boldsymbol{x}) := \sum_{n=1}^{N} \alpha_n k(\boldsymbol{x}, \hat{\boldsymbol{x}}_n) \tag{3.72}$$

[1] *completed* means that every *Cauchy sequence* in $\mathcal{H}$ converges in the induced metric to an element of $\mathcal{H}$

and

$$g(\boldsymbol{x}) := \sum_{m=1}^{M} \beta_m k(\boldsymbol{x}, \check{\boldsymbol{x}}_m). \tag{3.73}$$

where $\alpha_i, \beta_m \in \mathbb{R}$.

Now, we show that

$$\langle f, g \rangle_{\mathcal{H}} := \sum_{n=1}^{N} \sum_{m=1}^{M} \alpha_n \beta_m k(\hat{\boldsymbol{x}}_n, \check{\boldsymbol{x}}_m) \tag{3.74}$$

is a well-defined inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}} : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ (cf. App. (A.2.3)):

1. By definition, it holds $\langle f, g \rangle_{\mathcal{H}} = \sum_{n=1}^{N} \alpha_n g(\hat{\boldsymbol{x}}_n) = \sum_{m=1}^{M} \beta_m f(\check{\boldsymbol{x}}_m) = \langle g, f \rangle_{\mathcal{H}}$ (symmetry and bilinearity).
2. Because k is a kernel it holds $\langle f, f \rangle_{\mathcal{H}} = \sum_{n=1}^{N} \sum_{m=1}^{M} \alpha_n \alpha_m k(\hat{\boldsymbol{x}}_n, \hat{\boldsymbol{x}}_m) \geq 0$ (positivity).
3. By definition, it holds $\langle k(\cdot, \boldsymbol{x}), f \rangle_{\mathcal{H}} = f(\boldsymbol{x})$ and in particular

$$\langle k(\cdot, \boldsymbol{x}), k(\cdot, \hat{\boldsymbol{x}}) \rangle_{\mathcal{H}} = k(\boldsymbol{x}, \hat{\boldsymbol{x}}) \text{ (\textit{reproducing kernel property})}, \tag{3.75}$$

thus in virtue of the *Cauchy-Schwarz inequality* we get

$$|f(\boldsymbol{x})|^2 = |\langle k(\cdot, \boldsymbol{x}), f \rangle_{\mathcal{H}}|^2 \leq k(\boldsymbol{x}, \boldsymbol{x}) \cdot \langle f, f \rangle_{\mathcal{H}}. \tag{3.76}$$

This implies $\langle f, f \rangle_{\mathcal{H}} = 0$ if and only if $f = 0$.

It follows, the *Reproducing Kernel Hilbert Space* (RKHS) $\mathcal{R} := (\mathcal{H}, \langle \cdot, \cdot \rangle_{\mathcal{H}})$ is a well-defined inner product space. Moreover, because of the reproducing kernel property, it holds $\langle \boldsymbol{\Phi}(\boldsymbol{x}), \boldsymbol{\Phi}(\boldsymbol{y}) \rangle_{\mathcal{H}} = k(\boldsymbol{x}, \boldsymbol{y})$ showing that $\mathcal{R}$ is a further possible example of a feature space implied by a kernel k, respectively the feature map $\boldsymbol{\Phi}$.

3.6.5 Kernel Functions

In the preceding sections, we presented two feature spaces, namely Mercer Kernel Hilbert Spaces and Reproducing Kernel Hilbert Spaces. Both are implied by the choice of a particular kernel function. Recall, a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a kernel, if it satisfies the Mercer's condition (Theorem (3.6.1)). In the following, we summarize the few functions k that are proved to be kernels usable for SVMs (for some proofs and more details, we refer the reader to e.g. [Burges 1998], [Schölkopf 2002]):

- **Radial Basis Function (RBF) kernel**

$$k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\gamma \left\|\boldsymbol{x} - \boldsymbol{y}\right\|_2^2\right), \gamma > 0 \tag{3.77}$$

- **Polynomial kernel**

$$k(\boldsymbol{x},\boldsymbol{y}) = \left(\langle \boldsymbol{x},\boldsymbol{y}\rangle_2 + c\right)^d, c \geq 0, d \in \mathbb{N} \tag{3.78}$$

- **Dirichlet kernel**

$$k(\boldsymbol{x},\boldsymbol{y}) = \frac{\sin\left((N+1/2)(\boldsymbol{x}-\boldsymbol{y})\right)}{2\sin\left((\boldsymbol{x}-\boldsymbol{y})/2\right)},\ N \in \mathbb{N} \tag{3.79}$$

- **B-spline kernel**

$$k(\boldsymbol{x},\boldsymbol{y}) = B_{2p+1}(\|\boldsymbol{x}-\boldsymbol{y}\|_2),\ p \in \mathbb{N} \tag{3.80}$$

  The kernel computes B-splines of order $2p+1$ defined by $(2p+1)$-fold convolution of indicator functions $I_{[-\frac{1}{2},\frac{1}{2}]}(x) := 1$ if $x \in [-\frac{1}{2},\frac{1}{2}]$, otherwise $I_{[-\frac{1}{2},\frac{1}{2}]}(x) := 0$.
- **natural kernel**

$$k(\boldsymbol{x},\boldsymbol{y}) = \nabla_\vartheta \ln p(\boldsymbol{x}|\vartheta)\mathbf{M}^{-1}\nabla_\vartheta \ln p(\boldsymbol{y}|\vartheta), \mathbf{M} \succ 0 \tag{3.81}$$

  Natural kernels are particular scalar products, defined by a positive definite matrix $\mathbf{M}$, measuring similarities between data which has been transformed to a so-called *score-space* [Schölkopf 2002]. In case of $\mathbf{M}_{ij} := \int_{\mathcal{X}} \partial_{\vartheta_i} \ln p(\boldsymbol{x}|\vartheta)\partial_{\vartheta_j} \ln p(\boldsymbol{x}|\vartheta)p(\boldsymbol{x})\ d\boldsymbol{x}$ the natural kernel is called **Fisher-kernel** [Jaakkola 1999]. Setting $\mathbf{M} := \mathbf{I}$ one obtains the so-called **Plain kernel** [Schölkopf 2002].
- **convolution kernel** [Haussler 1999]

$$k(\boldsymbol{x},\boldsymbol{y}) = \sum_{\vec{x}\in\mathcal{R}_x}\sum_{\vec{y}\in\mathcal{R}_y}\prod_{d=1}^{D} k_d(x_d,y_d) \tag{3.82}$$

  Strictly speaking, a convolution kernel is a general way of constructing kernels using a set of kernels $k_d : \mathcal{X}_d \times \mathcal{X}_d \to \mathbb{R}$ for structured objects $\boldsymbol{x} \in \mathcal{X}$, which can be composed by parts $\vec{x} := (x_1,\ldots,x_D) \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_D$. The set R_x contains all possible decompositions of $\boldsymbol{x}$ defined through a specific relation [Haussler 1999]. For example, for the decomposition $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_D$, the relation $R(\vec{x},\boldsymbol{x}) := \{(\vec{x},\boldsymbol{x}) : \vec{x} = \boldsymbol{x}\}$ and $k_d(x_d,y_d) := \exp(-\frac{|x_d-y_d|^2}{2\sigma^2})$, the associated convolution kernel equals the RBF-kernel. Also the **ANOVA-kernel** [Wahba 1990] and the **Gibbs-kernel** [Haussler 1999] are special cases of convolution kernels.

Among the explicitly mentioned kernels, the closure properties of the set of feasible kernels enables to construct kernels $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ from following operations [Schölkopf 2002]:

$$k(\boldsymbol{x},\boldsymbol{y}) = \alpha_1 k_1(\boldsymbol{x},\boldsymbol{y}) + \alpha_2 k_2(\boldsymbol{x},\boldsymbol{y}) \quad \text{(linearity)} \tag{3.83}$$

$$k(\boldsymbol{x},\boldsymbol{y}) = k_1(\boldsymbol{x},\boldsymbol{y}) \cdot k_2(\boldsymbol{x},\boldsymbol{y}) \quad \text{(pointwise product)} \tag{3.84}$$

$$k(\boldsymbol{x},\boldsymbol{y}) = f(\boldsymbol{x})k(\boldsymbol{x},\boldsymbol{y})f(\boldsymbol{y}),\ f\text{positive} \quad \text{(conformal transform)}. \tag{3.85}$$

Summary

In this chapter, we introduced statistical learning theory for the purpose of motivating the idea of a maximum margin classification. In comparison to the empirical risk minimization, the structural risk minimization (SRM) principle proposes to select among all separating hyperplanes the one that maximizes the margin. The maximum margin hyperplane has the lowest bound on the VC-Dimension (Lemma (3.5.1)) and thus minimizes the bound on the actual risk. We discussed that the SVM approximates the idea of SRM, and we derived the SVM algorithms. Although the SVM learns a linear decision function it can be generalized to nonlinear ones using kernel functions. Kernel functions induce a feature map into a high-dimensional feature space (Hilbert space), if and only if the Mercer's condition is satisfied. We reviewed two concepts for identifying kernels with an associated Hilbert space, and functions known to be kernels are presented too.

In the next chapter we discuss that SVMs have a drawback albeit their nice properties. This drawback is due to the severe restriction of SVMs to the use of kernel functions satisfying the Mercer's condition, only. For example it limits the application of SVMs in cases where similarity functions are the appropriate choice. However, we show in this thesis that it is possible to overcome the restriction to kernel functions while maintaining the preferable maximum margin concept not just theoretically but also practically.

Chapter 4

The Shortcoming of SVMs and Ways to Resolve It

Contents

Chapter (3) introduced the fundamentals of statistical learning theory to explain the benefits of SVMs compared to other pattern recognition approaches (Chapter (2)). It was concluded, that maximum margin hyperplanes are crucial for a good generalization performance. In Section (3.6) we demonstrated how SVMs algorithmically realize a maximum margin hyperplane while Section (3.6.3) discussed how linear SVMs can be generalized to nonlinear SVMs via the "kernel-trick". Although kernel functions permit the efficient learning of maximum margin hyperplanes in spaces of very high dimensions, they impose severe restrictions on the decision functions that are possible to implement with SVM classifiers. It was shown that a solution of the SVM problem is build up of a finite expansion of inner products between given data points lying within some feature space (Hilbert Space). Kernels must satisfy specific mathematical properties, namely the Mercer's condition (Theorem (3.6.1)), before they represent the geometrical structure of a feature space. Mercer's condition, however, has only be proven for a handful of kernels, which means that SVMs are not suitable for handling classification problems that are not appropriately solvable with kernels such as those presented in Section (3.6.5) or derivatives of them. Vice versa, generalization properties are not guaranteed if using decision functions based on other types of functions in SVMs. At first glance, one could argue that the presented kernels, albeit few, cover most cases that occur in practice. However, in practice,

there is a need of using decision functions which e.g. facilitate a priori knowledge of the problem or represent a particular structure of the data that has to be classified. Particular knowledge of a classification problem is based on the experience of the classifier's designer, or is dedicated by the preprocessing (feature extraction) of the recognition system or other design goals. To make matters worse, a priori knowledge is often formulated in a very intuitive way. For example, in *Optical Character Recognition* (OCR) it is known that the data is subject to any transformations, like shifts or rotations. Clearly, at best the decision function should be invariant with respect to such transformations. An intuitive way to reach this goal is to classify samples based on invariant similarity measures comparing two samples. Unfortunately, many similarity measures suitable for such a task are not usable in SVMs. In general it is mostly not possible to translate a desired similarity to a kernel function, nor via such general construction methods like the convolution kernel presented in Section (3.6.5). On the other hand, if using arbitrary functions in the SVM algorithm violating the Mercer's condition, then one might not expect improved generalization performance justified due to the maximum margin philosophy. Moreover, the SVM algorithm does not guarantee to converge to a global solution of the SVM optimization problem. So, strictly speaking, the learnt classifier must not be called SVM. Despite this, many researcher had tried to use non-kernel functions in the SVM algorithm in the past. Examples of non-kernel functions used in SVMs are given in Section (4.1). These examples suggest that there is a practical need of a larger variety of decision functions suitable for an application, and in particular in a maximum margin concept due to the good generalization performance compared to other approaches. This is the reason, why recently new methods have been proposed to overcome the limitations of SVMs (Sec. (4.2)). In particular, the approach summarized in Section (4.2.1) can be viewed as an extension of the classical SVM to a subset of distance functions. A more general concept is the mapping of the data space and the decision function space simultaneously into general spaces endowed with suitable mathematical properties. This enables to define separating hyperplanes and a margin for larger function classes than accessible to SVMs. Such a particular embedding is presented in Section (4.2.2). We also review in Section (4.2.3) a related approach for distance based classification that uses a subset of the space of so-called Lipschitz functions. By now, all these methods have in common to be related to the most general framework focused in the next Chapter (5) because of its importance for the present thesis.

4.1 Pseudo-Kernels used in SVMs

We emphasize again, the interpretation of a SVM as a maximum margin separating hyperplane in a Hilbert space holds, if and only if the associated inner product can be represented by a function, called kernel (Sec. (3.6.5), satisfying the Mercer's condition (Theorem (3.6.1)).

Despite this, a misleading misnomer can be observed in literature: functions are also called kernels and are actually used in SVMs, although they do not satisfy the Mercer's condition. Therefore, we use in the following the term *pseudo-kernels* to distinguish explicitly kernels from non-kernel functions.

In fact, if using pseudo-kernels in the SVM algorithms the resulting classifier is not a SVM and justification by the maximum margin concept as concluded from statistical learning theory (Chapter (3)) does not apply. Moreover, a global solution of the quadratic SVM objective (3.60) is also not guaranteed due to the indefiniteness of the involved kernel matrix. Nevertheless, the many applications, where pseudo-kernels have been used in SVMs, suggest the necessity of more flexibility in choosing suitable decision functions, but without a loss of generalization performances expected from maximum margin classifiers. In our opinion, this goal can only be reached satisfyingly through the development of new classification methods generalizing SVMs. This can also be concluded by reported results obtained in applications of pseudo-kernels in SVMs. For example, in [Haasdonk 2002] so-called **tangent distance-"kernels"** are used in SVMs for solving an OCR application. The idea behind *tangent-distances* is to approximate locally at the training points the nonlinear manifold of possible transformations the data is subject to by a linear space (*tangent-space*). Then the similarity of a test point and a training point is defined as the Euclidean distance between the test point and its projection to the tangent-space associated to the training point. In this way, tangent-distances incorporate a priori knowledge about local invariances. Although the tangent-distance $d_{tan}(\boldsymbol{x}, \boldsymbol{y})$ is not a metric, [Haasdonk 2002] used it regardless in the RBF-kernel via substitution, i.e. $k_{tan}(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\gamma d_{tan}(\boldsymbol{x}, \boldsymbol{y}))$, for classifying the USPS data (cf. Sec. (2.4.4)). Because d_{tan} is not a metric it follows k_{tan} is not a kernel function. However, this approach resulted in a test error of 3.4% compared to 4.0% using SVMs with standard RBF-kernel. On the other hand, in essence a very similar approach to incorporate local invariances is to generate Virtual Support Vectors (VSV) by applying transformations like shifts and rotations [Schölkopf 1996] to the set of Support Vectors found by a standard RBF-SVM. Then a new SVM is trained on the set of VSVs in order to get a classifier that is invariant to such transformations. This method reached a test error of 3.2% using the USPS dataset, which is a better result than obtained by using tangent-distances indirectly in RBF-SVMs. Together

with the superior result of 2.4% test error applying a Bayesian approach using also tangent-distances indirectly in RBF-kernel-densities [Keysers 2000] let us conclude that SVMs do not completely facilitate the incorporation of transformation invariances when using tangent-distances in RBF-kernels.

Another very prominent example of a pseudo-kernel used in SVMs is the function

$$f(\boldsymbol{x}, \boldsymbol{y}) := \tanh\left(\kappa\langle \boldsymbol{x}, \boldsymbol{y}\rangle + \vartheta\right) \quad (\textbf{sigmoid "pseudo-kernel"}). \tag{4.1}$$

The sigmoid pseudo-kernel is not positive definite for most of its parameters $(\kappa, \vartheta) \in \mathbb{R} \times \mathbb{R}$ and even of the data itself. A few settings for which the sigmoid function satisfies the Mercer's condition were observed just empirically in [Vapnik 1999]. Hence in general the sigmoid function is not a kernel function. Nevertheless, this pseudo-kernel is very popular for SVMs because of its relation to the activation function of neural networks (Sec. (2.4.1)). It was used for example in handwritten digit recognition [Vapnik 1999].

Further examples are given by the

- **dynamic time warping-"pseudo-kernel"** e.g. [Lei 2007]
- **jittering-"pseudo-kernel"** [DeCoste 2002], [Bahlmann 2002]
- **Kullback-Leibler Divergence-"pseudo-kernel"** [Moreno 2003]

which are frequently used in SVMs for image recognition or for classification of vector sequences of different length.

Mostly, pseudo-kernels are constructed by simply substituting the euclidean metric $\|\boldsymbol{x} - \boldsymbol{y}\|_2$ that is used in the RBF-kernel function (3.77) with a suitable similarity function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, $d(\boldsymbol{x}, \boldsymbol{y}) \geq 0$, $d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}, \boldsymbol{x})\ \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$, i.e.

$$k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\gamma d(\boldsymbol{x}, \boldsymbol{y})^2\right), \gamma > 0. \tag{4.2}$$

The RBF-kernel can be shown to satisfy the Mercer's condition as long as the euclidean metric $\|\boldsymbol{x} - \boldsymbol{y}\|_2$ is substituted with a metric function, i.e. it also holds $d(\boldsymbol{x}, \boldsymbol{y}) = 0 \Leftrightarrow \boldsymbol{x} = \boldsymbol{y}$ as well as the triangle inequality (App. (A.2.1)). But this condition is mostly not true for an arbitrary similarity function.

On the other hand, using similarity functions to build up a classification function is particularly desirable in cases in which prior knowledge can be encoded in terms of a measure of similarity (for example the encoding of local invariances via tangent-distances). In biological informatics, often such similarity functions are defined to emphasize specific gene expressions. For example in [Selinski 2005], similarities for clustering particular gene expressions are investigated for the purpose of analyzing the influence and interaction of single nucleotide polymorphic (SNP) loci and exogenous risk factors

to develop breast cancer. In [Cha 2006], different distance measures between binary vectors are investigated on biometric and handwritten digit data.

Likewise, for the classification of vector sequences of different length the use of similarities is tempting. For example in [Moreno 2003], the Kullback-Leibler divergence is used to handle the sequence classification problem by applying the symmetric divergence

$$d\left(p(\boldsymbol{x}|\boldsymbol{\vartheta}_i), p(\boldsymbol{x}|\boldsymbol{\vartheta}_j)\right) := \int p(\boldsymbol{x}|\boldsymbol{\vartheta}_i) \ln\left(\frac{p(\boldsymbol{x}|\boldsymbol{\vartheta}_i)}{p(\boldsymbol{x}|\boldsymbol{\vartheta}_j)}\right) + p(\boldsymbol{x}|\boldsymbol{\vartheta}_j) \ln\left(\frac{p(\boldsymbol{x}|\boldsymbol{\vartheta}_j)}{p(\boldsymbol{x}|\boldsymbol{\vartheta}_i)}\right) d\boldsymbol{x} \tag{4.3}$$

to the RBF-kernel function yielding

$$k(p(\boldsymbol{x}|\boldsymbol{\vartheta}_i), p(\boldsymbol{x}|\boldsymbol{\vartheta}_j)) := \exp\left(-\gamma d\left(p(\boldsymbol{x}|\boldsymbol{\vartheta}_i), p(\boldsymbol{x}|\boldsymbol{\vartheta}_j)\right)^2\right). \tag{4.4}$$

Thus, each data point $\boldsymbol{x} \in \mathcal{X}$ can be viewed to be preprocessed by a transformation to the space spanned by the likelihoods of generating the example from an associated generative model. The symmetric Kullback-Leibler divergence (4.3) is not a metric and therefore, if used in the SVM quadratic the resulting classifier is again not a maximum margin classifier nor a SVM. However, the use of information theoretic measures of similarity between generative models is a promising way to handle vector sequences. Additionally, prior knowledge about the problem at hand could also be incorporated using (4.3) in a static maximum margin classifier via prior probabilities. An admissible application of the Kullback-Leibler distance would also extend the investigations (Chapter (2.4.3)) in replacing the acoustic models used in speech recognition with static SVM classifiers.

A further motivation for similarity-based classification is the use of distances from cluster algorithms. Often, cluster algorithms are used to preprocess data in order to subsequently learn a classification. A natural way would be to use the same distance functions to construct classifiers and to facilitate the same geometrical structure. In the past, many similarities have been developed to solve highly data-dependent problems of clustering. Unfortunately, these similarities can not be used directly in a maximum margin classifier like SVMs, because they do not satisfy the metric property.

In the next sections, we review related work concerning to overcome the restrictions of SVMs and to admissibly use similarity functions.

4.2 Distance based Maximum Margin Classification

In case of non metric functions, recently in [Chen 2009] some heuristics, called *spectrum clip*, *spectrum flip* or *spectrum shift*, have been experimentally stud-

ied and compared using different similarity-based classifiers. The proposed heuristics are based on the idea to modify the eigenvalues of a similarity matrix $\mathbf{D} := (d(\boldsymbol{x}_i, \boldsymbol{x}_j))_{i,j} \in \mathbb{R}^{N \times N}$ (spectrum of $\mathbf{D}$) such that it becomes positive definite, i.e. it becomes a kernel matrix valid for SVMs. Improvements for SVMs in some situations are reported in [Chen 2009], using similarities in this way. The results show, that similarity based decision functions can perform well, if chosen appropriately to the data. On the other hand, the spectrum modifications distort the original similarity such that it is difficult to select the similarity based on prior knowledge in advance or to interpret the results. Thus, to use similarity functions without any modification directly in a maximum margin classifier, one has to generalize the idea of the maximum margin separating hyperplane to other spaces than Hilbert spaces.

4.2.1 Hilbert Spaces induced by Semi-Metric Spaces

In the past, feature spaces more general than Hilbert spaces had attracted little attention from the machine learning community. The reason might be, that the few available publications addressing this issue are almost completely of theoretical value from which it seems difficult to derive usable algorithms.

In this context, the work of [Hein 2004] has to be noted, in which two different *injective* and *structure preserving* mappings (also called *embeddings*) are proposed.

Given a (semi-)metric $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, it is shown in [Hein 2004] that any *(semi-)metric space* $\mathcal{M} := (\mathcal{X}, d)$ (cf. App. A.2.1) can be embedded (not uniquely) in a corresponding Hilbert space $\mathcal{H}$ as long as $-d^2$ is *Conditional Positive Definite* (CPD). Conditional positive definiteness of $-d^2$ is equivalent with the requirement that $\sum_{n,m=1}^{N} c_n c_m d^2(\boldsymbol{x}_n, \boldsymbol{x}_m) \leq 0$ holds for all $N \in \mathbb{N}, c_n, c_m \in \mathbb{R}$ with $\sum_{n=1}^{N} c_n = 0$ (cf. [Schölkopf 2002][pp. 48-50]).

If CPD is satisfied, a (semi-)metric space $\mathcal{M}$ can be (isometrically[1]) embedded into a *Reproducing Kernel Hilbert Space* (RKHS) (Sec. (3.6.4)) via the feature map

$$\boldsymbol{x} \mapsto \Phi_{\boldsymbol{x}}^{d}(\cdot) := k(\boldsymbol{x}, \cdot) = -\frac{1}{2} d^2(\boldsymbol{x}, \cdot) + \frac{1}{2} d^2(\boldsymbol{x}, \boldsymbol{x}_0) + \frac{1}{2} d^2(\boldsymbol{x}_0, \cdot) \tag{4.5}$$

with arbitrary chosen $\boldsymbol{x}_0 \in \mathcal{X}$ in advance. Vice versa, it has been proved, that any CPD-kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ induces a (semi-)metric

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{k(\boldsymbol{x}, \boldsymbol{x}) + k(\boldsymbol{y}, \boldsymbol{y}) - 2k(\boldsymbol{x}, \boldsymbol{y})} \tag{4.6}$$

on $\mathcal{X}$, such that $-d^2$ is also CPD. One can show a optimal SVM decision function using $\Phi_{\boldsymbol{x}}^{d}$ is spanned by functions $-\frac{1}{2} d^2(\boldsymbol{x}, \boldsymbol{y})$.

[1] *Isometrically* means that the embedding is *bijective* and (semi-)metric preserving.

In addition it is important to note, the use of semi-metrics in SVMs assumes implicitly some kind of global invariance in the data [Hein 2004]. This is often not the case, or if there is some kind of global invariance it is difficult to construct a semi-metric that reflects it. For example imagine the case, in which there are points $\boldsymbol{x} \neq \boldsymbol{y} \in \mathcal{M}$ such that $d(\boldsymbol{x}, \boldsymbol{y}) = 0$ holds. Obviously, these points are not separable by hyperplanes defined according to the semi-metric d. If this situation is not consistent with some invariance in the data, such a SVM- or any other distance based classifier is doomed to failure.

4.2.2 Kuratowski Embedding

However, the (semi-)metric spaces that can be embedded into Hilbert spaces usable in SVMs are limited to a subspace of all (semi-)metric spaces, because of the constraint that $-d^2$ has to be CPD.

Therefore, in [Hein 2004] also a more general embedding is proposed and analyzed. The proposed embedding is called *Kuratowski embedding* and it is an isometric mapping of a metric space $\mathcal{M} = (\mathcal{X}, d)$ to a *Banach space* (cf. App. A.2.2) $\mathcal{B} := (\bar{\mathcal{D}}, \|\cdot\|_\infty) \subset (\mathcal{C}(\mathcal{X}, \mathbb{R}), \|\cdot\|_\infty)$ with $\mathcal{D} := \operatorname{span}\{\Phi_{\boldsymbol{x}} : \boldsymbol{x} \in X\}, \boldsymbol{x} \in \mathcal{X} \mapsto \Phi_{\boldsymbol{x}} := d(\boldsymbol{x}, \cdot) - d(\boldsymbol{x}_0, \cdot)$ That means, $\mathcal{B}$ is a subset of the space $\mathcal{C}(\mathcal{X}, \mathbb{R})$ of continuous functions defined on a compact $\mathcal{X}$ endowed with the supremum norm $\|f\|_\infty := \sup_{\boldsymbol{x} \in \mathcal{X}} |f(\boldsymbol{x})|$.

Simultaneously, it is shown [Hein 2004] that the space of all linear functionals $\mathcal{B}'$ defined on $\mathcal{B}$ is isometrically isomorphic to the Banach space $(\bar{\mathcal{E}}, \|\cdot\|_\mathcal{E})$ with $\mathcal{E} := \operatorname{span}\{\Psi_{\boldsymbol{x}} : \boldsymbol{x} \in \mathcal{X}\}, \boldsymbol{x} \in \mathcal{X} \mapsto \Psi_{\boldsymbol{x}} := d(\cdot, \boldsymbol{x}) - d(\boldsymbol{x}_0, \boldsymbol{x})$ ($\boldsymbol{x}_0 \in \mathcal{X}$ arbitrary chosen) and $\|e\|_\mathcal{E} := \inf\left\{\sum_{i \in I} |\beta_i| : e = \sum_{i \in I} \beta_i \Psi_{\boldsymbol{x}_i}, \boldsymbol{x}_i \in \mathcal{X}, |I| < \infty\right\}$.

This observation enables us to determine a margin and a separating hyperplane in $\mathcal{B}'$, which in turn can be expressed in the Banach space $(\bar{\mathcal{E}}, \|\cdot\|_\mathcal{E})$ resulting in the following maximum margin classification algorithm:

Algorithm 4.2.1 Max. Margin Classifier using $(\bar{\mathcal{E}}, \|\cdot\|_\mathcal{E})$ (Hard-Margin) [Hein 2004]

Require: Training set $\emptyset \neq \mathcal{O}_l \subset \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, $\boldsymbol{x}_0 \in \mathcal{X}$, a metric d

Ensure: $\exists (\boldsymbol{x}_j, y_j), (\boldsymbol{x}_i, y_i) \in \mathcal{O}_l \ : \ y_j = -1 \wedge y_i = 1$, both classes are linearly separable in $\mathcal{B}$

$$\inf_{e \in \mathcal{E}, b \in \mathbb{R}} \|e\|_\mathcal{E} = \inf_{m \in \mathbb{N}, (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m) \in \mathcal{X}^m, b \in \mathbb{R}, (\beta_1, \ldots, \beta_m) \in \mathbb{R}^m} \sum_{i=1}^m |\beta_i| \tag{4.7}$$

$$\text{s.t.} \ \ \forall (\boldsymbol{x}_j, y_j) \in \mathcal{O}_l : y_j \left(\sum_{i=1}^m \beta_i \left(d(\boldsymbol{x}_j, \boldsymbol{x}_i) - d(\boldsymbol{x}_0, \boldsymbol{x}_i)\right) + b\right) \geq 1 \tag{4.8}$$

Algorithm (4.2.1) can be derived, because the geometry of maximum margin hyperplanes in Hilbert spaces carries over to Banach spaces [Zhou 2002]. For this purpose consider two separable data sets $\mathcal{H}_1, \mathcal{H}_2$ of a Hilbert space $\mathcal{H}$. Then, the standard optimization formulation for finding the optimal maximum margin hyperplane as shown in Section (3.6.1) can be equivalently restated as:

$$\min_{w \in \mathcal{H}',\ b \in \mathbb{R}} \|w\| \tag{4.9}$$

$$\text{s.t.} \quad \forall \mathbf{\Phi}_j \in \mathcal{H}_1 \cup \mathcal{H}_2 : \ (w(\mathbf{\Phi}_j) + b) \cdot y_j \geq 1 \quad , \tag{4.10}$$

where $\mathcal{H}'$ denotes the space of all linear functionals $w : \mathcal{H} \to \mathbb{R}$, and $\|\cdot\|$ is the usual operator norm. One can show [Zhou 2002], that this formulation holds also for finding optimal maximum margin hyperplanes in Banach spaces $\mathcal{B}$ assuming two separable training sets $\mathcal{B}_1, \mathcal{B}_2 \subset \mathcal{B}$. Thus, in particular for $\mathcal{B}'$ we get

$$\min_{w \in \bar{\mathcal{D}}',\ b \in \mathbb{R}} \|w\| \tag{4.11}$$

$$\text{s.t.} \quad \forall \mathbf{\Phi}_{\boldsymbol{x}_j} \in \mathcal{B}_1 \cup \mathcal{B}_2 : \ (w(\mathbf{\Phi}_{\boldsymbol{x}_j}) + b) \cdot y_j \geq 1 \quad . \tag{4.12}$$

Using the isomorphism of $\bar{\mathcal{D}}'$ and $\bar{\mathcal{E}}$ yields

$$\min_{e \in \bar{\mathcal{E}},\ b \in \mathbb{R}} \|e\|_{\mathcal{E}} \tag{4.13}$$

$$\text{s.t.} \quad \forall \mathbf{\Phi}_{\boldsymbol{x}_j} \in \mathcal{B}_1 \cup \mathcal{B}_2 : \ (\textstyle\sum_{i \in I} \beta_i \Psi_{\boldsymbol{x}_i}(\mathbf{\Phi}_{\boldsymbol{x}_j}) + b) \cdot y_j \geq 1 \quad . \tag{4.14}$$

By the continuity of the norm $\|\cdot\|_{\mathcal{E}}$ and because $\mathcal{E}$ is dense in $\bar{\mathcal{E}}$ the minimum on $\bar{\mathcal{E}}$ can be replaced by an infimum on $\mathcal{E}$. Together with the definition of $\Psi_{\boldsymbol{x}_i}$ and $\mathbf{\Phi}_{\boldsymbol{x}_j}$ it finally results Algorithm (4.2.1).

Unfortunately, the proposed algorithm is computational impractical, because of the optimization over the sets $\mathcal{X}^m$ $\forall m \in \mathbb{N}$. Thus, to make the problem tractable, the space $\mathcal{E}$ must be restricted to some finite dimensional subspace. The simplest way is to choose some finite subset $\mathcal{V} \subset \mathcal{X}, |\mathcal{V}| = m$ that spans $\mathcal{E}$. For example (cf. [Hein 2004]) using the setting $\mathcal{V} = \mathcal{O}_m$ let us recover an algorithm already proposed by [Graepel 1999], which is a standard linear programming problem:

Algorithm 4.2.2 LP Machine by [Graepel 1999] (Hard-Margin)

Require: Training set $\emptyset \neq \mathcal{O}_m \subset \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, a metric d

Ensure: $\exists (\boldsymbol{x}_j, y_j), (\boldsymbol{x}_i, y_i) \in \mathcal{O}_m \ : \ y_j = -1 \wedge y_i = 1$, both classes are linearly separable in $\mathcal{B}$

$$\inf_{(\beta_1,\ldots,\beta_m)\in\mathbb{R}^m, c\in\mathbb{R}} \sum_{i=1}^{m} |\beta_i| \tag{4.15}$$

$$\text{s.t.} \ \ \forall (\boldsymbol{x}_j, y_j) \in \mathcal{O}_m : y_j \left(\sum_{i=1}^{m} \beta_i d(\boldsymbol{x}_j, \boldsymbol{x}_i) + c\right) \geq 1 \tag{4.16}$$

Note, Algorithm (4.2.1) and Algorithm (4.2.2) can also be formulated in the case of nonlinearly separable data in $\mathcal{B}$ by e.g. introducing slack variables (cf. Sec. (3.6.2)).

Unlike the SVM, there holds no Representer Theorem [Kimeldorf 1971] for Algorithm (4.2.1) that means a solution is not guaranteed to be expressible in form of the training data only. Nevertheless, from a theoretical point of view the work of [Hein 2004] is of particular value because it demonstrates the use of the powerful concept of duality and isometric isomorphisms of function spaces in order to carry over maximum margin hyperplanes from Hilbert spaces induced by a kernel to more general spaces, like Banach spaces. The derivation of a Representer Theorem in the setting of learning in metric spaces is the focus of the work of [Minh 2004], which we summarize in the next section due to a surprising relation to our preceding discussion and the next Chapter (5).

4.2.3 Subalgebra of Lipschitz Functions

In [Minh 2004] a general framework is derived to obtain a *Representer Theorem* for the minimization of regularized loss functionals of the form

$$R_l(f) + \gamma \Omega(f), \ \ f \in \mathcal{F}, \gamma \geq 0, \tag{4.17}$$

where Ω denotes some regularization operator satisfying some mild conditions and R_l is the empirical risk (cf. Eq. (3.2)) with a convex lower semi-continuous loss function L_f. Considering an operator $A : \mathcal{F} \to \mathbb{R}^l$ with $A(f) := (f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_l))$ evaluating f at the training points $\boldsymbol{x}_i \in \mathcal{X}$, it is shown that a minimizer (if exists) of (4.17) always lies in a finite dimensional space $\mathcal{F}_l$ with dimension at most l. The space $\mathcal{F}_l$ is a linear subspace of $\mathcal{F}$ such that the decomposition $\mathcal{F} = \mathcal{F}_l \oplus null(A)$ holds for the null space $null(A) := \{f \in \mathcal{F} \ : \ A(f) = 0\}$. That means, if we are able to decompose

the considered decision function space $\mathcal{F}$ in this way and explicitly express $\mathcal{F}_l$ in terms of the training data, we can explicitly characterize the solution.

For example, it is well-known that a soft-margin SVM in a Hilbert space $\mathcal{H}$ is equivalently reformulated in a regularized loss functional form (cf. e.g. [Schölkopf 2002]) with $R_l(f) := 1/l \cdot \sum_{i=1}^{l} \max\{0, 1 - y_i f(\boldsymbol{x}_i)\}$ (*Soft-Margin Loss* [Bennett 1992]) and $\Omega(f) := \langle f, f \rangle_{\mathcal{H}}$ (cf. Sec. 3.6.4). Now, by setting $\mathcal{F}$ to be a Reproducing Kernel Hilbert Space $\mathcal{R}$ (cf. Sec. 3.6.4) it follows from the reproducing property $f(\boldsymbol{x}) = \langle f, k(\boldsymbol{x}, \cdot) \rangle$ that $null(A) = span\{k(\boldsymbol{x}_1, \cdot), \ldots, k(\boldsymbol{x}_l, \cdot)\}^{\perp}$. Using the unique *orthogonal decomposition* well-known from functional analysis we get $\mathcal{F} = \mathcal{F}_l \oplus span\{k(\boldsymbol{x}_1, \cdot), \ldots, k(\boldsymbol{x}_l, \cdot)\}^{\perp}$ with $\mathcal{F}_l = span\{k(\boldsymbol{x}_1, \cdot), \ldots, k(\boldsymbol{x}_l, \cdot)\}$. Thus, a solution of the SVM problem is represented by a finite expansion of kernel functions evaluated at the training points, although the decision function space $\mathcal{F}$ might be of infinite dimension.

In the context of learning in general spaces than Hilbert spaces, [Minh 2004] also derived a Representer Theorem for learning in compact metric spaces $\mathcal{M} = (\mathcal{X}, d)$. For this purpose the regularized loss functional (4.17) is minimized over a subalgebra[2] $\mathcal{A}$ of $\mathcal{C}(\mathcal{X}, \mathbb{R})$ generated by the family $\{1, d(x, \cdot)\}_{x \in \mathcal{X}}$. Remarkably, in this case, a minimizer (if one exists) always admits a finite expansion $f^* = \sum_{i=1}^{l} c_i^* M_i,\ M_i := \prod_{j \neq i} d(\boldsymbol{x}_j, \cdot)$ in terms of the training data $\boldsymbol{x}_j$. Using this result, it follows the most important case with respect to our discussion of maximum margin classifiers overcoming the restriction to kernels [Minh 2004]: minimizing the regularized functional (4.17) over $\mathcal{A}$ using $R_l(f) := 1/l \cdot \sum_{i=1}^{l} \Theta(1 - y_i f(\boldsymbol{x}_i))$ and $\Omega(f) := \sum_i |c_i|$ yields the unique solution

$$f^* = \sum_{i=1}^{l} c_i^* \prod_{j \neq i} d(\boldsymbol{x}_j, \cdot) \tag{4.18}$$

with $c_i^* = \frac{y_i}{M_i}$. The function Θ denotes the *Heaviside*-function. The loss $\Theta(1 - y_n f(\boldsymbol{x}_n))$ of the empirical risk $R_l(f)$ is equivalent with the hard-margin loss used in the standard SVM formulation (cf. e.g. [Schölkopf 2002]), but the regularizer is different and equals the coefficient regularizer also used in the *Lasso regression* method (e.g. [Hastie 2001]). Although it will become clear in the next chapter, by now, $\mathcal{A}$ consists of Lipschitz continuous functions (Def. 5.1.1) and the minimization of the regularized loss functional restated in Algorithm (4.2.3) can be shown to be an approximation to a maximum margin algorithm.

[2] *algebra*: vector space additionally endowed with a bilinear operator defining multiplication

Algorithm 4.2.3 LP Machine by [Minh 2004] (Hard-Margin)

Require: Training set $\emptyset \neq \mathcal{O}_m \subset \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, a metric d

Ensure: $\exists(\boldsymbol{x}_j, y_j), (\boldsymbol{x}_i, y_i) \in \mathcal{O}_m \ : \ y_j = -1 \wedge y_i = 1$, both classes are linearly separable in $\mathcal{A}$

$$\inf_{(\beta_1,\ldots,\beta_m)\in\mathbb{R}^m} \sum_{i=1}^{m} |\beta_i| \tag{4.19}$$

$$\text{s.t.} \ \ \forall(\boldsymbol{x}_j, y_j) \in \mathcal{O}_m : y_j \left(\sum_{i=1}^{m} \beta_i \prod_{k\neq i} d(\boldsymbol{x}_k, \boldsymbol{x}_j) \right) \geq 1 \tag{4.20}$$

Note, Algorithm (4.2.3) is very similar to the LP Machine by [Graepel 1999] (Alg. 4.2.2) but in contrast another decision function class is used. However, as we will see in the next chapter, both algorithms share the same drawback that a solution leads just to a crude approximation to a maximum margin classifier. Even worse the decision functions are again restricted to a particular form build up of basis functions satisfying the metric properties (cf. App. (A.2.1)). Thus, these algorithms are not suitable in a similarity based setting nor they can be regarded as (approximate) generalizations of the SVM algorithm, because the use of kernels is denied completely.

Summary

In this chapter, we discussed the drawback of SVMs due to the restriction to kernel functions. We emphasized that the interpretation of SVMs as maximum margin separating hyperplanes in some high-dimensional Hilbert space holds if and only if the used functions satisfy the Mercer's condition. It was demonstrated that kernels cover not all cases that occur in practice and sample applications were presented indicating the need of more flexibility in choosing appropriate decision functions in a maximum margin concept. This need and the lack of suitable algorithms had forced many researchers to use also not valid kernels (pseudo-kernels) in SVMs anyway. In this case there is no reasoning to expect a good generalization performance justified by statistical learning theory. Strictly speaking, if using pseudo-kernels in the SVM algorithms then the learnt classifier is not a SVM. Moreover, an example for similarity based classification was given that let us conclude that SVMs are not able to facilitate the advantage of a priori knowledge encoded in pseudo-kernels.

However, in order to extend the application of SVMs, the kernel approach was generalized to a subset of similarity functions that have to be conditional

positive definite (Sec. (4.2.1)). Unfortunately, it is very difficult to develop similarity functions for a specific application that encode prior knowledge or encode some invariance in the data while satisfying the CPD-constraint.

Therefore, a much more general approach for generalizing the maximum margin concept and for overcoming the restriction to kernels was proposed in [Hein 2004]. Therein, the concept of duality and isometric isomorphisms of function spaces is applied to translate the optimal maximum margin hyperplane to a special function space that is more general than a Hilbert space. In this way decision functions can be implemented build up of metric functions (Sec. (4.2.2)). This is unfortunately again a severe restriction in particular in the context of similarity based classification, because many usful similarities do not satisfy the properties of a metric. A further drawback is, that the use of kernels or other basis functions the decision function is built up is denied completely. Moreover, the proposed method results in an Algorithm (4.2.1) that is impractical in general and thus it can only be solved as a crude approximation to a maximum margin algorithm (Alg. (4.2.2)). Another approach proposed by [Minh 2004] resulted in a similar approximation (Alg. (4.2.3)) also permitting solely the use of metric functions in a specific form of decision function (Sec. (4.2.3)) and also denying any other kind of basis function.

Below the line, the related work summarized in this chapter, although of theoretical value, do not provide practical learning algorithms that reached the sophisticated goal of overcoming the restrictions to kernels in a maximum margin concept generalizing SVMs without imposing new severe restrictions or without being very crude approximation to a maximum margin classifier.

Regardless, in the next chapter, we show that it is indeed possible to derive maximum margin algorithms that facilitate a very huge decision function class. But in contrast, only mild restrictions have to be imposed on the decision functions and the algorithms are manageable without crude approximations. The development of these new maximum margin algorithms are theoretically based on an isometric isomorphism generalizing SVMs as well as subsuming other learning approaches.

Chapter 5

Lipschitz Classifier

Contents

The few published articles that aim on a generalization of the maximum margin concept implemented in SVMs to more general spaces than Hilbert spaces, are presented in the preceding chapter.

In the following part of the present thesis, the theory behind a further generalization, called *Lipschitz embedding*, is introduced (Sec. 5.1). Using the Lipschitz embedding, a maximum margin classifier results that is presented in Section (5.1.2). The new classifier subsumes the SVM, 1st-Nearest-Neighbor classifier and the algorithms presented in the preceding chapter.

After the theory is introduced, together with some mathematical terms, we go step-by-step into the details on solving the new maximum margin classifier. The derived algorithm maintains most of its generality while it is implementable without some crude approximations that have to be considered in advance (Sec. 5.2).

It turns out, that a formulation of the classifier can be derived, that enables the use of functions from a very rich function class in order to learn more general decision functions in a maximum margin concept than it is the case for SVMs.

5.1 Embedding of Lipschitz Functions

As pointed out in the preceding chapter, the nature of maximum margin classification requires mathematical definitions of distances between points and separating hyperplanes. In case of the SVM a distance d (actually a metric) is induced by a inner product of some appropriate chosen Hilbert space represented by an associated kernel function.

Although many metric spaces $\mathcal{M} := (\mathcal{X}, d)$ (cf. App. A.2.1) can be imagined, only a subset of all metric spaces inherit the structure of an associated Hilbert space $\mathcal{H}$ (cf. App. A.2.3). This is also true for Banach spaces $\mathcal{B}$ (cf. App. A.2.2). Obviously, any Hilbert space $\mathcal{H}$ passes its structure to a Banach space $\mathcal{B}$, and any Banach space $\mathcal{B}$ passes its structure to a metric space $\mathcal{M}$, but the reverse is not always true. For example, the euclidean inner product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_2, \boldsymbol{x}, \boldsymbol{y} \in \mathcal{H}$ induces the euclidean norm $\|\boldsymbol{x}\|_2 := \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle_2}, \boldsymbol{x} \in \mathcal{B}$, and the metric $d(\boldsymbol{x}, \boldsymbol{y}) := \|\boldsymbol{x} - \boldsymbol{y}\|_2 = \sqrt{\langle \boldsymbol{x} - \boldsymbol{y}, \boldsymbol{x} - \boldsymbol{y} \rangle_2}, \boldsymbol{x}, \boldsymbol{y} \in \mathcal{M}$, and vice versa. In contrast, the metric

$$d(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 0 \text{ if } \boldsymbol{x} = \boldsymbol{y} \\ 1 \text{ else} \end{cases} \tag{5.1}$$

induces no corresponding norm, respectively inner product.

However, in the very general setting of metric spaces $\mathcal{M}$, it is possible to transfer the data space $\mathcal{X}$ to a (high-dimensional) feature space $\Phi(\mathcal{X})$ endowed with more structure in such a way that distances are preserved. Technically spoken, this can be achieved by the use of *injective* and *structure preserving* mappings, also called *embeddings*. In order to identify hyperplanes defined on a feature space $\Phi(\mathcal{X})$ with (non-linear) decision functions $f \in \mathcal{F}$ defined on the original data space $\mathcal{X}$, one is particularly interested in *bijective* embeddings (*isomorphisms*). Isomorphisms preserve the natural metric d defined on $\mathcal{X}$ at least up to a constant factor. This ensures that the structure of the data space will not be distorted too much.

The most important embedding in the context of binary classification, was recently proposed by [von Luxburg 2004]. Therein, a maximum margin classification is derived based on a simultaneous isometric embbeding of a metric space and its associated space of Lipschitz continuous functions into an appropriate Banach space.

5.1.1 Lipschitz Spaces

We first introduce some spaces and their properties important for what follows. For more details, we refer to [Weaver 1999].

5.1.1.1 The Space $\mathcal{L}IP(\mathcal{M})$

Definition 5.1.1 (Lipschitz function space)**.** *Consider a metric space $\mathcal{M} := (\mathcal{X}, d)$. Then, a function $f : \mathcal{X} \to \mathbb{R}$ is called Lipschitz function, if there exists a constant L such that $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X} \ : \ |f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq L \cdot d(\boldsymbol{x}, \boldsymbol{y})$. The smallest constant L is called Lipschitz constant $L(f)$. The space $\mathcal{L}IP(\mathcal{M})$ denotes the space of all Lipschitz functions defined on the metric space $\mathcal{M}$ with bounded metric.*

Because for $f, g \in \mathcal{L}IP(\mathcal{M})$ it holds $L(f+g) \leq L(f)+L(g)$ and $L(\alpha \cdot f) = |\alpha| \cdot L(f)$, but not $L(f) = 0 \Leftrightarrow f = 0$, it follows that $L : \mathcal{F} \to \mathbb{R}$ with

$$L(f) := \sup_{\boldsymbol{x},\boldsymbol{y} \in \mathcal{X}, \boldsymbol{x} \neq \boldsymbol{y}} \frac{|f(\boldsymbol{x}) - f(\boldsymbol{y})|}{d(\boldsymbol{x}, \boldsymbol{y})} \tag{5.2}$$

is a semi-norm on the space $\mathcal{F} \subseteq \mathcal{L}IP(\mathcal{M})$. Note, $L(f)$ is just a semi-norm, because for any constant function f it holds $L(f) = 0$. But one can augment the semi-norm to get a convenient norm on the vector space[1] $\mathcal{L}IP(\mathcal{M})$:

$$\max\{L(f), \|f\|_\infty\}. \tag{5.3}$$

Now suppose Lipschitz functions $\mathcal{F}$ for classifying two classes. Reasonable decision functions should take positive and negative values on $\mathcal{X}$ dependent on the category of a considered data point $\boldsymbol{x} \in \mathcal{X}$. In case of such positive and negative valued functions, i.e.

$$\mathcal{F}^+_- := \left\{ f \in \mathcal{L}IP(\mathcal{M}) \;\middle|\; \exists \boldsymbol{a}, \boldsymbol{b} \in \mathcal{X} : f(\boldsymbol{a}) f(\boldsymbol{b}) \leq 0 \right\}, \tag{5.4}$$

it holds

$$\|f\|_\infty = \sup_{\boldsymbol{x} \in \mathcal{X}} |f(\boldsymbol{x})| \leq \sup_{\boldsymbol{x},\boldsymbol{y} \in \mathcal{X}} |f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq diam(\mathcal{X}) \cdot L(f) \tag{5.5}$$

with $0 < diam(\mathcal{X}) := \sup_{\boldsymbol{x},\boldsymbol{y} \in \mathcal{X}} d(\boldsymbol{x}, \boldsymbol{y}) < \infty$.

Modifying the norm (5.3) to

$$\|f\|_L := \max\left\{ L(f), \frac{\|f\|_\infty}{diam(\mathcal{X})} \right\} \tag{5.6}$$

[1] Indeed, one can prove [Weaver 1999] that $(\mathcal{L}IP(\mathcal{M}), \max\{L(f), \|f\|_\infty\})$ is a Banach space.

it immediately follows from (5.5), if $0 < diam(\mathcal{X})$, that $\frac{\|f\|_\infty}{diam(\mathcal{X})} \leq L(f)$ and thus

$$\forall f \in \mathcal{F}_-^+ \ : \ \|f\|_L = L(f). \tag{5.7}$$

5.1.1.2 The Space $\mathcal{L}IP_0(\mathcal{M}_0)$

Consider an augmentation of the metric space $\mathcal{M}$ with a distinguished point $\boldsymbol{e}$, which is fixed in advance such that $\mathcal{M}_0 := (X_0, d_{\mathcal{X}_0}), \mathcal{X}_0 := \mathcal{X} \cup \{\boldsymbol{e}\}$ (*pointed metric space*). Then another Lipschitz space is defined by

$$\mathcal{L}IP_0(\mathcal{M}_0) := \left\{ f \in \mathcal{L}IP(\mathcal{M}_0) \;\middle|\; f(\boldsymbol{e}) = 0 \right\}. \tag{5.8}$$

Clearly, for this space $L(f)$ is indeed a norm. Unfortunately, the constraint $f(\boldsymbol{e}) = 0$ is an improper assumption in a classification setting. On the other hand, one can easily circumvent this constraint in two steps: First, by defining the metric of $\mathcal{M}_0$ to be

$$d_{\mathcal{X}_0}(\boldsymbol{x}, \boldsymbol{y}) := \begin{cases} d(\boldsymbol{x}, \boldsymbol{y}) & \text{if } \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X} \\ diam(\mathcal{X}) & \text{if } \boldsymbol{x} \in \mathcal{X}, \boldsymbol{y} = \boldsymbol{e}. \end{cases} \tag{5.9}$$

Second, by embedding the space $\mathcal{L}IP(\mathcal{M})$ into the space $\mathcal{L}IP_0(\mathcal{M}_0)$ via the bijective and isometric mapping $\psi : \mathcal{L}IP(\mathcal{M}) \to \mathcal{L}IP_0(\mathcal{M}_0)$ with

$$\psi(f)(\boldsymbol{x}) := \begin{cases} f(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in \mathcal{X} \\ 0 & \text{if } \boldsymbol{x} = \boldsymbol{e}. \end{cases} \tag{5.10}$$

Bijectivity of ψ follows trivially, and isometry follows directly from

$$\begin{aligned} L(\psi(f)) &= \sup_{\boldsymbol{x},\boldsymbol{y} \in \mathcal{X}_0, \boldsymbol{x} \neq \boldsymbol{y}} \frac{|\psi(f)(\boldsymbol{x}) - \psi(f)(\boldsymbol{y})|}{d_{\mathcal{X}_0}(\boldsymbol{x}, \boldsymbol{y})} && (5.11) \\ &= \max \left\{ \sup_{\boldsymbol{x},\boldsymbol{y} \in \mathcal{X}, \boldsymbol{x} \neq \boldsymbol{y}} \frac{|f(\boldsymbol{x}) - f(\boldsymbol{y})|}{d(\boldsymbol{x}, \boldsymbol{y})}, \sup_{\boldsymbol{x} \in \mathcal{X}} \frac{|f(\boldsymbol{x}) - f(\boldsymbol{e})|}{diam(\mathcal{X})} \right\} && (5.12) \\ &= \|f\|_L. && (5.13) \end{aligned}$$

In particular, for positive and negative valued functions, i.e. $f \in \mathcal{F}_-^+$, it holds $L(\psi(f)) = L(f)$.

The space $\mathcal{L}IP_0(\mathcal{M}_0)$ has some interesting dual properties, as we will see in the next subsection.

5.1.1.3 The Arens-Eells Space (Predual of $\mathcal{L}IP_0(\mathcal{M}_0)$)

Definition 5.1.2 (xy-Atom)**.** *Let be $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$, then a xy-atom is a function $m_{\boldsymbol{xy}} : \mathcal{X} \to \{-1, 1, 0\}$ defined as*

$$m_{\boldsymbol{xy}}(\boldsymbol{z}) := \begin{cases} -1 & \textit{if } \ \boldsymbol{y} = \boldsymbol{z} \\ 1 & \textit{if } \ \boldsymbol{x} = \boldsymbol{z} \\ 0 & \textit{otherwise} \ . \end{cases} \tag{5.14}$$

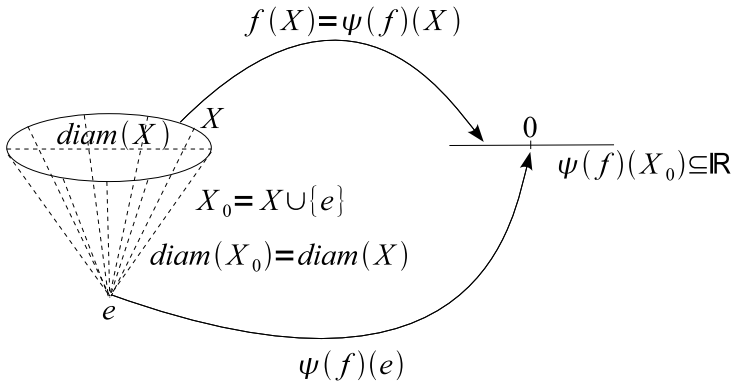


Figure 5.1: A visualization of the embedding $\psi : \mathcal{LIP}(\mathcal{M}) \to \mathcal{LIP}_0(\mathcal{M}_0)$. The extended space $\mathcal{X}_0$ is a conjunction of the space $\mathcal{X}$ (here a closed curve) with $diam(X)$ and the base point $\{\boldsymbol{e}\}$.

Definition 5.1.3 (Molecule)**.** *A function $m : \mathcal{X} \to \mathbb{R}$ with finite support $\{\boldsymbol{z} \in \mathcal{X} \mid m(\boldsymbol{z}) \neq 0\}$ and the property $\sum_{\boldsymbol{z}\in\mathcal{X}} m(\boldsymbol{z}) = 0$ is called a molecule of $\mathcal{X}$. Any molecule has a not unique expansion of atoms, i.e.*

$$m(\boldsymbol{z}) = \sum_{i=1}^{n} a_i \cdot m_{\boldsymbol{x}_i\boldsymbol{y}_i}(\boldsymbol{z}),\ n \in \mathbb{Z}^+,\ \boldsymbol{x}_i, \boldsymbol{y}_i \in \mathcal{X},\ a_i \in \mathbb{R}.$$

Definition 5.1.4 (Arens-Eells space)**.** *The completion of the space of all molecules (Def. 5.1.3) endowed with the semi-norm*

$$\|m\|_{AE} := \inf\left\{ \sum_{i=1}^{n} |a_i| \cdot d(\boldsymbol{x}_i, \boldsymbol{y}_i) \;\middle|\; m(\boldsymbol{z}) = \sum_{i=1}^{n} a_i \cdot m_{\boldsymbol{x}_i\boldsymbol{y}_i}(\boldsymbol{z}), \right.$$
$$\left. n \in \mathbb{Z}^+,\ \boldsymbol{x}_i, \boldsymbol{y}_i \in \mathcal{X},\ a_i \in \mathbb{R} \right\} \qquad (5.15)$$

modulo the subspace $\{m : \|m\|_{AE} = 0\}$ is called Arens-Eells space $AE(\mathcal{X})$.

In [Weaver 1999][Corollary 2.2.3] it is proved, that for pointed metric spaces $\mathcal{M}_0$ the Arens-Eells semi-norm $\|\cdot\|_{AE}$ is even a norm on the space of molecules of $\mathcal{X}_0$. Therefore, the Arens-Eells space is a Banach space (cf. App. A.2.2).

