Robust Boosting via Convex Optimization: Theory and Applications

Dissertation

vorgelegt von
Gunnar Rätsch

zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften
– Dr. rer. nat. –

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät
der Universität Potsdam

Gutachter:
Prof. Dr. K.-R. Müller
Prof. Dr. M.K. Warmuth
Prof. Dr. K.P. Bennett

Potsdam, im Oktober 2001

Summary

In this work we consider *statistical learning problems*. A learning machine aims to extract information from a set of training examples such that it is able to predict the associated label on unseen examples. We consider the case where the resulting *classification* or *regression* rule is a combination of *simple rules* – also called *base hypotheses*. The so-called *boosting algorithms* iteratively find a *weighted linear combination* of base hypotheses that predict well on unseen data. We address the following issues:

- *The statistical learning theory framework for analyzing boosting methods.*

We study learning theoretic guarantees on the prediction performance on unseen examples. Recently, *large margin classification techniques* emerged as a practical result of the theory of generalization, in particular Boosting and Support Vector Machines. A large margin implies a good generalization performance. Hence, we analyze how large the margins in boosting are and find an improved algorithm that is able to generate the *maximum* margin solution.

- *How can boosting methods be related to mathematical optimization techniques?*

To analyze the properties of the resulting classification or regression rule, it is of high importance to understand whether and under which conditions boosting converges. We show that boosting can be used to solve large scale constrained optimization problems, whose solutions are well characterizable. To show this, we relate boosting methods to methods known from mathematical optimization, and derive convergence guarantees for a quite general family of boosting algorithms.

- *How to make Boosting noise robust?*

One of the problems of current boosting techniques is that they are sensitive to noise in the training sample. In order to make boosting robust, we transfer the soft margin idea from support vector learning to boosting. We develop theoretically motivated regularized algorithms that exhibit a high noise robustness.

- *How to adapt boosting to regression problems?*

Boosting methods are originally designed for classification problems. To extend the boosting idea to regression problems, we use the previous convergence results and relations to semi-infinite programming to design boosting-like algorithms for regression problems. We show that these *leveraging algorithms* have desirable theoretical and practical properties.

- *Can boosting techniques be useful in practice?*

The presented theoretical results are guided by simulation results either to illustrate properties of the proposed algorithms or to show that they work well in practice. We report on successful applications in a non-intrusive power monitoring system, chaotic time series analysis and a drug discovery process.

Keywords Boosting, Large Margin Classification, Support Vectors, Regression, Regularization, Mathematical Optimization, Power Monitoring, Time Series Analysis, Drug Discovery

Preface

I started my research on *boosting methods* already in December 1997 while working on my Master's thesis. In May 1998 – when I was starting my PhD – I decided to continue working on boosting, since these seemingly simple algorithms are quite difficult to analyze and not fully understood. I noticed that the more I understood how they are functioning, the more challenging and exciting problems emerged that were to solve. Since I had to focus on the most interesting aspects, some questions still remain un-answered, although the understanding is now much deeper than before and, in fact, lead to new practical algorithms.

The work for my Master was based on empirical results and the algorithms proposed there were mainly motivated by intuitions or analogies. In the present thesis I worked out these intuitions more theoretically, and hence, it is clear that it contains more theory than applications. In this thesis I will report on two real-world applications only. There are indeed more, and I actually devoted a large part of my efforts in applying machine learning techniques to e.g. DNA and protein analysis problems [cf. Zien et al., 2000, Tsuda et al., 2002, Sonnenburg et al., 2002], drug discovery tasks [cf. Rätsch et al., 2002, Warmuth et al., 2002], optical character recognition (OCR) problems [e.g. Mika et al., 2000a], particle physics data [cf. Vannerem et al., 1999], time series analysis [cf. Müller et al., 1999, Rätsch et al., 2002], problems originating from the electric power industry [cf. Onoda et al., 2000], and recently also fraud detection.

This thesis does not contain my contributions to supervised kernel methods [cf. Mika et al., 1999a, 2000b, 2001, Müller et al., 2001], nonlinear feature extraction [cf. Schölkopf et al., 1998, Mika et al., 1999b, Schölkopf et al., 1999b, Mika et al., 2000a], novelty & outlier detection [cf. Rätsch et al., 2002], time series segmentation [cf. Kohlmorgen et al., 2000], and meta-learning [cf. Tsuda et al., 2001].

I tried to make the presentation as self-contained as possible. However, due to a lack of space, I had to omit details, which might be necessary for a full understanding – but they can easily be looked up in referenced work. For readers not familiar with the machine learning setting, I suggest to start reading in Chapter 1, which gives a brief review of statistical learning theory, Boosting and Support Vector Machines for classification. The others might start reading in Chapter 2, which gives an analysis of Boosting. Nevertheless, this chapter is not absolutely necessary for understanding the subsequent chapters. Those that are interested in the convergence analysis of Boosting-type algorithms should read Chapter 3, which introduces the basic tools for subsequent chapters. Finally, practitioners might just want to read some parts of Chapter 4 and Chapter 5 to understand how the new algorithms work and to become convinced that the proposed algorithms are useful and easy to use in practice.

Acknowledgments

First of all, I would like to express my thanks to Prof. Dr. K.-R. Müller, Prof. Dr. M. Warmuth and Prof. Dr. K. Bennett for supervising and reviewing the present dissertation, and to Prof. Dr. L. Budach for chairing the committee in the "Wissenschaftliche Aussprache". I am very grateful to Klaus Müller for introducing me to the field of machine learning during my work on the Master and the doctoral dissertation. I appreciated many interesting and guiding discussions and the stimulating research environment he was able to create in his group at Fraunhofer institute FIRST in Berlin (former GMD FIRST). To Manfred Warmuth, I am very grateful for his warm hospitality at UC Santa Cruz and for many enlightening discussions on Boosting, Bregman distances and statistical learning theory. I would like to thank Kristin Bennett for several fruitful discussions on mathematical programming techniques useful for Boosting. All three influenced my work in a way, such that the resulting algorithms are useful in practice, learning-theoretically motivated and build on a good basis of optimization theory.

Several other people contributed to this thesis in one way or another. Let me start with Sebastian Mika, Anja Pannek, Steven Lemm, Stefan Harmeling, and Bernhard Schölkopf for proofreading parts of the manuscript and for helping to improve the presentation of my work in many respects.

Most of the work of this dissertation has been done at the Fraunhofer institute FIRST in Berlin. I would like to thank all current and former members of this group for creating an open research atmosphere, including B. Blankertz, S. Harmeling, M. Kawanabe, J. Kohlmorgen, F. Meinicke, P. Philips, B. Schölkopf, A. Smola, S. Sonnenburg, K. Tsuda, R. Vigario, and A. Ziehe. I particularly thank my room-mates Sebastian Mika and Steven Lemm which helped in many ways. Moreover, I would like to thank the administrative staff particularly R. Holst, A. Schulz, E. Simons, and B. Sad for doing a great job. I also thank all people at Fraunhofer-FIRST that allowed me to use their computers for many long simulations.

Other parts of my work have been done at research visits at the Australian National University (ANU) in Canberra, the Central Research Institute of the Electric Power Institute (CRIEPI) in Tokyo and at the University of California at Santa Cruz (UCSC). For very interesting and encouraging discussions during my short visit of ANU, I particularly thank R. Williamson and also J. Barnes, L. Mason, J. Baxtor, P. Bartlett and M. Hegland. Special thanks goes to Takashi Onoda, for inviting me to CRIEPI for three months. He partially supervised my work on the Master and always asked the right questions. I gratefully acknowledge the warm hospitality from all people at CRIEPI. Moreover, I spend almost a half year at UCSC working with Manfred Warmuth. I would like to thank him and also

A. Jagota, R. Karchin, N. Duffy, D. Helmbold, T. Furey, M. Diekhans and K. Karplus for interesting discussions and helpful comments on my research work.

Many other discussions took place at conferences or workshops. So I would also like to thank J. Weston, O. Bousquet, O. Chapelle, R. Herbrich, T. Graepel, C. Campbell, C. Watkins, J. Shawe-Taylor and from which my work has profited. Moreover, I thank all my teachers and friends who helped me to become a scientist – in particular H. Haase, A. Daniel, W. Schröder-Preikschat, A. Pannek, H. Hohberger, H. Junek, L. Budach, E. Horn, W. Schubert, and other aforementioned.

Most of the contents of this thesis has already been published in journals or conference proceedings. Without my co-authors, these publications and the present thesis would not have been possible. I like to thank all of them for doing a great job and also for allowing me to use text-pieces and figures for this dissertation. Chapter 1 contains material of Müller et al. [2001], Chapter 2 summarizes parts of Rätsch et al. [2001, 2000c], Rätsch and Warmuth [2001], Chapter 3 contains material of Rätsch et al. [2002], Chapter 4 is based on Rätsch et al. [2001, 2000a], Onoda et al. [2000], and parts of Chapter 5 are taken from Rätsch et al. [2002].

I gratefully acknowledge partial support from DFG grants MU 987/1-1, JA 379/91 and JA 379/71, from the EC Neurocolt and STORM projects, and the NSF grant CCR 9821087.

Finally, I thank my parents and my grandfather for giving me a push into the right direction. Moreover, I thank my love Anja for her patience, many scientific discussions and help in many other ways.

Contents

1 Introduction and Preliminaries **1**
- 1.1 Overview . . . 1
- 1.2 Statistical Learning – Some Background . . . 4
  - 1.2.1 The Learning Setting . . . 4
  - 1.2.2 VC Dimension and PAC Learning . . . 5
  - 1.2.3 VC Bounds and Structural Risk Minimization . . . 6
- 1.3 Boosting and Support Vector Machines . . . 8
  - 1.3.1 PAC Boosting . . . 8
  - 1.3.2 Support Vector Machines . . . 12
  - 1.3.3 p-Norm Margins in Feature Space . . . 16
  - 1.3.4 Boosting vs. SVMs . . . 17

2 Boosting vs. Margin Maximization **19**
- 2.1 von Neumanns Min-Max Theorem . . . 20
- 2.2 AdaBoost$_\varrho$ . . . 21
- 2.3 Asymptotic Analysis . . . 23
  - 2.3.1 The Asymptotical Length of $\boldsymbol{\alpha}$ . . . 23
  - 2.3.2 The Limiting Distribution $\mathbf{d}$ and Support Vectors . . . 24
  - 2.3.3 How Large is the Margin? . . . 25
  - 2.3.4 Experimental Illustration of Asymptotical Properties . . . 26
  - 2.3.5 Summarizing Remarks . . . 28
- 2.4 Marginal Boosting . . . 29
  - 2.4.1 Motivation . . . 29
  - 2.4.2 The Algorithm and its Analysis . . . 30
  - 2.4.3 Experimental Illustration . . . 31
  - 2.4.4 Summarizing Remarks . . . 33
- 2.5 Relation to Barrier Optimization . . . 34
  - 2.5.1 Preliminaries . . . 34
  - 2.5.2 Relating Arc-GV to Barrier Optimization . . . 35
  - 2.5.3 Finding a Separation with AdaBoost$_\varrho$ . . . 36
- 2.6 Discussion and Summary . . . 37

3 On the Convergence of Leveraging **39**
- 3.1 Leveraging algorithms . . . 40
  - 3.1.1 AdaBoost & Logistic Regression . . . 40
  - 3.1.2 Least-Square-Boost . . . 41

3.1.3 The General Case . . . 42
3.1.4 Assumptions . . . 43
3.2 The Dual Algorithm and Bregman Distances . . . 44
3.2.1 AdaBoost as Entropy Projection . . . 44
3.2.2 Generalized Distances and Generalized Projections . . . 45
3.2.3 Generalized Projections onto Intersections of Hyperplanes . . . 46
3.2.4 Summarizing Remarks . . . 50
3.3 Coordinate Descent . . . 51
3.3.1 Relation to Generalized Projections . . . 51
3.3.2 Convergence Theorems . . . 52
3.3.3 Summarizing Remarks . . . 54
3.4 Application to Leveraging . . . 55
3.5 Discussion and Summary . . . 57

4 Soft Margins **59**
4.1 Hard margins and overfitting . . . 60
4.2 Reducing the Influence of Hard Examples . . . 63
4.2.1 Trade-off Between Margin and Influence . . . 63
4.2.2 $\text{AdaBoost}_{\text{Reg}}$ . . . 64
4.2.3 Experimental Illustration . . . 66
4.2.4 Summarizing Remarks . . . 67
4.3 Algorithms based on Linear Programs . . . 68
4.3.1 The ν-LP Problem . . . 68
4.3.2 ν-Arc . . . 70
4.3.3 A Barrier Algorithm . . . 72
4.3.4 An Illustrating Toy Experiment . . . 76
4.3.5 Summarizing Remarks . . . 77
4.4 Evaluation and an Application . . . 78
4.4.1 Evaluation on Benchmark Data Sets . . . 79
4.4.2 An Application to a Non-intrusive Power Monitoring System . . . 82
4.5 Discussion and Summary . . . 85

5 Ensembles for Regression **87**
5.1 Optimization Problems and Loss Functions for Regression . . . 88
5.1.1 Problem definition and Preliminaries . . . 88
5.1.2 Linear Regression in Feature Spaces . . . 90
5.1.3 Loss Functions . . . 90
5.2 Sparseness induced by Regularization . . . 94
5.2.1 Strictly Convex vs. Concave Regularization . . . 95
5.2.2 Convex Sparseness Regularization . . . 96
5.2.3 Boosting vs. SVMs again . . . 97
5.3 Infinite Hypothesis Sets and Semi-Infinite Programming . . . 98
5.3.1 Dual formulation . . . 98
5.3.2 The Dual Problem for Infinite Hypothesis Sets . . . 99

5.3.3 Primal Regression SIP . . . 102
5.4 Optimization Algorithms . . . 104
5.4.1 Column Generation Method for the ε-insensitive Loss . . . 104
5.4.2 A Regularized Leveraging Approach . . . 106
5.4.3 A Barrier Approach for the ε-insensitive Loss . . . 108
5.5 Evaluation and an Application . . . 110
5.5.1 An Experiment on toy data . . . 111
5.5.2 Time Series Prediction . . . 112
5.5.3 Experiments on Drug data . . . 115
5.6 Discussion and Summary . . . 117

6 Synopsis **119**

References **121**

A Proofs **135**
A.1 Proofs from Chapter 2 . . . 135
A.1.1 Proof of Lemma 2.1 from page 24 . . . 135
A.1.2 Proof of Lemma 2.2 from page 23 . . . 135
A.1.3 Proof of Lemma 2.3 from page 24 . . . 135
A.1.4 Proof of Lemma 2.4 from page 24 . . . 136
A.1.5 Proof of Theorem 2.2 from page 26 . . . 136
A.1.6 Proof of Corollary 2.2 from page 26 . . . 137
A.1.7 Proof of Theorem 2.3 from page 31 . . . 137
A.2 Proofs from Chapter 3 . . . 138
A.2.1 Proof of Proposition 3.2 from page 50 . . . 138
A.2.2 Proof of Proposition 3.3 from page 53 . . . 140
A.2.3 Proof of Proposition 3.4 from page 53 . . . 140
A.2.4 Proof of Theorem 3.5 from page 56 . . . 141
A.2.5 Proof of Corollary 3.1 from page 56 . . . 142
A.3 Proofs from Chapter 4 . . . 142
A.3.1 Proof of Proposition 4.3 from page 70 . . . 142
A.3.2 Proof of Proposition 4.4 from page 75 . . . 143
A.3.3 Proof of Theorem 4.3 from page 76 . . . 145
A.4 Proofs from Chapter 5 . . . 146
A.4.1 Proof of Proposition 5.1 from page 92 . . . 146
A.4.2 Proof of Proposition 5.2 from page 93 . . . 147
A.4.3 Proof of Proposition 5.3 from page 95 . . . 148
A.4.4 Proof of Proposition 5.4 from page 95 . . . 149
A.4.5 Proof of Corollary 5.1 from page 101 . . . 150
A.4.6 Derivation of the dual Regression Problem with ℓ_1-norm regularization (page 98) . . . 150
A.4.7 Proof of Lemma 5.2 from page 99 . . . 151
A.4.8 Proof of Lemma 5.3 from page 99 . . . 152
A.4.9 Proof of Corollary 5.2 from page 103 . . . 152

A.4.10 Proof of Theorem 5.7 from page 105 152
A.4.11 Proof of Theorem 5.8 from page 107 153

B Technical Addenda **155**
B.1 Notation 155
B.2 Barrier Optimization 157
B.3 Another proof for Arc-GV 158
B.4 A Note on Infinite Hypothesis Spaces for Marginal Boosting 159
B.5 Bounds with Compression Schemes 160
B.6 Examples for Base Learning Algorithms 162
B.6.1 RBF nets with adaptive centers 162
B.6.2 Kernel functions 163
B.6.3 (Active) Kernel Functions 165
B.6.4 Classification Functions 165

1 Introduction and Preliminaries

1.1 Overview

This thesis falls into the broad field of *Artificial Intelligence*, which aims to mimic intelligent abilities of humans by machines. In the field of *Machine Learning*, which evolved from artifical intelligence research, one considers the important question of how to make machines able to "learn". Learning in this context is understood as *inductive inference*, where one observes *examples* that represent incomplete information about some "statistical phenomenon". In *supervised learning* as considered in this thesis, there is a *label* associated with each example. It is supposed to be the answer to a question about the example. If the label is discrete, then the task is called *classification problem* – otherwise, for real-valued labels we speak of a *regression problem*. Based on these examples (including the labels), one is particularly interested to *predict* the answer for other cases before they are explicitly observed. Hence, learning is not only a question of remembering but also of *generalization to unseen cases*.

Machine learning rests upon the theoretical foundation of *Statistical Learning Theory* [e.g. Vapnik, 1995]. It provides conditions and guarantees for good generalization of learning algorithms. In Section 1.2 we will give some basic definitions and results of this theory. Within the last decade, *large margin classification techniques* have emerged as a practical result of the theory of generalization. Roughly speaking, the margin is the distance of the example to the separation boundary and a large margin classifier generates decision boundaries with large margins to almost all training examples. The two most widely studied classes of large margin classifiers are *Support Vector Machines* (SVMs) [Boser et al., 1992, Cortes and Vapnik, 1995] and *Boosting* [Valiant, 1984, Schapire, 1992]. In this thesis we consider boosting-type algorithms.

The basic idea of boosting and *ensemble learning algorithms* in general is to iteratively combine relatively simple *base hypotheses* – sometimes called *rules of thumb* – for the final prediction. One uses a *base learner* that generates the base hypotheses. In boosting and most other ensemble learning algorithms the base hypotheses are linearly combined. In the case of *two-class classification*, the final prediction is the weighted majority of the votes. In regression it is the weighted average of the individual predictions. The combination of these simple rules can boost the performance drastically.

Past years have seen strong interest in boosting methods, in particular *AdaBoost* [Freund and Schapire, 1994], due to their initial success in practical classification applications [e.g. Drucker et al., 1993, 1994, LeCun et al., 1995, Freund and Schapire, 1996b, Maclin and Opitz, 1997, Schwenk and Bengio, 1997]. The good performance of boosting algorithms

was explained by the PAC theory [cf. Valiant, 1984]. It was shown that AdaBoost has the so-called *PAC boosting* property: If the learner generating the base hypotheses is just slightly better than random guessing, AdaBoost is able to find a combined hypothesis with arbitrary high accuracy (if enough training examples are available).

Later it was found that there is another explanation based on *large margins*, showing that AdaBoost and SVMs are intimately related. It has been pointed out that AdaBoost generates combined hypotheses with large margins on *all* examples [cf. Schapire et al., 1998, Grove and Schuurmans, 1998, Breiman, 1999, Onoda et al., 1998]. A learning-theoretic guarantee was derived, which states that combined hypotheses with large margins generalize well [Schapire et al., 1998]. However, it was never shown whether one can actually *maximize the margin* with boosting techniques. We give an answer to this question in Chapter 2.

The large margin separation is found by minimizing some *exponential loss function* depending on the predictions of the combined hypothesis on the training examples. There is the interesting question, why boosting-type algorithms tend to maximize the margin only by minimizing such loss function. We have found an explanation by relating AdaBoost-like algorithms to techniques known from mathematical optimization. In Section 2.5 it is shown that they are implementing a so-called *barrier optimization technique* to approximately solve a *linear optimization problem*.

Later it was found empirically that Boosting methods tend to generalize well on low noise problems only [e.g. Quinlan, 1996, Breiman, 1999, Grove and Schuurmans, 1998, Rätsch et al., 2001, Mason et al., 1998], namely, if the labels of the examples in the training set can be assumed to be correct and deterministically computed by some *target hypothesis*. So boosting methods were restricted to the analysis of phenomenona with low noise. Whereas the problem of non-separability had become clear very early for SVMs since some equations do not have a solution, it was not clear for a long time for AdaBoost, since the combined hypothesis found by AdaBoost was often still meaningful, even if no combined hypothesis with large margin exists. This issue will be explicitly analyzed and clarified in this thesis.

These practical problems led to new developments of boosting-type algorithms that take the statistical nature of the examples and labels into account [e.g. Mason et al., 1998, Rätsch et al., 2001, Friedman et al., 2000, Domingo and Watanabe, 2000, Moerland and Mayoraz, 1999, Singer, 2000, Aslam, 2000, O'Sullivan et al., 2000, Rätsch et al., 2000a, Krieger et al., 2001, Demiriz et al., 2001b]. In Chapter 4 it will be been shown that the *soft margin* maximization techniques utilized in support vector machines [Bennett and Mangasarian, 1992, Cortes and Vapnik, 1995] can be readily adapted to produce *regularized* ensembles for classification [cf. Rätsch et al., 2001, 2000a, Bennett et al., 2000]. Our approach was one of the first solutions to reduce the problem of *overfitting* in boosting. The proposed algorithms extend boosting-type algorithms to the case of noisy data and thus make them a powerful state-of-the-art learning technique, next to SVMs.

Modifications of AdaBoost are often based on the idea to replace the exponential loss function with another function or to directly modify the algorithm [see also Margineantu

and Dietterich, 1997, Schapire and Singer, 1999, Das, 2001, Avnimelech and Intrator, 1999a, Meir et al., 2000, Lazarevic and Obradovic, 2001]. However, for most of the proposed algorithms it is not clear whether and under which conditions they converge to a meaningful result. In Chapter 3 we analyze a large family of such algorithms in terms of their convergence properties. We derive sufficient conditions for convergence to the minimum of some empirical loss. This result closes a central gap between existing algorithms and their theoretical understanding in terms of their convergence to well-defined solutions. We exploit theses results in Chapter 4 and Chapter 5 to prove the convergence of the proposed algorithms.

Motivated by the success of boosting methods in the classification setting, it has been attempted by several researchers to transfer boosting techniques to regression problems [e.g. Freund and Schapire, 1994, Fisher, 1997, Bertoni et al., 1997, Friedman, 1999, Avnimelech and Intrator, 1999b, Ridgeway et al., 1999, Duffy and Helmbold, 2000a, Rätsch et al., 2000c, Shawe-Taylor and Karakoulas, 2000, Zemel and Pitassi, 2001]. However, it was experienced that this transformation is rather difficult, and for long no convincing boosting-like algorithm has been found for regression that is theoretically founded and works well in practice. In Chapter 5 we develop such algorithms. One major difficulty is rigorously defining the regression problem in an infinite hypothesis space. Whereas in classification the hypothesis space is often finite, for regression even relatively simple hypothesis spaces consist of an uncountable infinite set of hypotheses. It is *a priori* not clear how to even express a regression problem in an infinite hypothesis space from an optimization point of view. Clearly one can only practically consider ensemble functions that are a linear combination of some *small subset* of the set of possible hypotheses. In Section 5.3, we study directly the issue of infinite hypothesis spaces and implications for regression algorithms.

A central part of this work considers the convergence properties of boosting-like algorithms. Our analysis relates those algorithms to convex optimization techniques, leading to clear statements, whether, under which conditions and how fast the output of the considered algorithms converge to a solution of a mathematical programming problem. This builds the basis for designing new algorithms and for analyzing them in the context of learning theory.

The presented theory is guided with simulation results either to illustrate properties of the proposed algorithms or to show that they work in practice. In comparative simulation studies we show that our algorithms are indeed very useful and ready to use for solving practical problems. We report on successful real-world applications in a non-intrusive power monitoring system and in the drug discovery process.

1.2 Statistical Learning – Some Background

1.2.1 The Learning Setting

Let us start with a general notion of the learning problems that we consider in this work. The task of classification is to find a rule, which, based on external observations, assigns an object to one of several classes. In the simplest case there are only two different classes. One possible formalization of this task is to estimate a function $f : \mathbb{X} \to \mathbb{Y}$, where $\mathbb{X}$ is the input domain and $\mathbb{Y} = \{-1, +1\}$ is the set of possible labels (cf. Appendix B.1 for a summary of the notation used in this work). One uses input-output training data pairs generated independent identically distributed (i.i.d.) according to an unknown probability distribution P on $\mathbb{Z} = \mathbb{X} \times \mathbb{Y}$ with density $p(\boldsymbol{x}, y)$

$$S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\} \in \mathbb{Z}^N, \tag{1.1}$$

such that f will correctly classify unseen examples $(\boldsymbol{x}, y) \in \mathbb{Z}$. An example is assigned to the class $+1$ if $f(\boldsymbol{x}) \geq 0$ and to the class -1 otherwise. The test examples are assumed to be generated from the same probability distribution P as the training data. The best function f that one can obtain is the one minimizing the expected risk (also *generalization error*)

$$R[f] = \int \mathrm{g}(y, f(\boldsymbol{x}))\, dp(\boldsymbol{x}, y) = \mathbf{E}_{\mathsf{X},\mathsf{Y}}\{\mathrm{g}(\mathsf{Y}, f(\mathsf{X}))\} \tag{1.2}$$

where g denotes a suitably chosen loss function and X and Y are the random variables distributed according to P. For classification one often considers the $0/1$-loss: $\mathrm{g}(y, f(\boldsymbol{x})) = \mathbf{I}(-yf(\boldsymbol{x}))$, where $\mathbf{I}(z) = 0$ for $z < 0$ and $\mathbf{I}(z) = 1$ otherwise. The same framework can be applied for regression problems, where $y \in \mathbb{R}$. Here, the most common loss function is the *squared loss*: $\mathrm{g}(y, f(\boldsymbol{x})) = (f(\boldsymbol{x}) - y)^2$; see Chapter 5 and e.g. Smola et al. [1998c], Smola [1998] for discussions of other loss functions. However, for the rest of this introduction, we assume the $0/1$ loss.

Unfortunately the risk cannot be minimized directly, since the underlying probability distribution $P(\boldsymbol{x}, y)$ is unknown. Therefore, we have to try to estimate a function that is *close* to the optimal one based on the available information, i.e. the training sample and properties of the function class F the solution f is chosen from. To this end, we need what is called an *induction principle*. A particular simple one – called *empirical risk minimization* (ERM) – consists in approximating the minimum of the risk (1.2) by the minimum of the *empirical risk*

$$R_{\mathrm{emp}}[f, S] = \frac{1}{N} \sum_{n=1}^{N} \mathrm{g}(y_n, f(\boldsymbol{x}_n)). \tag{1.3}$$

Clearly, if the function class F is "large" the empirical risk may deviate from the expected risk drastically – this effect is called *overfitting* (cf. Figure 1.1). One of the most important problems in learning theory is to characterize, under which conditions the empirical risk minimization leads to estimates with small expected risk.

In the following sections we will briefly introduce some main concepts of statistical learning theory.

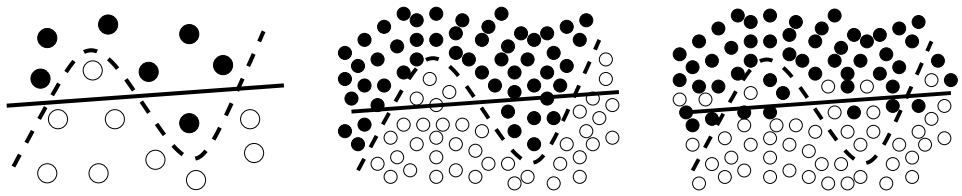

Figure 1.1 Illustration of the overfitting dilemma: Given only a small sample [left] either, the solid or the dashed hypothesis might be true, the dashed one being more complex, but also having a smaller training error. Only with a large sample we are able to see which decision reflects the true distribution more closely. If the dashed hypothesis is correct the solid would underfit [middle]; if the solid were correct the dashed hypothesis would overfit [right]. (Figure taken from Müller et al. [2001].)

1.2.2 VC Dimension and PAC Learning

A specific way of characterizing the complexity of a function class is given by VC theory [Vapnik and Chervonenkis, 1974, Vapnik, 1995, 1998]. Here the concept of complexity is captured by the Vapnik-Chervonenkis (VC) dimension ϑ of the function class F that the estimate f is chosen from. Roughly speaking, the VC dimension measures how many (training) examples can be shattered (i.e. separated) for all possible labelings using functions of the class [for an exact definition see e.g. Vapnik and Chervonenkis, 1971, Vapnik, 1995].

Although the VC dimension is an important concept in statistical learning theory, it is often difficult to determine for function classes of interest. However, for some particular cases, e.g. linear hyperplanes one can bound the VC dimension either in terms of the dimensionality of the input space $\mathbb{X}$ or by another quantity, the *margin*, which we will use in Sections 1.3.1 and 1.3.2 and define more formally in Section 1.3.3.

One of the main approaches to analyze the performance of learning machines is the PAC framework. The term PAC means *probably approximately correct* and has been introduced in Valiant [1984]. The fundamental assumption made in this framework is that the exists a target function $f^* \in \mathsf{F}$ that generates the labels of all observed data. Thus, with probability one, the label y of an example $\boldsymbol{x}$ is $f^*(\boldsymbol{x})$.

We have the following slightly simplified[1] definition of a PAC learner:

Definition 1.1 (e.g. Valiant [1984], Blumer et al. [1989]). *A (strong) PAC learner for a concept class F has the property that for every distribution P_{X}, all concepts $f \in \mathsf{F}$, and all $0 < \epsilon, \delta \leq 1$: With probability at least $1 - \delta$ the algorithm $\mathfrak{L}$ outputs a hypothesis h with $P(h(\mathsf{X}) \neq f(\mathsf{X})) \leq \epsilon$. The learning algorithm is given $\mathsf{F}, \epsilon, \delta$ and the ability to draw random examples of f with respect to the distribution P, and must run in polynomial time in $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$.*

1. In a refined version one uses polynomial growth in the parameters of the concept class Blumer et al. [1989].

The parameters δ and ϵ are called *confidence* and *accuracy*, respectively. The PAC learner returns *probably* an *approximately correct* hypothesis in polynomial time, i.e. using only a polynomial number of examples.

Using standard results from PAC theory one can show that the probability that two functions agree on a training sample, but deviate strongly on the rest of the domain, is small, if the sample is large enough:

Theorem 1.1 (Risk Deviation, Blumer et al. [1989]). *Suppose that* F *is a function class of finite VC dimension* $\vartheta \geq 1$ *and that* $0 < \delta, \epsilon < 1$. *Let*

$$N_0(\mathsf{F}, \delta, \epsilon) = \left\lceil \frac{4}{\epsilon} \left(\vartheta \log_2 \left(\frac{12}{\epsilon} \right) + \log_2 \left(\frac{2}{\delta} \right) \right) \right\rceil .$$

With probability at least $1 - \delta$ *over the random draw of a training sample* S *of size* $N \geq N_0(\mathsf{F}, \delta, \epsilon)$*, for any two functions* $f^*, f \in \mathsf{F}$ *that agree on* S *holds*

$$P(f(\mathsf{X}) \neq f^*(\mathsf{X})) \leq \epsilon.$$

Note that this theorem does not depend on the distribution of X and holds for any two functions f and f^* uniformly. The *uniform convergence* property is very important for the analysis of machine learning algorithms. In traditional statistics one has often *point-wise* convergence only – leading to much weaker results.

By Theorem 1.1 it can readily be seen that an algorithm that always finds a hypothesis $f \in \mathsf{F}$ consistent with the training set is a PAC learner. We therefore have:

Corollary 1.1 (Blumer et al. [1989]). *Suppose that the hypothesis space* F *has VC dimension* $1 \leq \vartheta < \infty$. *Then any consistent learning algorithm* $\mathcal{L}$ *for* F *is a PAC learner.*

So, if the complexity of the function class is small enough, one can ensure that asymptotically (as $N \to \infty$), the empirical risk will converge towards the expected risk. However, for small sample sizes large deviations are possible and *overfitting* might occur (see Figure 1.1). Then a small expected risk can often not be obtained by simply minimizing the training error (1.3). We will come back to PAC learning in the context of Boosting in Section 1.3.1.

1.2.3 VC Bounds and Structural Risk Minimization

One way to avoid the overfitting dilemma is to *restrict* the complexity of the function class F where one chooses the function f from [Vapnik, 1995]. The intuition, which will be formalized in the following is that a "simple" (e.g. linear) function that explains most of the data is preferable to a complex one [Occam's razor, e.g. Blumer et al., 1987]. Typically one introduces a *regularization* term [e.g. Kimeldorf and Wahba, 1971, Tikhonov and Arsenin, 1977, Poggio and Girosi, 1990, Cox and O'Sullivan, 1990] to limit the complexity of the function class F from which the learning machine can choose. This raises the problem of model selection, i.e. how to find the optimal complexity of the function class [e.g. Akaike, 1974, Poggio and Girosi, 1990, Moody, 1992, Murata et al., 1994].

A specific way of controlling the complexity of a function class is given by the *structural risk minimization* (SRM) induction principle [Vapnik and Chervonenkis, 1974, Vapnik,

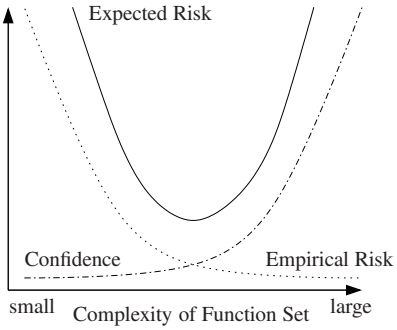

Figure 1.2 Schematic illustration of (1.4). The dotted line represents the training error (empirical risk), the dash-dotted line the upper bound on the complexity term (confidence). With higher complexity the empirical error decreases but the upper bound on the risk confidence becomes worse. For a certain complexity of the function class the best expected risk (solid line) is obtained. Thus, in practice the goal is to find the best trade-off between empirical error and complexity. (Figure taken from Müller et al. [2001].)

1995, 1998]. Constructing a nested family of function classes $\mathsf{F}_1 \subset \cdots \subset \mathsf{F}_K$ with non-decreasing VC dimension, the SRM principle proceeds as follows: Let $f_1, \ldots, f_K$ be the solutions of the empirical risk minimization (1.3) in the function classes $\mathsf{F}_k, k = 1, \ldots, K$. SRM chooses the function class F_k (and the function f_k) such that an upper bound on the generalization error is minimized, which can be computed by making use of theorems such as the following one (see also Figure 1.2):

Theorem 1.2 (VC Bound, Vapnik [1995, 1998]). *Let ϑ denote the VC dimension of the function class F, S be a sample of size N drawn i.i.d. from some probability distribution P and let R_{emp} be defined by (1.3) using the $0/1$-loss. For all $\delta > 0$ and $f \in \mathsf{F}$ the inequality bounding the risk*

$$R[f] \leq R_{\mathrm{emp}}[f, S] + \sqrt{\frac{8}{N}\left[\vartheta\left(\log_2 \frac{2N}{\vartheta} + 1\right) + \log_2\left(\frac{4}{\delta}\right)\right]} \tag{1.4}$$

holds with probability of at least $1 - \delta$ for $N > h$.

For the special case where the empirical loss is zero, we get the following theorem, where the square root disappears:[2]

Theorem 1.3 (PAC Bound, Vapnik [1995, 1998]). *Let ϑ denote the VC dimension of the function class F and S be a sample of size N drawn i.i.d. from P. For all $\delta > 0$ and $f \in \mathsf{F}$ being a function consistent with the sample S the inequality bounding the risk*

$$R[f] \leq \frac{4}{N}\left[\vartheta\left(\log_2\left(\frac{2N}{\vartheta}\right) + 1\right) + \log_2\left(\frac{2}{\delta}\right)\right] \tag{1.5}$$

holds with probability of at least $1 - \delta$ for $N > \vartheta$.

This theorem can be easily derived from Theorem 1.1. Note, these bounds are only examples and similar formulations are available for other loss functions [Vapnik, 1998] and other complexity measures [see e.g. Williamson et al., 1998].

2. Theorem 1.3 is the noise-free version of Theorem 1.2. The square root makes (1.4) considerable weaker than (1.5). Generally, bounds considering the risk, while assuming that the empirical risk is zero, are called PAC bounds, whereas VC bounds allow the empirical risk to be non-zero [Herbrich, 2001].

Let us discuss inequality (1.4): the goal is to minimize the generalization error $R[f]$, which can be achieved by obtaining a small training error $R_{\mathrm{emp}}[f, S]$, while keeping the function class as small as possible. Two extremes arise for (1.4): (i) a very small function class (like F_1) yields a vanishing square root term, but a large training error might remain, while (ii) a huge function class (like F_K) may give rise to a vanishing empirical error, but a large square root term. The best class is usually in between (cf. Figure 1.2), as one would like to obtain a function that explains the data well *and* to have a small risk in obtaining that function.

1.3 Boosting and Support Vector Machines

Within the last decade, *large margin classification techniques* have emerged as a practical result of the theory of generalization. The two most widely studied classes of large margin classifiers are *Support Vector Machines* (SVMs) [Boser et al., 1992] and *Boosting* [Schapire, 1992]. In this section we briefly review both algorithms and their theoretical motivation through statistical learning theory. In the last section we give a generalized margin definition and state some interesting connections between Boosting and SVMs.

1.3.1 PAC Boosting

Let us start with a very brief review of PAC boosting, which does not claim to be complete – for more details see e.g. Schapire [1990], Anthony and Biggs [1997], Freund and Schapire [1997], Schapire et al. [1998], Schapire [1999], Duffy and Helmbold [2000b].

Let us consider the PAC learning framework introduced in Section 1.2.2 again. The conditions on the (strong) PAC learner seem to be overly strong, since in practice one will often not be able to generate a hypothesis that has arbitrarily small expected risk. One therefore considers algorithms that with a seemingly much weaker requirement:[3]

Definition 1.2 (Kearns and Vazirani [1994]). *A* weak PAC learner *is similar to a strong PAC learner, except that it only satisfies the PAC condition for a particular* $0 < \epsilon_0, \delta_0 \leq \frac{1}{2}$ *pair, rather than for all* ϵ, δ *pairs.*

Thus, an algorithm that is only slightly better than random guessing on every distribution over the domain $\mathbb{X}$ is called a weak PAC learner. This is about the least assumption one can require on a learning algorithm [Freund and Schapire, 1997].

Interestingly, it has been shown that *any* weak PAC learner can be "boosted" to meet the strong PAC learning criteria. The first algorithm being able to implement this was proposed in Schapire [1992] – it was called *Boosting*.[4] The algorithm was able to increase the performance of a weak PAC learner such that the resulting (hybrid) algorithm satisfies the strong PAC learning criteria. Note, that it has already been known before that improving the confidence parameter δ is relatively easy [see e.g. Haussler et al., 1991, Freund, 1995, Anthony and Biggs, 1997], whereas it was an unanswered question whether one can improve the accuracy ϵ efficiently.

3. This definition is again simplified (cf. footnote 1).
4. Methods building strong PAC learning algorithms from weak PAC learning algorithms are called PAC boosting algorithms [Duffy and Helmbold, 2000b].

Later, an improved PAC boosting algorithm was found – called AdaBoost [Freund and Schapire, 1997] – which repeatedly calls a given "weak learner" (also called: base learning algorithm) $\mathcal{L}$ and finally produces a master hypothesis f, which is a linear combination of T functions h_t, $t = 1, \ldots, T$, produced by the base learning algorithm, i.e.

$$f(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}). \tag{1.6}$$

The weak learner $\mathcal{L}$ is called in each iteration t with different distributions (also weightings) $\mathbf{d}^{(t)} = (d_1^{(t)}, \ldots, d_N^{(t)})$ (where $\sum_{n=1}^{N} d_N^{(t)} = 1$, $d_n^{(t)} \geq 0, n = 1, \ldots, N$) on the training set, which are chosen in such a way that examples classified poorly are more emphasized than other examples. Roughly speaking, AdaBoost exploits the fact that the weak learner has to be better than random guessing on *any* distribution on the examples – also on those "hard" distributions generated by AdaBoost.[5] The pseudo-code of the AdaBoost algorithm is given in Algorithm 1.1.

Algorithm 1.1 The AdaBoost algorithm [Freund and Schapire, 1997].

1. **Input:** $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, Number of Iterations T
2. **Initialize:** $d_n^{(1)} = 1/N$ for all $n = 1, \ldots, N$
3. **Do for** $t = 1, \ldots, T$,
   (a) Train classifier with respect to the weighted sample set $\{S, \mathbf{d}^{(t)}\}$ and obtain hypothesis $h_t : \boldsymbol{x} \mapsto \{-1, +1\}$, i.e. $h_t = \mathcal{L}(S, \mathbf{d}^{(t)})$
   (b) Calculate the weighted training error ϵ_t of h_t:

$$\epsilon_t = \sum_{n=1}^{N} d_n^{(t)} \mathbf{I}(y_n \neq h_t(\boldsymbol{x}_n)) \ , \tag{1.7}$$

   (c) Set

$$\alpha_t = \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t} \tag{1.8}$$

   (d) Update weights:

$$d_n^{(t+1)} = d_n^{(t)} \exp\left\{-\alpha_t y_n h_t(\boldsymbol{x}_n)\right\} / Z_t \ , \tag{1.9}$$

   where Z_t is a normalization constant, such that $\sum_{n=1}^{N} d_n^{(t+1)} = 1$.
4. **Break if** $\epsilon_t = 0$ or $\epsilon_t \geq \frac{1}{2}$.
5. **Output:** $f(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})$

The intuitive idea behind boosting is that one can combine very simple *rules of thumb* to construct a combined hypothesis with arbitrary good performance. It was indeed not

5. To distinguish between both weightings we will call the weighting on the example "distribution" or just "sample weights" and the weighting on the hypotheses we will call the "hypothesis coefficients".

obvious that this is possible. Figure 1.3 illustrates how AdaBoost works. Examples that are misclassified get higher weights in the next iteration. The examples near the decision boundary are usually harder to classify and therefore get high weights after a few iterations [e.g. Rätsch et al., 1999].

One of the most important properties of AdaBoost is its fast convergence to a hypothesis, which is consistent with the training sample, if the base learning algorithm produces hypotheses with error rates consistently smaller than $\frac{1}{2}$:

Theorem 1.4 (Exponential Convergence, Freund and Schapire [1997]). *Assume, $\epsilon_1, \ldots, \epsilon_T$ are the weighted classification errors of $h_1, \ldots, h_T$ that are generated by running AdaBoost. Then the error on the training set is bounded above by:*

$$\frac{1}{N} \sum_{n=1}^{N} \mathbf{I}\left(y_n \neq \tilde{f}(\boldsymbol{x}_n)\right) \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t(1-\epsilon_t)}\,, \tag{1.10}$$

where $\tilde{f} = \operatorname{sign}(f)$ is the final hypothesis.

For instance, if the error rate is in each iteration bounded from above by $\epsilon_t \leq \frac{1}{2} - \frac{1}{2}\gamma$, $t = 1, 2, \ldots$, for some $\gamma > 0$, then the training error decreases exponentially in the number of iterations:

$$\frac{1}{N} \sum_{n=1}^{N} \mathbf{I}\left(y_n \neq \tilde{f}(\boldsymbol{x}_n)\right) \leq \exp\left(-\frac{T\gamma^2}{2}\right).$$

Thus, to find a consistent hypothesis one needs only $\frac{2\log(N)}{\gamma^2}$ iterations. This property can be exploited for proving the PAC bounds presented in the last sections:[6] Let us assume the base learner chooses hypotheses from a hypothesis space H with VC dimension ϑ. Then the VC dimension of the function class of linear combinations of T functions from H is $\mathcal{O}(T\vartheta\log(T))$ [Baum and Haussler, 1989, Blumer et al., 1989]. If we assume that $\mathcal{L}$ is a PAC weak learner, then there exists a uniform $\gamma > 0$ for all sample sizes N, such that the training errors are smaller than $\frac{1}{2} - \frac{1}{2}\gamma$ (by definition).[7] The number of hypotheses that need to be combined is $\mathcal{O}(\log(N)/\gamma^2)$ and hence the VC dimension of the consistent combined classifier is at most

$$\vartheta(N,\gamma) = \mathcal{O}\left(\frac{\vartheta\log(N)}{\gamma^2}\log\left(\frac{\log(N)}{\gamma^2}\right)\right).$$

Hence, the VC dimension increases roughly logarithmically in the number of training examples, and thus increases much slower than the number of examples. We can therefore use the Theorem 1.1 to show that if the sample size is large enough N, i.e. satisfying

$$N \geq \mathcal{O}\left(\frac{\vartheta(N,\gamma)}{\epsilon}\log_2\left(\frac{1}{\epsilon}\right) + \frac{1}{\epsilon}\log_2\left(\frac{1}{\delta}\right)\right),$$

6. For simplicity we leave out many minor details that would make the presentation much heavier.
7. We assume here that the base learner *always* achieves this error rate, i.e. the confidence parameter is assumed to be very close to 1 [see Haussler et al., 1991, Freund, 1995].

with probability at least $1-\delta$, the combined hypothesis f produced by AdaBoost $f \in \mathsf{F}$ has risk smaller than ϵ, i.e. $R[f] \leq \epsilon$. Such N exists, since the right hand side (rhs.) of the inequality is roughly $\mathcal{O}(\log(N))$, whereas the left hand side (lhs.) is N. *This argumentation [Schapire, 1992] shows that any weak learner can be transformed into a strong learner!*

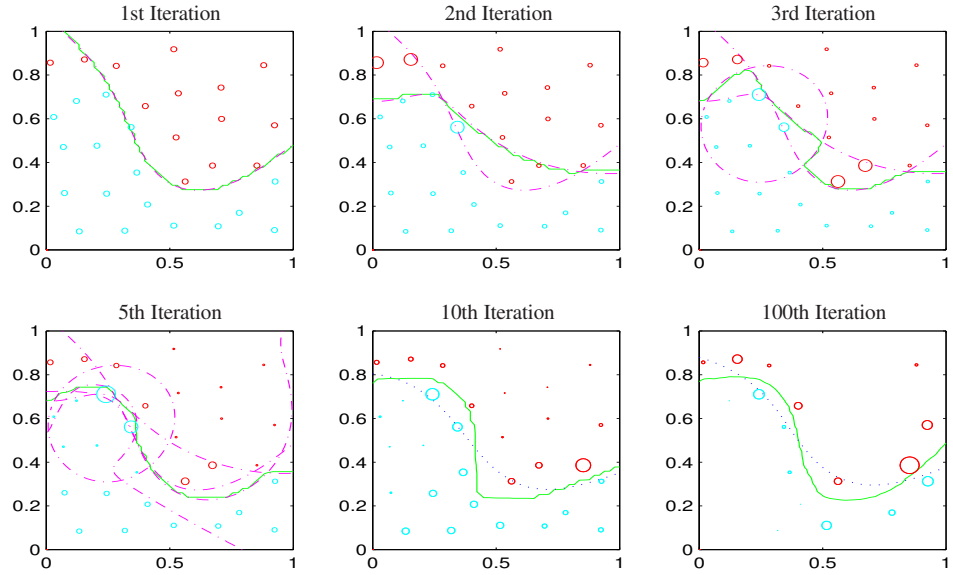

Figure 1.3 Illustration of AdaBoost on a 2D toy data set: The color indicates the label and the diameter is proportional to the weight of the examples in the first, second, third, 5th, 10th and 100th iteration. The dash-dotted lines show the decision boundaries of the single classifiers (up to the 5th iteration). The solid line shows the decision line of the combined classifier. In the last two plots the decision line of Bagging is plotted for a comparison. (Figure taken from Rätsch et al. [2001].)

The VC complexity of the function class (of combined hypotheses) grows in each iteration. However, it has been noticed [Schapire et al., 1998] that in some cases the generalization keeps decreasing – clearly contradicting the PAC analysis presented above. As a response to this empirical finding, a second explanation based on margins has been proposed by Schapire et al. [1998]. Their main result is a bound on the generalization error $P(\mathsf{Y} \neq \operatorname{sign}(f(\mathsf{X})))$ depending on the VC dimension ϑ of the base hypotheses class H and on the *margins* on the training set. The margin ρ_n of an example $\boldsymbol{x}_n$ is defined in terms of the *normalized* function value and the label:

$$\rho_n = y_n \frac{f(\boldsymbol{x}_n)}{\sum_{t=1}^{T} \alpha_t}, \tag{1.11}$$

where $\alpha_t \geq 0$, $t = 1, \ldots, T$, are the hypothesis coefficients of the combined hypothesis f, which are assumed to be non-negative. If the margin is positive, the example is classified correctly.

It has been found [Schapire et al., 1998] that with probability at least $1-\delta$, the expected

risk of a function f with margins $\rho_1, \ldots, \rho_N$ on the training set can be upper bounded by

$$R[f] \leq \sum_{n=1}^{N} \mathbf{I}(\rho_n \leq \theta) + \mathcal{O}\left(\sqrt{\frac{\vartheta \log^2(N/\vartheta)}{N\theta^2} + \frac{\log(1/\delta)}{N}}\right), \tag{1.12}$$

for any $\theta \in (0, 1]$. Instead of using the empirical risk, one uses the so-called *margin risk*, which is the fraction of examples with margin smaller or equal to some θ. The proof of (1.12) is based on results from Vapnik and Chervonenkis [1971], Devroye [1982] (see also Shawe-Taylor et al. [1996], Bartlett [1998]) and the idea is that the effective size of the function class is much smaller if there is a separating margin between the classes. This idea is illustrated in Figure 1.5.

In [Schapire et al., 1998] it was stated that the reason for the success of AdaBoost, compared to other ensemble learning methods, in particular Bagging [Breiman, 1996], is that it produces hypotheses with large margins on all examples, say larger than some θ. Then the first term on the rhs. of (1.12) is zero, while the second term is small (and the larger θ, the smaller is the second term).

There is an important difference to the other bound: whereas one requires that the number of combined hypotheses is small for Theorem 1.3, here the size of the ensemble is not important at all and could even be infinite. The important quantity is the margin. The second term of the bound would be smallest, if the margins generated by AdaBoost would be as large as possible. However, it is not clear whether AdaBoost actually achieves the maximum margin solution. We therefore consider the relation of AdaBoost to margin maximization in greater detail in the next chapter.

Remark 1.5. *Traditional statistical techniques often have the problem of the* curse of dimensionality.[8] *Here the bound on the generalization error (1.12) only depends on the margin and not on the dimensionality of the input space. Hence one is able to handle arbitrarily high dimensional data and learned efficiently if the data can be separated with large margin.*

In the next section we briefly review another very successful machine learning algorithm strongly motivated by the idea of large margins and finally in Section 1.3.3 we define the margin more formally.

1.3.2 Support Vector Machines

Let us for a moment assume that the training sample is separable by a *hyperplane* and we choose functions of the form

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b. \tag{1.13}$$

8. Curse of dimensionality [Bellman, 1961] refers to the exponential growth of hyper-volume as a function of dimensionality. To cover this space uniformly one would need exponentially many examples. Many learning algorithms suffer from the *curse of dimensionality* as they do not generalize well in high dimensions [cf. Scott, 1992, Bishop, 1995].

It was shown [e.g. Vapnik and Chervonenkis, 1974, Vapnik, 1995] that for the class of hyperplanes the VC dimension ϑ can be bounded in terms of the *margin* (see Figure 1.4). Here, the margin is defined as the minimal Euclidean distance of an example to the hyperplane.[9] The margin in turn can be measured by the length of the weight vector $\boldsymbol{w}$ in (1.13): as we assumed that the training sample is separable we can rescale $\boldsymbol{w}$ and b such that the examples closest to the hyperplane satisfy $|\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b| = 1$, $n = 1, \ldots, N$, i.e. obtain the so-called canonical representation of the hyperplane. Now consider two examples $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ from different classes with $\langle \boldsymbol{w}, \boldsymbol{x}_1 \rangle + b = 1$ and $\langle \boldsymbol{w}, \boldsymbol{x}_2 \rangle + b = -1$, respectively. Then the margin of the hyperplane is given by a half the distance of these two examples, measured perpendicular to the hyperplane, i.e. $\frac{1}{2} \left\langle \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}, \boldsymbol{x}_1 - \boldsymbol{x}_2 \right\rangle = \frac{1}{\|\boldsymbol{w}\|}$ (cf. Figure 1.4).

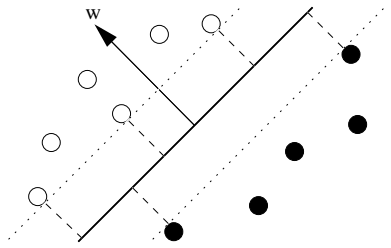

Figure 1.4 Linear classifier and margins: A linear classifier is defined by a hyperplane's normal vector $\boldsymbol{w}$ and an offset b, i.e. the decision boundary is $\{\boldsymbol{x} | \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0\}$ (thick line). Each of the two halfspaces defined by this hyperplane corresponds to one class, i.e. $f(\boldsymbol{x}) = \mathrm{sign}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b)$. The margin of a linear classifier is the minimal distance of any training examples to the hyperplane. In this case it is the distance between the dotted lines and the thick line. (Figure taken from Müller et al. [2001].)

The result linking the VC dimension of the class of separating hyperplanes to the margin or the length of the weight vector $\boldsymbol{w}$, respectively, is given by the following inequalities:

$$\vartheta \leq \Lambda^2 R^2 + 1 \quad \text{and} \quad \|\boldsymbol{w}\|_2 \leq \Lambda \tag{1.14}$$

where R is the radius of the smallest ball around the data [e.g. Vapnik, 1995]. Thus, if we bound the margin of a function class from below, say by $\frac{2}{\Lambda}$, we can control its VC dimension.[10] The bound (1.14) is illustrated in Figure 1.5.

Using this fact one can readily derive a bound on the generalization error from Theorem 1.2: With probability $1 - \delta$, the empirical risk can be bounded from above by

$$R[f] \leq R_{\mathrm{emp}}[f, S] + \mathcal{O}\left(\sqrt{\frac{R^2\Lambda^2}{N} \left(\log_2 \left(\frac{N}{R^2\Lambda^2} \right) + 1 \right) + \frac{\log_2(1/\delta)}{N}} \right), \tag{1.15}$$

where f is a linear hyperplane as in (1.13) with $\|\boldsymbol{w}\|_2 \leq \Lambda$ and satisfies

$$y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b) \geq 1, \quad n = 1, \ldots, N. \tag{1.16}$$

9. In Section 1.3.3 we will in fact show that one uses a different distance in Boosting.

10. There are some ramifications to this statement that go beyond the scope of this presentation. Strictly speaking, VC theory requires the structure to be defined a priori, which has implications for the definition of the class of separating hyperplanes [cf. Shawe-Taylor et al., 1996].

The goal of learning is to find $\boldsymbol{w}$ and b such that the expected risk is minimized. However, since one cannot obtain the expected risk itself, one minimizes the bound (1.15), which consists of the empirical risk and the complexity term.

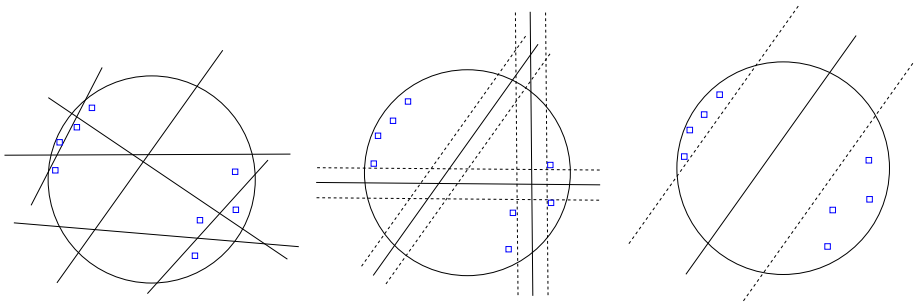

Figure 1.5 Complexity of hyperplanes: The unlabeled examples are contained in a ball. For hyperplanes with no margin [left] exist many ways to label the training examples (up to about $\left(\frac{e\,N}{s+1}\right)^{s+1}$ different dichotomies, where s is the dimensionality of $\mathbb{X}$). If the margin is greater than zero, the number of possibilities is reduced [middle]. If the margin is huge, there is only one way to label the data [right]. A small number of dichotomies implies a small VC dimension of the function class.

One strategy is to keep the empirical risk zero by constraining $\boldsymbol{w}$ and b to the perfect separation case, while minimizing the complexity term, which is a monotonically increasing function of the VC dimension ϑ. Thus, we can minimize the complexity term by minimizing $\|\boldsymbol{w}\|^2$. This can be formulated as a quadratic optimization problem [Boser et al., 1992]

$$\min_{\boldsymbol{w},b} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 \quad \text{subject to (1.16).} \tag{1.17}$$

The result is the so-called the *maximum margin hyperplane* (cf. Figure 1.4).

Up to now we only considered the separable case. This corresponds to an empirical error of zero (cf. (1.15)). However for "noisy data" there might not exist a consistent hyperplane. Then there is no solution to (1.17). Also, even if there exists a separation, one might not obtain a minimum in the expected risk and face overfitting effects, since the margin is too small. Therefore a "good" trade-off between the empirical risk and the square root complexity term in (1.15) needs to be found. Using a technique, which was first proposed in Bennett and Mangasarian [1992] and later used for SVMs in Cortes and Vapnik [1995], one introduces *slack-variables* to relax the hard-margin constraints:

$$y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n\rangle + b) \geq 1 - \xi_n, \quad \xi_n \geq 0, \quad n = 1, \ldots, N, \tag{1.18}$$

additionally allowing for some classification errors. The SVM solution can then be found by (a) keeping the upper bound on the VC dimension small and (b) by minimizing the upper bound $\sum_{n=1}^{N} \xi_n$ on the empirical risk, i.e. the number of training errors. Thus, one

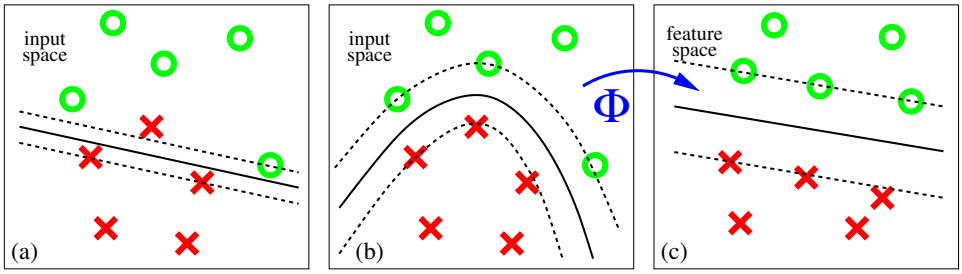

Figure 1.6 Three different views on the same dot versus cross separation problem. The examples closest to the separation line are called support vectors. (a) In this example, a linear separation of the examples is not possible without errors. Even the misclassification of one example permits only a small margin. The resulting linear classification function looks inappropriate for the data. (b) A better separation is permitted by nonlinear surfaces in input space. (c) These nonlinear surfaces correspond to linear surfaces in feature space. The examples are mapped from input space to feature space by the function Φ that is implied by the kernel function k. (Figure taken from Zien et al. [2000].)

minimizes

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N}\xi_n \tag{1.19}$$

subject to (1.18), where the regularization constant $C > 0$ determines the trade-off between the empirical error and the complexity term.

The choice of linear functions seems to be very limiting (consider for instance the *xor*-problem). Instead of being likely to overfit we are now more likely to underfit. Fortunately there is a way to have both, the theory for linear models *and* a very rich set of nonlinear decision functions, by using the "kernel-trick": One first maps the data into a high, possibly infinite dimensional feature space $\mathbb{F}$ using a mapping $\Phi : \mathbb{X} \to \mathbb{F}$ and then one finds the linear discrimination in $\mathbb{F}$. By using *kernels*, one can compute dot products in feature space efficiently: $\mathrm{k}(\boldsymbol{x}_n, \boldsymbol{x}_m) = \langle(\Phi(\boldsymbol{x}_n), \Phi(\boldsymbol{x}_m)\rangle$. By Mercer's theorem [Mercer, 1909], any kernel of a positive integral operator can be expanded in its Eigenfunctions ψ_j ($\lambda_j > 0$, $N_{\mathbb{F}} \leq \infty$):

$$\mathrm{k}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{j=1}^{N_{\mathbb{F}}} \lambda_j \psi_j(\boldsymbol{x})\psi_j(\boldsymbol{y}).$$

In this case $\Phi(\boldsymbol{x}) = (\sqrt{\lambda_1}\psi_1(\boldsymbol{x}), \sqrt{\lambda_2}\psi_2(\boldsymbol{x}), \dots)$ is a possible realization of the mapping. The SVM algorithm can be entirely formulated in terms of scalar products of the examples. Therefore one only needs to replace the scalar products with scalar products in feature space, i.e. by the kernel. Since the bounds presented above do not depend on the dimensionality of the space where the linear discrimination is performed, this transformation is reasonable and leads to a non-linear discrimination algorithm derived from statistical learning theory.

1.3.3 p-Norm Margins in Feature Space

Let us define the term "margin" more formally. To start with, we give some standard definitions and results for the *margin* of an example and of a hyperplane. In the next section we will discuss some consequences of using different norms for measuring the margin in feature space. Suppose we are given a sample of N examples in some space $\mathbb{X}$: $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\} \subseteq \mathbb{X} \times \{-1, 1\}$. For simplicity we assume here that $\mathbb{X}$ is finite dimensional. We are interested in the separation of the training set using a hyperplane $A = \{\tilde{\boldsymbol{x}} \mid \langle \tilde{\boldsymbol{x}}, \hat{\boldsymbol{\alpha}} \rangle + b = 0\}$ in $\mathbb{X}$ determined by some vector $\hat{\boldsymbol{\alpha}}$ and the bias b. We assume $\hat{\boldsymbol{\alpha}}$ is normalized with respect to some norm. The *margin* of an example $(\boldsymbol{x}_n, y_n)$ with respect to the hyperplane A is defined as $y_n \langle \boldsymbol{x}_n, \hat{\boldsymbol{\alpha}} \rangle$ (cf. Sections 1.3.1–1.3.2). A positive margin corresponds to a correct classification and the more positive the margin, the greater the confidence that the classification is correct.

The margin has been frequently used in the context of Support Vector Machines (SVMs) and Boosting. These so-called *large margin algorithms* are focusing on generating hyperplanes/functions with large margins on most training examples. Let us therefore study some properties of the maximum margin hyperplane.

For notational convenience we introduce the matrix $U \in \mathbb{R}^{N \times J}$, where $U_{nj} = y_n \boldsymbol{x}_{n,j}$. The n-th example $(\boldsymbol{x}_n, y_n)$ corresponds to the n-th row $U_{n,*}$ and the j-th dimension of $\boldsymbol{x}_n$ corresponds to the j-th column $U_{*,j}$ of U multiplied with the label. Using this notation, the ℓ_p-margin of the n-th example is

$$\rho_n^p(\hat{\boldsymbol{\alpha}}) := \frac{U_{n,*}\hat{\boldsymbol{\alpha}}}{\|\hat{\boldsymbol{\alpha}}\|_p},$$

where the subscript $p \in [1, \infty]$ specifies with respect to which norm $\hat{\boldsymbol{\alpha}}$ is normalized (the default is $p = 1$). The ℓ_p-margin of the hyperplane A is defined as the minimum margin over all N examples, i.e.

$$\rho^p(\hat{\boldsymbol{\alpha}}) = \min_{n=1,\ldots,N} \rho_n^p(\hat{\boldsymbol{\alpha}}) = \min_{n=1,\ldots,N} \frac{U_{n,*}\hat{\boldsymbol{\alpha}}}{\|\hat{\boldsymbol{\alpha}}\|_p}. \tag{1.20}$$

To maximize the margin of the hyperplane one has to solve the following convex optimization problem [Mangasarian, 1965, Boser et al., 1992]:

$$\max_{\hat{\boldsymbol{\alpha}}, \rho} \rho \quad \text{with} \quad \frac{U_{n,*}\hat{\boldsymbol{\alpha}}}{\|\hat{\boldsymbol{\alpha}}\|_p} \geq \rho, \quad n = 1, \ldots, N. \tag{1.21}$$

In both problems one finds a hyperplane that maximizes the margin. The following theorem gives a geometrical interpretation for (1.21): Using the ℓ_p-norm to normalize $\hat{\boldsymbol{\alpha}}$ corresponds to measuring the distance to the hyperplane with the dual ℓ_q-norm, where $1/p + 1/q = 1$.