Moreover, it holds the powerful

Theorem 5.1.1. *The dual space $AE^*(\mathcal{X}_0)$ of $AE(\mathcal{X}_0)$, i.e. the space of all linear functionals on $AE(\mathcal{X}_0)$, is isometrically isomorphic to $\mathcal{LIP}_0(\mathcal{M}_0)$.*

Proof. See [Weaver 1999][Theorem 2.2.2]. ∎

A nice consequence of Theorem (5.1.1) is that there exists an unique linear functional $T_f : AE(\mathcal{X}_0) \to \mathbb{R}$ associated to any Lipschitz function $f \in \mathcal{L}IP_0(\mathcal{M}_0)$ with

$$T_f(m) := \sum_{\boldsymbol{y} \in \mathcal{X}_0} f(\boldsymbol{y}) m(\boldsymbol{y}). \tag{5.16}$$

In particular, $T_f(m_{\boldsymbol{xe}}) = f(\boldsymbol{x})$.

Moreover, the mapping $f \mapsto T_f$ is an isometry, that is it holds

$$L(f) = \|f\|_L = \|T_f\| := \sup_{\substack{m \in AE(\mathcal{X}_0) \\ \|m\|_{AE} \neq 0}} \frac{|T_f(m)|}{\|m\|_{AE}}. \tag{5.17}$$

Now, from the Definition (5.15), it follows directly $\|m_{\boldsymbol{xy}}\|_{AE} \leq d_{\mathcal{X}_0}(\boldsymbol{x}, \boldsymbol{y})$. And in virtue of the *Hahn-Banach theorem*, for any $m \in AE(\mathcal{X}_0)$ exists a $f_0 \in \mathcal{L}IP_0(\mathcal{M}_0)$ and a linear functional $T_{f_0} \in AE^*(\mathcal{X}_0)$ with $T_{f_0}(m) = \|m\|_{AE}$. In particular, it holds

$$\|m\|_{AE} = \max \left\{ |T_f(m)| \ : \ f \in \mathcal{L}IP_0(\mathcal{M}_0), L(f) \leq 1 \right\}. \tag{5.18}$$

Defining $f_{\boldsymbol{y}} \in \mathcal{L}IP_0(\mathcal{M}_0),\ \boldsymbol{y} \in \mathcal{X}_0$ with $L(f_{\boldsymbol{y}}) = 1$ via $f_{\boldsymbol{y}}(\boldsymbol{z}) := d_{\mathcal{X}_0}(\boldsymbol{z}, \boldsymbol{y}) - d_{\mathcal{X}_0}(\boldsymbol{e}, \boldsymbol{y})$ it follows

$$\|m_{\boldsymbol{xy}}\|_{AE} \geq |T_{f_{\boldsymbol{y}}}(m_{\boldsymbol{xy}})| = |f_{\boldsymbol{y}}(\boldsymbol{x}) - f_{\boldsymbol{y}}(\boldsymbol{y})| = d_{\mathcal{X}_0}(\boldsymbol{x}, \boldsymbol{y}). \tag{5.19}$$

Thus, $\|m_{\boldsymbol{xe}} - m_{\boldsymbol{ye}}\|_{AE} = \|m_{\boldsymbol{xy}}\|_{AE} = d_{\mathcal{X}_0}(\boldsymbol{x}, \boldsymbol{y})$, and we have proved

Corollary 5.1.1. *The mapping* $(\boldsymbol{x} \mapsto m_{\boldsymbol{xe}}) : \mathcal{X}_0 \to AE(\mathcal{X}_0)$ *is an isometry.*

5.1.2 Maximum Margin Classifier using Lipschitz Continuous Decision Functions

In [von Luxburg 2004] the duality results presented in the preceding subsections were used to define a maximum margin classifier on the space $AE(\mathcal{X}_0)$.

For this purpose, consider the following implicit embeddings of the data space $\mathcal{X}$ and the decision function space $\mathcal{F}_-^+ \subseteq \mathcal{F} \subseteq \mathcal{L}IP(\mathcal{M})$

$$\Phi : \mathcal{X} \to \mathcal{X}_0 \to AE(\mathcal{X}_0), \boldsymbol{x} \mapsto m_{\boldsymbol{xe}} \tag{5.20}$$

$$\Psi : \mathcal{F} \to \mathcal{L}IP_0(\mathcal{M}_0) \to AE^*(\mathcal{X}_0), f \mapsto T_f. \tag{5.21}$$

Using these embeddings, a hyperplane H_f in $AE(\mathcal{X}_0)$ is straightforwardly defined by

$$H_f := \{ m \in AE(\mathcal{X}_0) \ : \ T_f(m) = 0 \}. \tag{5.22}$$

As the minimum distance of a set of data points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_l\} \subset \mathcal{X}$ to a hyperplane H_f (cf. Fig. (3.6), the margin ρ reads

$$\rho = \inf_{1 \leq i \leq l,\ m \in H_f} \|m_{\boldsymbol{x}_i \boldsymbol{e}} - m\|_{AE}. \tag{5.23}$$

Consider a canonical hyperplane H_f, meaning the linear functional T_f is scaled[2] such that $\min_{1 \leq i \leq l} |T_f(m_{\boldsymbol{x}_i \boldsymbol{e}})| = 1$, it holds by the *Cauchy-Schwarz inequality* for any $m_{\boldsymbol{x}_i \boldsymbol{e}}$ and $m_h \in H_f$

$$\|T_f\| \|m_{\boldsymbol{x}_i \boldsymbol{e}} - m_h\|_{AE} \geq |T_f(m_{\boldsymbol{x}_i \boldsymbol{e}} - m_h)| = |T_f(m_{\boldsymbol{x}_i \boldsymbol{e}}) - T_f(m_h)| = |T_f(m_{\boldsymbol{x}_i \boldsymbol{e}})|. \tag{5.24}$$

And it follows by taking the infimum, the margin associated to a canonical hyperplane is bounded from below

$$\rho = \inf_{1 \leq i \leq l,\ m \in H_f} \|m_{\boldsymbol{x}_i \boldsymbol{e}} - m\|_{AE} \geq \frac{\inf_{1 \leq i \leq l} |T_f(m_{\boldsymbol{x}_i \boldsymbol{e}})|}{\|T_f\|} = \frac{1}{\|T_f\|}, \tag{5.25}$$

if $\|T_f\| > 0$. In particular, it holds for all $f \in \mathcal{F}_-^+$ due to Theorem (5.1.1)

$$\rho \geq \frac{1}{L(f)}. \tag{5.26}$$

So, likewise in case of the SVM (Sec. (3.6)), a canonical hyperplane and classification of all training examples without error implies the (hard margin) inequality

$$y_i T_f(m_{\boldsymbol{x}_i \boldsymbol{e}}) = y_i f(\boldsymbol{x}_i) \geq 1. \tag{5.27}$$

In particular, if a training sample $\mathcal{O}_l$ contains examples of both classes (the common case), then inequality (5.27) implies $f \in \mathcal{F}_-^+$. Consequently and in analogy to the SVM (Sec. (3.6)), the resulting algorithms for learning a hard- and soft-maximum margin classifier using Lipschitz continuous decision functions $f \in \mathcal{F} \subseteq \mathcal{L}IP(\mathcal{M})$ can be restated as

Algorithm 5.1.1 $\mathcal{L}IP(\mathcal{M})$-Lipschitz Classifier (Hard-Margin) [von Luxburg 2004]

Require: Training set $\emptyset \neq \mathcal{O}_N \subset \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, decision function space $\mathcal{F} \subseteq \mathcal{L}IP(\mathcal{M})$

Ensure: $\exists (\boldsymbol{x}_j, \boldsymbol{y}_j), (\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathcal{O}_N \ : \ y_j = -1 \wedge y_i = 1$, both classes are separable in $AE(\mathcal{X}_0)$

$$\inf_{f \in \mathcal{F} \subseteq \mathcal{L}IP(\mathcal{M})} L(f) \tag{5.28}$$

$$\text{s.t. } \forall (\boldsymbol{x}_j, y_j) \in \mathcal{O}_N : y_j f(\boldsymbol{x}_j) \geq 1 \tag{5.29}$$

[2] Scaling a linear operator equals scaling its associated representative, i.e. $\alpha T_f = T_{\alpha f}$.

Algorithm 5.1.2 $\mathcal{L}IP(\mathcal{M})$-Lipschitz Classifier (Soft-Margin) [von Luxburg 2004]

Require: Training set $\emptyset \neq \mathcal{O}_N \subset \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, decision function space $\mathcal{F} \subseteq \mathcal{L}IP(\mathcal{M})$, trade-off parameter $C > 0$

Ensure: $\exists(\boldsymbol{x}_j, y_j), (\boldsymbol{x}_i, y_i) \in \mathcal{O}_N \ : \ y_j = -1 \wedge y_i = 1$

$$\inf_{f \in \mathcal{F} \subseteq \mathcal{L}IP(\mathcal{M}),\ (\xi_1,\ldots,\xi_l) \in \mathbb{R}^N} L(f) + C \sum_{j=1}^{N} \xi_j \tag{5.30}$$

$$\text{s.t.} \quad \forall 1 \leq j \leq N : \xi_j \geq 0, \tag{5.31}$$

$$\forall (\boldsymbol{x}_j, y_j) \in \mathcal{O}_N : y_j f(\boldsymbol{x}_j) \geq 1 - \xi_j \tag{5.32}$$

Algorithm (5.1.1) and (5.1.2) represent a prototype for constructing large margin classifiers. For example, the SVM can be obtained as a particular solution if restricting the decision function space $\mathcal{F} \subseteq \mathcal{L}IP(\mathcal{M})$ to the set of all linear functionals $\mathcal{R}'$. The reason is that the Lipschitz constant of linear functionals coincide with their operator norm, respectively Hilbert space norm, i.e. with $f \in \mathcal{H}$ holds $\forall T_f :=< \cdot, f >_{\mathcal{H}} \in \mathcal{R}' : L(T_f) = \|T_f\| = \|f\|_{\mathcal{H}}$ (cf. Sec. (3.6.4)). Also the 1-nearest neighbor classifier can be shown to be a particular solution [von Luxburg 2004]. Further, suppose the decision function space $\mathcal{F}$ to be the set of all linear combinations of distance functions of the type $f(\boldsymbol{x}) := \sum_{n=1}^{N} \beta_i d(\boldsymbol{x}_i, \boldsymbol{x}) + \boldsymbol{c}$. Then it is easy to see, that $L(f) \leq \sum_{n=1}^{N} |\beta_n|$. Thus, Algorithm (4.2.2) is an approximation to the maximum hard-margin Algorithm (5.1.1) as already mentioned in Section (4.2).

5.2 Toward Implementable Algorithms

The results presented in the last section are promising. The new maximum margin Algorithms (5.1.1) and (5.1.2) permit to implement decision functions from a very rich space $\mathcal{F} \subseteq \mathcal{L}IP(\mathcal{M})$. Now, we are actually in the nice position to implement nonlinear decision functions in a maximum margin concept that must not necessarily be build up with kernel functions, like in case of SVMs. For this purpose, the decision functions have to satisfy only the much milder restriction to be just Lipschitz continuous ($\mathcal{M}$ have to be bounded) than it is the case from SVMs which implement linear combinations of Mercer kernels. We showed in the preceding section that an optimal decision function found by Algorithm (5.1.1) with minimum Lipschitz constant can be interpreted geometrically as a large margin hyperplane in an appropriate Banach space, namely the Arens-Eells space $AE(\mathcal{X}_0)$ (the decision function itself corresponds to a linear functional of the dual $AE^*(\mathcal{X}_0)$ of the Arens-Eells space $AE(\mathcal{X}_0)$,

cf. Def. (5.1.4)). The margin between a hyperplane in $AE(\mathcal{X}_0)$ and the data which is implicitly transformed to $AE(\mathcal{X}_0)$ is bounded from below by the Lipschitz constant (cf. Eq. (5.26)).

However, what remains is the problem of how to realize the Lipschitz classifier algorithm practically. In order to implement the Lipschitz classifier one has to evaluate the Lipschitz constant $L(f)$ for any decision function f of a chosen subspace $\mathcal{F}$. In case of the LP Machine (Alg. (4.2.2)) it is possible to obtain a poor upper bound on the Lipschitz constant resulting in a poor approximation to a maximum margin classifier. In general, it is difficult to get a closed-form expression of $L(f)$. Thus from a practical point of view, a chosen space $\mathcal{F}$ shall satisfy two goals: First, the space should contain a rich class of decision functions in order to be able to solve nearly every real-world classification task. Second, the Lipschitz constant shall be easy to compute.

For this reason, we choose a subset $\mathcal{F}$ of the space of real-valued, at least one-time continuously differentiable functions $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R}) \subset \mathcal{LIP}(\mathcal{M})$ defined on a compact and convex Euclidean metric space $\mathcal{M} := (\mathcal{X}, \|\cdot - \cdot\|_2), \mathcal{X} \subset \mathbb{R}^m$ [Stuhlsatz 2007c] (Fig. 5.2). In that case, an analytic expression of the Lipschitz constant can be obtained:

Lemma 5.2.1 (Lipschitz Constant for $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$ functions)**.** *Suppose, the normed space $\mathcal{V} := (\mathbb{R}^m, \|\cdot\|_2)$, $\mathcal{D}_\mathcal{X} \subseteq \mathcal{V}$ open and $\mathcal{X} \subset \mathcal{D}_\mathcal{X}$ compact and convex. If $f : \mathcal{D}_\mathcal{X} \to \mathbb{R}$ is continuously differentiable, then the Lipschitz constant of f restricted to $\mathcal{X}$ is given by*

$$L(f) = \max_{\boldsymbol{x} \in \mathcal{X}} \|\nabla f(\boldsymbol{x})\|_2 \,. \tag{5.33}$$

Proof. Consider $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}, \boldsymbol{x}_1 \neq \boldsymbol{x}_2$ and the set

$$\mathcal{S} := \{\lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2 \ : \ \lambda \in (0,1)\} \subseteq \mathcal{X},$$

then by the *Mean-Value-Theorem* there exists $\hat{\boldsymbol{x}} \in \mathcal{S}$, so that

$$\begin{aligned} |f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)| &= \left\|\nabla f(\hat{\boldsymbol{x}})^T (\boldsymbol{x}_1 - \boldsymbol{x}_2)\right\|_2 \\ &\leq \|\nabla f(\hat{\boldsymbol{x}})\|_2 \, \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2 \\ \Leftrightarrow \quad \frac{|f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)|}{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2} &\leq \|\nabla f(\hat{\boldsymbol{x}})\|_2 \leq \max_{\boldsymbol{x} \in \mathcal{X}} \|\nabla f(\boldsymbol{x})\|_2 \end{aligned}$$

$$\begin{aligned} \Rightarrow L(f) &= \max_{\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}, \boldsymbol{x}_1 \neq \boldsymbol{x}_2} \frac{|f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)|}{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2} \\ &\leq \max_{\boldsymbol{x} \in \mathcal{X}} \|\nabla f(\boldsymbol{x})\|_2 \,. \end{aligned}$$

For $\boldsymbol{x} \in \mathcal{X}$ and some $\boldsymbol{v} \in \mathbb{R}^m$ with $\|\boldsymbol{v}\|_2 = 1$, the directional derivative reads as

$$\lim_{t \to 0} \frac{f(\boldsymbol{x} + t\boldsymbol{v}) - f(\boldsymbol{x})}{t} = \partial_{\boldsymbol{v}} f(\boldsymbol{x}) = \nabla f(\boldsymbol{x})^T \boldsymbol{v}.$$

Next, we conclude for $\|\nabla f(\boldsymbol{x})\|_2 \neq 0$ and $\boldsymbol{v} := \nabla f(\boldsymbol{x}) / \|\nabla f(\boldsymbol{x})\|_2$ that

$$\|\nabla f(\boldsymbol{x})\|_2 = \lim_{t \to 0} \frac{|f(\boldsymbol{x} + t\boldsymbol{v}) - f(\boldsymbol{x})|}{|t|} \leq L(f).$$

It follows the lower bound

$$\max_{\boldsymbol{x} \in \mathcal{X}} \|\nabla f(\boldsymbol{x})\|_2 \leq L(f).$$

■

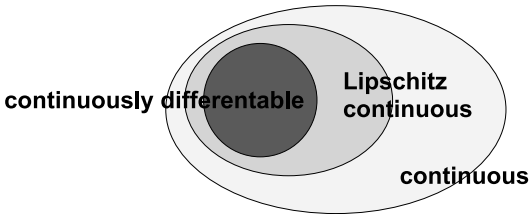


Figure 5.2: Subspaces of the space of continuous functions. The space $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$ of at least one-time continuously differentiable functions is a subspace of the Lipschitz function space.

Lemma (5.2.1) is very general permitting us the computation of Lipschitz constants for decision functions f of the space $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$ (Fig. 5.2). A typical decision function implemented by many common machine learning algorithms is chosen from the space of finite (affine) linear combinations of some basis functions $\Phi_n : \mathcal{X} \to \mathbb{R}$. For example in case of the SVM a decision function is selected from a RKHS $\mathcal{R} = (\mathcal{H}, \langle \cdot, \cdot \rangle_{\mathcal{H}})$ induced by a kernel k (cf. (3.71)) that must satisfy the Mercer's condition. It follows, a solution of the SVM is a finite series of basis functions $\Phi_n = k(\cdot, \boldsymbol{x}_n)$ (cf. Eq. (3.56) and the Representer Theorem [Kimeldorf 1971]).

Because our goal is to derive practical maximum margin algorithms using Lipschitz decision functions, we further focus our attention in compliance and due to technical reasons to a subspace $\mathcal{F} \subset \mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$ of finite affine combinations of $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-functions. However, the algorithms we derive in the following sections do not select a decision function from a RKHS induced by a chosen Mercer kernel k in advance. In contrast, our algorithms seek for a decision function that is a finite series of basis functions $\Phi_n \in \mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$ which corresponds implicitly to a separating hyperplane in the Arens-Eells space $AE(\mathcal{X}_0)$.

Due to a simultaneous minimization of the Lipschitz constant, a solution is associated to the hyperplane with maximum margin, respectively to the decision function which is of minimum possible variation. From now on, we may design a decision function for a maximum margin classification independently of Mercer's condition permitting us to use basis functions $\Phi_n \in \mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$ like for example continuously differentiable similarity functions, pseudo-kernels, kernels, trigonometric functions or polynomials as well as mixtures of them. Moreover, if appropriate the basis function must not necessarily be evaluated at the training data, as it is the case using SVMs.

5.2.1 Lipschitz Classifier as Minimax Problem

As mentioned in the preceding section, a typical function space implemented by many machine learning algorithms, e.g. the SVM, is the space of finite affine linear combinations of some basis functions $\Phi_n : \mathcal{X} \to \mathbb{R}$. In case of SVMs these basis functions are defined by kernels that have to satisfy the Mercer's condition, i.e. $\Phi_n = k(\cdot, \boldsymbol{x}_n)$. In the following, our focus is to derive maximum margin classifiers permitting us to use more basis function types than just kernels by the application of the Lipschitz classifier framework and Lemma (5.2.1) introduced in the preceding sections. For this purpose, we consider from now on basis functions $\Phi_n \in \mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$ for building up a decision function with respect to a compact and convex data space $\mathcal{X} \subset \mathbb{R}^m$:

Definition 5.2.1 (Decision function space of the Lipschitz classifier)**.** *Let be $\mathcal{D}_\mathcal{X} \subseteq \mathbb{R}^m$ open, $\mathcal{X} \subset \mathcal{D}_\mathcal{X}$ compact and convex as well as $\Phi_n : \mathcal{D}_\mathcal{X} \to \mathbb{R}$ with $1 \leq n \leq M$ arbitrary at least one-time continuously differentiable basis functions, i.e. $\Phi_n \in \mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$ is restricted to $\mathcal{X}$. Then, we define the decision function space under consideration to be*

$$\mathcal{F} := \left\{ f : \mathcal{X} \to \mathbb{R} \;\middle|\; f(\boldsymbol{x}) := \sum_{n=1}^{M} c_n \Phi_n(\boldsymbol{x}) + b,\ c_n, b \in \mathbb{R} \right\} \subset \mathcal{LIP}(\mathcal{X}).$$

Note, the boundedness of $f \in \mathcal{F}$ is implied by the compactness of $\mathcal{X}$. Considering the function space of Definition (5.2.1) together with Lemma (5.2.1) enables to state the original soft-margin algorithm (5.1.2) for finding an optimal decision function $f^* \in \mathcal{F}$ parameterized by $(\boldsymbol{c}^*, b^*) \in \mathbb{R}^{M+1}$ in the following minimax formulation:

Algorithm 5.2.1 $\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz Classifier (Minimax Form)

Require: Training set $\emptyset \neq \mathcal{O}_N \subset \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, $\mathcal{X}$ compact and convex, $1 \leq m \leq M$ basis functions $\Phi_m(\boldsymbol{x}) \in \mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$, $C > 0$

Ensure: $\exists(\boldsymbol{x}_j, y_j), (\boldsymbol{x}_i, y_i) \in \mathcal{O}_N \ : \ y_j = -1 \wedge y_i = 1$

$$\min_{(\boldsymbol{c},\boldsymbol{\xi},b)\in\mathcal{S}_P} \max_{\boldsymbol{x}\in\mathcal{X}} \frac{1}{2}\boldsymbol{c}^T\mathbf{K}(\boldsymbol{x})\boldsymbol{c} + C\mathbf{1}_N^T\boldsymbol{\xi} \tag{5.34}$$

with feasible set

$$\mathcal{S}_P := \left\{ (\boldsymbol{c},\boldsymbol{\xi},b) \in \mathbb{R}^M \times \mathbb{R}^N \times \mathbb{R} \;\middle|\; \boldsymbol{\xi} \geq \mathbf{0}, \mathbf{Y}\mathbf{G}\boldsymbol{c} + \boldsymbol{y}b - \mathbf{1}_N + \boldsymbol{\xi} \geq \mathbf{0} \right\} \tag{5.35}$$

with matrix notations:

- $\mathbf{K}(\boldsymbol{x}) \in \mathbb{R}^{M\times M}$ is symmetric positive-semidefinite with elements $\mathbf{K}(\boldsymbol{x})_{m,n} := \langle \nabla\Phi_m(\boldsymbol{x}), \nabla\Phi_n(\boldsymbol{x})\rangle_2 \ \forall \boldsymbol{x} \in \mathcal{X}$,
- $\mathbf{G} \in \mathbb{R}^{N\times M}$ is a data dependent design matrix with elements $\mathbf{G}_{n,m} := \Phi_m(\boldsymbol{x}_n)$,
- $\mathbf{Y} := diag(\boldsymbol{y}) \in \mathbb{R}^{N\times N}$ is a diagonal matrix of a given target vector $\boldsymbol{y} \in \{-1,1\}^N$ with components y_n,
- $\boldsymbol{\xi} \in \mathbb{R}^N$ is the slack-variable of the soft margin formulation with components ξ_n,
- $\mathbf{1}_N := (1,\ldots,1)^T \in \mathbb{R}^N$ is the vector of ones.

It is easy to see that algorithm (5.2.1) is equivalent[3] to algorithm (5.1.2) if using the subspace $\mathcal{F} \subset \mathcal{L}IP(\mathcal{X})$. That means a solution $(\boldsymbol{c}^*, \boldsymbol{\xi}^*, b^*, \boldsymbol{x}^*)$ of (5.2.1) implies a solution $(f^*, \boldsymbol{\xi}^*)$ of (5.1.2). Obviously, a maximizer of

$$\|\nabla f(\boldsymbol{x})\|_2 = \sqrt{\sum_{m=1}^{M}\sum_{n=1}^{M} c_m c_n \langle \nabla\Phi_m(\boldsymbol{x}), \nabla\Phi_n(\boldsymbol{x})\rangle_2} \tag{5.36}$$

is also a maximizer of $\|\nabla f(\boldsymbol{x})\|_2^2/2$.

It is important to note, that the objective function $\boldsymbol{c}^T\mathbf{K}(\boldsymbol{x})\boldsymbol{c}$ is convex in $\boldsymbol{c} \in \mathbb{R}^M$ because

$$0 \leq \|\nabla f(\boldsymbol{x})\|_2^2 = \sum_{m=1}^{M}\sum_{n=1}^{M} c_m c_n \langle \nabla\Phi_m(\boldsymbol{x}), \nabla\Phi_n(\boldsymbol{x})\rangle_2 \tag{5.37}$$

$$= \boldsymbol{c}^T\mathbf{K}(\boldsymbol{x})\boldsymbol{c}, \tag{5.38}$$

i.e. the positive-semidefinitness of $\mathbf{K}(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{X}$. Moreover, as a pointwise maximum of a family of convex functions the function $\max_{\boldsymbol{x}\in\mathcal{X}} \boldsymbol{c}^T\mathbf{K}(\boldsymbol{x})\boldsymbol{c}$

[3] Equivalent with respect to a solution.

is convex in $\boldsymbol{c} \in \mathbb{R}^M$. The feasible set $\mathcal{S}_P$ is convex as well, because it is the intersection of half-spaces induced by linear functions. Hence, for a fixed $\boldsymbol{x} \in \mathcal{X}$, the minimization problem is a typical (convex) quadratic optimization problem which is well studied in optimization theory. It is very similar to the SVM optimization problem (3.6.1). But in contrast to the SVM's regularizer $(1/2)\|\boldsymbol{w}\|_2^2$, the regularization operator in (5.34) is a solution of a global maximization problem $\Omega(\boldsymbol{c}) := 1/2 \cdot \max_{\boldsymbol{x}\in\mathcal{X}} \boldsymbol{c}^T\mathbf{K}(\boldsymbol{x})\boldsymbol{c} = 1/2 \cdot L(f)^2$ which is proportional to the absolute value of the maximum slope of f on the data space $\mathcal{X}$. Actually, Algorithm (5.2.1) seeks for a decision function f which has a minimal maximum slope over its domain and classifies the training data with minimum error with respect to a soft-margin controlled by C. Thus, the optimal function $f^* \in \mathcal{F}$ implies the flattest decision boundary with respect to the data space $\mathcal{X}$ (and w.r.t. its first derivative) that can be constructed by a linear combination of M basis functions Φ_m. Surprisingly, for a solution $\boldsymbol{x}(\boldsymbol{c})$ of $\max_{\boldsymbol{x}\in\mathcal{X}} \boldsymbol{c}^T\mathbf{K}(\boldsymbol{x})\boldsymbol{c}$ the regularization operator equals the regularization operator $\int_{\mathcal{X}} \|\nabla f(\boldsymbol{x})\|_2^2 p(\boldsymbol{x}|\vartheta)\, d\boldsymbol{x}$ implied by a plain kernel (Sec. 3.6.5) if the distribution $p(\boldsymbol{x}|\vartheta)$ is supposed to be a delta distribution $\delta(\boldsymbol{x} - \boldsymbol{x}(\boldsymbol{c}))$ (cf. [Oliver 2000]).

However, solving the constrained minimax problem (5.34) is non-trivial avoiding the application of standard optimization techniques known from optimization theory. In the following section, we will reformulate problem (5.34) in order to exploit as much as possible of the inherent problem structure and to be able to apply standard optimization methods.

5.2.2 On Solving the Minimax Problem

In the previous section, a constrained multidimensional minimax problem (5.34) has been obtained representing the Lipschitz classifier (5.30) in case of the chosen function space $\mathcal{F}$ (Def. 5.2.1) and a convex and compact Euclidean data space $\mathcal{X}$. Unfortunately, this problem is difficult to solve directly because of the restricted inner global maximization depending on the constrained convex outer minimization. However, using the minimax formulation it is possible to exploit the inherent structure as much as possible through a further transformation of the problem in a *Semi-Infinite Program* (SIP). With the resulting reformulation, we will be in the position to apply standard optimization techniques.

We start our discussion with a brief introduction to Semi-Infinite Programming. Then we show how to derive a SIP formulation of the Lipschitz classifier that is equivalent to (5.34). In particular, we derive and prove a duality theorem for (5.34). Recall, duality is also the key to an efficient algorithmic implementation of the SVM (Sec. 3.6). The resulting new statement

of the Lipschitz classifier problem opens a way for an iterative optimization using standard methods.

5.2.2.1 Semi-Infinite Programming

A constrained minimization of an objective $F : \mathbb{R}^L \to \mathbb{R}$ of a finite number of variables is called *Semi-Infinite Program* (SIP) (Alg. 5.2.2), when the feasible set $\mathcal{S}_{SIP} \subseteq \mathbb{R}^L$ is described by a infinite number of constraints $G(\cdot, \boldsymbol{x}) : \mathbb{R}^L \to \mathbb{R},\ \boldsymbol{x} \in \mathcal{T}$. The set $\mathcal{T}$ is an infinite compact index set. Further, for each $\boldsymbol{x} \in \mathcal{T}$ the functions F and $G(\cdot, \boldsymbol{x})$ are supposed to be at least one-time continuously differentiable on $\mathbb{R}^L$. The constraints $G(\boldsymbol{w}, \cdot)$ are assumed to be continuous on $\mathcal{T}$ for each $\boldsymbol{w} \in \mathbb{R}^L$. Such problems are well-known in different fields, like Chebyshev approximation, optimal control and mathematical physics. For example, refer to [Hettich 1993], [Goberna 2001], [Lopez 2007] for a review of some applications and more details on this topic.

Algorithm 5.2.2 Semi-infinite Program (SIP)

Require: objective function $F : \mathbb{R}^L \to \mathbb{R}$, infinite compact index set $\mathcal{T}$, constraints $G(\cdot, \boldsymbol{x}) : \mathbb{R}^L \to \mathbb{R},\ \boldsymbol{x} \in \mathcal{T}$

Ensure: F and $G(\cdot, \boldsymbol{x})$ are at least one-time continuously differentiable for all $\boldsymbol{x} \in \mathcal{T}$, $G(\boldsymbol{w}, \cdot)$ is continuous for all $\boldsymbol{w} \in \mathbb{R}^L$

$$\min_{\boldsymbol{w} \in \mathcal{S}_{SIP}} F(\boldsymbol{w}) \tag{5.39}$$

with feasible set

$$\mathcal{S}_{SIP} := \left\{ \boldsymbol{w} \in \mathbb{R}^L \mid G(\boldsymbol{w}, \boldsymbol{x}) \leq 0\ \forall \boldsymbol{x} \in \mathcal{T} \right\} \neq \emptyset. \tag{5.40}$$

In what follows, we need a few definitions:

Definition 5.2.2 (Active set)**.** *For $\boldsymbol{w} \in \mathcal{S}_{SIP}$ the set*

$$\mathcal{T}^0(\boldsymbol{w}) := \{ \boldsymbol{x} \in \mathcal{T} \ :\ G(\boldsymbol{w}, \boldsymbol{x}) = 0 \} \tag{5.41}$$

is called the active set at $\boldsymbol{w} \in \mathbb{R}^L$.

Definition 5.2.3 (Convex SIP)**.** *A Semi-Infinite Program (5.39) is called convex SIP, iff the objective function $F : \mathbb{R}^L \to \mathbb{R}$ and the feasible set $\mathcal{S}_{SIP}$ are both convex (cf. App. A.3.1, A.3.2), i.e. $G(\cdot, \boldsymbol{x}) : \mathbb{R}^L \to \mathbb{R},\ \boldsymbol{x} \in \mathcal{T}$ is convex for all $\boldsymbol{x} \in \mathcal{T}$.*

Definition 5.2.4 (Convex hull)**.** *The convex hull of a set $\mathcal{H}$ of real valued functions is defined as*

$$conv(\mathcal{H}) := \left\{ \sum_{j=1}^{k} \mu_j h_j \;\middle|\; k \in I\!N, h_j \in \mathcal{H}, \sum_{j=1}^{k} \mu_j = 1 \text{ and } \mu_j \geq 0 \right\}. \quad (5.42)$$

In order to get necessary conditions for a semi-infinite program to be optimal at $\boldsymbol{w}^* \in \mathcal{S}_{SIP}$, the feasible set $\mathcal{S}_{SIP}$ respectively the constraint functions $G(\cdot, \boldsymbol{x})$ have to satisfy so-called *Constraint Qualifications*:

Definition 5.2.5 (MFCQ, e.g. [Lopez 2007])**.** *The Mangasarian-Fromovitz Constraint Qualification (MFCQ) holds at $\boldsymbol{w} \in \mathcal{S}_{SIP}$, iff there exists a direction $\boldsymbol{d} \in \mathbb{R}^L$ such that $\nabla_{\boldsymbol{w}} G(\boldsymbol{w}, \boldsymbol{x})^T \boldsymbol{d} < 0 \; \forall \boldsymbol{x} \in \mathcal{T}^0(\boldsymbol{w})$.*

Descriptively speaking, the MFCQ ensures that at a point $\boldsymbol{w} \in \mathcal{S}_{SIP}$ (which may be a boundary point of the feasible set) there exists a direction that strictly points into the interior of the feasible set. This ensures, that the feasible set is not degenerated and it is possible to derive so-called *Karush-Kuhn-Tucker* (KKT) conditions [Karush 1939], [Kuhn 1951] (see App. (A.3.3)):

Theorem 5.2.1 (Necessary KKT Optimality Condition, e.g. [Lopez 2007])**.** *Let $\boldsymbol{w}^* \in \mathcal{S}_{SIP}$ be a local minimizer of SIP and MFCQ (Def. 5.2.5) holds, then there exist multipliers $\boldsymbol{0} \leq (\mu_1^*, \ldots, \mu_k^*) =: \boldsymbol{\mu}^*$ and $\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_k^* \in \mathcal{T}^0(\boldsymbol{w}^*)$ with $k \leq L$, such that*

$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}^*, \boldsymbol{\mu}^*) = \nabla F(\boldsymbol{w}^*) + \sum_{j=1}^{k} \mu_j^* \nabla_{\boldsymbol{w}} G(\boldsymbol{w}^*, \boldsymbol{x}_j^*) = \boldsymbol{0} \quad (5.43)$$

with Lagrange function

$$L(\boldsymbol{w}, \boldsymbol{\mu}) := F(\boldsymbol{w}) + \sum_{j=1}^{k} \mu_j G(\boldsymbol{w}, \boldsymbol{x}_j). \quad (5.44)$$

Proof. A proof can be found for example in [Lopez 2007][Theorem 2]. ∎

It is important to note, a nice conclusion of this theorem is that only a finite number $k \leq L$ of elements $(\boldsymbol{x}_i^*, \mu_i^*) \in \mathcal{T} \times \mathbb{R}^+$ characterize an optimal point. Also an upper bound on their quantity is given. Unfortunately, one does not know the points $\boldsymbol{x}_i^* \in \mathcal{T}$ in advance.

Without further information, e.g. second-order optimality, it is impossible in case of a general nonlinear SIP to derive sufficient conditions for $\boldsymbol{w}^*$ to be optimal. But in the particular case of a convex SIP, i.e. the objective function F and the set $\mathcal{S}_{SIP}$ are both convex, the KKT condition is necessary and sufficient for a (global) minimizer $\boldsymbol{w}^* \in \mathcal{S}_{SIP}$:

Theorem 5.2.2 (Sufficient KKT Optimality Condition, e.g. [Lopez 2007])**.** *Let $(\boldsymbol{w}^*, \boldsymbol{\mu}^*) \in \mathcal{S}_{SIP} \times \mathbb{R}^k$ be a solution of the KKT-condition (5.43) of a convex SIP (Def. 5.2.3) with $\mathbf{0} \leq (\mu_1^*, \ldots, \mu_k^*) = \boldsymbol{\mu}^*$ and $\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_k^* \in \mathcal{T}^0(\boldsymbol{w}^*)$, $k \leq L$. Then $\boldsymbol{w}^*$ is a (global) minimizer.*

Proof. A proof can be found for example in [Lopez 2007][Theorem 3]. ∎

Usually, the MFCQ is not easy to prove for general feasible sets. But in case of convex SIP there is also a more handy constraint qualification implying MFCQ:

Definition 5.2.6 (SCQ, e.g. [Lopez 2007])**.** *The Slater constraint qualification (SCQ) holds, iff there exists $\boldsymbol{w} \in \mathcal{S}_{SIP}$ such that $G(\boldsymbol{w}, \boldsymbol{x}) < 0 \ \forall \boldsymbol{x} \in \mathcal{T}$.*

That SCQ indeed implies MFCQ is stated in the following lemma:

Lemma 5.2.2. *Let a convex SIP (Def. 5.2.3) satisfy SCQ (Def. 5.2.6) then MFCQ (Def. 5.2.5) holds at every $\boldsymbol{w} \in \mathcal{S}_{SIP}$.*

Proof. Suppose SCQ is satisfied at $\hat{\boldsymbol{w}} \in \mathcal{S}_{SIP}$. Then, using convexity of $G(\cdot, \boldsymbol{x})$, for any $\boldsymbol{w} \in \mathcal{S}_{SIP}$ holds

$$\nabla_{\boldsymbol{w}} G(\boldsymbol{w}, \boldsymbol{x})^T (\hat{\boldsymbol{w}} - \boldsymbol{w}) \leq G(\hat{\boldsymbol{w}}, \boldsymbol{x}) - G(\boldsymbol{w}, \boldsymbol{x}) < 0 \ \forall \boldsymbol{x} \in \mathcal{T}^0(\boldsymbol{w}) \neq \emptyset \tag{5.45}$$

Choosing $\boldsymbol{d} := (\hat{\boldsymbol{w}} - \boldsymbol{w})$ satisfies MFCQ. If $\mathcal{T}^0(\boldsymbol{w})$ is empty, then MFCQ is trivially satisfied. ∎

5.2.2.2 Primal SIP of the Lipschitz Classifier

The mathematical tools from semi-infinite programming have been introduced in the preceding section. Because of its importance for this work, we first summarize the technical results in following corollary:

Corollary 5.2.1 (Necessary and sufficient KKT conditions for convex SIP)**.** *Suppose SCQ holds for a convex SIP (Def. 5.2.3). Then $\boldsymbol{w}^* \in \mathbb{R}^L$ is a (global) optimal point, iff there exist $\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_k^* \in \mathcal{T}$, $k \leq L$ and $(\mu_1^*, \ldots, \mu_k^*) = \boldsymbol{\mu}^* \geq \mathbf{0}$ such that $(\boldsymbol{w}^*, \boldsymbol{\mu}^*) \in \mathbb{R}^L \times \mathbb{R}^k$ is a solution of the following system of equations,*

$$\nabla F(\boldsymbol{w}^*) + \sum_{j=1}^{k} \mu_j^* \nabla_{\boldsymbol{w}} G(\boldsymbol{w}^*, \boldsymbol{x}_j^*) = \mathbf{0} \tag{5.46}$$

$$\forall \boldsymbol{x} \in \mathcal{T} \ : \ G(\boldsymbol{w}^*, \boldsymbol{x}) \leq 0 \tag{5.47}$$

$$\boldsymbol{\mu}^* \geq \mathbf{0} \tag{5.48}$$

$$\sum_{j=1}^{k} \mu_j^* G(\boldsymbol{w}^*, \boldsymbol{x}_j^*) = 0. \tag{5.49}$$

Proof. The proof follows directly from Theorem (5.2.1), Theorem (5.2.2) and Lemma (5.2.2). ∎

In order to reformulate the minimax problem of the Lipschitz classifier (5.34) aiming to apply standard convex optimization methods, we have only to verify the relationship to a convex SIP and to use Corollary (5.2.1).

This relationship is formalized in the following lemma:

Lemma 5.2.3 ([Stuhlsatz 2008b]). *Let be $\mathcal{X} \subset \mathbb{R}^m$ infinite and compact, $\emptyset \neq \mathcal{Y} \subset \mathbb{R}^m$ compact such that $\mathcal{X} \cap \mathcal{Y} = \emptyset$, $\mathcal{T} := \mathcal{X} \cup \mathcal{Y}$, $f : \mathbb{R}^l \to \mathbb{R}$, $g(\cdot, \boldsymbol{x}) : \mathbb{R}^l \to \mathbb{R}$ $\forall \boldsymbol{x} \in \mathcal{T}$ convex and continuously differentiable and $g(\boldsymbol{z}, \cdot) : \mathcal{T} \to \mathbb{R}$ $\forall \boldsymbol{z} \in \mathbb{R}^l$ continuous. Suppose SCQ holds for the convex feasible set*

$$\mathcal{S}_P := \left\{ \boldsymbol{z} \in \mathbb{R}^l \;\middle|\; g(\boldsymbol{z}, \boldsymbol{v}) \leq 0 \;\forall \boldsymbol{v} \in \mathcal{Y} \right\}. \tag{5.50}$$

Further, define $L := 1 + l$, $\boldsymbol{w} := (z_0, \boldsymbol{z}) \in \mathbb{R}^L$, the constraint functions

$$G(\boldsymbol{w}, \boldsymbol{x}) := \begin{cases} g(\boldsymbol{z}, \boldsymbol{x}) & \text{if } \boldsymbol{x} \in \mathcal{Y} \\ g(\boldsymbol{z}, \boldsymbol{x}) - z_0 & \text{if } \boldsymbol{x} \in \mathcal{X} \end{cases}, \tag{5.51}$$

the objective function

$$F(\boldsymbol{w}) := z_0 + f(\boldsymbol{z}) \tag{5.52}$$

and the set

$$\mathcal{H} := \left\{ g(\cdot, \boldsymbol{x}) : \mathbb{R}^l \to \mathbb{R} \;\middle|\; \boldsymbol{x} \in \mathcal{X} \right\}. \tag{5.53}$$

Then there exist a point $\boldsymbol{w}^* := (z_0^*, \boldsymbol{z}^*) \in \mathcal{S}_{SIP}$ and a function $h^* \in conv(\mathcal{H})$ (Def. 5.2.4) with

$$z_0^* = \max_{\boldsymbol{x} \in \mathcal{X}} g(\boldsymbol{z}^*, \boldsymbol{x}) = h^*(\boldsymbol{z}^*). \tag{5.54}$$

Moreover, it follows the convex SIP version of the minimax problem:

$$\begin{aligned} z_0^* + f(\boldsymbol{z}^*) &= \min_{(z_0, \boldsymbol{z}) \in \mathcal{S}_{SIP}} z_0 + f(\boldsymbol{z}) && (5.55) \\ &= \min_{\boldsymbol{z} \in \mathcal{S}_P} \max_{\boldsymbol{x} \in \mathcal{X}} g(\boldsymbol{z}, \boldsymbol{x}) + f(\boldsymbol{z}). && (5.56) \end{aligned}$$

Proof. Solutions $(\boldsymbol{z}^*, \boldsymbol{x}^*) \in \mathcal{S}_P \times \mathcal{X}$ of (5.56) are guaranteed by the compactness of $\mathcal{X} \neq \emptyset$, the continuity of $g(\boldsymbol{z}, \cdot)$, the convexity of the functions f, $g(\cdot, \boldsymbol{x})$ and the convexity of the set $\mathcal{S}_P \neq \emptyset$. Because $\mathcal{X} \cap \mathcal{Y} = \emptyset$ implies $dist(\mathcal{X}, \mathcal{Y}) > 0$, it follows $G(\boldsymbol{w}, \cdot)$ is continuous.

Consider a point $\boldsymbol{z} \in \mathcal{S}_P$, which satisfies the SCQ for $\mathcal{S}_P$. Then, with $z_0 := \max_{\boldsymbol{x} \in \mathcal{X}} g(\boldsymbol{z}, \boldsymbol{x}) + 1$, we conclude $G(\boldsymbol{w}, \boldsymbol{x}) = g(\boldsymbol{z}, \boldsymbol{x}) - z_0 \leq -1 < 0$ $\forall \boldsymbol{x} \in \mathcal{X}$, and thus SCQ holds also for $\mathcal{S}_{SIP}$.

Using Corollary (5.2.1), there exist $\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_k^* \in \mathcal{X}, \boldsymbol{v}_1^*, \ldots, \boldsymbol{v}_n^* \in \mathcal{Y}$, $k + n \leq L = l + 1$ and $(\mu_1^*, \ldots, \mu_k^*) = \boldsymbol{\mu} \geq \boldsymbol{0}, (\lambda_1^*, \ldots, \lambda_n^*) =: \boldsymbol{\lambda}^* \geq \boldsymbol{0}$, such that

a solution $\boldsymbol{w}^* = (z_0^*, \boldsymbol{z}^*) \in \mathcal{S}_{SIP}$ of the convex SIP (5.55) admits a solution $(z_0^*, \boldsymbol{z}^*, \boldsymbol{\mu}^*, \boldsymbol{\lambda}^*) \in \mathbb{R} \times \mathbb{R}^l \times \mathbb{R}^k \times \mathbb{R}^n$ of the KKT-system

$$1 - \sum_{j=1}^{k} \mu_j^* = 0, \boldsymbol{\mu}^* \geq \mathbf{0}, \boldsymbol{\lambda}^* \geq \mathbf{0} \tag{5.57}$$

$$\nabla f(\boldsymbol{z}^*) + \sum_{j=1}^{k} \mu_j^* \nabla_{\boldsymbol{z}} g(\boldsymbol{z}^*, \boldsymbol{x}_j^*) + \sum_{i=1}^{n} \lambda_i^* \nabla_{\boldsymbol{z}} g(\boldsymbol{z}^*, \boldsymbol{v}_i^*) = \mathbf{0} \tag{5.58}$$

$$\forall \boldsymbol{x} \in \mathcal{X} \ : \ g(\boldsymbol{z}^*, \boldsymbol{x}) - z_0^* \leq 0 \tag{5.59}$$

$$\forall \boldsymbol{v} \in \mathcal{Y} \ : \ g(\boldsymbol{z}^*, \boldsymbol{v}) \leq 0 \tag{5.60}$$

$$\sum_{j=1}^{k} \mu_j^* (g(\boldsymbol{z}^*, \boldsymbol{x}_j^*) - z_0^*) + \sum_{i=1}^{n} \lambda_i^* g(\boldsymbol{z}^*, \boldsymbol{v}_i^*) = 0. \tag{5.61}$$

Because of (5.57), it follows $\mathcal{T}^0(\boldsymbol{w}^*) \neq \emptyset$. Using (5.61), we obtain $\boldsymbol{x}_j^* \in \mathcal{T}^0(\boldsymbol{w}^*) \subseteq \mathcal{T}$ for all $\boldsymbol{x}_j^* \in \mathcal{X}$ with $\mu_j^* > 0$. Therefore, it follows from (5.59) that

$$\forall \boldsymbol{x} \in \mathcal{X}, \boldsymbol{x}^* \in \mathcal{T}^0(\boldsymbol{w}^*) \ : \ g(\boldsymbol{z}^*, \boldsymbol{x}) \leq z_0^* = g(\boldsymbol{z}^*, \boldsymbol{x}^*). \tag{5.62}$$

In particular, it holds

$$\forall \boldsymbol{x}^* \in \mathcal{T}^0(\boldsymbol{w}^*) \ : \ \max_{\boldsymbol{x} \in \mathcal{X}} g(\boldsymbol{z}^*, \boldsymbol{x}) = z_0^* = g(\boldsymbol{z}^*, \boldsymbol{x}^*). \tag{5.63}$$

Defining $h^* \in conv(\mathcal{H})$ by

$$\forall \boldsymbol{z} \in I\!R^l : \ h^*(\boldsymbol{z}) := \sum_{j=1}^{k} \mu_j^* g(\boldsymbol{z}, \boldsymbol{x}_j^*), \tag{5.64}$$

and using (5.57),(5.61) and (5.63), it follows our first statement

$$\max_{\boldsymbol{x} \in \mathcal{X}} g(\boldsymbol{z}^*, \boldsymbol{x}) = g(\boldsymbol{z}^*, \boldsymbol{x}^*) = z_0^* = h^*(\boldsymbol{z}^*). \tag{5.65}$$

For any $\boldsymbol{z} \in \mathcal{S}_P$ choose $\hat{z}_0 := \max_{\boldsymbol{x} \in \mathcal{X}} g(\boldsymbol{z}, \boldsymbol{x})$. Because $(\hat{z}_0, \boldsymbol{z}) \in \mathcal{S}_{SIP}$, it holds

$$\begin{aligned} z_0^* + f(\boldsymbol{z}^*) &= \min_{(z_0, \boldsymbol{z}) \in \mathcal{S}_{SIP}} z_0 + f(\boldsymbol{z}) && (5.66) \\ &\leq \hat{z}_0 + f(\boldsymbol{z}) = \max_{\boldsymbol{x} \in \mathcal{X}} g(\boldsymbol{z}, \boldsymbol{x}) + f(\boldsymbol{z}). && (5.67) \end{aligned}$$

Taking the minimum over all $\boldsymbol{z} \in \mathcal{S}_P$ and using (5.63), it follows our the second statement for $\boldsymbol{z}^* \in \mathcal{S}_P$:

$$\begin{aligned} z_0^* + f(\boldsymbol{z}^*) &= \min_{(z_0, \boldsymbol{z}) \in \mathcal{S}_{SIP}} z_0 + f(\boldsymbol{z}) && (5.68) \\ &\leq \min_{\boldsymbol{z} \in \mathcal{S}_P} \max_{\boldsymbol{x} \in \mathcal{X}} g(\boldsymbol{z}, \boldsymbol{x}) + f(\boldsymbol{z}) && (5.69) \\ &\leq \max_{\boldsymbol{x} \in \mathcal{X}} g(\boldsymbol{z}^*, \boldsymbol{x}) + f(\boldsymbol{z}^*) = z_0^* + f(\boldsymbol{z}^*). && (5.70) \end{aligned}$$

■

The feasible set $\mathcal{S}_P$ of the minimax version of the Lipschitz classifier (5.40) satisfies SCQ trivially. For example choose $\hat{\boldsymbol{z}} := (\mathbf{0}, \pi\mathbf{1}_N, 0) \in \mathcal{S}_P$. Next, we choose an arbitrary infinite compact set $\mathcal{X} \subset \mathbb{R}^m$ and $2N$ points $\mathcal{Y} := \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{2N}\}$ with $\boldsymbol{v}_j \in \mathbb{R}^m \setminus \mathcal{X}$, $\boldsymbol{v}_j \neq \boldsymbol{v}_i \; \forall i \neq j$. Then, we identify by inspection of (5.34) and (5.40), that

$$l := M + N + 1, \tag{5.71}$$

$$\boldsymbol{z} := (\boldsymbol{c}, \boldsymbol{\xi}, b) \in \mathbb{R}^l, \tag{5.72}$$

$$f(\boldsymbol{z}) := C\mathbf{1}_N^T\boldsymbol{\xi}, \tag{5.73}$$

$$(g(\boldsymbol{z}, \boldsymbol{v}_1), \ldots, g(\boldsymbol{z}, \boldsymbol{v}_N))^T := -\boldsymbol{\xi}, \tag{5.74}$$

$$(g(\boldsymbol{z}, \boldsymbol{v}_{N+1}), \ldots, g(\boldsymbol{z}, \boldsymbol{v}_{2N}))^T := -\mathbf{Y}\mathbf{G}\boldsymbol{c} - \boldsymbol{y}b + \mathbf{1}_N - \boldsymbol{\xi}, \tag{5.75}$$

$$g(\boldsymbol{z}, \boldsymbol{x}) := \frac{1}{2}\boldsymbol{c}^T\mathbf{K}(\boldsymbol{x})\boldsymbol{c} \; \forall \boldsymbol{x} \in \mathcal{X} \tag{5.76}$$

have to be defined in order to apply lemma (5.2.3) and to yield a primal convex SIP version of the Lipschitz classifier (Alg. 5.2.3). Moreover, in virtue of corollary (5.2.1), we obtain an equation system (5.57)-(5.61), which is necessary and sufficient for optimality and reminds of the KKT systems obtained from problems in finite convex settings (App. A.3.3). This equation system is also known as *primal KKT-system* associated to the *primal SIP* (5.39), and a solution is called *primal solution*.

Algorithm 5.2.3 $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz Classifier (Primal Convex SIP Version)

Require: Training set $\emptyset \neq \mathcal{O}_N \subset \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, $\mathcal{X} \subset \mathbb{R}^m$ compact and convex, $1 \leq n \leq M$ basis functions $\Phi_n(\boldsymbol{x}) \in \mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$, $C > 0$

Ensure: $\exists(\boldsymbol{x}_j, y_j), (\boldsymbol{x}_i, y_i) \in \mathcal{O}_N \; : \; y_j = -1 \wedge y_i = 1$

$$\min_{(z_0, \boldsymbol{c}, \boldsymbol{\xi}, b) \in \mathcal{S}_{SIP}} z_0 + C\mathbf{1}_N^T\boldsymbol{\xi} \tag{5.77}$$

with feasible set

$$\mathcal{S}_{SIP} := \left\{ (z_0, \boldsymbol{c}, \boldsymbol{\xi}, b) \in \mathbb{R} \times \mathbb{R}^M \times \mathbb{R}^N \times \mathbb{R} \;\middle|\; \mathbf{Y}\mathbf{G}\boldsymbol{c} + \boldsymbol{y}b - \mathbf{1}_N + \boldsymbol{\xi} \geq \mathbf{0}, \right.$$
$$\left. \boldsymbol{\xi} \geq \mathbf{0}, \frac{1}{2}\boldsymbol{c}^T\mathbf{K}(\boldsymbol{x})\boldsymbol{c} - z_0 \leq 0 \; \forall \boldsymbol{x} \in \mathcal{X} \right\} \tag{5.78}$$

The novel primal convex SIP version of the Lipschitz classifier enables to solve the problem by a sufficient fine discretization of the set $\mathcal{X}$ with a Newton-type optimization of the corresponding KKT system [Stuhlsatz 2008c]. This is possible, because given a grid $\hat{\mathcal{X}} \subset \mathcal{X}$ the infinite number of convex inequalities

$\frac{1}{2}\boldsymbol{c}^T\mathbf{K}(\boldsymbol{x})\boldsymbol{c} - z_0 \leq 0 \; \forall \boldsymbol{x} \in \mathcal{X}$ reduces to finitely many inequalities $\frac{1}{2}\boldsymbol{c}^T\mathbf{K}(\boldsymbol{x}_i)\boldsymbol{c} - z_0 \leq 0 \; \forall \boldsymbol{x}_i \in \hat{\mathcal{X}}$. One can show [Hettich 1993], in case of a convex SIP the optimal solutions using stepwise refined grids tend in the limit to the optimal solution of the continuous problem. Thus, in order to expect an accurate solution, a fine discretization grid should be chosen. On the other hand, discretizing the primal SIP is technically not satisfactory due to two reasons: First, the discrete primal problem has a very complicated feasible set breaking down standard solvers. Second, it involves many matrix evaluations which all have to be stored during optimization. This requires prohibitive storage capacities. It follows, that discretization is extremely impractical for higher dimensional spaces due to the exponentially growing number of grid points with increasing number of dimensions of the data space $\mathcal{X}$. Hence, one is in favor to apply some kind of duality as it is used for example for solving the SVM more efficiently (Sec. 3.6).

5.2.2.3 Dual SIP of the Lipschitz Classifier

Recall the *Lagrange function* $L : \mathbb{R}^L \times \mathbb{R}^k \to \mathbb{R}$ with

$$L(\boldsymbol{w}, \boldsymbol{\mu}) = F(\boldsymbol{w}) + \sum_{j=1}^{k} \mu_j G_j(\boldsymbol{w}), \;\; F : \mathbb{R}^L \to \mathbb{R}, \;\; G_j : \mathbb{R}^L \to \mathbb{R} \tag{5.79}$$

introduced in Theorem (5.2.1) of the optimality conditions for general semi-infinite programming problems, respectively with $G_j(\boldsymbol{w}) := G(\boldsymbol{w}, \boldsymbol{x}_j), \;\; \boldsymbol{x}_j \in \mathcal{T}$. The weights $(\mu_1, \ldots, \mu_k)^T =: \boldsymbol{\mu} \in \mathbb{R}^k$ are called *Lagrange multipliers*.

In optimization theory, the *dual* of a Lagrangian (5.79) is defined as

Definition 5.2.7 (Dual of a Lagrange function)**.** *A function* $\mathcal{D}L_{\mathcal{S}} : \mathbb{R}^k \to \mathbb{R}$ *with*

$$\mathcal{D}L_{\mathcal{S}}(\boldsymbol{\mu}) := \inf_{\boldsymbol{w} \in \mathbb{R}^L} L(\boldsymbol{w}, \boldsymbol{\mu}) \tag{5.80}$$

is called the dual of the Lagrange function

$$L(\boldsymbol{w}, \boldsymbol{\mu}) = F(\boldsymbol{w}) + \sum_{j=1}^{k} \mu_j G_j(\boldsymbol{w}), \;\; \mu_j \in \mathbb{R}. \tag{5.81}$$

In particular, the dual is a concave function on the convex domain

$$\left\{ \boldsymbol{\mu} \in \mathbb{R}^k \;\; : \;\; \boldsymbol{\mu} \geq \mathbf{0}, \mathcal{D}L_{\mathcal{S}}(\boldsymbol{\mu}) > -\infty \right\}, \tag{5.82}$$

(cf. App. A.3.4).

Definition 5.2.8 (Dual problem)**.** *The problem*

$$\sup_{\boldsymbol{\mu}\geq\mathbf{0}} \mathcal{D}L_{\mathcal{S}}(\boldsymbol{\mu}) \tag{5.83}$$

is called the dual problem associated to the primal problem

$$\inf\{F(\boldsymbol{w}) \ : \ \boldsymbol{w}\in\mathcal{S}\} \tag{5.84}$$

with feasible set $\mathcal{S} := \left\{\boldsymbol{w}\in\mathbb{R}^L \ : \ G_j(\boldsymbol{w})\leq 0 \ \forall 1\leq j\leq k\right\}$.

It is important to note, that always *weak duality* holds

$$\sup_{\boldsymbol{\mu}\geq\mathbf{0}} \mathcal{D}L_{\mathcal{S}}(\boldsymbol{\mu}) \leq \inf_{\boldsymbol{w}\in\mathcal{S}} F(\boldsymbol{w}). \tag{5.85}$$

Thus, the primal optimal value is lower bounded by the dual optimal value (cf. App. A.3.4). In the particular situation, that equality in (5.85) holds throughout, namely *strong duality* holds (cf. App. A.3.4), then one can solve alternatively the dual instead of the primal problem.