Theorem 1.6 (p-norm Projections, Mangasarian [1999]). *Let $\boldsymbol{x} \in \mathbb{R}^J$ be any point which is not on the plane $A := \{\tilde{\boldsymbol{x}} \mid \langle \tilde{\boldsymbol{x}}, \hat{\boldsymbol{\alpha}} \rangle + b = 0\}$. Then for $p \in [1, \infty]$:*

$$\frac{|\langle \boldsymbol{x}, \hat{\boldsymbol{\alpha}} \rangle + b|}{\|\hat{\boldsymbol{\alpha}}\|_p} = \|\boldsymbol{x} - A\|_q = \min_{\tilde{\boldsymbol{x}} \in A} \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_q, \tag{1.22}$$

where $\|\boldsymbol{x} - A\|_q$ denotes the distance of $\boldsymbol{x}$ to the plane A measured with the dual norm ℓ_q.

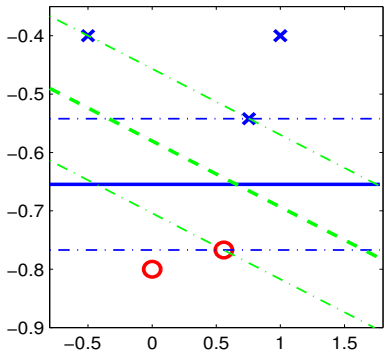

Figure 1.7 The maximum margin solution for different norms on a toy example: ℓ_∞-norm (solid) and ℓ_2-norm (dashed). The margin areas are indicated by the dash-dotted lines. The examples with label $+1$ and -1 are shown as '$\circ$' and '$\square$', respectively. To maximize the ℓ_∞-norm and ℓ_2-norm margin, we solved (1.21) with $p = 1$ and $p = 2$, respectively.

Thus, the ℓ_p-margin of $\boldsymbol{x}_n$ is the signed ℓ_q-distance of the point to the hyperplane. If the point is on the right side of the hyperplane, the margin is positive. Figure 1.7 illustrates Theorem 1.6 for $p = 1$ and $p = 2$.

1.3.4 Boosting vs. SVMs

For the definition of the margin we considered linear separations only. We have already shown that in the case of SVMs one can still use this interpretation for non-linear classifiers, by mapping the data first into some feature space $\mathbb{F}$, i.e. $\hat{\boldsymbol{x}} = \Phi(\boldsymbol{x})$. Since one uses the ℓ_2-norm, the computation of the separating hyperplane can be expressed in terms of scalar products in feature space (cf. (1.22)). These are then replaced by the kernel function and the feature space is implicitly defined by the kernel. It is therefore often called *kernel feature space*.

Conversely, in Boosting one is interested in the ℓ_1-margin (cf. (1.11)). If it is large, then one can guarantee good generalization (cf. (1.12)). Here one considers a different feature space, which is defined by the hypothesis set H [cf. Schapire et al., 1998, Freund and Schapire, 1999b]. Each hypothesis from H defines one dimension in this space, i.e.

$$\hat{\boldsymbol{x}}_n = \Phi(\boldsymbol{x}_n) = \begin{bmatrix} \hat{h}_1(\boldsymbol{x}_n) \\ \vdots \\ \hat{h}_J(\boldsymbol{x}_n) \end{bmatrix}, \tag{1.23}$$

if one assumes that the hypothesis set is finite and has J elements. The feature space induced by the hypothesis set, we will therefore call the *hypothesis feature space*.

In principle, however, one could define a kernel by explicitly computing the dot product in feature space, i.e. $k(\boldsymbol{x}, \boldsymbol{x}') = \sum_{j=1}^{J} \hat{h}_j(\boldsymbol{x})\hat{h}_j(\boldsymbol{x}')$, and one might ask, why one not just measures the ℓ_2-margin and then applies the bound (1.15). Whereas this might in principle be possible in finite dimensional spaces, its imposibility becomes immediately apparent for infinite hypothesis spaces: If one assumes that the output of the base hypotheses is binary (i.e. -1 or $+1$), then they lie on the vertices of some hypercube in feature space. If the feature space is infinite dimensional, then there exists no ball around the data and one cannot apply (1.15). However, if one measures distances with the ℓ_∞-norm, they will stay finite. Hence, one uses the ℓ_1-margin.

So both bounds, say (1.12) and (1.15), based on large margins exploit the particular structure of the feature space. For (1.12) one needs that the ℓ_∞-norm is bounded, whereas for (1.15) one assumes that the ℓ_2-norm is small. In Section 5.2.3 we will consider this issue again and show that the ℓ_1-norm regularization is indeed very well suited for ensemble learning and makes the design of efficient algorithms possible.

2 Boosting vs. Margin Maximization

In this chapter we analyze AdaBoost in the context of large margin algorithms. In particular, we try to shed light on the question whether and under which conditions boosting yields large margins. In Freund and Schapire [1994] it has been shown that AdaBoost finds quickly a combined hypothesis that is consistent with the training data. Schapire et al. [1998] and Breiman [1999] indicated that AdaBoost computes hypotheses with large margins, if one continues iterating after reaching consistency. It is arguable that the margin should be as large as possible on most training examples in order to minimize the complexity term in (1.12). If one assumes that the base learner always achieves a weighted training error $\epsilon_t \leq \frac{1}{2} - \frac{1}{2}\hat{\gamma}$ with $\hat{\gamma} > 0$, then AdaBoost generates a hypothesis with margin larger than $\frac{1}{2}\hat{\gamma}$ [Schapire et al., 1998, Breiman, 1999]. However, from the Min-Max theorem of linear programming [von Neumann, 1928] one finds that the maximal achievable margin ρ^* is at least $\hat{\gamma}$ [Freund and Schapire, 1996a, Breiman, 1999, Freund and Schapire, 1999b]. So there is gap in the theory to explain how large the margin of AdaBoost's combined hypotheses are.

We start with a brief review of some standard definitions and results for the margin of an example and of a hyperplane. Then we analyze the asymptotical properties of a slightly more general version of AdaBoost, called AdaBoost$_\varrho$, which is equivalent to AdaBoost for $\varrho = 0$, while assuming that the problem is separable. We show that there will be a subset of examples – the *support vectors* – asymptotically having the same smallest margin. On these examples asymptotically all weights $\mathbf{d}$ are concentrated. Furthermore, we find that AdaBoost$_\varrho$ is able to achieve larger margins than AdaBoost for $0 < \varrho < \rho^*$. We derive a slightly better lower bound on the margin of AdaBoost$_\varrho$, which is tight in the worst case.

However, empirically we find that if the base learning algorithm performs optimal, AdaBoost seems to approximate the maximum margin solution well, i.e. almost achieves a margin of ρ^*. By observing that the difference between worst and best performance decreases as the parameter ϱ approaches ρ^* from below, we propose an algorithm, called *Marginal Boosting*, that iteratively adapts ϱ. We can show that this algorithm actually maximizes the margin and comes with similar rates of convergence as AdaBoost.

Finally, in the last section we give a different kind of analysis based on barrier optimization. We show that boosting algorithms can be understood as particular implementations for minimizing a barrier function to solve a constraint optimization problem. Although this point of view does not give immediate new results, it gives some insights, why boosting-type algorithms tend to generate hypotheses with large margins and how they relate to convex optimization. These results will be exploited in later chapters to derive similar algorithms based on different optimization problems.

2.1 von Neumanns Min-Max Theorem

To start with we briefly define the terms *margin* and *edge*. A more general definition along with relations to the margin definition in SVMs is given Section 1.3.3.

It has been shown that AdaBoost finds a combined hypothesis that corresponds to a hyperplane with large margin in feature space $\mathbb{F}$, which is defined by the hypothesis space H [Schapire et al., 1998, Breiman, 1999]. The hyperplane $A = \{\tilde{\boldsymbol{x}} \mid \langle \tilde{\boldsymbol{x}}, \hat{\boldsymbol{\alpha}} \rangle = 0, \tilde{\boldsymbol{x}} \in \mathbb{F}\}$ is determined by some weight vector $\hat{\boldsymbol{\alpha}}$, where we assume that $\|\hat{\boldsymbol{\alpha}}\|_1 = 1$. The ℓ_1-*margin* of an example $(\boldsymbol{x}_n, y_n)$ with respect to the hyperplane A is defined as the *signed* ℓ_∞-distance of the example to the hyperplane in feature space (definition see below).[1] A positive margin corresponds to a correct classification and the more positive the margin the greater the confidence that the classification is correct.

For simplicity, we assume in this section that the hypothesis space H finite.[2] Then the feature space spanned by the base hypotheses is finite, say, J-dimensional and one may define the matrix $U \in \mathbb{R}^{N \times J}$, where $U_{nj} = y_n \hat{h}_j(\boldsymbol{x}_n)$. Let $U_{n,*}$ and $U_{*,j}$ denote the n-th row and j-th column of U, respectively. Using this notation, the margin of the n-th example with respect to a hyperplane with normal vector $\hat{\boldsymbol{\alpha}}$ is $\rho_n(\hat{\boldsymbol{\alpha}}) := U_{n,*} \frac{\hat{\boldsymbol{\alpha}}}{\|\hat{\boldsymbol{\alpha}}\|_1}$. The margin ρ of the hyperplane is defined as the minimum margin over all N examples, i.e.

$$\rho(\hat{\boldsymbol{\alpha}}) = \min_{n=1,\ldots,N} \rho_n(\hat{\boldsymbol{\alpha}}) = \min_{n=1,\ldots,N} U_{n,*} \frac{\hat{\boldsymbol{\alpha}}}{\|\hat{\boldsymbol{\alpha}}\|_1}. \tag{2.1}$$

Boosting algorithms maintain a probability weighting $\mathbf{d} \in \mathcal{P}^N$ on the examples, where $\mathcal{P}^N$ is the N dimensional probability simplex, i.e. $\mathcal{P}^N := \{\mathbf{d} \mid \mathbf{d} \in \mathbb{R}_+^N, \sum_{n=1}^N d_n = 1\}$. Assume for a moment that the outputs of the hypotheses are binary, i.e. $\hat{h}_j(\boldsymbol{x}_n) \in \{\pm 1\}$. Then for a distribution $\mathbf{d} \in \mathcal{P}$, the dot product $\langle U_{*,j}, \mathbf{d} \rangle = \sum_{n=1}^N d_n y_n \hat{h}_j(\boldsymbol{x}_n)$ is the expectation that $\hat{h}_j$ predicts the $\{\pm 1\}$ label correctly.[3] This is called the *edge* of the hypothesis $\hat{h}_j$ [cf. Breiman, 1999]. Note that a "hypothesis" that randomly guesses, has an expected edge of 0, if one assumes that the classes are equally likely. We define the *edge of the probability vector* $\mathbf{d}$ as the maximum edge over the set of hypotheses, i.e.

$$\gamma(\mathbf{d}) = \max_{j=1,\ldots,J} \langle U_{*,j}, \mathbf{d} \rangle = \max_{j=1,\ldots,J} \sum_{n=1}^N d_n y_n \hat{h}_j(\boldsymbol{x}_n). \tag{2.2}$$

Roughly speaking, Boosting aggressively chooses the distribution in each iteration such that it is hard for the weak learner to return a hypothesis with large edge [Freund and Schapire, 1997].

---

1. Since the hypothesis coefficient vector $\hat{\boldsymbol{\alpha}}$ is normalized with respect to the ℓ_1-norm, we call it ℓ_1-norm margin; however, the margin is measured with the ℓ_∞-norm.
2. If the outputs of each base hypothesis are binary, then there is only a finite number of *distinct* hypotheses. If the hypothesis are real valued the hypothesis space can be infinite, in particular in regression. Then the linear programming problems become semi-infinite problems. The necessary theory for dealing with semi-infinite linear programs is given in Section 5.3 for regression problems. It can be transferred to the classification case (cf. Appendix B.4 and Rätsch and Warmuth [2001].
3. Here correct means $+1$ and incorrect -1. If correct is encoded as $+1$ and incorrect as 0, then it would become $\frac{1}{2}(1 - \langle U_{*,j}, \mathbf{d} \rangle)$.

We start with the connection between *maximal margin* and *minimal edge* using the duality theorem for Linear Programs (LPs) [e.g. Nash and Sofer, 1996] that was originally proved by von Neumann [1928] and first used in connection with Boosting in Freund and Schapire [1996a], Grove and Schuurmans [1998], Breiman [1999]:

Theorem 2.1 (von Neumann [1928]). $\max_{\hat{\boldsymbol{\alpha}} \geq \mathbf{0}} \rho(\hat{\boldsymbol{\alpha}}) = \min_{\mathbf{d} \in \mathcal{P}^N} \gamma(\mathbf{d})$.

Both sides in Theorem 2.1 are equivalent to the following linear programming problems:

$$\begin{array}{lll} & \max_{\rho, \hat{\boldsymbol{\alpha}}} \rho & \min_{\gamma, \mathbf{d}} \gamma \\ \text{with} & \hat{\boldsymbol{\alpha}} \geq \mathbf{0}, \sum_{j=1}^J \hat{\alpha}_j = 1 & \mathbf{d} \geq \mathbf{0}, \sum_{n=1}^N d_n = 1 \\ & U\hat{\boldsymbol{\alpha}} \geq \rho \mathbf{1} & U^\top \mathbf{d} \leq \gamma \mathbf{1} \\ & \text{margin-LP problem} & \text{edge-LP problem} \end{array} \tag{2.3}$$

which are dual to each other. Thus the equality in the theorem follows from the fact that the primal and the dual objective have the same value at optimality [shown later in general, cf. e.g. Bertsekas, 1995, Propositon 5.2.1].

This has an immediate consequence for Boosting: Theorem 2.1 states that if the weak learner has accuracy of at least $\frac{1}{2} - \frac{1}{2}\gamma$ (i.e. edge greater that γ; cf. Definition 1.2), then there exists a linear combination of the learners with margin at least γ. In the context of PAC learning this can be stated as:

Corollary 2.1. *Given a weak learner that always generates a hypothesis $\hat{h} \in \mathsf{H}$ with error rate smaller than $\frac{1}{2} - \frac{1}{2}\gamma$. Then there exists a linear combination in the convex hull of H that has a margin ρ^* of at least γ.*

Proof If for any distribution there exists a hypothesis $\hat{h} \in \mathsf{H}$ with error rate $\frac{1}{2} - \frac{1}{2}\gamma$, i.e. with edge γ, then the objective of the edge-LP problem is least γ. By Theorem 2.1 the margin must be at least γ, i.e. $\rho^* \geq \gamma$ ■

The central question considered in this chapter is to understand whether and under which conditions Boosting algorithms generate a combined hypothesis with largest margin. For AdaBoost, it has only been shown that it achieves a margin of at least $\frac{1}{2}\gamma$ [Schapire et al., 1998, Breiman, 1999]. In Section 2.3.3 we show that it can be lower bounded by a slightly better bound $\frac{\log(1-\gamma^2)}{\log(1-\gamma)-\log(1+\gamma)} \approx \frac{1}{2}\gamma + \frac{1}{12}\gamma^3$, which is shown to be tight in empirical cases.

In the next section we propose a generalization of AdaBoost, called AdaBoost$_\varrho$, which is equivalent to AdaBoost for $\varrho = 0$. If the parameter ϱ is chosen close to γ with $\varrho < \gamma$, we can show that AdaBoost$_\varrho$ returns a combined hypothesis with margin close to γ. However, this requires the prior knowledge of γ. In Section 2.4 we will therefore derive an algorithm that estimates γ iteratively and so is able to maximize the margin.

2.2 AdaBoost$_\varrho$

For AdaBoost it has been shown that it quickly generates a combined hypothesis that is consistent with the training set, if the base learning algorithm has consistently smaller error

than $\frac{1}{2}$. This is desirable for the analysis in a PAC setting (cf. Section 1.3.1). Consistency on the training set means that the margin of each training example is larger than 0, i.e. $\rho = \min_{n=1,\ldots,N} \rho_n > 0$. It has been shown that AdaBoost achieves a positive margin on all examples as fast as possible (in the given framework) [Freund, 1995, Theorem 2.4].

We will start with a slight modification of AdaBoost (cf. Algorithm 2.2), which does not aim to find a hypothesis with margin of at least 0 but with margin at least ϱ, where ϱ is pre-specified and the base learning algorithm is assumed to have consistently smaller error rates than $\frac{1}{2} - \frac{1}{2}\varrho$. In order to achieve this goal, it turned out that one only needs to compute hypothesis coefficient differently (cf. Algorithm 2.2).

Algorithm 2.2 The AdaBoost$_\varrho$ algorithm [Breiman, 1999, Rätsch et al., 2001]

1. **Input:** $S = \langle (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N) \rangle$, Number of Iterations T, Target margin ϱ
2. **Initialize:** $d_n^{(1)} = 1/N$ for all $n = 1 \ldots N$
3. **Do for** $t = 1, \ldots, T$,
   (a) Train classifier with respect to the weighted sample set $\{S, \mathbf{d}^{(t)}\}$ and obtain hypothesis $h_t : \boldsymbol{x} \mapsto [-1, +1]$
   (b) Set
   $$\alpha_t = \operatorname*{argmin}_{\alpha \in \mathbb{R}} \sum_{n=1}^{N} d_n^{(t)} \exp\left\{\alpha \left(\varrho - y_n h_t(\boldsymbol{x}_n)\right)\right\} \tag{2.4}$$
   (c) Update weights:
   $$d_n^{(t+1)} = d_n^{(t)} \exp\left\{-\alpha_t y_n h_t(\boldsymbol{x}_n)\right\} / Z_t \ , \tag{2.5}$$
   where Z_t is a normalization constant, such that $\sum_{n=1}^{N} d_n^{(t+1)} = 1$.
4. **Break if** $\alpha_t \leq 0$ or $\alpha_t = \infty$.
5. **Output:** $f(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})$

We will call this algorithm AdaBoost$_\varrho$, as it naturally generalizes AdaBoost for the case where the *target margin* is not 0. It will turn out to be useful for understanding and developing new algorithms.[4] We will call the original algorithm AdaBoost$_0$ (or just AdaBoost), as it targets a margin of $\varrho = 0$.

One difference between original AdaBoost and Algorithm 2.2 is the choice of the hypothesis coefficients α_t in each iteration. If the base hypothesis output is either -1 or $+1$, then the step size can be computed analytically [cf. Breiman, 1999, Section 10.1] by

$$\alpha_t = \frac{1}{2} \log \frac{1 + \hat{\gamma}_t}{1 - \hat{\gamma}_t} - \frac{1}{2} \log \frac{1 + \varrho}{1 - \varrho}, \tag{2.6}$$

where $\hat{\gamma}_t$ is the edge of h_t. Note that the addition term $-\frac{1}{2} \log \frac{1+\varrho}{1-\varrho}$ is zero in AdaBoost$_0$.

4. A similar algorithm is known as *unnormalized Arcing* [Breiman, 1999] or *AdaBoost-type algorithm* [Rätsch et al., 2001]. Moreover, it is related to an algorithm proposed in Freund and Schapire [1999a].

It has been found that AdaBoost and also AdaBoost$_\varrho$ perform an approximate gradient descent in the space of linear combinations $f_{\hat{\boldsymbol{\alpha}}}$ of hypotheses from H [e.g. Breiman, 1999] in order to minimize

$$\mathrm{G}[f_{\hat{\boldsymbol{\alpha}}}] = \sum_{n=1}^{N} \exp\left\{(\varrho - \rho_n(\hat{\boldsymbol{\alpha}})) \sum_{j=1}^{J} \hat{\alpha}_j\right\}. \tag{2.7}$$

In Chapter 3 we will define more formally what is meant by the "gradient descent behavior". We show that AdaBoost$_\varrho$ is related to the Gauss-Southwell method used in numerical optimization. Here we only note that the particular way to compute the hypothesis coefficient and the example weighting as in (2.4) and (2.5) can directly be derived from (2.7).

2.3 Asymptotic Analysis

Depending on the data at hand, AdaBoost$_\varrho$ can have quite different asymptotical properties. It crucially depends on the choice of ϱ and the margin ρ^* achievable on the given data set. Important for understanding the asymptotic behavior is whether the sum of the hypothesis coefficients goes to infinity. This is the case if $\varrho < \rho^*$. Then AdaBoost$_\varrho$ achieves *large margins*.

In the subsequent analysis presented here we will always assume that the sum of the $\hat{\alpha}$'s tends to infinity. The other case where it stays finite is considered in Chapter 3 in a more general setting. In this case the smallest margin is generally not large (e.g. not necessarily larger than 0).

2.3.1 The Asymptotical Length of $\boldsymbol{\alpha}$

The asymptotical length of $\boldsymbol{\alpha}$ is the key to understand the asymptotical behavior of AdaBoost. We therefore give sufficient conditions when the sum of the $\hat{\alpha}$'s stays bounded and when it goes to infinity. The proofs of the following lemmas are found in Appendix A.1.1-A.1.4 on page 135.

Let us assume that the base learning algorithm is guaranteed to always find a hypothesis with edge larger than some fixed $\hat{\gamma} > \varrho$ (weak PAC-learner assumption, cf. Definition 1.2). Then the following lemma shows that the length of $\boldsymbol{\alpha}^{(t)}$ increases *at least linearly* with the number of iterations.

Lemma 2.1. *If, in the learning process of AdaBoost$_\varrho$ with $0 < \varrho < 1$, all edges $\hat{\gamma}_t$, $t = 1, 2, \ldots$ are bounded by $\hat{\gamma}_t \geq \varrho + c$ $(c > 0)$, then $\sum_{r=1}^{t} \alpha_r \geq ct$, i.e. $\sum_{r=1}^{t} \alpha_r$ increases at least linearly with the number of iterations t.*

Note that the assumption made in Lemma 2.1 are the natural extension of the assumptions for original AdaBoost (cf. Section 1.3.1). We believe that only in this case AdaBoost$_\varrho$ achieves the exponential convergence required for the analysis in the PAC setting.

Let us consider the case where the $\hat{\gamma}_t$'s are not bounded away from ϱ, but always greater than ϱ. This is the case where $\varrho = \rho^*$. For a special case of binary valued base hypotheses we have:

Lemma 2.2. *Assume $h_t(\boldsymbol{x}_n) \in \{-1, +1\}$ for all $n = 1, \ldots, N$ and $t = 1, 2, \ldots$. If, in the learning process of AdaBoost$_\varrho$ with $0 \leq \varrho < 1$, all edges $\hat{\gamma}_t$, $t = 1, \ldots$ are bounded by $\hat{\gamma}_t \geq \varrho + \frac{c}{t}$ $(c > 0)$, then $\sum_{r=1}^{t} \alpha_r \stackrel{t\to\infty}{\longrightarrow} \infty$.*

This lemma is in particular important if one uses AdaBoost$_0$ and complementation closed hypothesis space. Then there always exists a "separation" with margin 0 (even if the data is not separable).

In cases where the last two lemmas apply, we can use the following lemma to show that the achieved margin is asymptotically at least ϱ:

Lemma 2.3. *Suppose AdaBoost$_\varrho$ generates a sequence of hypotheses $h_1, h_2, \ldots$ and coefficients $\alpha_1, \alpha_2, \ldots$. If $\sum_{r=1}^{t} \alpha_r \stackrel{t\to\infty}{\longrightarrow} \infty$, then $\lim_{t\to\infty} \rho_n(\boldsymbol{\alpha}^{(t)}) \geq \varrho$.*

For the case where the $\hat{\gamma}_t$'s converge to ρ^* in the real valued case, we can show the statement in Lemma 2.3 directly with a slightly stronger assumption:

Lemma 2.4. *If, in the learning process of AdaBoost$_\varrho$ with $0 \leq \varrho < 1$, all edges $\hat{\gamma}_t$, $t = 1, \ldots$ are bounded by $\hat{\gamma}_t \geq \varrho + \frac{c}{\sqrt{t}}$ $(c > 0)$, then $\lim_{t\to\infty} \rho_n(\hat{\boldsymbol{\alpha}}^{(t)}) \geq \varrho$.*

The Lemmas 2.2 and 2.4 are partial answer to the open problem posed in Schapire et al. [1998] (page 1665). However, it is still an open question what happens if the $\hat{\gamma}_t$'s converge faster to ϱ than above.

The last case is $\rho^* > \gamma$. Then the outputs of the combined hypothesis $f_{\hat{\boldsymbol{\alpha}}^{(t)}}(\boldsymbol{x}_n)$ generated by AdaBoost$_\varrho$ will stay bounded or AdaBoost$_\varrho$ stops after a finite number of iterations. This is e.g. the case for AdaBoost$_0$, if H is not closed under complementation and the data is not separable. This case is considered in detail in Chapter 3.

From the results of this section we can conclude that AdaBoost$_\varrho$ achieves large margins, if ϱ is chosen to be smaller than ρ^*. Then $\sum_{r=1}^{t} \alpha_r \stackrel{t\to\infty}{\longrightarrow} \infty$, which we assume for the rest of the analysis in this chapter.

2.3.2 The Limiting Distribution $\mathbf{d}$ and Support Vectors

From the definition of $\mathbf{d}^{(t+1)}$ and $\rho_n(\boldsymbol{\alpha}^{(t)})$, (2.5) can be rewritten as

$$d_n^{(t+1)} = \frac{\exp\left(-\rho_n(\boldsymbol{\alpha}^{(t)})\right)^{1/\beta_t}}{\sum_{j=1}^{N} \exp\left(-\rho_j(\boldsymbol{\alpha}^{(t)})\right)^{1/\beta_t}}\,, \tag{2.8}$$

where we emphasize that $\beta_t^{-1} \equiv \sum_{r=1}^{t} \alpha_t$ can be written in the exponent. Inspecting this equation more closely, we see that AdaBoost$_\varrho$ uses a soft-max function [e.g. Bishop, 1995] with parameter β_t that has been interpreted as an *annealing parameter* [Onoda et al., 1998]. In the beginning β is large and all examples have similar weights (if $\beta = \infty$ then all weights are the same). As β decreases, the example with smallest margin will get higher and higher weights. In the limit of small β, we arrive at the maximum function: Only the example(s) with the smallest margin will be taken into account for learning and get a non-zero weight (asymptotically).

In Section 2.5 we will show a connection to *barrier optimization techniques* frequently used in constrained optimization and will see that β acts as the barrier parameter.

From (2.8) we see that as soon as the "annealing parameter" β is small, AdaBoost$_\varrho$ learning becomes a hard competition case: only the examples with smallest margin will get high weights, other examples are effectively neglected in the learning process. The following proposition shows that there will be a subset of examples of each class that has the same smallest margin. We will call these examples *Support Vectors* (SVs) , since these examples support the decision hyperplane [cf. Boser et al., 1992].

Proposition 2.1 (Rätsch et al. [2001]). *During the learning process of AdaBoost$_0$, the smallest margin of the training examples of each class will asymptotically converge to the same value, i.e.*

$$\lim_{t\to\infty} \min_{n:y_n=1} \rho_n(\boldsymbol{\alpha}^{(t)}) = \lim_{t\to\infty} \min_{n:y_n=-1} \rho_n(\boldsymbol{\alpha}^{(t)}) \,, \tag{2.9}$$

if the following assumptions are fulfilled:

1. *the weight of each hypothesis is bounded by $0 < \alpha_l < \alpha_t < \alpha_u < \infty$, where α_l and α_u are fixed constants, and*
2. *the learning algorithm must be able to classify all examples to one class $c \in \{\pm 1\}$, if the sum over the weights of examples of class c is larger than a constant δ, i.e. $\sum_{n:y_n=c} d_n > \delta \;\Rightarrow\; \hat{h}(\boldsymbol{x}_n) = c$ for all $n = 1, \ldots, N$.*

Note that the proposition is proven for AdaBoost$_0$ only (i.e. $\varrho = 0$), but can be extended to the more general case. Both assumptions in Proposition 2.1 can usually be satisfied. The first is true if $\rho^* \in (0, 1)$. The second assumption requires from the base learner that, if almost all weight is concentrated to one class, it predicts for all examples this label. This is not a common assumption, but is arguably easy to satisfy.

Proposition 2.1 does not state how many examples will become SVs, but there is at least one SV in each class. Assuming that the combined hypothesis of AdaBoost$_\varrho$ does not change when removing a non-SV, the number of SVs is an upper bound on the leave-one-out error [e.g. Luntz and Brailowsky, 1969]. Thus, if there are only a few SVs, AdaBoost$_\varrho$ is expected to generalize well. Hence, in practice one would expect to have more than just two SVs. However, since it is not known where the original AdaBoost algorithm is converging to,[5] it is not known whether removing a non-SV can change the solution. Clearly, if AdaBoost$_\varrho$ would solve the margin-LP problem (2.3) asymptotically, then removing inactive constraints does asymptotically not change the solution.

2.3.3 How Large is the Margin?

AdaBoost's good generalization performance can be explained in terms of the size of the margin: for low noise, the hypothesis with the largest margin will generalize well [Vapnik, 1995, Schapire et al., 1998] (cf. discussion in Section 1.2). Thus, it is interesting to understand what the margin size depends on.

5. The limit point (if it exist) obviously satisfies some known conditions, however, these conditions do not determine the limit uniquely.

We start with a theorem that generalizes Theorem 1.4 and also Theorem 5 of Schapire et al. [1998] for $\varrho \neq 0$. It will serve as a basis for deriving asymptotic and non-asymptotic convergence guarantees.

Theorem 2.2 (Rätsch et al. [2001]). *Let $\hat{\gamma}_1, \ldots, \hat{\gamma}_T$ be the edges of the hypotheses $h_1, \ldots, h_T \mapsto [-1, +1]$ generated by AdaBoost$_\varrho$ (cf. Algorithm 2.2). Furthermore let $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_T]$ be the vector hypothesis coefficients satisfying $\hat{\boldsymbol{\alpha}} \geq \mathbf{0}$ and f be the combined hypothesis. Then the following inequality holds for all $\theta \in [-1, 1]$:*

$$\frac{1}{N} \sum_{n=1}^{N} \mathbf{I}\left(\rho_n(\boldsymbol{\alpha}) \leq \theta\right) \leq \prod_{t=1}^{T} \sqrt{\left(\frac{1-\hat{\gamma}_t}{1-\varrho}\right)^{1-\theta} \left(\frac{1+\hat{\gamma}_t}{1+\varrho}\right)^{1+\theta}}. \tag{2.10}$$

The proof is shown in Appendix A.1.5 on page 136. The theorem is related to a result obtained in in Freund and Schapire [1999a].

As long as the square root is smaller than 1, one gets a reduction of the rhs. of the bound. If it is bounded away from 1,[6] then the rhs. converges exponentially fast to zero. The following corollary considers the asymptotical case and gives a lower bound on the margin. We will show later empirically that the bound is tight.

Corollary 2.2 (Rätsch et al. [2001]). *Assume AdaBoost$_\varrho$ generates hypotheses $h_1, h_2, \ldots$ with edges $\hat{\gamma}_1, \hat{\gamma}_2, \ldots$ and coefficients $\alpha_1, \alpha_2, \ldots$. Let $\hat{\gamma} = \inf_{t=1,2,\ldots} \hat{\gamma}_t$ and assume $\hat{\gamma} > \varrho$. Then the asymptotically ($t \to \infty$) smallest margin $\hat{\rho}$ of the combined hypothesis is bounded from below by*

$$\hat{\rho} \geq \frac{\log(1-\hat{\gamma}^2) - \log(1-\varrho^2)}{\log((1+\varrho)(1-\hat{\gamma})) - \log((1-\varrho)(1+\hat{\gamma}))}. \tag{2.11}$$

The proof can be found in Appendix A.1.6 on page 137. We conjecture that this result can be extended such that the average edge is used instead of the minimal edge.

From (2.11) one can understand the interaction between ϱ and $\hat{\gamma}$: if the difference between $\hat{\gamma}$ and ϱ is small, then the right hand side of (2.11) is small. Thus, if ϱ with $\varrho \leq \hat{\gamma}$ is large, then $\hat{\rho}$ must be large, i.e. choosing a larger ϱ results in a larger margin on the training examples (cf. Figure 2.1). Note that (2.11) implies the weaker bound $\hat{\rho} \geq \frac{\hat{\gamma}+\varrho}{2}$. Previously it was only known that $\hat{\rho} \geq \varrho$ [Breiman, 1999, Theorem 7.2].

However, in the Section 2.1 we have shown that the maximal achievable margin is at least $\hat{\gamma}$. Thus if ϱ is chosen too small, then we guarantee only a *suboptimal asymptotical margin*. In the original formulation of AdaBoost we have $\varrho = 0$ and we guarantee only that AdaBoost$_0$ achieves a margin of at least $\frac{1}{2}\hat{\gamma} + \frac{1}{12}\hat{\gamma}^3$. This gap in the theory motivates the algorithm proposed in Section 2.4, where ϱ is adaptively chosen.

2.3.4 Experimental Illustration of Asymptotical Properties

To validate the analysis of Section 2.3.2 we performed numerical simulations on toy data with an almost asymptotic number of 10^4 boosting steps. These experiments (cf. Fig-

6. This means that there exists a constant $\mu > 0$ such that the square-root term is *always* smaller than $1 - \mu$. Otherwise, the sequence of square-root terms might converge to 1 and the product of them might not converge to zero.

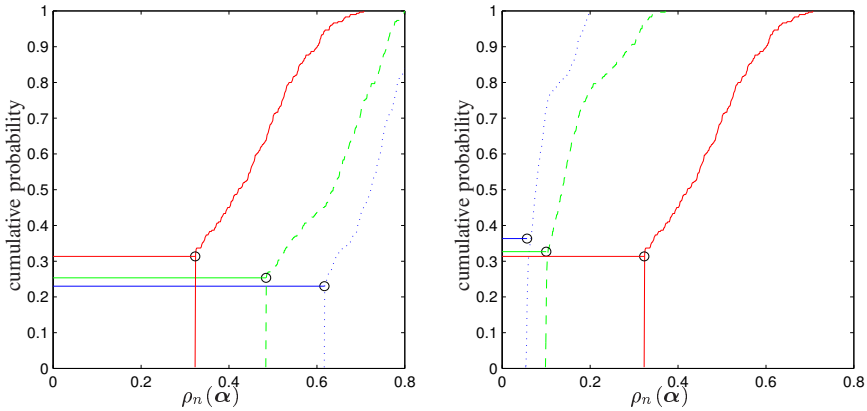

Figure 2.1 Margin distributions of AdaBoost$_0$ for different noise levels in the `banana` data set [cf. Onoda et al., 1998]: $\sigma^2 = 0\%$ (dotted), 9% (dashed), 16% (solid) with RBF nets (13 centers; cf. Appendix B.6.1) as base hypotheses [left]; and with 7 (dotted), 13 (dashed), 30 (solid) centers in the base hypotheses for data with $\sigma^2 = 16\%$ [right] after 10^4 AdaBoost$_0$ iterations. These graphs support the discussion in Section 2.3.2 and confirm the trends expected from (2.11). (Figure taken from Rätsch et al. [2001].)

ure 2.1, see Rätsch et al. [2001] for details) support that the margin distribution asymptotically has a *step at a specific margin size* and that some subset of the training examples all have similar margins that correspond to the minimal margin discussed above. If the noise level is high or the complexity of the base hypothesis is low, one gets higher training errors ϵ_t, i.e. smaller edges $\hat{\gamma}_t$, and therefore smaller values of the margin ρ.

Thus, there is also empirical evidence that AdaBoost$_\varrho$ achieves asymptotically a decision with *hard margin*, very similar to the one of SVMs for the separable case. There is a subset of examples that play asymptotically a crucial role for defining the combined hypothesis – the most difficult examples are emphasized strongly and become support vectors asymptotically.

We performed a second experiment showing how tight we can lower bound the asymptotical margin using (2.11). We analyze two different settings: the base learner selects (i) the hypothesis with largest edge over all hypotheses and (ii) the hypothesis with minimal edge among all hypothesis with edge larger than ρ^*. This is the best and the worst case, respectively. Note that Corollary 2.2 holds for the *worst case*, where the base learner is allowed to return any hypothesis with edge larger than ρ^* (the maximal achievable margin). In practice, however, the base learner often performs better.

We use random data with N training examples, where N is drawn uniformly between 10 and 200. The labels are drawn at random from a binomial distribution with equal probability. We use a hypothesis set with 10^4 hypotheses. The output of every hypothesis is either -1 or $+1$ and is chosen such that with probability p it is equal to the previously generated label.[7] First we compute the solution of the margin-LP problem (2.3) and get to know ρ^*. Then we compute the combined hypothesis generated by AdaBoost$_\varrho$ after 10^4

7. With low probability, the output of an hypothesis is equal to all labels. These cases are excluded.

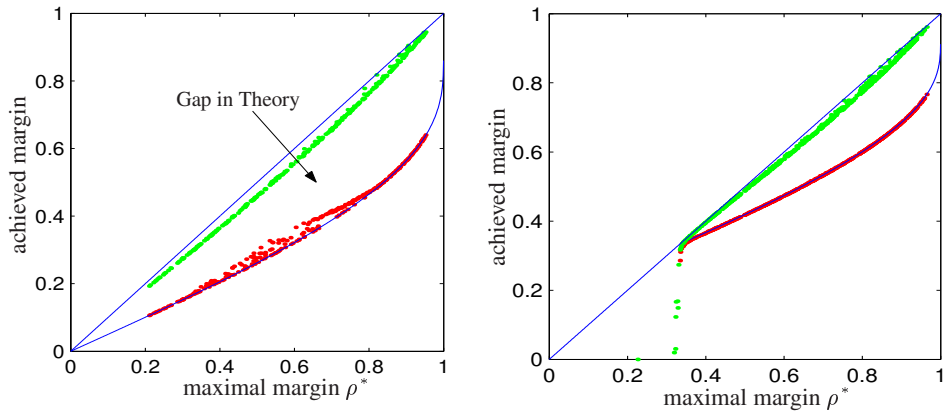

Figure 2.2 Achieved margins of AdaBoost$_\varrho$ using the best (green) and the worst (red) selection on random data for $\varrho = 0$ [left] and $\varrho = \frac{1}{3}$ [right]: On the abscissa is the maximal achievable margin ρ^* and on the ordinate the margin achieved by AdaBoost$_\varrho$. For comparison the line $y = x$ and the bound (2.11) are plotted. On the interval $[\varrho, 1]$, Eq. (2.11) lower bounds the achieved margin of the worst strategy tightly. The optimal strategy almost achieves the maximal margin. There is a clear gap between best and worst performance of the selection scheme, which we cannot explain by theory. If ϱ is chosen appropriately, then this gap is reduced to 0.

iterations for $\varrho = 0$ and $\varrho = \frac{1}{3}$ using the best and the worst selection strategy, respectively. The latter depends on ρ^*. This procedure is repeated for 300 random draws of p from an uniform distribution. The parameter p ensures that there are cases with small and large optimal margins.

The resulting margins computed by the three algorithms are plotted in Figure 2.2. On the abscissa is the maximal achievable margin ρ^* for a particular data realization. On the ordinate is the margin of AdaBoost$_0$ using the best and the worst strategy. We observe that there is a big difference between both selection strategies. Whereas the margin of the worst strategy is *tightly lower bounded* by (2.11), the best strategy has near maximal margin.

These experiments show that one obtains different results by changing some properties of the base learning algorithm. These properties are not used for computing the bound (only the minimal edge is used). This is indeed a problem, as one is not able to predict where AdaBoost$_\varrho$ is converging to.

2.3.5 Summarizing Remarks

We have analyzed the asymptotical properties of AdaBoost$_\varrho$. By assuming the base learner always returns a hypothesis with edge at least $\hat{\gamma}$, we have shown if $\varrho < \rho^* \leq \hat{\gamma}$, then the sum of the hypothesis coefficients tends to infinity. For this case, we lower bounded the asymptotically achieved margin by (2.11), which is quite tight in the worst case. In the best case performance of the base learner, however, we find that the margin is usually much larger. This indicates that other conditions are necessary to fill the gap between best and worst case performance.

In the next section we consider an algorithm that does not have this problem. We can show that it converges to the maximum margin solution.

2.4 Marginal Boosting

In this section we avoid the problems we have encountered in the last section by adaptively choosing ϱ. We will propose a boosting algorithm that is able to maximize the margin and comes with good desirable theoretical properties.

2.4.1 Motivation

From Figure 2.2 [right] it becomes apparent, if one sets $\varrho \approx \rho^*$, then the gap between the best and the worst case becomes very small. Breiman [1999] proposed the idea to adaptively modify ϱ in each iteration of AdaBoost$_\varrho$. This lead to an algorithm called Arc-GV (**Arc**ing-**G**ame **V**alue). Here one sets ϱ_t as the achieved margin of the previous iteration:

$$\varrho_t = \min_{n=1,\ldots,N} \rho_n(\boldsymbol{\alpha}^{t-1}). \tag{2.12}$$

Assuming that the base learning algorithm always returns a hypothesis with the *largest* edge among all hypotheses [called *weighted minimization* in Breiman, 1999], it has been shown that Arc-GV *asymptotically* generates a hypothesis maximizing the margin. It can in fact be shown that the rather strong assumption on the base learner can be relaxed considerably[8] (cf. Appendix B.3), however, still leading to an asymptotical result only. This is theoretically unsatisfying since for AdaBoost$_\varrho$ we already know that its exponential loss (cf. (2.7)) convergences exponential fast to zero, if ϱ is specified correctly.

In this section we therefore propose an algorithm for which we can show that it quickly converges to the maximum margin solution. It combines the merits of AdaBoost (fast proven convergence) and of the maximum margin hyperplane (good generalization).

Before we start with the actual algorithm we briefly analyze Algorithm 2.2 where ϱ is not fixed but adapted in each iteration. The following discussion is to be seen as a motivation for our algorithm. Later we will come back to the case of fixed ϱ. Let us consider sequences $\{\varrho_t\}_{t=1}^T$, which might either be specified before running the algorithm or computed based on results during the algorithm. For this case we can easily generalize Theorem 2.2 and have

$$\frac{1}{N}\sum_{n=1}^{N} \mathbf{I}\left(\rho_n(\boldsymbol{\alpha}^T) \le \theta\right) \le \prod_{t=1}^{T} \sqrt{\left(\frac{1-\hat{\gamma}_t}{1-\varrho_t}\right)^{1-\theta} \left(\frac{1+\hat{\gamma}_t}{1+\varrho_t}\right)^{1+\theta}}. \tag{2.13}$$

Furthermore, from the proof of Theorem 2.2 we have for $\theta \le \min_{t=1,\ldots,T} \varrho_t$,

$$\frac{1}{N}\sum_{n=1}^{N} \mathbf{I}\left(\rho_n(\boldsymbol{\alpha}^T) \le \theta\right) \le \exp\left\{-\sum_{t=1}^{T} \Delta_2(\varrho_t, \hat{\gamma}_t)\right\}, \tag{2.14}$$

where $\Delta_2(\varrho, \hat{\gamma}_t) := \frac{1+\varrho}{2}\log\frac{1+\varrho}{1+\hat{\gamma}_t} + \frac{1-\varrho}{2}\log\frac{1-\varrho}{1-\hat{\gamma}_t}$ is the binary relative entropy.

8. Here (as in the previous section), one only needs to require that the base learner finds hypotheses with edge not smaller than ρ^*.

Thus, the algorithm makes progress reducing the rhs. of (2.13), if the term under the square-root is smaller than 1. This is e.g. the case if $\hat{\gamma}_t$ is large compared to θ and ϱ_t or, by (2.14), if $\theta \leq \varrho_t < \hat{\gamma}_t$. The larger $\hat{\gamma}_t$, the more one progresses.

Suppose we would like to reach a margin θ on all training examples, where we obviously need to assume $\theta \leq \rho^*$. Then the question arises, which sequence of $\{\varrho_t\}_{t=1}^T$ one should use to find a combined hypothesis in as few iterations as possible. Assume for a moment that $\hat{h}_t(\boldsymbol{x}_n) \in \{-1, +1\}$. The rhs. of (2.13) is equal to $\prod_{t=1}^T \exp\{\theta\hat{\alpha}_t + \log \hat{Z}_t\}$, where $\hat{\alpha}_t$ is given in step (2.6) and $\hat{Z}_t = \frac{1+\varrho}{2}\exp(-\hat{\alpha}_t) + \frac{1-\varrho}{2}\exp(\hat{\alpha}_t)$ is an upper bound [Schapire and Singer, 1999] on Z_t as used in step (2.5) of Algorithm 2.2. We achieve a minimum of the right hand side, if we independently choose ϱ_t in each iteration t such that $\exp\{\theta\hat{\alpha}_t + \log \hat{Z}_t\}$ is minimized. Setting the derivative with respect to ϱ_t to 0 and solving for ϱ_t yields $\varrho_t = \theta$. *So, it turns out that one should use $\varrho_t \equiv \theta$, independently how the base learner performs!*

Thus we can return to the case $\varrho = \text{const}$ and think about a different approach.

2.4.2 The Algorithm and its Analysis

Motivated by the discussion above, we propose – instead of modifying the target margin ϱ within the algorithm – to have a second algorithm that specifies ϱ (from outside). We re-start AdaBoost$_\varrho$ for each value of ϱ and can prove fast convergence.

Before we go on, let us upper bound the number of iterations of AdaBoost$_\varrho$ to achieve a margin on all examples of at least ϱ:

Corollary 2.3 (Rätsch and Warmuth [2001]). *Assume the base learner always achieves an edge $\hat{\gamma}_t \geq \rho^*$. If $\varrho \leq \rho^* - \varepsilon$, $\varepsilon > 0$, then AdaBoost$_\varrho$ will converge to a solution with margin of at least ϱ on all examples in at most $\lceil \frac{2\log(N)}{\varepsilon^2} \rceil + 1$ steps.*

Proof We use (A.5) for $\theta \equiv \varrho$, yielding $(A.5) \leq \exp\{-\sum_t \frac{1}{2}(\varrho - \hat{\gamma}_t)^2\} \leq \exp\{-\frac{T\varepsilon^2}{2}\}$, where we use a bound on the binary entropy. If $\exp\{-\frac{T\varepsilon^2}{2}\} < \frac{1}{N}$, there is no example left with margin smaller than ϱ, which proves the corollary. ■

Since one does not know the value of ρ^* beforehand, one also needs to find ρ^*. We propose an algorithm that constructs a sequence $\{\varrho_r\}_{r=1}^R$ converging to ρ^*: A fast way to find a real value up to a certain accuracy ε on the interval $[-1, 1]$ is to use a *binary search* – one needs only $\log_2(2/\varepsilon)$ steps. Our idea is to use the binary search to find ρ^*, where we call Algorithm 2.2 to decide whether the current guess ϱ is *larger* or *smaller* than ρ^*. This leads to Algorithm 2.3, in which we assume that Algorithm 2.2 also returns the minimal observed edge $\hat{\gamma} = \min_{t=1,\ldots,T} \hat{\gamma}_t$ and the maximal observed margin $\hat{\rho} = \max_{t=1,\ldots,T} \rho(\boldsymbol{\alpha}^{(t)})$ in all iterations [cf. Rätsch and Warmuth, 2001].

The algorithm proceeds in R iterations, where R is determined by the desired accuracy ε: In each iteration r it calls AdaBoost$_{\varrho_r}$ (cf. step 3a in Algorithm 2.3), where ϱ_r is chosen to be in the middle of an interval $[l_r, u_r]$ (cf. step 3c). Based on the success of AdaBoost$_{\varrho_r}$ to achieve a margin large enough, the interval is updated (cf. step 3b). We can show that the interval is chosen such that it always contains ρ^*, the *unknown* maximal margin, while the length of the interval is almost reduced by a factor of two. Finally, in the last step of the

Algorithm 2.3 The Marginal AdaBoost algorithm [Rätsch and Warmuth, 2001]

1. **Input:** $S = \langle (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{y}_N) \rangle$, Accuracy ε
2. **Initialize:** $\varrho_1 = 0$, $l_1 = -1$, $u_1 = 1$, $R = \lceil \log_2(1/\varepsilon) \rceil$, $T = \lceil 2\log(N)/\varepsilon^2 \rceil + 1$.
3. **Do for** $r = 1, \ldots R$,
   (a) $[f_r, \hat{\gamma}_r, \hat{\rho}_r] = \mathrm{AdaBoost}_{\varrho_r}(S, T)$
   (b) **if** $\hat{\rho}_r \geq \varrho_r$, **then** $l_{r+1} = \max(\hat{\rho}_r, l_r)$, $u_{r+1} = \min(\hat{\gamma}_r, u_r)$
   else $l_{r+1} = \max(\hat{\rho}_r, l_r)$, $u_{r+1} = \min(\hat{\gamma}_r, \varrho_r + \varepsilon, u_r)$
   (c) $\varrho_{r+1} = \frac{l_r + u_r}{2}$
4. **Break if** $u_{r+1} - l_{r+1} \leq 3\varepsilon$
5. **Output:** $f = \mathrm{AdaBoost}_{l_{R+1}-\varepsilon}(S, 2\log(N)/\varepsilon^2)$

algorithm, one has reached a good estimate ϱ_R of ρ^* and calls $\mathrm{AdaBoost}_\varrho$ for $\varrho = l_{R+1} - \varepsilon$ generating a combined hypothesis with margin at least $\rho^* - 4\epsilon$.We have the following result:

Theorem 2.3 (Rätsch and Warmuth [2001]). *Assume the base learner always achieves an edge $\hat{\gamma}_t \geq \rho^*$. Then Algorithm 2.3 will find a combined hypothesis f that maximizes the margin up to accuracy 4ε in at most $\lceil \frac{2\log(N)}{\varepsilon^2} + 1 \rceil \lceil \log_2(1/\varepsilon) + 1 \rceil$ calls of the base learner. The final hypothesis combines at most $\lceil \frac{2\log(N)}{\varepsilon^2} + 1 \rceil$ base hypotheses.*

Sketch of Proof. Let us analyze the cases where our guess ϱ is too large or too small:

1. If $\varrho > \rho^*$, then one cannot reach the margin of ϱ since the maximum achievable margin is ρ^*. By Corollary 2.3, if $\mathrm{AdaBoost}_\varrho$ has not reached a margin of at least ϱ in $2\log(N)/\varepsilon^2$ steps, we can conclude that $\varrho > \rho^* - \varepsilon$ (cf. step 3b in Algorithm 2.3). This is the worst case. The better case is, if the distribution $\mathbf{d}$ generated by $\mathrm{AdaBoost}_\varrho$ will be too difficult for the base learner and it eventually fails to achieve an edge $\hat{\gamma}$ of at least ϱ. Then $\mathrm{AdaBoost}_\varrho$ can stop (cf. step 4 in Algorithm 2.2).
2. Assume ϱ is chosen too low, say $\varrho < \rho^* - \varepsilon$, then one achieves a margin of ϱ in a few steps by Corollary 2.3. Since the maximum margin is always greater than a certain achieved margin, one can conclude that $\rho^* \geq \varrho$ (cf. step 3b in Algorithm 2.3). This suggests to stop $\mathrm{AdaBoost}_\varrho$ as soon as it has reached a margin of at least ϱ.

   Note that there is a *small gap* in the proposed binary search procedure: We are not able to identify the case $\rho^* - \varepsilon \leq \varrho \leq \rho^*$ efficiently. This means that we cannot reduce the length of the search interval by *exactly* a factor of two in *each* iteration. This makes the analysis slightly more difficult, but eventually leads to the proof theorem on the *worst case performance* of Marginal AdaBoost. More details of the proof can be found in Appendix A.1.7 on page 137. ■

2.4.3 Experimental Illustration

First of all, we would like to note that we are aware of the fact that maximizing the margin of the ensemble does not lead in all cases to an improved generalization performance. For

fairly noisy data sets even the opposite has been reported [cf. Quinlan, 1996, Breiman, 1999, Grove and Schuurmans, 1998, Rätsch et al., 2001]. See Section 4.1 for a detailed discussion. However, at least for well separable data the PAC theory applies and, hence, one should be able to measure differences in the generalization error, if one function approximately maximizes the margin while another function does not. A similar setting has been examined experimentally in Schapire et al. [1998] and Onoda et al. [1998] on optical character recognition problems, leading to similar results.

Here we report experiments on artificial data only to illustrate (a) how our algorithm works and (b) how it compares to AdaBoost. Our data is 100 dimensional and contains 98 nuisance dimensions with uniform noise. The other two dimensions are plotted exemplary in Figure 2.3. For training we use only 100 examples and there is obviously the need to carefully control the capacity of the ensemble.

As base learning algorithm we use C4.5 decision trees provided by Quinlan [1992] using an option to control the number of nodes in the tree. We have set it such that C4.5 generates trees with about three nodes. Otherwise, the base learner often classifies all training examples correctly and over-fits the data already. We therefore need to limit the complexity of the base learner, in agreement with the bound (1.12) on the generalization error.

In Figure 2.4 [left] we see a typical run of Marginal AdaBoost for $\epsilon = 0.1$. It calls AdaBoost$_\varrho$ three times. The first call of AdaBoost$_\varrho$ for $\varrho = 0$ already stops after four iterations, since it has generated a consistent combined hypothesis. The lower bound l on ρ^* as computed by our algorithm is $l = 0.07$ and the upper bound u is 0.94 (cf. step 3b in Algorithm 2.3). The second time ϱ is chosen to be in the middle of the interval $[l, u]$ and AdaBoost$_\varrho$ reaches the margin of $\varrho = 0.51$ after 80 iterations. The interval is now $[0.51, 0.77]$. Since the length of the interval $u - l = 0.27$ is small enough, Marginal AdaBoost leaves the loop through exit condition 4, calls AdaBoost$_\varrho$ the last time for $\varrho = u - \epsilon = 0.41$ and finally achieves a margin of $\hat{\rho} = 0.55$ (on average 0.58) after 916 iterations (total 1000). For comparison we also plot the margins of the hypotheses generated by AdaBoost (cf. Figure 2.4 [right]). One observes that (original) AdaBoost is not able to achieve a large margin efficiently ($\hat{\rho} = 0.37$ after 1000 iterations; on average 0.31).

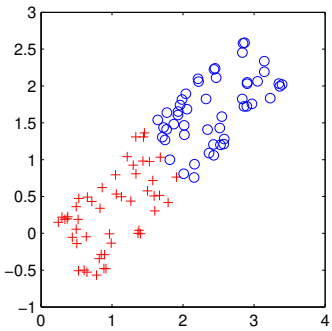

Figure 2.3: The two *discriminative dimensions* of our separable 100-dimensional data set.

In Table 2.1 we see the average performance of the three classifiers. For AdaBoost we combined 200 hypotheses for the final prediction. For Marginal AdaBoost we use $\varepsilon = 0.1$ and let the algorithm combine only 200 hypotheses for the final prediction to get a fair comparison. We see a large improvement of both ensemble methods compared to the single classifier. There is also a slight, but – according to a t-test with confidence level 98% – significant difference between the generalization performances of both boosting algorithms. Note also that the margins of the combined hypothesis achieved by Marginal AdaBoost are on average almost twice as large as for AdaBoost.

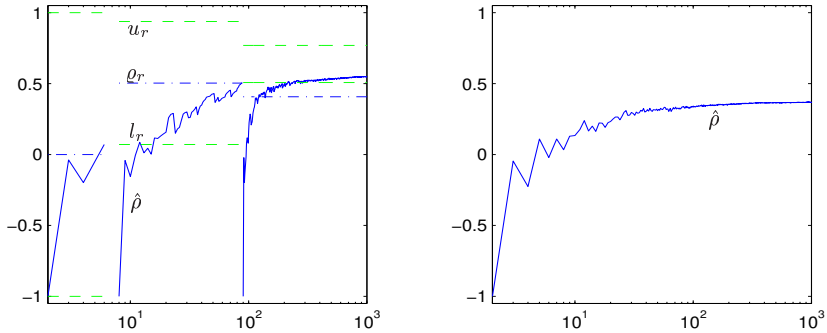

Figure 2.4 Illustration of the achieved margin of Marginal AdaBoost [left] and AdaBoost$_0$ [right] at each iteration. Our algorithm calls AdaBoost$_\varrho$ three times while adapting ϱ (dash-dotted). The values of l and u are plotted as dashed line. (Figure taken from Rätsch and Warmuth [2001].)

	C4.5	AB	Marginal AB
E_{gen}	$7.4 \pm 0.11\%$	$4.0 \pm 0.11\%$	$3.6 \pm 0.10\%$
ρ	—	0.31 ± 0.01	0.58 ± 0.01
wins	$1/200$	$59/200$	$140/200$

Table 2.1 Estimated generalization performances and margins with confidence intervals for decision trees (C4.5), AdaBoost (AB) and Marginal AB on the toy data. The last row shows the number of times the algorithm had the smallest error. All numbers are averaged over 200 splits into 100 training and 19900 test examples.

2.4.4 Summarizing Remarks

We proposed a boosting algorithm that approximately maximizes the margin of the combined hypothesis. In Rätsch and Warmuth [2001] we have shown that our analysis also holds for infinite hypothesis spaces if the hypothesis space is compact (see also Section 5.3). To the best of our knowledge this is the first result on the *non-asymptotical convergence* of a boosting algorithm to the maximum margin solution that also holds for infinite hypothesis spaces.

We have shown theoretically and empirically that our algorithm converges quite fast to the maximum margin solution, whereas AdaBoost is usually not able to achieve largest margins. We could prove this result without assuming additional properties of the base learning algorithm. In a toy experiment we have illustrated the validity of our analysis and also that a larger margin can decrease the generalization error when learning on high dimensional data with a few informative dimensions.

2.5 Relation to Barrier Optimization

In this section we would like to point out how the algorithms considered so far can be seen in the context of barrier optimization.[9] This connection illustrates how Boosting algorithms are related to the margin-LP problem [Rätsch et al., 2000c]. In particular, we show that the exponential function acts as barrier function for some constraints $\rho_n \geq \rho$ in the margin-LP problem (2.3). It becomes intuitively clear why it is important that $\sum_{t=1}^{T} \alpha_t \overset{T\to\infty}{\longrightarrow} \infty$, which we already found in the previous analysis.

Asymptotically, the example weights $\mathbf{d}$ as used in the algorithms indeed turn out to be equal to the Lagrange multipliers of the constraints $\rho_n \geq \rho$, $n = 1, \ldots, N$. Furthermore, we will show how AdaBoost [Freund and Schapire, 1994] and Arc-GV [Breiman, 1999] can be understood as particular implementations of a barrier optimization approach, asymptotically solving a convex optimization problem. It is shown that both algorithms implement a particular method for minimizing a barrier function. Such minimization methods (also called Gauss-Southwell methods) will be explicitly analyzed in Chapter 3.

2.5.1 Preliminaries

Throughout this section we will assume that the hypothesis space is finite: $\mathsf{H} = \{\hat{h}_j \mid j = 1, \ldots, J\}$. We will think of dealing in each iteration t of Boosting with a full hypothesis weight vector $\hat{\boldsymbol{\alpha}}^{(t)} \in \mathbb{R}^J$. However, one changes only one entry j at a time. Suppose Algorithm 2.2 selects the j-th hypothesis in iteration t (denoted by h_t), then the weight α_t as computed in the algorithm is the *update* of the j-th coefficient, i.e. $\hat{\alpha}_j^{(t+1)} = \hat{\alpha}_j^{(t)} + \alpha_t$. The hat and the indices are used to distinguish between both domains.

As in the previous sections of this chapter, we assume that the base learner always finds a base hypothesis with edge larger than ϱ. Then the hypothesis coefficient α_t as computed in AdaBoost$_\varrho$ is always positive. Thus $\hat{\boldsymbol{\alpha}}^{(t)} \geq \mathbf{0}$, for all $t = 1, 2, \ldots$. Furthermore we assume that $\sum_{j=1}^{J} \hat{\alpha}_j^{(t)} \overset{t\to\infty}{\longrightarrow} \infty$. For convenience, we will in the sequel write $\langle \mathbf{1}, \hat{\boldsymbol{\alpha}} \rangle$ instead of $\sum_{j=1}^{J} \hat{\alpha}_j$. Finally we assume that each α_t is finite.

It has been shown that Arc-GV and Marginal AdaBoost solve the following optimization problem:

$$\begin{array}{ll} \max\limits_{\hat{\boldsymbol{\alpha}} \geq 0} & \rho \\ \text{with} & \rho_n(\hat{\boldsymbol{\alpha}}) \geq \rho, \end{array} \tag{2.15}$$

where $\rho_n(\hat{\boldsymbol{\alpha}}) = y_n \sum_{j=1}^{J} \frac{\hat{\alpha}_j}{\langle \mathbf{1}, \hat{\boldsymbol{\alpha}} \rangle} \hat{h}_j(\boldsymbol{x}_n) = U\hat{\boldsymbol{\alpha}} / \langle \mathbf{1}, \hat{\boldsymbol{\alpha}} \rangle$ and U is as in (2.1). Note that we do not fix the length of $\hat{\boldsymbol{\alpha}}$ here, since the margin $\rho_n(\hat{\boldsymbol{\alpha}})$ is invariant under scaling of $\hat{\boldsymbol{\alpha}}$.

We now derive the *barrier function* associated with the optimization problem above. Following the standard methodology (cf. Appendix B.2) we find the corresponding barrier function using the *exponential barrier* [cf. (B.6) in Appendix B.2 and e.g. Cominetti and

9. The idea of this technique is to solve a sequence of unconstraint optimization problems in order to solve a constraint optimization problem, where the unconstraint problem is considered to be much easier. See Appendix B.2 for some details.

Dussault, 1994]:

$$\mathrm{F}_\beta(\hat{\boldsymbol{\alpha}}, \rho) = -\rho + \beta \sum_{n=1}^{N} \exp\left(\frac{\rho\langle \mathbf{1}, \hat{\boldsymbol{\alpha}}\rangle - U_{n,*}\hat{\boldsymbol{\alpha}}}{\beta\langle \mathbf{1}, \hat{\boldsymbol{\alpha}}\rangle}\right) \tag{2.16}$$

which needs to be minimized with respect to ρ and $\hat{\boldsymbol{\alpha}}$. Let

$$(\hat{\boldsymbol{\alpha}}_\beta, \rho_\beta) = \operatorname*{argmin}_{\hat{\boldsymbol{\alpha}}, \rho} \mathrm{F}_\beta(\hat{\boldsymbol{\alpha}}, \rho) \tag{2.17}$$

and let $\mathcal{S}^* = \{(\hat{\boldsymbol{\alpha}}^*, \rho^*)\}$ be the set of a global solution of (2.15). One can show (cf. (B.5) in Appendix B.2), that any limit point of $(\hat{\boldsymbol{\alpha}}_\beta, \rho_\beta)$ converges to an element in $\mathcal{S}^*$ for $\beta \to 0$.[10]

2.5.2 Relating Arc-GV to Barrier Optimization

Let us now consider one particular iterative strategy for minimizing (2.16) which will turn out to be exactly Arc-GV [cf. Appendix B.3 and Breiman, 1999]: We start with $\hat{\boldsymbol{\alpha}}^0 = \mathbf{0}$ and $f_{\hat{\boldsymbol{\alpha}}^0} \equiv 0$. In each iteration t we (i) approximate ρ_β for a given $\hat{\boldsymbol{\alpha}}$ and (ii) find a hypothesis $\hat{h}_j \in \mathsf{H}$ (i.e. an index j) and update $\hat{\alpha}_j$ to get the new hypothesis weight vector $\hat{\boldsymbol{\alpha}}^{(t)}$.

First we set $\beta_t := \frac{1}{\langle \mathbf{1}, \hat{\boldsymbol{\alpha}}^{(t)}\rangle}$, which is reasonable since we assumed $\langle \mathbf{1}, \hat{\boldsymbol{\alpha}}^{(t)}\rangle \to \infty$. In step (i) one approximates the minimum margin ρ_β by (cf. (2.12))

$$\rho_t = \min_{n=1,\ldots,N} \rho_n(\hat{\boldsymbol{\alpha}}^{(t-1)}).$$

Noteworthy, the minimizer $\rho_\beta(\hat{\boldsymbol{\alpha}}^{(t-1)})$ of (2.16) for fixed $\hat{\boldsymbol{\alpha}}^{(t-1)}$ is given by

$$\rho_\beta(\hat{\boldsymbol{\alpha}}) = -\beta \log\left(\sum_{n=1}^{N} \exp\left\{-\frac{\rho_n(\hat{\boldsymbol{\alpha}})}{\beta}\right\}\right),$$

which converges to ρ_t for $\beta \to 0$.

In step (ii) one selects a hypothesis $h_t = \hat{h}_j$ and updates the hypothesis coefficient $\hat{\alpha}_j$ by solving

$$\hat{\alpha}_j = \operatorname*{argmin}_{\hat{\alpha}_j} \sum_{n=1}^{N} \exp\left(\rho_t\langle \mathbf{1}, \hat{\boldsymbol{\alpha}}\rangle - U_{n,*}\hat{\boldsymbol{\alpha}}\right), \tag{2.18}$$

where $\hat{\alpha}_{j'} = \hat{\alpha}_{j'}^{(t-1)}$ for all $j' \neq j$. Again, one can show that the minimizer in (2.18) converges to the minimizer of (2.16) for $\beta \to 0$, when all other variables are fixed. Roughly speaking, the β in front of the sum of (2.16) does not change the optimal $\hat{\alpha}_j$ considerably, if β is small enough.