In the following, we prove a strong duality theorem for convex SIP using Lemma (5.2.3) [Stuhlsatz 2008b], [Stuhlsatz 2008c]. The strong duality theorem interconnects the minimax formulation of the Lipschitz classifier (5.34) with a dual SIP via the primal convex SIP (5.77):

Theorem 5.2.3 (Strong Duality, [Stuhlsatz 2008b])**.** *Suppose all preliminaries and assumptions of lemma (5.2.3) are satisfied.*

Then strong duality holds with

$$\min_{\boldsymbol{z}\in\mathcal{S}_P}\max_{\boldsymbol{x}\in\mathcal{X}} g(\boldsymbol{z},\boldsymbol{x}) + f(\boldsymbol{z}) = \max_{h\in conv(\mathcal{H})}\min_{\boldsymbol{z}\in\mathcal{S}_P} h(\boldsymbol{z}) + f(\boldsymbol{z}) \tag{5.86}$$

and a solution $\boldsymbol{z}^*\in\mathcal{S}_P$ *is primal and dual optimal point.*

Proof. Choosing any $\boldsymbol{z}\in\mathbb{R}^l$ and $h\in conv(\mathcal{H})$, by definition of the convex hull (Def. 5.2.4), there are $k\in\mathbb{N}$ points $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_k\in\mathcal{X}$ and $\mu_1,\ldots,\mu_k\geq 0$ with

$$h(\boldsymbol{z}) = \sum_{j=1}^{k}\mu_j g(\boldsymbol{z},\boldsymbol{x}_j) \leq \underbrace{\left(\sum_{j=1}^{k}\mu_j\right)}_{=1}\max_{\boldsymbol{x}\in\mathcal{X}} g(\boldsymbol{z},\boldsymbol{x}). \tag{5.87}$$

Taking the minimum of the convex Lagrangian $L(\cdot,h) = h+f$ with respect to the convex set $\mathcal{S}_P$, we get

$$\min_{\boldsymbol{z}\in\mathcal{S}_P} h(\boldsymbol{z}) + f(\boldsymbol{z}) \leq \min_{\boldsymbol{z}\in\mathcal{S}_P}\max_{\boldsymbol{x}\in\mathcal{X}} g(\boldsymbol{z},\boldsymbol{x}) + f(\boldsymbol{z}) \tag{5.88}$$

and in particular

$$\sup_{h\in conv(\mathcal{H})} \min_{\boldsymbol{z}\in\mathcal{S}_P} h(\boldsymbol{z}) + f(\boldsymbol{z}) \leq \min_{\boldsymbol{z}\in\mathcal{S}_P} \max_{\boldsymbol{x}\in\mathcal{X}} g(\boldsymbol{z},\boldsymbol{x}) + f(\boldsymbol{z}). \tag{5.89}$$

Using the optimal point $\boldsymbol{w}^* := (z_0^*, \boldsymbol{z}^*)$ of lemma (5.2.3), the right-hand side of the last inequality equals $z_0^* + f(\boldsymbol{z}^*)$. Hence using (5.54), we obtain

$$\sup_{h\in conv(\mathcal{H})} \min_{\boldsymbol{z}\in\mathcal{S}_P} h(\boldsymbol{z}) + f(\boldsymbol{z}) \leq h^*(\boldsymbol{z}^*) + f(\boldsymbol{z}^*). \tag{5.90}$$

Because $h^* \in conv(\mathcal{H})$ and $\boldsymbol{z}^* \in \mathcal{S}_P$, the supremum is attained with

$$\max_{h\in conv(\mathcal{H})} \min_{\boldsymbol{z}\in\mathcal{S}_P} h(\boldsymbol{z}) + f(\boldsymbol{z}) = \min_{\boldsymbol{z}\in\mathcal{S}_P} \max_{\boldsymbol{x}\in\mathcal{X}} g(\boldsymbol{z},\boldsymbol{x}) + f(\boldsymbol{z}). \tag{5.91}$$

This also implies that $\boldsymbol{z}^* \in \mathcal{S}_P$ is primal and dual optimal point.

∎

Theorem (5.2.3) is a very nice result. Because the parametrization of the original minimax problem as a max-min problem over the convex hull of functions $h \in \mathcal{H}$ opens a direction to solve the problem iteratively without any discretization:

Theorem 5.2.4 (Solution of the $\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz Classifier without discretization)**.** *Any solution $\boldsymbol{c}^*$ of the system of equations $\mathbf{K}^*\boldsymbol{c} = \mathbf{G}^T\mathbf{Y}\boldsymbol{\alpha}^*$ associated to a dual optimal point $(\mathbf{K}^*, \boldsymbol{\alpha}^*) \in conv(\mathcal{K})\times\mathbb{R}^N$ of the dual $\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz Classifier problem*

$$\max_{\mathbf{K}\in conv(\mathcal{K})} \left(\max_{\boldsymbol{\alpha},\boldsymbol{\beta}\geq\mathbf{0}\ :\ \boldsymbol{\alpha}^T\boldsymbol{y}=0, C\mathbf{1}_N-\boldsymbol{\alpha}-\boldsymbol{\beta}=0} \boldsymbol{\alpha}^T\mathbf{1}_N - \frac{1}{2}(\boldsymbol{c}^*)^T\mathbf{K}(\boldsymbol{c}^*) \right) \tag{5.92}$$

admits the same optimal value

$$\begin{aligned} (\boldsymbol{\alpha}^*)^T\mathbf{1}_N - \frac{1}{2}(\boldsymbol{c}^*)^T(\mathbf{K}^*)(\boldsymbol{c}^*) &= \min_{(\boldsymbol{c},\boldsymbol{\xi},b)\in\mathcal{S}_P} \max_{\boldsymbol{x}\in\mathcal{X}} \frac{1}{2}\boldsymbol{c}^T\mathbf{K}(\boldsymbol{x})\boldsymbol{c} + C\mathbf{1}_N^T\boldsymbol{\xi} \\ &= \frac{1}{2}(\boldsymbol{c}^*)^T\mathbf{K}(\boldsymbol{x}^*)(\boldsymbol{c}^*) + C\mathbf{1}_N^T\boldsymbol{\xi}^*. \end{aligned} \tag{5.93}$$

Any $\boldsymbol{c}^*$ is a primal feasible solution of the $\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz Classifier.

Proof. First, recall definition (5.76). Then for any $h \in \mathcal{H}$ it holds

$$h(\boldsymbol{z}) = \sum_{j=1}^{k} \mu_j g(\boldsymbol{z},\boldsymbol{x}_j) = \frac{1}{2}\boldsymbol{c}^T \left(\sum_{j=1}^{k} \mu_j \mathbf{K}(\boldsymbol{x}_j) \right) \boldsymbol{c} = \frac{1}{2}\boldsymbol{c}^T\mathbf{K}\boldsymbol{c} \tag{5.94}$$

with $\mathbf{K} \in conv(\mathcal{K})$. $\mathcal{K}$ denotes the set of all positive semi-definite matrices $\mathbf{K}(\boldsymbol{x})$ with $\mathbf{K}(\boldsymbol{x})_{m,n} := \langle \nabla\Phi_m(\boldsymbol{x}), \nabla\Phi_n(\boldsymbol{x})\rangle_2$ defined on $\mathcal{X}$, i.e.

$$\mathcal{K} := \left\{ 0 \preceq \mathbf{K}(\boldsymbol{x}) \in \mathbb{R}^{M\times M} \mid \boldsymbol{x} \in \mathcal{X} \right\}. \tag{5.95}$$

Defining the Lagrangian

$$\begin{aligned} L_{\mathcal{S}_P}(\boldsymbol{z}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{K}) &:= h(\boldsymbol{z}) + f(\boldsymbol{z}) - \boldsymbol{\beta}^T\boldsymbol{\xi} - \boldsymbol{\alpha}^T\left(\mathbf{YG}\boldsymbol{c} + \boldsymbol{y}b - \mathbf{1}_N + \boldsymbol{\xi}\right) \\ &= \frac{1}{2}\boldsymbol{c}^T\mathbf{K}\boldsymbol{c} - \boldsymbol{\alpha}^T\boldsymbol{y}\cdot b - \boldsymbol{\alpha}^T\mathbf{YG}\boldsymbol{c} + \boldsymbol{\alpha}^T\mathbf{1}_N + \\ &\quad \boldsymbol{\xi}^T\left(C\mathbf{1}_N - \boldsymbol{\alpha} - \boldsymbol{\beta}\right) \end{aligned} \tag{5.96}$$

with $h \in \mathcal{H}$ and $f(\boldsymbol{z})$ defined as in (5.73), yields the dual

$$\mathcal{D}L_{\mathcal{S}_P}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{K}) := \inf\left\{ L_{\mathcal{S}_P}(\boldsymbol{z}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{K}) \mid \boldsymbol{z} \in \mathbb{R}^l \right\}. \tag{5.97}$$

The infimum is attained for all $(\boldsymbol{z}^*, \boldsymbol{\alpha}, \boldsymbol{\beta}) \in \mathbb{R}^{l+N+M}$ satisfying the constraints $\mathbf{K}\boldsymbol{c}^* - \mathbf{G}^T\mathbf{Y}\boldsymbol{\alpha} = 0$, $\boldsymbol{\alpha}^T\boldsymbol{y} = 0$ and $C\mathbf{1}_N - \boldsymbol{\alpha} - \boldsymbol{\beta} = \mathbf{0}$ with optimal value

$$\mathcal{D}L_{\mathcal{S}_P}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{K}) = L_{\mathcal{S}_P}(\boldsymbol{z}^*, \boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{K}) = \boldsymbol{\alpha}^T\mathbf{1}_N - \frac{1}{2}(\boldsymbol{c}^*)^T\mathbf{K}(\boldsymbol{c}^*). \tag{5.98}$$

From (5.97), the identity (5.94) and Definition (5.73) follows immediately in virtue of the strong duality theorem of finite convex programming (App. A.3.4) the duality

$$\max_{\boldsymbol{\alpha},\boldsymbol{\beta}\geq\mathbf{0}\,:\,\boldsymbol{\alpha}^T\boldsymbol{y}=0, C\mathbf{1}_N-\boldsymbol{\alpha}-\boldsymbol{\beta}=0} \mathcal{D}L_{\mathcal{S}_P}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{K}) = \min_{\boldsymbol{z}\in\mathcal{S}_P} h(\boldsymbol{z}) + f(\boldsymbol{z}). \tag{5.99}$$

Now, applying Theorem (5.2.3) to (5.99) gives

$$\begin{aligned} & \frac{1}{2}(\boldsymbol{c}^*)^T\mathbf{K}(\boldsymbol{x}^*)(\boldsymbol{c}^*) + C\mathbf{1}_N^T\boldsymbol{\xi}^* \\ = \; & \min_{\boldsymbol{z}\in\mathcal{S}_P}\max_{\boldsymbol{x}\in\mathcal{X}} \frac{1}{2}\boldsymbol{c}^T\mathbf{K}(\boldsymbol{x})\boldsymbol{c} + C\mathbf{1}_N^T\boldsymbol{\xi} \\ = \; & \max_{\mathbf{K}\in conv(\mathcal{K})} \left(\max_{\boldsymbol{\alpha},\boldsymbol{\beta}\geq\mathbf{0}\,:\,\boldsymbol{\alpha}^T\boldsymbol{y}=0, C\mathbf{1}_N-\boldsymbol{\alpha}-\boldsymbol{\beta}=0} \mathcal{D}L_{\mathcal{S}_P}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{K}) \right) \\ = \; & (\boldsymbol{\alpha}^*)^T\mathbf{1}_N - \frac{1}{2}(\boldsymbol{c}^*)^T(\mathbf{K}^*)(\boldsymbol{c}^*) \end{aligned} \tag{5.100}$$

Because any solution $\boldsymbol{c}^*$ can be represented as $\boldsymbol{c}^* = \boldsymbol{c}(\boldsymbol{\alpha}^*) + \lambda\boldsymbol{c}_0$ for any $\boldsymbol{c}_0 \in null(\mathbf{K}^*)$, i.e. the null space of $\mathbf{K}^*$, and $\lambda \in \mathbb{R}$, one easily verifies from (5.100) that the null space does not change the optimal value. The last statement follows directly from the proof of Theorem (5.2.3).

■

Making the constraint $\mathbf{K}\boldsymbol{c}^* = \mathbf{G}^T\mathbf{Y}\boldsymbol{\alpha}$ for attaining the infimum in (5.97) explicit, we obtain by Theorem (5.2.4) the dual formulation of the Lipschitz classifier algorithm:

Algorithm 5.2.4 $\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz Classifier (Dual Convex SIP Version)

Require: Training set $\emptyset \neq \mathcal{O}_N \subset \mathcal{Z} = \mathcal{X}\times\mathcal{Y}$, $\mathcal{X}\subset\mathbb{R}^m$ compact and convex, $1 \leq n \leq M$ basis functions $\Phi_n(\boldsymbol{x}) \in \mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$, $C > 0$

Ensure: $\exists(\boldsymbol{x}_j, y_j), (\boldsymbol{x}_i, y_i) \in \mathcal{O}_N \ : \ y_j = -1 \wedge y_i = 1$

$$\max_{\mathbf{K}\in conv(\mathcal{K})} \max_{(\boldsymbol{\alpha},\boldsymbol{c})\in\mathcal{S}_D(\mathbf{K})} \boldsymbol{\alpha}^T\mathbf{1}_N - \frac{1}{2}\boldsymbol{c}^T\mathbf{K}\boldsymbol{c} \tag{5.101}$$

with feasible set

$$\mathcal{S}_D(\mathbf{K}) := \left\{ (\boldsymbol{\alpha},\boldsymbol{c}) \in \mathbb{R}^N \times \mathbb{R}^M \;\middle|\; \mathbf{K}\boldsymbol{c} = \mathbf{G}^T\mathbf{Y}\boldsymbol{\alpha}, \boldsymbol{\alpha}^T\boldsymbol{y} = 0, \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}_N \right\} \tag{5.102}$$

If an optimal point $(\mathbf{K}^*, \boldsymbol{\alpha}^*, \boldsymbol{c}^*)$ of (5.101) is found, one can compute the optimal bias b^* of the decision function $f(\boldsymbol{x}) := \sum_{n=1}^M c_n^*\Phi_n(\boldsymbol{x}) + b^*$ by taking the KKT conditions (5.57)-(5.61) into account. From the KKT conditions follows, that for each $0 < \alpha_i^* < C$ holds $y_i\sum_{n=1}^M c_n^*\Phi_n(\boldsymbol{x}_i) + by_i - 1 + \xi_i^* = 0$ and $\xi_i^* = 0$. Therefore, $b = y_i - \sum_{n=1}^M c_n^*\Phi_n(\boldsymbol{x}_i)$. For example, taking the average over equally sized index sets $\mathcal{A}_+ \subseteq \{i \in \mathbb{N} \ : \ 0 < \alpha_i < C, y_i = 1\}$ and $\mathcal{A}_- \subseteq \{j \in \mathbb{N} \ : \ 0 < \alpha_j < C, y_j = -1\}$ with $|\mathcal{A}_+| = |\mathcal{A}_-|$, yields an estimate of b^*:

$$b^* = \frac{1}{2|\mathcal{A}_+|}\left(\sum_{i\in\mathcal{A}_+\cup\mathcal{A}_-} y_i - \sum_{n=1}^M c_n^*\Phi_n(\boldsymbol{x}_i)\right) \tag{5.103}$$

$$= -\frac{1}{2|\mathcal{A}_+|}\left(\sum_{i\in\mathcal{A}_+}\sum_{n=1}^M c_n^*\Phi_n(\boldsymbol{x}_i) + \sum_{j\in\mathcal{A}_-}\sum_{n=1}^M c_n^*\Phi_n(\boldsymbol{x}_j)\right). \tag{5.104}$$

The dual problem (5.101) exploits the structure compared to the original minimax problem (5.34) such that the inner problem is now a standard constrained QP-problem, which is relatively easy to solve. Fortunately, the nonlinear global optimization over an arbitrary complicated feasible set (data space $\mathcal{X}$) is now transformed to an outer maximization over a convex hull of matrices enabling us to solve the Lipschitz classifier iteratively by a sequence of QP-problems as presented in next chapter.

Summary

At the beginning of this chapter, we introduced the theory behind the most general embedding for the generalization of a maximum margin classification to more general spaces than Hilbert spaces induced by a kernel. Using the embedding of a metric space into a Banach space $AE(\mathcal{X}_0)$ and a simultaneous embedding of the space of bounded Lipschitz continuous functions into the dual $AE^*(\mathcal{X}_0)$, we reviewed the Lipschitz classifier framework proposed by [von Luxburg 2004]. The presented theory justifies that the Lipschitz constant $L(f)$ lower bounds a margin in a implicitly defined Banach space $AE(\mathcal{X}_0)$. Thus, minimizing $L(f)$ also maximizes the margin. Moreover, the SVM and the 1-Nearest-Neighbor classifier are special cases of the Lipschitz classifier. It follows Algorithms (4.2.2) and (4.2.3) use very crude approximations of the Lipschitz constant. The most general versions of the Lipschitz classifier algorithm (Alg. 5.1.1 and Alg. 5.1.2) are difficult to compute due to the evaluation of the Lipschitz constant for each function of a considered decision function space. Rarely any analytic expression for $L(f)$ can be derived permitting the development of practical algorithms.

However, we showed that restricting the function space to the still very rich space of at least one-time continuously differentiable functions (Def. (5.2.1)), which are defined on a compact and convex Euclidean metric space, let us compute the Lipschitz constant (Lemma (5.2.1)) explicitly. The proposed setting enables us to implement almost all decision functions important for machine learning in a maximum margin concept without severe restrictions like the Mercer's condition. Moreover, the explicit expression of the Lipschitz constant resulted in a novel minimax soft-margin algorithm (Alg. 5.2.1), in which the Lipschitz constant is a regularization operator of the optimization problem implying an optimal solution that induces a flat decision boundary.

Because Algorithm (5.2.1) is a constrained minimax problem, standard optimization methods do not apply. Therefore, we reformulated the problem into a so-called semi-infinite program in order to exploit its structure as much as possible. We obtained an additional new Lipschitz classifier algorithm (Alg. 5.2.3). The primal SIP version is the first implementable formulation of the Lipschitz classifier. Using a sufficiently fine discretized data space, it can be solved with standard optimization methods. Because, discretization is practically not satisfactory for high-dimensional data spaces and often the complicated feasible set of the primal problem breaks down standard solvers, we proved a duality theorem for a convex SIP (Theorem (5.2.3)). Duality enables us to interconnect the minimax problem with a max-min problem in which the maximization has to be performed over a convex hull of positive semi-definite matrices. The obtained new dual Lipschitz classifier algorithm

(Alg. 5.2.4) further exploits the problem structure, so that we are able for the first time to apply standard optimization methods without discretization and to solve the Lipschitz classifier iteratively. Thus, at this point, we can state without doubt that we reached the main objective of this thesis successfully.

In the next chapter, we want to discuss the details of implementing the new Lipschitz classifier algorithms divided into parts: the inner constrained QP-problem and the outer problem of maximizing the optimal value of inner problem over a convex hull of positive semi-definite matrices. For solving the inner QP-problem, we adapt a *Primal-Dual Interior Point method* (Section (6.1)). And for solving the outer problem, an optimal convex combination of matrices is iteratively constructed using a stochastic search based on *Simulated Annealing* and a *Spectral-Gradient method* (Section (6.2)). Then, in Section (6.3) both solvers are used to develop two different realizations of the dual $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm.

Chapter 6

Implementing the Lipschitz Classifier

Contents

In the last chapter, we derived a new dual formulation of the $\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz classifier which can be solved using standard optimization techniques without the need of any discretization of the data space. In this chapter, we discuss in details all components necessary for a real implementation of the dual $\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz classifier algorithm in software.

In its abstract form, the dual $\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz classifier Algorithm (5.2.4) consists in essence of two coupled optimization problems, namely the

inner problem, that is a *Quadratic Problem* (QP) with convex feasible set, and the *outer problem* of finding an optimal matrix from a convex hull of positive semidefinite matrices such that the optimal value of the inner problem is maximized.

Details on a QP solver implementation adapted to the particular inner problem structure is the topic of Section (6.1). The implementation details regarding a sequential optimization scheme for the outer problem are discussed in Section (6.2).

After developing these building blocks, in Section (6.3.1) and (6.3.2) two detailed realizations of the dual $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm (Alg. 5.2.4) are introduced and serve as a template for our own implementations used in the experimental part of this thesis. Note, although aiming to solve the same problem, both realizations are technically different in the way how they handle the ambiguity of multiple equivalent solutions as discussed in the context of Theorem (5.2.4).

6.1 A QP-Solver for the Inner Constrained Problem

Recall Section (5.2.2.3), for any $\mathbf{K} \in conv(\mathcal{K})$ (Def. (5.2.4)) with

$$\mathcal{K} = \left\{ 0 \preceq \mathbf{K}(\boldsymbol{x}) \in \mathbb{R}^{M \times M} \mid \boldsymbol{x} \in \mathcal{X} \right\} \tag{6.1}$$

the *inner problem* of the dual $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier Algorithm (5.2.4) is a *Quadratic Optimization* (QP) problem

$$\begin{aligned} q(\mathbf{K}) &:= \max_{(\boldsymbol{\alpha}, \boldsymbol{c}) \in \mathcal{S}_D(\mathbf{K})} \boldsymbol{\alpha}^T \mathbf{1}_N - \frac{1}{2} \boldsymbol{c}^T \mathbf{K} \boldsymbol{c} \\ &= - \min_{(\boldsymbol{\alpha}, \boldsymbol{c}) \in \mathcal{S}_D(\mathbf{K})} \frac{1}{2} \boldsymbol{c}^T \mathbf{K} \boldsymbol{c} - \boldsymbol{\alpha}^T \mathbf{1}_N \end{aligned} \tag{6.2}$$

with convex feasible set

$$\mathcal{S}_D(\mathbf{K}) := \left\{ (\boldsymbol{\alpha}, \boldsymbol{c}) \in \mathbb{R}^N \times \mathbb{R}^M \;\middle|\; \mathbf{K}\boldsymbol{c} = \mathbf{G}^T \mathbf{Y} \boldsymbol{\alpha}, \boldsymbol{\alpha}^T \boldsymbol{y} = 0, \mathbf{0} \leq \boldsymbol{\alpha} \leq C \mathbf{1}_N \right\}. \tag{6.3}$$

Such QP problems are well-known (e.g. the SVM algorithm in Sec. (3.6)) and efficiently solvable with Newton-type methods applied to a modified KKT-system such that the iterates converge to a solution starting from a point of the interior of the feasible set. The logarithmic barrier *Interior-Point* (IP) approach for constrained problems [Frisch 1955] is based on a logarithmic barrier

function making inequalities $g_i(\boldsymbol{x}) \geq 0$ implicit in the objective function, i.e.

$$\begin{aligned} & \min_{\boldsymbol{x}>\mathbf{0}} f(\boldsymbol{x}) - \tau \sum_{i=1}^{N} \log(g_i(\boldsymbol{x})) \\ \text{s.t.} \quad & h(\boldsymbol{x}) = 0. \end{aligned} \tag{6.4}$$

For an appropriate sequence of decreasing $\tau > 0$, it is possible to solve a sequence of equality constraint problems with Newton's method, such that the sequence of solutions converges to a solution of the inequality constrained problem. This idea was first introduced by [Frisch 1955] and then formally studied in [Fiacco 1968]. Due to the seminal work of [Karmarkar 1984] about a polynomial time projective algorithm, the logarithmic barrier method gained popularity after [Gill 1986] pointed out the close connections with Karmarkar's algorithm. Nowadays IP methods are the most powerful and reliable algorithms for solving linear programming problems in polynomial time, and they are applicable even for efficiently solving convex quadratic optimization problems [Monteiro 1989b]. Very similar to the logarithmic barrier approach is the class of so-called *primal-dual interior-point methods*. The search directions in a primal-dual IP method are obtained by a relaxation of the KKT-system and applying Newton's method [Monteiro 1989a], [Monteiro 1989b]. Contrary to logarithmic barrier approaches eliminating dual variables, the primal-dual IP method computes search directions for the primal and dual variables simultaneously. Primal-dual IP approaches are often more efficient than barrier methods, in particular if high accuracy is required. For a practical implementation the *primal-dual-predictor-corrector algorithm* proposed by [Mehrotra 1992] emerged as the algorithm of choice in case of linear and quadratic programming problems. Upon the many available and well-written textbooks on nonlinear constrained optimization, we would like to refer the reader to e.g. [Boyd 2004] for more details on convex optimization.

In the following, we derive Mehrotra's algorithm adapted to our problem structure given by (6.2) and (6.3). The resulting QP-solver is needed later on as a subroutine in the complete $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz Classifier implementation.

For this purpose, we first abstract from our problem formulation:

Definition 6.1.1 (Primal Convex QP-Problem (PCQP))**.** *Let be* $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, $f(\boldsymbol{x}, \boldsymbol{y}) := \frac{1}{2}\boldsymbol{y}^T\mathbf{Q}\boldsymbol{y} + \boldsymbol{d}^T\boldsymbol{x}$, $\mathbf{Q} \in \mathbb{R}^{m\times m}, \mathbf{Q} = \mathbf{Q}^T \succ 0$, $\mathbf{A} \in \mathbb{R}^{p\times n}$ *and*

$\mathbf{B} \in \mathbb{R}^{m \times n}$. *We define the primal convex QP-problem as:*

$$\min_{\boldsymbol{x},\boldsymbol{y}} f(\boldsymbol{x},\boldsymbol{y}) \tag{6.5}$$

$$\begin{aligned} s.t. \quad & \mathbf{0} \leq \boldsymbol{x} \leq \boldsymbol{u}_b \\ & \mathbf{A}\boldsymbol{x} + \boldsymbol{b} = \mathbf{0} \\ & \mathbf{Q}\boldsymbol{y} + \mathbf{B}\boldsymbol{x} = \mathbf{0}. \end{aligned}$$

Using so-called slack variables $\mathbf{0} \leq \boldsymbol{s} \in \mathbb{R}^n$ *we obtain the more appropriate equivalent form of (6.5):*

$$\min_{\boldsymbol{x}\geq\mathbf{0},\boldsymbol{s}\geq\mathbf{0},\boldsymbol{y}} f(\boldsymbol{x},\boldsymbol{y}) \tag{6.6}$$

$$\begin{aligned} s.t. \quad & \boldsymbol{x} - \boldsymbol{u}_b + \boldsymbol{s} = \mathbf{0} \\ & \mathbf{A}\boldsymbol{x} + \boldsymbol{b} = \mathbf{0} \\ & \mathbf{Q}\boldsymbol{y} + \mathbf{B}\boldsymbol{x} = \mathbf{0}. \end{aligned}$$

Because PCQP has a convex objective and a convex feasible set, the KKT-conditions are necessary and sufficient for optimality (App. (A.3.3)). This means, in order to solve PCQP we have to solve the associated KKT-system. Therefore, we define the Lagrangian function

$$\begin{aligned} L^P(\boldsymbol{x},\boldsymbol{y},\boldsymbol{s},\boldsymbol{\kappa},\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\nu},\boldsymbol{\xi}) := \frac{1}{2}\boldsymbol{y}^T\mathbf{Q}^T\boldsymbol{y} + \boldsymbol{d}^T\boldsymbol{x} + \boldsymbol{\kappa}^T(\boldsymbol{x} - \boldsymbol{u}_b + \boldsymbol{s}) + \\ \boldsymbol{\lambda}^T(\mathbf{A}\boldsymbol{x} + \boldsymbol{b}) + \boldsymbol{\mu}^T(\mathbf{Q}\boldsymbol{y} + \mathbf{B}\boldsymbol{x}) - \boldsymbol{\nu}^T\boldsymbol{x} - \boldsymbol{\xi}^T\boldsymbol{s} \end{aligned} \tag{6.7}$$

with Lagrange multipliers $(\boldsymbol{\kappa},\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\nu},\boldsymbol{\xi}) \in \mathbb{R}^n \times \mathbb{R}^p \times \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n$.

It follows the primal KKT-system

$$\begin{aligned} \nabla_{\boldsymbol{x}} L^P(\boldsymbol{x},\boldsymbol{y},\boldsymbol{s},\boldsymbol{\kappa},\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\nu},\boldsymbol{\xi}) &= \boldsymbol{d} + \boldsymbol{\kappa} + \mathbf{A}^T\boldsymbol{\lambda} + \mathbf{B}^T\boldsymbol{\mu} - \boldsymbol{\nu} = \mathbf{0} && (6.8) \\ \nabla_{\boldsymbol{y}} L^P(\boldsymbol{x},\boldsymbol{y},\boldsymbol{s},\boldsymbol{\kappa},\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\nu},\boldsymbol{\xi}) &= \mathbf{Q}\boldsymbol{y} + \mathbf{Q}\boldsymbol{\mu} = \mathbf{0} \Leftrightarrow \boldsymbol{y} = -\boldsymbol{\mu} && (6.9) \\ \nabla_{\boldsymbol{s}} L^P(\boldsymbol{x},\boldsymbol{y},\boldsymbol{s},\boldsymbol{\kappa},\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\nu},\boldsymbol{\xi}) &= \boldsymbol{\kappa} - \boldsymbol{\xi} = \mathbf{0} \Leftrightarrow \boldsymbol{\kappa} = \boldsymbol{\xi} && (6.10) \\ & \mathbf{Q}\boldsymbol{y} + \mathbf{B}\boldsymbol{x} = \mathbf{0} && (6.11) \\ & \mathbf{A}\boldsymbol{x} + \boldsymbol{b} = \mathbf{0} && (6.12) \\ & \boldsymbol{x} - \boldsymbol{u}_b + \boldsymbol{s} = \mathbf{0} && (6.13) \\ & \mathbf{N}\boldsymbol{x} = 0,\ \boldsymbol{\Xi}\boldsymbol{s} = 0 && (6.14) \\ & \boldsymbol{x} \geq \mathbf{0},\ \boldsymbol{s} \geq \mathbf{0} && (6.15) \\ & \boldsymbol{\nu} \geq \mathbf{0},\ \boldsymbol{\xi} \geq \mathbf{0} && (6.16) \end{aligned}$$

with diagonal matrices $\mathbf{N} := diag(\boldsymbol{\nu}) \in \mathbb{R}^{n \times n}$ and $\boldsymbol{\Xi} := diag(\boldsymbol{\xi}) \in \mathbb{R}^{n \times n}$. One can show (cf. App. B.1.1) that a solution $\boldsymbol{w}^* := (\boldsymbol{y}^*,\boldsymbol{\lambda}^*,\boldsymbol{\xi}^*,\boldsymbol{x}^*,\boldsymbol{\nu}^*,\boldsymbol{s}^*)$

is a primal and dual feasible optimal point of PCQP. This is also the reason for the name of *primal-dual interior-point methods*, which apply Netwon's method to the primal KKT-system that results in iterates for the primal and dual variables simultaneously.

6.1.1 Newton's Method for Solving the KKT-System

For solving the KKT-system (6.8)-(6.14) ignoring for a moment the inequality constraints (6.15) and (6.16) Newton's method suggest to solve at the current iterate $\boldsymbol{w}$ the Newton-equation

$$D\boldsymbol{\Psi}_0(\boldsymbol{w})\Delta\boldsymbol{w} = -\boldsymbol{\Psi}_0(\boldsymbol{w}) \tag{6.17}$$

with respect to the Newton step $\Delta\boldsymbol{w}$. The matrix $D\boldsymbol{\Psi}_0(\boldsymbol{w})$ is the *Jakobi*-matrix of

$$\boldsymbol{\Psi}_0(\boldsymbol{w}) := \begin{pmatrix} \mathbf{Q}\boldsymbol{y} + \mathbf{B}\boldsymbol{x} \\ \mathbf{A}\boldsymbol{x} + \boldsymbol{b} \\ \boldsymbol{x} - \boldsymbol{u}_b + \boldsymbol{s} \\ \mathbf{A}^T\boldsymbol{\lambda} - \mathbf{B}^T\boldsymbol{y} + \boldsymbol{\xi} - \boldsymbol{\nu} + \boldsymbol{d} \\ \mathbf{N}\boldsymbol{x} \\ \boldsymbol{\Xi}\boldsymbol{s} \end{pmatrix} \tag{6.18}$$

with $\boldsymbol{w} := (\boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{\xi}, \boldsymbol{x}, \boldsymbol{\nu}, \boldsymbol{s}) \in \mathbb{R}^m \times \mathbb{R}^p \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n$. A new iterate is obtained by the update $\tilde{\boldsymbol{w}} = \boldsymbol{w} + \mu\Delta\boldsymbol{w}$ that is known to be a fixed point iteration with fixed point $\boldsymbol{w}^*$ satisfying $\boldsymbol{\Psi}_0(\boldsymbol{w}^*) = 0$, if convergence takes place. In order that the Newton's iteration converges to an unique (local) fixed point the Jakobi-matrix $D\boldsymbol{\Psi}_0(\boldsymbol{w})$ must be nonsingular[1]:

Theorem 6.1.1. *Let be* $\mathbf{Q} = \mathbf{Q}^T \in \mathbb{R}^{m\times m}$ *positive definite,* $\mathbf{A} \in \mathbb{R}^{p\times n}$ *with* $rank(\mathbf{A}) = p \leq n$ *and* $\mathbf{0} < \boldsymbol{\nu} \in \mathbb{R}^n, \mathbf{0} < \boldsymbol{s} \in \mathbb{R}^n, \mathbf{0} < \boldsymbol{\xi} \in \mathbb{R}^n, \mathbf{0} < \boldsymbol{x} \in \mathbb{R}^n$. *Then the matrix*

$$D\boldsymbol{\Psi}_0(\boldsymbol{w}) := \begin{pmatrix} \mathbf{Q} & \mathbf{0} & \mathbf{0} & \mathbf{B} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_n & \mathbf{0} & \mathbf{I}_n \\ -\mathbf{B}^T & \mathbf{A}^T & \mathbf{I}_n & \mathbf{0} & -\mathbf{I}_n & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{N} & \mathbf{X} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{S} & \mathbf{0} & \mathbf{0} & \boldsymbol{\Xi} \end{pmatrix} \tag{6.19}$$

is nonsingular. Where $\mathbf{S} := diag(\boldsymbol{s}) \in \mathbb{R}^{n\times n}, \mathbf{N} := diag(\boldsymbol{\nu}) \in \mathbb{R}^{n\times n}, \boldsymbol{\Xi} := diag(\boldsymbol{\xi}) \in \mathbb{R}^{n\times n}, \mathbf{X} := diag(\boldsymbol{x}) \in \mathbb{R}^{n\times n}$ *denotes positive diagonal matrices and* $\mathbf{I}_n \in \mathbb{R}^{n\times n}$ *is an identity matrix.*

[1] Global convergence requires definiteness of the Jacobian.

Proof. Suppose there exists a vector $\boldsymbol{v} := (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{e}, \boldsymbol{f}) \neq \mathbf{0}$ that solves $D\boldsymbol{\Psi}_0(\boldsymbol{w})\boldsymbol{v} = \mathbf{0}$. Then it follows from the 4th block-row by left multiplication with $\boldsymbol{d}^T$ that $-\boldsymbol{d}^T\mathbf{B}^T\boldsymbol{a} + \boldsymbol{d}^T\mathbf{A}^T\boldsymbol{b} + \boldsymbol{d}^T\boldsymbol{c} - \boldsymbol{d}^T\boldsymbol{e} = \mathbf{0}$ holds. Now, substituting $\mathbf{Q}\boldsymbol{a} = -\mathbf{B}\boldsymbol{d}$ from the 1st block-row and using the 2nd block-row $\mathbf{A}\boldsymbol{d} = \mathbf{0}$ yields $\boldsymbol{a}^T\mathbf{Q}\boldsymbol{a} + \boldsymbol{d}^T\boldsymbol{c} - \boldsymbol{d}^T\boldsymbol{e} = \mathbf{0}$. With $\boldsymbol{d} = -\mathbf{N}^{-1}\mathbf{X}\boldsymbol{e}$ from the 5th block-row, $\boldsymbol{c} = -\mathbf{S}^{-1}\boldsymbol{\Xi}\boldsymbol{f}$ from the 6th block-row, as well as $\boldsymbol{d} = -\boldsymbol{f}$ from the 3rd block-row, it follows $\boldsymbol{a}^T\mathbf{Q}\boldsymbol{a} + \boldsymbol{f}^T\mathbf{S}^{-1}\boldsymbol{\Xi}\boldsymbol{f} + \boldsymbol{e}^T\mathbf{X}\mathbf{N}^{-1}\boldsymbol{e} = \mathbf{0}$. The latter equation implies immediately $\boldsymbol{a} = \mathbf{0}$, $\boldsymbol{f} = \mathbf{0}$ and $\boldsymbol{e} = \mathbf{0}$, because $\mathbf{Q}$, $\mathbf{S}^{-1}\boldsymbol{\Xi}$ and $\mathbf{X}\mathbf{N}^{-1}$ are positive definite. Further, it follows $\boldsymbol{d} = \mathbf{0}$ and $\boldsymbol{c} = \mathbf{0}$. The reduced 4th block-row then reads as $\mathbf{A}^T\boldsymbol{b} = \mathbf{0}$. Because $\mathbf{A}$ has full row-rank, i.e. $\mathbf{A}^T$ has full column rank, it follows $\boldsymbol{b} = \mathbf{0}$. Thus, our assumption is contradicted.

■

Thus, due to Theorem (6.1.1) we can apply Newton's method to solve $\boldsymbol{\Psi}_0(\boldsymbol{w}) = \mathbf{0}$ as long as we can ensure that all iterates remain strictly feasible with respect to the inequalities (6.15), (6.16). Unfortunately, the latter is not the case as we will see in the next section.

6.1.2 Primal-Dual Interior Point Method

Consider the the complementary constraints $\mathbf{N}\boldsymbol{x} = \mathbf{0}$ and $\boldsymbol{\Xi}\boldsymbol{s} = \mathbf{0}$. Obviously, due to $\boldsymbol{\nu} \geq \mathbf{0}, \boldsymbol{x} \geq \mathbf{0}$ and $\boldsymbol{\xi} \geq \mathbf{0}, \boldsymbol{s} \geq \mathbf{0}$, a solution of the KKT-system must be at the boundary of the feasible set. That means, the strict feasibility with respect to the inequalities (6.15), (6.16), which is necessary for the Jakobi-matrix $D\boldsymbol{\Psi}(\boldsymbol{w})$ to be nonsingular (Theorem 6.1.1), is violated because always at least one component of $\boldsymbol{\nu}$ or $\boldsymbol{x}$, respectively $\boldsymbol{\xi}$ or $\boldsymbol{s}$, has to be zero to satisfy the complementary constraints. However, to apply Newton's method, primal-dual IP methods slack the complementary constraints to be $\mathbf{N}\boldsymbol{x} - \tau\mathbf{1}_n = \mathbf{0}$ and $\boldsymbol{\Xi}\boldsymbol{s} - \tau\mathbf{1}_n = \mathbf{0}$ with an appropriate $\tau > 0$. Then, starting with a sufficiently large initial value τ_0, τ is step-wise reduced and a solution $\boldsymbol{w}_\tau$ of the disturbed KKT-system is iterated via Newton's method. The mapping $\tau \mapsto \boldsymbol{w}_\tau$ is called the *central path*. One can show [Monteiro 1989b], if the strictly feasible set is not empty, then for any $\tau > 0$ the associated *logarithmic barrier problem*

$$\begin{aligned} & \min_{\boldsymbol{x}>\mathbf{0}, \boldsymbol{s}>\mathbf{0}, \boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y}) - \tau \sum_{i=1}^{N} \log(x_i) - \tau \sum_{j=1}^{N} \log(s_j) \qquad (6.20) \\ \text{s.t.} \quad & \boldsymbol{x} - \boldsymbol{u}_b + \boldsymbol{s} = \mathbf{0} \\ & \mathbf{A}\boldsymbol{x} + \boldsymbol{b} = \mathbf{0} \\ & \mathbf{Q}\boldsymbol{y} + \mathbf{B}\boldsymbol{x} = \mathbf{0}. \end{aligned}$$

is unique and completely characterized by a solution $\boldsymbol{w}_\tau$ of the slacked KKT-system

$$\begin{aligned}\boldsymbol{\Psi}_\tau(\boldsymbol{w}) &:= \begin{pmatrix} \mathbf{Q}\boldsymbol{y} + \mathbf{B}\boldsymbol{x} \\ \mathbf{A}\boldsymbol{x} + \boldsymbol{b} \\ \boldsymbol{x} - \boldsymbol{u}_b + \boldsymbol{s} \\ \mathbf{A}^T\boldsymbol{\lambda} - \mathbf{B}^T\boldsymbol{y} + \boldsymbol{\xi} - \boldsymbol{\nu} + \boldsymbol{d} \\ \mathbf{N}\boldsymbol{x} - \tau\mathbf{1}_n \\ \boldsymbol{\Xi}\boldsymbol{s} - \tau\mathbf{1}_n \end{pmatrix} = \mathbf{0}, \qquad (6.21) \\ \boldsymbol{x} > \mathbf{0}, \boldsymbol{s} > \mathbf{0}, \boldsymbol{\xi} > \mathbf{0}, \boldsymbol{\nu} > \mathbf{0}. \qquad (6.22)\end{aligned}$$

Thus, the objective is to solve Newton's equation $D\boldsymbol{\Psi}_\tau(\boldsymbol{w})\Delta\boldsymbol{w} = -\boldsymbol{\Psi}_\tau(\boldsymbol{w})$. Because it holds $D\boldsymbol{\Psi}_\tau(\boldsymbol{w}) = D\boldsymbol{\Psi}_0(\boldsymbol{w})$ (6.19), we obtain a Newton step $\Delta\boldsymbol{w} := (\Delta\boldsymbol{y}, \Delta\boldsymbol{\lambda}, \Delta\boldsymbol{\xi}, \Delta\boldsymbol{x}, \Delta\boldsymbol{\nu}, \Delta\boldsymbol{s})$ for a given $\tau > 0$ by a solution of

$$\begin{pmatrix} \mathbf{Q} & \mathbf{0} & \mathbf{0} & \mathbf{B} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_n & \mathbf{0} & \mathbf{I}_n \\ -\mathbf{B}^T & \mathbf{A}^T & \mathbf{I}_n & \mathbf{0} & -\mathbf{I}_n & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{N} & \mathbf{X} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{S} & \mathbf{0} & \mathbf{0} & \boldsymbol{\Xi} \end{pmatrix} \begin{pmatrix} \Delta\boldsymbol{y} \\ \Delta\boldsymbol{\lambda} \\ \Delta\boldsymbol{\xi} \\ \Delta\boldsymbol{x} \\ \Delta\boldsymbol{\nu} \\ \Delta\boldsymbol{s} \end{pmatrix} = \begin{pmatrix} -\mathbf{Q}\boldsymbol{y} - \mathbf{B}\boldsymbol{x} \\ -\mathbf{A}\boldsymbol{x} - \boldsymbol{b} \\ -\boldsymbol{x} + \boldsymbol{u}_b - \boldsymbol{s} \\ -\mathbf{A}^T\boldsymbol{\lambda} + \mathbf{B}^T\boldsymbol{y} - \boldsymbol{\xi} + \boldsymbol{\nu} - \boldsymbol{d} \\ -\mathbf{N}\mathbf{X}\mathbf{1}_n + \tau\mathbf{1}_n \\ -\boldsymbol{\Xi}\mathbf{S}\mathbf{1}_n + \tau\mathbf{1}_n \end{pmatrix}. \qquad (6.23)$$

By multiplication of the 5th, respectively 6th, block-row with $\boldsymbol{N}^{-1}$, respectively $-\boldsymbol{S}^{-1}$, as well as the 2nd, 3rd and 4th block-row with -1 gives

$$\underbrace{\begin{pmatrix} \mathbf{Q} & \mathbf{0} & \mathbf{0} & \mathbf{B} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I}_n & \mathbf{0} & -\mathbf{I}_n \\ \mathbf{B}^T & -\mathbf{A}^T & -\mathbf{I}_n & \mathbf{0} & \mathbf{I}_n & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_n & \mathbf{N}^{-1}\mathbf{X} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\mathbf{I}_n & \mathbf{0} & \mathbf{0} & -\mathbf{S}^{-1}\boldsymbol{\Xi} \end{pmatrix}}_{=:D\hat{\boldsymbol{\Psi}}(\boldsymbol{w})} \begin{pmatrix} \Delta\boldsymbol{y} \\ \Delta\boldsymbol{\lambda} \\ \Delta\boldsymbol{\xi} \\ \Delta\boldsymbol{x} \\ \Delta\boldsymbol{\nu} \\ \Delta\boldsymbol{s} \end{pmatrix} = \underbrace{\begin{pmatrix} -\mathbf{Q}\boldsymbol{y} - \mathbf{B}\boldsymbol{x} \\ \mathbf{A}\boldsymbol{x} + \boldsymbol{b} \\ \boldsymbol{x} - \boldsymbol{u}_b + \boldsymbol{s} \\ \mathbf{A}^T\boldsymbol{\lambda} - \mathbf{B}^T\boldsymbol{y} + \boldsymbol{\xi} - \boldsymbol{\nu} + \boldsymbol{d} \\ -\mathbf{X}\mathbf{1}_n + \tau\mathbf{N}^{-1}\mathbf{1}_n \\ \boldsymbol{\Xi}\mathbf{1}_n - \tau\mathbf{S}^{-1}\mathbf{1}_n \end{pmatrix}}_{=:\hat{\boldsymbol{\Psi}}_\tau(\boldsymbol{w})}. \qquad (6.24)$$

Now, the Jakobi-matrix is symmetric. Because the 3rd, 5th and 6th block-rows are of simple structure, we solve them explicitly:

$$\Delta \boldsymbol{x} + \Delta \boldsymbol{s} = -\boldsymbol{x} + \boldsymbol{u}_b - \boldsymbol{s} =: \boldsymbol{\rho}_{p1}(\boldsymbol{x}, \boldsymbol{s}) \tag{6.25}$$

$$\Leftrightarrow \quad \boxed{\Delta \boldsymbol{s} = \boldsymbol{\rho}_{p1}(\boldsymbol{x}, \boldsymbol{s}) - \Delta \boldsymbol{x}} \tag{6.26}$$

$$\Delta \boldsymbol{x} + \mathbf{N}^{-1}\mathbf{X}\Delta \boldsymbol{\nu} = -\mathbf{X}\mathbf{1}_n + \tau \mathbf{N}^{-1}\mathbf{1}_n \tag{6.27}$$

$$\Leftrightarrow \quad \mathbf{X}^{-1}\mathbf{N}\Delta \boldsymbol{x} + \Delta \boldsymbol{\nu} = -\mathbf{N}\mathbf{1}_n + \tau \mathbf{X}^{-1}\mathbf{1}_n =: \boldsymbol{\rho}^{\tau}_{kkt1}(\boldsymbol{x}, \boldsymbol{\nu}) \tag{6.28}$$

$$\Leftrightarrow \quad \boxed{\Delta \boldsymbol{\nu} = \boldsymbol{\rho}^{\tau}_{kkt1}(\boldsymbol{x}, \boldsymbol{\nu}) - \mathbf{X}^{-1}\mathbf{N}\Delta \boldsymbol{x}} \tag{6.29}$$

$$\Delta \boldsymbol{\xi} + \mathbf{S}^{-1}\boldsymbol{\Xi}\Delta \boldsymbol{s} = -\boldsymbol{\Xi}\mathbf{1}_n + \tau \mathbf{S}^{-1}\mathbf{1}_n =: \boldsymbol{\rho}^{\tau}_{kkt2}(\boldsymbol{s}, \boldsymbol{\xi}) \tag{6.30}$$

$$\Leftrightarrow \quad \boxed{\Delta \boldsymbol{\xi} = \boldsymbol{\rho}^{\tau}_{kkt2}(\boldsymbol{s}, \boldsymbol{\xi}) - \mathbf{S}^{-1}\boldsymbol{\Xi}\Delta \boldsymbol{s}}. \tag{6.31}$$

Eliminating the associated rows in (6.24) yields

$$\begin{pmatrix} \mathbf{Q} & \mathbf{0} & \mathbf{0} & \mathbf{B} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{B}^T & -\mathbf{A}^T & -\mathbf{I}_n & \mathbf{0} & \mathbf{I}_n & \mathbf{0} \end{pmatrix} \begin{pmatrix} \Delta \boldsymbol{y} \\ \Delta \boldsymbol{\lambda} \\ \Delta \boldsymbol{\xi} \\ \Delta \boldsymbol{x} \\ \Delta \boldsymbol{\nu} \\ \Delta \boldsymbol{s} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\rho}_{p3}(\boldsymbol{x}, \boldsymbol{y}) \\ \boldsymbol{\rho}_{p2}(\boldsymbol{x}) \\ \boldsymbol{\rho}_{d1}(\boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{\xi}, \boldsymbol{\nu}) \end{pmatrix}, \tag{6.32}$$

with substitutions

$$\boldsymbol{\rho}_{p3}(\boldsymbol{x}, \boldsymbol{y}) := -\mathbf{Q}\boldsymbol{y} - \mathbf{B}\boldsymbol{x} \tag{6.33}$$

$$\boldsymbol{\rho}_{p2}(\boldsymbol{x}) := \mathbf{A}\boldsymbol{x} + \boldsymbol{b} \tag{6.34}$$

$$\boldsymbol{\rho}_{d1}(\boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{\xi}, \boldsymbol{\nu}) := \mathbf{A}^T\boldsymbol{\lambda} - \mathbf{B}^T\boldsymbol{y} + \boldsymbol{\xi} - \boldsymbol{\nu} + \boldsymbol{d}. \tag{6.35}$$

Using $\Delta \boldsymbol{\xi}$ and $\Delta \boldsymbol{\nu}$ in the 3rd block-row of (6.32) gives

$$\begin{aligned} & \mathbf{B}^T\Delta \boldsymbol{y} - \mathbf{A}^T\Delta \boldsymbol{\lambda} - \boldsymbol{\rho}^{\tau}_{kkt2}(\boldsymbol{s}, \boldsymbol{\xi}) + \mathbf{S}^{-1}\boldsymbol{\Xi}\Delta \boldsymbol{s} + \boldsymbol{\rho}^{\tau}_{kkt1}(\boldsymbol{x}, \boldsymbol{\nu}) - \mathbf{X}^{-1}\mathbf{N}\Delta \boldsymbol{x} \\ = \; & \boldsymbol{\rho}_{d1}(\boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{\xi}, \boldsymbol{\nu}) \end{aligned} \tag{6.36}$$

and using $\Delta \boldsymbol{s}$ yields

$$\begin{aligned} & \mathbf{B}^T\Delta \boldsymbol{y} - \mathbf{A}^T\Delta \boldsymbol{\lambda} + \mathbf{S}^{-1}\boldsymbol{\Xi}\boldsymbol{\rho}_{p1}(\boldsymbol{x}, \boldsymbol{s}) - \mathbf{S}^{-1}\boldsymbol{\Xi}\Delta \boldsymbol{x} - \mathbf{X}^{-1}\mathbf{N}\Delta \boldsymbol{x} \\ = \; & \boldsymbol{\rho}_{d1}(\boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{\xi}, \boldsymbol{\nu}) - \boldsymbol{\rho}^{\tau}_{kkt1}(\boldsymbol{x}, \boldsymbol{\nu}) + \boldsymbol{\rho}^{\tau}_{kkt2}(\boldsymbol{s}, \boldsymbol{\xi}) \end{aligned} \tag{6.37}$$

$$\begin{aligned} \Leftrightarrow \quad & \mathbf{B}^T\Delta \boldsymbol{y} - \mathbf{A}^T\Delta \boldsymbol{\lambda} - \left(\mathbf{X}^{-1}\mathbf{N} + \mathbf{S}^{-1}\boldsymbol{\Xi}\right)\Delta \boldsymbol{x} \qquad (6.38) \\ = \; & \boldsymbol{\rho}_{d1}(\boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{\xi}, \boldsymbol{\nu}) - \boldsymbol{\rho}^{\tau}_{kkt1}(\boldsymbol{x}, \boldsymbol{\nu}) + \boldsymbol{\rho}^{\tau}_{kkt2}(\boldsymbol{s}, \boldsymbol{\xi}) - \mathbf{S}^{-1}\boldsymbol{\Xi}\boldsymbol{\rho}_{p1}(\boldsymbol{x}, \boldsymbol{s}). \qquad (6.39) \end{aligned}$$

Finally, it results the *reduced KKT-Newton-system*

$$\begin{pmatrix} \mathbf{H} & \mathbf{M} \\ \mathbf{M}^T & -\mathbf{D} \end{pmatrix} \begin{pmatrix} \Delta \boldsymbol{u} \\ \Delta \boldsymbol{x} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\rho}_1(\boldsymbol{x}, \boldsymbol{y}) \\ \boldsymbol{\rho}_2^\tau(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s}, \boldsymbol{\lambda}, \boldsymbol{\nu}, \boldsymbol{\xi}) \end{pmatrix} \tag{6.40}$$

with

$$\begin{aligned} \mathbf{H} &:= \begin{pmatrix} \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \in \mathbb{R}^{(m+p)\times(m+p)}, \ \mathbf{M} := \begin{pmatrix} \mathbf{B} \\ -\mathbf{A} \end{pmatrix} \in \mathbb{R}^{(m+p)\times n}, \\ \mathbf{D} &:= \mathbf{X}^{-1}\mathbf{N} + \mathbf{S}^{-1}\boldsymbol{\Xi} \in \mathbb{R}^{n\times n}, \ \Delta \boldsymbol{u} := \begin{pmatrix} \Delta \boldsymbol{y} \\ \Delta \boldsymbol{\lambda} \end{pmatrix} \in \mathbb{R}^{m+p}, \\ \boldsymbol{\rho}_1(\boldsymbol{x}, \boldsymbol{y}) &:= \begin{pmatrix} \boldsymbol{\rho}_{p3}(\boldsymbol{x}, \boldsymbol{y}) \\ \boldsymbol{\rho}_{p2}(\boldsymbol{x}) \end{pmatrix}, \\ \boldsymbol{\rho}_2^\tau(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s}, \boldsymbol{\lambda}, \boldsymbol{\nu}, \boldsymbol{\xi}) &:= \boldsymbol{\rho}_{d1}(\boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{\xi}, \boldsymbol{\nu}) - \boldsymbol{\rho}_{kkt1}^\tau(\boldsymbol{x}, \boldsymbol{\nu}) + \\ &\qquad \boldsymbol{\rho}_{kkt2}^\tau(\boldsymbol{s}, \boldsymbol{\xi}) - \mathbf{S}^{-1}\boldsymbol{\Xi}\boldsymbol{\rho}_{p1}(\boldsymbol{x}, \boldsymbol{s}). \end{aligned}$$

The reduced KKT-Newton-system can be solved by resolving for $\Delta \boldsymbol{x}$ first:

$$\mathbf{M}^T \Delta \boldsymbol{u} - \mathbf{D}\Delta \boldsymbol{x} = \boldsymbol{\rho}_2^\tau(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s}, \boldsymbol{\lambda}, \boldsymbol{\nu}, \boldsymbol{\xi}) \tag{6.41}$$

$$\Leftrightarrow \quad \boxed{\Delta \boldsymbol{x} = \mathbf{D}^{-1}\left(\mathbf{M}^T \Delta \boldsymbol{u} - \boldsymbol{\rho}_2^\tau(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s}, \boldsymbol{\lambda}, \boldsymbol{\nu}, \boldsymbol{\xi})\right)} \tag{6.42}$$

and then by substitution of $\Delta \boldsymbol{x}$ in

$$\mathbf{H}\Delta \boldsymbol{u} + \mathbf{M}\Delta \boldsymbol{x} = \boldsymbol{\rho}_1(\boldsymbol{x}, \boldsymbol{y}) \tag{6.43}$$

$$\Leftrightarrow \ \mathbf{H}\Delta \boldsymbol{u} + \mathbf{M}\mathbf{D}^{-1}\left(\mathbf{M}^T \Delta \boldsymbol{u} - \boldsymbol{\rho}_2^\tau(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s}, \boldsymbol{\lambda}, \boldsymbol{\nu}, \boldsymbol{\xi})\right) = \boldsymbol{\rho}_1(\boldsymbol{x}, \boldsymbol{y}) \tag{6.44}$$

$$\Leftrightarrow \ \mathbf{H}\Delta \boldsymbol{u} + \mathbf{M}\mathbf{D}^{-1}\mathbf{M}^T \Delta \boldsymbol{u} - \mathbf{M}\mathbf{D}^{-1}\boldsymbol{\rho}_2^\tau(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s}, \boldsymbol{\lambda}, \boldsymbol{\nu}, \boldsymbol{\xi}) = \boldsymbol{\rho}_1(\boldsymbol{x}, \boldsymbol{y}) \tag{6.45}$$

$$\Leftrightarrow \boxed{\left(\mathbf{H} + \mathbf{M}\mathbf{D}^{-1}\mathbf{M}^T\right)\Delta \boldsymbol{u} = \left[\boldsymbol{\rho}_1(\boldsymbol{x}, \boldsymbol{y}) + \mathbf{M}\mathbf{D}^{-1}\boldsymbol{\rho}_2^\tau(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s}, \boldsymbol{\lambda}, \boldsymbol{\nu}, \boldsymbol{\xi})\right]}. \tag{6.46}$$

The last equation (6.46) is called *normal equation* and is best solved using a *Cholesky factorization* [Golub 1996].