To show convergence of Arc-GV to the solution of (2.15) in the framework of barrier optimization, we exploit that $\varrho_t \nearrow \rho^*$, where ρ^* is the solution of (2.15) [cf. Appendix B.3 and Breiman, 1999]. Then it is easy to show that the sub-gradient $\nabla_{\hat{\boldsymbol{\alpha}} \geq \mathbf{0}} \mathrm{F}(\hat{\boldsymbol{\alpha}})$ converges

10. Not that if one would set $\beta = 0$ immediately, then F_β would be infinity if the solution is not feasible and one would arrive at the original optimization problem.

to zero.[11] Roughly speaking, since only hypotheses with edge larger than $\rho_t \leq \rho^*$ will be chosen (by assumption), the coefficients of other hypothesis stay bounded, while $\langle \mathbf{1}, \hat{\boldsymbol{\alpha}} \rangle$ goes to ∞. Thus, in the normalized vector these weights go to zero.

Since $\langle \mathbf{1}, \hat{\boldsymbol{\alpha}} \rangle$ tends to infinity, β goes to zero. By the reasoning above, the gradient $\nabla_{\hat{\alpha}_j \geq 0} \mathrm{F}(\hat{\boldsymbol{\alpha}})$ also tends to zero. We can therefore apply the convergence result for exponential barriers (cf. Proposition B.1), leading to a proof that Arc-GV converges to an optimal solution of (2.15).

Finally, note that under rather mild assumptions [cf. Cominetti and Dussault, 1994], the Lagrange multiplier of the constraint $\rho_n(\hat{\boldsymbol{\alpha}}) \geq \rho$ in (2.15) is approximated by

$$d_n \approx \exp\left\{\frac{\rho_t - \rho_n(\hat{\boldsymbol{\alpha}}^{(t)})}{\beta_t}\right\} = \exp\left\{\rho\langle \mathbf{1}, \hat{\boldsymbol{\alpha}}^{(t)}\rangle - y_n f_{\hat{\boldsymbol{\alpha}}^{(t)}}(\boldsymbol{x}_n)\right\}, \tag{2.19}$$

where in the limit for $\beta \to 0$ the approximation becomes an equality. Thus the example weighting $\mathbf{d}$ computed in each iteration, asymptotically converges to the Lagrange multipliers of (2.15) (up to scaling).

Summarizing, in each iteration Arc-GV updates ρ and some $\hat{\alpha}_j$ – asymptotically such that the barrier function is minimized with respect to these variables. This procedure is related to the Gauss-Southwell method, which will be analyzed in detail in Chapter 3. Based on this argumentation, we would like to view Arc-GV as a barrier algorithm using a particular strategy for minimizing F_β and choosing β.

2.5.3 Finding a Separation with AdaBoost$_\varrho$

Its more difficult to explain AdaBoost$_\varrho$ in the barrier optimization framework. In fact we encounter the same kind of problems as in the analysis before and observe that the convergence depends on the properties of the base learner. So we restrict ourself to present only the main idea and omit a detailed description of this issue, which is discussed in greater detail in Rätsch et al. [2000c].

The exponential barrier can also be used to solve the feasibility problem for convex programming:

$$\begin{aligned} &\text{find} \quad \hat{\boldsymbol{\alpha}} \\ &\text{with} \quad \rho_n(\hat{\boldsymbol{\alpha}}) \geq \varrho \qquad \forall n = 1 \ldots N, \end{aligned} \tag{2.20}$$

where ϱ is fixed now. We can argue as before and find that AdaBoost$_\varrho$ can be interpreted as a barrier algorithm for computing a separation of the training examples with margin at least ϱ. The barrier function of (2.20) is

$$\mathrm{F}_\beta(\hat{\boldsymbol{\alpha}}) = \beta \sum_{n=1}^{N} \exp\left(\frac{\varrho\langle \mathbf{1}, \hat{\boldsymbol{\alpha}}\rangle - U_{n,*}\hat{\boldsymbol{\alpha}}}{\beta\langle \mathbf{1}, \hat{\boldsymbol{\alpha}}\rangle}\right). \tag{2.21}$$

Originally, the β in front of the sum in (2.21) down-weights the constraint penalties against

11. For simplicity, we do not define the sub-gradient exactly, instead we use $\nabla_{\hat{\alpha}_j \geq 0} \mathrm{F}(\hat{\boldsymbol{\alpha}}) = \begin{cases} \nabla_{\hat{\alpha}_j} \mathrm{F}(\hat{\boldsymbol{\alpha}}) & \hat{\alpha}_j > 0 \\ \min(0, \nabla_{\hat{\alpha}_j} \mathrm{F}(\hat{\boldsymbol{\alpha}})) & \hat{\alpha}_j = 0 \end{cases}$. However, we note that it can in fact be done exactly leading to the same result. For some more details cf. Appendix A.3.3.

the objective. Thus, we can omit the β here (see discussion in Section 2.5.2). By using the same simplified optimization strategy (coordinate–wise descent) and setting $\beta \equiv \frac{1}{\langle \mathbf{1}, \hat{\boldsymbol{\alpha}} \rangle}$ in (2.21) as before, we obtain the formulation of AdaBoost$_\varrho$. As already shown in previous sections, one will quickly obtain a solution with margin at least ϱ (if $\langle \mathbf{1}, \hat{\boldsymbol{\alpha}} \rangle \to \infty$) and solve (2.20).

Although this connection does not lead to new theoretical results, it explains why Arc-GV and AdaBoost$_\varrho$ generate combined hypotheses with large margins: The exponential loss minimized by both algorithm acts as a barrier of some linear constraints resulting from the margin-LP problem. This idea will be exploited in later chapters to derive similarly motivated algorithms.

2.6 Discussion and Summary

We have analyzed a generalized version of AdaBoost in the context of large margin algorithms. From von Neumann's Min-Max theorem we found that the maximal achievable margin ρ^* is at least $\hat{\gamma}$, if the base learner always returns a hypothesis with weighted classification error less than $\frac{1}{2} - \frac{1}{2}\hat{\gamma}$. The asymptotical analysis lead us to a lower bound on the margin of hypotheses that are generated by AdaBoost$_\varrho$ in the limit, which was empirically shown to be tight in the worst case. Our results indicate that AdaBoost generally does not maximize the margin, but achieves a reasonable large margin.[12]

To overcome these problems we proposed an algorithm for which we have shown the convergence to the maximum margin solution. Roughly speaking this is achieved by increasing ϱ iteratively, such that the gap between the best and the worst case becomes arbitrarily small. In our analysis we did not need to assume additional properties of the base learning algorithm and in Rätsch and Warmuth [2001] we have shown that the analysis also holds for infinite hypothesis spaces. To the best of our knowledge this is the first algorithm that combines the merits of Boosting with the maximum margin solution.

Moreover, we found that some training examples, which are in the area of the decision boundary, have asymptotically the same margin. The other examples are effectively neglected in the learning process. We call these examples *Support Vectors* and, in fact, in another study we found a high overlap between the SVs generated by a boosting-type algorithm and a Support Vector Machine [cf. Rätsch et al., 2001].

In a simulation experiment we have illustrated the validity of our analysis and also that a larger margin can improve the generalization ability when learning on high dimensional data with a few informative dimensions. However, as we will discuss in detail in Chapter 4, the maximum margin approach is not the best choice when analyzing noisy data. Then, for instance, the data might not be separable and the theory of this chapter does not apply. We therefore analyze the convergence properties of boosting-type algorithms – also called *leveraging* – without the assumption of separable data in Chapter 3. Then in Chapter 4 we propose to relax the *hard margin* constraint and eventually find algorithms that generalize significantly better than AdaBoost on noisy data.

12. One can even construct cases where AdaBoost does not converge to a stable hypothesis, if the base learner is adversary.

3 On the Convergence of Leveraging

Whereas we have in the last section considered only AdaBoost, we extend our view now to *ensemble learning methods* with arbitrary convex cost functions – they are also called *leveraging algorithms* [Duffy and Helmbold, 1999]. We will relate these methods to numerical optimization techniques such as the Bregman algorithm and coordinate descent methods. These techniques will serve as useful tools to show the convergence of leveraging algorithms and also in later sections to develop *regularized* leveraging techniques.

We show the convergence of ensemble learning methods such as AdaBoost [Freund and Schapire, 1997], Logistic Regression (LR) [Friedman et al., 2000, Collins et al., 2000] and, as an example of a regression technique, the Least-Square regression algorithm called LS-Boost [Friedman, 1999]. These algorithms have in common that they iteratively call a base learning algorithm $\mathfrak{L}$ (also called *weak learner*) on a weighted training sample. The base learner is expected to return in each iteration t a hypothesis h_t from the hypothesis set H that has small *weighted training error* (see Chapter 2 for details). These hypotheses are then linearly combined to form the final or *combined hypothesis*

$$f_{\boldsymbol{\alpha}}(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}).$$

The hypothesis coefficient α_t is determined at iteration t, such that a certain objective is minimized or approximately minimized, and it is fixed for later iterations. We will work out sufficient conditions on the base learning algorithms to achieve *asymptotical* and *linear convergence* without assuming separability as in the last chapter.

For AdaBoost and the Logistic Regression algorithm Friedman et al. [2000] it has been shown [Collins et al., 2000] that they generate a combined hypothesis minimizing a loss functional G – in the limit as the number of iterations goes to infinity. The loss depends only on the output of the combined hypothesis $f_{\boldsymbol{\alpha}}$ on the training sample. However, the assumed conditions (discussed later in detail) in Collins et al. [2000] on the performance of the base learner are rather strict and can usually not be satisfied in practice. Although the part of the analysis in Collins et al. [2000] holds in principle for any strictly convex cost function of Legendre-type [Rockafellar, 1970, p. 258], one needs to show the existence of a so-called *auxiliary function* (cf. [Della Pietra et al., 1997, Collins et al., 2000], similar techniques have been used before in Littlestone et al. [1995], Kivinen and Warmuth [1997]) for each cost function other than the exponential or the logistic loss. In Section 3.2 we extend the analysis of Collins et al. [2000] and show that the auxiliary function exists under very mild assumptions on the base learning algorithm and the loss function.

In an earlier attempt to show the convergence of such methods for arbitrary loss functions [Mason et al., 2000], one needed to assume that the hypothesis coefficients α_t are upper

bounded by a rather small constant. For this case it has been shown that the algorithm asymptotically converges to a combined hypothesis minimizing G. However, since the α_t's need to be small, the algorithm requires many iterations to achieve this goal. Our analysis shows that this assumption is not necessary.

In Duffy and Helmbold [2000b] it has been shown that for loss functions which are (essentially) exponentially decreasing (including exponential and logistic loss), the loss is $\mathcal{O}(1/\sqrt{t})$ in the first $\tilde{t}$ iterations and afterwards $\mathcal{O}(\eta^{\tilde{t}-t})$. This only holds if the loss reaches zero – this is if the data is separable.

In this work we propose a family of algorithms that are able to generate a combined hypothesis f converging to the minimum of some loss functional $\mathrm{G}[f]$ (if it exists). Special cases are AdaBoost, Logistic Regression and LS-Boost. While assuming rather mild conditions on the base learning algorithm and the loss function G, we can show *linear convergence rates* [e.g. Luenberger, 1984] of the type $\mathrm{G}[f_{t+1}]-\mathrm{G}[f^*] \leq \eta(\mathrm{G}[f_t]-\mathrm{G}[f^*])$ for some fixed $\eta \in [0,1)$. This means that the difference to the minimum loss converges exponentially fast to zero (in the number of iterations). Similar convergence rates have been proven for AdaBoost in the special case of *separable data* [cf. Chapter 2 and Freund and Schapire, 1997], although the constant η shown in Freund and Schapire [1997] can be considerable smaller than in our analysis.

3.1 Leveraging algorithms

We first briefly review three of the most well known leveraging algorithms for classification and regression: AdaBoost, Logistic Regression and LS-Boost. For more details on the the latter two methods see e.g. Friedman et al. [2000], Friedman [1999], Duffy and Helmbold [1999]. We already introduced AdaBoost in detail in the previous chapters, for completeness, however, we will repeat some of its algorithmical details and adapt the notation.

We work with Algorithm 3.4 [Rätsch et al., 2002] as a template of a *generic* leveraging algorithm, since these algorithms have the same algorithmical structure.

3.1.1 AdaBoost & Logistic Regression

Both methods are designed for classification tasks. In each iteration they call a base learning algorithm on the training set $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\} \subseteq X \times \{-1, +1\}$ (cf. step 3a in Algorithm 3.4). Here a weighting $\mathbf{d}^{(t)} = (d_1^{(t)}, \ldots, d_N^{(t)})$ on the sample is used that is recomputed in each iteration t. The base learner is expected to return a hypothesis h_t from some finite hypothesis set[1] $\mathsf{H} := \{\hat{h}_j \mid \hat{h}_j : X \mapsto \{-1, +1\}, j = 1, \ldots, J\}$ that has a small *weighted classification error* $\epsilon_t = \sum_{n=1}^{N} |d_n^{(t)}| \mathbf{I}(y_n \neq h_t(\boldsymbol{x}_n))$ or equivalently a large *edge*, which is given by $\hat{\gamma}_t = \sum_{n=1}^{N} d_n^{(t)} h_t(\boldsymbol{x}_n)$. Notice that different from common convention and from the definition in Chapter 2, we include the label y_n in the weighting

1. Notice that H always contains only a finite number of *different* hypotheses when evaluated on the training set and is effectively finite [Bennett et al., 2000].

d_n to make the presentation simpler. Thus, d_n can also be negative. After selecting the hypothesis, its weight α_t is computed such that it minimizes a certain functional (cf. step 3b). In AdaBoost one minimizes

$$\mathrm{G}^{AB}(\alpha) = \sum_{n=1}^{N} \exp\left\{-y_n \left(\alpha h_t(\boldsymbol{x}_n) + f_{t-1}(\boldsymbol{x}_n)\right)\right\} \tag{3.1}$$

and in Logistic Regression

$$\mathrm{G}^{LR}(\alpha) = \sum_{n=1}^{N} \log\left\{1 + \exp\left(-y_n(\alpha h_t(\boldsymbol{x}_n) + f_{t-1}(\boldsymbol{x}_n))\right)\right\} , \tag{3.2}$$

where f_{t-1} is the combined hypothesis of the previous iteration given by

$$f_{t-1}(\boldsymbol{x}_n) = \sum_{r=1}^{t-1} \alpha_r h_r(\boldsymbol{x}_n). \tag{3.3}$$

For AdaBoost it has been shown that α_t in (3.1) can be computed analytically. For LR there exists an analytic approximation to the solution of (3.2). Based on the new combined hypothesis, the weighting $\mathbf{d}$ on the sample is updated:

$$d_n^{(t+1)} = y_n \exp(-y_n f_t(\boldsymbol{x}_n)) \qquad \text{and} \qquad d_n^{(t+1)} = y_n \frac{\exp(-y_n f_t(\boldsymbol{x}_n))}{1 + \exp(-y_n f_t(\boldsymbol{x}_n))}, \tag{3.4}$$

for AdaBoost and Logistic Regression, respectively. The initial weighting $\mathbf{d}^{(1)}$ is a fixed constant, which could in principle be arbitrary. In particular, $d_n^{(1)} = 1$ and $d_n^{(1)} = \frac{1}{2}$ $(n = 1, \ldots, 1)$, respectively, can be used resulting from (3.4), when setting $f_0 = \mathbf{0}$.

In Schapire and Singer [1999] the so-called *confidence rated boosting* has been proposed. Here one uses real valued base hypotheses. One finds the hypothesis weight α_t and the hypothesis h_t in parallel, such that (3.1) is minimized. This algorithm is similar to the next approach.

3.1.2 Least-Square-Boost

This algorithm is designed to solve regression tasks [Friedman, 1999]. A more detailed analysis of regression algorithms can be found in Chapter 5. In this case one has $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\} \subseteq X \times Y$ and some finite hypothesis set $\mathsf{H} := \{\hat{h}_j \mid \hat{h}_j : X \to Y, j = 1, \ldots, J\}$, where Y is a bounded subset of $\mathbb{R}$.[2] LS-Boost works in a similar way as AdaBoost and LR. It first selects a hypothesis solving

$$h_t = \operatorname*{argmin}_{h \in \mathsf{H}} \frac{1}{2} \sum\nolimits_{n=1}^{N} \left(d_n^{(t)} - h(\boldsymbol{x}_n)\right)^2, \tag{3.5}$$

2. We consider regression with one dimensional targets only.

Algorithm 3.4 A Leveraging algorithm for the loss function G.

1. **Input:** $S = \langle (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N) \rangle$, No. of Iterations T
   Loss function $\mathrm{G} : \mathbb{R}^N \to \mathbb{R}$
2. **Initialize:** $f_0 \equiv 0$, $d_n^{(1)} = \nabla \mathrm{G}[f_0]$
3. **Do for** $t = 1, \ldots, T$,
   (a) Train classifier on $\{S, \mathbf{d}^{(t)}\}$ and obtain hypothesis $h_t : X \to Y$
   (b) Set $\alpha_t = \operatorname{argmin}_{\alpha \in \mathbb{R}} \mathrm{G}[f_t + \alpha h_t]$
   (c) Update $f_{t+1} = f_t + \alpha_t h_t$ and $\mathbf{d}^{(t+1)} = \nabla \mathrm{G}[f_{t+1}]$
4. **Output:** f_T

for some appropriately defined d_n's. Then one finds the hypothesis weight α_t by minimizing the squared error of the new combined hypothesis:

$$\mathrm{G}^{LS}(\alpha) = \frac{1}{2} \sum\nolimits_{n=1}^{N} \left(y_n - f_{t-1}(\boldsymbol{x}_n) - \alpha h_t(\boldsymbol{x}_n) \right)^2 . \tag{3.6}$$

The "weighting" of the sample is computed as $d_n^{(t+1)} = y_n - f_t(\boldsymbol{x}_n)$, which is actually the residual of f_t [Friedman, 1999]. In a second version of LS-Boost, the base hypothesis and its weight are found simultaneously by solving [Friedman, 1999]:

$$[h_t, \alpha_t] = \operatorname*{argmin}_{\alpha \in \mathbb{R}, h \in \mathsf{H}} \frac{1}{2} \sum\nolimits_{n=1}^{N} \left(y_n - f_{t-1}(\boldsymbol{x}_n) - \alpha h(\boldsymbol{x}_n) \right)^2 \tag{3.7}$$

We call this the *maximum improvement* scheme. Since in (3.7) one reaches a lower loss function value than in (3.5) and (3.6), it might be the favorable strategy t achieve fast convergence.

3.1.3 The General Case

These algorithms can be summarized[3] in Algorithm 3.4 for an appropriate function-pair G and g′: plug-in $\mathrm{G}[f] = \sum_{n=1}^{N} \mathrm{g}(y_n, f(\boldsymbol{x}_n))$ and choose g as

1. $\mathrm{g}(y, f(\boldsymbol{x})) = \exp(-y f(\boldsymbol{x}))$ for AdaBoost (cf. (3.1)),
2. $\mathrm{g}(y, f(\boldsymbol{x})) = \log(1 + \exp(-y f(\boldsymbol{x})))$ for Logistic Regression (cf. (3.2)) and
3. $\mathrm{g}(y, f(\boldsymbol{x})) = \frac{1}{2}(y - f(\boldsymbol{x}))^2$ for LS-Boost (cf. (3.6)).

It can easily be verified that the function g′, used for computing the weights $\mathbf{d}$, is indeed the derivative of g with respect to the second argument [Breiman, 1999, Friedman, 1999, Mason et al., 2000], i.e. $\mathrm{g}'(y, \upsilon) := \nabla_\upsilon \mathrm{g}(y, \upsilon)$ (compare with AdaBoost, Logistic Regression and the LS-Boost algorithm).

It has been argued in Breiman [1999], Onoda et al. [1998], Friedman et al. [2000], Mason et al. [2000] and finally shown in Collins et al. [2000] that AdaBoost and Logistic

3. Here, the maximum improvement scheme as in (3.7) is slightly degenerated, but can be understood as a special case (cf. Section 3.4).

Regression asymptotically converge to a combined hypothesis f minimizing the respective loss G on the training sample, where f is a linear combination of hypotheses from H:

$$f_{\hat{\boldsymbol{\alpha}}} \in \operatorname{lin}(\mathsf{H}) := \left\{ \sum_{j=1}^{J} \hat{\alpha}_j \hat{h}_j \mid \hat{h}_j \in \mathsf{H}, \hat{\alpha}_j \in \mathbb{R} \right\}.$$

Thus, they solve the optimization problem:

$$\min_{f \in \operatorname{lin}(\mathsf{H})} \mathrm{G}[f] = \min_{\hat{\boldsymbol{\alpha}} \in \mathbb{R}^J} \mathrm{G}(H\hat{\boldsymbol{\alpha}}), \tag{3.8}$$

where we define a matrix $H \in \mathbb{R}^{N \times J}$ with $H_{nj} = \hat{h}_j(\boldsymbol{x}_n)$. We denote by $H_{n,*}$ and $H_{*,j}$ the n-th row and j-th column, respectively. To avoid confusions, note that the elements of the hypothesis set and their weights are indexed by j and marked with a hat, while hypotheses and coefficients generated during the iterative algorithm are not marked. In the algorithms discussed so far, the optimization takes place by employing the leveraging scheme outlined in Algorithm 3.4. The output of such an algorithm is a set of pairs (α_t, h_t) and a combined hypothesis $f_t(\boldsymbol{x}) = \sum_{r=1}^{t} \alpha_r h_r(\boldsymbol{x})$. With

$$\hat{\alpha}_j^{(t)} = \sum_{r=1}^{t} \alpha_r \mathbf{I}(h_r = \hat{h}_j), \quad j = 1, \ldots, J, \tag{3.9}$$

it is easy to verify that $\sum_{r=1}^{t} \alpha_r h_r(\boldsymbol{x}) = \sum_{j=1}^{J} \hat{\alpha}_j^{(t)} \hat{h}_j(\boldsymbol{x})$, which is in $\operatorname{lin}(\mathsf{H})$ (note the hat, cf. (3.3)).

3.1.4 Assumptions

In this chapter we will in some cases assume that the loss function G is of the form

$$\mathrm{G}[f_{\hat{\boldsymbol{\alpha}}}] = \mathrm{G}(H\hat{\alpha}) = \sum_{n=1}^{N} \mathrm{g}(y_n, f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}_n)),$$

Although this assumption is often not necessary, the presentation becomes easier. Note that additive loss functions are commonly used if one considers samples drawn i.i.d. from some source.

We assume that each element H_{nj} and y_n is finite ($j = 1, \ldots, J$, $n = 1, \ldots, N$) and H does not contain a zero column. Furthermore, the function $\mathrm{g}(y, \cdot) : \mathbb{R} \to \mathbb{R}$ is assumed to be strictly convex for all $y \in Y$.

For simplicity we assume for the rest of the chapter that H is finite and complementation closed ($\hat{h} \in \mathsf{H} \;\Rightarrow\; -\hat{h} \in \mathsf{H}$). The assumption on the finiteness is not a problem for classification, if the output of the base hypotheses is binary (cf. footnote 1). For regression problems the hypothesis set might be infinite. This case will explicitely be analyzed Section 5.3.

3.2 The Dual Algorithm and Bregman Distances

3.2.1 AdaBoost as Entropy Projection

AdaBoost has been derived from results on online-learning algorithms. In online learning one receives in each iteration one example, then predicts its label and receives a loss – based on the true label received after prediction. The important question in this setting is, how fast the algorithm is able to learn to produce predictions with small loss. In Cesa-Bianchi et al. [1994a], Kivinen and Warmuth [1997], Kivinen et al. [1997] the total loss, which is the sum of all losses over all examples, has been bounded in terms of the loss of an arbitrary comparator (e.g. the best predictor). To derive these results, Bregman divergencies [Bregman, 1967] and generalized projections have been extensively used. We will briefly introduce the key terms in Section 3.2.2.

In the case of boosting, one takes a dual view [Freund and Schapire, 1997]: Here the set of examples is fixed, whereas in each iteration the base learning algorithm generates a new hypothesis – which is understood as an "example" in the on-line learning setting. Using similar techniques as in the online-learning setting, the convergence in the online learning domain (we call it the *dual domain*) has been analyzed and lead to convergence results in the *primal domain* – similar to the ones presented in the previous chapter [for details see Freund and Schapire, 1997]. In the primal domain the hypothesis coefficients $\hat{\boldsymbol{\alpha}}$ and in the dual domain the weighting $\mathbf{d}$ are optimized.

AdaBoost has been understood as *entropy projection* in the dual domain [Kivinen and Warmuth, 1999, Lafferty, 1999]. The key observation is that the weighting $\mathbf{d}^{(t+1)}$ on the examples in the t-th iteration is computed as *generalized projection* of $\mathbf{d}^{(t)}$ onto a linear constraint, where one uses a generalized distance – here the *unnormalized relative entropy*. The update of the distribution is the solution to the following optimization problem:

$$\mathbf{d}^{(t+1)} = \operatorname*{argmin}_{\mathbf{d}\in\mathbb{R}_+^N} \sum_{n=1}^{N} d_n \log \frac{d_n}{d_n^{(t)}} + d_n - d_n^{(t)} \qquad (3.10)$$
$$\text{with} \quad \sum_{n=1}^{N} d_n y_n \hat{h}_t(\boldsymbol{x}_n) = 0,$$

Thus, the new weighting $\mathbf{d}^{(t+1)}$ in AdaBoost is chosen such that the edge of the previous hypothesis becomes zero [as e.g. observed in Freund and Schapire, 1997] and the (unnormalized) *relative entropy* between the new and the old distribution, say $\mathbf{d}^{(t+1)}$ and $\mathbf{d}^{(t)}$, is as small as possible [Kivinen and Warmuth, 1999].

Note that optimization problems like (3.10) appear in online learning as a special case of noise free data. More generally one uses a cost function, which is another Bregman divergency, to compute the update of the weight vector – instead of using a hard constraint.

The work of Kivinen and Warmuth [1999], Lafferty [1999] and later of Collins et al. [2000] lead to an understanding of Boosting methods in the context of Bregman distance optimization and generalized projections. In the following section we will first introduce the basic terms and notation. Then we briefly review the results of Collins et al. [2000] and Della Pietra et al. [2001]. We show that the rather restrictive assumptions made in Collins et al. [2000] are not necessary to prove the asymptotical convergence.

3.2.2 Generalized Distances and Generalized Projections

Let $\mathcal{S}$ be a nonempty, open, convex set, such that $\overline{\mathcal{S}} \subseteq \Lambda$, where $\Lambda \subseteq \mathbb{R}^N$ is the domain of a function $\mathcal{G} : \Lambda \to \mathbb{R}$ and $\overline{\mathcal{S}}$ denotes the closure of $\mathcal{S}$. Let $\tilde{\mathrm{g}} \equiv \nabla \mathcal{G}$. From $\mathcal{G}$ one can construct the *generalized distance function* $\Delta_{\mathcal{G}} : \overline{\mathcal{S}} \times \mathcal{S} \to \mathbb{R}$ by:

$$\Delta_{\mathcal{G}}(\boldsymbol{x}, \boldsymbol{y}) := \mathcal{G}(\boldsymbol{x}) - \mathcal{G}(\boldsymbol{y}) - \langle \tilde{\mathrm{g}}(\boldsymbol{x}), \boldsymbol{x} - \boldsymbol{y} \rangle. \tag{3.11}$$

The generalized distance has particularly interesting properties, if $\mathcal{G}$ is a Bregman function:

Definition 3.1 (Bregman Functions). *A function $\mathcal{G} : \Lambda \subseteq \mathbb{R}^N \to \mathbb{R}$ is called a Bregman function if there exists an open, convex set $\mathcal{S}$, such that $\overline{\mathcal{S}} \subseteq \Lambda$ and the following conditions hold:*

1. *$\mathcal{G}(\boldsymbol{x})$ has continuous first partial derivatives at every $\boldsymbol{x} \in \mathcal{S}$*
2. *$\mathcal{G}(\boldsymbol{x})$ is strictly convex and continuous on $\overline{\mathcal{S}}$*
3. *For every $\hat{\alpha} \in \mathbb{R}$, the partial level sets $\{\boldsymbol{x} \in \overline{\mathcal{S}} \mid \Delta_{\mathcal{G}}(\boldsymbol{x}, \hat{\boldsymbol{y}}) \leq \hat{\alpha}\}$ and $\{\boldsymbol{y} \in \mathcal{S} \mid \Delta_{\mathcal{G}}(\hat{\boldsymbol{x}}, \boldsymbol{y}) \leq \hat{\alpha}\}$ are bounded, for every $\hat{\boldsymbol{y}} \in \mathcal{S}$ and $\hat{\boldsymbol{x}} \in \overline{\mathcal{S}}$*
4. *If $\boldsymbol{y}^{(t)} \in \mathcal{S}$, $t = 1, 2, \ldots$ and $\lim_{t\to\infty} \boldsymbol{y}^{(t)} = \boldsymbol{y}^*$ then $\lim_{t\to\infty} \Delta_{\mathcal{G}}(\boldsymbol{y}^*, \boldsymbol{y}^{(t)}) = 0$*
5. *If $\boldsymbol{y}^{(t)} \in \mathcal{S}$ and $\boldsymbol{x}^{(t)} \in \overline{\mathcal{S}}$ for all $t = 1, 2, \ldots$, and if $\lim_{t\to\infty} \Delta_{\mathcal{G}}(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}) = 0$, and $\lim_{t\to\infty} \boldsymbol{y}^{(t)} = \boldsymbol{y}^*$, and $\{\boldsymbol{x}^{(t)}\}$ is bounded, then $\lim_{t\to\infty} \boldsymbol{x}^{(t)} = \boldsymbol{y}^*$.*

Here we followed the definition given in Censor and Zenios [1997]. The original definition was given in Bregman [1967] and alternate definitions are given in e.g. Herbster and Warmuth [01], Bauschke and Borwein [1997].[4]

Among several other useful properties, we have $\Delta_{\mathcal{G}}(\boldsymbol{x}, \boldsymbol{y}) > 0$ for every $\boldsymbol{x} \neq \boldsymbol{y}$ and $\Delta_{\mathcal{G}}(\boldsymbol{x}, \boldsymbol{y}) = 0$ only if $\boldsymbol{x} = \boldsymbol{y}$ (cf. Figure 3.1).

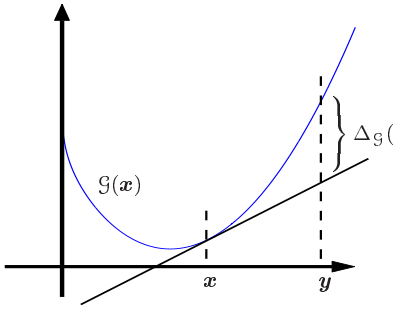

Figure 3.1 Illustration of how the generalized distance is defined: $\mathcal{G}(\boldsymbol{x})$ is a convex function. Given two points $\boldsymbol{x}$ and $\boldsymbol{y}$, the Bregman distance with respect to $\mathcal{G}$ is given as the difference of the function values minus the difference of a linear approximation. Since $\mathcal{G}$ is strictly convex, this generalized distance between $\boldsymbol{x}$ and $\boldsymbol{y}$ is always positive and zero, if and only if $\boldsymbol{x} = \boldsymbol{y}$.

A key role in our algorithms is played by generalized projections onto hyperplanes (cf. Figure 3.2):

Definition 3.2 (Generalized Projection onto a Hyperplane, e.g. Censor and Zenios [1997]). *Given a hyperplane $\tilde{\mathrm{H}} = \{\boldsymbol{x} \mid \langle \mathbf{h}, \boldsymbol{x} \rangle = b\}$, a Bregman function $\mathcal{G}$ on $\mathcal{S}$ and a point $\boldsymbol{y} \in \mathcal{S}$. The point $\mathrm{P}_{\tilde{\mathrm{H}}}(\boldsymbol{y}) = \boldsymbol{x}^* \in \tilde{\mathrm{H}} \cap \overline{\mathcal{S}}$ is called the (generalized) projection of $\boldsymbol{y}$*

4. Note that only the last three conditions are neccessary to let generalized projections be uniquely defined.

onto $\tilde{\mathrm{H}}$*, if it satisfies*

$$\min_{\boldsymbol{z}\in\tilde{\mathrm{H}}\cap\overline{\mathcal{S}}} \Delta_{\mathcal{G}}(\boldsymbol{z},\boldsymbol{y}) = \Delta_{\mathcal{G}}(\boldsymbol{x}^*,\boldsymbol{y}). \tag{3.12}$$

Furthermore, the hyperplane $\tilde{\mathrm{H}}$ *is called zone consistent, if for all* $\boldsymbol{y}\in\mathcal{S}$ *holds* $\mathrm{P}_{\tilde{\mathrm{H}}}(\boldsymbol{y})\in\mathcal{S}$.

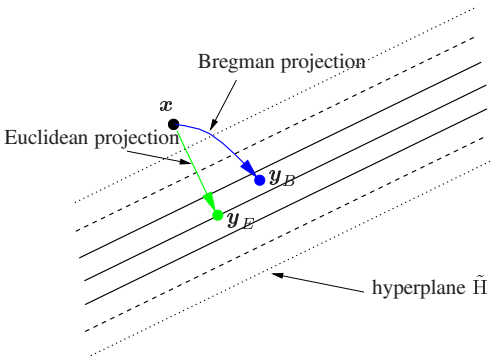

Figure 3.2 Illustration of generalized projections: one projects a point $\boldsymbol{x}$ onto a plane $\tilde{\mathrm{H}}$ by finding the point $\boldsymbol{y}$ on the plane that has the smallest generalized distance from $\boldsymbol{x}$, i.e. smallest $\Delta_{\mathcal{G}}(\boldsymbol{x},\boldsymbol{y})$. If $\mathcal{G}(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|_2^2$, then the generalized distance is equal to the squared Euclidean distance, hence, the projected point $\boldsymbol{y}_E$ is the closest (in the common sense) point on the hyperplane $\tilde{\mathrm{H}}$. For another Bregman function $\mathcal{G}$, one finds another projected point $\boldsymbol{y}_B$. Then closeness is measured differently.

Lemma 3.1 (Censor and Zenios [1997]). *Given a Bregman function* $\mathcal{G}$ *on* $\mathcal{S}$ *and a zone consistent hyperplane* $\tilde{\mathrm{H}} = \{\boldsymbol{x} \mid \langle \mathbf{h}, \boldsymbol{x}\rangle = b\}$*. Let* $\tilde{\mathrm{g}} = \nabla\mathcal{G}$*. Then for any* $\boldsymbol{y}\in\mathcal{S}$*, the system*

$$\tilde{\mathrm{g}}(\boldsymbol{x}^*) = \tilde{\mathrm{g}}(\boldsymbol{y}) + \lambda^*\mathbf{h} \tag{3.13}$$

$$\langle \mathbf{h}, \boldsymbol{x}^*\rangle = b \tag{3.14}$$

determines uniquely the point $\boldsymbol{x}^*$*. If* $\mathbf{h}\neq\mathbf{0}$*, then* λ^* *is uniquely determined.*

3.2.3 Generalized Projections onto Intersections of Hyperplanes

In this section we briefly review the results of Bregman [1967], Della Pietra et al. [2001] and Collins et al. [2000] to iteratively solve the problem of projecting a point onto the intersection of several hyperplanes. We are interested in the following type of optimization problem:

$$\begin{aligned} \min_{\mathbf{d}\in\overline{\mathcal{S}}} \quad & \Delta_{\mathcal{G}}(\mathbf{d},\mathbf{d}^{(0)}) \\ \text{with} \quad & H^\top\mathbf{d} = H^\top\overline{\mathbf{d}} = -\boldsymbol{\gamma}. \end{aligned} \tag{3.15}$$

where $\mathcal{G}$ is a Bregman function on $\mathcal{S}$, $\mathbf{d}^{(0)}$ is an comparator with $\mathbf{d}^{(0)}\in\mathcal{S}$ and H is a matrix of size $N\times J$. The vector $\boldsymbol{\gamma}\in\mathbb{R}^J$ is chosen such that a solution to the equality constraint exists: $-\boldsymbol{\gamma} = H^\top\overline{\mathbf{d}}$ for some $\overline{\mathbf{d}}\in\overline{\mathcal{S}}$. We use $-\boldsymbol{\gamma}$ instead of γ to be consistent with the next sections. The equality constraint in (3.15) is satisfied if and only if $\mathbf{d}$ is in the intersection of the J hyperplanes defined by $\tilde{\mathrm{H}}_j = \{\boldsymbol{x} \mid \langle H_{*,j}, \boldsymbol{x}\rangle = -\gamma_j\}$, $j = 1,\ldots,J$.

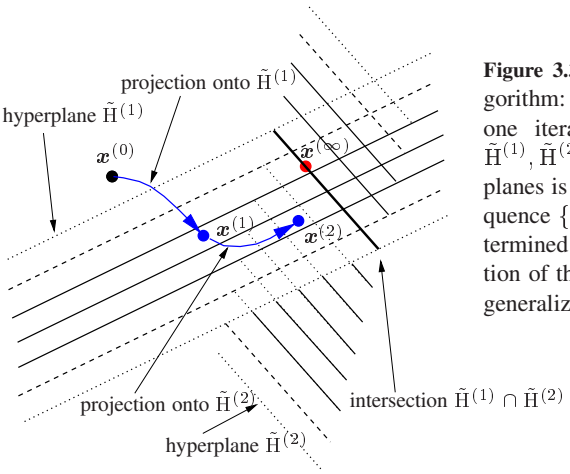

Figure 3.3 Illustration of the Bregman algorithm: starting from any point $\boldsymbol{x}^{(0)}$, one iteratively projects onto hyperplanes $\tilde{\mathrm{H}}^{(1)}, \tilde{\mathrm{H}}^{(2)}, \ldots$. If the sequence of hyperplanes is chosen (almost) cyclic, then the sequence $\{\boldsymbol{x}^{(t)}\}$ converges to the uniquely determined point $\boldsymbol{x}^{(\infty)}$ located in the intersection of the hyperplanes and has the smallest generalized distance to $\boldsymbol{x}^{(0)}$.

Bregman [1967] has proposed an algorithm for solving such problems. The algorithm starts at $\mathbf{d}^{(0)}$, selects a hyperplane j_1 and projects $\mathbf{d}^{(0)}$ onto $\tilde{\mathrm{H}}_{j_1}$ – leading to $\mathbf{d}^{(1)}$. Then the next hyperplane j_2 is chosen onto which $\mathbf{d}^{(1)}$ is projected, and so on. It has been shown that this algorithm – called the Bregman algorithm – converges to the optimal solution, if one chooses the hyperplanes cyclically [for details see Bregman, 1967, Censor and Zenios, 1997]. In the rest of this section we try to relax the restrictions on the selection scheme, since in particular for ensemble learning a cyclic order would not be very useful.

We start with a theorem[5] characterizing the solution of (3.15):

Theorem 3.1 (Della Pietra et al. [2001]). *Let $\mathcal{G}$ be a Bregman function on $\mathcal{S} \subseteq \mathbb{R}^N$, $\mathbf{d}^{(0)} \in \mathcal{S}$, $\overline{\mathbf{d}} \in \overline{\mathcal{S}}$, $H \in \mathbb{R}^{N \times J}$. Furthermore let $\tilde{\mathrm{g}} = \nabla \mathcal{G}$, $\mathrm{g} = \tilde{\mathrm{g}}^{-1}$, and*

$$\mathcal{M} = \left\{ \mathrm{g}\left(\tilde{\mathrm{g}}(\mathbf{d}^{(0)}) - H\hat{\boldsymbol{\alpha}} \right) \mid \hat{\boldsymbol{\alpha}} \in \mathbb{R}^J \right\}, \quad \textit{and} \quad \mathcal{N} = \left\{ \mathbf{d} \in \overline{\mathcal{S}} \mid H^\top \mathbf{d} = H^\top \overline{\mathbf{d}} \right\}.$$

Assume $\Delta_{\mathcal{G}}(\overline{\mathbf{d}}, \mathbf{d}^{(0)}) < \infty$ and $\mathcal{N}$ is not empty. Then there exists a unique $\mathbf{d}^* \in \overline{\mathcal{S}}$ satisfying

1. $\mathbf{d}^* \in \overline{\mathcal{M}} \cap \mathcal{N}$
2. $\mathbf{d}^* = \operatorname{argmin}_{\mathbf{d} \in \overline{\mathcal{M}}} \Delta_{\mathcal{G}}(\overline{\mathbf{d}}, \mathbf{d})$
3. $\mathbf{d}^* = \operatorname{argmin}_{\mathbf{d} \in \mathcal{N}} \Delta_{\mathcal{G}}(\mathbf{d}, \mathbf{d}^{(0)})$.

Moreover, any one of these three properties determines $\mathbf{d}^*$ uniquely.

For a proof of convergence for a more relaxed selection scheme than in the Bregman algorithm we need the *auxiliary function technique* introduced in Della Pietra et al. [1997]:[6]

5. We found this theorem in Collins et al. [2000], but could not verify the proof, since the corresponding publication [Della Pietra et al., 2001] was not available. It has been proven for the normalized relative entropy in Della Pietra et al. [1997]. A minor change in the definition of Bregman functions might be necessary: One needs to assume that the Bregman function is of Legendre type (cf. Bauschke and Borwein [1997], private communication with John Lafferty). In particular the generalized distance $\Delta_{\mathcal{G}}$ needs to be defined on $\overline{\mathcal{S}} \times \overline{\mathcal{S}}$ (and not only on $\overline{\mathcal{S}} \times \mathcal{S}$), since otherwise the solution of $\min_{\mathbf{d} \in \overline{\mathcal{M}}} \Delta_{\mathcal{G}}(\boldsymbol{\gamma}, \mathbf{d})$ might not exist. This can e.g. be the case where the dual variables $\hat{\boldsymbol{\alpha}}$ tend to infinity. For the rest of this section we assume the correctness of Theorem 3.1.

6. Similar techniques have been used before, see e.g. Herbster and Warmuth [01] and references therein.

Definition 3.3 (Auxiliary Function, Della Pietra et al. [1997], Collins et al. [2000]). *The continuous function $\mathcal{A} : \mathcal{S} \to \mathbb{R}_+$, $\mathcal{S} \subseteq \mathbb{R}^N$, is called an auxiliary function for a sequence $\mathbf{d}^{(1)}, \mathbf{d}^{(2)}, \ldots$ and a matrix $H \in \mathbb{R}^{N \times J}$, if it satisfies the two conditions:*

$$\Delta_{\mathcal{G}}(\boldsymbol{\gamma}, \mathbf{d}^{(t)}) - \Delta_{\mathcal{G}}(\boldsymbol{\gamma}, \mathbf{d}^{(t+1)}) \geq \mathcal{A}(\mathbf{d}^{(t)}) \geq 0 \tag{3.16}$$

$$\mathcal{A}(\mathbf{d}) = 0 \quad \Rightarrow \quad H^\top \mathbf{d} = H^\top \overline{\mathbf{d}} = -\boldsymbol{\gamma}, \tag{3.17}$$

where $\mathcal{G}$ is a Bregman function on $\mathcal{S}$.

If one can define an auxiliary function, one can show the asymptotical convergence:

Lemma 3.2 (Collins et al. [2000], Della Pietra et al. [1997]). *Let $\mathcal{A}$ be an auxiliary function for $\mathbf{d}^{(1)}, \mathbf{d}^{(2)}, \ldots$ and matrix H. Assume the $\mathbf{d}^{(t)}$'s lie in a compact subspace of $\overline{\mathcal{M}}$ as defined in Theorem 3.1. Then $\lim_{t\to\infty} \mathbf{d}^{(t)} = \mathbf{d}^* = \operatorname{argmin}_{\mathbf{d}\in\overline{\mathcal{M}}} \Delta_{\mathrm{G}}(\overline{\mathbf{d}}, \mathbf{d})$.*

We give the proof following Collins et al. [2000] for completeness:

Proof By (3.16), $\Delta_{\mathcal{G}}(\overline{\mathbf{d}}, \mathbf{d}^{(t)})$ is a non-increasing sequence bounded from below by 0. Therefore, the sequence of differences $\Delta_{\mathcal{G}}(\overline{\mathbf{d}}, \mathbf{d}^t) - \Delta_{\mathcal{G}}(\overline{\mathbf{d}}, \mathbf{d}^{t+1}) \geq 0$ converges to zero. By (3.16), $\mathcal{A}(\mathbf{d}^{(t)})$ must also converge to zero. Since the $\mathbf{d}^{(t)}$'s lie in a compact space, the sequence of $\mathbf{d}^{(t)}$'s must have at least one limit point $\tilde{\mathbf{d}} \in S$. By continuity of $\mathcal{A}$, we have $\mathcal{A}(\tilde{\mathbf{d}}) = 0$. Therefore, $\tilde{\mathbf{d}} \in \mathcal{N}$ by (3.17). Thus $\tilde{\mathbf{d}} \in \mathcal{M} \cap \overline{\mathcal{N}}$ and $\tilde{\mathbf{d}} = \mathbf{d}^*$ by Theorem 3.1. This argument and the uniqueness of $\mathbf{d}^*$ show that the sequence of $\mathbf{d}^{(t)}$'s can only have one limit point $\mathbf{d}^*$. ■

Lemma 3.2 can e.g. be applied to show the convergence of the following three extensions to the Bregman algorithm:

(1) for noncyclic selection of the hyperplanes,

(2) for the case of a *approximate projections* onto hyperplanes,[7] and

(3) for the selection of several hyperplanes at each iteration, i.e. for so-called *parallel updates*.

These extension have been analyzed in [Collins et al., 2000], however, only for the exponential and logistic loss. Furthermore, in (1) they require that one selects the hyperplane with *maximum constraint* violation. This requirement is hard to meet in practice.

We now consider these extensions in greater detail and will e.g. find much more relaxed conditions on the selection scheme. For simplicity we assume for the rest of this section that the system of equalities is of the form $H^\top \mathbf{d} = \mathbf{0}$ (i.e. $\boldsymbol{\gamma} = \mathbf{0}$). The extension to the general case is straightforward.

To solve (3.15), one may project $\mathbf{d}^{(t-1)}$ in each iteration t onto a hyperplane $\tilde{\mathrm{H}}^{(t)} = \{\boldsymbol{x} \mid \langle \boldsymbol{x}, \mathbf{h}^{(t)} \rangle = 0\}$. Here $\mathbf{h}^{(t)}$ is not restricted to be one of the columns of H. We also allow that $\mathbf{h}^{(t)}$ is a linear combination of columns of H, i.e. $\mathbf{h}^{(t)}$ can be expressed as $\mathbf{h}^{(t)} = H\boldsymbol{\beta}$ for some $\boldsymbol{\beta}$. Assume $\mathcal{G}$ is zone consistent with $\tilde{\mathrm{H}}^{(t)}$. Then we can consider sequences of the type $\mathbf{d}^{(t)} = \mathrm{g}(\mathrm{G}(\mathbf{d}^{(t-1)}) + \hat{\alpha}_t \mathbf{h}^{(t)})$, and $\hat{\alpha}_t$ is chosen such that $\langle \mathbf{d}^{(t)}, \mathbf{h}^{(t)} \rangle = 0$.

7. We define an approximate projection of $\mathbf{d}$ onto $\tilde{\mathrm{H}}$ as the projection of $\mathbf{d}$ onto some parallel hyperplane to $\tilde{\mathrm{H}}$.

Then holds,

$$\begin{aligned}\Delta_{\mathcal{G}}(\mathbf{0},\mathbf{d}^{(t)})-\Delta_{\mathcal{G}}(\mathbf{0},\mathbf{d}^{(t+1)}) &= \mathcal{G}(\mathbf{d}^{(t+1)})-\mathcal{G}(\mathbf{d}^{(t)})+\mathbf{d}^{(t)}\tilde{\mathrm{g}}(\mathbf{d}^{(t)})-\mathbf{d}^{(t+1)}\tilde{\mathrm{g}}(\mathbf{d}^{(t+1)})\\ &= \Delta_{\mathcal{G}}(\mathbf{d}^{(t+1)},\mathbf{d}^{(t)})-\hat{\alpha}_{t+1}\langle\mathbf{d}^{(t+1)},\mathbf{h}^{(t+1)}\rangle\end{aligned}$$

We have $\mathbf{d}^{(t)}=\mathrm{g}\left(\tilde{\mathrm{g}}(\mathbf{d}^{(0)})+\sum_{r=1}^{t}\hat{\alpha}_r\mathbf{h}^{(r)}\right)$ and can write

$$\begin{aligned}\Delta_{\mathcal{G}}(\mathbf{d}^{(t+1)},\mathbf{d}^{(t)}) &= \Delta_{\mathcal{G}}\left(\mathrm{g}\left(\tilde{\mathrm{g}}(\mathbf{d}^{(0)})+\sum_{r=1}^{t+1}\hat{\alpha}_r\mathbf{h}^{(r)}\right),\mathrm{g}\left(\tilde{\mathrm{g}}(\mathbf{d}^{(0)})+\sum_{r=1}^{t}\hat{\alpha}_r\mathbf{h}^{(r)}\right)\right)\\ &= \mathrm{G}\left(\tilde{\mathrm{g}}(\mathbf{d}^{(0)})+\sum_{r=1}^{t}\hat{\alpha}_r\mathbf{h}^{(r)}\right)-\mathrm{G}\left(\tilde{\mathrm{g}}(\mathbf{d}^{(0)})+\sum_{r=1}^{t+1}\hat{\alpha}_r\mathbf{h}^{(r)})\right)\\ &\quad+\hat{\alpha}_{t+1}\langle\mathbf{d}^{(t+1)},\mathbf{h}^{(t+1)}\rangle,\end{aligned}\tag{3.18}$$

where G is the conjugate function of $\mathcal{G}$. Since the function $\mathcal{G}$ is differentiable, its conjugate is $\mathrm{G}(\boldsymbol{z})=\langle\boldsymbol{z},\mathrm{g}(\boldsymbol{z})\rangle-\mathcal{G}(\mathrm{g}(\boldsymbol{z}))$ [Rockafellar, 1970]. Thus,

$$\Delta_{\mathrm{G}}(\mathbf{0},\mathbf{d}^{(t)})-\Delta_{\mathrm{G}}(\mathbf{0},\mathbf{d}^{(t+1)})=\mathrm{G}\left(\tilde{\mathrm{g}}(\mathbf{d}^{(0)})+\sum_{r=1}^{t}\hat{\alpha}_r\mathbf{h}^{(r)}\right)-\mathrm{G}\left(\tilde{\mathrm{g}}(\mathbf{d}^{(0)})+\sum_{r=1}^{(t+1)}\hat{\alpha}_r\mathbf{h}^{(r)}\right),\tag{3.19}$$

where the rhs. is the difference (or progress) in the conjugate function G.

We propose the following function

$$\mathcal{A}^{\Delta}(\mathbf{d}^{(t)}):=\sum_{j=1}^{J}\left(\mathrm{G}\left(\tilde{\mathrm{g}}(\mathbf{d}^{(t)})\right)-\mathrm{G}\left(\tilde{\mathrm{g}}(\mathbf{d}^{(t+1,j)})\right)\right),\tag{3.20}$$

where $\mathbf{d}^{(t+1,j)}$ is the generalized projection of $\mathbf{d}^{(t)}$ onto the j-th hyperplane, i.e. $\tilde{\mathrm{g}}(\mathbf{d}^{(t+1,j)})=\tilde{\mathrm{g}}(\mathbf{d}^{(t)})+\hat{\alpha}^{(t+1,j)}H_{*,j}$, where $\hat{\alpha}^{(t+1,j)}$ is chosen such that $\langle H_{*,j},\mathbf{d}^{(t+1,j)}\rangle=0$. Thus $\mathcal{A}^{\Delta}(\mathbf{d}^{(t)})$ is the sum of the improvements in G that can be achieved by projecting onto each single hyperplane with normal vector $H_{*,j}$, $j=1,\ldots,J$. Then we have

Proposition 3.1. *Let $H\in\mathbb{R}^{N\times J}$, $\mathcal{G}$ be a Bregman function on $\mathcal{S}$, and $\tau:\mathbb{R}_+\to\mathbb{R}_+$ be a strictly monotonically increasing and continuous function with $\tau(0)=0$. Let $\{\mathbf{d}^{(t)}\}$ be a sequence of points with $\mathbf{d}^{(0)}\in\mathcal{S}$ and $\{\tilde{\mathrm{H}}^{(t)}\}$ be a sequence of hyperplanes with $\tilde{\mathrm{H}}^{(t)}=\{\boldsymbol{x}\mid\langle\boldsymbol{x},\mathbf{h}^{(t)}\rangle=0\}$ satisfying*

$$\mathrm{G}\left(\tilde{\mathrm{g}}(\mathbf{d}^{(t)})\right)-\mathrm{G}\left(\tilde{\mathrm{g}}(\mathbf{d}^{(t+1)})\right)\geq\tau(\mathcal{A}^{\Delta}(\mathbf{d}^{(t)})),\tag{3.21}$$

and $\mathbf{d}^{(t)}$ is the generalized projection of $\mathbf{d}^{(t-1)}$ onto $\tilde{\mathrm{H}}^{(t)}$ for all $t=1,2,\ldots$. Assume $\mathcal{G}$ is zone consistent with $\tilde{\mathrm{H}}^{(t)}$, $t=1,2,\ldots$, and $\mathbf{h}^{(t)}$ is a linear combination of columns of H. Then $\tau(\mathcal{A}^{\Delta}(\mathbf{d}^{(t)}))$ is an auxiliary function for $\{\mathbf{d}^{(t)}\}$ and H.

Proof Since $\Delta_{\mathcal{G}}(\mathbf{0},\mathbf{d}^{(t)})-\Delta_{\mathcal{G}}(\mathbf{0},\mathbf{d}^{(t+1)})=\mathrm{G}\left(\tilde{\mathrm{g}}(\mathbf{d}^{(t)})\right)-\mathrm{G}\left(\tilde{\mathrm{g}}(\mathbf{d}^{(t+1)})\right)$, condition (3.16) is satisfied. Since $\tilde{\mathrm{H}}^{(t)}$ is a linear combination of columns of H, holds $\mathbf{d}^{(t)}\in\overline{\mathcal{M}}$ as defined in Theorem 3.1. Also condition (3.17) can easily be verified. Since $\tilde{\mathrm{g}}$ is continuous,

the generalized projection is continuous in $\mathbf{d}$ (cf. (3.13) and [Lafferty et al., 1997, Lemma 2]). Moreover, since τ is continuous, $\tau(A^{\Delta}(\mathbf{d}^{(t)}))$ is continuous. ■

Proposition 3.1 defines an auxiliary function in terms of the progress in the dual domain, i.e. in G. The proof can easily be extended to the case where one only approximately projects onto the hyperplane (cf. extension (2) mentioned above).

We are now going to define an auxiliary function based on a relation of $\mathbf{h}^{(t)}$ to the constraint violation at $\mathbf{d}^{(t)}$. We propose the following function:[8]

$$\mathcal{A}^{\nabla}(\mathbf{d}) = \|H^{\top}\mathbf{d}\|_2, \tag{3.22}$$

which is the norm of constraint violations. We can prove the following:

Proposition 3.2. *Let H, $\mathcal{G}$ and τ be as in Proposition 3.1. Assume $\mathrm{g} = (\nabla\mathcal{G})^{-1}$ is continuously differentiable on $\{\nabla\mathcal{G}(\mathbf{d}) \mid \mathbf{d} \in \mathcal{S}\}$. Let $\{\mathbf{d}^{(t)}\}$ be a sequence of points with $\mathbf{d}^{(0)} \in \mathcal{S}$ and $\{\tilde{\mathrm{H}}^{(t)}\}$ be a sequence of hyperplanes with $\tilde{\mathrm{H}}^{(t)} = \{\boldsymbol{x} \mid \langle \boldsymbol{x}, \mathbf{h}^{(t)}\rangle = 0\}$ satisfying*

$$|\langle \mathbf{d}^{(t)}, \mathbf{h}^{(t+1)}\rangle| \geq \tau(\mathcal{A}^{\nabla}(\mathbf{d}^{(t)})), \tag{3.23}$$

where $\mathbf{d}^{(t)}$ is the projection of $\mathbf{d}^{(t-1)}$ onto the hyperplane $\tilde{\mathrm{H}}^{(t)}$. Assume $\mathcal{G}$ is zone consistent with $\tilde{\mathrm{H}}^{(t)}$, $\mathbf{h}^{(t)}$ is a linear combination of columns of H and the sequence $\{\mathbf{h}^{(t)}\}$ is bounded (for all $t = 1, \ldots, T$). Then there exists an auxiliary function for $\{\mathbf{d}^{(t)}\}$ and H.

The proof is shown in Appendix A.2.1 on page 138.

3.2.4 Summarizing Remarks

We have worked out two very mild, but sufficient requirements on the sequence of hyperplanes, such that the sequence of $\mathbf{d}^{(t)}$'s converges to the optimal solution.

If one "projects" on single columns of H, then the algorithm converges, as long as one selects in each iteration a column that leads to an improvement in G that is not arbitrarily small compared to the average improvement of all other columns. We have shown that this condition is satisfied, if one selects a column that corresponds to a constraint whose violation is not arbitrary small compared to the average violation of the other constraints.

Our analysis also includes the case where one does not project onto a hyperplane with normal vector equal to a column of H. It is also allowed to use hyperplanes that are "linear combinations" of the constraining hyperplanes (parallel update). However, in the following sections we will not use the results on parallel updates, since we are particularly interested in the analysis of our generic leveraging algorithm (cf. Algorithm 3.4).

8. Note the difference of the lhs. in (3.20) and (3.22): we use $\mathcal{A}^{\Delta}$ and $\mathcal{A}^{\nabla}$.

3.3 Coordinate Descent

So far we have considered the optimization in the domain of $\mathbf{d}^{(t)}$'s. We now take a dual view. Here the optimization is in terms of the hypothesis coefficients $\hat{\boldsymbol{\alpha}}$. We will show that the generalized projection onto a constraining hyperplane is equivalent to the optimization in one variable, i.e. in one coordinate direction. Optimization methods minimizing only along one coordinate direction at a time are called *coordinate descent methods*.

In this section we consider the following type of optimization problem:

$$\min_{\hat{\boldsymbol{\alpha}}} \mathrm{G}(H\hat{\boldsymbol{\alpha}}) + \langle \boldsymbol{\gamma}, \hat{\boldsymbol{\alpha}} \rangle, \tag{3.24}$$

where G is assumed to be strictly convex and twice continuously differentiable. To solve (3.24), one solves a sequence of one dimensional problems of the type:

$$\min_{\hat{\alpha}} \mathrm{G}_j(\hat{\alpha}) + \gamma_j \hat{\alpha}, \tag{3.25}$$

where $\mathrm{G}_j(\hat{\alpha}) = \mathrm{G}(H(\hat{\alpha}_1^{(t)}, \ldots, \hat{\alpha}_{j-1}^{(t)}, \hat{\alpha}_j^{(t)} + \hat{\alpha}, \hat{\alpha}_{j+1}^{(t)}, \ldots, \hat{\alpha}_J^{(t)})^\top)$. Here the question of how to choose the order of the coordinates arises. The most well-known method is called the (non-linear) *Gauss-Seidel* method. It selects the coordinates cyclically. This is certainly infeasible in our case. A more appropriate method is the *Gauss-Southwell* (GS) method, which selects the coordinate with *largest* absolute gradient. In the sequel we will consider methods related to the GS method. Under mild assumptions on the coordinate selection scheme $j_1, j_2, \ldots$ and the function G, we show that the sequence $\{\hat{\boldsymbol{\alpha}}^{(t)}\}$ obtained by iteratively minimizing with respect to the variable $\hat{\alpha}_{j_t}$ – starting at some $\hat{\boldsymbol{\alpha}}^{(0)}$ – converges linearly to the optimal solution of (3.25).

3.3.1 Relation to Generalized Projections

We briefly illustrate the connection between the projection onto a hyperplane and the minimization with respect to one variable. A similar observation has been made by Kiwiel [1998]. We use a Bregman function $\mathcal{G}$ as defined in Section 3.2.2 and assume G is its conjugate.

The solution of (3.25) satisfies $\frac{\partial \mathrm{G}_j(\hat{\alpha})}{\partial \hat{\alpha}} + \gamma_j = 0$. We may rewrite $\mathrm{G}_j(\hat{\alpha}) = \mathrm{G}(H\hat{\boldsymbol{\alpha}}^o + \hat{\alpha} H_{*,j})$, where $\hat{\boldsymbol{\alpha}}^o$ is the starting vector and $H_{*,j}$ is the j-th column of H. Thus the optimal $\hat{\alpha}^*$ satisfies

$$\frac{\partial \mathrm{G}(H\hat{\boldsymbol{\alpha}}^o + \hat{\alpha}^* \mathbf{h})}{\partial \hat{\alpha}} + b = \langle \mathbf{h}, \mathrm{g}(H\hat{\boldsymbol{\alpha}}^o + \hat{\alpha}^* H_{*,j}) \rangle + \gamma_j = 0 \tag{3.26}$$

where $\mathrm{g} \equiv \nabla \mathrm{G}$. If $H_{*,j} \neq \mathbf{0}$ and the solution to (3.25) exists, then it is uniquely determined by (3.26), since G is strictly convex.

Let $\mathbf{d}^*$ be the generalized projection of $\mathbf{d} = \tilde{\mathrm{g}}(H\hat{\boldsymbol{\alpha}}^o)$ onto a hyperplane $\tilde{\mathrm{H}} = \{\boldsymbol{x} \mid \langle H_{*,j}, \boldsymbol{x} \rangle = -\gamma_j\}$. According to Lemma 3.1, $\mathbf{d}^*$ and the step size λ^* are uniquely determined by (3.13)–(3.14), if $\mathcal{G}$ is zone consistent with the hyperplane $\tilde{\mathrm{H}}$. By (3.13) we have $\mathbf{d}^* = \mathrm{g}(\tilde{\mathrm{g}}(\mathbf{d}) + \lambda^* H_{*,j})) = \mathrm{g}(H\hat{\boldsymbol{\alpha}}^o + \lambda^* H_{*,j})$. Using (3.14), it is shown that $\langle \mathbf{h}, \mathrm{g}(H\hat{\boldsymbol{\alpha}}^o + \lambda^* H_{*,j}) \rangle = -\gamma_j$. By uniqueness under the given conditions, we can conclude $\lambda^* = \hat{\alpha}^*$.

Thus, the optimization of the j-th coordinate is equivalent to the generalized projection of the dual variables onto the j-th hyperplane (under the conditions given above).

Moreover we would like to note that the problems (3.15) and (3.24) are dual to each other (up to constants):

Lemma 3.3. *Let $\boldsymbol{\gamma}, \boldsymbol{\mu}, \boldsymbol{\varphi} \in \mathbb{R}^N$ be constant vectors, $\mathcal{G} : \mathbb{R}^N \to \mathbb{R}$ be a Bregman function, $\tilde{\mathrm{g}} = \nabla \mathcal{G}$, $\mathrm{G} : \mathbb{R}^N \to \mathbb{R}$ its conjugate and $\mathrm{g} = \nabla \mathrm{G} = \tilde{\mathrm{g}}^{-1}$. Then holds*

$$\begin{aligned} &\min_{\mathbf{d}} \quad \Delta_{\mathcal{G}}(\mathbf{d}, \mathbf{d}^{(0)}) \qquad \text{with } H^\top \mathbf{d} = -\boldsymbol{\gamma} \\ &= -\min_{\hat{\boldsymbol{\alpha}}} \quad \mathrm{G}(\tilde{\mathrm{g}}(\mathbf{d}^{(0)}) + H\hat{\boldsymbol{\alpha}}) - \underbrace{\mathrm{G}(\tilde{\mathrm{g}}(\mathbf{d}^{(0)}))}_{\text{const.}} + \langle \boldsymbol{\gamma}, \hat{\boldsymbol{\alpha}} \rangle, \end{aligned}$$

if both solutions exist.

The proof of a more general lemma can be found in Appendix A.1. A special case can be found in Kivinen and Warmuth [1999].

Since we chose $\mathbf{d}^{(0)} = \mathrm{g}(\mathbf{0})$ (cf. Algorithm 3.4 and Section 3.2), the objective simplifies: $\mathrm{G}(H\hat{\boldsymbol{\alpha}}) - \mathrm{G}(\mathbf{0}) + \langle \boldsymbol{\gamma}, \hat{\boldsymbol{\alpha}} \rangle$.

3.3.2 Convergence Theorems

We start with the following general convergence result, which seemed to be fallen into oblivion even in the optimization community. It will be very useful in the analysis of leveraging algorithms.