6.1.3 Mehrotra's Primal-Dual Predictor-Corrector Algorithm

As discussed in the last sections, the KKT-system (6.8)-(6.16) of problem (6.5), respectively (6.6), can be solved approximately using Newton's Method applied to a disturbed KKT-system (6.21), while maintaining strict feasibility (6.22). The sequence of solutions $\boldsymbol{w}_\tau$ of the disturbed KKT-system is called the central path which is tracked by practical implementations of the IP method. Because tracking the central path is numerically not exactly possible, so-called *path following methods* define valid search directions $\Delta \boldsymbol{w}$ in a

proximity of the central path. A very popular algorithm of this kind is *Mehrotra's Primal-Dual Predictor-Corrector Algorithm* [Mehrotra 1992]. Beside a few heuristics established to be useful in practice, Mehrotra proposed to use a predictor-corrector approach to obtain improved search directions. To motivate the idea, the equation system $\boldsymbol{\Psi}_0(\boldsymbol{w}) = \mathbf{0}$ can be considered as a first order *Taylor-series* approximation of the KKT-system (6.8)-(6.16) neglecting nonlinear terms $\Delta\mathbf{N}\Delta\mathbf{X}\mathbf{1}_n$ and $\Delta\boldsymbol{\Xi}\Delta\mathbf{S}\mathbf{1}_n$, i.e.

$$\boldsymbol{\Psi}_0(\boldsymbol{x}+\Delta\boldsymbol{x}, \boldsymbol{y}+\Delta\boldsymbol{y}, \boldsymbol{s}+\Delta\boldsymbol{s}, \boldsymbol{\lambda}+\Delta\boldsymbol{\lambda}, \boldsymbol{\nu}+\Delta\boldsymbol{\nu}, \boldsymbol{\xi}+\Delta\boldsymbol{\xi})$$

$$= \begin{pmatrix} \mathbf{Q}\boldsymbol{y}+\mathbf{Q}\Delta\boldsymbol{y}+\mathbf{B}\boldsymbol{x}+\mathbf{B}\Delta\boldsymbol{x} \\ \mathbf{A}\boldsymbol{x}+\mathbf{A}\Delta\boldsymbol{x}+\boldsymbol{b} \\ \boldsymbol{x}+\Delta\boldsymbol{x}-\boldsymbol{u}_b+\boldsymbol{s}+\Delta\boldsymbol{s} \\ \mathbf{A}^T\boldsymbol{\lambda}+\mathbf{A}^T\Delta\boldsymbol{\lambda}-\mathbf{B}^T\boldsymbol{y}-\mathbf{B}^T\Delta\boldsymbol{y}+\boldsymbol{\xi}+\Delta\boldsymbol{\xi}-\boldsymbol{\nu}-\Delta\boldsymbol{\nu}+\boldsymbol{d} \\ \mathbf{N}\mathbf{X}\mathbf{1}_n+\mathbf{N}\Delta\boldsymbol{x}+\mathbf{X}\Delta\boldsymbol{\nu}+\Delta\mathbf{N}\Delta\mathbf{X}\mathbf{1}_n \\ \boldsymbol{\Xi}\mathbf{S}\mathbf{1}_n+\mathbf{S}\Delta\boldsymbol{\xi}+\boldsymbol{\Xi}\Delta\boldsymbol{s}+\Delta\boldsymbol{\Xi}\Delta\mathbf{S}\mathbf{1}_n \end{pmatrix} \tag{6.47}$$

$$\approx \begin{pmatrix} \mathbf{Q}\boldsymbol{y}+\mathbf{Q}\Delta\boldsymbol{y}+\mathbf{B}\boldsymbol{x}+\mathbf{B}\Delta\boldsymbol{x} \\ \mathbf{A}\boldsymbol{x}+\mathbf{A}\Delta\boldsymbol{x}+\boldsymbol{b} \\ \boldsymbol{x}+\Delta\boldsymbol{x}-\boldsymbol{u}_b+\boldsymbol{s}+\Delta\boldsymbol{s} \\ \mathbf{A}^T\boldsymbol{\lambda}+\mathbf{A}^T\Delta\boldsymbol{\lambda}-\mathbf{B}^T\boldsymbol{y}-\mathbf{B}^T\Delta\boldsymbol{y}+\boldsymbol{\xi}+\Delta\boldsymbol{\xi}-\boldsymbol{\nu}-\Delta\boldsymbol{\nu}+\boldsymbol{d} \\ \mathbf{N}\mathbf{X}\mathbf{1}_n+\mathbf{N}\Delta\boldsymbol{x}+\mathbf{X}\Delta\boldsymbol{\nu} \\ \boldsymbol{\Xi}\mathbf{S}\mathbf{1}_n+\mathbf{S}\Delta\boldsymbol{\xi}+\boldsymbol{\Xi}\Delta\boldsymbol{s} \end{pmatrix} = \mathbf{0}. \tag{6.48}$$

Now, in order to get as close as possible to an exact solution of (6.47), resulting in an improved convergence rate and accuracy of the IP method, one first solves (Eq. (6.24)), i.e.

$$D\hat{\boldsymbol{\Psi}}(\boldsymbol{w})\Delta\boldsymbol{w}^P = \hat{\boldsymbol{\Psi}}_\tau(\boldsymbol{w}) \tag{6.49}$$

yielding an estimate $\Delta\boldsymbol{w}^P := (\Delta\boldsymbol{y}^P, \Delta\boldsymbol{\lambda}^P, \Delta\boldsymbol{\xi}^P, \Delta\boldsymbol{x}^P, \Delta\boldsymbol{\nu}^P, \Delta\boldsymbol{s}^P)$ of the exact Newton-direction. This step is called *predictor-step*. Then, an improved estimate $\Delta\boldsymbol{w}^K := (\Delta\boldsymbol{y}^K, \Delta\boldsymbol{\lambda}^K, \Delta\boldsymbol{\xi}^K, \Delta\boldsymbol{x}^K, \Delta\boldsymbol{\nu}^K, \Delta\boldsymbol{s}^K)$ is obtained by solving (6.47) using the predictor estimates, i.e. by resolving

$$D\hat{\boldsymbol{\Psi}}(\boldsymbol{w})\Delta\boldsymbol{w}^K = \begin{pmatrix} -\mathbf{Q}\boldsymbol{y}-\mathbf{B}\boldsymbol{x} \\ \mathbf{A}\boldsymbol{x}+\boldsymbol{b} \\ \boldsymbol{x}-\boldsymbol{u}_b+\boldsymbol{s} \\ \mathbf{A}^T\boldsymbol{\lambda}-\mathbf{B}^T\boldsymbol{y}+\boldsymbol{\xi}-\boldsymbol{\nu}+\boldsymbol{d} \\ -\mathbf{X}\mathbf{1}_n+\tau\mathbf{N}^{-1}\mathbf{1}_n-\mathbf{N}^{-1}\Delta\mathbf{N}^P\Delta\mathbf{X}^P\mathbf{1}_n \\ \boldsymbol{\Xi}\mathbf{1}_n-\tau\mathbf{S}^{-1}\mathbf{1}_n+\mathbf{S}^{-1}\Delta\mathbf{S}^P\Delta\boldsymbol{\Xi}^P\mathbf{1}_n \end{pmatrix}. \tag{6.50}$$

This step is called *corrector-step*. In the same manner as for the predictor-step (Eq. (6.26), (6.29), (6.31), (6.42), (6.46)), the reduced KKT-Newton-system for the corrector estimates can be derived to be

$$\begin{pmatrix} \mathbf{H} & \mathbf{M} \\ \mathbf{M}^T & -\mathbf{D} \end{pmatrix} \begin{pmatrix} \Delta \boldsymbol{u}^K \\ \Delta \boldsymbol{x}^K \end{pmatrix} = \begin{pmatrix} \boldsymbol{\rho}_1(\boldsymbol{x}, \boldsymbol{y}) \\ \boldsymbol{\rho}_2^\tau(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s}, \boldsymbol{\lambda}, \boldsymbol{\nu}, \boldsymbol{\xi}) + \boldsymbol{\rho}^K(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{\nu}, \boldsymbol{\xi}, \Delta \boldsymbol{x}^P, \Delta \boldsymbol{s}^P, \Delta \boldsymbol{\nu}^P, \Delta \boldsymbol{\xi}^P) \end{pmatrix} \tag{6.51}$$

with

$$\boldsymbol{\rho}^K(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{\nu}, \boldsymbol{\xi}, \Delta \boldsymbol{x}^P, \Delta \boldsymbol{s}^P, \Delta \boldsymbol{\nu}^P, \Delta \boldsymbol{\xi}^P) := \mathbf{X}^{-1} \Delta \mathbf{N}^P \Delta \mathbf{X}^P \mathbf{1}_n - \mathbf{S}^{-1} \Delta \mathbf{S}^P \Delta \boldsymbol{\Xi}^P \mathbf{1}_n. \tag{6.52}$$

Crucial for the corrector-step is that only the right-hand side in (6.51) has changed compared to (6.40). That means the expensive computation for solving the normal equation using a Cholesky factorization must be performed only once in the predictor-step.

The corrector-updates can be computed by

$$\Delta \boldsymbol{s}^K = \boldsymbol{\rho}_{p1}(\boldsymbol{x}, \boldsymbol{s}) - \Delta \boldsymbol{x}^K \tag{6.53}$$

$$\Delta \boldsymbol{\nu}^K = \boldsymbol{\rho}_{kkt1}^\tau(\boldsymbol{x}, \boldsymbol{\nu}) - \mathbf{X}^{-1} \mathbf{N} \Delta \boldsymbol{x}^K - \mathbf{X}^{-1} \Delta \mathbf{N}^P \Delta \mathbf{X}^P \mathbf{1}_n \tag{6.54}$$

$$\Delta \boldsymbol{\xi}^K = \boldsymbol{\rho}_{kkt2}^\tau(\boldsymbol{s}, \boldsymbol{\xi}) - \mathbf{S}^{-1} \boldsymbol{\Xi} \Delta \boldsymbol{s}^K - \mathbf{S}^{-1} \Delta \mathbf{S}^P \Delta \boldsymbol{\Xi}^P \mathbf{1}_n \tag{6.55}$$

$$\Delta \boldsymbol{x}^K = \mathbf{D}^{-1} \Big(\mathbf{M}^T \Delta \boldsymbol{u}^K - \boldsymbol{\rho}_2^\tau(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s}, \boldsymbol{\lambda}, \boldsymbol{\nu}, \boldsymbol{\xi}) - \boldsymbol{\rho}^K(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{\nu}, \boldsymbol{\xi}, \Delta \boldsymbol{x}^P, \Delta \boldsymbol{s}^P, \Delta \boldsymbol{\nu}^P, \Delta \boldsymbol{\xi}^P) \Big) \tag{6.56}$$

and the normal equation

$$\left(\mathbf{H} + \mathbf{M}\mathbf{D}^{-1}\mathbf{M}^T\right) \Delta \boldsymbol{u}^K = \boldsymbol{\rho}_1(\boldsymbol{x}, \boldsymbol{y}) + \mathbf{M}\mathbf{D}^{-1} \Big(\boldsymbol{\rho}_2^\tau(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s}, \boldsymbol{\lambda}, \boldsymbol{\nu}, \boldsymbol{\xi}) + \boldsymbol{\rho}^K(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{\nu}, \boldsymbol{\xi}, \Delta \boldsymbol{x}^P, \Delta \boldsymbol{s}^P, \Delta \boldsymbol{\nu}^P, \Delta \boldsymbol{\xi}^P) \Big). \tag{6.57}$$

To ensure the strict feasibility of the iterates $\boldsymbol{w}^{k+1} = \boldsymbol{w}^k + \mu \Delta \boldsymbol{w}$, while obtaining a fast convergence rate, one chooses the maximum possible step-length $\mu^{max} \in [0, 1]$ such that all iterates remain nonnegative (*backtracking-line-search*):

$$\mu^{max} := \sup \Big\{ \mu \in [0,1] \;\Big|\; \boldsymbol{x}^k + \mu \Delta \boldsymbol{x} \geq 0, \boldsymbol{s}^k + \mu \Delta \boldsymbol{s} \geq 0, \boldsymbol{\nu}^k + \mu \Delta \boldsymbol{\nu} \geq 0, \boldsymbol{\xi}^k + \mu \Delta \boldsymbol{\xi} \geq 0 \Big\} \tag{6.58}$$

$$= \min \Big\{ 1, \min_{i:\Delta x_i < 0} \Big\{ -\frac{x_i^k}{\Delta x_i} \Big\}, \min_{i:\Delta s_i < 0} \Big\{ -\frac{s_i^k}{\Delta s_i} \Big\}, \min_{i:\Delta \nu_i < 0} \Big\{ -\frac{\nu_i^k}{\Delta \nu_i} \Big\}, \min_{i:\Delta \xi_i < 0} \Big\{ -\frac{\xi_i^k}{\Delta \xi_i} \Big\} \Big\}. \tag{6.59}$$

Then, the step-length for the updates is computed by $\mu = \eta \cdot \mu^{max}$ using a damping factor $\eta \in [0.8, 1.0]$ that avoids the iterates to converge to the vicinity of the boundary of the feasible set.

6.1.4 A Practical Implementation of the QP-Solver

For a practical implementation of the IP method proposed by [Mehrotra 1992], we use the predictor and corrector updates derived in the preceding section with respect to the problem structure of the inner problem (Def. (6.1.1)) of the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm. It is important to note that our adapted IP solver implements the heuristics proposed in [Mehrotra 1992], except the computation of the step-length μ and the relaxation parameter τ are justified regarding the particular problem we want to solve. Additionally, we do not use different step-lengths for the variables $(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{\nu}, \boldsymbol{\xi})$ because of their linear coupling via the KKT-conditions. The IP solver is summarized in Algorithm (6.1.1).

Algorithm 6.1.1 QP-Solver for the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm (cf. [Mehrotra 1992])

Require: start values $\boldsymbol{u}^0 := ((\boldsymbol{y}^0)^T, (\boldsymbol{\lambda}^0)^T)^T \in \mathbb{R}^{m+p}$, $\mathbf{0} < (\boldsymbol{x}^0, \boldsymbol{s}^0, \boldsymbol{\nu}^0, \boldsymbol{\xi}^0) \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n$, accuracies $\epsilon_1, \epsilon_2 > 0$, max. number of iterations $k_{max} \in \mathbb{N}$, input data $\mathbf{A} \in \mathbb{R}^{p \times n}$, $\mathbf{B} \in \mathbb{R}^{m \times n}$, $\mathbf{Q} \in \mathbb{R}^{m \times m}$, $\boldsymbol{d} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{R}^p$, upper bounds $\mathbf{0} < \boldsymbol{u}_b \in \mathbb{R}^n$

Ensure: $\mathbf{Q}$ is positive definite, $\mathbf{A}$ is of full row rank

1: $k \leftarrow 0$.
2: **while** $k \leq k_{max}$ **do**
3: $(\boldsymbol{u}, \boldsymbol{x}, \boldsymbol{s}, \boldsymbol{\nu}, \boldsymbol{\xi}) \leftarrow (\boldsymbol{u}^k, \boldsymbol{x}^k, \boldsymbol{s}^k, \boldsymbol{\nu}^k, \boldsymbol{\xi}^k)$
4: $\tau \leftarrow \left[(\boldsymbol{\nu})^T \boldsymbol{x} + (\boldsymbol{\xi})^T \boldsymbol{s}\right] / (2n)$
5: **if** $\|\mathbf{A}^T \boldsymbol{\lambda} - \mathbf{B}^T \boldsymbol{y} + \boldsymbol{\xi} - \boldsymbol{\nu} + \boldsymbol{d}\| < \epsilon_1$, $\|\boldsymbol{x} - \boldsymbol{u}_b + \boldsymbol{s}\| < \epsilon_1$, $\|\mathbf{A}\boldsymbol{x} + \boldsymbol{b}\| < \epsilon_1$, $\|\mathbf{Q}\boldsymbol{y} + \mathbf{B}\boldsymbol{x}\| < \epsilon_1$ and $\tau < \epsilon_2$ **then** {solution is found}
6: **return** $(\boldsymbol{x}, \boldsymbol{y})$
7: **end if**
8: Compute predictor-steps using $\tau_P := 0$:

$$\text{Solve } \left(\mathbf{H} + \mathbf{M}\mathbf{D}^{-1}\mathbf{M}^T\right) \Delta \boldsymbol{u}^P = \left[\boldsymbol{\rho}_1(\boldsymbol{x}, \boldsymbol{y}) + \mathbf{M}\mathbf{D}^{-1} \boldsymbol{\rho}_2^{\tau_P}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s}, \boldsymbol{\lambda}, \boldsymbol{\nu}, \boldsymbol{\xi})\right]$$

$$\begin{aligned}
\Delta \boldsymbol{x}^P &\leftarrow \mathbf{D}^{-1}\left(\mathbf{M}^T \Delta \boldsymbol{u}^P - \boldsymbol{\rho}_2^{\tau_P}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s}, \boldsymbol{\lambda}, \boldsymbol{\nu}, \boldsymbol{\xi})\right) \\
\Delta \boldsymbol{s}^P &\leftarrow \boldsymbol{\rho}_{p1}(\boldsymbol{x}, \boldsymbol{s}) - \Delta \boldsymbol{x}^P \\
\Delta \boldsymbol{\nu}^P &\leftarrow \boldsymbol{\rho}_{kkt1}^{\tau_P}(\boldsymbol{x}, \boldsymbol{\nu}) - \mathbf{X}^{-1}\mathbf{N}\Delta \boldsymbol{x}^P \\
\Delta \boldsymbol{\xi}^P &\leftarrow \boldsymbol{\rho}_{kkt2}^{\tau_P}(\boldsymbol{s}, \boldsymbol{\xi}) - \mathbf{S}^{-1}\boldsymbol{\Xi}\Delta \boldsymbol{s}^P.
\end{aligned}$$

9: Compute maximum predictor step-length

$$\mu_P^{max} \leftarrow \min\Bigg\{1, \min_{i:\Delta x_i^P<0}\left\{-\frac{x_i}{\Delta x_i^P}\right\}, \min_{i:\Delta s_i^P<0}\left\{-\frac{s_i}{\Delta s_i^P}\right\}, \min_{i:\Delta \nu_i^P<0}\left\{-\frac{\nu_i}{\Delta \nu_i^P}\right\}, \min_{i:\Delta \xi_i^P<0}\left\{-\frac{\xi_i}{\Delta \xi_i^P}\right\}\Bigg\}.$$

10: Compute corrector relaxation parameter

$$\tau_+ \leftarrow \frac{(\boldsymbol{\nu}+\mu_P^{max}\Delta\boldsymbol{\nu}^P)^T(\boldsymbol{x}+\mu_P^{max}\Delta\boldsymbol{x}^P)+(\boldsymbol{\xi}+\mu_P^{max}\Delta\boldsymbol{\xi}^P)^T(\boldsymbol{s}+\mu_P^{max}\Delta\boldsymbol{s}^P)}{2n}$$

and

$$\tau_K \leftarrow \tau\cdot\sigma \text{ with centering-parameter } \sigma := \left(\frac{\tau_+}{\tau}\right)^3.$$

11: Compute corrector-steps:

$$\text{Solve } \left(\mathbf{H}+\mathbf{M}\mathbf{D}^{-1}\mathbf{M}^T\right)\Delta\boldsymbol{u}^K = \boldsymbol{\rho}_1(\boldsymbol{x},\boldsymbol{y})+\mathbf{M}\mathbf{D}^{-1}\Big(\boldsymbol{\rho}_2^{\tau}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{s},\boldsymbol{\lambda},\boldsymbol{\nu},\boldsymbol{\xi})+\boldsymbol{\rho}^K(\boldsymbol{x},\boldsymbol{s},\boldsymbol{\nu},\boldsymbol{\xi},\Delta\boldsymbol{x}^P,\Delta\boldsymbol{s}^P,\Delta\boldsymbol{\nu}^P,\Delta\boldsymbol{\xi}^P)\Big)$$

$$\begin{aligned}
\Delta\boldsymbol{x}^K &\leftarrow \mathbf{D}^{-1}\Big(\mathbf{M}^T\Delta\boldsymbol{u}^K-\boldsymbol{\rho}_2^{\tau_K}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{s},\boldsymbol{\lambda},\boldsymbol{\nu},\boldsymbol{\xi})-\boldsymbol{\rho}^K(\boldsymbol{x},\boldsymbol{s},\boldsymbol{\nu},\boldsymbol{\xi},\Delta\boldsymbol{x}^P,\Delta\boldsymbol{s}^P,\Delta\boldsymbol{\nu}^P,\Delta\boldsymbol{\xi}^P\Big)\\
\Delta\boldsymbol{s}^K &\leftarrow \boldsymbol{\rho}_{p1}(\boldsymbol{x},\boldsymbol{s})-\Delta\boldsymbol{x}^K\\
\Delta\boldsymbol{\nu}^K &\leftarrow \boldsymbol{\rho}_{kkt1}^{\tau_K}(\boldsymbol{x},\boldsymbol{\nu})-\mathbf{X}^{-1}\mathbf{N}\Delta\boldsymbol{x}^K-\mathbf{X}^{-1}\Delta\mathbf{N}^P\Delta\mathbf{X}^P\mathbf{1}_n\\
\Delta\boldsymbol{\xi}^K &\leftarrow \boldsymbol{\rho}_{kkt2}^{\tau_K}(\boldsymbol{s},\boldsymbol{\xi})-\mathbf{S}^{-1}\boldsymbol{\Xi}\Delta\boldsymbol{s}^K-\mathbf{S}^{-1}\Delta\mathbf{S}^P\Delta\boldsymbol{\Xi}^P\mathbf{1}_n.
\end{aligned}$$

12: Choose damping factor $\eta\in[0.8,1.0]$ and compute maximum corrector step-length

$$\mu_K^{max} \leftarrow \min\Bigg\{1, \min_{i:\Delta x_i^K<0}\left\{-\frac{x_i}{\Delta x_i^K}\right\}, \min_{i:\Delta s_i^K<0}\left\{-\frac{s_i}{\Delta s_i^K}\right\}, \min_{i:\Delta \nu_i^K<0}\left\{-\frac{\nu_i}{\Delta \nu_i^K}\right\}, \min_{i:\Delta \xi_i^K<0}\left\{-\frac{\xi_i}{\Delta \xi_i^K}\right\}\Bigg\}.$$

13: Compute Newton-step-length

$$\mu \leftarrow \min\left\{1, \eta \cdot \mu_K^{max}\right\}.$$

14: Update

$$\begin{aligned}
\boldsymbol{u}^{k+1} &\leftarrow \boldsymbol{u} + \mu\Delta\boldsymbol{u}^K \\
\boldsymbol{x}^{k+1} &\leftarrow \boldsymbol{x} + \mu\Delta\boldsymbol{x}^K \\
\boldsymbol{s}^{k+1} &\leftarrow \boldsymbol{s} + \mu\Delta\boldsymbol{s}^K \\
\boldsymbol{\nu}^{k+1} &\leftarrow \boldsymbol{\nu} + \mu\Delta\boldsymbol{\nu}^K \\
\boldsymbol{\xi}^{k+1} &\leftarrow \boldsymbol{\xi} + \mu\Delta\boldsymbol{\xi}^K.
\end{aligned}$$

15: $k \leftarrow k + 1$

16: **end while**

Note, in step 9 and step 12, if no i exists satisfying $\Delta x_i < 0$, $\Delta s_i < 0$, $\Delta\nu_i < 0$, $\Delta\xi_i < 0$, then the step-length is set to $\mu = 1$. The damping factor η is usually geared to the quality of the corrector step. As rule of thumb, for small τ_K and large μ_K^{max} one chooses $\eta \approx 1$ to improve convergence.

Another issue is the determination of appropriate start values for Algorithm (6.1.1), i.e.

$$\boldsymbol{u}^0 = ((\boldsymbol{y}^0)^T, (\boldsymbol{\lambda}^0)^T)^T \in \mathbb{R}^{m+p}, \mathbf{0} < (\boldsymbol{x}^0, \boldsymbol{s}^0, \boldsymbol{\nu}^0, \boldsymbol{\xi}^0) \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n.$$

As a heuristic, we first compute start values $(\boldsymbol{u}^0, \boldsymbol{x}^0)$ by solving the reduced KKT-Newton-system (6.40) setting $\Delta\boldsymbol{u} = \boldsymbol{u}^0$, $\Delta\boldsymbol{x} = \boldsymbol{x}^0$, $\mathbf{S}^{-1}\boldsymbol{\Xi}\boldsymbol{\rho}_{p1}(\boldsymbol{x}, \boldsymbol{s}) = \boldsymbol{\rho}_{kkt1}^{\tau}(\boldsymbol{x}, \boldsymbol{\nu}) = \boldsymbol{\rho}_{kkt2}^{\tau}(\boldsymbol{s}, \boldsymbol{\xi}) = \mathbf{0}$, $\boldsymbol{x} = \mathbf{0}$, $\boldsymbol{y} = \mathbf{0}$, $\boldsymbol{\lambda} = \mathbf{0}$, $\boldsymbol{\xi} = \mathbf{0}$, $\boldsymbol{\nu} = \mathbf{0}$ and $\mathbf{D} = \mathbf{I}_n$. Moreover, $\mathbf{H}$ is substituted by the matrix $\tilde{\mathbf{H}} := \mathbf{H} + \boldsymbol{\Delta}$ yielding

$$\begin{pmatrix} \tilde{\mathbf{H}} & \mathbf{M} \\ \mathbf{M}^T & -\mathbf{I}_n \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^0 \\ \boldsymbol{x}^0 \end{pmatrix} = \begin{pmatrix} (\mathbf{0}^T, \boldsymbol{b}^T)^T \\ \boldsymbol{d} \end{pmatrix}, \tag{6.60}$$

$$\boldsymbol{\Delta} := \begin{pmatrix} \mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \in \mathbb{R}^{(m+p)\times(m+p)}. \tag{6.61}$$

Second, we initialize the variables $(\boldsymbol{x}^0, \boldsymbol{s}^0, \boldsymbol{\nu}^0, \boldsymbol{\xi}^0)$ dependent on the components of $\boldsymbol{x}^0$ such that the nonnegative condition is satisfied. The initialization heuristics works well in practice and is summarized in Algorithm (6.1.2).

Algorithm 6.1.2 QP-Solver's Initializer

Require: input data $\mathbf{A} \in \mathbb{R}^{p\times n}$, $\mathbf{B} \in \mathbb{R}^{m\times n}$, $\mathbf{Q} \in \mathbb{R}^{m\times m}$, $\boldsymbol{d} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{R}^p$, upper bounds $\mathbf{0} < \boldsymbol{u}_b \in \mathbb{R}^n$

Ensure: $\mathbf{Q}$ is positive definite, $\mathbf{A}$ is of full row rank

1: Solve $\left(\mathbf{H}+\boldsymbol{\Delta}+\mathbf{M}\mathbf{M}^T\right)\boldsymbol{u}^0=\left[\begin{pmatrix}\mathbf{0}\\ \boldsymbol{b}\end{pmatrix}+\mathbf{M}\boldsymbol{d}\right]$
2: $\boldsymbol{x}^0 \leftarrow \mathbf{M}^T\boldsymbol{u}^0-\boldsymbol{d}$
3: **for** $i=1$ to n **do**
4: **if** $x_i^0<0$ **then**
5: $x_i^0 \leftarrow eps^{2/3}\{eps \equiv$machine precision$\}$
6: **end if**
7: **if** $x_i^0>u_{b,i}^0$ **then**
8: $x_i^0 \leftarrow 0.9\cdot u_{b,i}^0$
9: **end if**
10: $s_i^0 \leftarrow u_{b,i}^0-x_i^0$
11: **if** $x_i^0=0$ **then**
12: $\nu_i^0 \leftarrow 1.0$, $\xi_i^0 \leftarrow 1.0$
13: **end if**
14: **if** $x_i^0>0$ **then**
15: $\xi_i^0 \leftarrow 5/4x_i^0$, $\nu_i^0 \leftarrow x_i^0/4$
16: **end if**
17: **end for**
18: **return** $(\boldsymbol{u}^0,\boldsymbol{x}^0,\boldsymbol{s}^0,\boldsymbol{\nu}^0,\boldsymbol{\xi}^0)$

At this point, all necessary components for solving the inner problem of the $\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz classifier algorithm are introduced. Hence, the next sections focus the development of an iterative solver for the outer problem.

6.2 An Iterative Solver for the Outer Optimization over the Convex Hull of Matrices

In the previous section an algorithm for solving the inner QP problem of the $\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz classifier algorithm was developed. In this section, we focus on the development of a solver that iterates a solution of the outer problem.

For this purpose, recall the *outer problem* of Algorithm (5.2.4)

$$\max_{\mathbf{K}\in conv(\mathcal{K})} q(\mathbf{K}) = -\min_{\mathbf{K}\in conv(\mathcal{K})} -q(\mathbf{K}) \tag{6.62}$$

where $q(\boldsymbol{K})=(\boldsymbol{\alpha}^*)^T\mathbf{1}_N-\frac{1}{2}(\boldsymbol{c}^*)^T\mathbf{K}(\boldsymbol{c}^*)$ is the optimal value of the QP problem (6.2) for a given matrix $\mathbf{K}$ of the convex hull $conv(\mathcal{K})$.

Solving the outer problem (6.62) is based on the idea to iterate a global solution by a sequence of solutions found with respect to a more and more growing subset of the convex hull $conv(\mathcal{K})$, which itself is spanned by matrices $\mathbf{K}_{(t)} \in conv(\mathcal{K}),\ 1 \leq t \leq T$ (Fig. (6.1)). The sequence of matrices $(\mathbf{K}_{(t)})_{t=1}^{T}$ is constructed such that the associated sequences of optimal values $(q(\mathbf{K}_{(t)}))_{t=1}^{T}$ is monotonously increasing, i.e. it holds

$$q(\mathbf{K}_{(1)}) < q(\mathbf{K}_{(2)}) < \cdots < q(\mathbf{K}_{(T)}). \tag{6.63}$$

The scheme of sequentially solving appropriate QP problems is attractive, because it enables a derivative-free optimization with respect to the matrix $\mathbf{K} \in conv(\mathcal{K})$. Moreover, in each step a new candidate matrix $\mathbf{K}_{(t+1)}$ is determined in an optimal sense, i.e. such that a maximum possible improvement is reached with respect to the next performed inner QP optimization $q(\mathbf{K}_{(t+1)})$.

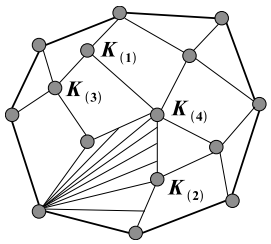


Figure 6.1: The convex hull $conv(\mathcal{K})$ can be step-wise constructed via a sequence of matrices $\mathbf{K}_{(t)} \in conv(\mathcal{K})$.

6.2.1 Optimizing for An Optimal Convex Combination

Let $(\boldsymbol{\alpha}_{(t)}, \boldsymbol{c}_{(t)})$ denote the optimal variables of the solved QP problem $q(\mathbf{K}_{(t)})$ in the t-th outer iteration. Further suppose a candidate matrix $\mathbf{K}^* \in conv(\mathcal{K})$ with $q(\mathbf{K}^*) > q(\mathbf{K}_{(t)})$. Then we can seek for the best new convex combination

$$\mathbf{K}_{(t+1)} := (1 - \mu^*) \cdot \mathbf{K}_{(t)} + \mu^* \cdot \mathbf{K}^* \tag{6.64}$$

with respect to $\mu^* \in [0, 1]$. By definition of $\mathbf{K}_{(t+1)}$ and the convex hull $conv(\mathcal{K})$ (Def. (5.2.4)), it follows $\mathbf{K}_{(t+1)} \in conv(\mathcal{K})$ for all $\mu^* \in [0, 1]$. Thus, it holds

$$\max_{\mu \in [0,1]} q\left((1 - \mu) \cdot \mathbf{K}_{(t)} + \mu \cdot \mathbf{K}^*\right) = q(\mathbf{K}_{(t+1)}) \geq q(\mathbf{K}^*) > q(\mathbf{K}_{(t)}) \tag{6.65}$$

where $\mathbf{K}_{(t+1)}$ is the new convex combination satisfying inequalities (6.63). Moreover, $\mathbf{K}_{(t+1)}$ is an optimal matrix from the convex hull in the sense that it maximizes the optimal value $q(\mathbf{K}_{(t+1)})$ of the inner QP problem which has to be solved in the next iteration.

In order to apply derivative based techniques, like e.g. projected-gradient methods, for solving the constrained maximization problem in (6.65) we need the following result about the right directional derivative of

$$g(\mu) := -q\left((1-\mu)\cdot\mathbf{K}_{(t)} + \mu\cdot\mathbf{K}^*\right). \tag{6.66}$$

Theorem 6.2.1. *Let be $\mathcal{S}\subseteq\mathbb{R}^m$ a nonempty, compact and convex set, and let be $\mathcal{T}\subset\mathbb{R}^n$ a nonempty, open and convex set. Assume the function $G : \mathcal{S}\times\mathcal{T}\to\mathbb{R}$ to be continuous and convex on $\mathcal{S}\times\mathcal{T}$. Then the right directional derivative of the function $g:\mathcal{T}\to\mathbb{R}$ defined as*

$$g(\boldsymbol{\mu}) := \min\left\{G(\boldsymbol{\alpha},\boldsymbol{\mu}) : \boldsymbol{\alpha}\in\mathcal{S}\right\} \tag{6.67}$$

with nonempty solution set

$$\mathcal{M}(\boldsymbol{\mu}) := \left\{\boldsymbol{\alpha}\in\mathcal{S} : G(\boldsymbol{\alpha},\boldsymbol{\mu}) = g(\boldsymbol{\mu})\right\} \tag{6.68}$$

in the direction $\boldsymbol{d}\in\mathcal{T}$ exists and reads as follows

$$g'(\boldsymbol{\mu};\boldsymbol{d}) := \min\left\{G'(\boldsymbol{\alpha},\boldsymbol{\mu};\boldsymbol{d}) : \boldsymbol{\alpha}\in\mathcal{M}(\boldsymbol{\mu})\right\}. \tag{6.69}$$

Here $G'(\boldsymbol{\alpha},\boldsymbol{\mu};\boldsymbol{d})$ denotes the right directional derivative of G with respect to $\boldsymbol{\mu}\in\mathcal{T}$ in the direction $\boldsymbol{d}\in\mathcal{T}$.

Proof. To prove the theorem we need the following three quite technical lemmata:

Lemma 6.2.1. *Let be $G:\mathcal{S}\times\mathcal{T}\to\mathbb{R}$ convex, and let be the set $\mathcal{S}\subseteq\mathbb{R}^m$ and $\mathcal{T}\subseteq\mathbb{R}^n$ nonempty and convex, then the function $g:\mathcal{T}\to\mathbb{R}$ defined as*

$$g(\boldsymbol{\mu}) := \inf\left\{G(\boldsymbol{\alpha},\boldsymbol{\mu}) : \boldsymbol{\alpha}\in\mathcal{S}\right\} \tag{6.70}$$

is convex if $g(\boldsymbol{\mu}) > -\infty$ for any $\boldsymbol{\mu}\in\mathcal{T}$.

Proof. For a proof see Appendix (B.2.1). ∎

Lemma 6.2.2. *Let be $\mathcal{T}\subseteq\mathbb{R}^n$ a nonempty, open and convex set, $g:\mathcal{T}\to\mathbb{R}$ a convex function, $\boldsymbol{\mu}\in\mathcal{T}$ and $\boldsymbol{d}\in\mathbb{R}^n$. Then it holds*

1. *any sequence $(q(t))_{t\in\mathbb{N}}$ of the differential quotient*

$$q(t) := \frac{g(\boldsymbol{\mu}+t\boldsymbol{d}) - g(\boldsymbol{\mu})}{t} \tag{6.71}$$

   is monotonously decreasing for $t\to 0^+$.

2. *the right directional derivative*

$$g'(\boldsymbol{\mu};\boldsymbol{d}) := \lim_{t\to 0^+} q(t) \tag{6.72}$$

   exists for all $\boldsymbol{\mu}\in\mathcal{T}$ in the direction $\boldsymbol{d}\in\mathbb{R}^n$, $\|\boldsymbol{d}\| = 1$. In particular, it even holds

$$g'(\boldsymbol{\mu};\boldsymbol{d}) = \inf_{t>0} q(t). \tag{6.73}$$

Proof. For a proof see Appendix (B.2.2). ∎

Lemma 6.2.3. *Let be the function $G : \mathcal{S} \times \mathcal{T} \to \mathbb{R}$ continuous, $\mathcal{T} \subseteq \mathbb{R}^m$, and let be $\mathcal{S} \subseteq \mathbb{R}^m$ a nonempty and compact set, then the function $g : \mathcal{T} \to \mathbb{R}$ with $g(\boldsymbol{\mu}) := \min \{G(\boldsymbol{\alpha}, \boldsymbol{\mu}) : \boldsymbol{\alpha} \in \mathcal{S}\}$ is continuous.*

Proof. For a proof see Appendix (B.2.3). ∎

Now, we are ready to prove the theorem. Because G is a convex function and $\mathcal{S}, \mathcal{T}$ are both nonempty convex sets, in virtue of Lemma (6.2.1) the function g is convex too. Therefore, and because $\mathcal{T}$ is open, due to Lemma (6.2.2) the right directional derivative $g'(\boldsymbol{\mu}; \boldsymbol{d})$ in the direction $\boldsymbol{d} \in \mathcal{S}$ exists. Further, for all $\boldsymbol{\alpha} \in \mathcal{M}(\boldsymbol{\mu})$ and $t > 0$ it holds

$$\frac{g(\boldsymbol{\mu} + t\boldsymbol{d}) - g(\boldsymbol{\mu})}{t} = \frac{\min \{G(\boldsymbol{\alpha}, \boldsymbol{\mu} + t\boldsymbol{d}) : \boldsymbol{\alpha} \in \mathcal{S}\} - G(\boldsymbol{\alpha}, \boldsymbol{\mu})}{t} \tag{6.74}$$

$$\leq \frac{G(\boldsymbol{\alpha}, \boldsymbol{\mu} + t\boldsymbol{d}) - G(\boldsymbol{\alpha}, \boldsymbol{\mu})}{t}. \tag{6.75}$$

Due to convexity, Lemma (6.2.2) also applies to the function $G(\boldsymbol{\alpha}, \cdot)$ for all $\boldsymbol{\alpha} \in S$. Thus, the limiting process $t \to 0^+$ in (6.74) and (6.75) implies

$$g'(\boldsymbol{\mu}; \boldsymbol{d}) \leq G'(\boldsymbol{\alpha}, \boldsymbol{\mu}; \boldsymbol{d}) \; \forall \boldsymbol{\alpha} \in \mathcal{M}(\boldsymbol{\mu}) \tag{6.76}$$

and in particular we have

$$g'(\boldsymbol{\mu}; \boldsymbol{d}) \leq \inf \{G'(\boldsymbol{\alpha}, \boldsymbol{\mu}; \boldsymbol{d}) : \boldsymbol{\alpha} \in \mathcal{M}(\boldsymbol{\mu})\}. \tag{6.77}$$

Now, we show that indeed equality holds throughout in (6.77). For this purpose let be $(t_n)_{n\in\mathbb{N}}$ a zero-sequence, i.e. $(t_n)_{n\in\mathbb{N}} \overset{n\to\infty}{\longrightarrow} 0^+$. Due to the compactness and non-emptiness of $\mathcal{S}$ it exist $\boldsymbol{\alpha}_n \in \mathcal{M}(\boldsymbol{\mu} + t_n\boldsymbol{d}) \subseteq \mathcal{S}$ for each t_n such that $g(\boldsymbol{\mu} + t_n\boldsymbol{d}) = G(\boldsymbol{\alpha}_n, \boldsymbol{\mu} + t_n\boldsymbol{d})$. Likewise, it exists a convergent subsequence $(\boldsymbol{\alpha}_{n_k})_{k\in\mathbb{N}}$ of $(\boldsymbol{\alpha}_n)_{n\in\mathbb{N}}$ with limit $\boldsymbol{\alpha}^* := \lim_{k\to\infty} \boldsymbol{\alpha}_{n_k}, \boldsymbol{\alpha}^* \in \mathcal{S}$.

Because of Lemma (6.2.3) the function g is continuous, and with

$$g(\boldsymbol{\mu} + t_{n_k}\boldsymbol{d}) = G(\boldsymbol{\alpha}_{n_k}, \boldsymbol{\mu} + t_{n_k}\boldsymbol{d}) \tag{6.78}$$

it follows that

$$g(\boldsymbol{\mu}) = \lim_{k\to\infty} g(\boldsymbol{\mu} + t_{n_k}\boldsymbol{d}) = \lim_{k\to\infty} G(\boldsymbol{\alpha}_{n_k}, \boldsymbol{\mu} + t_{n_k}\boldsymbol{d}) = G(\boldsymbol{\alpha}^*, \boldsymbol{\mu}). \tag{6.79}$$

This implies $\boldsymbol{\alpha}^* \in \mathcal{M}(\boldsymbol{\mu})$.

By definition $\forall \boldsymbol{\alpha} \in \mathcal{S} \wedge \boldsymbol{\alpha} \notin \mathcal{M}(\boldsymbol{\mu}) : g(\boldsymbol{\mu}) < G(\boldsymbol{\alpha}, \boldsymbol{\mu})$, thus for all $t_{n_k} > 0$ and $\boldsymbol{\alpha}_{n_k} \in \mathcal{M}(\boldsymbol{\mu} + t_{n_k}\boldsymbol{d})$ we have

$$g(\boldsymbol{\mu} + t_{n_k}\boldsymbol{d}) - g(\boldsymbol{\mu}) \geq G(\boldsymbol{\alpha}_{n_k}, \boldsymbol{\mu} + t_{n_k}\boldsymbol{d}) - G(\boldsymbol{\alpha}_{n_k}, \boldsymbol{\mu}). \tag{6.80}$$

In virtue of the *mean-value theorem of differential calculus*, there exists for any $t_{n_k} > 0$ a point $\boldsymbol{\xi}_{n_k} \in (\boldsymbol{\mu}, \boldsymbol{\mu} + t_{n_k}\boldsymbol{d})$ (i.e. the line between $\boldsymbol{\mu}$ and $\boldsymbol{\mu} + t_{n_k}\boldsymbol{d}$) with

$$G(\boldsymbol{\alpha}_{n_k}, \boldsymbol{\mu} + t_{n_k}\boldsymbol{d}) - G(\boldsymbol{\alpha}_{n_k}, \boldsymbol{\mu}) = t_{n_k} G'(\boldsymbol{\alpha}_{n_k}, \boldsymbol{\xi}_{n_k}; \boldsymbol{d}). \tag{6.81}$$

Insertion in (6.80) gives

$$\frac{g(\boldsymbol{\mu} + t_{n_k}\boldsymbol{d}) - g(\boldsymbol{\mu})}{t_{n_k}} \geq G'(\boldsymbol{\alpha}_{n_k}, \boldsymbol{\xi}_{n_k}; \boldsymbol{d}) \tag{6.82}$$

and for $k \to \infty$ we get

$$g'(\boldsymbol{\mu}; \boldsymbol{d}) \geq G'(\boldsymbol{\alpha}^*, \boldsymbol{\mu}; \boldsymbol{d}) \tag{6.83}$$

because $\boldsymbol{\xi}_{n_k} \overset{k\to\infty}{\longrightarrow} \boldsymbol{\mu}$ and $\boldsymbol{\alpha}_{n_k} \overset{k\to\infty}{\longrightarrow} \boldsymbol{\alpha}^*$.

■

In particular, if $G(\boldsymbol{\alpha}, \cdot)$ is differentiable we have

$$g'(\boldsymbol{\mu}; \boldsymbol{d}) = \min \left\{ \nabla_{\boldsymbol{\mu}} G(\boldsymbol{\alpha}, \boldsymbol{\mu})^T \boldsymbol{d} : \boldsymbol{\alpha} \in \mathcal{M}(\boldsymbol{\mu}) \subset \mathcal{S} \right\} = \nabla_{\boldsymbol{\mu}} G(\boldsymbol{\alpha}^*, \boldsymbol{\mu})^T \boldsymbol{d} \tag{6.84}$$

enabling us to solve the constrained maximization in (6.65) iteratively using e.g. steepest descent methods.

For this purpose, suppose we have given a parametrization $\boldsymbol{c} : \mathbb{R}^N \times [0,1] \to \mathbb{R}^M$ defined by a solution of the equation system $\overline{\mathbf{K}}(\mu)\boldsymbol{c}(\boldsymbol{\alpha}, \mu) = \mathbf{G}^T\mathbf{Y}\boldsymbol{\alpha}$ for all $\boldsymbol{\alpha} \in \mathbb{R}^N$ and $\mu \in [0,1]$ with

$$\overline{\mathbf{K}}(\mu) := (1 - \mu) \cdot \mathbf{K}_{(t)} + \mu \cdot \mathbf{K}^* \tag{6.85}$$

such that

$$G(\boldsymbol{\alpha}, \mu) := \frac{1}{2}\boldsymbol{c}(\boldsymbol{\alpha}, \mu)^T \overline{\mathbf{K}}(\mu)\boldsymbol{c}(\boldsymbol{\alpha}, \mu) - \boldsymbol{\alpha}^T \mathbf{1}_N \tag{6.86}$$

is continuous and convex. Further, let be $\mathcal{T} := \mathbb{R}$ and

$$g(\mu) = -q(\overline{\mathbf{K}}(\mu)) = \min_{\boldsymbol{\alpha} \in \mathcal{S}} G(\boldsymbol{\alpha}, \mu) \tag{6.87}$$

with feasible set

$$\mathcal{S} := \left\{ \boldsymbol{\alpha} \in \mathbb{R}^N \middle| \boldsymbol{\alpha}^T \boldsymbol{y} = 0, \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}_N \right\}. \tag{6.88}$$

Then, by these assumptions it holds

$$\max_{\mu \in [0,1]} q(\overline{\mathbf{K}}(\mu)) \quad = \quad - \min_{\mu \in [0,1]} -q(\overline{\mathbf{K}}(\mu)) = - \min_{\mu \in [0,1]} g(\mu) \tag{6.89}$$

and we can apply Theorem (6.2.1).

Note, compared to the QP-problem $q(\overline{\mathbf{K}}(\mu))$ (6.2) we eliminated here the equation system from the feasible set $\mathcal{S}_D(\overline{\mathbf{K}}(\mu))$ (6.3) which is now implicitly contained in the objective function $G(\boldsymbol{\alpha}, \mu)$ of the QP-problem $g(\mu)$. Thus, the feasible set $\mathcal{S}$ is independent of μ. Clearly, both representations of the problem are completely equivalent and they can be interchanged if necessary. Although we consider here the simple case of $\mu \in [0,1] \subset \mathbb{R}$, in virtue of Theorem (6.2.1) one could also try to solve for an optimal convex combination consisting of more than two matrices from the convex hull $conv(\mathcal{K})$.

By the way, Lemma (6.2.1) proves that optimizing for $\mu^* \in [0,1]$ is a convex optimization problem, that means a solution is always a global solution.

Corollary 6.2.1. *Let be $G : \mathcal{S} \times \mathcal{T} \to \mathbb{R}$ convex, then $\mu^* \in [0,1]$ is a global solution of the convex problem*

$$\min_{\mu \in [0,1]} g(\mu). \tag{6.90}$$

Proof. The proof follows directly from Lemma (6.2.1) and the equality (6.89).

■

However, the directional derivative depends on the parameterization $\boldsymbol{c}(\boldsymbol{\alpha}, \mu)$ of the solutions of the equation system $\overline{\mathbf{K}}(\mu)\boldsymbol{c} = \mathbf{G}^T\mathbf{Y}\boldsymbol{\alpha}$. Later on, in Section (6.3), we revisit this issue and specify two continuously differentiable parameterizations $\boldsymbol{c}(\boldsymbol{\alpha}, \mu)$ resulting in different implementations of the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz Classifier algorithm.

Next, we will present a particular projected gradient method which we use for solving the constrained maximization in (6.65), respectively problem (6.65), using (6.84).

6.2.2 Spectral Projected Gradient Method

The well-known *steepest descent method* from unconstrained optimization seeks for a search direction $\boldsymbol{d} \in \mathbb{R}^n$ that makes the approximate change in a function $g : \mathbb{R}^n \to \mathbb{R}$, i.e. the value of the directional derivative $g'(\boldsymbol{\mu}, \boldsymbol{d})$, as negative as possible for a sufficiently small step in the direction $\boldsymbol{d}$. The motivation is that g can be approximated near $\boldsymbol{\mu}$ in the direction $\boldsymbol{d}$ for small $\eta > 0$ by

$$g(\boldsymbol{\mu} + \eta\boldsymbol{d}) \approx g(\boldsymbol{\mu}) + \eta g'(\boldsymbol{\mu}; \boldsymbol{d}). \tag{6.91}$$

Thus, generating a sequence of iterates $\boldsymbol{\mu}_{k+1} := \boldsymbol{\mu}_k + \eta_k \Delta\boldsymbol{\mu}_k$ with a steepest descent direction

$$\Delta\boldsymbol{\mu}_k := \operatorname*{arg\,min}_{\boldsymbol{d} \in \mathbb{R}^n} \{g'(\boldsymbol{\mu}_k; \boldsymbol{d}) : \|\boldsymbol{d}\| = 1, g'(\boldsymbol{\mu}_k; \boldsymbol{d}) < 0\} \tag{6.92}$$

yields a maximum decrease $g(\boldsymbol{\mu}_{k+1}) - g(\boldsymbol{\mu}_k) < 0$. That means, the sequence $(g(\boldsymbol{\mu}_k))_{k\in\mathbb{N}}$ is monotonously decreasing if the step-length $\eta^k \geq 0$ is chosen appropriately in each iteration (assuming the directional derivative exists for all $\boldsymbol{\mu}_k$). If there exists no such descent direction at a point $\boldsymbol{\mu}^* \in \mathbb{R}^n$ then the point is said to be *stationary*. In particular, if g is differentiable then it follows $\Delta\boldsymbol{\mu}_k = -\nabla g(\boldsymbol{\mu}_k)/\|\nabla g(\boldsymbol{\mu}_k)\|_2$ from (6.92) with respect to the Euclidean norm $\|\boldsymbol{d}\|_2 = 1$. In this case, the steepest descent method reduces to the well-known *gradient descent method* for which it holds $\nabla g(\boldsymbol{\mu})^T\boldsymbol{d} < 0$ for a descent direction $\boldsymbol{d}$. Using the gradient descent method, any point $\boldsymbol{\mu}^* \in \mathbb{R}^n$ is stationary if the necessary condition $\nabla g(\boldsymbol{\mu}^*) = \mathbf{0}$ is satisfied.

In steepest descent methods the step-length η_k is sometimes exactly computed via a solution of

$$g(\boldsymbol{\mu}_k + \eta_k\Delta\boldsymbol{\mu}_k) = min\,\{g(\boldsymbol{\mu}_k + \eta\Delta\boldsymbol{\mu}_k) : \eta \geq 0\} \tag{6.93}$$

which is called *exact line search*. Most line searches used in practice are *inexact*: the step length is chosen to approximately minimize g along the ray $\{\boldsymbol{\mu}_k + \eta\Delta\boldsymbol{\mu}_k : \eta \geq 0\}$. A very simple and quite effective inexact search in unconstrained optimization is called *backtracking line search* which starts with $\eta = 1$ and then reduces it by some factor $0 < \beta < 1$ until the stopping condition

$$g(\boldsymbol{\mu}_k + \eta\Delta\boldsymbol{\mu}_k) < g(\boldsymbol{\mu}_k) + \alpha \cdot \eta \cdot g'(\boldsymbol{\mu}_k; \Delta\boldsymbol{\mu}_k) \tag{6.94}$$

holds for fixed $0 < \alpha < 0.5$ (for details see e.g. [Boyd 2004][Sec. 9.2]). However, in constrained optimization, one must be careful in applying backtracking because one could leave the feasible set.

As discussed in the preceding section, given an appropriate parameterization, G can be assumed to be convex and differentiable and it holds $g'(\mu; d) = \partial_\mu G(\boldsymbol{\alpha}^*, \mu) \cdot d$ where $\boldsymbol{\alpha}^*$ is a (global) solution of the parameterized problem (6.87). Because μ is constrained to the convex set $[0, 1]$, we implemented a *projected gradient descent method*. Projected gradient descent methods maintain feasibility by projecting the iterates on the feasible set (Fig. 6.2). This process is in general expensive. Moreover, even if the projection is inexpensive to compute, like e.g. in case of box-constraints, the method is considered to be slow, because it suffers typically from zig-zagging trajectory like its analogue, the gradient descent method. On the other hand, the projected gradient method is quite simple to implement. In our simple case of $[0, 1]$ the projection is very easily derived to be

$$proj_{[0,1]}(\mu) := \begin{cases} \mu = 0 & \text{if } \mu < 0 \\ \mu & \text{if } 0 < \mu < 1 \\ \mu = 1 & \text{if } \mu > 1 \end{cases} . \tag{6.95}$$

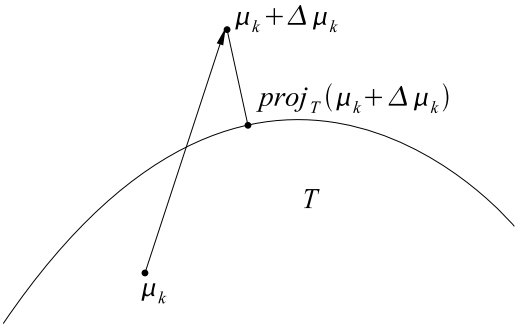


Figure 6.2: Projection $proj_{\mathcal{T}}(\boldsymbol{\mu}_k + \Delta\boldsymbol{\mu}_k)$ of an iterate to a feasible set $\mathcal{T}$.

Unfortunately, even in this simple case, our first attempts showed that often the computed derivatives $\partial_\mu G(\boldsymbol{\alpha}^*, \mu)$ are very small making the determination of appropriate step-length very difficult. Thus, a standard implementation of a projected gradient descent method with monotone line-search, like backtracking, failed most of the time with slow convergence rates (if convergence took place at all). Thus, we concluded that without using second-order derivatives solving for the best convex combination is doomed to failure. On the other hand, the effort of computing second order derivatives of the parametrization we will later use in (6.87) is prohibitive. Therefore, we decided to use a *non-monotone spectral projected gradient method* proposed by [Birgin 2000], [Birgin 2009]. Their method combines two ingredients, first a non-monotone line search yielding a not necessarily monotone decreasing sequence $(g(\boldsymbol{\mu}_k))_{k\in\mathbb{N}}$, and second a generalized projected gradient descent method, called *spectral projected gradient method*, which is related to the family of *quasi-Newton* methods [Dennis 1977]. That means, the non-monotone spectral projected gradient method uses first- and second-order derivatives. But contrary to Newton's method, in quasi-Newton methods second-order derivatives are approximated using first-order derivatives.

The non-monotone spectral projected gradient method by [Birgin 2000] is summarized in Algorithm (6.2.1) with adapted notations for solving problem (6.87), respectively problem (6.65).

Algorithm 6.2.1 Non-monotone Spectral Projected Gradient Solver (cf. [Birgin 2000])

Require: the function $g : [0,1] \to \mathbb{R}$, the derivative function $\partial_\mu G : \mathcal{S} \times [0,1] \to \mathbb{R}$, a start value $\mu_0 \in [0,1]$, $1 \leq M \in \mathbb{N}$, a small $\alpha_{min} > 0$, a large $\alpha_{max} > \alpha_{min}$, a sufficient decrease parameter $\gamma \in (0,1)$, safeguarding parameters $0 < \sigma_1 < \sigma_2 < 1$, an accuracy $\epsilon > 0$, and a maximum number of iterations $k_{max} \in \mathbb{N}$.

Ensure: $\alpha_0 \in [\alpha_{min}, \alpha_{max}]$

for $k = 0$ to k_{max} **do**
  if $|proj_{[0,1]}(\mu_k - \partial_\mu G(\boldsymbol{\alpha}^*_{\mu_k}, \mu_k))| < \epsilon$ **then**
    return μ_k {stop, μ_k is stationary}
  end if
  {Backtracking:}
  Compute $d_k = proj_{[0,1]}(\mu_k - \alpha_k \partial_\mu G(\boldsymbol{\alpha}^*_{\mu_k}, \mu_k)) - \mu_k$.
  Set $\lambda \leftarrow 1$.
  loop
    Set $\mu_+ = \mu_k + \lambda d_k$.
    if {Test non-monotone step-length criterion}

$$g(\mu_+) \leq \max_{0 \leq j \leq \min\{k, M-1\}} g(\mu_{k-j}) + \gamma \cdot \lambda \cdot d_k \cdot \partial_\mu G(\boldsymbol{\alpha}^*_{\mu_k}, \mu_k) \tag{6.96}$$

    then
      break loop
    else
      {Compute safeguarded new trial step-length:}
      Set $\delta = d_k \cdot \partial_\mu G(\boldsymbol{\alpha}^*_{\mu_k}, \mu_k)$.
      Set $\lambda_{new} = -\frac{1}{2}\lambda^2 \delta / (g(\mu_+) - g(\mu_k) - \lambda\delta)$.
      if $(\lambda_{new} \geq \sigma_1)$ and $(\lambda_{new} \leq \sigma_2 \lambda)$ **then**
        $\lambda \leftarrow \lambda_{new}$
      else
        $\lambda \leftarrow \lambda/2$
      end if
    end if
  end loop
  {Updates:}
  $\lambda_k = \lambda$
  $\mu_{k+1} = \mu_+$
  $s_k = \mu_{k+1} - \mu_k$
  $y_k = \partial_\mu G(\boldsymbol{\alpha}^*_{\mu_{k+1}}, \mu_{k+1}) - \partial_\mu G(\boldsymbol{\alpha}^*_{\mu_k}, \mu_k)$.
  Compute $b_k = s_k \cdot y_k$.
  if $b_k \leq 0$ **then**
    Set $\alpha_{k+1} = \alpha_{max}$
  else
    Compute $a_k = s_k \cdot s_k$.
    Set $\alpha_{k+1} = \min\{\alpha_{max}, \max\{\alpha_{min}, a_k/b_k\}\}$.
  end if
end for

Obviously, the most expensive part of Algorithm (6.2.1) is the computation of the test criterion (6.96) in the backtracking loop, because we have

to solve $g(\mu_+) = \min_{\boldsymbol{\alpha}\in\mathcal{S}} G(\boldsymbol{\alpha}, \mu_+)$ using Algorithm (6.1.1) in each iteration. Additionally, we have to store the $0 \leq j \leq M-1$ previous function values $g(\mu_{k-j})$ with respect to $g(\mu_k)$ in a look-up table to determine (6.96). Fortunately, due to the update $\mu_{k+1} = \mu_+$ we can use a solution $\boldsymbol{\alpha}^*_{\mu_{k+1}}$ associated to $g(\mu_+)$ in each backtracking and main iteration to determine $\partial_\mu G(\boldsymbol{\alpha}^*_{\mu_{k+1}}, \mu_{k+1})$ without extra costs. In the practical implementation the settings $\gamma = 10^{-4}$, $\alpha_{min} = 10^{-30}$, $\alpha_{max} = 10^{30}$, $\sigma_1 = 0.1$, $\sigma_2 = 0.9$, $\alpha_0 = 1/|\partial_\mu G(\boldsymbol{\alpha}^*_{\mu_0}, \mu_0)|$ and $M = 10$ turned out to be useful.

So far, we assumed a new candidate matrix $\mathbf{K}^* \in conv(\mathcal{K})$ is given. The question remains how to get a matrix satisfying $q(\mathbf{K}^*) > q(\mathbf{K}_{(t)})$. In the next section we answer this question by proposing a stochastic search heuristics.

6.2.3 Stochastically Searching for A Candidate Matrix via Simulated Annealing

For solving the outer optimization (6.62), we propose a two stage optimization scheme: In the first stage, a candidate matrix $\mathbf{K}^* \in conv(\mathcal{K})$ has to be found such that $q(\mathbf{K}^*) > q(\mathbf{K}_{(t)})$ is satisfied with respect to the current iterate $\mathbf{K}_{(t)} \in conv(\mathcal{K})$. Then in the second stage the best linear combination $\mathbf{K}_{(t+1)} = (1-\mu^*) \cdot \mathbf{K}_{(t)} + \mu^* \cdot \mathbf{K}^*$ is determined by minimization of $g(\mu)$ regarding $\mu \in [0,1]$ and using Algorithm (6.2.1). Because the convex hull $conv(\mathcal{K})$ is spanned by matrices $\mathbf{K}(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{X}$, the idea is to stochastically search for a candidate matrix $\mathbf{K}^* = \mathbf{K}(\boldsymbol{x}^*)$ over the domain $\mathcal{X}$. Even if the function g is convex, the function $q(\mathbf{K}(\cdot)) : \mathcal{X} \to \mathbb{R}$ may be highly nonlinear over its domain $\mathcal{X}$. If it fluctuates not too dramatically, it is plausible that the next new candidate matrix will be in some small vicinity of the predecessor candidate matrix. On the other hand, if $q(\boldsymbol{K}(\cdot))$ varies strongly it is very likely that a stochastic search based on a simple random walk will trap in some local minimum $q(\mathbf{K}(\boldsymbol{x}^*)) \leq q(\mathbf{K}_{(t)})$. To give an idea, a hypothetical search space is outlined in Figure (6.3).

The solid curve in Figure (6.3) represents the function $q(\mathbf{K}(\cdot))$ on $\mathcal{X}$ and the dashed curves show the situation for the concave function $-g(\mu)$ which interpolates $q(\mathbf{K}_{(t)})$ and $q(\mathbf{K}(\boldsymbol{x}_{(t)}))$ for $\mu \in [0,1]$. As shown, once trapped into the minimum at the point $\mathbf{K}(\boldsymbol{x}^*)$ it is not possible to improve the q function via optimizing the interpolant g. Clearly, to get as much improvement as possible per iteration we would like to find, for example when starting at $\mathbf{K}_{(t)}$, directly the point $\boldsymbol{K}(\boldsymbol{x}_{(t+1)})$ instead of the point $\boldsymbol{K}(\boldsymbol{x}_{(t)})$. Ideally, we would find the global maximum. But this requires a prohibitive number of trial steps in $\mathcal{X}$. On the other hand, the benefit from optimizing the convex combination is that we need just good local maxima as supporting points for the interpolant in order to sufficiently improve the q function in each iteration. Thus, a search

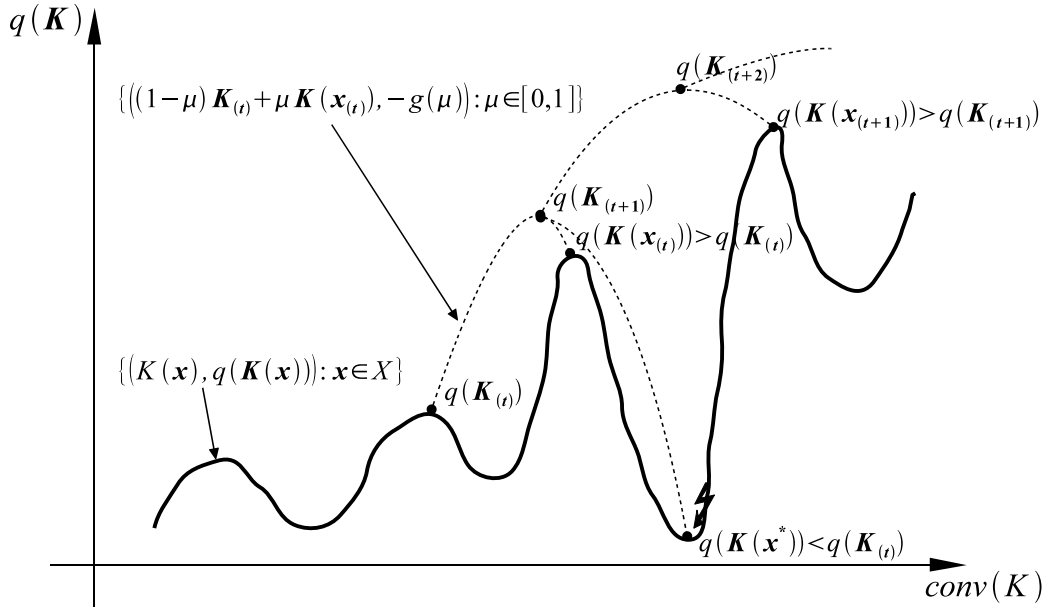


Figure 6.3: A hypothetical search space.

heuristic is needed that negotiate a compromise between improvement and number of trial steps per iteration. Additionally, the heuristic should try to avoid to get stuck in local minima. For this reason, we used a *Simulated Annealing* (SA) [Kirkpatrick 1983], [Schneider 2006] based stochastic search.

The SA approach is a classic algorithm for finding solutions, also called low-energy states or optimum configurations, of complex optimization problems that can not be solved analytically. SA has its analogon in the ancient technique for creating metal with desirable mechanical properties. After a period of heating up the metal to its melting temperature, the metal can be slowly cooled resulting in a softer, more ductile material. If desired, the metal can also be rapidly quenched to a low temperature yielding a harder surface. The motivation to use SA for optimization is that usually physical systems end up in low energy states if cooling is performed slowly enough. And such systems only reach a less desirable local minimum energy state if quenched down rapidly. Thus, SA simulates a sufficiently slow cooling process in order to end up in a (global) minimum of the energy landscape of a considered objective function. Starting at a high temperature the system is free to randomly take any configuration even with high energy value, i.e. all system states can be visited with nearly equal probability. Then proceeding with gradually lowering the temperature to zero the randomness decreases, with the effect that less and less high energy configurations are explored, and finally the system relaxes into a low energy state (also called *ground state*).

In SA one assumes physical systems that behave like moving gas particles

at a temperature T_k. In this case the *Boltzmann distribution*

$$p(\sigma) = \frac{1}{Z(T_k)} \exp\left(-\frac{h(\sigma)}{k_B T_k}\right) \tag{6.97}$$

describes the probability the system is in a specific energetic state $\sigma \in \Gamma$ with respect to the energy function $h : \Gamma \to \mathbb{R}$, respectively the fraction of particles that have a specific kinetic energy. In (6.97), $Z(T_k)$ is the partition sum

$$Z(T_k) = \sum_{\sigma \in \Gamma} \exp\left(-\frac{h(\sigma)}{k_B T_k}\right) \tag{6.98}$$

with the Boltzmann constant $k_B = 1.3807 \cdot 10^{-23} J/K$ that is usually neglected in practical SA algorithms. For the purpose of randomly exploring the state space Γ, one has to sample from the Boltzmann distribution (6.97) which is intractable due to the computation of the partition sum (6.98). Fortunately, [Metropolis 1953] observed that *Markov chains*, which have the desired distribution as equilibrium distribution, can be used for generating a sequence of moves in the state space instead of a simple random walk. One can show (e.g. [Liu 2001]), starting from a state σ and moving to a new state τ with probability $\pi(\sigma \to \tau)$ (*transition probability*) that satisfies the *detailed balance equation*

$$p(\sigma)\pi(\sigma \to \tau) = p(\tau)\pi(\tau \to \sigma) \tag{6.99}$$

is sufficient for the associated Markov chain to have $p(\sigma)$ as its invariant equilibrium distribution. In particular this holds even for any arbitrary starting distribution.

Inserting Equation (6.97) into Equation (6.99) yields

$$\frac{\pi(\sigma \to \tau)}{\pi(\tau \to \sigma)} = \exp\left(-\frac{h(\tau) - h(\sigma)}{k_B T_k}\right). \tag{6.100}$$

Thus, the transition probability ratio of a move $\sigma \to \tau$ and the inverse move $\tau \to \sigma$ has to depend only on the energy difference $\Delta h = h(\tau) - h(\sigma)$ as well as the temperature T. Most often, one chooses the *Metropolis criterion*

$$\pi(\sigma \to \tau) = \begin{cases} \exp\left(-\frac{\Delta h}{k_B T_k}\right) & \text{if } \Delta h > 0 \\ 1 & \textit{otherwise} \end{cases} \tag{6.101}$$

and $\pi(\tau \to \sigma) = 1$.

That means with respect to the SA algorithm, we have to run a Markov chain to equilibrium at temperature T_k using the Metropolis criterion first. Then from adjacent samples produced by the chain we can obtain a new configuration ν which is effectively produced from the Boltzmann distribution

(6.97). The procedure is repeated at each temperature level T_k of a sequence $(T_k)_{k\in\mathbb{N}}$ with $\lim_{k\to\infty} T_k = 0$. A function producing a temperature sequence is called *cooling schedule*. In [Geman 1984] it is shown that with probability one a global optimum can be reach using a cooling schedule like

$$T_k = \frac{a}{b + log(k)}. \tag{6.102}$$

The parameters a and b are problem dependent. This cooling schedule is extremely slow and is therefore not used in practice. Usually, an exponential cooling is employed, i.e.

$$T_k = T_0 \cdot r^k \tag{6.103}$$

with start temperature T_0 and some reduction factor $r \in [0.8, 0.999]$.

The SA algorithm we implemented is summarized in Algorithm (6.2.2).

Algorithm 6.2.2 SA Algorithm for Finding a New Candidate Matrix (cf. [Kirkpatrick 1983])

Require: start state $\boldsymbol{x}_0 \in \mathcal{X}$, start temperature $T_0 > 0$, reduction factor $r \in [0.8, 0.999]$, an energy function $h_{\mathbf{K}} : \mathcal{X} \to \mathbb{R}$ implicitly dependent on $\mathbf{K}(\boldsymbol{x})$, max. energy value h_{max}, variance σ^2 of a Gaussian random walker, max. number of SA loops $k_{max} \in \mathbb{N}$, max. number of equilibration steps $i_{max} \in \mathbb{N}$

Set $k = 0$.
Set $h_0 = h_{\mathbf{K}}(\boldsymbol{x}_0)$.
while $k \leq k_{max}$ **do**
  {Metropolis-Hastings-loop:}
  for $i = 1$ to i_{max} **do**
    Simulate $\boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \cdot \mathbf{I})$
    Compute $\boldsymbol{x}_p = \boldsymbol{x}_k + \boldsymbol{\delta}$ {proposal state}
    if $\boldsymbol{x}_p \notin \mathcal{X}$ **then**
      Reflect $\boldsymbol{x}_p$ at boundary.
    end if
    Compute $h_p = h_{\mathbf{K}}(\boldsymbol{x}_p)$ {proposal energy}
    if $h_p \leq h_k$ **then**
      {proposal state has lower or equal energy}
      $h_k \leftarrow h_p$
      $\boldsymbol{x}_k \leftarrow \boldsymbol{x}_p$
    else
      {proposal state has higher energy}
      Simulate uniformly distributed $u \sim \mathcal{U}[0, 1]$.
      Compute $\Delta h = h_p - h_k$.
      if $\ln(u) \leq -\Delta h / T_k$ **then**

21: {accept proposal state with Boltzmann probability}
22: $h_k \leftarrow h_p$
23: $\boldsymbol{x}_k \leftarrow \boldsymbol{x}_p$
24: **end if**
25: **end if**
26: **end for**
27: **if** $h_k < h_{max}$ **then**
28: {new candidate matrix found!}
29: **return** $\mathbf{K}(\boldsymbol{x}^*)$ with $\boldsymbol{x}^* = \boldsymbol{x}_k$
30: **end if**
31: {updates:}
32: Set $h_{k+1} = h_k$.
33: Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$.
34: Set $T_{k+1} = T_0 \cdot r^k$.
35: $k \leftarrow k + 1$
36: **end while**

Clearly, the energy function $h_{\mathbf{K}}$ depends on $\mathcal{X}$ via the matrix $\mathbf{K}(\boldsymbol{x})$. But in Algorithm (6.2.2) we made this dependency not explicit. The reason is, that we have to change the energy function suitable for the different realizations of the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm, as we will see in Section (6.3.1) and Section (6.3.2).

Contrary to a standard SA implementation, we stop the stochastic search in Line (27) if an improving matrix $\mathbf{K}(\boldsymbol{x}^*)$ is found, because good local maxima are sufficient for the outer optimization (6.62). Further, in Line (8), we reflect the random walker at the boundary of $\mathcal{X}$, if he is going to leave the set $\mathcal{X}$. For example, in the practical implementations of the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm presented in the next sections, we assume $\mathcal{X}$ to be a hypercube $[a, b]^d$. Thus, a reflection can be effectively performed by scaling all components $x_p^j \notin [a, b]$ of a proposal state $\boldsymbol{x}_p$ by a small factor, say e.g. 0.85. It is important to note that in Line (2) and Line (11) a QP-problem has to be solved using Algorithm (6.1.1). We will come back to this issue in Section (6.3.1).

The number of Metropolis-Hastings iterations, i_{max}, influences the equilibration of the Markov chain for sampling from the Boltzmann distribution at temperature T_k. In many practical SA implementations this number is set to unity, i.e. no equilibration takes place and sampling is performed only approximately from a Boltzmann distribution. In general, one has to carefully select the starting temperature, the number of equilibration steps, the number of SA loops, the reduction factor and the variance of the random walker dependent of the problem data. As rule of thumb, we set these parameters such that

approximately for the first 25% of the SA loops, k_{max}, a 100% acceptance rate of the proposal configurations is maintained during equilibration, and for the rest of the SA loops the acceptance rate decreases slowly with at most a 25% acceptance rate in the last 25% of all loops. Starting with a relative high temperature value, e.g. $T_0 = 1000$, we carefully adjusted the variance of the random walker for a trade-off between quality and overhead of the search. If the variance is high, the search is very crude and many local minima are ignored. On the other hand, if the variance is set to very small values, many steps are involved for finding good configurations. We found out, that the variance is often well adjusted if on average an improving matrix is found somewhere between the iterations marked by the 100% and 25% acceptance rates. In cases it is not possible to maintain the corresponding 100% and 25% acceptance rates using a high start temperature T_0, we reduced T_0 to control the upper rate. Clearly, if the temperature is decreased, the acceptance rates must increase eventually. The lower rate can be controled via the number of SA loops k_{max} and the reduction factor r. Most of the time, a suitable value for the reduction factor turned out to be 0.8, thus we changed only k_{max} to reach the lower rate.

6.3 The Assembling of All Parts: Complete Implementations of the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz Classifier

In Section (6.1) and Section (6.2), we developed the building blocks for an implementation of the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz Classifier. In the following, we assemble all components considering particular realizations of the algorithm. We suggest two realizations that are due to different parameterizations of the solutions of the involved equation system $\overline{\mathbf{K}}(\mu)\boldsymbol{c} = \mathbf{G}^T\mathbf{Y}\boldsymbol{\alpha}$. A suitable parameterization $\boldsymbol{c}(\boldsymbol{\alpha}, \mu)$ should ideally satisfy two goals: First, the matrix $\mathbf{Q}$ considered for the QP-solver (Def. 6.1.1) has to be positive definite. Second, in order to apply Theorem (6.2.1) the feasible set $\mathcal{S}_D$ has to be equivalently reformulated independently of μ in terms of the feasible set $\mathcal{S}$, while the function G has to be continuous and convex. Starting in the next section with a very simple solution using regularization, in the subsequent section another approach based on an algebraic parameterization is presented.

6.3.1 The Regularized $\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz Classifier Algorithm

The matrix $\overline{\mathbf{K}}(\mu)=(1-\mu)\cdot\mathbf{K}_{(t)}+\mu\cdot\mathbf{K}^*,\ \mu\in[0,1]$ is positive semi-definite as a convex combination of positive semi-definite matrices. Hence, the inhomogeneous system of equations $\overline{\mathbf{K}}(\mu)\boldsymbol{c}=\mathbf{G}^T\mathbf{Y}\boldsymbol{\alpha}$ of the feasible set $\mathcal{S}_D(\overline{\mathbf{K}}(\mu))$ has no unique solution $\boldsymbol{c}^*\in\mathbb{R}^M$. If there is given a particular solution $\boldsymbol{c}(\boldsymbol{\alpha}^*,\mu)$ associated to an optimal point $(\boldsymbol{\alpha}^*,\boldsymbol{c}(\boldsymbol{\alpha}^*,\mu))\in\mathcal{S}_D(\overline{\mathbf{K}}(\mu))$ for some $\mu\in[0,1]$, then every $\boldsymbol{c}^*=\boldsymbol{c}(\boldsymbol{\alpha}^*,\mu)+\lambda\boldsymbol{c}_0$ with $\boldsymbol{c}_0\in\mathit{null}(\overline{\mathbf{K}}(\mu))$, i.e. the null space of $\overline{\mathbf{K}}(\mu)$, and $\lambda\in\mathbb{R}$, solves the problem with identical optimal value $q(\overline{\mathbf{K}}(\mu))$. This can easily verified by insertion of $\boldsymbol{c}^*$ into the QP-problem (6.2).

An intuitive way to avoid this ambiguity is a perturbation of the set $\mathcal{K}=\left\{0\preceq\mathbf{K}(\boldsymbol{x})\in\mathbb{R}^{M\times M}\mid\boldsymbol{x}\in\mathcal{X}\right\}$ [Stuhlsatz 2007c]. For this purpose, we introduce the set

$$\mathcal{K}_\epsilon:=\{\mathbf{K}+\epsilon\mathbf{I}_M:\mathbf{K}\in\mathcal{K}\}\tag{6.104}$$

for a sufficient small $\epsilon>0$. In this case we get

Lemma 6.3.1. *Any matrix $\mathbf{K}\in conv(\mathcal{K}_\epsilon)$ is positive definite.*

Proof. For any $\epsilon>0$, $\mathbf{0}\neq\boldsymbol{x}\in\mathbb{R}^M$ and $\mathbf{K}\in\mathcal{K}$ it holds

$$\frac{\boldsymbol{x}^T\left(\mathbf{K}+\epsilon\mathbf{I}\right)\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}}\geq\min_{\mathbf{0}\neq\boldsymbol{x}\in\mathbb{R}^M}\frac{\boldsymbol{x}^T\mathbf{K}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}}+\min_{\mathbf{0}\neq\boldsymbol{x}\in\mathbb{R}^M}\frac{\boldsymbol{x}^T\epsilon\mathbf{I}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}}=\lambda_{min}^{\mathbf{K}}+\epsilon>0\tag{6.105}$$

where $\lambda_{min}^{\mathbf{K}}\geq0$ is the smallest eigenvalue of $\mathbf{K}\succeq0$.

■

By the way, it is easy to see, that such a perturbation can also equivalently be introduced by an additional constraint $\|\boldsymbol{c}\|_2^2\leq\delta$, $\delta>0$, in the feasible set (5.78) of the Primal Convex SIP Algorithm (5.2.3) of the $\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz classifier algorithm. The reason is, that there is an one-to-one correspondence between the Lagrange coefficient of the additional constraint and the perturbation ϵ.

Due to Lemma (6.3.1), for matrices $\mathbf{K}_{(t)},\mathbf{K}^*\in conv(\mathcal{K}_\epsilon)$ the matrix $\overline{\mathbf{K}}(\mu)$ is positive definite yielding an unique solution $\boldsymbol{c}^*=\boldsymbol{c}(\boldsymbol{\alpha},\mu)=\overline{\mathbf{K}}(\mu)^{-1}\mathbf{G}^T\mathbf{Y}\boldsymbol{\alpha}$ for any $\mu\in[0,1]$ and $\boldsymbol{\alpha}\in\mathbb{R}^N$. Insertion of $\boldsymbol{c}(\boldsymbol{\alpha},\mu)$ in (6.86) gives

$$G(\boldsymbol{\alpha},\mu)=\frac{1}{2}\boldsymbol{\alpha}^T\mathbf{Y}\mathbf{G}\overline{\mathbf{K}}(\mu)^{-1}\mathbf{G}^T\mathbf{Y}\boldsymbol{\alpha}-\boldsymbol{\alpha}^T\mathbf{1}_N\tag{6.106}$$

which is continuous on $\mathbb{R}^N\times[0,1]$. Convexity with respect to $\boldsymbol{\alpha}$ is obvious and convexity with respect to μ follows by

Lemma 6.3.2. *Let be* $\mathbf{A}(\mu) := (1-\mu)\mathbf{A}_1 + \mu\mathbf{A}_2$ *for* $\mu \in \mathcal{T}$ *with open set* $\mathcal{T} \subseteq \mathbb{R}$, *and let be* $\mathbf{A}_1, \mathbf{A}_2 \in \mathbb{R}^{N\times N}$ *symmetric positive definite matrices. Then, for* $\boldsymbol{x} \in \mathbb{R}^N$ *the function* $f_{\boldsymbol{x}} : \mathcal{T} \to \mathbb{R}$ *with* $f_{\boldsymbol{x}}(\mu) := \boldsymbol{x}^T\mathbf{A}(\mu)^{-1}\boldsymbol{x} = \boldsymbol{y}(\boldsymbol{x},\mu)^T\mathbf{A}(\mu)\boldsymbol{y}(\boldsymbol{x},\mu)$ *and* $\boldsymbol{y}(\boldsymbol{x},\mu) := \mathbf{A}(\mu)^{-1}\boldsymbol{x}$ *is convex. Its first- and second-order derivatives read as*

$$\begin{aligned} f'_{\boldsymbol{x}}(\mu) &= \boldsymbol{x}^T\mathbf{A}(\mu)^{-1}\mathbf{B}\mathbf{A}(\mu)^{-1}\boldsymbol{x} = \boldsymbol{y}(\boldsymbol{x},\mu)^T\mathbf{B}\boldsymbol{y}(\boldsymbol{x},\mu) && (6.107)\\ f''_{\boldsymbol{x}}(\mu) &= 2\boldsymbol{x}^T\mathbf{A}(\mu)^{-1}\mathbf{B}\mathbf{A}(\mu)^{-1}\mathbf{B}\mathbf{A}(\mu)^{-1}\boldsymbol{x} \\ &= 2\boldsymbol{y}(\boldsymbol{x},\mu)^T\mathbf{B}\mathbf{A}(\mu)^{-1}\mathbf{B}\mathbf{A}(\mu)^{-1}\boldsymbol{y}(\boldsymbol{x},\mu) && (6.108)\end{aligned}$$

with $\mathbf{B} := \mathbf{A}_1 - \mathbf{A}_2$.

Proof. Because it holds for all $\mu \in \mathcal{T}$

$$\left(\mathbf{A}(\mu)^{-1}\right)' = -\mathbf{A}(\mu)^{-1}\mathbf{A}'(\mu)\mathbf{A}(\mu)^{-1} = \mathbf{A}(\mu)^{-1}\left(\mathbf{A}_1 - \mathbf{A}_2\right)\mathbf{A}(\mu)^{-1} \quad (6.109)$$

it follows Equation (6.107). Due to symmetry, we get $(\mathbf{A}(\mu)^{-1}\mathbf{B}\mathbf{A}(\mu)^{-1})' = (\mathbf{A}(\mu)^{-1})'\mathbf{B}\mathbf{A}(\mu)^{-1} + \mathbf{A}(\mu)^{-1}\mathbf{B}(\mathbf{A}(\mu)^{-1})' = 2(\mathbf{A}(\mu)^{-1})'\mathbf{B}\mathbf{A}(\mu)^{-1}$, i.e. it follows Equation (6.108) with

$$\left(\mathbf{A}(\mu)^{-1}\right)'' = 2\left(\mathbf{A}(\mu)^{-1}\right)'\left(\mathbf{A}_1 - \mathbf{A}_2\right)\mathbf{A}(\mu)^{-1}. \quad (6.110)$$

Now, because of the symmetry of $\mathbf{A}(\mu)$, we have the eigenvalue decomposition $\mathbf{A}(\mu)^{-1} = \mathbf{U}_\mu\mathbf{D}_\mu\mathbf{U}_\mu^T$ for all $\mu \in \mathcal{T}$ where $\mathbf{D}_\mu$ is diagonal with positive diagonal elements and $\mathbf{U}_\mu\mathbf{U}_\mu^T = \mathbf{I}$. Hence, with $\mathbf{C}_\mu := \sqrt{2}\mathbf{D}_\mu^{1/2}\mathbf{U}_\mu^T\mathbf{B}\mathbf{A}(\mu)^{-1}$ it holds for all $\mu \in \mathcal{T}$ that

$$f''_{\boldsymbol{x}}(\mu) = \boldsymbol{x}^T\mathbf{C}_\mu^T\mathbf{C}_\mu\boldsymbol{x} = \|\mathbf{C}_\mu\boldsymbol{x}\|_2^2 \geq 0 \quad (6.111)$$

implying the convexity of $f_{\boldsymbol{x}}(\mu)$.

∎

Thus, from Lemma (6.3.2), we obtain for $\mathbf{K}_{(t)}, \mathbf{K}^* \in conv(\mathcal{K}_\epsilon)$ that

$$\partial_\mu G(\boldsymbol{\alpha},\mu) = \frac{1}{2}\boldsymbol{c}(\boldsymbol{\alpha},\mu)^T\left(\mathbf{K}_{(t)} - \mathbf{K}^*\right)\boldsymbol{c}(\boldsymbol{\alpha},\mu). \quad (6.112)$$

Because the primal variables $\boldsymbol{c}$ are eliminated from the feasible set $\mathcal{S}_D(\overline{\mathbf{K}}(\mu))$ of Algorithm (5.2.4), it remains the feasible set

$$\mathcal{S} = \left\{\boldsymbol{\alpha} \in \mathbb{R}^N \middle| \boldsymbol{\alpha}^T\boldsymbol{y} = 0, \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}_N\right\} \quad (6.113)$$

that is independent of $\overline{\mathbf{K}}(\mu)$, respectively μ.

In virtue of Theorem (6.2.1), the right directional derivative of $g(\mu) = -q(\overline{\mathbf{K}}(\mu)) = \min_{\boldsymbol{\alpha}\in\mathcal{S}} G(\boldsymbol{\alpha},\mu)$ is given by

$$g'(\mu;1) = \partial_\mu G(\boldsymbol{\alpha}^*_\mu,\mu) \tag{6.114}$$

with the global minimizer $\boldsymbol{\alpha}^*_\mu := \arg\min_{\boldsymbol{\alpha}\in S} G(\boldsymbol{\alpha},\mu)$. And due to Corollary (6.2.1) the problem of finding the optimal convex combination, $\min_{\mu\in[0,1]} g(\mu)$, has a global solution $\mu^* \in [0,1]$. In order to iterate a global minimizer μ^* we used the spectral projected gradient solver (Alg. (6.2.1)) together with the QP-solver (Alg. 6.1.1) to solve for $\boldsymbol{\alpha}^*_\mu$ in each iteration. The benefit from using our modified QP-solver is that the large matrix $\overline{\mathbf{K}}(\mu)$ must not be inverted explicitly in each iteration. Instead $\boldsymbol{c}(\boldsymbol{\alpha}^*_\mu,\mu) = \boldsymbol{c}^*_\mu$ is directly returned by the QP-solver as an optimal feasible solution $(\boldsymbol{\alpha}^*_\mu,\boldsymbol{c}^*_\mu) \in \mathcal{S}_D(\overline{\mathbf{K}}(\mu))$.

A very nice property of the perturbation heuristics, is that the stochastic search can be performed very efficiently. During stochastic search we have to check the criterion whether the current energy value $h_k = -q(\mathbf{K}(\boldsymbol{x}_k))$ is greater than the proposal energy value $h_p = -q(\mathbf{K}(\boldsymbol{x}_p))$, and to terminate if $h_k < h_{max} = -q(\mathbf{K}_{(t)})$. Thus, in each iteration of the SA algorithm (Alg. 6.2.2) we must solve a QP-problem. Fortunately, in case of a perturbed matrices $\mathbf{K} \in conv(\mathcal{K}_\epsilon)$ it is possible to bound the q values.

For this purpose, let be

$$Q(\boldsymbol{\alpha},\boldsymbol{c},\boldsymbol{K}) := \boldsymbol{\alpha}^T\mathbf{1}_N - \frac{1}{2}\boldsymbol{c}^T\mathbf{K}\boldsymbol{c}. \tag{6.115}$$

Recall, that for optimal points $(\boldsymbol{\alpha}^*,\boldsymbol{c}(\boldsymbol{\alpha}^*,\mathbf{K})) \in \mathcal{S}_D(\mathbf{K})$ and $(\boldsymbol{\alpha}^*_{(t)},\boldsymbol{c}^*_{(t)}) \in \mathcal{S}_D(\mathbf{K}_{(t)})$ with $Q(\boldsymbol{\alpha}^*,\boldsymbol{c}(\boldsymbol{\alpha}^*,\mathbf{K}),\mathbf{K}) = q(\mathbf{K})$ and $Q(\boldsymbol{\alpha}^*_{(t)},\boldsymbol{c}^*_{(t)},\mathbf{K}_{(t)}) = q(\mathbf{K}_{(t)})$, it holds $\boldsymbol{c}(\boldsymbol{\alpha}^*,\mathbf{K}) = \mathbf{K}^{-1}\mathbf{G}^T\mathbf{Y}\boldsymbol{\alpha}^*$, $\boldsymbol{c}^*_{(t)} = \mathbf{K}^{-1}_{(t)}\mathbf{G}^T\mathbf{Y}\boldsymbol{\alpha}_{(t)}$ and $\boldsymbol{\alpha}^*,\boldsymbol{\alpha}^*_{(t)} \in \mathcal{S}$. Thus, if we seek via the SA algorithm for a matrix $\mathbf{K} \in conv(\mathcal{K}_\epsilon)$ satisfying $Q(\boldsymbol{\alpha}^*_{(t)},\boldsymbol{c}(\boldsymbol{\alpha}^*_{(t)},\mathbf{K}),\mathbf{K}) > Q(\boldsymbol{\alpha}^*_{(t)},\boldsymbol{c}^*_{(t)},\mathbf{K}_{(t)})$ then it follows the inequality

$$q(\mathbf{K}) \geq Q(\boldsymbol{\alpha}^*_{(t)},\boldsymbol{c}(\boldsymbol{\alpha}^*_{(t)},\mathbf{K}),\mathbf{K}) > Q(\boldsymbol{\alpha}^*_{(t)},\boldsymbol{c}^*_{(t)},\mathbf{K}_{(t)}) = q(\mathbf{K}_{(t)}). \tag{6.116}$$

Fortunately, this means a quadratic problem $q(\mathbf{K}_{(t)})$ has to be solved for $\boldsymbol{\alpha}^*_{(t)} \in \mathcal{S}_D(\mathbf{K}_{(t)})$ only once. Practically, for the requirements of the SA Algorithm (6.2.2), one simply has to set the function $h_{\mathbf{K}}$ to be

$$h_{\mathbf{K}}(\boldsymbol{x}) := -Q(\boldsymbol{\alpha}^*_{(t)},\boldsymbol{c}(\boldsymbol{\alpha}^*_{(t)},\mathbf{K}(\boldsymbol{x})),\mathbf{K}(\boldsymbol{x})) \tag{6.117}$$

with $\mathbf{K}(\boldsymbol{x}) \in \mathcal{K}_\epsilon$ and $h_{max} := -Q(\boldsymbol{\alpha}^*_{(t)},\boldsymbol{c}^*_{(t)},\mathbf{K}_{(t)})$.

On the other hand, the computation of $Q(\boldsymbol{\alpha}^*_{(t)},\boldsymbol{c}^*,\mathbf{K}(\boldsymbol{x}))$ requires the computation of the inverse of $\mathbf{K}(\boldsymbol{x}) \in \mathbb{R}^{M\times M}$. However, this inversion can be performed very efficiently using the *Woodbury matrix identity* [Woodbury 1950]:

$$(\mathbf{E}+\mathbf{FGH})^{-1} = \mathbf{E}^{-1} - \mathbf{E}^{-1}\mathbf{F}\left(\mathbf{G}^{-1}+\mathbf{HE}^{-1}\mathbf{F}\right)^{-1}\mathbf{HE}^{-1}. \tag{6.118}$$

By definition (5.95), any matrix $\mathbf{K}(\boldsymbol{x}) \in \mathcal{K}$ can be factorized as $\mathbf{K}(\boldsymbol{x}) = \boldsymbol{\Psi}(\boldsymbol{x})^T\boldsymbol{\Psi}(\boldsymbol{x})$ with $\boldsymbol{\Psi}(\boldsymbol{x}) := (\nabla\Phi_1(\boldsymbol{x}),\ldots,\nabla\Phi_M(\boldsymbol{x})) \in \mathbb{R}^{m\times M}$. Hence, for any matrix $\mathbf{K}(\boldsymbol{x}) \in \mathcal{K}_\epsilon$, the Woodbury matrix identity yields

$$\begin{aligned}\mathbf{K}(\boldsymbol{x})^{-1} &= \left(\epsilon\mathbf{I}_M + \boldsymbol{\Psi}(\boldsymbol{x})^T\mathbf{I}_m\boldsymbol{\Psi}(\boldsymbol{x})\right)^{-1} &(6.119)\\ &= \frac{1}{\epsilon}\mathbf{I}_M - \frac{1}{\epsilon^2}\boldsymbol{\Psi}(\boldsymbol{x})^T\left(\mathbf{I}_m + \boldsymbol{\Psi}(\boldsymbol{x})\boldsymbol{\Psi}(\boldsymbol{x})^T\right)^{-1}\boldsymbol{\Psi}(\boldsymbol{x}). &(6.120)\end{aligned}$$

Because the matrix $\mathbf{I}_m+\boldsymbol{\Psi}(\boldsymbol{x})\boldsymbol{\Psi}(\boldsymbol{x})^T$ is mostly of dimension $m \ll M$, inversion is quite practicable with low computational costs.

The fully implementable regularized $\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz classifier algorithm is summarized in Algorithm (6.3.1) [Stuhlsatz 2007c].

Algorithm 6.3.1 Regularized $\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz Classifier Algorithm

Require: Training set $\emptyset \neq \mathcal{O}_N \subset \mathcal{Z} = \mathcal{X}\times\mathcal{Y}$, $\mathcal{X} \subset \mathbb{R}^m$ compact and convex, $1 \leq n \leq M$ basis functions $\Phi_n(\boldsymbol{x}) \in \mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$, trade-off parameter $C > 0$, distortion $\epsilon > 0$, $\mathbf{K}_{(0)} \in conv(\mathcal{K}_\epsilon)$, $T \in \mathbb{N}$

Ensure: $\mathcal{K}_\epsilon := \{\mathbf{K} + \epsilon\mathbf{I}_M : \mathbf{K} \in \mathcal{K}\}$, $\mathcal{K} := \left\{0 \preceq \mathbf{K}(\boldsymbol{x}) \in \mathbb{R}^{M\times M} \mid \boldsymbol{x} \in \mathcal{X}\right\}$, $\exists(\boldsymbol{x}_j,y_j),(\boldsymbol{x}_i,y_i) \in \mathcal{O}_N \;:\; y_j = -1 \wedge y_i = 1$

1: Set $t = 0$.

2: **loop**

3: Solve for

$$(\boldsymbol{\alpha}^*_{(t)},\boldsymbol{c}^*_{(t)}) \longleftarrow \underset{(\boldsymbol{\alpha},\boldsymbol{c})\in\mathcal{S}_D(\mathbf{K}_{(t)})}{\arg\min} -Q(\boldsymbol{\alpha},\boldsymbol{c},\mathbf{K}_{(t)})$$

with feasible set

$$\mathcal{S}_D(\mathbf{K}_{(t)}) := \left\{(\boldsymbol{\alpha},\boldsymbol{c}) \in \mathbb{R}^N\times\mathbb{R}^M \;\middle|\; \mathbf{K}_{(t)}\boldsymbol{c} = \mathbf{G}^T\mathbf{Y}\boldsymbol{\alpha}, \boldsymbol{\alpha}^T\boldsymbol{y} = 0, \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}_N\right\}$$

using the QP-Solver Algorithm (6.1.1) and the Initializer (6.1.2).

4: **if** (t=T) **then**

5: **return**

$$\begin{aligned}\boldsymbol{c}^* &\longleftarrow \boldsymbol{c}^*_{(t)}\\ b^* &\longleftarrow -\frac{1}{2|\mathcal{A}_+|}\left(\sum_{i\in\mathcal{A}_+}\sum_{n=1}^{M}c^*_n\Phi_n(\boldsymbol{x}_i) + \sum_{j\in\mathcal{A}_-}\sum_{n=1}^{M}c^*_n\Phi_n(\boldsymbol{x}_j)\right)\end{aligned}$$

with $\mathcal{I} := \{i \in \mathbb{N} : 0 < \alpha_i < C\}$, $\mathcal{A}_+ \subseteq \{i \in \mathcal{I} : y_i = 1\}$ and $\mathcal{A}_- \subseteq \{j \in \mathcal{I} : y_j = -1\}$ such that $|\mathcal{A}_+| = |\mathcal{A}_-|$.

6: **end if**

7: Search for a candidate matrix $\mathbf{K}^* := \mathbf{K}(\boldsymbol{x}^*) \in conv(\mathcal{K}_\epsilon)$, $\boldsymbol{x}^* \in \mathcal{X}$, satisfying

$$h_{\mathbf{K}}(\boldsymbol{x}^*) = -Q(\boldsymbol{\alpha}^*_{(t)},\boldsymbol{c}(\boldsymbol{\alpha}^*_{(t)},\mathbf{K}(\boldsymbol{x}^*)),\mathbf{K}(\boldsymbol{x}^*)) < h_{max} := -Q(\boldsymbol{\alpha}^*_{(t)},\boldsymbol{c}^*_{(t)},\mathbf{K}_{(t)})$$

using the SA-Algorithm (6.2.2) with

$$h_{\mathbf{K}}(\boldsymbol{x}) := -Q(\boldsymbol{\alpha}^*_{(t)}, \boldsymbol{c}(\boldsymbol{\alpha}^*_{(t)}, \mathbf{K}(\boldsymbol{x})), \mathbf{K}(\boldsymbol{x}))$$

and the inversion-formula (6.120) to compute

$$\boldsymbol{c}(\boldsymbol{\alpha}^*_{(t)}, \mathbf{K}(\boldsymbol{x})) := \mathbf{K}(\boldsymbol{x})^{-1}\mathbf{G}^T\mathbf{Y}\boldsymbol{\alpha}^*_{(t)}.$$

8: Solve for

$$\mu^* \longleftarrow \operatorname*{arg\,min}_{\mu\in[0,1]} \; g(\mu)$$

using the Gradient Solver Algorithm (6.2.1) with

$$g(\mu) := -Q(\boldsymbol{\alpha}^*_\mu, \boldsymbol{c}^*_\mu, \overline{\mathbf{K}}(\mu))$$

and the derivative

$$\partial_\mu G(\boldsymbol{\alpha}^*_\mu, \mu) := \frac{1}{2}\boldsymbol{c}(\boldsymbol{\alpha}^*_\mu, \mu)^T \left(\mathbf{K}_{(t)} - \mathbf{K}^*\right) \boldsymbol{c}(\boldsymbol{\alpha}^*_\mu, \mu)$$

whereas $\boldsymbol{c}(\boldsymbol{\alpha}^*_\mu, \mu) := \boldsymbol{c}^*_\mu$ is computed for each $\mu \in [0,1]$ by

$$(\boldsymbol{\alpha}^*_\mu, \boldsymbol{c}^*_\mu) \longleftarrow \operatorname*{arg\,min}_{(\boldsymbol{\alpha},\boldsymbol{c})\in\mathcal{S}_D(\overline{\mathbf{K}}(\mu))} -Q(\boldsymbol{\alpha}, \boldsymbol{c}, \overline{\mathbf{K}}(\mu))$$

using the QP-Solver Algorithm (6.1.1) and the Initializer (6.1.2).

9: Set $\mathbf{K}_{(t+1)} = \overline{\mathbf{K}}(\mu^*)$.

10: $t \longleftarrow t + 1$.

11: **end loop**

6.3.2 The Non-Regularized $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz Classifier Algorithm

In the preceding section, we presented the fully implementable $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm in case of a regularization of the coefficients $\boldsymbol{c} \in \mathbb{R}^M$, which results in a perturbed version $\mathcal{K}_\epsilon$ of the set $\mathcal{K}$. It turned out, that the use of the set $\mathcal{K}_\epsilon$ yields some desirable properties for an efficient implementation. On the other hand, the distortion $\epsilon > 0$ can not be made arbitrarily small due to numerical reasons. This can be seen, because any matrix $\mathbf{K} \in \mathcal{K}$ is positive semi-definite and can be factorized by $\mathbf{K} = \mathbf{\Psi}^T\mathbf{\Psi}$ with $\mathbf{\Psi} \in \mathbb{R}^{m\times M}$. It follows the rank of $\mathbf{K}$ is at most m. Thus, for a very small $\epsilon > 0$ the matrix $\mathbf{K}$ becomes ill-conditioned and has adverse effects on

the QP-Solver as well as on the inversion of $\mathbf{K}$ in the SA-algorithm. Because the size of the distortion ϵ depends on the problem data and in particular on the chosen basis functions, we need a modified parameterization $\boldsymbol{c}(\boldsymbol{\alpha},\mu)$ which avoids large perturbations if the problem data and the basis functions constitute an extremely ill-conditioned situation.

We start our discussion with a result about the rank of a positive linear combination of two symmetric positive semi-definite matrices:

Lemma 6.3.3. *Let be* $\mathbf{K}_1,\mathbf{K}_2 \in \mathbb{R}^{M\times M}$ *symmetric positive semi-definite,* $r_1 := rank(\mathbf{K}_1)$, $r_2 := rank(\mathbf{K}_2)$, $\mu_1,\mu_2 > 0$ *and*

$$\mathbf{K}(\mu_1,\mu_2) := \mu_1\mathbf{K}_1 + \mu_2\mathbf{K}_2. \tag{6.121}$$

Then, the rank of $\mathbf{K}(\mu_1,\mu_2)$ *is independent of* μ_1,μ_2, *i.e.*

$$r_1 + r_2 \geq rank\,(\mathbf{K}(\mu_1,\mu_2)) = r \geq \max\{r_1,r_2\}\ \forall\mu_1,\mu_2 > 0 \tag{6.122}$$

with $r := M - dim(null(\mathbf{K}(\mu_1,\mu_2)))$ *and*

$$null\,(\mathbf{K}(\mu_1,\mu_2)) = null(\mathbf{K}_1) \cap null(\mathbf{K}_2)\ \forall\mu_1,\mu_2 > 0. \tag{6.123}$$

Proof. Because $\forall\mu_1,\mu_2 \geq 0$ the matrix $\mathbf{K}(\mu_1,\mu_2)$ is positive semi-definite. It follows

$$\forall\mu_1,\mu_2 > 0, \boldsymbol{v} \in null(\mathbf{K}(\mu_1,\mu_2)) : \boldsymbol{v}^T\mathbf{K}(\mu_1,\mu_2)\boldsymbol{v} = 0 \tag{6.124}$$

which implies $\boldsymbol{v}^T\mathbf{K}_1\boldsymbol{v} = \boldsymbol{v}^T\mathbf{K}_2\boldsymbol{v} = 0$, respectively $\|\mathbf{K}_1^{\frac{1}{2}}\boldsymbol{v}\|^2 = \|\mathbf{K}_2^{\frac{1}{2}}\boldsymbol{v}\|^2 = 0$.

Hence, it holds $\forall\mu_1,\mu_2 > 0 : null(\mathbf{K}(\mu_1,\mu_2)) = null(\mathbf{K}_1) \cap null(\mathbf{K}_2)$ and

$$\begin{aligned} r_1 + r_2 \geq rank\,(\mathbf{K}(\mu_1,\mu_2)) &= M - dim(null(\mathbf{K}(\mu_1,\mu_2)) \\ &\geq M - \min\{dim(null(\mathbf{K}_1)), dim(null(\mathbf{K}_2))\} \\ &= \max\{r_1,r_2\}. \end{aligned} \tag{6.125}$$

■

Due to Lemma (6.3.3) the rank of a matrix

$$\overline{\mathbf{K}}(\mu) = (1-\mu)\cdot\mathbf{K}_{(t)} + \mu\cdot\ \mathbf{K}^* \tag{6.126}$$

is independent of $\mu \in (0,1)$. This suggests, that it is possible to find a algebraic parameterization $\boldsymbol{c}(\boldsymbol{\alpha},\mu)$ of a particular solution of the equation system $\overline{\mathbf{K}}(\mu)\boldsymbol{c} = \mathbf{G}^T\mathbf{Y}\boldsymbol{\alpha}$ for any matrix $\overline{\mathbf{K}}(\mu) \in conv(\mathcal{K})$:

Lemma 6.3.4. *Let be* $\mathbf{K}_1, \mathbf{K}_2 \in \mathbb{R}^{M\times M}$ *symmetric positive semi-definite,* $r_1 := rank(\mathbf{K}_1)$, $r_2 := rank(\mathbf{K}_2)$. *Then for all* $\mu \in (0|1)$ *the matrix*

$$\mathbf{K}(\mu) := (1-\mu)\cdot\mathbf{K}_1 + \mu\cdot\mathbf{K}_2 \tag{6.127}$$

can be transformed such that

$$\mathbf{V}^T\mathbf{K}(\mu)\mathbf{V} = \left(\begin{array}{c|c} \mathbf{J}(\mu) & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} \end{array}\right) \in \mathbb{R}^{M\times M}. \tag{6.128}$$

The matrix $\mathbf{J}(\mu) \in \mathbb{R}^{r\times r}$ *is symmetric and positive definite for all* $\mu \in (0,1)$ *and it holds*

$$\mathbf{J}(\mu) = (1-\mu)\mathbf{A} + \mu\mathbf{B} \tag{6.129}$$

with $\mathbf{A} := \mathbf{V}_r^T\mathbf{K}_1\mathbf{V}_r$ *and* $\mathbf{B} := \mathbf{V}_r^T\mathbf{K}_2\mathbf{V}_r$.

The matrix $\mathbf{V}_r = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r) \in \mathbb{R}^{M\times r}$ *calculated by the Eigenvalue Decomposition (EV)*

$$\mathbf{V}^T\left(\mathbf{K}_1 + \mathbf{K}_2\right)\mathbf{V} = \mathbf{\Lambda} \tag{6.130}$$

consists of the orthonormal eigenvectors $\boldsymbol{v}_k \in \mathbb{R}^M$ *associated to the eigenvalues* $\lambda_k > 0, 1 \le k \le r$. *It holds* $r_1 + r_2 \ge r = rank(\mathbf{J}(\mu)) \ge \max\{r_1, r_2\}$ $\forall\mu \in (0,1)$.

Proof. Consider the eigenvalue decomposition of the matrix

$$\mathbf{K}_1 + \mathbf{K}_2 = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T \tag{6.131}$$

where $\mathbf{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_M) \in \mathbb{R}^{M\times M}$ is an unitary matrix consisting of eigenvectors $\boldsymbol{v}_k \in \mathbb{R}^M$ and $\mathbf{\Lambda} \in \mathbb{R}^{M\times M}$ is a diagonal matrix with eigenvalues $\lambda_k \ge 0, 1 \le k \le M$ on its diagonal entries.
For any eigenvector $\boldsymbol{v}_k \in \mathbb{R}^M$ associated to its eigenvalue $\lambda_k \ge 0$ it holds

$$(\mathbf{K}_1 + \mathbf{K}_2)\boldsymbol{v}_k = \lambda_k\boldsymbol{v}_k,\ 1 \le k \le M. \tag{6.132}$$

In virtue of Lemma (6.3.3) it holds $rank(\mathbf{K}_1 + \mathbf{K}_2) = r$. Thus, it follows $\lambda_{r+1} = \cdots = \lambda_M = 0$.

Because for any $\mu \in (0,1)$ there corresponds an $\eta \in (-1,1)$ such that

$$\mathbf{K}(\mu) = (1-\mu)\cdot\mathbf{K}_1 + \mu\cdot\mathbf{K}_2 = \frac{\mathbf{K}_1 + \mathbf{K}_2}{2} + \eta\frac{\mathbf{K}_2 - \mathbf{K}_1}{2}. \tag{6.133}$$

it follows that $\mathbf{K}(\mu)$ can be transformed using $\mathbf{V}$ in the right hand side matrix (6.128) with positive definite $\mathbf{J}(\mu)$ of rank r for all $\mu \in (0,1)$.

■

The application of Lemma (6.3.4) let us parameterize all solutions $\boldsymbol{c}^* = \boldsymbol{c}(\boldsymbol{\alpha}, \mu) + \lambda \boldsymbol{c}_0$ with $\boldsymbol{c}_0 \in \mathit{null}(\overline{\mathbf{K}}(\mu))$ of the equation system $\overline{\mathbf{K}}(\mu)\boldsymbol{c} = \mathbf{G}^T\mathbf{Y}\boldsymbol{\alpha}$ for all $\boldsymbol{\alpha} \in \mathcal{S}_D(\overline{\mathbf{K}}(\mu))$ and $\mu \in (0, 1)$. For this purpose, let be

$$\boldsymbol{c} := \mathbf{V}\boldsymbol{d} = (\mathbf{V}_r, \mathbf{V}_{r+1})\boldsymbol{d} \tag{6.134}$$

with $\boldsymbol{d} := (\boldsymbol{d}_r^T, \boldsymbol{d}_{r+1}^T)^T, \boldsymbol{d}_r \in \mathbb{R}^r, \boldsymbol{d}_{r+1} \in \mathbb{R}^{M-r}$, $\mathbf{V}_r \in \mathbb{R}^{M\times r}$ as defined in Lemma (6.3.4), and $\mathbf{V}_{r+1} := (\boldsymbol{v}_{r+1}, \ldots, \boldsymbol{v}_M) \in \mathbb{R}^{M\times M-r}$ consisting of the eigenvectors $\boldsymbol{v}_k \in \mathbb{R}^M$ associated to the eigenvalues $\lambda_k = 0$, $r+1 \leq k \leq M$.

It follows $\overline{\mathbf{K}}(\mu)\boldsymbol{c} = \mathbf{G}^T\mathbf{Y}\boldsymbol{\alpha}$, $\mu \in (0, 1)$, can be equivalently reformulated as

$$\mathbf{V}^T\overline{\mathbf{K}}(\mu)\mathbf{V}\boldsymbol{d} = \begin{pmatrix} \overline{\mathbf{J}}(\mu)\boldsymbol{d}_r \\ \mathbf{0} \end{pmatrix} = \mathbf{V}^T\mathbf{G}^T\mathbf{Y}\boldsymbol{\alpha}. \tag{6.135}$$

Hence, with $\boldsymbol{d}_r = \overline{\mathbf{J}}(\mu)^{-1}\mathbf{V}_r^T\mathbf{G}^T\mathbf{Y}\boldsymbol{\alpha}$ and arbitrary $\boldsymbol{d}_{r+1} \in \mathbb{R}^{M-r}$, all solutions are given by

$$\boldsymbol{c}^* = \mathbf{V}_r\overline{\mathbf{J}}(\mu)^{-1}\mathbf{V}_r^T\mathbf{G}^T\mathbf{Y}\boldsymbol{\alpha} + \mathbf{V}_{r+1}\boldsymbol{d}_{r+1}. \tag{6.136}$$

That means the parametrization we are looking for is given by

$$\boldsymbol{c}(\boldsymbol{\alpha}, \mu) := \mathbf{V}_r\overline{\mathbf{J}}(\mu)^{-1}\mathbf{V}_r^T\mathbf{G}^T\mathbf{Y}\boldsymbol{\alpha} \tag{6.137}$$

while $\boldsymbol{c}_0 := \mathbf{V}_{r+1}\boldsymbol{d}_{r+1}$ parameterizes the null space of $\overline{\mathbf{K}}(\mu)$. Because $\boldsymbol{d}_{r+1}$ is arbitrary, and it does not have any influence on the objective function value $Q(\boldsymbol{\alpha}, \boldsymbol{c}^*, \overline{\mathbf{K}}(\mu))$, it can be set to zero or to any particular setting that results in a desired property. For example, one could try to optimize $\boldsymbol{d}_{r+1}$ such that the solution $\boldsymbol{c}^*$ is sparse, i.e. many components are nearly zero. However, in the present thesis, we do not further discuss this issue.

In what follows, let be

$$Q_r(\boldsymbol{\alpha}, \boldsymbol{d}_r, \mathbf{J}) := \boldsymbol{\alpha}^T\mathbf{1}_N - \frac{1}{2}\boldsymbol{d}_r^T\mathbf{J}\boldsymbol{d}_r \tag{6.138}$$

the *reduced objective function* and let be

$$q_r(\mathbf{J}) := \max_{(\boldsymbol{\alpha}, \boldsymbol{d}_r)\in\mathcal{S}_{D_r}(\mathbf{J})} Q_r(\boldsymbol{\alpha}, \boldsymbol{d}_r, \mathbf{J}) \tag{6.139}$$

the associated *reduced QP-problem* with *reduced feasible set*

$$\begin{aligned}\mathcal{S}_{D_r}(\mathbf{J}) \;:=\; \Big\{ (\boldsymbol{\alpha}, \boldsymbol{d}_r) \in \mathbb{R}^N \times \mathbb{R}^r \;\Big|\; \mathbf{J}\boldsymbol{d}_r = (\mathbf{V}_r^{\mathbf{K}})^T\mathbf{G}^T\mathbf{Y}\boldsymbol{\alpha}, \boldsymbol{\alpha}^T\boldsymbol{y} = 0, \\ (\mathbf{V}_{r+1}^{\mathbf{K}})^T\mathbf{G}^T\mathbf{Y}\boldsymbol{\alpha} = \mathbf{0}, \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}_N \Big\}.\end{aligned} \tag{6.140}$$

Note, the superscript of $\mathbf{V}_r^{\mathbf{K}}$ respectively $\mathbf{V}_{r+1}^{\mathbf{K}}$ shall indicate the eigenvalue decomposition $\mathbf{K} = (\mathbf{V}_r^{\mathbf{K}}, \mathbf{V}_{r+1}^{\mathbf{K}})\mathbf{\Lambda}(\mathbf{V}_r^{\mathbf{K}}, \mathbf{V}_{r+1}^{\mathbf{K}})^T$ for any $\mathbf{K} \in conv(\mathcal{K})$.

Insertion of $\boldsymbol{c}(\boldsymbol{\alpha}, \mu)$ in $G(\boldsymbol{\alpha}, \mu)$ (6.86) yields

$$G(\boldsymbol{\alpha}, \mu) = \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{Y}\mathbf{G}\mathbf{V}_r\overline{\mathbf{J}}(\mu)^{-1}\mathbf{V}_r^T\mathbf{G}^T\mathbf{Y}\boldsymbol{\alpha} - \boldsymbol{\alpha}^T\mathbf{1}_N \tag{6.141}$$

which is continuous and convex for $\mu \in (0,1)$ due to Lemma (6.3.2). The feasible set $\mathcal{S}_D(\overline{\mathbf{K}}(\mu))$ reduces independently of $\mu \in (0,1)$ to the set

$$\mathcal{S}(\mathbf{V}_{r+1}) := \left\{ \boldsymbol{\alpha} \in \mathbb{R}^N \middle| \mathbf{V}_{r+1}^T\mathbf{G}^T\mathbf{Y}\boldsymbol{\alpha} = \mathbf{0}, \boldsymbol{\alpha}^T\boldsymbol{y} = 0, \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}_N \right\}. \tag{6.142}$$

Therefore, and due to the linearity of the transformation, we can apply Theorem (6.2.1) by the use of the derivative

$$\partial_\mu G(\boldsymbol{\alpha}, \mu) = \frac{1}{2}\boldsymbol{c}(\boldsymbol{\alpha}, \mu)^T \left(\mathbf{K}_{(t)} - \mathbf{K}^*\right) \boldsymbol{c}(\boldsymbol{\alpha}, \mu) \tag{6.143}$$

obtained from Lemma (6.3.2).