Theorem 3.2 (Convergence of Coordinate Descent, Luo and Tseng [1992]). *Suppose $\mathrm{G} : \mathbb{R}^N \to \mathbb{R}$ is twice continuously differentiable and strictly convex on $\operatorname{dom} \mathrm{G}$. Assume that $\operatorname{dom} \mathrm{G}$ is open, the set of solutions $\mathcal{S}^* \subset \mathbb{R}^J$ to*

$$\min_{\hat{\boldsymbol{\alpha}} \in \mathcal{S}} \ \mathrm{F}(\hat{\boldsymbol{\alpha}}), \quad \text{where} \quad \mathrm{F}(\hat{\boldsymbol{\alpha}}) = \mathrm{G}(H\hat{\boldsymbol{\alpha}}) + \langle \boldsymbol{\gamma}, \hat{\boldsymbol{\alpha}} \rangle \tag{3.27}$$

is not empty, where $H \in \mathbb{R}^{N \times J}$ is a fixed matrix having no zero column, $\boldsymbol{\gamma} \in \mathbb{R}^J$ fixed and $\mathcal{S} \subseteq \mathbb{R}^J$ is a (possibly unbounded) box-constrained *set. Furthermore assume that the Hessian $\nabla^2 \mathrm{G}(H\hat{\boldsymbol{\alpha}}^*)$ is a positive definite matrix for all $\hat{\boldsymbol{\alpha}}^* \in \mathcal{S}^*$. Let $\{\hat{\boldsymbol{\alpha}}^{(t)}\}$ be the sequence generated by coordinate descent, where the coordinate selection $j_1, j_2, \ldots$ satisfies the following β-optimality criterion:*

$$|\hat{\alpha}_{j_t}^{(t+1,j_t)} - \hat{\alpha}_{j_t}^{(t)}| \geq \beta \max_{j=1,\ldots,J} |\hat{\alpha}_j^{(t+1,j)} - \hat{\alpha}_j^{(t)}| \tag{3.28}$$

for some $\beta \in (0, 1]$, where $\hat{\alpha}_j^{(t+1,j)}$ is the optimal value of $\hat{\alpha}_j^{t+1}$ if it would be selected:[9]

$$\hat{\alpha}_j^{(t+1,j)} := \min_{\hat{\alpha} \in \mathcal{S}_j} \mathrm{G}\left(H\hat{\boldsymbol{\alpha}}^{(t)} + H_{*,j}(\hat{\alpha} - \hat{\alpha}_j^{(t)})\right) + \gamma_j \hat{\alpha}. \tag{3.29}$$

Then $\{\hat{\boldsymbol{\alpha}}^{(t)}\}$ converges to an element in $\mathcal{S}^*$.

9. For convenience, let the rest of the vector be unchanged, i.e. $\hat{\alpha}_{j'}^{(t+1,j)} = \hat{\alpha}_{j'}^{(t)}$, for all $j \neq j'$.

This theorem is slightly more general than we currently need.[10] In this section we consider only $\mathcal{S} = \mathbb{R}^J$ and $\boldsymbol{\gamma} = 0$.

The coordinate selection defined in the theorem is slightly different from the Gauss-Southwell selection rule described before: (i) it allows the selection scheme to be suboptimal. So one has some freedom in approximating the "best" coordinate selection. (ii) The condition is not formulated in terms of the gradient, but in terms of the step size to the optimum. If the function is strictly convex, then there is a relation between step size and gradient. We can prove the following:

Proposition 3.3 (GS scheme on $\mathbb{R}^J$, Rätsch et al. [2002]). *Assume the conditions on* G, F, H, $\boldsymbol{\gamma}$ *and* $\mathcal{S} = \mathbb{R}^J$ *as in Theorem 3.2. Let* $\mathcal{S}^b$ *be a convex subset of* $\mathcal{S}$. *Then a coordinate selection* $j_1, j_2, \ldots$ *starting at* $\hat{\boldsymbol{\alpha}}^{(0)}$ *satisfying*

1. $\left|\frac{\partial \mathrm{F}(\hat{\boldsymbol{\alpha}}^{(t)})}{\partial \hat{\alpha}_{j_t}}\right| \geq \delta \max_{j=1,\ldots,J} \left|\frac{\partial \mathrm{F}(\hat{\boldsymbol{\alpha}}^{(t)})}{\partial \hat{\alpha}_j}\right|$, *for some fixed* $\delta \in (0,1]$ $(t = 1, 2, \ldots)$,
2. $\hat{\boldsymbol{\alpha}}^{(t,j)} \in \mathcal{S}^b$ $(j = 1, \ldots, J,\ t = 0, 1, \ldots)$,
3. $\frac{\partial^2 \mathrm{G}(H\hat{\boldsymbol{\alpha}})}{\partial \hat{\alpha}_j^2} \leq \eta_u$, *and* $\frac{\partial^2 \mathrm{G}(H\hat{\boldsymbol{\alpha}})}{\partial \hat{\alpha}_j^2} \geq \eta_l$, *for some fixed* $\eta_l, \eta_u > 0$ $(\hat{\boldsymbol{\alpha}} \in \mathcal{S}^b,\ j = 1, \ldots, J)$

also satisfies condition (3.28) of Theorem 3.2.

The proof is shown in Appendix A.2.2 on page 140. Thus the *approximate Gauss-Southwell method on* $\mathbb{R}^J$ as defined above converges in the sense of Theorem 3.2.

To show the convergence of the second variant of LS-Boost (using the maximal improvement (MI) scheme, cf. (3.7)), we need the following:

Proposition 3.4 (MI scheme on $\mathbb{R}^J$, Rätsch et al. [2002]). *Assume the conditions on* G, F *and* H *as in Theorem 3.2 are satisfied. Let* $\mathcal{S} = \mathbb{R}^J$ *and* $\mathcal{S}^b$ *be a convex subset of* $\mathcal{S}$. *Then a coordinate selection* $j_1, j_2, \ldots$ *starting at* $\hat{\boldsymbol{\alpha}}^{(0)}$ *that satisfies*

1. $\mathrm{F}(\hat{\boldsymbol{\alpha}}^{(t)}) - \mathrm{F}(\hat{\boldsymbol{\alpha}}^{(t+1,j_t)}) \geq \delta \max_{j=1,\ldots,J} \left(\mathrm{F}(\hat{\boldsymbol{\alpha}}^{(t)}) - \mathrm{F}(\hat{\boldsymbol{\alpha}}^{(t+1,j)})\right)$ *for some fixed* $\delta \in (0,1]$ $(\forall t = 1, 2, \ldots)$,
2. *conditions 2–3 of Proposition 3.3,*

also satisfies (3.28) of Theorem 3.2.

The proof is shown in Appendix A.2.3 on page 140, where we show that the conditions on the sequence in Proposition 3.4 imply the conditions in Proposition 3.3. In fact, it also holds vice versa. We conclude that the *approximate maximal improvement scheme* as defined above converges in the sense of Theorem 3.2.

Finally we can also state a rate of convergence, which is surprisingly not worse than the rates for standard gradient descent methods:

Theorem 3.3 (Rate of Convergence, Luo and Tseng [1992]). *Under the conditions of Theorem 3.2 and the assumption that* F *is strongly convex (cf. footnote 13) along the path*

10. Luo and Tseng [1992] also prove the (linear) convergence for an almost cyclic selection scheme.

of the $\hat{\boldsymbol{\alpha}}^{(t)}$'s, we have

$$\epsilon_{t+1} := \mathrm{F}(\hat{\boldsymbol{\alpha}}^{(t+1)}) - \mathrm{F}(\hat{\boldsymbol{\alpha}}^*) \leq \left(1 - \frac{1}{\eta}\right) (\mathrm{F}(\hat{\boldsymbol{\alpha}}^{(t)}) - \mathrm{F}(\hat{\boldsymbol{\alpha}}^*)), \tag{3.30}$$

where $\hat{\boldsymbol{\alpha}}^t$ is the estimate after the t-th coordinate descent step, $\hat{\boldsymbol{\alpha}}^*$ denotes a optimal solution, and $1 < \eta < \infty$. In particular at iteration t holds:

$$\epsilon_t \leq \left(1 - \frac{1}{\eta}\right)^t \epsilon_0.$$

Following Luo and Tseng [1992] one can show that the constant η is $\mathcal{O}\left(\frac{\kappa^2 \rho \sigma J^4 N^2}{\beta^2}\right)$. Here, ρ is the Lipschitz constant of $\nabla \mathrm{G}$, σ is a lower bound on the eigenvalues of $\nabla^2 \mathrm{G}(\hat{\boldsymbol{\alpha}}^{(t)})$, $t = 1, \ldots$ and κ is a constant that only depends on H [for details see Luo and Tseng, 1992, Hoffmann, 1952, Robinson, 1973, Mangasarian and Shiau, 1987]. While the upper bound on η can be rather large – making the shown convergence slow – it is important to note (i) that this is only a rough estimate of the true constant and (ii) still guarantees an *exponential decrease* in the functional with the number of iterations.

A special case of (3.27) has already been considered in a paper that predates the formulation of AdaBoost Cesa-Bianchi et al. [1994b]. This optimization problem concerns the likelihood maximization for some exponential family of distributions. In this work convergence is proven for the general non-separable case including lower and upper bounds on the constant η; however, only for the exponential loss, i.e. for the case of AdaBoost.[11] The framework set up in the current work is more general and we are able to treat any strictly convex loss function.

3.3.3 Summarizing Remarks

We worked out two condition on the selection scheme, for which coordinate descent converges *linearly* to the optimal solution. These conditions are slightly more strict but similar in spirit to the ones derived in Section 3.2 (cf. Proposition 3.1 and Proposition 3.2). In this section the selection scheme needs to satisfy that each selected coordinate leads to an improvement comparable to the coordinate with *maximum improvement*. In the last section it was sufficient that selected coordinate leads to an improvement comparable to the average improvement over all coordinates. It can easily be verified that these conditions are equivalent, if the function τ used to formulate the conditions in Section 3.2 is a linear function. We may conclude that if the function τ is linear or sub-linear, the Bregman algorithm converges linearly. If it is super-linear, then we only know that it converges asymptotically.

11. We will expand on this connection in future work.

3.4 Application to Leveraging

We now return from the abstract convergence results of the previous sections to leveraging algorithms. We show how to retrieve the (approximate) Gauss-Southwell algorithm on $\mathbb{R}^J$ as a part of Algorithm 3.4. The gradient of $\mathrm{F}(\hat{\boldsymbol{\alpha}}) \equiv \mathrm{G}(H\hat{\boldsymbol{\alpha}}) + \langle \boldsymbol{\gamma}, \hat{\boldsymbol{\alpha}} \rangle$ with respect to $\hat{\alpha}_j$ is given by

$$\begin{aligned}\frac{\partial \mathrm{F}(\hat{\boldsymbol{\alpha}})}{\partial \hat{\alpha}_j} &= \gamma_j + \sum_{n=1}^{N} \frac{\partial}{\partial \hat{\alpha}_j} \mathrm{g}(y_n, f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}_n)) \\ &= \gamma_j + \sum_{n=1}^{N} \mathrm{g}'(y_n, f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}_n)) \frac{\partial}{\partial \hat{\alpha}_j} f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}_n) \\ &= \gamma_j + \sum_{n=1}^{N} \mathrm{g}'(y_n, f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}_n)) \hat{h}_j(\boldsymbol{x}_n) = \gamma_j + \sum_{n=1}^{N} d_n \hat{h}_j(\boldsymbol{x}_n) \end{aligned} \tag{3.31}$$

where d_n is given as in step 3c of Algorithm 3.4. Let us assume $\boldsymbol{\gamma} = \mathbf{0}$. Then, the coordinate with maximal absolute gradient corresponds to the hypothesis with largest absolute edge (see definition on page 41). However, since we assume the hypothesis space is closed under complementation and according to Proposition 3.1 and Proposition 3.2, we need to assume less on the base learner to ensure the convergence: it either has to return a hypothesis that (approximately) maximizes the edge, or alternatively (approximately) minimizes the loss function. This leads to the following definition:

Definition 3.4 (τ-Optimality). *Let H be a finite, complementation closed hypothesis set, G be a functional as defined above and $\tau : \mathbb{R}_+ \to \mathbb{R}_+$ be a strictly monotonically increasing and continuous function with $\tau(0) = 0$. A base learning algorithm $\mathcal{L}$ is called* τ-optimal *for H, G and $\boldsymbol{\gamma}$, if it always returns a hypothesis $\hat{h}_j \in \mathsf{H}$ that either satisfies*

1. $\hat{\gamma}_j - \gamma_j \geq \tau\left(\max_{j'=1,\ldots,J}(\hat{\gamma}_{j'} - \gamma_{j'})\right)$, *where* $\hat{\gamma}_j = \sum_{n=1}^{N} d_n \hat{h}_j(\boldsymbol{x}_n)$ *for any weighting* $\mathbf{d} \in \mathrm{range}(\nabla \mathrm{G})$,
   or
2. $\mathrm{F}(\hat{\boldsymbol{\alpha}}^{(t)}) - \mathrm{F}(\hat{\boldsymbol{\alpha}}^{(t+1,j)}) \geq \tau\left(\max_{j'=1,\ldots,J}\left(\mathrm{F}(\hat{\boldsymbol{\alpha}}^{(t)}) - \mathrm{F}(\hat{\boldsymbol{\alpha}}^{(t+1,j')})\right)\right)$ *for any*[12] $\hat{\boldsymbol{\alpha}}^{(t)} \in \mathbb{R}_+^J$,

   and never returns a hypothesis with $\hat{h}(\boldsymbol{x}_n) = 0$ for all $n = 1, \ldots, N$.

By Proposition 3.3 and Proposition 3.4 we can motivate the following:

Definition 3.5 (δ-Optimality, Rätsch et al. [2002]). *A base learning algorithm $\mathcal{L}$ is called δ-optimal, if it is τ optimal and $\tau(x) = \delta x$, for some fixed $\delta \in (0, 1]$.*

Since we have assumed H is closed under complementation, there always exist two hypotheses having the same absolute gradient ($\hat{h}$ and $-\hat{h}$). To fulfill (3.23) of Proposition 3.2 or condition 1 of Proposition 3.4, we therefore only need to consider the hypothesis with

12. Here we only require $\hat{\boldsymbol{\alpha}}^{(t)} \geq 0$ since H is closed under complementation.

maximum edge as opposed to the *maximum absolute edge*. For classification this means: if the base learner returns the hypothesis with (approximately) smallest weighted training error, this condition is satisfied.

The next two theorems show that Definition 3.4 and Definition 3.5 state sufficient conditions on the base learner to achieve the asymptotical and linear convergence, respectively. For the rest of the chapter we will assume $\boldsymbol{\gamma} = \mathbf{0}$. In the next chapter we will use $\boldsymbol{\gamma} \neq \mathbf{0}$ to implement regularized loss functions.

Theorem 3.4. *Given a training sample S and a finite and complementation closed hypothesis set H. Let $\mathrm{G} : \mathbb{R}^N \to \mathbb{R}$ be the conjugate of a Bregman function. Suppose Algorithm 3.4 generates a sequence of combined hypotheses $f_{\hat{\boldsymbol{\alpha}}^{(0)}}, f_{\hat{\boldsymbol{\alpha}}^{(1)}}, \ldots$ (cf. (3.9)) using a τ-optimal learning algorithm $\mathcal{L}$ for H and G. Then any limit point of $\{\hat{\boldsymbol{\alpha}}^{(t)}\}$ is a solution of (3.8).*

The proof directly follows by an application of Proposition 3.1 and Proposition 3.2, respectively. From the results in Section 3.3 we have a stronger result, but also with slightly stronger assumptions:

Theorem 3.5 (Rätsch et al. [2002]). *Given a training sample S and a finite and complementation closed hypothesis set H. Let $\mathrm{G} : \mathbb{R}^N \to \mathbb{R}$ be twice differentiable, strongly convex*[13] *and its Hessian being uniformly upper bounded on any bounded subset of $\mathbb{R}^N$. Suppose Algorithm 3.4 generates a sequence of combined hypotheses $f_{\hat{\boldsymbol{\alpha}}^{(0)}}, f_{\hat{\boldsymbol{\alpha}}^{(1)}}, \ldots$ (cf. (3.9)) using a δ-optimal learning algorithm $\mathcal{L}$ for H and G. If any solution of (3.8) is finite and $\limsup_{t\to\infty} \|\hat{\boldsymbol{\alpha}}^{(t)}\| < \infty$, then any limit point of $\{\hat{\boldsymbol{\alpha}}^{(t)}\}$ is a solution of (3.8) and the sequence converges* linearly *in the sense of Theorem 3.3.*

The proof is shown in Appendix A.2.4 on page 141. Finally we can apply these theorems to the special cases described in Section 3.1:

Corollary 3.1 (Rätsch et al. [2002]). *Given a training sample S and a finite hypothesis set H. Suppose AdaBoost, the Logistic Regression algorithm or LS-Boost as described in Section 3.1 (cf. Algorithm 3.4) generates a sequence of hypotheses $h_1, h_2, \ldots$ and weights $\alpha_1, \alpha_2, \ldots$ using a δ-optimal base learner. Assume every solution of (3.8) using the respective loss function is finite. Then any limit point of $\{f_t\}$ solves (3.8).*

The proof is shown in Appendix A.2.5 on page 142. For the selection scheme of LS-Boost (3.5) both conditions in Definition 3.4 *cannot* be satisfied in general, unless $\sum_{n=1}^N \hat{h}_j(\boldsymbol{x}_n)^2$ is constant for all hypotheses $\hat{h}_j$, $j = 1, \ldots, J$. Since $\sum_{n=1}^N (d_n - \hat{h}_j(\boldsymbol{x}_n))^2 = \sum_{n=1}^N \hat{h}_j(\boldsymbol{x}_n)^2 - 2d_n\hat{h}_j(\boldsymbol{x}_n) + \text{const.}$, the base learner prefers hypotheses with small $\sum_{n=1}^N \hat{h}_j(\boldsymbol{x}_n)^2$ and could therefore stop improving the objective while being not optimal (see [Rätsch et al., 2002, Section 4.3] and Section 5.3 for more details).

Note that Corollary 3.1 also applies to the confidence rated boosting algorithm, if the base learner satisfies condition 2 of Definition 3.4 and the hypothesis set is finite.

13. A function is called strongly convex on $\mathbb{S}$, if there exists a fixed $\eta > 0$ such that the matrix $\nabla^2_{\mathbf{o}} \mathrm{G}(\mathbf{o}) - \eta\mathbf{I}$ is positive semidefinite for all $\mathbf{o} \in \mathbb{S}$ (cf. [Bertsekas, 1995, p. 563]). Here, we require strong convexity on any bounded subset or $\mathbb{R}^N$.

3.5 Discussion and Summary

We gave an unifying convergence analysis for a fairly general family of leveraging methods. In the first part we examined sufficient conditions on the selection scheme for asymptotical convergence. Roughly speaking, it is sufficient if one selects the coordinates such that one progresses only slightly as long as one has not reached the optimum. By assuming somewhat stronger conditions on the selection scheme, we could show that the convergence is linear.

We applied these results to leveraging algorithms. Using natural assumptions on the base learning algorithm that are in spirit similar to the assumptions in the last chapter, we have shown that leveraging converges. If the base learner satisfies the δ-optimality criterion, then leveraging converges even linearly.

While the main theorem for the convergence of coordinate descent was already proven in Luo and Tseng [1992], its application to leveraging closes a central gap between existing algorithms and their theoretical understanding in terms of convergence.

Let us come back to the AdaBoost: it has been shown that the loss drops exponentially fast, if the base learner is able to achieve a training error consistently smaller than $\frac{1}{2}$ (cf. Section 1.2 and Chapter 2). This is desirable for the analysis in the PAC framework (cf. Section 1.3.1). However, if the data is not separable, then this analysis does not apply and the speed of convergence is unknown. In this chapter we have shown that one can prove the same type of convergence without assuming the separability of the data. This results is slightly surprising and also relevant in practice, where the data is usually not separable.[14]

In Rätsch et al. [2002] we have extended our analysis to ℓ_1-norm regularized cost functions. These results can readily be used to derive regularized leveraging algorithms in a clean and general way. This will be done in Chapter 4 for classification and in Chapter 5 for regression tasks. Future investigations include the generalization to infinite hypotheses spaces (see also Chapter 5 and Zhang [2002]) and an improvement of the convergence rates. We conjecture that our results can be extended to many other variants of boosting type algorithms proposed recently in the literature (cf. `http://www.boosting.org`).

14. However, we have to note here, that for the analysis in the PAC setting the linear convergence in general is not sufficient. Here one needs that the constant η does not depend on the sample size. In the case of AdaBoost the loss function is decreased by a factor that only depends on the achievable margin of the problem at hand. But this requires that the data is separable.

4 Soft Margins

It has been shown that AdaBoost rarely overfits in the low noise regime, however, we show here that it clearly does so for higher noise levels. In this chapter, we develop techniques that yield state-of-the-art results on noisy problems.

Central to the understanding of this fact is the margin distribution. We have shown in Chapter 2 that AdaBoost asymptotically achieves a *hard margin* separation, i.e. the algorithm concentrates its resources on a few hard-to-learn examples that are very similar to Support Vectors. A hard margin is clearly a sub-optimal strategy in the noisy case, and regularization must be introduced in the algorithm to alleviate the distortions that single difficult examples (e.g. outliers) can cause to the decision boundary. This will be discussed in detail in Section 4.1.

In Chapter 3 we considered leveraging algorithms similar to AdaBoost that aim to find a combined hypothesis minimizing some loss functional. Although some of these algorithms are more robust than AdaBoost, there is always the chance to overfit the data, since the number of hypotheses that could be potentially combined is very large or even infinite. Here, the original method of controlling the capacity by achieving the largest margin is not effective.

We therefore propose two regularization methods and generalizations of the original AdaBoost algorithm to achieve a *soft margin*. In particular we suggest (1) $\text{AdaBoost}_{\text{Reg}}$, where the coordinate decent is done in a modified loss function and (2) a regularized linear programming problem, where the soft margin is attained by introducing *slack variables*. To solve the resulting linear optimization problem we propose two algorithms: ν-Arc and a barrier algorithm. For the latter we can show the convergence.

The barrier algorithm is in fact very similar to logistic regression (cf. Section 3.1), but is employing some ℓ_1-norm regularization on the hypothesis coefficients to control the size of the hypothesis space. As an important auxiliary result, we show that any leveraging algorithm can be regularized without losing the convergence properties shown in Chapter 3, eventually leading to a more robust algorithm.

Finally, in Section 4.4 numerical experiments on several benchmark data sets show the validity and competitiveness of our regularized algorithms. Furthermore, we discuss an application in a *real-world* setting – a non-intrusive power monitoring system.

4.1 Hard margins and overfitting

In this section, we give reasons why the AdaBoost and most of its variants are *not noise robust* and exhibit suboptimal generalization ability in the presence of noise. According to our understanding, noisy data has at least one of the following properties:

1. overlapping class probability distributions,
2. outliers and
3. mislabeled examples.

All three types of noise appear very often simultaneously in data analysis. Therefore, the development of noise robust versions of AdaBoost is mandatory.

We have shown in Chapter 2 that AdaBoost, AdaBoost$_\gamma$, Arc-GV and also Marginal AdaBoost aim to find a hypothesis that is consistent with the training data within a few iterations. For other leveraging methods like Logistic regression as presented in Section 3.1, similar results are available [e.g. Duffy and Helmbold, 2000b]. This property is desirable for the analysis in the PAC setting as presented in Section 1.2.2, where we assumed that the data is separable.[1] In this case one can arbitrarily close approximate the target concept (if one has enough training examples available). If the data has any of the three properties above, this assumption is not satisfied. Then the labels are generated according to some probability distribution and not necessarily computed by a deterministic rule (e.g. the target concept).

Let us therefore come back to the analysis of AdaBoost based on margin distributions as mentioned in Section 1.3.1. In Schapire et al. [1998] it has been proven that with probability at least $1-\delta$ over the random draw of a training set S of size N the generalization error of a function with margins $\rho_1, \ldots, \rho_N$ can be bounded by

$$R[f] \leq \frac{1}{N}\sum_{n=1}^{N} \mathbf{I}(\rho_n \leq \theta) + \mathcal{O}\left(\sqrt{\frac{\vartheta \log^2(N/\vartheta)}{N\theta^2} + \frac{\log(1/\delta)}{N}}\right), \tag{4.1}$$

where $\theta \in (0,1]$ and ϑ is the VC dimension of the base hypothesis space. It was stated that a reason for the success of AdaBoost, compared to other ensemble learning methods (e.g. Bagging Breiman [1996]), is that it generates combined hypotheses with large margins on the training examples. It asymptotically finds a linear combination $f_{\hat{\boldsymbol{\alpha}}}$ of base hypotheses satisfying

$$\rho_n(\hat{\boldsymbol{\alpha}}) = y_n \frac{f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}_n)}{\sum_j \hat{\alpha}_j} \geq \rho \quad n = 1, \ldots, N, \tag{4.2}$$

for some large margin ρ. Then the first term of (4.1) can be made zero for $\theta = \rho$ and the second term becomes small, if ρ is large. In Breiman [1999], Grove and Schuurmans [1998] and in this work (cf. Chapter 2), algorithms have been proposed that generate combined hypotheses with even larger margins than AdaBoost. In Section 2.4.3, we have shown

1. In *agnostic PAC learning* this assumption is not necessary.

that as the margin increases, the generalization performance can become better on data sets with almost no noise [see also Onoda et al., 1998]. However, on problems with a large amount of noise, it has been found that the generalization ability usually becomes worse for hypotheses with larger margins [see also Quinlan, 1996]. As an example for overlapping classes, Figure 4.1 [left] shows a typical overfitting behavior of AdaBoost on the `banana` data set [Rätsch et al., 2001]. Here, already after only 80 boosting iterations the best generalization performance is achieved.

We will call an algorithm, (approximately) maximizing the *smallest margin* on the training set, a *hard margin algorithm*. That is if the resulting classification function satisfies (4.2) for some $\rho \gg 0$. It classifies *all* training examples according to their possibly wrong labels (cf. Figure 4.1 [right]). This is at the expense that the margin of other examples are reduced, the complexity of the combined hypotheses increases and the decision boundary becomes e.g. less smooth. In this case the achieved decision boundary is far away from the Bayes optimal boundary (cf. dashed line in Figure 4.1 [right]) [see Rätsch, 1998].

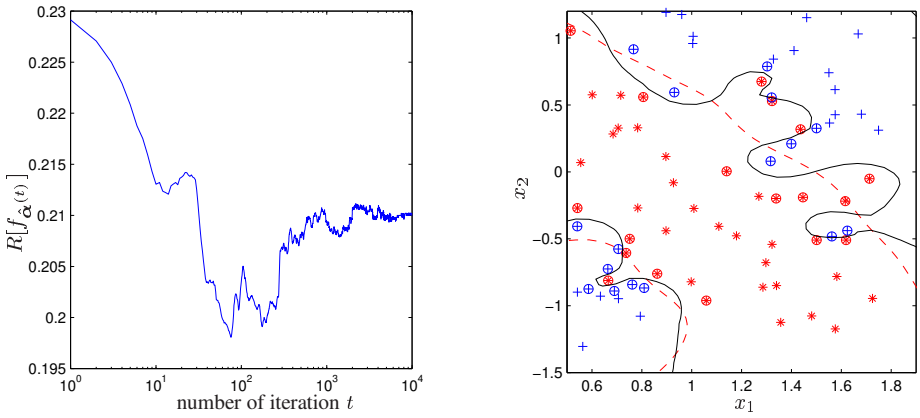

Figure 4.1 Typical overfitting behavior: the generalization error (smoothed) as a function of the number of iterations [left] and a typical decision boundary [right] generated by AdaBoost (10^4 iterations) using RBF networks with 30 centers (cf. Appendix B.6.1 and Müller et al. [1999]) in the case of noisy data (`banana` data set with 300 examples, $\sigma^2 = 16\%$). The positive and negative training examples are shown as '$+$' and '$*$' respectively, the support vectors (cf. Section 2.3.2) are marked with '$\circ$'. An approximation to the Bayes decision boundary is plotted dashed [cf. Rätsch, 1998]. (Figure taken from Rätsch et al. [2001].)

To discuss the suboptimal performance of hard margin classifiers in the presence of outliers and mislabeled examples in a more abstract way, we analyze Figure 4.2. Let us first consider the case without noise [left]. Here, we can estimate the separating hyperplane correctly. In Figure 4.2 [middle] we have one outlier, which corrupts the estimation. Hard margin algorithms will concentrate on this outlier and spoil the good estimate that we would get without the outlier. Next, let us consider more complex decision boundaries. Here the overfitting problem gets even more distinct, if we can generate more and more complex functions by combining many hypotheses. Then all training examples (even mislabeled ones or outliers) can be classified correctly. In Figure 4.1 [right] and Figure 4.2 [right] we see that the decision boundary is rather rough and can result in bad generalization.

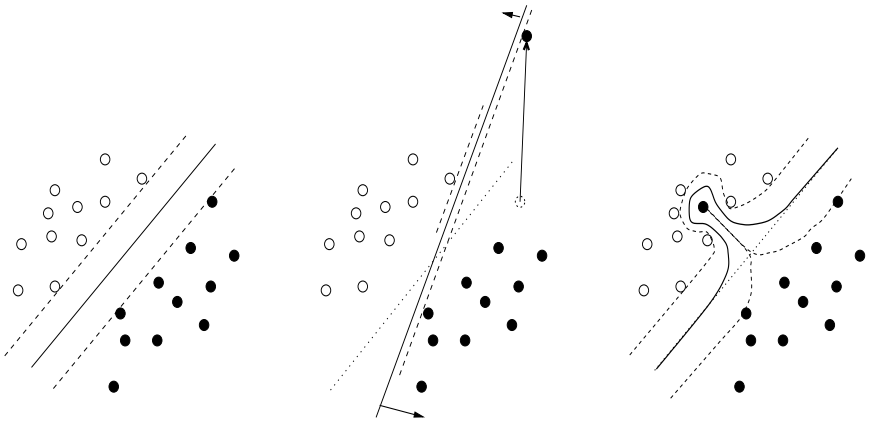

Figure 4.2 The problem of finding a maximum margin "hyperplane" on reliable data [left], data with outlier [middle] and with a mislabeled example [right]. The solid line shows the resulting decision boundary, whereas the dashed line marks the margin area. In the middle and on the left the original decision boundary is plotted with dots. The hard margin implies noise sensitivity, because only one example can spoil the whole estimation of the decision boundary. (Figure taken from Rätsch et al. [2001].)

From these cartoons, it becomes apparent that AdaBoost and any other algorithm with large hard margin is noise sensitive and maximizing the smallest margin in the case of noisy data can (and will) lead to suboptimal results. Therefore, we need to relax the hard margin and allow for a possibility of "mistrusting" the data.[2]

From the bound (4.1) it is indeed not clear that one should maximize the smallest margin: the first term on the right hand side of (4.1) takes the whole margin distribution into account. If we would allow a non-zero training error in the settings of Figure 4.2, then the first term of the right hand side of (4.1) becomes non-zero. But then θ can be considerably larger, such that the second term is much smaller. So we expect that a soft margin algorithm is able to achieve smaller upper bounds on the error than the hard margin algorithm, if the data is noisy.

Recently, the concept of *algorithmic stability* has been proposed. Since it fits well to the discussion above, we give a brief (informal) summary of the work presented in Bousquet and Elisseeff [2001a]. It has been shown that an algorithm only depending *weakly* on each example will generalize well. A learning algorithm $\mathfrak{L}$ is said to be β-*classification stable* [Bousquet and Elisseeff, 2001b], if for any set of examples $S \in \mathbb{Z}^N$, the function $f = \mathfrak{L}(S)$ is uniformly stable, if one removes any example from the training set, i.e.

$$\forall S \in \mathbb{Z}^N,\ \forall n \in \{1,\dots,N\},\ \sup_{\boldsymbol{x}\in\mathbb{X}} |\mathfrak{L}(S)(\boldsymbol{x}) - \mathfrak{L}(S^{\setminus n})(\boldsymbol{x})| \leq \beta, \tag{4.3}$$

where $S^{\setminus n}$ is the training set S without the example n. For any β-classification stable learning algorithm $\mathfrak{L}$ and any fixed γ can be shown [Bousquet and Elisseeff, 2001b, Theorem 17]: with probability $1-\delta$, $\delta \in (0,1)$, over the random draw of the training

2. Also the original SVM algorithm [Boser et al., 1992] assumed separable classes and pursued a hard margin strategy and it had similarly poor generalization performance on noisy data as AdaBoost. Only the introduction of soft margins for SVM [Cortes and Vapnik, 1995] allowed them to achieve much better generalization results.

sample S holds

$$R[f] \leq R^{\gamma}_{\text{emp}}[f] + \frac{2\beta}{\gamma} + \frac{4N\beta}{\gamma}\sqrt{\frac{\log 1/\delta}{2N}}, \tag{4.4}$$

where $f = \mathcal{L}(S)$ and $R^{\gamma}_{\text{emp}}[f]$ is the margin loss, i.e. $R^{\gamma}_{\text{emp}}[f] = \frac{1}{N}\sum_{n=1}^{N} \mathbf{I}(y_n f(\boldsymbol{x}_n) \leq \gamma)$.