In virtue of Theorem (6.2.1), for $\mu \in (0,1)$ the directional derivative of $g(\mu) = -q_r(\overline{\mathbf{J}}(\mu)) = \min_{\boldsymbol{\alpha}\in\mathcal{S}(\mathbf{V}_{r+1})} G(\boldsymbol{\alpha}, \mu)$ is given by

$$g'(\mu; 1) = \partial_\mu G(\boldsymbol{\alpha}_\mu^*, \mu) \tag{6.144}$$

with the global minimizer $\boldsymbol{\alpha}_\mu^* := \arg\min_{\boldsymbol{\alpha}\in\mathcal{S}(\mathbf{V}_{r+1})} G(\boldsymbol{\alpha}, \mu)$. And due to Corollary (6.2.1) the problem of finding the optimal convex combination, i.e. solving the problem $\min_{\mu\in(0,1)} g(\mu)$, has a global solution $\mu^* \in (0,1)$. In order to iterate a global minimizer μ^* we used the spectral projected gradient solver (Alg. (6.2.1)) together with the QP-solver (Alg. 6.1.1) to solve for $\boldsymbol{\alpha}_\mu^*$ in each iteration. The benefit from using our modified QP-solver is that the matrix $\overline{\mathbf{J}}(\mu)$ must not be inverted explicitly in each iteration. Instead $\boldsymbol{c}(\boldsymbol{\alpha}_\mu^*, \mu) = \mathbf{V}_r\boldsymbol{d}_\mu^*$ is directly returned by the QP-solver as an optimal feasible solution $(\boldsymbol{\alpha}_\mu^*, \boldsymbol{d}_\mu^*) \in \mathcal{S}_{D_r}(\overline{\mathbf{J}}(\mu))$.

Although the feasible set $\boldsymbol{S}(\mathbf{V}_{r+1})$ is independent of μ, it remains dependent on the matrix $\overline{\mathbf{K}}(\mu)$ via the transformation $\mathbf{V}_{r+1}$. In this case, it follows the analogon of inequality (6.116) can not be maintained anymore, because the left hand side inequality

$$q_r(\mathbf{J}) \geq Q_r(\boldsymbol{\alpha}_{(t)}^*, \boldsymbol{d}_r^*, \mathbf{J}) \tag{6.145}$$

might not hold for an optimal point $(\boldsymbol{\alpha}^*, \boldsymbol{d}_r^*) \in \mathcal{S}_{D_r}(\mathbf{J})$ with $Q_r(\boldsymbol{\alpha}_{(t)}^*, \boldsymbol{d}_r^*, \mathbf{J}) = q_r(\mathbf{J})$ and $\boldsymbol{d}_r^* = \mathbf{J}^{-1}\mathbf{V}_r^T\mathbf{G}^T\mathbf{Y}\boldsymbol{\alpha}^*$. The reason is that $\boldsymbol{\alpha}_{(t)}^* \in \mathcal{S}(\mathbf{V}_{r+1}^{\mathbf{K}_{(t)}})$ might not be in the feasible set $\mathcal{S}(\mathbf{V}_{r+1}^{\mathbf{K}})$ and hence $q_r(\mathbf{J})$ could be smaller than $Q_r(\boldsymbol{\alpha}_{(t)}^*, \boldsymbol{d}_r^*, \mathbf{J})$.

Therefore, and contrary to the regularized $\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz Classifier Algorithm, we must now solve in each iteration of the SA Algorithm (6.2.2) a QP-problem of the form

$$q_r(\mathbf{J}_{\mu_0}(\boldsymbol{x})) = \max_{(\boldsymbol{\alpha},\boldsymbol{d}_r)\in\mathcal{S}_{D_r}(\mathbf{J}_{\mu_0}(\boldsymbol{x}))} Q_r(\boldsymbol{\alpha},\boldsymbol{d}_r,\mathbf{J}_{\mu_0}(\boldsymbol{x})) \tag{6.146}$$

with

$$\mathbf{J}_{\mu_0}(\boldsymbol{x}) := (\mathbf{V}_r^{\mathbf{K}_{\mu_0}(\boldsymbol{x})})^T \mathbf{K}_{\mu_0}(\boldsymbol{x}) \mathbf{V}_r^{\mathbf{K}_{\mu_0}(\boldsymbol{x})} \tag{6.147}$$

and

$$\mathbf{K}_{\mu_0}(\boldsymbol{x}) := (1-\mu_0)\cdot\mathbf{K}_{(t)} + \mu_0\cdot\mathbf{K}(\boldsymbol{x}) \tag{6.148}$$

until the inequality

$$q_r(\mathbf{J}_{\mu_0}(\boldsymbol{x}^*)) > Q_r(\boldsymbol{\alpha}^*_{(t)},\boldsymbol{d}^*_{r(t)},\mathbf{J}_{(t)}) \tag{6.149}$$

is satisfied for a matrix $\mathbf{J}^* := \mathbf{J}_{\mu_0}(\boldsymbol{x}^*)$, $\boldsymbol{x}^* \in \mathcal{X}$. For this purpose, we have to define $h_{\mathbf{K}}(\boldsymbol{x}) := -q_r(\mathbf{J}_{\mu_0}(\boldsymbol{x}))$ and $h_{\max} := -Q_r(\boldsymbol{\alpha}^*_{(t)},\boldsymbol{d}^*_{r(t)},\mathbf{J}_{(t)})$ in the requirements of the SA algorithm (6.2.2). The weight $\mu_0 \in (0,1)$ is arbitrary chosen and fixed in advance.

To solve $h_p = -q_r(\mathbf{J}_{\mu_0}(\boldsymbol{x}_p))$ for any trial step $\boldsymbol{x}_p \in \mathcal{X}$ of the random walker in Line (11) of the SA Algorithm (6.2.2), we can use our proposed QP-Solver (Alg. 6.1.1). Additionally, the associated eigenvalue decompositions have to be computed.

In analogy to Algorithm (6.3.1), the fully implementable non-regularized $\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz classifier algorithm is summarized in Algorithm (6.3.2) [Stuhlsatz 2008c]. A few special notes on an efficient implementation that go beyond the present thesis can be found in the Addendum (B.3.1) to this section.

Algorithm 6.3.2 Non-regularized $\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz Classifier Algorithm

Require: Training set $\emptyset \neq \mathcal{O}_N \subset \mathcal{Z} = \mathcal{X}\times\mathcal{Y}$, $\mathcal{X}\subset\mathbb{R}^m$ compact and convex, $1 \leq n \leq M$ basis functions $\Phi_n(\boldsymbol{x}) \in \mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$, trade-off parameter $C>0$, $\mathbf{K}_{(0)} \in conv(\mathcal{K})$, $T\in\mathbb{N}$

Ensure: $\mathcal{K} := \{0 \preceq \mathbf{K}(\boldsymbol{x}) \in \mathbb{R}^{M\times M} \mid \boldsymbol{x}\in\mathcal{X}\}$, $\exists(\boldsymbol{x}_j,y_j),(\boldsymbol{x}_i,y_i)\in\mathcal{O}_N \;:\; y_j=-1 \wedge y_i = 1$

1: Set $t=0$.
2: Compute $(\mathbf{V}_{r(0)},\mathbf{V}_{r+1(0)}) \longleftarrow EV(\mathbf{K}_{(0)})$. {Eigenvalue Decomposition}.
3: Compute $\mathbf{J}_{(0)} := \mathbf{V}^T_{r(0)}\mathbf{K}_{(0)}\mathbf{V}_{r(0)}$.
4: **loop**
5: Solve for

$$(\boldsymbol{\alpha}^*_{(t)},\boldsymbol{d}^*_{r(t)}) \longleftarrow \operatorname*{arg\,min}_{(\boldsymbol{\alpha},\boldsymbol{d}_r)\in\mathcal{S}_{D_r}(\mathbf{J}_{(t)})} -Q_r(\boldsymbol{\alpha},\boldsymbol{d}_r,\mathbf{J}_{(t)})$$

with feasible set

$$\mathcal{S}_{D_r}(\mathbf{J}_{(t)}) := \left\{ (\boldsymbol{\alpha}, \boldsymbol{d}_r) \in \mathbb{R}^N \times \mathbb{R}^r \;\middle|\; \mathbf{J}_{(t)}\boldsymbol{d}_r = \mathbf{V}_{r(t)}^T \mathbf{G}^T \mathbf{Y} \boldsymbol{\alpha}, \boldsymbol{\alpha}^T \boldsymbol{y} = 0, \right.$$
$$\left. \mathbf{V}_{r+1(t)}^T \mathbf{G}^T \mathbf{Y} \boldsymbol{\alpha} = \mathbf{0}, \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}_N \right\}.$$

using the QP-Solver Algorithm (6.1.1) and the Initializer (6.1.2).

6: **if** (t=T) **then**

7: **return**

$$\boldsymbol{c}^* \longleftarrow \mathbf{V}_{r(t)}\boldsymbol{d}^*_{r(t)} + \mathbf{V}_{r+1(t)}\boldsymbol{d}^*_{r+1(t)},\ \boldsymbol{d}^*_{r+1(t)} \in \mathbb{R}^{M-r} \text{ arbitrary}$$
$$b^* \longleftarrow -\frac{1}{2|\mathcal{A}_+|}\left(\sum_{i\in\mathcal{A}_+}\sum_{n=1}^{M} c_n^* \Phi_n(\boldsymbol{x}_i) + \sum_{j\in\mathcal{A}_-}\sum_{n=1}^{M} c_n^* \Phi_n(\boldsymbol{x}_j) \right)$$

with $\mathcal{I} := \{i \in \mathbb{N} : 0 < \alpha_i < C\}$, $\mathcal{A}_+ \subseteq \{i \in \mathcal{I} : y_i = 1\}$ and $\mathcal{A}_- \subseteq \{j \in \mathcal{I} : y_j = -1\}$ such that $|\mathcal{A}_+| = |\mathcal{A}_-|$.

8: **end if**

9: Search for a candidate matrix $\mathbf{K}^* := \mathbf{K}(\boldsymbol{x}^*) \in conv(\mathcal{K})$, $\boldsymbol{x}^* \in \mathcal{X}$, satisfying

$$h_{\mathbf{K}}(\boldsymbol{x}^*) = -q_r(\mathbf{J}_{\mu_0}(\boldsymbol{x}^*)) < h_{max} := -Q_r(\boldsymbol{\alpha}^*_{(t)}, \boldsymbol{d}^*_{r(t)}, \mathbf{J}_{(t)})$$

for $\mu_0 \in (0,1)$ using the SA-Algorithm (6.2.2) with

$$h_{\mathbf{K}}(\boldsymbol{x}) := -q_r(\mathbf{J}_{\mu_0}(\boldsymbol{x})).$$

The value of $q_r(\mathbf{J}_{\mu_0}(\boldsymbol{x}))$ is computed using QP-Solver Algorithm (6.1.1), the Initializer (6.1.2) and an eigenvalue decomposition

$$(\mathbf{V}_r^*, \mathbf{V}_{r+1}^*) \longleftarrow EV(\mathbf{K}_{\mu_0}(\boldsymbol{x})) \tag{6.150}$$

to compute

$$\mathbf{J}_{\mu_0}(\boldsymbol{x}) := (\mathbf{V}_r^*)^T \mathbf{K}_{\mu_0}(\boldsymbol{x}) \mathbf{V}_r^*.$$

10: Solve for

$$\mu^* \longleftarrow \underset{\mu\in(0,1)}{\arg\min}\ g(\mu)$$

using the Gradient Solver Algorithm (6.2.1) with

$$g(\mu) := -Q(\boldsymbol{\alpha}^*_\mu, \boldsymbol{d}^*_\mu, \overline{\mathbf{J}}(\mu))$$

and the derivative

$$\partial_\mu G(\boldsymbol{\alpha}^*_\mu, \mu) := \frac{1}{2}\boldsymbol{c}(\boldsymbol{\alpha}^*_\mu, \mu)^T \left(\mathbf{K}_{(t)} - \mathbf{K}^*\right) \boldsymbol{c}(\boldsymbol{\alpha}^*_\mu, \mu)$$

whereas $\boldsymbol{c}(\boldsymbol{\alpha}^*_\mu, \mu) := \mathbf{V}^*_r \boldsymbol{d}^*_\mu$ is computed for each $\mu \in (0,1)$ by

$$(\boldsymbol{\alpha}^*_\mu, \boldsymbol{d}^*_\mu) \longleftarrow \underset{(\boldsymbol{\alpha}, \boldsymbol{d}_r) \in \mathcal{S}_{D_r}(\overline{\mathbf{J}}(\mu))}{\arg\min} -Q(\boldsymbol{\alpha}, \boldsymbol{d}_r, \overline{\mathbf{J}}(\mu))$$

using the QP-Solver Algorithm (6.1.1) and the Initializer (6.1.2).

11: Set $(\mathbf{V}_{r(t+1)}, \mathbf{V}_{r+1(t+1)}) = (\mathbf{V}^*_r, \mathbf{V}^*_{r+1})$.
12: Set $\mathbf{K}_{(t+1)} = \overline{\mathbf{K}}(\mu^*)$.
13: Set $\mathbf{J}_{(t+1)} = \overline{\mathbf{J}}(\mu^*)$.
14: $t \longleftarrow t + 1$
15: **end loop**

Summary

In this chapter, we developed in a *divide-and-conquer* manner the components necessary for implementing the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm. Namely, these components are an adapted interior point QP-solver (Alg. (6.1.1)), a spectral projected gradient solver (Alg. (6.2.1)) and a stochastic search heuristics (Alg. (6.2.2)) based on simulated annealing. In Figure (6.4) a global view of the framework is sketched in which the QP-solver takes a central role.

We showed if providing an appropriate parameterization of all solutions of the involved linear equation system of the feasible set, one obtains different realizations of the dual $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm. The first realization (Alg. (6.3.1)) we introduced is based on a perturbation of the matrix set $\mathcal{K}$ [Stuhlsatz 2007c]. It turned out, that a perturbation yields desirable properties of the algorithm with respect to an efficient implementation. On the other hand, a perturbation $\epsilon > 0$ of the set $\mathcal{K}$ results in a change of the original problem in particular if ϵ is chosen to be large. Because the ϵ-parameter depends on the data space and the employed basis functions, it may happen, if $\epsilon > 0$ is very small, that the QP-solver or the involved matrix inversion run in numerical problems for ill-conditioned situations. Therefore, we developed an alternative based on an algebraic parameterization avoiding any kind of perturbations [Stuhlsatz 2008c]. Because there is *no free lunch* [Wolpert 1997], the resulting non-regularized $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier Algorithm (6.3.2) is computationally more demanding than its regularized counterpart. Fortunately, due to the component-wise architecture of the developed framework,

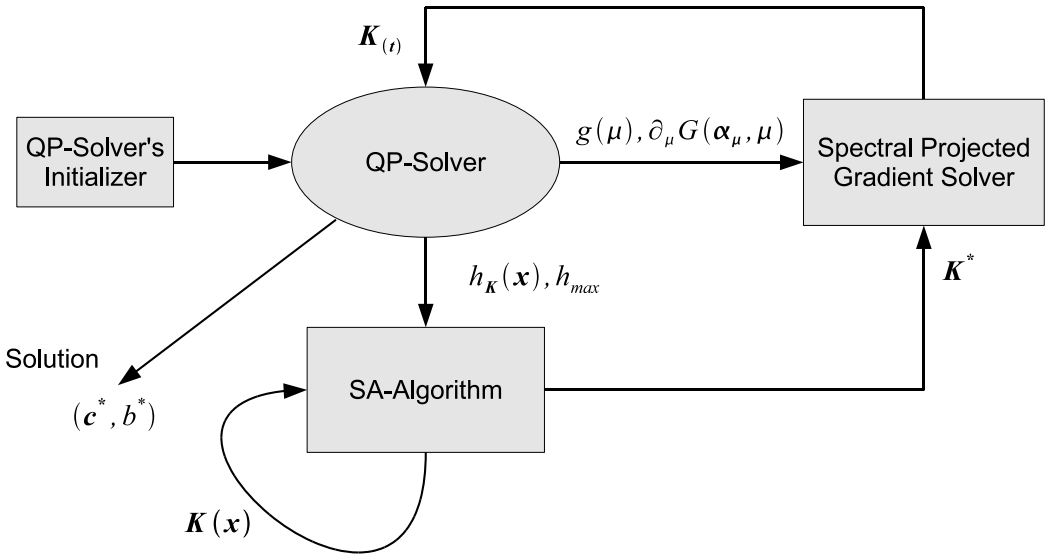


Figure 6.4: A global view of the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier framework with its specific components and dependencies.

it is possible to optimize all components with respect to an optimal efficiency independently of each other in future. However, for this thesis we coded both realizations in software without any additional efficiency optimizations.

In the next chapter, results will be presented using our implementations experimentally tested on different data sets. These experiments shall explore the performance of the new learning algorithms compared with the SVM in standard situations on the one hand, and on the other hand they shall prove the benefit obtained from the new learning algorithms in cases the SVM fails completely.

Chapter 7

Lipschitz Classifier - Experimental Testing

Contents

We have presented new implementable maximum margin algorithms. To prove their performance, this chapter presents experimental results. For this purpose, we coded both realizations of the dual $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm introduced in Chapter (6) in the MATLAB® programming language[1]. As classification tasks we chose three different datasets. The first is a 2-dimensional artificial dataset consisting of 10 subsets simulated from a mixture of Gaussian distributions. This artificial dataset serves as a measurement of robustness of the learning algorithms over varying data. Additionally, due to the 2-dimensional nature, it facilitates a visualization of the data and the learnt decision boundaries. As real world scenarios, we selected two datasets from the UCI Machine Learning Repository [Asuncion 2007] with different number of training and test samples and different number of dimensions of the data points.

We start in Section (7.1) with experiments, that shall evaluate the performance of the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier learnt by our algorithms in comparison to the SVM classifier when employing a kernel function as basis function, namely the RBF-kernel. Using the same decision function, we want to find out if both approaches have the ability to reach comparable performance in standard situations.

[1] The $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier's MATLAB-code can be sourced from the author for academic use only.

In Section (7.2), we present results obtained from an experimental setup on real world data using the hyperbolic tangent function as basis function to build up the decision function. The hyperbolic tangent function is well-known to violate the Mercer's condition for most of its parameter space. Thus, this experiment serves as an archetype for scenarios in which decision functions, which are being build from basis functions that do not satisfy Mercer's condition, and hence are not suitable for a use in SVMs, are a reasonable choice for a specific application. This experiment is impressive, because it shows exemplarily that our new learning algorithms work with basis functions that go beyond an application of standard kernel functions as shown in Section (7.1). This opens the way to a practical design of new classifiers that are able to facilitate a specific application in a robust maximum margin concept.

7.1 Performance Test using Mercer Kernel Functions

In this section, we evaluate the classification accuracy of the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier learnt using Algorithm (6.3.1) and Algorithm (6.3.2) presented in Chapter (6).

For this purpose, we employ the RBF-kernel function (3.77) as basis function (cf. Def (5.2.1)), i.e.

$$\Phi_n^{rbf}(\boldsymbol{x}) := \exp\left(-\gamma \|\boldsymbol{x}_n - \boldsymbol{x}\|_2^2\right), \boldsymbol{x}_n \in \mathcal{O}_N \subset \mathcal{X}, \tag{7.1}$$

which satisfies the Mercer's condition for all $\gamma > 0$ and is therefore suitable for use in SVMs. In particular, the RBF-kernel function is the most popular basis function for SVMs and has shown good classification accuracies on many different datasets.

For our experiments we consider three different datasets. The first one is a 2-dimensional artificial dataset consisting of 10 subsets for training and testing which were generated from a random mixture of Gaussian distributions (Fig. (7.1)). Due to the multimodality of the data distribution, there is a large overlap of the two classes and the subsets differ strongly rendering the classification problem not easy to solve. On the one hand, the different subsets let us analyze the performance over varying datasets, and on the other hand, the 2-dimensional nature let us visualize the classifications.

The other real world datasets, namely the UCI heart dataset and the UCI breast-cancer dataset[2], are taken from the UCI Machine Learning Reposi-

[2]This breast cancer domain was obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia. Thanks go to M. Zwitter and M. Soklic for providing the data.

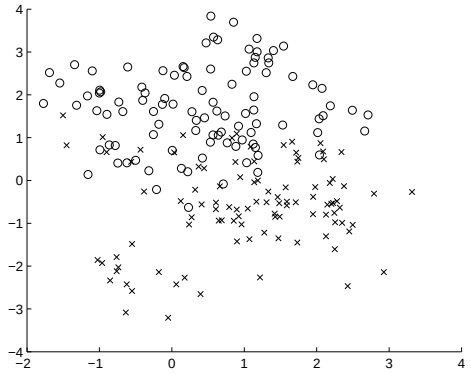


Figure 7.1: A scatter plot of a training set of the 2d-artificial dataset.

feature	2d-artificial	UCI heart	UCI breast-cancer
#dimensions	2	13	9
#data total	6100	270	257
#data test	10x 400	70	87
#data training	10x 200	200	170
val. splits:			
#val. training	5x 200	5x 150	5x 130
#val. eval	1x 100	5x 50	5x 40

Table 7.1: Summary of the datasets used for the comparison experiments.

tory [Asuncion 2007]. These sets are real-world datasets of different size and dimension. A summary of the used datasets is given in Table (7.1).

Note, the 2d-artificial dataset contains 10 test sets of 400 points, 10 training sets of 200 points and an extra evaluation set of 100 points. All these sets are disjoint. We used the first 5 training sets and the evaluation set to tune the trade-off parameter C of the algorithms and the parameter γ of the RBF-kernel. The UCI dataset contains a test set of 70 points and a training set of 200 points, and the UCI breast-cancer dataset consists of a test set of 87 points and a training set of 170 points. In both cases, the test set and the training set are disjoint too. We randomly partitioned the training set of the UCI datasets in 5 subsets for parameter tuning, because in contrast to the 2d-artificial dataset only a very limited amount of data is available. The subsets are then used for selecting appropriate values of $\gamma > 0$ and $C > 0$ via 5-fold cross-validation. The evaluation procedure for parameter tuning is described in more detail in the next section.

parameter	2d-artificial	UCI heart	UCI breast-cancer
k_{max}	5,000	5,000	5,000
i_{max}	100	100	100
T_0	1,000	1,000	$1 \cdot 10^{-5}$
σ^2	$2 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	$3.5 \cdot 10^{-6}$
$\mathcal{X}$	$[-10, 10]^2$	$[-100, 100]^{13}$	$[-10, 10]^9$
ϵ	$1 \cdot 10^{-6}$	$1 \cdot 10^{-6}$	$5 \cdot 10^{-4}$

Table 7.2: Algorithm specific settings fixed for evaluation and testing of the non-/regularized $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz Classifier.

7.1.1 Evaluation of Algorithmic Parameters

Before the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier and the SVM both using the RBF-kernel can be tested, algorithmic parameters as in particular the parameters $(\gamma, C) \in \mathbb{R}^+ \times \mathbb{R}^+$ have to be tuned for each dataset. In Table (7.2), some important algorithmic settings are listed which have to be carefully selected and fixed for all evaluation and test procedures in advance. Namely, for the stochastic search (Alg. 6.2.2), these parameters are the maximum number of SA loops k_{max}, the maximum number of MH loops i_{max}, the start temperature T_0, the variance σ^2 of the random walker and the search region $\mathcal{X}$. Table (7.2) also shows the distortion factor ϵ used for the regularized $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz Classifier. Note, that for all three datasets the regularization parameter ϵ is fixed to very small values. In all cases, the start configuration $\boldsymbol{x}_0 \in \mathcal{X}$ is set to the mean vector of the training data. We determined all settings summarized in Table (7.2) by trial runs on the training data such that acceptance rates as described in the end of Section (6.2.3) are maintained. This part of parameter selection is performed manually without the use of any measure of quality. The space $\mathcal{X}$ is chosen such that it covers a large data space including all available training data.

In order to determine $(\gamma, C) \in \mathbb{R}^+ \times \mathbb{R}^+$ in case of the 2d-artificial dataset, we used the first 5 realizations of the training data and an extra validation set. The evaluation procedure was as follows: We trained independently a RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier as well as a RBF-SVM classifier on each of the first 5 realization with a fixed parameter configuration. Then, we evaluated each classifier by computing the median of the classification error rates obtained from classifying the validation data (*validation errors*). For parameter selection, we repeated the evaluation procedure for each parameter (γ, C) of a discretized and predefined parameter range.

In Table (7.3) our selection for the 2d-artificial dataset is summarized including the median validation errors and standard deviations. In Figures (7.2)-

	reg. Lip.	**non-reg. Lip.**	**SVM**
dataset	2d-art.	2d-art.	2d-art.
basis function	RBF-kernel	RBF-kernel	RBF-kernel
parameter (γ, C)	$(0.02, 0.01)$	$(0.006, 8)$	$(0.0714, 150)$
median val. err.	$14.00 \pm 0.00\%$	$15.00 \pm 1.14\%$	$14.00 \pm 0.55\%$

Table 7.3: Selected parameters (γ, C) for the non-/regularized $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier and the SVM classifier due to the median validation errors using the 2d-artificial dataset and the RBF-kernel.

	reg. Lip.	**non-reg. Lip.**	**SVM**
dataset	heart	heart	heart
basis function	RBF-kernel	RBF-kernel	RBF-kernel
parameter (γ, C)	$(0.0001, 0.25)$	$(5 \cdot 10^{-5}, 200)$	$(0.000256, 100)$
median val. err.	$14.00 \pm 2.61\%$	$14.00 \pm 2.52\%$	$14.00 \pm 3.74\%$

Table 7.4: Selected parameters (γ, C) for the non-/regularized $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier and the SVM classifier due to the median cross-validation errors using the UCI heart dataset and the RBF-kernel.

	reg. Lip.	**non-reg. Lip.**	**SVM**
dataset	breast-cancer	breast-cancer	breast-cancer
basis function	RBF-kernel	RBF-kernel	RBF-kernel
parameter (γ, C)	$(0.175, 2.5 \cdot 10^{-5})$	$(0.001, 0.01)$	$(0.0556, 1.5)$
median val. err.	$22.50 \pm 3.95\%$	$27.50 \pm 8.51\%$	$20.00 \pm 8.73\%$

Table 7.5: Selected parameters (γ, C) for the non-/regularized $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier and the SVM classifier due to the median cross-validation errors using the UCI breast-cancer dataset and the RBF-kernel.

(7.5), respectively Figure (7.6) and Figure (7.7), the medians of the validation errors obtained by applying the regularized, respectively non-regularized, RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier to the 2d-artificial dataset are depicted. While Figure (7.2) and Figure (7.6) show the full explored parameter range, Figures (7.3)-(7.5) and Figure (7.7) present a finer parameter grid of the area marked by the dashed boxes around the best configuration of the coarser grid. Likewise, in Figure (7.8) and Figure (7.9) the results of the parameter evaluation in case of the RBF-SVM classifier are shown. Note, each grid point represents an inspected parameter configuration. Thus, the contours visualize roughly the areas of equal median validation errors. Small white squares represent parameter settings resulting in a minimum median validation error.

In case of more than one minimum median validation error, black triangles mark the parameter settings which have the minimum standard deviation. We always chose a parameter setting that has resulted in a minimum median validation error and minimum standard deviation.

In case of the UCI datasets, we performed a slightly different evaluation procedure. Due to the few available training data, we randomly partitioned the training set in 5 cross-validation sets each comprising a training set and a disjoint validation set (see Tab. (7.1)). Then, to evaluate good parameter settings for (γ, C), we trained independently a non-/regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier and a RBF-SVM classifier on the training set of each cross-validation set and evaluated the accuracy on the associated validation set (*5-fold cross-validation*). Thus, for a particular parameter configuration we obtained 5 results for each classifier. The median of these 5 results for each parameter configuration of the considered parameter range is used for a parameter selection. Table (7.4) summarizes the selected parameters for testing the classifiers on the UCI heart dataset. Figure (7.10) and (7.11), respectively Figure (7.12) and (7.13), show the scanned parameter range and associated medians of the 5-fold cross-validation errors using the regularized, respectively non-regularized, RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier on the UCI heart dataset. Likewise, Figure (7.14) depicts the situation using the RBF-SVM classifier.

We also performed a 5-fold cross-validation on a random partition of the training set of the UCI breast-cancer dataset. Table (7.5) summarizes the selected parameters for testing the classifiers on the UCI breast-cancer dataset. Figure (7.15) and (7.17), respectively Figure (7.18) and (7.19), show the scanned parameter range and associated medians of the 5-fold cross-validation errors using the regularized, respectively non-regularized, RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier. Likewise, Figure (7.20) and Figure (7.21) depict the situation using the RBF-SVM classifier.

It is important to note, that although completely different parameter settings were evaluated for the regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier and the RBF-SVM classifier, the median validation errors are equal (except on UCI breast-cancer). Moreover, the standard deviations of the regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier over all validation runs are always less than the standard deviations of the RBF-SVM classifier. This indicates that decision functions found by the regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier are more stable with respect to the classification performance over varying data sets. The non-regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier performs slightly worse than its regularized counterpart, but also the standard deviations are less than the standard deviations of the RBF-SVM (except on 2d-artificial). On the other hand, the evaluated trade-off parameter C is much

larger than the setting for the regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier. This indicates, that the decision functions found by the non-regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier are able to separate the data with fewer margin errors. In the next section we will use the evaluated parameters to get classification performances with respect to the test sets.

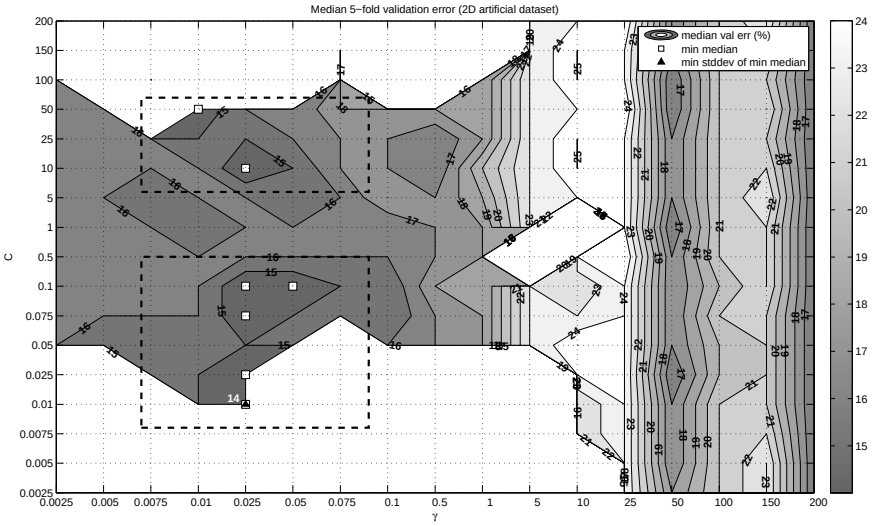


Figure 7.2: **Median validation errors** of the **regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier** on the **2d-artificial dataset** for different parameter configurations (γ, C).

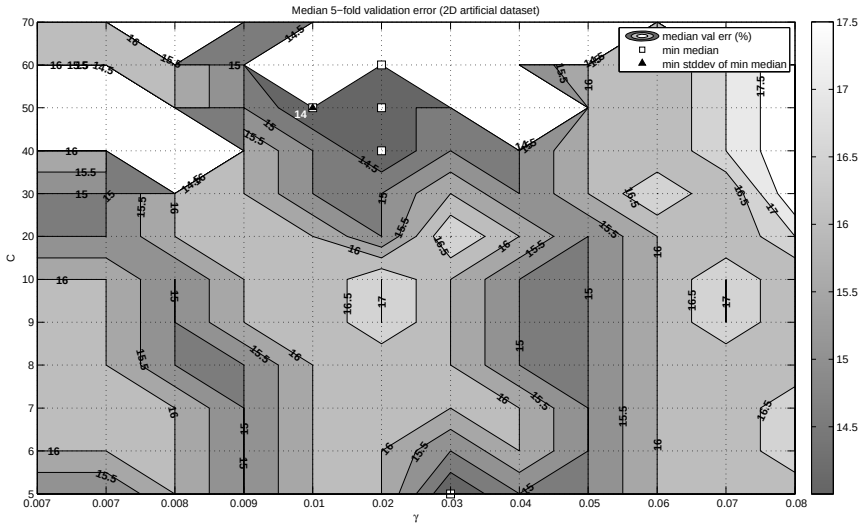


Figure 7.3: **Median validation errors** of the **regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier** on the **2d-artificial dataset** in an enlarged parameter region surrounded by the **upper dashed box in Figure (7.2)**.

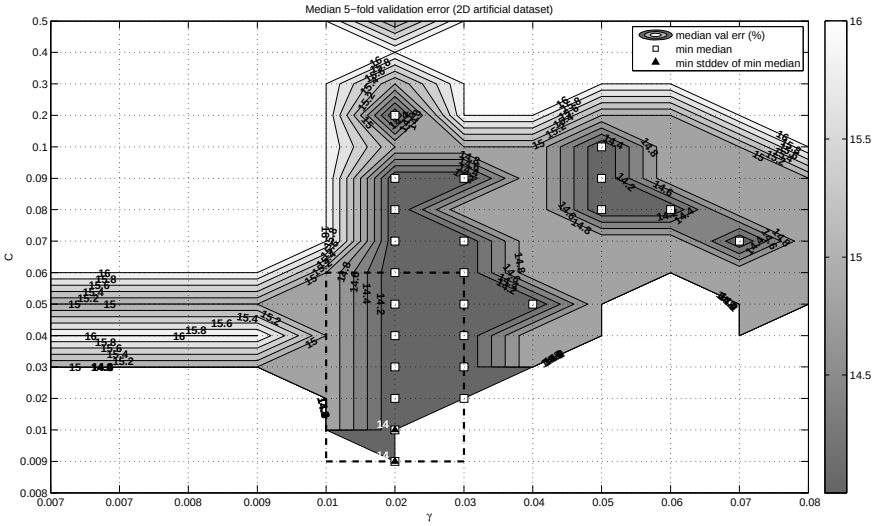


Figure 7.4: **Median validation errors** of the **regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier** on the **2d-artificial dataset** in an enlarged parameter region surrounded by the **lower dashed box in Figure (7.2)**.

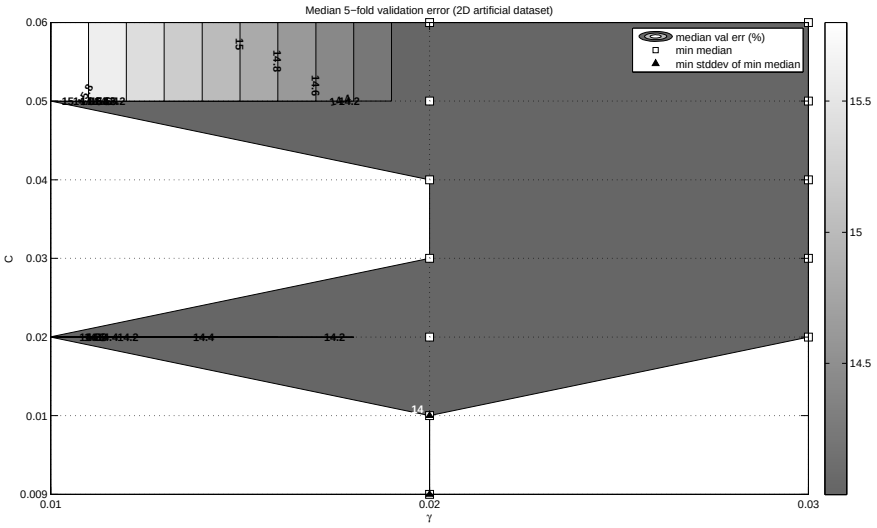


Figure 7.5: **Median validation errors** of the **regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier** on the **2d-artificial dataset** in an enlarged parameter region surrounded by the **dashed box in Figure (7.4)**.

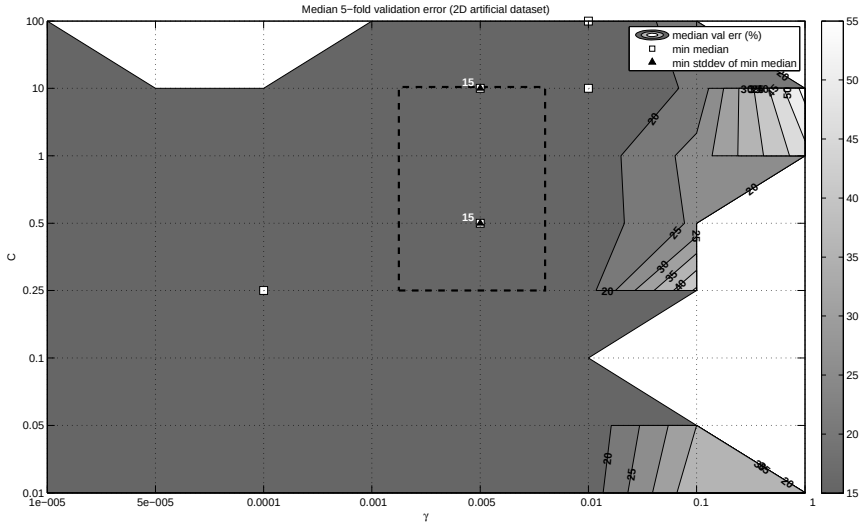


Figure 7.6: **Median validation errors** of the **non-regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier** on the **2d-artificial dataset** for different parameter configurations (γ, C).

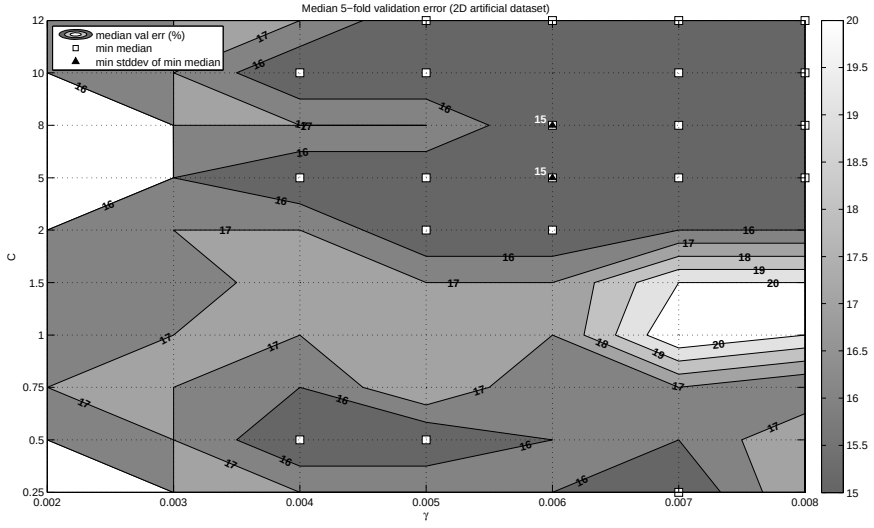


Figure 7.7: **Median validation errors** of the **non-regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier** on the **2d-artificial dataset** in an enlarged parameter region surrounded by the **dashed box in Figure (7.6)**.

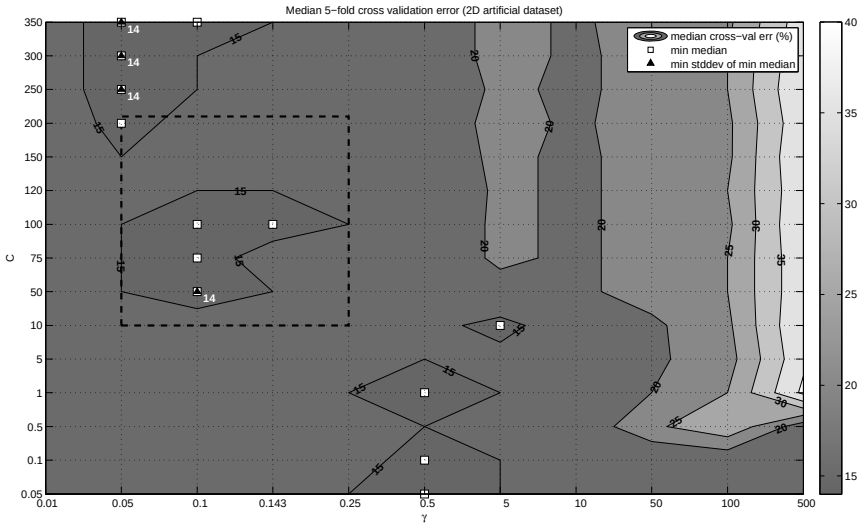


Figure 7.8: **Median validation errors** of the **RBF-SVM classifier** on the **2d-artificial dataset** for different parameter configurations (γ, C).

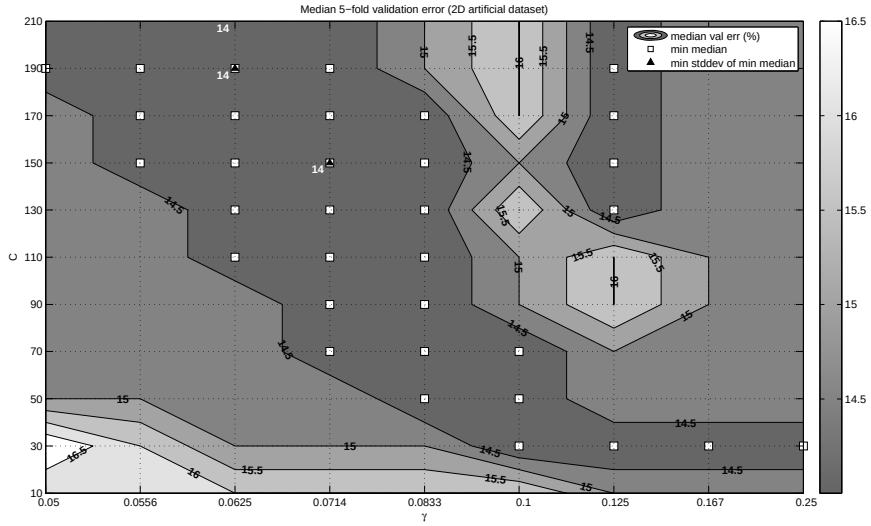


Figure 7.9: **Median validation errors** of the **RBF-SVM classifier** on the **2d-artificial dataset** in an enlarged parameter region surrounded by the **dashed box in Figure (7.8)**.

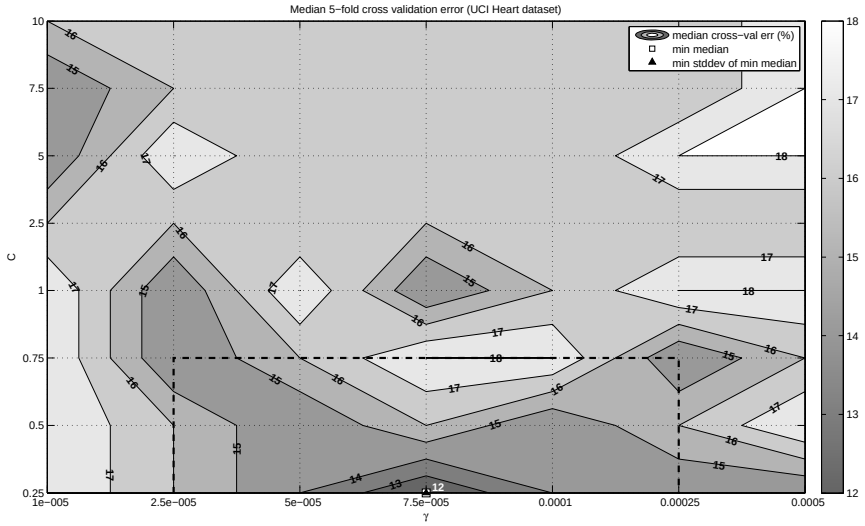


Figure 7.10: **Median 5-fold cross-validation errors** of the **regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier** on the **UCI heart dataset** for different parameter configurations (γ, C).

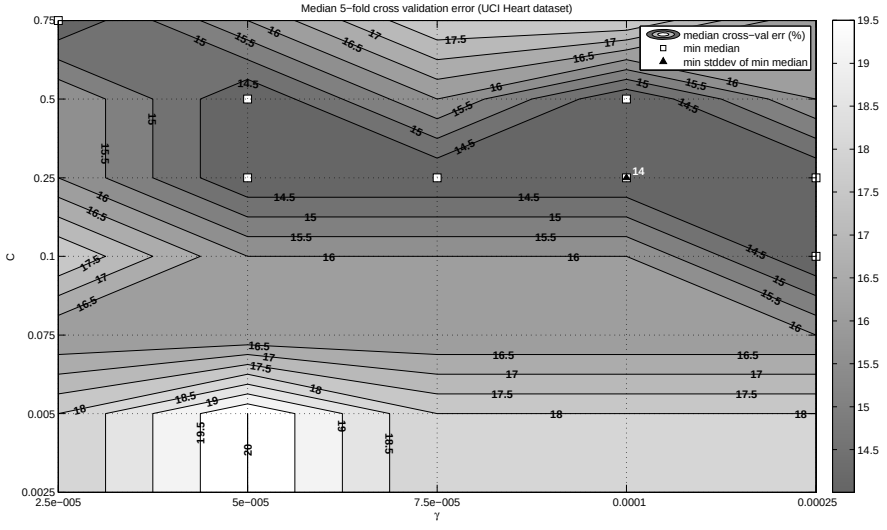


Figure 7.11: **Median 5-fold cross-validation errors** of the **regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier** on the **UCI heart dataset** for an enlarged parameter region surrounded by the **dashed box in Figure (7.10)**.

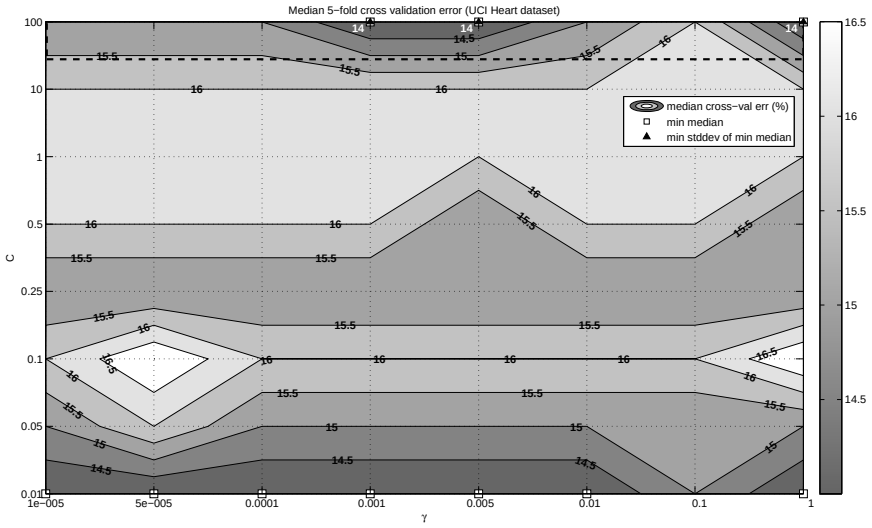


Figure 7.12: **Median 5-fold cross-validation errors** of the **non-regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier** on the **UCI heart dataset** for different parameter configurations (γ, C).

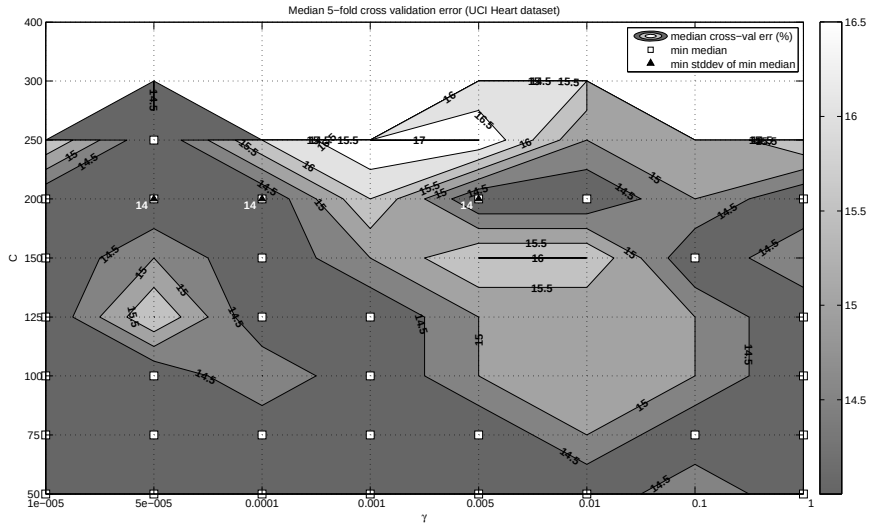


Figure 7.13: **Median 5-fold cross-validation errors** of the **non-regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier** on the **UCI heart dataset** for an enlarged parameter region surrounded by the **dashed box in Figure (7.12)**.

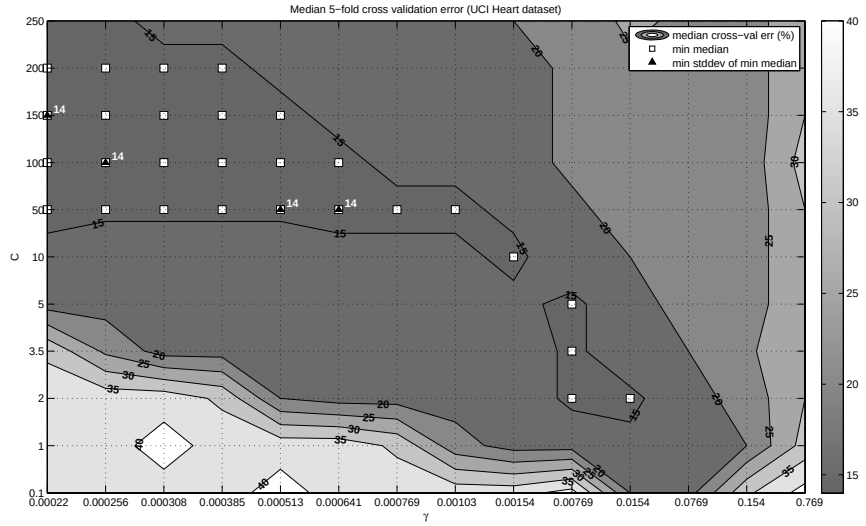


Figure 7.14: **Median 5-fold cross-validation errors** of the **RBF-SVM classifier** on the **UCI heart dataset** for different parameter configurations (γ, C).

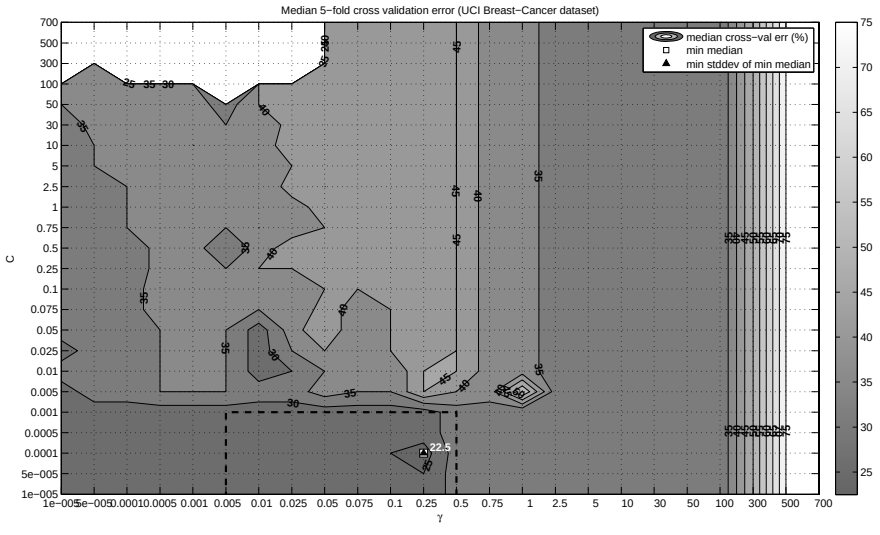


Figure 7.15: **Median 5-fold cross-validation errors** of the **regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier** on the **UCI breast-cancer dataset** for different parameter configurations (γ, C).

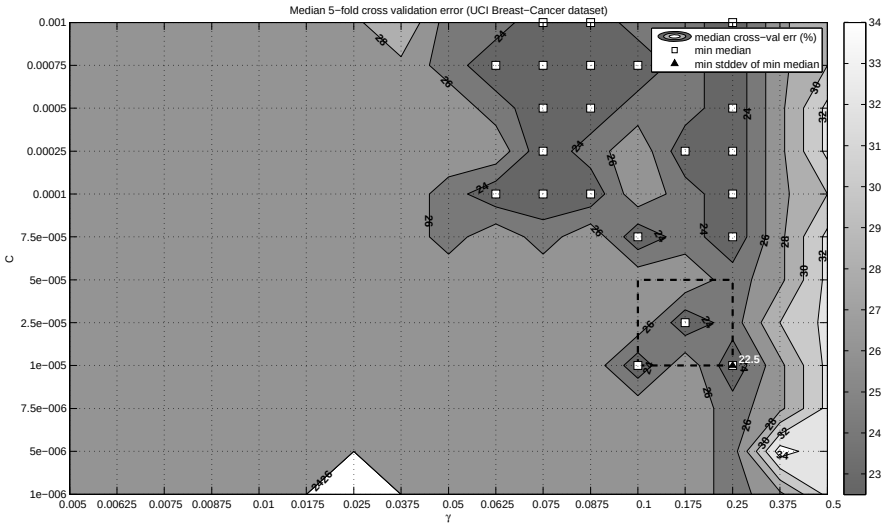


Figure 7.16: **Median 5-fold cross-validation errors** of the **regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz classifier** on the **UCI breast-cancer dataset** for an enlarged parameter region surrounded by the **dashed box in Figure (7.15)**.

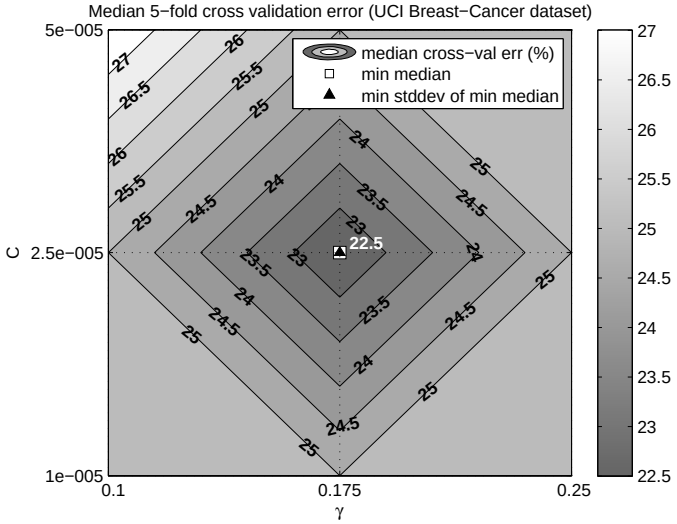


Figure 7.17: **Median 5-fold cross-validation errors** of the **regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz classifier** on the **UCI breast-cancer dataset** for an enlarged parameter region surrounded by the **dashed box in Figure (7.16)**.

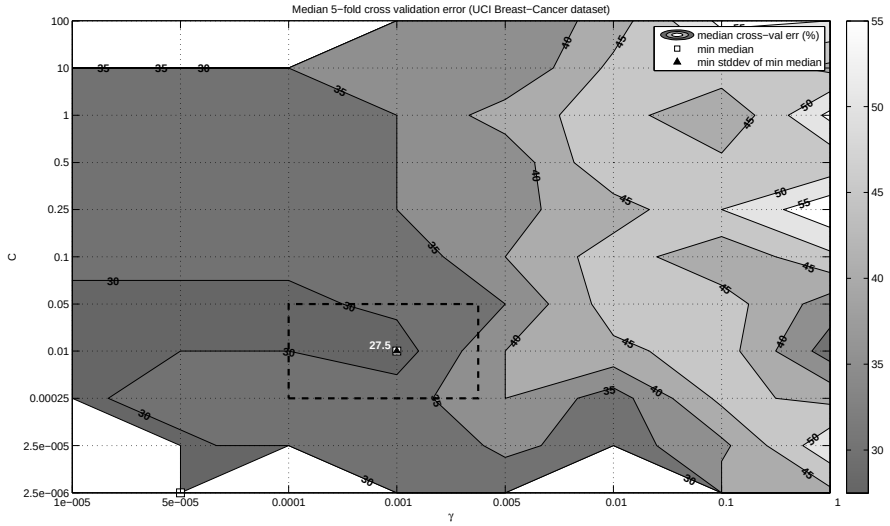


Figure 7.18: **Median 5-fold cross-validation errors** of the **non-regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier** on the **UCI breast-cancer dataset** for different parameter configurations (γ, C).

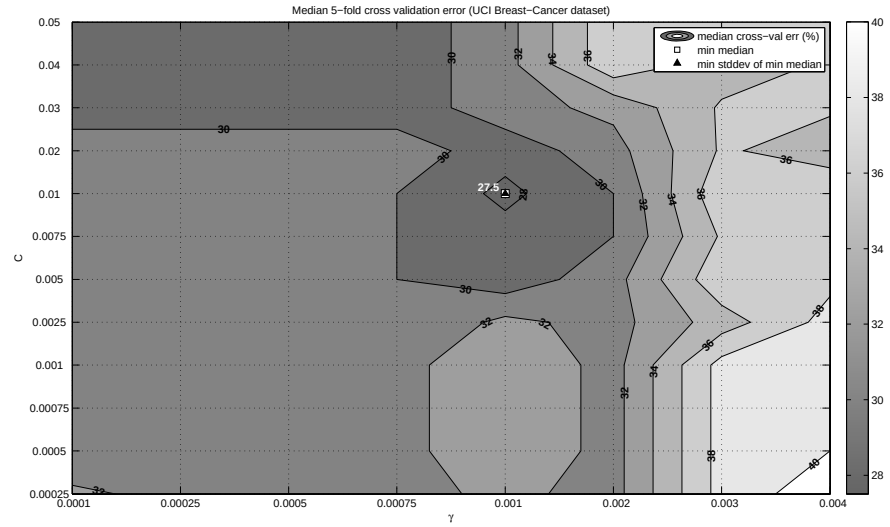


Figure 7.19: **Median 5-fold cross-validation errors** of the **non-regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier** on the **UCI breast-cancer dataset** for an enlarged parameter region surrounded by the **dashed box in Figure (7.18)**.

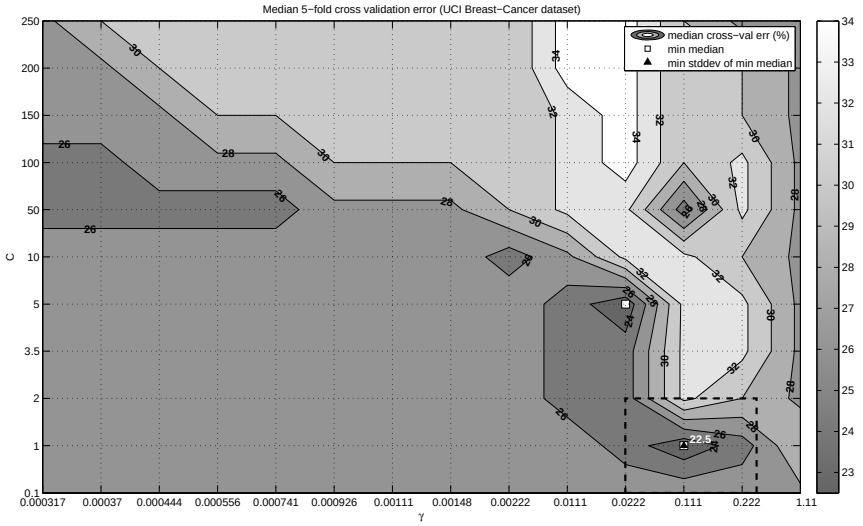


Figure 7.20: **Median 5-fold cross-validation errors** of the **RBF-SVM classifier** on the **UCI breast-cancer dataset** for different parameter configurations (γ, C).

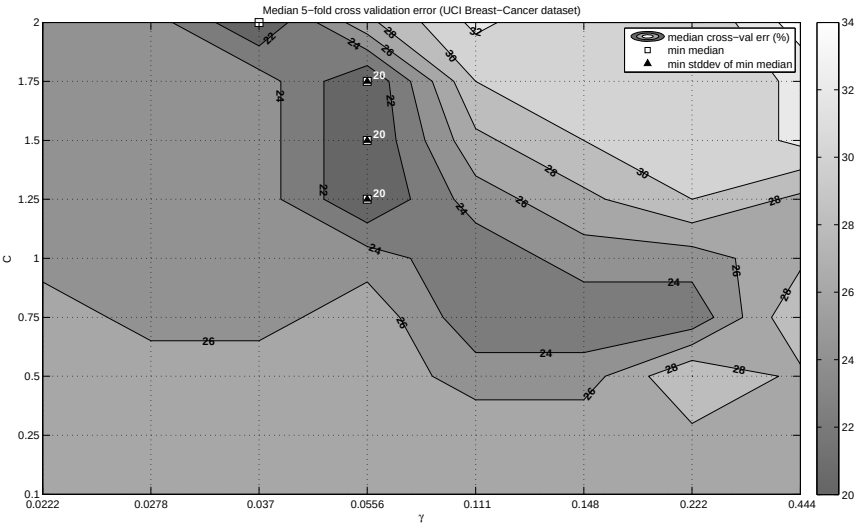


Figure 7.21: **Median 5-fold cross-validation errors** of the **RBF-SVM classifier** on the **UCI breast-cancer dataset** for an enlarged parameter region surrounded by the **dashed box in Figure (7.20)**.

7.1.2 Testing the RBF-$\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz Classifier

In the preceding section, we discussed the tuning of algorithmic parameters, in particular the evaluation of the parameters $(\gamma, C) \in \mathbb{R}^+ \times \mathbb{R}^+$. Additionally, results obtained from the evaluation are presented. In this section, we present the test results for the different datasets using these parameters. For this purpose, we fixed the evaluated parameters to the tuned values in advance and trained the RBF-$\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz classifier and the RBF-SVM classifier on the 2d-artificial, UCI heart and UCI breast-cancer datasets. Then, the trained classifiers were applied to the corresponding test sets for yielding test error rates for the so far unseen data. That means, the test data was separated from evaluation and training data and was only used during the test phase of the classifiers. In particular, in case of the 2d-artificial data, we had 10 training and 10 test sets at hand. Thus, we trained independently 10 classifiers for the RBF-$\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz classifiers as well as the RBF-SVM classifier using the 10 training sets. After training, we computed the test error rates with respect to the associated disjoint 10 test sets. Table (7.6) summarizes the test results.

	RBF-$\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lip. class.		
	reg.	**non-reg.**	**RBF-SVM**
2d-artificial	**12.25±2.28%**	$14.25 \pm 3.26\%$	$12.28 \pm 2.86\%$
heart	18.57% (13/70)	18.57% (13/70)	18.57% (13/70)
breast-cancer	28.74% (25/87)	32.18% (28/87)	28.74% (25/87)

Table 7.6: Test error rates obtained using the RBF-$\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz classifier in comparison to the RBF-SVM classifier for different test datasets.

In Table (7.6), the error rates on the 2d-artificial dataset are given as averaged test error rates and standard deviations with respect to the 10 different test sets. For the UCI datasets we have only one training and test set, hence standard deviations are not presented in Table (7.6). However, the fraction of misclassified test examples to the total number of test examples are listed in brackets. In Table (7.7) the individual test error rates for the 2d-artificial dataset are shown. Additionally, in Figures (7.22(a))-(7.23(f)), the decision boundaries learnt by the regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz classifier algorithm as well as the RBF-SVM together with the test data for the last six test sets of the 2d-artificial dataset are plotted column-wise.

For the UCI datasets, the regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz classifier yields test error rates equal to the RBF-SVM classifier results. Moreover, on the 2d-artificial dataset the regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz classifier is superior to the RBF-SVM on average. In particular, as already observed

no.	reg. Lip.	non-reg. Lip.	SVM
1	0.1200	0.1150	0.1150
2	0.1375	0.2075	0.1200
3	0.0925	0.1750	0.0800
4	0.1425	**0.1225**	0.1400
5	**0.0950**	0.1100	0.1050
6	0.1275	0.1320	0.1175
7	0.0875	0.1075	0.0850
8	**0.1475**	**0.1425**	0.1675
9	**0.1425**	0.1425	0.1450
10	**0.1325**	0.1700	0.1525
mean	**0.1225±0.0228**	0.1425 ± 0.0326	0.1228 ± 0.0286

Table 7.7: Single test error rates on the 2d-artificial dataset produced by the non-/regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier and the RBF-SVM classifier.

dataset	avrg. #sol. $\mathbf{K}^*$ 2d-artificial	#sol. $\mathbf{K}^*$ heart	#sol. $\mathbf{K}^*$ breast-cancer
reg. Lip.	9	50	6
non-reg. Lip.	9	49	0

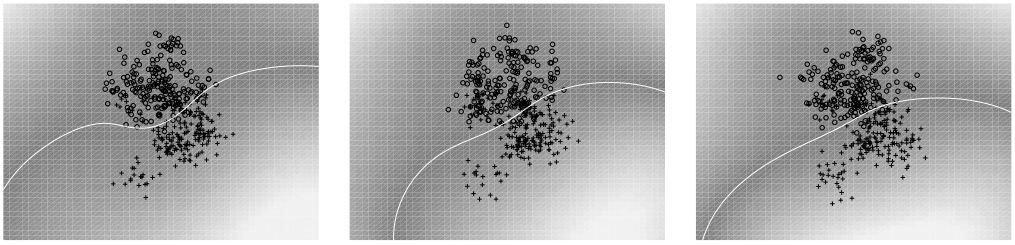
Table 7.8: Number of solution matrices $\mathbf{K}^*$ found during training the non-/regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier.

after the evaluation phase, the variance over the 10 training and test sets is smaller compared to the RBF-SVM's variance. This indicates a more robust decision function found by the regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier with respect to varying datasets. Qualitatively, this can also be noticed from the 2d-decision boundaries drawn in Figures (7.22(a))-(7.23(f)). Compared to the RBF-SVM's decision boundaries, the boundaries learnt by the regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier do not vary much in shape over different training sets. In particular, in Figure (7.22(a)) and Figures (7.23(a))-(7.23(c)), the boundaries are very similar in shape and the resulting classification of the test data is superior to the RBF-SVM's classification. This indicates that these boundaries reflect the true multimodal nature of the data very well yielding a good generalization performance.

On the other hand, the classification performance of the non-regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier is worse than its regularized counterpart, except for the UCI heart dataset. However, from a theoretical point of view as

mentioned in the end of Chapter (6), the non-regularized $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm can be used for an ill-conditioned classification problem that would require a prohibitively large perturbation $\epsilon > 0$ for an application of the regularized $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm. However, ϵ is fairly small for all used datasets (cf. Tab. (7.2)). From a practical point of view, we found out that the non-regularized $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm is sensitive to small changes in the algorithmic parameter settings rendering its evaluation very difficult. Also the computational overhead of the non-regularized $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm forced us to use much fever total search iterations for its parameter evaluation. This might be a reason for worse parameter settings of (γ, C) yielding the worse accuracies compared to the accuracies using the regularized $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm. In Table (7.8), we listed the number of solutions found during training the non-/regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier.

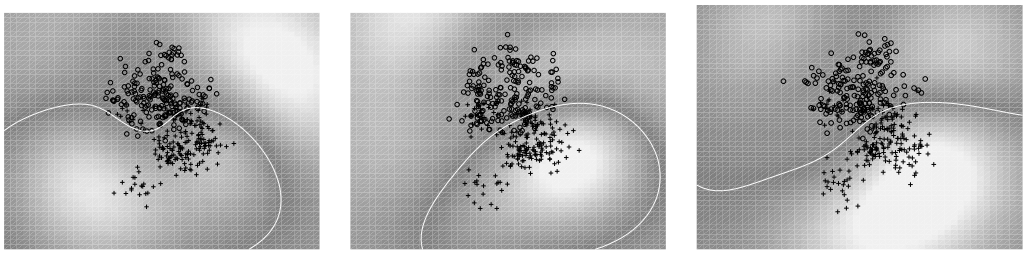
It is conspicuous, that no solution matrix $\mathbf{K}^*$ was found in case of the UCI breast cancer dataset using the non-regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier. The reason is, that due to the training data and the evaluated parameters (γ, C), the first computed matrix $\mathbf{K}_{(0)} = \mathbf{K}(\boldsymbol{x}_0)$ (with $\boldsymbol{x}_0$ set to the mean training vector) resulted in a large q-value that could not be improved further. This may happen if for example γ is small and the training data is such that the superposition of broad RBF-peaks at the data points result in a global peak near the center of all data points. Indeed for UCI breast cancer, the evaluated γ is small compared to the one for the regularized algorithm (cf. Tab. (7.5)). To confirm this argument, we started the search with a matrix $\mathbf{K}(\boldsymbol{x}_0)$ with $\boldsymbol{x}_0$ randomly initialized. In this case, the algorithm's first q-value was much smaller than the former q-value and when climbing the hill many solutions were found improving the subsequent q-values. However, during the same number of total search iterations, the algorithm did not reach the same high q-value corresponding to the mean training vector thus the resulting accuracy was inferior. It is important to note, that this behavior is an artifact of the RBF basis function. Using other basis functions the situation could be totally different and the non-regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier might work well.



(a) reg. RBF-$\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lip. class. (test set #5, err. 9.50%)

(b) reg. RBF-$\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lip. class. (test set #6, err. 12.75%)

(c) reg. RBF-$\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lip. class. (test set #7, err. 8.75%)

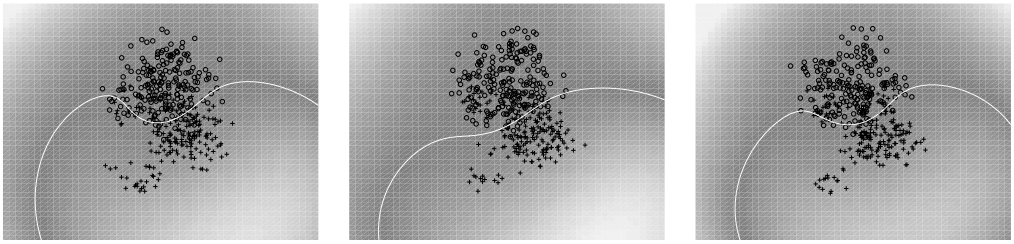


(d) RBF-SVM class. (test set #5, err. 10.50%)

(e) RBF-SVM class. (test set #6, err. 11.75%)

(f) RBF-SVM class. (test set #7, err. 8.50%)

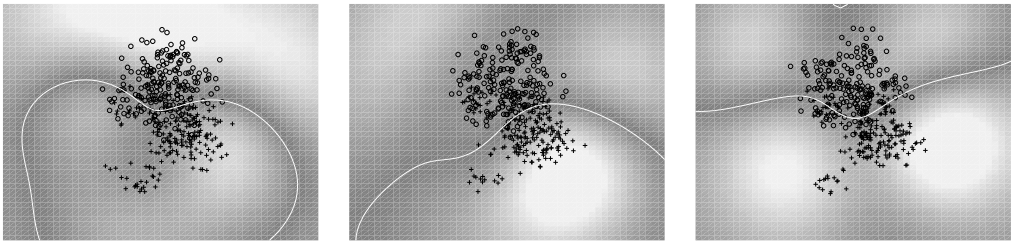
Figure 7.22: Plots showing column-wise the 2d-artificial data of test sets #5-#7 and the learnt decision boundaries. The first row presents the learnt boundaries of the regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz classifier and the second row of the RBF-SVM classifier, respectively.



(a) reg. RBF-$\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lip. class. (test set #8, err. 14.75%)

(b) reg. RBF-$\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lip. class. (test set #9, err. 14.25%)

(c) reg. RBF-$\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lip. class. (test set #10, err. 13.25%)



(d) RBF-SVM class. (test set #8, err. 16.75%)

(e) RBF-SVM class. (test set #9, err. 14.50%)

(f) RBF-SVM class. (test set #10, err. 15.25%)

Figure 7.23: Plots showing column-wise the 2d-artificial data of test sets #8-#10 and the learnt decision boundaries. The first row presents the learnt boundaries of the regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X},\mathbb{R})$-Lipschitz classifier and the second row of the RBF-SVM classifier, respectively.

7.2 Performance Test using Non-Mercer Kernel Functions

In the preceding section, we analyzed the performance of our new learning algorithms in a setting where the SVM is applicable. In this case, the least we would like to expect from the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier is that it works with nearly the same classification performance than the SVM. The experiments using the RBF-kernel function as basis function show that this expectation is indeed satisfied for the 2d-artificial dataset, the UCI heart dataset and the UCI breast-cancer data set. Moreover, on the 2d-artificial dataset, the performance of the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier is slightly superior to the SVM classifier with respect to the averaged performance and the standard deviation.

In this section, we study the applicability of the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier in a scenario where the SVM is known to collapse. As discussed in Chapter (4), one popular function not satisfying Mercer's condition is the hyperbolic tangent basis function:

$$f^{tanh}(\boldsymbol{x}, \boldsymbol{y}) := \tanh\left(a\langle \boldsymbol{x}, \boldsymbol{y}\rangle + r\right). \tag{7.2}$$

It is well-known, that the hyperbolic tangent basis function is not positive definite for most of its parameter $(a, r) \in \mathbb{R} \times \mathbb{R}$, thus it does not satisfy the Mercer's condition. Recently, in [Lin 2003] different parameter ranges of (a, r) are examined for the use of basis functions $\Phi_n^{tanh}(\boldsymbol{x}) := f^{tanh}(\boldsymbol{x}, \boldsymbol{x}_n), \boldsymbol{x}_n \in \mathcal{O}_N \subset \mathcal{X}$ in SVMs. It turns out that only for $a > 0$ and sufficiently small $r < 0$ the function Φ_n^{tanh} is CPD (cf. Sec. 4.2.1). In all other cases, the SVM cannot be trained because the SVM problem (3.52)-(3.54) is mostly infeasible due to an indefinite kernel matrix or a solution corresponds most likely to a very poor local optimum. In Table (7.9) qualitative results due to [Lin 2003] are summarized using a tanh-SVM classifier for different parameter ranges of (a, r).

a	r	qualitative result due to [Lin 2003]
> 0	< 0	Φ_n^{tanh} is CPD if r is small; similar to Φ_n^{rbf} for small a
> 0	> 0	is worse than $(> 0, < 0)$-case or SVM problem is infeasible
< 0	> 0	SVM problem is most likely to be infeasible
< 0	< 0	SVM problem is almost always infeasible

Table 7.9: Behavior of the SVM using $\tanh\left(a\langle \boldsymbol{x}_n, \boldsymbol{x}\rangle + r\right)$.

Contrary to the work of [Lin 2003] focusing only the $(a > 0, r < 0)$-case and the $(a > 0, r > 0)$-case for the tanh-SVM experiments, we are now

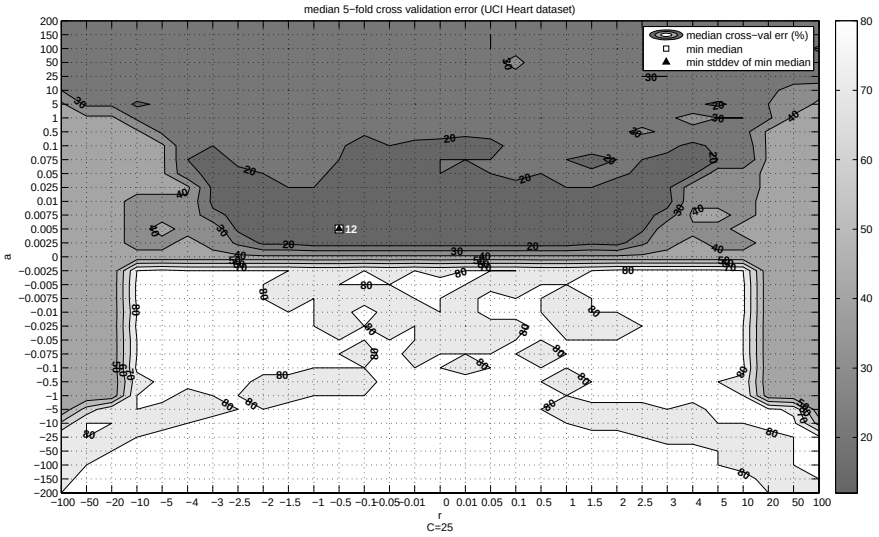


Figure 7.24: Median 5-fold cross-validation errors using the UCI heart dataset and a tanh-SVM for different parameter settings (a, r) and C fixed to the value yielding the lowest cross-validation error (white square).

particularly interested in exploring the behavior of the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier for all parameter ranges in Table (7.9) for which the SVM can be expected to fail.

In accordance with [Lin 2003], the UCI heart dataset is used and split into five subsets for a 5-fold cross-validation. Due to our experience, which we discussed in the preceding section, and in order to limit the computational effort of exploring three parameters $(a, r, C) \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}^+$ via 5-fold cross-validation, we used in the following experiments the regularized $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier, only. Additionally, the maximum number of SA loops, k_{max}, is set to 1% of k_{max} that was used for the SVM performance comparision on the UCI heart dataset (Section (7.1.1), Table (7.2)).

In Figure (7.24) the median 5-fold cross-validation errors are shown for the evaluated parameter ranges of (a, r) summarized in Table (7.9). The parameter C is fixed to the value that yielded the lowest median cross-validation error. The white filled areas mark cross-validation errors greater or equal than 80%, respectively 70% for the light gray filled areas. The white square marks the best evaluted parameter setting, $a = 0.005$ and $r = -0.5$, with 12% median cross-validation error. One observes the SVM indeed has extremely poor performance for the cases $(a < 0, r > 0)$ and $(a < 0, r < 0)$. This agrees with the results reported in [Lin 2003] and summarized in Table (7.9).

A comparison of the results shown in Figure (7.24) with the results de-

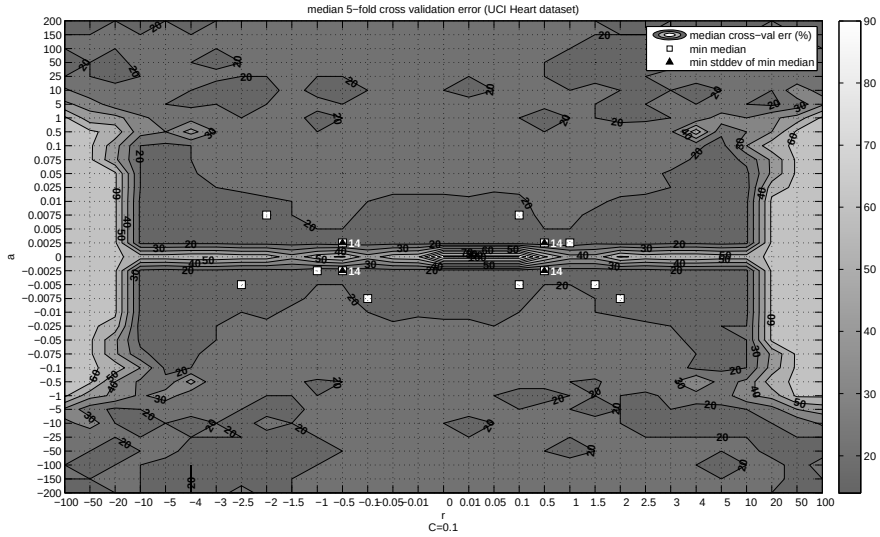


Figure 7.25: Median 5-fold cross-validation errors using the UCI heart dataset and a tanh-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier for different parameter settings (a, r) and C fixed to the value yielding the lowest cross-validation error (white squares).

picted in Figure (7.25) makes obvious the benefit of the tanh-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier against the tanh-SVM. The tanh-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier does not suffer from the indefiniteness of the hyperbolic tangent basis function for the cases $(a < 0, r > 0)$ and $(a < 0, r < 0)$! The entire considered parameter range is well defined except the very uninteresting setting of $a = 0$ has high error rates in the vicinity of $r = 0$. Figure (7.25) also reflects the symmetry of the hyperbolic tangent function. The lowest median cross-validation error (white squares) of 14% obtained using the tanh-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier is slightly worse than the one reached by the tanh-SVM. The reason is that we used only a marginal number of SA-iterations, thus the solutions found by our algorithm are probably very sub-optimal compared to the deterministic solutions found by the SVM. However, in Section (7.1.2), we already showed that the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier performs as well as the SVM in standard situations on different datasets. Thus, we may also expect that the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier reaches similar or better performances than the SVM classifier in the tanh-case if we would increase the number of search steps.

Additionally, in accordance to the work of [Lin 2003], in Figure (7.26) and Figure (7.27) the median 5-fold cross-validation errors obtained by using the tanh-SVM are depicted for different parameter settings of (a, C).

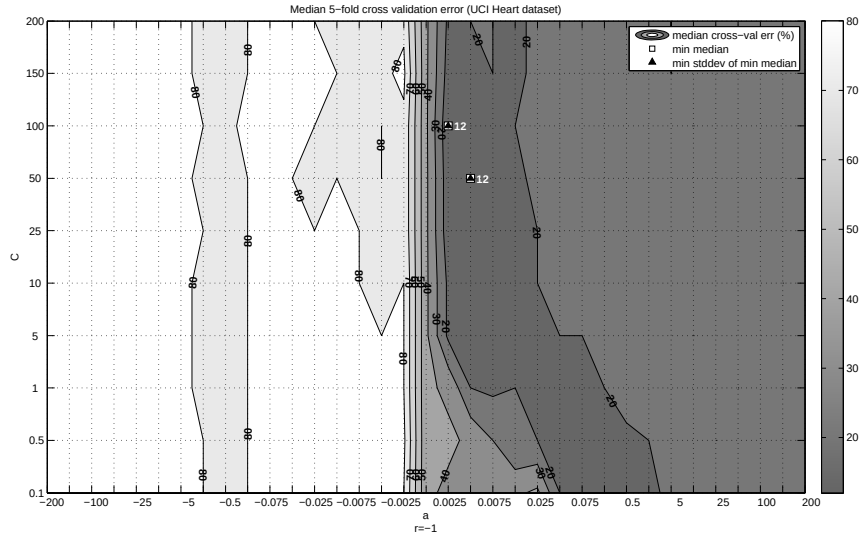


Figure 7.26: Median 5-fold cross-validation errors using the UCI heart dataset and a tanh-SVM for different parameter settings (a, C) and $r < 0$ fixed to the value yielding the lowest cross-validation error (white squares).

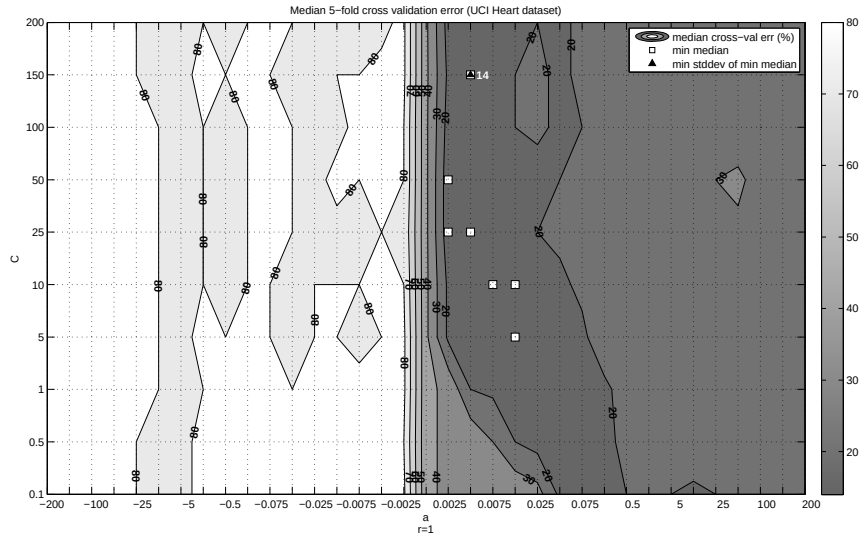


Figure 7.27: Median 5-fold cross-validation errors using the UCI heart dataset and a tanh-SVM for different parameter settings (a, C) and $r > 0$ fixed to the value yielding the lowest cross-validation error (white squares).

The parameter r is fixed to the positive respectively negative value with lowest cross-validation error. These figures indicate how well the data is separated with a large margin for different trade-off parameters C by the SVM algorithm. Again, one observes the SVM indeed has extremely poor performance for the cases $(a < 0, r > 0)$ and $(a < 0, r < 0)$. That means for the corresponding C values it is not possible to separate the data well using a tanh-SVM. Although it is difficult to interpret the value of C quantitatively, it is possible to state qualitatively that for decreasing C more points may rest in the margin area or may even be misclassified, and vice versa for increasing C. In our experiments, for all indefinite cases of the hyperbolic tangent basis function, the SVM algorithm terminated successfully but with a solution $\boldsymbol{\alpha}^*$ with nearly all components $\alpha_i^* = C$. This indicates the severe tendency of the SVM problem to become infeasible for any C. Moreover, it justifies the theory presented in Section (3.6.3) as well as the SVM's shortcomings discussed in Chapter (4) that there exists no feasible SVM solution separating the data well in any case of non-kernel functions.

Fortunately, the situation for the tanh-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm is totally different. In Figure (7.28) and Figure (7.29) one clearly observes that our algorithm has found many solutions with relatively low cross-validation errors for parameter settings the SVM fails with errors greater than 80%. Thus, the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier has always found a feasible solution.

Additionally, in order to quantify the completely checked parameter range of (a, r, C), we summarized in Table (7.10) the averaged 5-fold cross-validation results and corresponding standard deviations. As shown in Table (7.10), the tanh-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier performs better than the tanh-SVM on average with respect to varying parameter settings.

a	**r**	**tanh-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz class.**	**tanh-SVM class.**
> 0	< 0	$24.83 \pm 10.46\%$	$27.43 \pm 8.80\%$
> 0	> 0	$24.29 \pm 10.33\%$	$26.60 \pm 8.38\%$
< 0	> 0	$24.86 \pm 10.61\%$	$75.61 \pm 12.66\%$
< 0	< 0	$24.30 \pm 10.30\%$	$75.78 \pm 12.79\%$

Table 7.10: Averaged 5-fold cross-validation errors obtained by the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier and the SVM classifier using the non-kernel function $\tanh\left(a\langle \boldsymbol{x}_n, \boldsymbol{x}\rangle + r\right)$.

The very nice experimental results of this section justify our theoretical treatment and let us conclude the benefit from using the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier - the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier is usable when the SVM is not

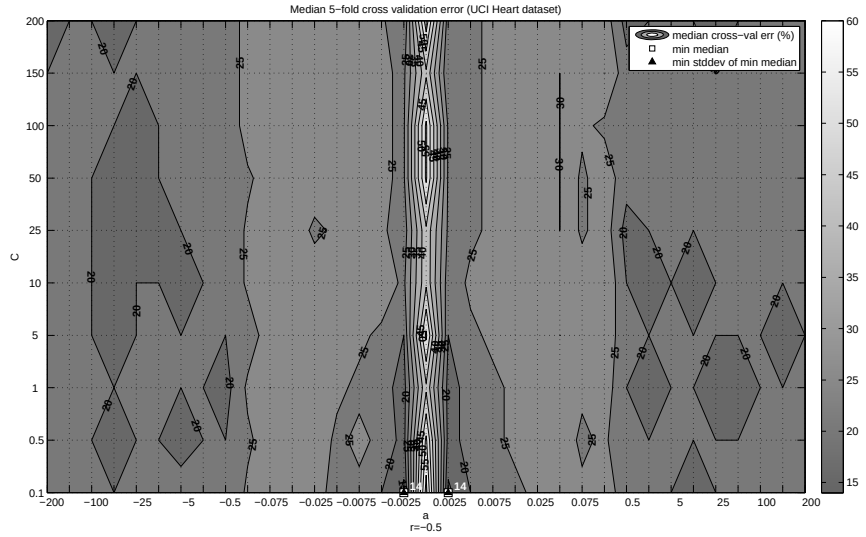


Figure 7.28: Median 5-fold cross-validation errors using the UCI heart dataset and a $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier for different parameter settings (a, C) and $r < 0$ fixed to the value yielding the lowest cross-validation error (white squares).

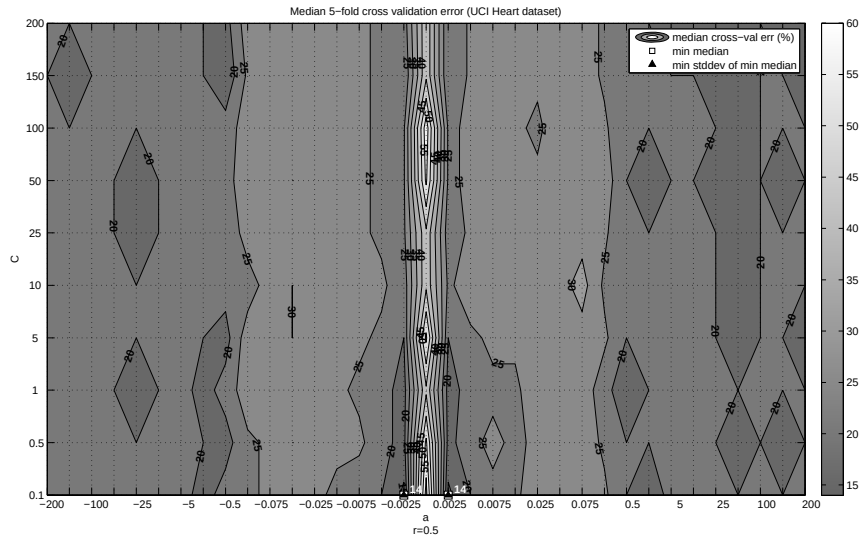


Figure 7.29: Median 5-fold cross-validation errors using the UCI heart dataset and a $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier for different parameter settings (a, C) and $r > 0$ fixed to the value yielding the lowest cross-validation error (white squares).

capable. In particular, the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier is able to find feasible solutions also in cases of indefinite basis functions and for large C-values. This indicates the ability to separate the data well with a large margin in a feature space implicitly defined by indefinite basis functions used by the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier.

Summary

In this chapter, experiments using the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm are presented. We used three different databases: a 2d-artificial dataset, and two real world datasets, namely the UCI heart and UCI breast-cancer dataset.

The first experiment is introduced in Section (7.1). Therein, after summarizing the parameter tuning in Section (7.1.1), classification results of the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier compared to results of the SVM classifier are presented in Section (7.1.2) in a standard situation using the popular RBF-kernel function as basis function. The evidence of this experiment is that the regularized $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm has equal or better classification performance than the SVM on the considered datasets and basis function. The non-regularized counterpart performs slightly worse, which might be reasoned by an inferior algorithmic parameter selection as discussed in Section (7.1.2). In turn, the inadequate parameter selection is due to the severe sensitivity of the present implementation of the non-regularized $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm to algorithmic parameter changes and due to the higher computational effort prohibiting a large number of search steps during validation.

The second experiment introduced in Section (7.2) aims at pointing out the benefit from using the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm. For this purpose, we used a non-kernel function as basis function, namely the hyperbolic tangent function well-known as artificial neural network's activation function. The tanh-function is indefinite respectively Mercer's condition is violated, thus the tanh-function is not usable as basis function in SVMs for a large range of function parameters. That means, the SVM problem is infeasible or it results in a very poor classification performance due to poor local minima of the SVM objective function that is not convex anymore. Following the analysis of [Lin 2003] for the tanh-SVM, we examined the parameter ranges for which the SVM fails.

Our experimental results impressively show, that the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm does not suffer from the indefiniteness of the hyperbolic tangent function for any parameter selection. Moreover, because the tanh-experiment is conceived as an archetype of situations for which non-kernel

functions might be the better choice, the results let us conclude also from a practical point of view, that the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm opens the way for the design of new decision functions in a maximum margin based classifier. Furthermore, as we showed in Section (7.1.2), using Lipschitz classifiers is in no way inferior to using conventional machine learning methods (like SVMs with kernel-functions), hence the methodology proposed here does not lead to performance loss and can be used without restriction. An expert is now able to design a maximum margin classifier facilitating his experience and prior knowledge of the classification problem at hand. There is no reason to choose only from kernel-functions, anymore, neither theoretically nor practically.

Chapter 8

Conclusion, Open Issues and Future Work

This thesis focused on the development of new learning methods for classification satisfying two main objectives:

1. new algorithms have to implement the same generalizing ideas making the SVM one of the most successful learning methods,
2. new algorithms have to overcome the SVM's restrictions, that prohibit their use in applications where a priori knowledge shall be functionally encoded in the classifier, or where decision functions that are not built up of Mercer-kernel functions would be more suitable.

Our new learning algorithms satisfy (1.) and (2.) and thus open the way for a design of new classifiers that go beyond the SVM. Moreover, the backbone of our learning algorithms is a justified theoretical concept, namely that maximum margin separating hyperplanes, which already made SVMs robust and superior to many empirical learning approaches, on the one hand. On the other hand, our learning algorithms facilitate new and specific applications and data domains due to the flexibility of handling a very large class of basis functions to construct a decision function. Because a designer of a maximum margin classifier is not restricted to a handful of kernel functions anymore, he is now free to exploit any experience or prior knowledge about the classification problem in a more general learning framework.

After a general survey on classification and methods in Chapter (2), we discussed in Chapter (3) the theoretical ideas leading to the maximum margin concept and in particular to the SVM. Our first objective of developing maximum margin based learning methods is motivated by the theoretical conclusion from Chapter (3) that maximum margin hyperplanes may yield a lower expected error than other separating hyperplanes. This theoretical conclusion is also practically proven by the use of the SVM in many applications (e.g. Chapter (2)).

In Chapter (4), the shortcomings of SVMs were presented and examples were given motivating our second objective to overcome the SVM's restrictions. We presented related work also trying to overcome these drawbacks.

Chapter (4) was concluded with the insight that the available related work is of theoretical value but does not provide practical algorithms that reached the sophisticated goal of generalizing the SVM to larger function classes than kernel functions, so far. Even though abstract algorithms are proposed, they are far apart from a practical implementation or they impose new severe restrictions on the usable decision function class.

Guided by our objectives, we reviewed in Chapter (5), Section (5.1), a theoretical framework, namely the Lipschitz classifier [von Luxburg 2004], that generalizes the SVM via a feature space transformation called Lipschitz embedding. This framework leads to a maximum margin classifier respectively a learning algorithm using decision functions which have to be Lipschitz continuous. Therefore, the decision function space of the Lipschitz classifier constitutes many more functions than are accessible to the SVM due to the SVM's restrictions imposed by Mercer's condition.

Unfortunately, the most general version of the Lipschitz classifier is not easy to implement, because the Lipschitz constant of any decision function of the entire considered space must be computed somehow.

Nevertheless, we observed in Section (5.2) that the Lipschitz constant can be derived analytically for any decision function in a still very large decision function space - the space of at least one-time continuously differentiable functions defined on a Euclidean domain (5.2.1). Having an implementable algorithm in mind, and in compliance with the SVM, we further restricted ourselves to the decision function space of affine-linear combinations of at least one-time continuously differentiable basis functions (Def. (5.2.1)).

From this starting point, we obtained a constrained minimax formulation of the Lipschitz classifier (Alg. 5.2.1). Due to the difficult problem structure, we derived from the minimax problem an equivalent semi-infinite program (SIP) yielding a completely new and for the first time practical $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm (Alg. (5.2.3)). The primal $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm is actually implementable via a discretization of the data space, but it can break down standard solvers due to the complicated feasible set. In particular, discretization is impractical in case of high dimensional data spaces.

In order to circumvent these drawbacks of the primal problem, and to further exploit the problem's inherent structure as much as possible, we derived a dual SIP and proved a strong duality theorem showing the equality of the primal and the dual optimal values of the SIP (Theorem (5.2.3)). Finally, we developed in Chapter (5) the dual version of the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm (Alg. (5.2.4)).

We conclude that the benefit from the dual formulation is the transformation of the very difficult to solve constrained minimax problem into a concave

max-max problem with a much simpler structured feasible set. The max-max problem consists of a concave quadratic problem constrained to a convex feasible set, which can be easily solved with standard methods, on the one hand. And on the other hand, we have to perform a maximization of the optimal values of the quadratic programming problem over a convex hull of positive semi-definite matrices. Although sounding still complex, this is a significant simplification compared to the nonlinear minimax optimization over an arbitrary complicated feasible set, which is the data space itself. Moreover, the dual version of the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm (Alg. (5.2.3)) goes without any discretization and it opens the way to iterate a global solution using well-known standard optimization methods.

In Chapter (6), we concretized the abstract algorithms derived in Chapter (5) towards an implementation in software. For this purpose, we adapted Mehrotra's primal-dual-predictor-corrector algorithm (Alg. 6.1.1) to solve the involved convex constrained quadratic programming problem. Further, we proposed a simulated annealing based search heuristics (Alg. 6.2.2) for finding candidate matrices used to construct the convex hull in the second stage of the optimization. In order to find the optimal convex combination of two matrices of the convex hull, we developed a gradient solver (Alg. 6.2.1) based on a nonmonotone spectral projected gradient method proposed by [Birgin 2000]. To apply the projected gradient method to our particular problem, we proved the general Theorem (6.2.1) about the right-directional derivative of the optimal values of the constrained quadratic problem with respect to the parameter of the considered convex combination. It turned out, if using regularization or an algebraic reparameterization of the involved linear equation system of the QP-problem's feasible set then finding the optimal convex combination between two matrices is itself a convex optimization problem with an unique global solution. The global solution is easily found by Algorithm (6.2.1) using Theorem (6.2.1).

We concluded Chapter (6) with two new maximum margin based learning methods that overcome the shortcomings of SVMs (Chapter (4)), namely the limitation to kernel based decision functions: If using regularization, we derived the regularized $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm (Alg. 6.3.1) and in case of the algebraic reparameterization we obtained the non-regularized $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm (Alg. 6.3.2).

In the last Chapter (7), we performed experiments with our new learning algorithms to validate whether our main objectives are satisfied not just from a theoretical but also from a practical point of view. For this purpose we implemented all components of the non-/regularized $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm developed in Chapter (6) in the MATLAB® programming

environment[1]. Then we investigated two different scenarios:

The first experiment is a comparison between the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm and the SVM algorithm on three different datasets, namely a 2d-artificial dataset, the UCI heart dataset and the UCI breast-cancer dataset, in a standard situation using positive definite basis functions Φ_n. To fairly compare both learning machines, we used in both algorithms the popular RBF-kernel, i.e. $\Phi_n(\boldsymbol{x}) = \exp\left(-\gamma\|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2\right)$, as basis function for learning a decision function. Our experiment showed that the regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithms performs equal or slightly better than the RBF-SVM (Table (7.6)). Moreover, the classification results obtained by the RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier have always lower variance with respect to the different 2d-artificial datasets. This indicates the robustness of the RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier against varying datasets is superior to the RBF-SVM's robustness. On the other hand, the non-regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithms performs worse than its regularized counterpart on the 2d-artificial dataset and the UCI breast-cancer set. But the classification results were equal on the UCI heart dataset. We argued, that a reason for the worse accuracies is the sensitivity of the non-regularized RBF-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithms with respect to the algorithmic parameters. This made an optimal parameter tuning very difficult and led to suboptimal settings. However, the classification performance of the non-regularized $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm can be very good, if other basis functions than RBF-kernels are used resulting in a lower sensitivity to algorithmic parameter changes.

We conclude that it makes no great difference if using the regularized $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier or the SVM in standard situations with respect to the classification accuracy. Regarding the current implementations of the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm, we suggest to use the regularized algorithm instead of the non-regularized version due to its lower computational costs and a better manageable adjustment of algorithmic parameters. Nevertheless, the non-regularized $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm will be a good choice if a classification problem requires very large perturbations.

In order to motivate the benefit of the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier compared to the SVM classifier, we examined in our second experiment the situation the SVM indeed fails. For this purpose, we employed the indefinite hyperbolic tangent function, i.e. $\tanh\left(a\langle \boldsymbol{x}, \boldsymbol{y}\rangle + r\right)$, as basis functions in both algorithms and computed 5-fold cross-validation errors on the UCI heart dataset for a large set of parameter configurations (a, r). We chose this scenario as

[1]The $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier's MATLAB-code can be sourced from the author for academic use only.

archetype for non-standard situations, because the tanh-function is a popular function due to its relation to artificial neural networks (Chapter (2)), and because recently [Lin 2003] has studied its use in SVMs extensively on the UCI heart dataset too. Our results for the tanh-SVM classifier depicted in the Figures (7.24), (7.26), (7.27) and Table (7.10) agree completely with the results reported in [Lin 2003].

We conclude the tanh-SVM fails with high cross-validation errors greater than 70% for a large range of parameters, namely for $a < 0, r > 0$ and $a < 0, r < 0$.

In contrast, Figures (7.25), (7.28), (7.29) and Table (7.10) show that the tanh-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm does not suffer from the indefiniteness of the hyperbolic tangent function for the entire considered parameter range. Moreover, the averaged cross-validation error for all checked parameter configurations (a, r, C), including the trade-off parameter C, reads as $24.86 \pm 10.61\%$ $(a < 0, r > 0)$ and $24.30 \pm 10.30\%$ $(a < 0, r < 0)$ using the tanh-$\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier, respectively $75.61 \pm 12.66\%$ $(a < 0, r > 0)$ and $75.78 \pm 12.79\%$ $(a < 0, r < 0)$ using the tanh-SVM.

We conclude the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm finds a maximum margin separating hyperplane induced by indefinite basis functions Φ_n for which the SVM fails to find an appropriate solution. Therefore, the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm is usable for a larger range of applications than the SVM. This includes applications for which prior knowledge has to be functionally encoded or where the designer of the classifier needs more flexibility in choosing from appropriate decision functions.

The benefit of the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithms comes not without extra costs. We presented in this thesis a number of fruitful ways to exploit the structure of the difficult to solve optimization problem such that a usable implementation is now actually available - however, our new maximum margin algorithms are computational expensive compared to the SVM for example. Currently, our implementations of the algorithms are in a similar position the SVM was in the late 1980s when convex optimization, and in particular interior point methods, were in their infancy. At that time, SVMs were generally considered to be inapplicable to real world problems due to their computational effort. After years of research, matters have changed, and the SVM is one of the most sophisticated and successful machine learning methods, so far.

In the present thesis, we proposed a foundation for a new type of learning machine and its implementation that inherits the theoretical ideas of the SVM but without suffering from the restrictions imposed by Mercer's condition. We opened the way to the design of new classifiers suitable to particular applications and data domains. Moreover, we also opened the way for inter-

esting questions for future research and development of the algorithms. In particular, our prearrangement enables to examine possibilities of improving independently each component of the algorithms on its own. Several of the evolving lines of research will be addressed now.

For example, a better characterization of the image of the matrix valued mapping $\mathbf{K} : \mathcal{X} \to \mathcal{S}_{0+}^n$ into the convex cone of symmetric positive semi-definite $n \times n$-matrices, denoted $\mathcal{S}_{0+}^n$, would give probably further usable insights. Such a structural statement could guide the stochastic search for a candidate matrix more efficiently or it might lead to a deterministic optimization approach. Moreover, other search heuristics like for example genetic algorithms and evolution strategies could be investigated instead of the proposed simulated annealing approach.

Another question regards the possibility to derive a representer theorem. In this case, one could immediately derive a deterministic algorithm. It is known, that for the general Lipschitz classifier algorithm using the entire space of Lipschitz functions this is not possible in terms of the individual training data [von Luxburg 2004]. But here, we restricted our attention to the space of finite linear combinations of at least one-time continuously differentiable basis functions. Because the space of at least one-time continuously differentiable real functions is dense in the space of all continuous real functions, it could lead to a representer theorem in terms of the training data even for an infinite linear combination of basis functions. This is also likely, because recently in [Minh 2004] a Representer Theorem has been proved for a subalgebra of Lipschitz functions that is dense in the space of Lipschitz functions (Section (4.2.3)).

A great impact on the overall performance, particularly for large scale problems, would be an improvement of the QP-solver. In case of the SVM it is possible to use a decomposition strategy for solving the constrained QP-problem very fast by much smaller quadratic sub-problems. Fortunately, in case of the regularized $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm we showed that it is possible to reformulate the QP-problem equivalently such that the feasible set is identical with the feasible set of the SVM problem. Thus, one can apply a similar decomposition strategy as used for the SVM. Unfortunately, in case of the non-regularized $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm the feasible set consists of an additional linear equation system making an efficient decomposition strategy difficult at first.

So far, we did not use any termination criterion measuring the quality of the iterates toward a solution. Such a criterion is in principle derivable from the KKT optimality conditions yielding an estimate of the gap between the dual optimal value and the primal optimal value that must vanish at a optimal point (App. B.3.1). We suggest such a criterion for a future implementation

of the proposed algorithms.

Additionally, it would be advantageous to force a more sparse solution, because this would reduce the computational effort of evaluating the decision function. In the regularized $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm this could be reached by using another regularization enforcing sparsity. In the non-regularized version one could try to optimize the degrees of freedom of the solution associated to the null space of the involved linear equation system.

Appendix A

Mathematical Background

A.1 Important Inequalities

A.1.1 Hoeffding's Inequality

Let be $x_1, \ldots, x_k$ independent observations of a random variable $X \in [a, b]$ and let be $E_k := \frac{1}{k} \sum_{i=1}^{k} x_k$ its empirical mean. Then, for any $\epsilon > 0$,

$$P\left(|E(X) - E_k| \geq \epsilon\right) \leq \exp\left(-\frac{2k\epsilon^2}{(b-a)^2}\right) \tag{A.1}$$

is called *Hoeffding's inequality* [Hoeffding 1963]. It means, that the empirical mean E_k of a random variable X converges to its actual expectation $E(X)$ with exponential rate.

A.1.2 Jensen's Inequality

Let be f a convex function and $\lambda_1, \ldots, \lambda_k \geq 0$ with $\lambda_1 + \cdots + \lambda_k = 1$, then

$$f\left(\sum_{i=1}^{k} \lambda_i \boldsymbol{x}_i\right) \leq \sum_{i=1}^{k} \lambda_i f\left(\boldsymbol{x}_i\right) \tag{A.2}$$

is called *Jensen's inequality* [Jensen 1906]. The inequality extends to infinite sums, integrals and expectation values.

A.2 Mathematical Spaces

A.2.1 (Semi-)Metric Space

Definition A.2.1 ((semi-)metric space)**.** *A (semi-)metric space is a pair $(\mathcal{X}, d)$ with nonempty set $\mathcal{X}$ and a function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, called (semi-)metric, with the properties for all $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathcal{X}$:*

1. *$d(\boldsymbol{x}, \boldsymbol{y}) \geq 0$ (positivity)*
2. *$d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}, \boldsymbol{x})$ (symmetry)*
3. *$d(\boldsymbol{x}, \boldsymbol{y}) \leq d(\boldsymbol{x}, \boldsymbol{z}) + d(\boldsymbol{z}, \boldsymbol{y})$ (triangle inequality)*
4. *in case of a metric space, it must also hold $d(\boldsymbol{x}, \boldsymbol{y}) = 0 \Leftrightarrow \boldsymbol{x} = \boldsymbol{y}$.*

A.2.2 Banach Space

Definition A.2.2 (Banach space)**.** *A Banach space $\mathcal{B}$ is a vector space endowed with a function $\|\cdot\| : \mathcal{B} \to \mathbb{R}$ such that every Cauchy sequence in $\mathcal{B}$ converges in the induced metric $d(\boldsymbol{x}, \boldsymbol{y}) := \|\boldsymbol{x} - \boldsymbol{y}\|$ to an element of $\mathcal{B}$ (i.e. the space is completed). For all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{B}$, $\alpha \in \mathbb{R}$, the function $\|\cdot\|$ has the properties:*

1. $\|\boldsymbol{x}\| \geq 0$ *with* $\|\boldsymbol{x}\| = 0 \Leftrightarrow \boldsymbol{x} = \mathbf{0}$
2. $\|\alpha \cdot \boldsymbol{x}\| = |\alpha| \cdot \|\boldsymbol{x}\|$
3. $\|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}| + \|\boldsymbol{y}\|$.

A.2.3 Hilbert Space

Definition A.2.3 (Hilbert space)**.** *A Hilbert space $\mathcal{H}$ is a vector space endowed with a bilinear function $\langle\cdot,\cdot\rangle : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ (inner product) such that every Cauchy sequence in $\mathcal{H}$ converges in the induced metric $d(\boldsymbol{x}, \boldsymbol{y}) := \sqrt{\langle \boldsymbol{x} - \boldsymbol{y}, \boldsymbol{x} - \boldsymbol{y}\rangle}$ to an element of $\mathcal{H}$ (i.e. the space is completed). For all $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathcal{H}$, $\alpha \in \mathbb{R}$, the function $\langle\cdot,\cdot\rangle$ has the properties:*

1. $\langle \boldsymbol{x}, \boldsymbol{x}\rangle \geq 0$ *with* $\langle \boldsymbol{x}, \boldsymbol{x}\rangle = 0 \Leftrightarrow \boldsymbol{x} = \mathbf{0}$
2. $\langle \boldsymbol{x} + \boldsymbol{y}, \boldsymbol{z}\rangle = \langle \boldsymbol{x}, \boldsymbol{z}\rangle + \langle \boldsymbol{y}, \boldsymbol{z}\rangle$
3. $\langle \alpha \cdot \boldsymbol{x}, \boldsymbol{y}\rangle = \alpha \cdot \langle \boldsymbol{x}, \boldsymbol{y}\rangle$

Note, from 1.)-4.) follows the Cauchy-Schwarz's inequality:

$$|\langle \boldsymbol{x}, \boldsymbol{y}\rangle| \leq \sqrt{\langle \boldsymbol{x}, \boldsymbol{x}\rangle}\sqrt{\langle \boldsymbol{y}, \boldsymbol{y}\rangle}.$$

A vector space endowed with an inner product which is not completed is called Pre-Hilbert space.

A.3 Convex Optimization Theory

A.3.1 Convex Sets

Definition A.3.1 (Convex Sets)**.** *A set $\mathcal{X}$ is called convex, if for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$ and $0 \leq \lambda \leq 1$ it holds*

$$\lambda \boldsymbol{x}_1 + (1 - \lambda)\boldsymbol{x}_2 \in \mathcal{X}. \tag{A.3}$$

See Figure (A.1).

A.3.2 Convex (Concave) Functions

Definition A.3.2 (Convex (Concave) Functions)**.** *A function $f : \mathcal{X} \to \mathbb{R}$ is called convex, if for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$ and $0 \leq \lambda \leq 1$ it holds*

$$f(\lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2) \leq \lambda f(\boldsymbol{x}_1) + (1-\lambda) f(\boldsymbol{x}_2). \tag{A.4}$$

See Figure (A.2). A functions f is concave, if $-f$ is convex.

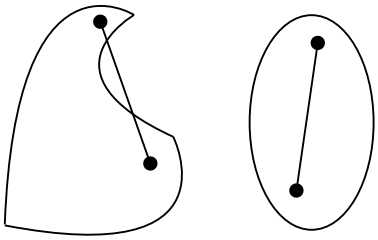


Figure A.1: A non-convex and a convex set.

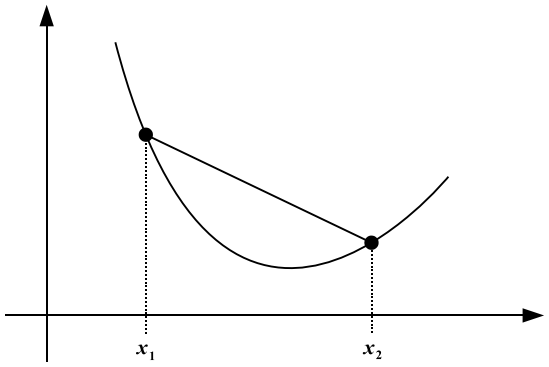


Figure A.2: Graph of a convex function.

A.3.3 Karush-Kuhn-Tucker Optimality Conditions

Let be $F : \mathbb{R}^L \to \mathbb{R}$, $G_j : \mathbb{R}^L \to \mathbb{R}$ $\forall 1 \leq j \leq k$ convex functions. Consider the convex (*primal*) problem

$$inf \left\{ F(\boldsymbol{w}) : \boldsymbol{w} \in \mathcal{S} \right\} \tag{A.5}$$

with convex feasible set

$$\mathcal{S} := \left\{ \boldsymbol{w} \in \mathbb{R}^L : G_j(\boldsymbol{w}) \leq 0 \ \forall 1 \leq j \leq k \right\}. \tag{A.6}$$

Further, the function $L : \mathbb{R}^L \times \mathbb{R}^k \to \mathbb{R}$ with

$$L(\boldsymbol{w}, \boldsymbol{\mu}) := F(\boldsymbol{w}) + \sum_{j=1}^{k} \mu_j G_j(\boldsymbol{w}) \tag{A.7}$$

is called *Lagrange function* with *Lagrange multipliers* $\mu_j \geq 0 \;\; \forall 1 \leq j \leq k$.

Note, without loss of generality, the following statements about problems of the form (A.5) carry over to feasible sets $\mathcal{S}$ described by equality constraints too.

Theorem A.3.1 (Kuhn-Tucker Saddle Point Condition [Kuhn 1951])**.** *Consider a optimization problem of the form (A.5) with Lagrange function (A.7). If a pair* $(\boldsymbol{w}^*, \boldsymbol{\mu}^*) \in \mathbb{R}^L \times \mathbb{R}^k$ *with* $\boldsymbol{\mu}^* \geq \mathbf{0}$ *exists, such that for all* $\boldsymbol{w} \in \mathbb{R}^L$ *and* $\boldsymbol{\mu} \in [0, \infty)^k$ *holds*

$$L(\boldsymbol{w}^*, \boldsymbol{\mu}) \leq L(\boldsymbol{w}^*, \boldsymbol{\mu}^*) \leq L(\boldsymbol{w}, \boldsymbol{\mu}^*), \tag{A.8}$$

i.e. $(\boldsymbol{w}^*, \boldsymbol{\mu}^*)$ *is a saddle point, then* $\boldsymbol{w}^*$ *is a feasible solution of (A.5) with* $F(\boldsymbol{w}^*) = L(\boldsymbol{w}^*, \boldsymbol{\mu}^*)$.

Proof. For a proof see e.g. [Mangasarian 1969].

∎

Now, suppose the feasible set $\mathcal{S}$ satisfies

Definition A.3.3 (Slater's Constraint Qualification)**.** *A convex optimization problem (A.5) satisfies the Slater's constraint qualification, if it exists a vector* $\hat{\boldsymbol{w}} \in \mathbb{R}^L$ *such that*

$$G_j(\hat{\boldsymbol{w}}) < 0 \;\forall 1 \leq j \leq k. \tag{A.9}$$

Then, the following theorems well-known from optimization theory hold:

Theorem A.3.2 (Karush-Kuhn-Tucker (KKT) conditions [Karush 1939], [Kuhn 1951])**.** *Let be* $\boldsymbol{w}^* \in \mathcal{S}$ *a minimum point of the convex optimization problem (A.5) with feasible set* $\mathcal{S}$ *satisfying the Slater's constraint qualification. Then it exist Lagrange multipliers* $\boldsymbol{\mu}^* \in \mathbb{R}^k$ *such that the tuple* $(\boldsymbol{w}^*, \boldsymbol{\mu}^*)$ *satisfies the KKT-conditions*

$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}^*, \boldsymbol{\mu}^*) = \mathbf{0} \tag{A.10}$$

$$\nabla_{\boldsymbol{\mu}} L(\boldsymbol{w}^*, \boldsymbol{\mu}^*) \leq \mathbf{0} \tag{A.11}$$

$$\mu_j^* G_j(\boldsymbol{w}^*) = 0 \;\forall 1 \leq j \leq k \tag{A.12}$$

$$\mu_j^* \geq 0 \;\forall 1 \leq j \leq k. \tag{A.13}$$

Proof. For a proof see e.g. [Nocedal 1999], [Mangasarian 1969].

∎

Theorem A.3.3. *Let be $(\boldsymbol{w}^*, \boldsymbol{\mu}^*)$ a KKT-point satisfying the KKT-conditions (A.3.2) of the convex optimization problem (A.5). Then $\boldsymbol{w}^* \in \mathcal{S}$ is a minimum point of (A.5).*

Proof. For a proof see e.g. [Nocedal 1999], [Mangasarian 1969].

■

Together, Theorem (A.3.2) and Theorem (A.3.3) show that the KKT-conditions are necessary and sufficient for optimality of convex optimization problems. Clearly, the KKT-conditions extend to linear constraints too, because linear functions are convex as well as concave.

A.3.4 Lagrange Duality

Definition A.3.4 (Dual of a Lagrange function)**.** *The function $\mathcal{D}L_{\mathcal{S}} : \mathbb{R}^k \to \mathbb{R}$ with*

$$\mathcal{D}L_{\mathcal{S}}(\boldsymbol{\mu}) := \inf_{\boldsymbol{w} \in \mathbb{R}^L} L(\boldsymbol{w}, \boldsymbol{\mu}) \tag{A.14}$$

is called the dual of the Lagrange function (A.7).

Theorem A.3.4. *The dual function $\mathcal{D}L_{\mathcal{S}}$ (A.3.4) is a concave function on the convex domain $dom(\mathcal{D}L_{\mathcal{S}}) := \left\{\boldsymbol{\mu} \in \mathbb{R}^k \ : \ \boldsymbol{\mu} \geq \mathbf{0}, \mathcal{D}L_{\mathcal{S}}(\boldsymbol{\mu}) > -\infty\right\}$.*

Proof. For all $\boldsymbol{w} \in \mathbb{R}^L$, $\boldsymbol{\mu}^1, \boldsymbol{\mu}^2 \in dom(\mathcal{D}L_{\mathcal{S}})$ and any $\alpha \in [0,1]$ it holds

$$\begin{aligned} L(\boldsymbol{w}, \alpha\boldsymbol{\mu}^1 + (1-\alpha)\boldsymbol{\mu}^2) &= F(\boldsymbol{w}) + \sum_{j=1}^{k} \left(\alpha\mu_j^1 + (1-\alpha)\mu_j^2\right) G_j(\boldsymbol{w}) \\ &= \alpha\left(F(\boldsymbol{w}) + \sum_{j=1}^{k} \mu_j^1 G_j(\boldsymbol{w})\right) \\ &\quad + (1-\alpha)\left(F(\boldsymbol{w}) + \sum_{j=1}^{k} \mu_j^2 G_j(\boldsymbol{w})\right) \\ &= \alpha L(\boldsymbol{w}, \boldsymbol{\mu}^1) + (1-\alpha) L(\boldsymbol{w}, \boldsymbol{\mu}^2). \end{aligned} \tag{A.15}$$

Taking the infimum, one gets

$$\inf_{\boldsymbol{w} \in \mathbb{R}^L} L(\boldsymbol{w}, \alpha\boldsymbol{\mu}^1 + (1-\alpha)\boldsymbol{\mu}^2) \geq \alpha \inf_{\boldsymbol{w} \in \mathbb{R}^L} L(\boldsymbol{w}, \boldsymbol{\mu}^1) + (1-\alpha) \inf_{\boldsymbol{w} \in \mathbb{R}^L} L(\boldsymbol{w}, \boldsymbol{\mu}^2).$$

Thus, we have

$$\mathcal{D}L_{\mathcal{S}}(\alpha\boldsymbol{\mu}^1 + (1-\alpha)\boldsymbol{\mu}^2) \geq \alpha\mathcal{D}L_{\mathcal{S}}(\boldsymbol{\mu}^1) + (1-\alpha)\mathcal{D}L_{\mathcal{S}}(\boldsymbol{\mu}^2)$$

that means concavity of $\mathcal{D}L_{\mathcal{S}}$. Further, this implies due to $\boldsymbol{\mu}^1, \boldsymbol{\mu}^2 \in dom(\mathcal{D}L_{\mathcal{S}})$ that $\alpha\boldsymbol{\mu}^1 + (1-\alpha)\boldsymbol{\mu}^2 \in dom(\mathcal{D}L_{\mathcal{S}})$. Thus, $dom(\mathcal{D}L_{\mathcal{S}})$ is a convex set.

■

Note, Theorem (A.3.4) is also true, if problem (A.5) is not convex.

A.3.4.1 Weak Duality

Definition A.3.5 (Dual problem)**.** *The problem*

$$\sup(\mathcal{D}) := \sup_{\boldsymbol{\mu} \geq \mathbf{0}} \mathcal{D}L_{\mathcal{S}}(\boldsymbol{\mu}) \tag{A.16}$$

with $\boldsymbol{\mu} \in \mathbb{R}^k$ *is called the dual problem associated to the primal problem*

$$\inf(\mathcal{P}) := \inf \{F(\boldsymbol{w}) \ : \ \boldsymbol{w} \in \mathcal{S}\} \tag{A.17}$$

with feasible set $\mathcal{S} := \left\{\boldsymbol{w} \in \mathbb{R}^L \ : \ G_j(\boldsymbol{w}) \leq 0 \ \forall 1 \leq j \leq k\right\}$.

Note, due to Theorem (A.3.4) a maximizer $\boldsymbol{\mu}^*$ of the dual problem $\sup(\mathcal{D})$ is always a global one.

Theorem A.3.5 (Weak Duality)**.** *Let be* $\boldsymbol{w} \in \mathcal{S}$ *and* $\boldsymbol{\mu} \in \mathbb{R}^k, \boldsymbol{\mu} \geq \mathbf{0}$*, then it always holds*

$$\sup(\mathcal{D}) \leq \inf(\mathcal{P}). \tag{A.18}$$

Proof. Because $\boldsymbol{w}$ and $\boldsymbol{\mu}$ are feasible, we have

$$\begin{aligned} \mathcal{D}L_{\mathcal{S}}(\boldsymbol{\mu}) &= \inf_{\boldsymbol{w} \in \mathbb{R}^L} L(\boldsymbol{w}, \boldsymbol{\mu}) \\ &\leq L(\boldsymbol{w}, \boldsymbol{\mu}) = F(\boldsymbol{w}) + \sum_{j=1}^{k} \mu_j G_j(\boldsymbol{w}) \\ &\leq F(\boldsymbol{w}) \end{aligned}$$

■

Obviously, the question arises under which circumstances equality holds in the inequality of Theorem (A.3.5). That means *strong duality* holds.

A.3.4.2 Strong Duality

Due to Theorem (A.3.5), weak duality holds always. This is not true for *strong duality*. But in case of a convex optimization problem (A.5), it is well-known from optimization theory that indeed $\inf(\mathcal{P}) = \sup(\mathcal{D})$ is attainable:

Theorem A.3.6 (Strong Duality)**.** *Consider the convex optimization problem (A.5). If the feasible set* $\mathcal{S}$ *satisfies Slater's constraint qualification, then the dual problem is solvable and it holds*

$$\inf(\mathcal{P}) = \sup(\mathcal{D}). \tag{A.19}$$

Proof. For a proof see e.g. [Nocedal 1999], [Mangasarian 1969].

■

The *duality gap* $\epsilon_{dual}(\boldsymbol{w}, \boldsymbol{\mu}) := F(\boldsymbol{w}) - \mathcal{D}L_{\mathcal{S}}(\boldsymbol{\mu})$ for a primal feasible point $\boldsymbol{w} \in \mathcal{S}$ and a dual feasible point $\boldsymbol{\mu} \in \mathbb{R}^k, \boldsymbol{\mu} \geq \mathbf{0}$ is given by

Theorem A.3.7 (Duality Gap)**.** *Suppose a convex optimization problem (A.5) with solution $\boldsymbol{w}^* \in \mathcal{S}$. Then for any primal and dual feasible pair $(\hat{\boldsymbol{w}}, \hat{\boldsymbol{\mu}}) \in \mathcal{S} \times [0, \infty)^k$ satisfying $\nabla_{\boldsymbol{w}} L(\hat{\boldsymbol{w}}, \hat{\boldsymbol{\mu}}) = \mathbf{0}$ it holds*

$$\epsilon_{dual}(\hat{\boldsymbol{w}}, \hat{\boldsymbol{\mu}}) = -\sum_{j=1}^{k} \hat{\mu}_j G_j(\hat{\boldsymbol{w}}) \geq F(\hat{\boldsymbol{w}}) - F(\boldsymbol{w}^*) \geq 0. \tag{A.20}$$

In particular, the duality gap attains zero at $(\boldsymbol{w}^*, \boldsymbol{\mu}^*)$ if strong duality holds.

Proof. Due to theorem (A.3.1) for a saddle point $(\boldsymbol{w}^*, \boldsymbol{\mu}^*)$ we have for any $\boldsymbol{w} \in \mathcal{S}, \boldsymbol{\mu} \geq \mathbf{0}$,

$$F(\boldsymbol{w}) \geq F(\boldsymbol{w}^*) = L(\boldsymbol{w}^*, \boldsymbol{\mu}^*) \geq L(\boldsymbol{w}^*, \boldsymbol{\mu}) \geq \inf_{\boldsymbol{w} \in \mathbb{R}^L} L(\boldsymbol{w}, \boldsymbol{\mu}) = \mathcal{D}L_{\mathcal{S}}(\boldsymbol{\mu}). \tag{A.21}$$

Because $L(\cdot, \boldsymbol{\mu})$ is convex for all $\boldsymbol{\mu} \geq \mathbf{0}$, it follows that any $\hat{\boldsymbol{w}} \in \mathcal{S}$ is a minimizer if $\nabla_{\boldsymbol{w}} L(\hat{\boldsymbol{w}}, \hat{\boldsymbol{\mu}}) = \mathbf{0}$ for some $\hat{\boldsymbol{\mu}} \geq \mathbf{0}$ is satisfied, i.e. it holds $L(\hat{\boldsymbol{w}}, \hat{\boldsymbol{\mu}}) = F(\hat{\boldsymbol{w}}) + \sum_{j=1}^{k} \hat{\mu}_j G_j(\hat{\boldsymbol{w}}) = \inf_{\boldsymbol{w} \in \mathbb{R}^L} L(\boldsymbol{w}, \hat{\boldsymbol{\mu}})$. Together with inequality (A.21) we get $0 \leq F(\hat{\boldsymbol{w}}) - F(\boldsymbol{w}^*) \leq -\sum_{j=1}^{k} \hat{\mu}_j G_j(\hat{\boldsymbol{w}}) = F(\hat{\boldsymbol{w}}) - \mathcal{D}L_{\mathcal{S}}(\hat{\boldsymbol{\mu}}) = \epsilon_{dual}(\hat{\boldsymbol{w}}, \hat{\boldsymbol{\mu}})$. In turn, this implies that the gap vanishes if and only if strong duality holds at $\hat{\boldsymbol{w}} = \boldsymbol{w}^*$ and $\hat{\boldsymbol{\mu}} = \boldsymbol{\mu}^*$.

∎

Appendix B

Appendix

B.1 Addendum to Section (6.1)

B.1.1 Equivalence of the Primal and Dual Feasible Optimal Points

Consider the Lagrange function (6.7) and the associated KKT-system (6.8)-(6.16), then it follows

$$\begin{aligned} L^P(\boldsymbol{x},\boldsymbol{y},\boldsymbol{s},\boldsymbol{\kappa},\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\nu},\boldsymbol{\xi}) &= \frac{1}{2}\boldsymbol{y}^T\mathbf{Q}^T\boldsymbol{y} + \boldsymbol{d}^T\boldsymbol{x} + \boldsymbol{\kappa}^T\boldsymbol{x} - \boldsymbol{\kappa}^T\boldsymbol{u}_b + \boldsymbol{\kappa}^T\boldsymbol{s} + \\ &\quad \boldsymbol{\lambda}^T\mathbf{A}\boldsymbol{x} + \boldsymbol{\lambda}^T\boldsymbol{b} + \boldsymbol{\mu}^T\mathbf{Q}\boldsymbol{y} + \boldsymbol{\mu}^T\mathbf{B}\boldsymbol{x} - \boldsymbol{\nu}^T\boldsymbol{x} - \boldsymbol{\xi}^T\boldsymbol{s} \\ &= \frac{1}{2}\boldsymbol{y}^T\mathbf{Q}^T\boldsymbol{y} + \boldsymbol{x}^T\Bigg(\underbrace{\boldsymbol{d} + \boldsymbol{\kappa} + \mathbf{A}^T\boldsymbol{\lambda} + \mathbf{B}^T\boldsymbol{\mu} - \boldsymbol{\nu}}_{=\mathbf{0}\ \text{(cf. 6.8)}}\Bigg) - \\ &\quad \underbrace{\boldsymbol{\kappa}^T\boldsymbol{u}_b}_{=\boldsymbol{\xi}^T\boldsymbol{u}_b\ \text{s. (cf. 6.10)}} + \underbrace{\boldsymbol{\kappa}^T\boldsymbol{s}}_{=\boldsymbol{\xi}^T\boldsymbol{s}\ \text{(cf. 6.10)}} + \boldsymbol{\lambda}^T\boldsymbol{b} + \\ &\quad \underbrace{\boldsymbol{\mu}^T\mathbf{Q}\boldsymbol{y}}_{=-\boldsymbol{y}^T\mathbf{Q}\boldsymbol{y}\ \text{(cf. 6.9)}} - \boldsymbol{\xi}^T\boldsymbol{s} \end{aligned} \tag{B.1}$$

and the dual problem reads as

$$\begin{aligned} &\min_{\boldsymbol{y},\boldsymbol{\lambda},\boldsymbol{\nu},\boldsymbol{\xi}} \frac{1}{2}\boldsymbol{y}^T\mathbf{Q}\boldsymbol{y} - \boldsymbol{\lambda}^T\boldsymbol{b} + \boldsymbol{\xi}^T\boldsymbol{u}_b \\ \text{u.d.N.} \quad &\mathbf{A}^T\boldsymbol{\lambda} - \mathbf{B}^T\boldsymbol{y} + \boldsymbol{\xi} - \boldsymbol{\nu} + \boldsymbol{d} = \mathbf{0} \\ &\boldsymbol{\nu} \geq \mathbf{0},\ \boldsymbol{\xi} \geq \mathbf{0}. \end{aligned} \tag{B.2}$$

Defining the associated dual Lagrange function

$$\begin{aligned} L^D(\boldsymbol{y},\boldsymbol{\lambda},\boldsymbol{\nu},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}) := \frac{1}{2}\boldsymbol{y}^T\mathbf{Q}\boldsymbol{y} - \boldsymbol{\lambda}^T\boldsymbol{b} + \boldsymbol{\xi}^T\boldsymbol{u}_b + \\ \boldsymbol{\alpha}^T\left(\mathbf{A}^T\boldsymbol{\lambda} - \mathbf{B}^T\boldsymbol{y} + \boldsymbol{\xi} - \boldsymbol{\nu} + \boldsymbol{d}\right) - \boldsymbol{\beta}^T\boldsymbol{\nu} - \boldsymbol{\gamma}^T\boldsymbol{\xi} \end{aligned} \tag{B.3}$$

with Lagrange multipliers $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n$, the dual KKT-system read as

$$\begin{aligned}
\nabla_{\boldsymbol{y}} L^D(\boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{\nu}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) &= \mathbf{Q}\boldsymbol{y} - \mathbf{B}\boldsymbol{\alpha} = \mathbf{0} && \text{(B.4)}\\
\nabla_{\boldsymbol{\lambda}} L^D(\boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{\nu}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) &= -\boldsymbol{b} + \mathbf{A}\boldsymbol{\alpha} = \mathbf{0} && \text{(B.5)}\\
\nabla_{\boldsymbol{\xi}} L^D(\boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{\nu}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) &= \boldsymbol{u}_b + \boldsymbol{\alpha} - \boldsymbol{\gamma} = \mathbf{0} && \text{(B.6)}\\
& \mathbf{A}^T\boldsymbol{\lambda} - \mathbf{B}^T\boldsymbol{y} + \boldsymbol{\xi} - \boldsymbol{\nu} + \boldsymbol{d} = \mathbf{0} && \text{(B.7)}\\
& \boldsymbol{\nu} \geq \mathbf{0},\ \boldsymbol{\xi} \geq \mathbf{0} && \text{(B.8)}\\
& \boldsymbol{\beta}^T\boldsymbol{\nu} = 0,\ \boldsymbol{\gamma}^T\boldsymbol{\xi} = 0 && \text{(B.9)}\\
& \boldsymbol{\beta} \geq \mathbf{0},\ \boldsymbol{\gamma} \geq \mathbf{0}. && \text{(B.10)}
\end{aligned}$$

Setting $\boldsymbol{\beta} = \boldsymbol{x},\ \boldsymbol{\gamma} = \boldsymbol{s},\ \boldsymbol{\alpha} = -\boldsymbol{x}$ one easily verifies that the dual KKT-system is identical to the primal KKT-system. Thus, a primal feasible optimal point is also a dual feasible optimal point. Moreover, in virtue of the strong duality theorem (App. A.3.4.2) the optimal values coincide too.

B.2 Addendum to Section (6.2)

B.2.1 Proof of Lemma (6.2.1)

Proof. Let be $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2 \in \mathcal{T}$ and $\epsilon > 0$. Then it exist $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2 \in \mathcal{S}$ such that $G(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i) \leq g(\boldsymbol{\mu}_i) + \epsilon$ for $i = 1, 2$. Let be $\lambda \in [0, 1]$, it follows by *Jensen's inequality* (App. (A.1.2)) that

$$\begin{aligned}
g(\lambda\boldsymbol{\mu}_1 + (1-\lambda)\boldsymbol{\mu}_2) &= \inf\{G(\boldsymbol{\alpha}, \lambda\boldsymbol{\mu}_1 + (1-\lambda)\boldsymbol{\mu}_2) : \boldsymbol{\alpha} \in \mathcal{S}\} && \text{(B.11)}\\
&\leq G(\lambda\boldsymbol{\alpha}_1 + (1-\lambda)\boldsymbol{\alpha}_2, \lambda\boldsymbol{\mu}_1 + (1-\lambda)\boldsymbol{\mu}_2) && \text{(B.12)}\\
&\leq \lambda G(\boldsymbol{\alpha}_1, \boldsymbol{\mu}_1) + (1-\lambda)G(\boldsymbol{\alpha}_2, \boldsymbol{\mu}_2) && \text{(B.13)}\\
&\leq \lambda g(\boldsymbol{\mu}_1) + (1-\lambda)g(\boldsymbol{\mu}_2) + \epsilon\ \forall \epsilon > 0. && \text{(B.14)}
\end{aligned}$$

Thus, it follows for $\epsilon \to 0^+$

$$g(\lambda\boldsymbol{\mu}_1 + (1-\lambda)\boldsymbol{\mu}_2) \leq \lambda g(\boldsymbol{\mu}_1) + (1-\lambda)g(\boldsymbol{\mu}_2)\ \forall \boldsymbol{\mu}_1, \boldsymbol{\mu}_2 \in \mathcal{T}. \quad \text{(B.15)}$$

■

B.2.2 Proof of Lemma (6.2.2)

Proof. Let be $0 < \lambda_1 < \lambda_2$ with $\boldsymbol{\mu} + \lambda_2\boldsymbol{d} \in \mathcal{T}$ (implying $\boldsymbol{\mu} + \lambda_1\boldsymbol{d} \in \mathcal{T}$). Because g is convex, it holds

$$\begin{aligned}
g(\boldsymbol{\mu} + \lambda_1\boldsymbol{d}) &= g\left(\frac{\lambda_1}{\lambda_2}(\boldsymbol{\mu} + \lambda_2\boldsymbol{d}) + \left(1 - \frac{\lambda_1}{\lambda_2}\right)\boldsymbol{\mu}\right) && \text{(B.16)}\\
&\leq \frac{\lambda_1}{\lambda_2} g(\boldsymbol{\mu} + \lambda_2\boldsymbol{d}) + \left(1 - \frac{\lambda_1}{\lambda_2}\right) g(\boldsymbol{\mu}). && \text{(B.17)}
\end{aligned}$$

This inequality implies

$$\frac{g(\boldsymbol{\mu}+\lambda_1\boldsymbol{d})-g(\boldsymbol{\mu})}{\lambda_1} \leq \frac{g(\boldsymbol{\mu}+\lambda_2\boldsymbol{d})-g(\boldsymbol{\mu})}{\lambda_2}. \tag{B.18}$$

Now, let be $\lambda, \tau > 0$ with $\boldsymbol{\mu}+\lambda\boldsymbol{d} \in \mathcal{T}$ and $\boldsymbol{\mu}-\tau\boldsymbol{d} \in \mathcal{T}$. Again, convexity of g implies

$$\begin{aligned} g(\boldsymbol{\mu}) &= g\left(\frac{\lambda}{\lambda+\tau}(\boldsymbol{\mu}-\tau\boldsymbol{d})+\frac{\tau}{\lambda+\tau}(\boldsymbol{\mu}+\lambda\boldsymbol{d})\right) && \text{(B.19)} \\ &\leq \frac{\lambda}{\lambda+\tau}g(\boldsymbol{\mu}-\tau\boldsymbol{d})+\frac{\tau}{\lambda+\tau}g(\boldsymbol{\mu}+\lambda\boldsymbol{d}), && \text{(B.20)} \end{aligned}$$

thus we have

$$q(\lambda) := \frac{g(\boldsymbol{\mu}+\lambda\boldsymbol{d})-g(\boldsymbol{\mu})}{\lambda} \geq \frac{g(\boldsymbol{\mu})-g(\boldsymbol{\mu}-\tau\boldsymbol{d})}{\tau}. \tag{B.21}$$

This means for $\lambda \to 0^+$ the differential quotient $q(\lambda)$ is bounded from below by the right-hand side of (B.21). Additionally, inequality (B.18) implies $q(\lambda_1) \leq q(\lambda_2),\ \lambda_1 < \lambda_2$ which means that $q(\lambda)$ is monotonically decreasing for $\lambda \to 0^+$. Together, this implies the existence of the directional derivative $g'(\boldsymbol{\mu};\boldsymbol{d})$ with the property

$$g'(\boldsymbol{\mu};\boldsymbol{d}) := \lim_{\lambda\to 0^+} q(\lambda) = \inf_{\lambda>0} q(\lambda). \tag{B.22}$$

■

B.2.3 Proof of Lemma (6.2.3)

Proof. Due to the continuity of G, it follows G is uniformly continuous for any $\boldsymbol{\mu}_0 \in \mathcal{T}$ on a compact set $\mathcal{S} \times \bar{\mathcal{U}}_\epsilon(\boldsymbol{\mu}_0)$(*Heine's Theorem*), $\epsilon > 0$, with a compact ϵ-neighborhood $\bar{\mathcal{U}}_\epsilon(\boldsymbol{\mu}_0)$ of $\boldsymbol{\mu}_0$. That means, for any sequence $(\boldsymbol{\mu}_n)_{n\in\mathbb{N}} \in \bar{\mathcal{U}}_\epsilon(\boldsymbol{\mu}_0) \overset{n\to\infty}{\longrightarrow} \boldsymbol{\mu}_0$ it exists for all $\epsilon > 0$ an index $n_0 \in \mathbb{N}$ such that

$$\forall n > n_0,\ \boldsymbol{\alpha} \in \mathcal{S} : |G(\boldsymbol{\alpha},\boldsymbol{\mu}_n) - G(\boldsymbol{\alpha},\boldsymbol{\mu}_0)| < \epsilon. \tag{B.23}$$

Thus, it holds

$$\forall n > n_0,\ \boldsymbol{\alpha} \in \mathcal{S} : -\epsilon + G(\boldsymbol{\alpha},\boldsymbol{\mu}_0) < G(\boldsymbol{\alpha},\boldsymbol{\mu}_n) < G(\boldsymbol{\alpha},\boldsymbol{\mu}_0) + \epsilon \tag{B.24}$$

implying by definition of $g(\boldsymbol{\mu})$ that

$$\forall n > n_0 : -\epsilon + g(\boldsymbol{\mu}_0) \leq g(\boldsymbol{\mu}_n) \leq g(\boldsymbol{\mu}_0) + \epsilon. \tag{B.25}$$

This is equivalent with

$$\forall n > n_0 : |g(\boldsymbol{\mu}_n) - g(\boldsymbol{\mu}_0)| \leq \epsilon \tag{B.26}$$

that means continuity of g for all $\boldsymbol{\mu} \in \mathcal{T}$.

■

B.3 Addendum to Section (6.3.1) and Section (6.3.2)

B.3.1 Some Special Implementation Notes

First, the search for a new candidate matrix $\boldsymbol{K}^*$ introduces an expensive quadratic problem $q_r(\boldsymbol{J}_{\mu_0}(\boldsymbol{x}))$ in each trial step. Hence, the search is more efficient, if a lower bound on the optimal value $q_r(\boldsymbol{J}_{\mu_0}(\boldsymbol{x}))$ is available and inexpensive to compute. Unfortunately, inequality (6.116) is contradicted in general for the non-regularized $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm. The reason is that $\boldsymbol{\alpha}^*_{(t)} \in \mathcal{S}(\mathbf{V}^{\mathbf{K}_{(t)}}_{r+1})$ might not be in the feasible set $\mathcal{S}(\mathbf{V}^{\mathbf{K}}_{r+1})$ for a matrix $\mathbf{K} \in conv(\mathcal{K})$. However, one could project the point $\boldsymbol{\alpha}^*_{(t)}$ onto the feasible set $\mathcal{S}(\mathbf{V}^{\mathbf{K}}_{r+1})$. This requires a solution of a different quadratic programming problem, which might be easier to solve than problem $q_r(\boldsymbol{J}_{\mu_0}(\boldsymbol{x}))$. Alternatively, one could guide the search if projecting $\boldsymbol{\alpha}^*_{(t)}$ to a point $\boldsymbol{\beta}^* \in \mathcal{S}(\mathbf{V}^{\mathbf{K}}_{r+1})$ such that $\|\boldsymbol{\alpha}^* - \boldsymbol{\beta}^*\| \leq \epsilon$ holds for sufficiently small $\epsilon > 0$, then one could reject a possible candidate matrix $\boldsymbol{K}$ if the associated inequality $Q_r(\boldsymbol{\beta}^*, d^*_r, \boldsymbol{J}) \leq Q_r(\boldsymbol{\alpha}^*, d^*_{(t)}, \mathbf{K}_{(t)})$ is satisfied.

Second, if the rank r of the matrix $\mathbf{K}_{\mu_0}(\boldsymbol{x})$ has reached $r = M$, then no further eigenvalue decomposition is necessary in subsequent iterations. In this case, one could terminate the non-regularized $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithm and it would be wise to continue with the regularized algorithm using a very small $\epsilon > 0$ for the remaining iterations. A further issue concerning the rank of $\mathbf{K}_{\mu_0}(\boldsymbol{x})$ is that r is bounded from above due to Lemma (6.3.1). Thus, a truncated eigenvalue decomposition can be used with same precision than a full decomposition.

Third, instead of running the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithms for a predefined maximum number of steps, one can (additionally) use a termination criterion obtained from Theorem (A.3.7) measuring the gap between the primal and dual optimal value: Let be $\boldsymbol{x}_{(t)} \in \mathcal{X}$ with candidate matrix $\mathbf{K}^* = \mathbf{K}(\boldsymbol{x}_{(t)})$ found in iteration t. Further, let be $\lambda_{(t)} \in \mathbb{R}$ with $\sum_t \lambda_{(t)} = 1$ the associated optimal weights of the matrix $\mathbf{K} = \sum_t \lambda_{(t)} \mathbf{K}(\boldsymbol{x}_{(t)}) \in conv(\mathcal{K})$. From the constraints of the feasible set of the primal SIP (5.78) it follows the duality gap

$$\begin{aligned} \epsilon_{dual}(\boldsymbol{z}_0, \boldsymbol{c}, \boldsymbol{\xi}, b, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\lambda}) = \quad & \boldsymbol{\alpha}^T \left(\mathbf{Y}\mathbf{G}\boldsymbol{c} + \boldsymbol{y}b - \mathbf{1}_N + \boldsymbol{\xi} \right) + \boldsymbol{\beta}^T \boldsymbol{\xi} \\ & - \sum_t \lambda_{(t)} \left(\frac{1}{2} \boldsymbol{c}^T \mathbf{K}(\boldsymbol{x}_{(t)}) \boldsymbol{c} - z_0 \right). \end{aligned} \tag{B.27}$$

The Lagrange function of the primal SIP (5.78) reads as

$$\begin{aligned} L(z_0, \boldsymbol{c}, \boldsymbol{\xi}, b, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\lambda}) = \quad & z_0 + C\mathbf{1}_N^T\boldsymbol{\xi} + \boldsymbol{\alpha}^T\left(-\mathbf{Y}\mathbf{G}\boldsymbol{c} - \boldsymbol{y}b + \mathbf{1}_N - \boldsymbol{\xi}\right) + \boldsymbol{\beta}^T(-\boldsymbol{\xi}) \\ & + \sum_t \lambda_{(t)}\left(\frac{1}{2}\boldsymbol{c}^T\mathbf{K}(\boldsymbol{x}_{(t)})\boldsymbol{c} - z_0\right) \end{aligned} \tag{B.28}$$

The dual variables $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\lambda})$ obtained in each iteration of the $\mathcal{C}^{(1)}(\mathcal{X}, \mathbb{R})$-Lipschitz classifier algorithms determine $(\boldsymbol{c}, b)$ such that

$$\nabla_{(\boldsymbol{c},b)} L(z_0, \boldsymbol{c}, \boldsymbol{\xi}, b, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\lambda}) = \mathbf{0}$$

is satisfied. Therefore, we can bound the duality gap (B.27) provided by Theorem (A.3.7) if we choose bounds on feasible values of $\boldsymbol{\xi}$ and z_0 satisfying $\nabla_{(\boldsymbol{\xi},z_0)} L(z_0, \boldsymbol{c}, \boldsymbol{\xi}, b, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\lambda}) = \mathbf{0}$ at the same time.

Rewriting the duality gap (B.27) and using

$$\nabla_{\boldsymbol{\xi}} L(z_0, \boldsymbol{c}, \boldsymbol{\xi}, b, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\lambda}) = C\mathbf{1}_N^T - \boldsymbol{\alpha} - \boldsymbol{\beta} = \mathbf{0} \tag{B.29}$$

$$\nabla_{\boldsymbol{\lambda}} L(z_0, \boldsymbol{c}, \boldsymbol{\xi}, b, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\lambda}) = 1 - \sum_t \lambda_{(t)} = 0 \tag{B.30}$$

we get

$$\begin{aligned} \epsilon_{dual}(\boldsymbol{z}_0, \boldsymbol{c}, \boldsymbol{\xi}, b, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\lambda}) &= \boldsymbol{\alpha}^T\left(\mathbf{Y}\mathbf{G}\boldsymbol{c} + \boldsymbol{y}b - \mathbf{1}_N\right) + (\boldsymbol{\beta} + \boldsymbol{\alpha})^T\boldsymbol{\xi} \\ &\quad - \sum_t \lambda_{(t)}\left(\frac{1}{2}\boldsymbol{c}^T\mathbf{K}(\boldsymbol{x}_{(t)})\boldsymbol{c}\right) + \left(\sum_t \lambda_{(t)}\right) z_0 \\ &= \boldsymbol{\alpha}^T\left(\mathbf{Y}\mathbf{G}\boldsymbol{c} + \boldsymbol{y}b - \mathbf{1}_N\right) + C\mathbf{1}_N^T\boldsymbol{\xi} \\ &\quad - \sum_t \lambda_{(t)}\left(\frac{1}{2}\boldsymbol{c}^T\mathbf{K}(\boldsymbol{x}_{(t)})\boldsymbol{c}\right) + z_0 \\ &\geq \boldsymbol{\alpha}^T\left(\mathbf{Y}\mathbf{G}\boldsymbol{c} + \boldsymbol{y}b - \mathbf{1}_N\right) + C\mathbf{1}_N^T\tilde{\boldsymbol{\xi}} \\ &\quad - \sum_t \lambda_{(t)}\left(\frac{1}{2}\boldsymbol{c}^T\mathbf{K}(\boldsymbol{x}_{(t)})\boldsymbol{c}\right) + \tilde{z}_0 \\ &\geq 0 \end{aligned} \tag{B.31}$$

with

$$\tilde{\xi}_i = \min\left\{\xi_i \geq 0 : 0 \geq -y_i f(\boldsymbol{x}_i) + 1 - \xi\right\} = \max\left\{0, 1 - y_i f(\boldsymbol{x}_i)\right\} \tag{B.32}$$

and

$$\tilde{z}_0 = \min\left\{z_0 : 0 \geq \frac{1}{2}\boldsymbol{c}^T\mathbf{K}(\boldsymbol{x}_{(t)})\boldsymbol{c} - z_0 \forall t\right\} = \max_t\left\{\frac{1}{2}\boldsymbol{c}^T\mathbf{K}(\boldsymbol{x}_{(t)})\boldsymbol{c}\right\}. \tag{B.33}$$

Note, all candidate matrices $\mathbf{K}(\boldsymbol{x}_{(t)})$ found by the algorithm up to the considered iteration have to be available in order to compute $\tilde{z}_0$.

Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Insbesondere habe ich nicht die Hilfe einer kommerziellen Promotionsberatung in Anspruch genommen. Dritte haben von mir weder unmittelbar noch mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form als Dissertation eingereicht und ist als Ganzes auch noch nicht veröffentlicht.

Düsseldorf, den 06. März 2010

Author's Publications

[Gaida 2008] D. Gaida, A. Stuhlsatz and H.-G. Meier. *Fusion of Visual and Inertial Measurements for Pose Estimation*. In A. Wendemuth and H.-G. Meier, editors, Proc. Research Workshop "Emotion-, Speech- and Face Recognition with advanced classifiers", Magdeburg, Germany, 2008. ISBN: 978-3-940961-24-2.

[Schuller 2009] B. Schuller, B. Vlasenko, F. Eyben, M. Wöllmer, A. Stuhlsatz, A. Wendemuth and G. Rigoll. *Cross-Corpus Acoustic Emotion Recognition: Variances and Strategies*. IEEE Transactions on Affective Computing, submitted Dec. 2009.

[Stuhlsatz 2003] A. Stuhlsatz, H.-G. Meier, M. Katz, S.E. Krüger and A. Wendemuth. *Classification of speech recognition hypotheses with Support Vector Machines*. In Proceedings of the Speech Processing Workshop in connection with DAGM, pages pp. 65–72. University of Magdeburg, ISBN 3-929757-59-1, 2003.

[Stuhlsatz 2006] A. Stuhlsatz, H.-G. Meier, M. Katz, S. E. Krüger and A. Wendemuth. *Support Vector Machines for Postprocessing of Speech Recognition Hypotheses*. In Proceedings of International Conference on Telecommunications and Multimedia (TEMU), Heraklion, Crete, Greece, 2006.

[Stuhlsatz 2007a] A. Stuhlsatz. *HSVM - A SVM Toolkit for Segmented Speech Data*. In Proceedings of Elektronische Sprachsignalverarbeitung (ESSV), volume 46 of *Studientexte zur Sprachkommunikation*, Cottbus, Germany, 2007. TUDpress, ISSN: 0940-6832.

[Stuhlsatz 2007b] A. Stuhlsatz. *Recognition of Ultrasonic Multi-Echo Sequences for Autonomous Symbolic Indoor Tracking*. In Proceedings of 6'th International Conference on Machine Learning and Applications (ICMLA '07), pages 178–185, Cincinnatti, OH, USA, 2007. IEEE Computer Society, ISBN: 978-0-7695-3069-7.

[Stuhlsatz 2007c] A. Stuhlsatz, H.-G. Meier and A. Wendemuth. *Maximum Margin Classification on Convex Euclidean Metric Spaces*. In Computer Recognition Systems 2, volume 45 of *Advances in Soft Computing*, pages 216–223. Springer Verlag, ISBN-13 978-3-540-75174-8, 2007.

[Stuhlsatz 2008a] A. Stuhlsatz. Hybride Spracherkennung - Eine HMM/SVM-Systemintegration. VDM Verlag, Fachbuch, ISBN: 978-3-639-10062-4, 2008.

[Stuhlsatz 2008b] A. Stuhlsatz, H.-G. Meier and A. Wendemuth. *A Dual Formulation to the Lipschitz Classifier.* In A. Wendemuth and H.-G. Meier, editors, Proc. Research Workshop "Emotion-, Speech- and Face Recognition with advanced classifiers", Magdeburg, Germany, 2008. ISBN: 978-3-940961-24-2.

[Stuhlsatz 2008c] A. Stuhlsatz, H.-G. Meier and A. Wendemuth. *Making the Lipschitz Classifier Practical via Semi-infinite Programming.* In Proceedings of 7'th International Conference on Machine Learning and Applications (ICMLA '07), pages 40–47, San Diego, CA, USA, 2008. IEEE Computer Society, ISBN: 978-0-7695-3495-4. Best Paper Award Winner.

References

[Andelic 2006] 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 ICPR 2006: 18th International Conference on Pattern Recognition, Hong Kong, 2006.

[Asuncion 2007] A. Asuncion and D.J. Newman. *UCI Machine Learning Repository.* University of California, Irvine, School of Information and Computer Sciences, 2007. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[Bahlmann 2002] C. Bahlmann, B.Haasdonk and H. Burkhardt. *On-line Handwriting Recognition with Support Vector Machines - A Kernel Approach.* In Proc. of the 8th IWFHR, pages 49–54, 2002.

[Bennett 1992] K. P. Bennett and O. L. Mangasarian. *Robust linear programming discriminantion of two linearly inseparable sets.* Optimization Methods and Software, vol. 1, pages 23–34, 1992.

[Birgin 2000] E. G. Birgin, J. M. Martinez and M. Raydan. *Nonmonotone Spectral Projected Gradient Methods on Convex Sets.* SIAM Journal on Optimization, vol. 10, pages 1196–1211, 2000.

[Birgin 2009] E. G. Birgin, J. M. Martinez and M. Raydan. *Spectral Projected Gradient Methods.* Encyclopedia of Optimization, pages 3652–3659, 2009.

[Bishop 2006] C. M. Bishop. Pattern recognition and machine learning. Springer Verlag, 2006.

[Boyd 2004] S. Boyd and L. Vandenberghe. Convex optimization. Cambridge University Press, 2004. available online at www.stanford.edu/boyd/cvxbook/bv_cvxbook.pdf.

[Bromley 1991] J. Bromley and E. Säckinger. *Neural-network and k-nearest-neighbour classifiers.* Rapport technique 11359-910819-16TM, AT&T, 1991.

[Brown 1999] M. Brown, W. Grundy, D. Lin, N. Cristianini, C. Sugnet, M. Ares Jr. and D. Haussler. *Support Vector Machine Classification of Microarray Gene Expression Data.* Rapport technique UCSC-CRL-99-09, University of California, Santa Cruz, 1999.

[Burges 1998] Ch. C. J. Burges. *A Tutorial on Support Vector Machines for Pattern Recognition*. Data Mining and Knowledge Discovery, vol. 2, pages 121–167, 1998.

[Campbell 2003] W. M. Campbell. *A SVM/HMM system for speaker recognition*. In IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), volume 2, 2003.

[Cha 2006] S.-H. Cha, C. Tappert and S. Yoon. *Enhancing binary feature vector similarity measures*. Journal of Pattern Recognition Research, vol. 1, pages 63–77, 2006.

[Chen 2009] Y. Chen, E. K. Garcia, M. R. Grupta, A. Rahimi and L. Cazzanti. *Similarity-Based Classification: Concepts and Algorithms*. Journal of Machine Learning and Research, vol. 10, pages 747–776, 2009.

[Courant 1953] R. Courant and D. Hilbert. Methods of mathematical physics. J. Wiley, New York, 1953.

[DeCoste 2002] D. DeCoste and B. Schölkopf. *Training invariant support vector machines*. Machine Learning, vol. 46(1), pages 161–190, 2002.

[Dennis 1977] J. E. Dennis and J. J. More. *Quasi-Newton methods, motivation and theory*. SIAM Review, vol. 19, pages 46–89, 1977.

[Duda 1973] R. O. Duda, P. E. Hart and D. G. Stork. Pattern classification and scene analysis. John Wiley & Sons, Inc., 1973.

[Duda 2000] R. O. Duda, P. E. Hart and D. G. Stork. Pattern classificatiom. John Wiley & Sons, Inc., 2000.

[Fiacco 1968] A. Fiacco and M. McCormick. Nonlinear programming: Sequential unconstrained minimization techniques. John Wiley & Sons, Inc., 1968.

[Frisch 1955] K.R. Frisch. The logarithmic potential method of convex programming. University Institute of Economics, 1955.

[Fukunaga 1972] K. Fukunaga. Introduction to statistical pattern recognition. Academic Press, 1972.

[Fukunaga 1990] K. Fukunaga. Statistical pattern recognition. Academic Press, 1990.

[Fukushima 1975] K. Fukushima. *Cognitron: A Self-organizing Multilayered Neural Network*. Biological Cybernetics, vol. 20, pages 121–136, 1975.

[Gallant 1990] S. I. Gallant. *Perceptron-based learning algorithms.* IEEE Transactions on Neural Networks, vol. 1 (2), pages 179–191, 1990.

[Ganapathiraju 2000] A. Ganapathiraju, J. Hamaker and J. Picone. *Hybrid SVM/HMM architectures for speech recognition.* In Speech Transcription Workshop, 2000.

[Geman 1984] S. Geman and D. Geman. *Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images.* IEEE Trans. Pattern Anal. Mach. Intell., vol. 6, pages 721–741, 1984.

[Ghent 2005] J. Ghent and J. McDonald. *Facial Expression Classification using One-Against-All Support Vector Machine.* Proceedings of the Irish Machine Vision and Image Processing Conference, 2005.

[Gill 1986] P. E. Gill, W. Murray, M. A. Saunders, J. A. Tomlin and M. H. Wright. *On projected netwon barrier methods for linear programming and an equivalence to Karmarkar's projective method.* Mathematical Programming, vol. 36, pages 183–209, 1986.

[Goberna 2001] M. A. Goberna and M. A. Lopez, editors. Semi-infinite programming: Recent advances, volume 57 of *Nonconvex Optimization and Its Applications.* Kluwer Academic Publishers, 2001.

[Golub 1996] G. H. Golub and Ch. F. Van Loan. Matrix computations. The Johns Hopkins University Press, 1996.

[Graepel 1999] T. Graepel, R. Herbich, B. Schölkopf, A. Smola, P. Bartlett, K.R. Müller, K. Obermayer and R. Williamson. *Classification on proximity data with LP-machines.* International Conference on Artificial Neural Networks, pages 304–309, 1999.

[Gurban 2005] M. Gurban and J. Thiran. *Audio-Visual Speech Recognition with a Hybrid SVM-HMM System.* In Hermes, Collection Informatique. EUSIPCO, 2005.

[Haasdonk 2002] B. Haasdonk. *Tangent Distance Kernels for Support Vector Machines.* In Proc. of the 16th Int. Conf. on Pattern Recognition (ICPR), volume 2, pages 864–868, 2002.

[Hastie 2001] T. Hastie, R. Tibshirani and J. Friedman. The elements of statistical learning; data mining, inference and prediction. Springer Verlag, 2001.

[Haussler 1999] David Haussler. *Convolution Kernels on Discrete Structures.* Rapport technique UCSC-CRL-99-10, Department of Computer Sciences, University of California at Santa Cruz, 1999.

[Hein 2004] M. Hein and O. Bousquet. *Maximal Margin Classification for Metric Spaces.* Learning Theory and Kernel Machines, pages 72–86, 2004.

[Hettich 1993] R. Hettich and K. O. Kortanek. *Semi-infinite Programming: Theory, methods and applications.* SIAM Review, vol. 35, pages 380–429, 1993.

[Hoeffding 1963] W. Hoeffding. *Probability inequalities for sums of bounded random variables.* Journal of the American Statistical Association, vol. 58, pages 13–30, 1963.

[Hopfield 1982] J. J. Hopfield. *Neural Networks and Physical Systems with Emergent Collective Computational Abilities.* In Proceedings of the National Academy of Sciences, volume 79, pages 2554–2558, 1982.

[Iplikci 2006] S. Iplikci. *Support vector machines-based generalized predictive control.* International Journal of Robust and Nonlinear Control, vol. 6, pages 843–862, 2006.

[Jaakkola 1999] T. S. Jaakkola and D. Haussler. *Exploiting generative models in discriminant classifiers.* In Proceedings of the 1999 Confernece on AI and Statistics, 1999.

[Jensen 1906] J. L. W. V. Jensen. *Sur les fonctions convexes et les inégalités entre les valeurs moyennes.* In Acta Math., volume 30, pages 175–193, 1906.

[Joachims 1998] T. Joachims. *Text categorization with support vector machines: Learning with many relevant features.* In I. Bratko and S. Dzeroski, editors, Proceedings of the 16th International Conference on Machine Learning, pages 200–209, San Francisco, 1998.

[Karmarkar 1984] N. Karmarkar. *A new polynomial time algorithm for linear programming.* Combinatorica, vol. 4, pages 373–395, 1984.

[Karush 1939] W. Karush. *Minima of Functions of Several Variables with Inequalities as Side Constraints.* PhD thesis, Dept. of Mathematics, Univ. of Chicago, Chicago, Illinois, 1939.

[Keysers 2000] D. Keysers, J. Dahmen, T. Theiner and H. Ney. *Experiments with an extended tangent distance.* In Proceedings 15th International Conference on Pattern Recognition, 2, pages 38–42. IEEE Computer Society, 2000.

[Kim 2002] K. I. Kim, K. Jung, S. H. Park and H. J. Kim. *Support Vector Machines for Texture Classification.* IEEE Transactions on Pattern Analysis and Machine Learning, vol. 24, 2002.

[Kimeldorf 1971] G. S. Kimeldorf and G. Wahba. *Some results on Tchebycheffian spline functions.* Journal of Mathematical Analysis and Applications, vol. 33, pages 82–95, 1971.

[Kirkpatrick 1983] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi. *Optimization by Simulated Annealing.* Science, vol. 220, pages 671–680, 1983.

[Kohonen 1972] T. Kohonen. *Correlation Matrix Memories.* IEEE Transactions on Computers, vol. 21, pages 353–359, 1972.

[Koninck 2007] P. De Koninck. *http://www.greenspine.ca/en/framed.html.* online, 2007. University of Laval.

[Krauth 1987] W. Krauth and M. Mezard. *Learning algorithms with optimal stability in neural networks.* Journal of Physics A: Mathematical and General, vol. 20, pages L745–L752, 1987.

[Krüger 2005a] S. E. Krüger, M. Schafföner, M. Katz, E. Andelic and A. Wendemuth. *Speech Recognition with Support Vector Machines in a Hybrid System.* In Isabel Trancoso, editors, Proceeding of Eurospeech/Interspeech 2005, 9th European Conference on Speech Communication and Technology, pages pp. 993–996. Causal Productions Pty Ltd. ISSN: 1018-4074 CDROM, 2005.

[Krüger 2005b] S. E. Krüger, M. Schafföner, M. Katz, E. Andelic and A. Wendemuth:. *Using Support Vector Machines in a HMM based Speech Recognition System.* In George Kokkinakis, editors, Specom 2005. Proceedings of 10th International Conference on Speech and Computer, volume 1, pages pp. 329–331. University of Patras Press, ISBN 5-7452-0110-x, 2005.

[Krüger 2006] S. E. Krüger, M. Schafföner, M. Katz, E. Andelic and A. Wendemuth. *Mixture of Support Vector Machines for HMM based Speech Recognition.* In Proceedings of ICPR 2006: 18th International Conference on Pattern Recognition, Hong Kong, 2006.

[Krüger 2007] S. E. Krüger, M. Schafföner, M. Katz, E. Andelic and A. Wendemuth. *Support Vector Machines as Acoustic Models in Speech Recognition.* In Proceedings of DAGA 2007. 33rd German Annual Conference on Acoustics, Stuttgart, Germany, 2007.

[Kuhn 1951] H. W. Kuhn and A. W. Tucker. *Nonlinear programming.* In Proceedings of 2nd Berkeley Symposium, 1951.

[LeCun 1985] Y. LeCun. *Une procedure d'apprentissage pour reseau a seuil asymmetrique (a Learning Scheme for Asymmetric Threshold Networks).* In Proceedings of Cognitiva, 1985.

[LeCun 1989] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel. *Backpropagation Applied to Handwritten Zip Code Recognition.* Neural Computation, vol. 1(4), pages 541–551, 1989.

[LeCun 1998] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner. *Gradient-Based Learning Applied to Document Recognition.* Proc. of the IEEE, 1998.

[Lee 2000] Y. Lee, O. L. Mangasarian and W. H. Wolberg. *Breast Cancer Survival and Chemotherapy: A Support Vector Machine Analysis.* DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 55, pages 1–10, 2000.

[Lei 2007] H. Lei and B. Sun. *A Study on the Dynamic Time Warping in Kernel Machines.* In Third International IEEE Conference on Signal-Image Technologies and Internet-Based System, 2007.

[Lin 2003] H.-T. Lin and C.-J. Lin. *A Study on Sigmoid Kernels for SVM and the Training of non-PSD Kernels by SMO-type Methods.* Rapport technique, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 2003. Available at http://www.csie.ntu.edu.tw/ cjlin/papers/tanh.pdf URL http://www.csie.ntu.edu.tw/ cjlin/papers/tanh.pdf.

[Liu 2001] J. S. Liu. Monte carlo strategies in scientific computing. Springer Verlag New York, 2001.

[Liu 2007] J. Liu, Z. Wang and X. Xiao. *A hybrid SVM/DDBHMM decision fusion modeling for robust continuous digital speech recognition.* Pattern Recognition Letters, vol. 28, no. 8, pages 912–920, June 2007.

[Lopez 2007] M. Lopez and G. Still. *Semi-Infinite Programming.* European Journal of Operational Research, vol. 2, pages 491–518, 2007.

[Mangasarian 1969] O. L. Mangasarian. Nonlinear programming. McGraw-Hill New York, 1969.

[Mazanec 2008] J. Mazanec, M. Melisek, M. Oravec and J. Pavlovicova. *Support Vector Machines, PCA and LDA in Face Recognition*. Journal of Electrical Engineering, vol. 59, pages 203–209, 2008.

[Mehrotra 1992] S. Mehrotra. *On the implementation of a primal-dual interior point method*. SIAM Journal of Optimization, vol. 2(4), pages 575–601, 1992.

[Mercer 1909] J. Mercer. *Functions of positive nd negative type and their connection with the theory of integral equations*. Philos. Trans. Roy. Soc., 1909.

[Metropolis 1953] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller. *Equations of state calculations for fast computing machines*. Journal of Chemical Physics, vol. 6, page 1087, 1953.

[Minh 2004] H. Q. Minh and Th. Hofmann. *Learning Over Compact Metric Spaces*. Lecture Notes in Computer Science, Springer Verlag, vol. 3120, pages 239–254, 2004.

[Monteiro 1989a] R. D. C. Monteiro and I. Adler. *Interior Path Following Primal-Dual Algorithms. Part I: Linear Programming*. Mathematical Programming, vol. 44, pages 27–41, 1989.

[Monteiro 1989b] R. D. C. Monteiro and I. Adler. *Interior Path Following Primal-Dual Algorithms. Part II: Convex Quadratic Programming*. Mathematical Programming, vol. 44, pages 43–66, 1989.

[Moreno 2003] P. J. Moreno, P. P. Ho and N. Vasconcelos. *A Kullback-Leibler divergence based kernel for SVM Classification in Multimedia Applications*. Advances in Neural Information Processing Systems, vol. 16, 2003.

[Nocedal 1999] J. Nocedal and S. J. Wright. Numerical optimization. Springer Verlag New York, 1999.

[Oliver 2000] N. Oliver, B. Schölkopf and A. J. Smola. *Natural regularization in SVMs*. In A. J. Smola, P. L. Bartlett, B. Schölkopf and D. Schuurmans, editors, Advances in Large Margin Classifiers, pages 51–60. MIT Press, 2000.

[Osuna 1997] E. Osuna, R. Freund and F. Girosi. *Training Support Vector Machines: an Application to Face Detection*. Proceedings of CVPR'97, 1997.

[Parker 1985] D. B. Parker. *Learning-logic*. Rapport technique, Center for Comp. Research in Economics and Managment Sci., MIT, 1985.

[Rosenblatt 1958] F. Rosenblatt. *The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain*. Psych. Rev., vol. 65, pages 386–407, 1958.

[Rumelhart 1986] D. E. Rumelhart, G. E. Hinton and R. J. Williams. *Learning representations by back-propagating errors*. Nature, vol. 323, pages 533–536, 1986.

[Samuel 1959] A. L. Samuel. *Some studies in machine learning using the game of checkers*. IBM Journal of Research and Development, vol. 3, pages 211–229, 1959.

[Schölkopf 1996] B. Schölkopf, C. Burges and V. Vapnik. *Incorporating invariances in support vector learning machines*. In Artificial Neural Networks - ICANN '96, volume 1112 of *Lecture Notes in Computer Science*, pages 47–52. Springer Verlag Berlin/Heidelberg, 1996.

[Schölkopf 2002] B. Schölkopf and A. Smola. Learning with kernels. MIT Press, 2002.

[Schneider 2006] J. J. Schneider and S. Kirkpatrick. Stochastic optimization. Springer Verlag Berlin, 2006.

[Schrader 2010] L. Schrader. *http://tulane.edu/sse/cmb/people/schrader/*. online, 2010. Tulane University.

[Selfridge 1959] O. G. Selfridge. *Pandemonium: a paradigm for learning*. Proceedings of the symposium on mechanization of thought processes, London, pages 511–529, 1959.

[Selinski 2005] S. Selinski and K. Ickstadt. *Similarity Measures for Clustering SNP Data*. Rapport technique, Universität Dortmund, SFB 475 Komplexitätsreduktion in Multivariaten Datenstrukturen, 2005.

[Vapnik 1968] V. N. Vapnik and A. Ja. Chervonenkis. *On the uniform convergence of relative frequencies of events to their probabilities*. Doklady Akademii Nauk, vol. 181, 1968.

[Vapnik 1971] V. Vapnik and A. Chervonenkis. *On the uniform convergence of relative frequencies of events to their probabilities.* Theory Probab. Apl., vol. 16, pages pp. 264–280, 1971.

[Vapnik 1974] V. N. Vapnik and A. Ja. Chervonenkis. *Theory of Pattern Recognition.* Nauka, 1974.

[Vapnik 1989] V. N. Vapnik and A. Ja. Chervonenkis. *The necessary and sufficient conditions for consistency of the method of empirical risk minimization.* Yearbook of the Academy of Sciences of the USSR on Recognition, Classification, and Forcasting, vol. 2, pages 217–249, 1989.

[Vapnik 1998] V. N. Vapnik. Statistical learning theory. J. Wiley, New York, 1998.

[Vapnik 1999] V. N. Vapnik. The nature of statistical learning theory. Springer Verlag New York, 1999.

[Vapnik 2006] V. N. Vapnik. Estimation of dependencies based on empirical data. Springer, New York, 2006.

[von Luxburg 2004] U. von Luxburg and O. Bousquet. *Distance-based Classification with Lipschitz Functions.* Journal of Machine Learning Research, vol. 5, pages 669–695, 2004.

[Wahba 1990] G. Wahba. *Spline Models for Observational Data.* SIAM (CBMS-NSF Regional Conference series in applied mathematics), vol. 59, 1990.

[Wang 1988] C. H. Wang and S. N. Srihari. *A framework for object recognition in a visually complex environment and its application to locating adress blocks on mail pieces.* International Journal of Computer Vision, vol. 2, page 125, 1988.

[Weaver 1999] N. Weaver. Lipschitz algebras. World Scientific, Singapore, 1999.

[Wendemuth 1995] A. Wendemuth. *Learning the Unlearnable.* Journal of Physics A: Mathematical and General, vol. 28, pages pp. 5423–5436, 1995.

[Widrow 1962] B. Widrow. *Generalization and Information Storage in Networks of Adaline 'Neurons'.* In Self-Organizing Systems, symposium proceedings, Washington, DC, 1962.

[Wolpert 1997] D. H. Wolpert and W. G. Macready. *No Free Lunch Theorems for Optimization*. IEEE Transactions on Evolutionary Computation, vol. 1, 67, 1997.

[Woodbury 1950] M. A. Woodbury. *Inverting modified matrices*. Memorandum Rept. 42, Statistical Research Group, Princeton University, Princeton, NJ, 1950.

[Zhou 2002] D. Zhou, B. Xiao, H. Zhou and R. Dai. *Global Geometry of SVM Classifier*. Rapport technique, AI Lab, Institute of Automation, Chinese Acadamy of Sciences, 2002.