Thus, if the algorithm at hand is not sensitive to a single example (i.e. β is small), the algorithm will generalize well. As discussed, hard margin algorithms can be very sensitive to a few examples (cf. Figure 4.2). This might be one reason why they can fail on noisy data [see also the discussion in Grandvalet, 2001].

In the following section we introduce several possibilities to "mistrust" parts of the data, which leads to a general *soft margin* concept.[3] Using these techniques we will derive algorithms that implement the ideas discussed above. We will show that these algorithm are much less noise sensitive than the hard margin algorithms and have better generalization abilities on noisy data.

4.2 Reducing the Influence of Hard Examples

First, we propose an improvement of original AdaBoost by using a regularization term that is originally based on the *intuition* that it has to be avoided that AdaBoost concentrates too much on the most difficult examples [Rätsch, 1998].

4.2.1 Trade-off Between Margin and Influence

We claim that any modification that improves AdaBoost on noisy data, must not force *all* margins beyond 0 (even if it is possible). In particular examples that are mislabeled and usually more difficult to classify have to have margins smaller than 0, i.e. violate (4.2). If we knew beforehand which examples are unreliable, we would simply remove them from the training set or, alternatively, we would not require that they have a large margin. Suppose we have defined a non-negative quantity ζ_n, which expresses our "mistrust" in an example $(\boldsymbol{x}_n, y_n)$. For instance, this could be a probability that the label of an example is incorrect.[4] Then we may relax (4.2) leading to

$$\rho_n(\hat{\boldsymbol{\alpha}}) \geq \rho - C\zeta_n, \quad n = 1, \ldots, N \tag{4.5}$$

where C is an *a priori* chosen constant. Furthermore, we can define the *soft margin* $\tilde{\rho}_n$ of an example $(\boldsymbol{x}_n, y_n)$ as a tradeoff between the margin and ζ_n

$$\tilde{\rho}_n(\hat{\boldsymbol{\alpha}}) := \rho_n(\hat{\boldsymbol{\alpha}}) + C\zeta_n \tag{4.6}$$

3. In Mason et al. [1998, 2000] a similarly motivated approach has been taken. They proposed a bound, which is then used to optimize the margin distribution directly.

4. However, we will not give a probabilistic interpretation and any reasonable quantity could be used. See discussion below.

and from (4.5) we obtain

$$\tilde{\rho}_n(\hat{\boldsymbol{\alpha}}) \geq \rho. \tag{4.7}$$

Now we can again easily find a solution with large *soft margin* on all examples, i.e. maximize ρ, and we expect to observe less overfitting.

The problem of course is how one should define ζ_n. We restrict ourselves to present only one definition of ζ based on the *influence* of examples on the combined hypotheses h_r

$$\mu_n^{(t)} = \sum_{r=1}^{t} \frac{\hat{\alpha}_r}{\sum_{r'=1}^{t} \hat{\alpha}_{r'}} d_n^{(r)},$$

which is the average weight of an example computed during the learning process. The rationale is: an example that is very often misclassified (i.e. hard to classify correctly) will have a high average weight, i.e. a high influence. We observed that in the noisy case there is (usually) a high overlap between examples with high influence and examples that are mislabeled or other examples very near to or only slightly beyond the decision boundary. Therefore, it makes sense to mistrust examples with high influences in the noisy case. Thus, we define ζ by

$$\zeta_n := \mu_n^{(t)}. \tag{4.8}$$

If a training example has a high influence ζ_n, then also the margin is virtually increased. If one maximizes the smallest soft margin, one does not force outliers to be classified according to their possibly wrong labels, but allows for some errors. Note that (4.8) implies a prior to weight all examples equally. This counterbalances the tendency of AdaBoost-like algorithms to overweight certain examples: we tradeoff between margin and influence.

Of course, other functional forms of ζ are possible and seem to be reasonable. In Rätsch et al. [2001] we also proposed $\zeta_n = (\mu_n^{(t)})^2$, which can have advantages in some cases. In the Section 4.3.2 we propose another choice motivated from the soft margin approach [Cortes and Vapnik, 1995] used in SVMs [see also Rätsch et al., 2000a]. Another idea, for instance, is to set $\zeta_n^{(t)} = \mathbf{P}[f_{\hat{\boldsymbol{\alpha}}^{(t)}}(\boldsymbol{x}_n)]$, where $\mathbf{P}$ is some *regularization operator*. Via $\mathbf{P}$ it is possible to incorporate prior knowledge about the problem at hand, like smoothness of the decision boundary much in the spirit of Tikhonov regularizers [e.g. Tikhonov and Arsenin, 1977, Smola et al., 1998b, Rokui and Shimodaira, 1998], or invariance regularization e.g. for OCR problems [e.g. Schölkopf, 1997, Mika et al., 2000a].

4.2.2 AdaBoost$_{\text{Reg}}$

Using the soft margin instead of the actual margin, one may reformulate any of the hard margin algorithms considered so far to develop a *soft margin algorithm*. To exemplify this translation procedure, we use AdaBoost as a basis. It could also be done with Arc-GV, Marginal Boosting and Logistic Regression.

As discussed in Chapter 2 and Chapter 3, AdaBoost minimizes the loss function

$$\mathrm{G}[f_{\hat{\alpha}}] = \sum_{n=1}^{N} \exp\left\{-\rho_n(\hat{\boldsymbol{\alpha}}) \sum_j \hat{\alpha}_j\right\}, \tag{4.9}$$

where $\rho_n(\hat{\boldsymbol{\alpha}})$ is as defined in (1.11). We have shown that the combined hypothesis generated by AdaBoost after many iterations is related to the solution of the margin-LP problem (2.3). Thus, AdaBoost and also the other algorithms can be understood as a general machine for (approximately) solving large, possibly non-linear min-max problems of the type:

$$\begin{array}{lrl} \max & \tilde{\rho} & \\ \text{with} & \tilde{\rho}_n(\hat{\boldsymbol{\alpha}}) \geq \tilde{\rho} & \text{for all } n = 1, \ldots, N \\ & \hat{\alpha}_j, \tilde{\rho} \geq 0 & \text{for all } j = 1, \ldots, J \\ & \sum_{j=1}^{J} \hat{\alpha}_j = 1, & \end{array} \tag{4.10}$$

where $\tilde{\rho}_n(\hat{\boldsymbol{\alpha}})$ can be a quite general function e.g. nonlinearly depending on $f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}_n)$. Note that in (2.3) we have $\tilde{\rho}_n(\hat{\boldsymbol{\alpha}}) = \rho_n(\hat{\boldsymbol{\alpha}})$ and $\tilde{\rho}_n$ is therefore a linear function in $\hat{\boldsymbol{\alpha}}$. Although it might be difficult or even NP hard to find a global solution of (4.10) in the general case, we have experienced that if the function is smooth, one often finds "good" local minima [e.g. Kohlmorgen et al., 2000].

The strategy of e.g. AdaBoost is to minimize the loss function (4.9). We propose to modify AdaBoost's cost function (4.9), where the actual margin ρ is replaced by the soft margin $\tilde{\rho}$, in particular, we set $\tilde{\rho}$ as in (4.7) and (4.8). By minimizing this modified cost function one can argue that one approximately solves (4.10). The same technique can be used for other soft margin definitions and our proposition is to be seen as a *recipe* to develop new soft margin algorithms based on a boosting-like algorithm [see also Friedman, 1999, Mason et al., 2000]. In fact, in the next section we will propose another algorithm that is closely related to the soft margins used in SVMs [cf. Cortes and Vapnik, 1995].

By this argumentation, we arrive at an algorithm with a new loss function by plugging-in our choice of ζ_n as in (4.8):

$$\begin{aligned} \mathrm{G}_{\mathrm{Reg}}[f_{\hat{\boldsymbol{\alpha}}}] &= \sum_{n=1}^{N} \exp\left\{-\tilde{\rho}_n(\hat{\boldsymbol{\alpha}}) \sum_j \hat{\alpha}_j\right\} \\ &= \sum_{n=1}^{N} \exp\left\{-\left[\rho_n(\hat{\boldsymbol{\alpha}}) + C\mu_n\right] \sum_j \hat{\alpha}_j\right\}. \end{aligned} \tag{4.11}$$

Having defined the cost function, we can use the techniques proposed in Chapter 3 (cf. Algorithm 3.4) to derive an iterative algorithm to reduce and ideally minimize $\mathrm{G}_{\mathrm{Reg}}[f_{\hat{\boldsymbol{\alpha}}}]$ [see also Mason et al., 2000, Rätsch et al., 2001, 2000a]. In particular, one can derive the example weighting $\mathbf{d}^{(t)}$ and how one should choose the hypothesis coefficient α_t. The pseudo-code of the resulting algorithm is given in Algorithm 4.5.

Algorithm 4.5 The AdaBoost$_{\text{Reg}}$ algorithm [Rätsch et al., 2001]

1. **Input:** N examples $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, Number of iterations T, Regularization constant C
2. **Initialize:** $d_n^{(1)} = 1/N$ for all $n = 1, \ldots, N$
3. **Do for** $t = 1, \ldots, T$,
   (a) Train classifier with respect to the weighted sample set $\{S, \mathbf{d}^{(t)}\}$ and obtain hypothesis $h_t : \boldsymbol{x} \mapsto [-1, 1]$
   (b) Find the hypothesis coefficient by solving
   $$\alpha_t = \operatorname*{argmin}_{\alpha_t \geq 0} \sum_{n=1}^{N} \exp\left\{ -\left[\rho_n(\boldsymbol{\alpha}^{(t)}) + C\mu_n^{(t)}\right] \sum_{r=1}^{t} \hat{\alpha}_t \right\}.$$
   (c) Update weights $d_n^{(t+1)} = \frac{1}{Z_t} \exp\left\{ -\left[\rho_n(\boldsymbol{\alpha}^{(t)}) + C\mu_n^{(t)}\right] \sum_{r=1}^{t} \alpha_t \right\}$, where Z_t is such that $\sum_{n=1}^{N} d_n^{(t+1)} = 1$.
4. **Break if** $\alpha_t \leq 0$
5. **Output:** Final hypothesis with weights $\boldsymbol{\alpha}^{(t)}$ as in Algorithm 1.1

From an optimization point of view, however, our choice of ζ as the influence of the example is difficult to analyze, since one essentially uses the dual variables $\mathbf{d}$ in the primal domain.[5] We conjecture that the optimization problem is rather hard to solve exactly. We can therefore not give a convergence result of this algorithm. However, we conjecture that the analysis of boosting based on the techniques presented in Section 2.5 can be extended to solving (4.10) (in particular for other choices of ζ; cf. discussion above). If the feasible set is convex, then it appears possible to prove that the algorithm converges to a minimum, as long as the sum of the hypothesis coefficients goes to infinity (cf. Section 2.5) [see also Mason et al., 2000], but this goes beyond the scope of this thesis.

4.2.3 Experimental Illustration

In Figure 4.3 we illustrate AdaBoost$_{\text{Reg}}$ on a two dimensional toy example. First, we run AdaBoost on the data without noise [left]: the decision boundary is smooth and supported only by a few examples. There are about five examples in each class that have non-negligible influence (diameter of the points are proportional to the influence of the examples). When adding three additional examples, the decision boundary changes significantly. In particular the three added examples (which are outliers or mislabeled and therefore hard to classify) have a high influence on the boundary – shared with about five other examples in each class. Removing any of the three examples would change the bound-

5. Motivated by Jaakkola and Haussler [1999], a similar approach for SVMs was taken in Weston [1999], Herbrich and Weston [1999]. They proposed a linear program that minimizes an estimate of the leave-one-out error for SVMs. Seen in this context, AdaBoost$_{\text{Reg}}$ might be understood as an algorithm that approximately minimizes its own leave-one-out error. This is an interesting property, which might partially explain its good performance. This topic will be subject to further research.

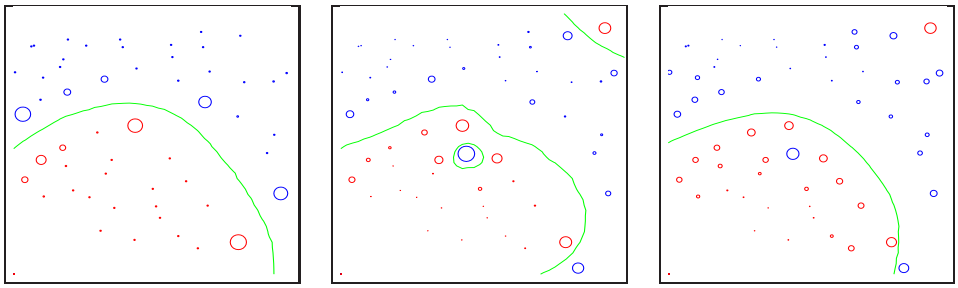

Figure 4.3 Illustration of AdaBoost$_{\text{Reg}}$: The decision boundary of AdaBoost on a toy data set (red: positive class, blue: negative class) without [left] and with noise [middle]. The boundary is considerably changed when adding only three examples (middle, upper/lower right). On the right the boundary generated by AdaBoost$_{\text{Reg}}$ is plotted. The boundary is almost unchanged compared to the case without noise. The diameter of the points is proportional to the influence of the example. When using AdaBoost$_{\text{Reg}}$, the influences are spread over more examples near the boundary.

ary considerably. In Figure 4.3 [right] the boundary and the influences of AdaBoost$_{\text{Reg}}$ are shown. We observe – as initially proposed – that the influences of all the examples with high weights in AdaBoost (cf. Figure 4.3 [middle]), now have lower weights, whereas other examples become more influencing: now there are at least 12 examples per class that support the decision boundary. Roughly speaking, AdaBoost$_{\text{Reg}}$ spreads the influences over more examples near the decision boundary. Seen in the framework of algorithmic stability, this demonstrates that AdaBoost$_{\text{Reg}}$ performs favorably on noisy data.

4.2.4 Summarizing Remarks

We proposed a modification of AdaBoost to reduce the influence of examples that are hard to learn. The essence of our approach is to achieve a soft margin in contrast to the hard margin classification used before. The proposed *trade-off between the margin and the influence* of an example allows to control how much we “trust” in the data. So we are permitted to ignore noisy examples (e.g. outliers), which would otherwise have spoiled our classification.

Note that we only gave one definition for the soft margin leading to AdaBoost$_{\text{Reg}}$. Other extensions that e.g. use regularization operators [e.g. Smola et al., 1998b, Rokui and Shimodaira, 1998, Bishop, 1995] or that have other functional forms [cf. Rätsch et al., 2000a] are also possible and seem reasonable. In the next section we will consider an algorithm (ν-Arc) that makes use of ideas proposed in this section.

One of the problems of AdaBoost$_{\text{Reg}}$ is that it is not known whether it converges, nor what the actual optimization problem is. The modification is done on an algorithmic level, which makes it difficult to relate the output of AdaBoost$_{\text{Reg}}$ to a solution of an optimization problem. Nevertheless, it was one of the first boosting-like algorithm that achieved state-of-the art generalization results on noisy data [cf. Rätsch, 1998]. In our experimental evaluation, we find this algorithm among the best performing ones. It is future work to follow the underlying idea of AdaBoost$_{\text{Reg}}$ and to give it a better theoretical foundation. In the next section we propose another technique based on a related idea, for which we have a much better theoretical understanding.

4.3 Algorithms based on Linear Programs

In Chapter 2 we have worked out connections of boosting to margin maximization. The algorithms under consideration are approximately solving a linear program – the margin-LP problem (2.3). As discussed in Section 4.1, for noisy problems these algorithms perform suboptimal. From (4.1), this is indeed not surprising. The minimum of the right hand side of inequality (4.1) is not necessarily achieved with the maximum (hard) margin hyperplane.

We now propose an algorithm, where one can directly control the number of margin errors and, hence, is able to control the contribution of both terms in inequality (4.1) separately. We first propose an extended linear program – the ν-LP problem – and analyze its solution. In Section 4.3.2 we propose to solve the problem using similar techniques as in the last section leading to the ν-Arc algorithm. Whereas this algorithm turned out to work rather well in practice and it is clear where it is supposed to converge to, it is difficult to prove its convergence to the solution to the ν-LP problem. We therefore propose a later developed barrier algorithm, for which we can finally show convergence. In our empirical evaluation (cf. Section 4.4.1), we will indeed find that the barrier algorithm algorithm performs slightly better than the ν-Arc. The improved algorithm is therefore preferable from a theoretical and a practical point of view.

4.3.1 The ν-LP Problem

Let us consider the case where we have given a (finite) set $\mathsf{H} = \{h_j : \boldsymbol{x} \mapsto [-1,1], j = 1,\dots,J\}$ of J hypotheses. To find the coefficients $\hat{\boldsymbol{\alpha}}$ for the combined hypothesis $f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x})$, we extend the margin-LP problem (2.3) and solve the following linear optimization problem [see also Bennett and Mangasarian, 1992, Rätsch et al., 2000a], which we call the ν-LP problem:

$$\begin{array}{lll} \max & \rho - \frac{1}{\nu N}\sum_{n=1}^{N}\xi_n & \\ \text{with} & y_n f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}_n) \geq \rho - \xi_n & \text{for all } n = 1,\dots,N \\ & \xi_n, \hat{\alpha}_j, \rho \geq 0 & \text{for all } j = 1,\dots,J \text{ and } n = 1,\dots,N \\ & \sum_{j=1}^{J}\hat{\alpha}_j = 1, & \end{array} \tag{4.12}$$

where $\nu \in (1/N, 1)$ is a parameter of the algorithm. Here, one does not force all margins to be greater than zero and we obtain a *soft margin* hyperplane with a regularization constant $\frac{1}{\nu N}$. The dual optimization problem of (4.12) is the same as the edge-LP problem (2.3) with one additional constraint: $d_n \leq \frac{1}{\nu N}$ [Rätsch and Warmuth, 2000, Problem L3]. Since d_n is the Lagrange multiplier to the constraint for the n-th example, its size characterizes how much the example influences the solution of (4.12). Seen in the context of *algorithmic stability*, we see that the introduction of the soft-margin by using slack-variables ξ_n limits the *influence* of each example: whereas one example can have an arbitrary high influence (large Lagrange multiplier d_n) in the edge-LP problem (2.3), here it is bounded by $\frac{1}{\nu N}$. Thus, our approach can also be justified by following the same lines of reasoning as for $\text{AdaBoost}_{\text{Reg}}$.

The following proposition shows that ν has an immediate interpretation:

Proposition 4.1 (ν-Property, e.g. Schölkopf et al. [2000], Graepel et al. [1999], Rätsch et al. [2002]). *Suppose we run the algorithm given in (4.12) on some data with the resulting optimal $\rho > 0$. Then*

1. *ν upper-bounds the fraction of margin errors.*
2. *$1 - \nu$ is an upper bound on the fraction of examples with a margin larger than ρ.*

For a sketch of the proof see Figure 4.4 [cf. also Schölkopf et al., 2000, Graepel et al., 1999, Rätsch et al., 2002].

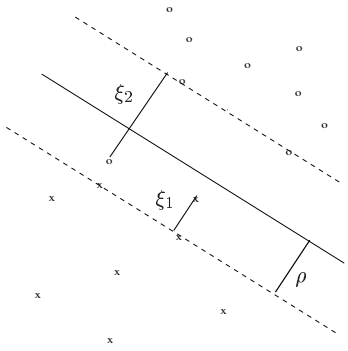

Figure 4.4 Graphical proof of the ν-property. Imagine decreasing ρ, starting from some large value. The first term in $\nu\rho - \frac{1}{N}\sum_{n=1}^{N}\xi_n$ (cf. (4.12)) will decrease proportionally to ν, while the second term will decrease proportionally to the fraction of points outside of the margin area. Hence, ρ will shrink as long as the latter fraction is larger than ν. At the optimum, it therefore must be $\leq \nu$ (Proposition 4.1, 1). Next, imagine increasing ρ, starting from 0. Again, the change in the first term is proportional to ν, but this time, the change in the second term is proportional to the fraction of examples in the margin area or *exactly on* the margin. Hence, ρ will grow as long as the latter fraction is smaller than ν, eventually leading to Proposition 4.1, 2.

Using this property we can now state a generalization error bound in terms of ν and $\rho(\nu) \equiv \rho^{(\nu)}$ using (4.1) that bounds $R[f]$ for ensemble methods in terms of the margin distribution. By Proposition 4.1 we have that the number of examples with a margin smaller than ρ is bounded by ν, i.e. $R_\rho[f] \leq \nu$. Hence, we obtain the following simple reformulation[6] of Eq. (4.1):

Proposition 4.2 (Rätsch et al. [2000a]). *Let $P(\mathsf{X}, \mathsf{Y})$ be a distribution over $\mathbb{X} \times \{-1, +1\}$, and let S be a sample of N examples chosen i.i.d. according to P. Suppose the base-hypothesis space H has VC dimension ϑ, and let $\delta > 0$. Then with probability at least $1 - \delta$ over the random choice of the training set X, Y, every function $f \in \mathrm{co}(\mathsf{H})$ generated by the algorithms above satisfies the following bound for all $\nu \in (0, 1)$ with $\rho^{(\nu)} > 0$, where $\rho^{(\nu)}$ is the solution to (4.12):*

$$R[f] \leq \nu + \mathcal{O}\left(\sqrt{\frac{1}{N}\left(\frac{\vartheta \log^2(N/\vartheta)}{(\rho^{(\nu)})^2} + \log\left(\frac{1}{\delta}\right)\right)}\right). \tag{4.13}$$

Thus, the tradeoff in minimizing the right hand side between the first and the second term is controlled directly by the easy interpretable regularization parameter ν.

Remark 4.1. *There is another kind of bound that could be derived using* compression schemes *(see Appendix B.5 for some definitions and results). It only depends on the fraction $\overline{\nu} \approx \nu$ of examples that are* in or on the edge of *the margin area. It does not depend on the*

6. A similar result has been obtained for SVMs in [Schölkopf et al., 2000, Proposition 16].

base hypothesis space. Furthermore, it can be shown [Floyd and Warmuth, 1995] that it already yields non-trivial results for $\overline{\nu} < \frac{1}{2}$ *(cf. Appendix B.5).*

Since the slack variables ξ_n only enter the cost function linearly, their absolute size is not important. Loosely speaking, this is due to the fact that for the optimum of the primal objective function, only derivatives wrt. the primal variables matter, and the derivative of a linear function is constant. This can be made more explicit: Any example *outside* the margin area, i.e. satisfying $y_n f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}_n) > \rho$, can be moved arbitrarily, as long as it does not enter the margin area. Only if the example is exactly on the edge of the margin area, i.e. $y_n f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}_n) = \rho$, then (almost) no local movement is possible without changing the solution. If the example $(\boldsymbol{x}_n, y_n)$ is in the margin area, i.e. $\xi_n > 0$, we have the following:

Proposition 4.3. *Given a training sample* S *and* $\nu \in (0,1)$*. Let* ρ^**,* $\hat{\boldsymbol{\alpha}}^*$ *and* $\boldsymbol{\xi}^*$ *be the solution of (4.12). Let* $\mathbf{d}^*$ *and* γ^* *be the corresponding dual solution. For any training example* $(\boldsymbol{x}_n, y_n)$ *with* $\xi_n > 0$ *(i.e.* $y_n f_{\hat{\boldsymbol{\alpha}}^*}(\boldsymbol{x}_n) < \rho^*$*) holds:*

1. *A local movement of the example in feature space* $\hat{\boldsymbol{x}}_n = \Phi(\boldsymbol{x}_n)$ *(cf. (1.23))*
   (a) *perpendicular to all coordinates* j *satisfying* $\langle U_{*,j}, d^* \rangle = \gamma^*$ *does not change the solution of (4.12), and*
   (b) *in direction parallel to* $\boldsymbol{\Delta}$ *with* $\Delta_j = \begin{cases} y_n & \langle U_{*,j}, d^* \rangle = \gamma^* \\ 0 & \text{otherwise} \end{cases}$ *does not change* $\hat{\boldsymbol{\alpha}}^*$ *and* ρ^**,*
   where $U_{nj} = y_n h_j(\boldsymbol{x}_n)$ *(as in Section 1.3.3). Also, any combination of 1. and 2. does not change* $\hat{\boldsymbol{\alpha}}^*$ *and* ρ^**.*
2. *If the label* y_n *is changed to the opposite class* ($y'_n = -y_n$)*, it does not change* $\hat{\boldsymbol{\alpha}}^*$ *and* ρ^**.*

The proof is shown in Appendix A.3.1 on page 142.

So, there is a large degree of "freedom" for distorting the example and even the label without changing the classification rule. In fact, if only a few hypotheses are combined ($\hat{\boldsymbol{\alpha}}$ is very sparse), then most of the coefficients are zero and there are more allowed directions to modify the examples. One can show that each example can be moved locally in an at least $J - N + 1$-dimensional subspace of the feature space $\mathbb{F}$ (cf. Section 5.2). This yields a desirable *robustness* property. Note, in the case of SVMs one can only move them locally parallel to the normal vector $\boldsymbol{w}$ of the hyperplane [Schölkopf et al., 2000], i.e. in an one dimensional subspace of $\mathbb{F}$. Summarizing, the solution essentially depends on the *number* of outliers (or other unreliable examples), not on the size of the margin error [Rätsch et al., 2000a]. This number could e.g. be known *a priori*, depending on how noisy the data is, potentially leading to good initial guess of ν.

4.3.2 ν-Arc

We are now going to present our first algorithm. Suppose we have a very large base hypothesis class H. Then it seems to be difficult to solve (4.12) directly. To this end, we propose an algorithm – called *ν-Arc* – that is supposed to approximate the solution of (4.12) [cf. Rätsch et al., 2000a]. The idea is to transform the linear program (4.12) to a min-max problem of type (4.10) and then use e.g. AdaBoost or Arc-GV to approximate the solution of (4.12) (cf. Section 4.2.2).

Suppose we are given some $\hat{\boldsymbol{\alpha}}$. Then, we may compute the margins $\rho_n(\hat{\boldsymbol{\alpha}})$ by (4.2). The optimal ρ for (4.12) is determined by

$$\rho = \operatorname*{argmax}_{\rho\in[0,1]} \left(\rho - \frac{1}{\nu N} \sum_{n=1}^{N} [\rho - \rho_n(\hat{\boldsymbol{\alpha}})]_+ \right). \tag{4.14}$$

where $[\kappa]_+ := \max(\kappa, 0)$. For a given margin ρ one can compute the optimal $\boldsymbol{\xi}$'s simply by $\xi_n := [\rho - \rho_n(\hat{\boldsymbol{\alpha}})]_+$. The constraint $\rho_n(\hat{\boldsymbol{\alpha}}) \geq \rho - \xi_n$ as in (4.12) can be rewritten as

$$\rho_n(\hat{\boldsymbol{\alpha}}) + \xi_n - \frac{1}{\nu N} \sum_{n=1}^{N} \xi_n \geq \rho - \frac{1}{\nu N} \sum_{n=1}^{N} \xi_n \,. \tag{4.15}$$

Two more substitutions are needed to transform the problem to a min-max problem as in (4.10). In particular we have to get rid of the slack variables ξ_n again by absorbing them into a quantity similar to $\rho_n(\hat{\boldsymbol{\alpha}})$. This works as follows: on the right hand side of (4.15) we have the objective function (cf. (4.12)) and on the left hand side a term that depends (nonlinearly) on $\hat{\boldsymbol{\alpha}}$. We define

$$\tilde{\rho}_n(\hat{\boldsymbol{\alpha}}) := \rho_n(\hat{\boldsymbol{\alpha}}) + \xi_n - \frac{1}{\nu N} \sum_{n=1}^{N} \xi_n. \tag{4.16}$$

The quantity $\tilde{\rho}_n(\hat{\boldsymbol{\alpha}})$ plays the role of a *soft margin* with $\zeta_n = \xi_n - \frac{1}{\nu N} \sum_{n=1}^{N} \xi_n$ defines some "mistrust" in the sense of Section 4.2 (cf. (4.8)). In (4.16), however, one does not need to define the "measure of mistrust" explicitly, but one specifies a *fraction* ν of examples that should be "mistrusted". The computation of $\tilde{\rho}_n(\hat{\boldsymbol{\alpha}})$ is illustrated in Figure 4.5.

Plugging-in $\tilde{\rho}_n(\hat{\boldsymbol{\alpha}})$ as in (4.16) into (4.10), we obtain a new optimization problem, which

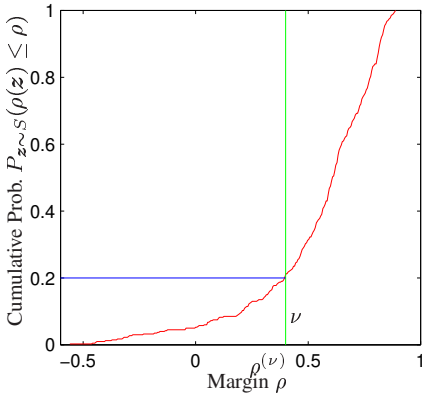

Figure 4.5 Illustration of the soft margins in ν-Arc: One computes the margin distribution graph [Schapire et al., 1998] and cuts the graph at a fraction of ν (here $\nu = 20\%$), leading to a margin value $\rho^{(\nu)}$. The soft-margin $\tilde{\rho}_n$ as in (4.16) is then computed as follows: If $\rho_n \leq \rho^{(\nu)}$, then $\tilde{\rho}_n = \rho^{(\nu)} - \zeta_n$. If $\rho_n > \rho^{(\nu)}$, then $\tilde{\rho}_n = \rho_n - \zeta_n$, where $\zeta_n = \xi_n - \sum_i \xi_i/(\nu N)$ as in (4.16).

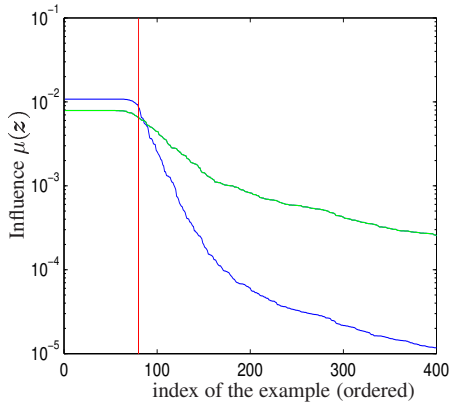

Figure 4.6 Illustration of the weighting scheme in ν-Arc: Based on the soft margin $\tilde{\rho}_n$ computed in (4.16) (cf. Figure 4.5), the weight d_n of the example $(\boldsymbol{x}_n, y_n)$ is computed by $\exp(-\tilde{\rho}_n \sum \hat{\alpha}_t)/Z_t$. A fraction of about ν examples have the same soft margin and therefore the same weight. The weights of the other examples are exponentially decaying (shown for two different values of $\sum \hat{\alpha}_t$).

has the same solution as (4.12). Following the reasoning of Section 4.2.2, we may e.g. use Arc-GV to approximately solve (4.12). Thus, by replacing the margin ρ_n with $\tilde{\rho}_n$ in $\mathrm{G}(\boldsymbol{\alpha}^{(t)})$ (cf. (4.9)), we obtain a loss function, which leads to a new algorithm. We call it *ν-Arc* [see also Rätsch et al., 2000a]. Again the example weighting $\mathbf{d}^{(t)}$ can be computed by the method proposed in Chapter 3 (cf. Figure 4.6). Algorithm 4.6 shows the pseudocode of ν-Arc.

Algorithm 4.6 The ν-Arc algorithm [Rätsch et al., 2000a]

1. **Input:** N examples $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, Number of iterations T, Regularization constant ν
2. **Initialize:** $d_n^{(1)} = 1/N$ for all $n = 1, \ldots, N$
3. **Do for** $t = 1, \ldots, T$,
   (a) Train classifier with respect to the weighted sample set $\{S, \mathbf{d}^{(t)}\}$ and obtain hypothesis $h_t : \boldsymbol{x} \mapsto [-1, 1]$
   (b) Find the hypothesis coefficient by solving $\alpha_t = \operatorname{argmin}_{\alpha_t \geq 0} \mathrm{G}(\boldsymbol{\alpha}^{(t)})$.
   (c) Compute $\rho^{(t)}$ by solving (4.14).
   (d) Update $d_n^{(t+1)} = \frac{1}{Z_t} \exp\left\{ -\left(\rho_n(\boldsymbol{\alpha}^{(t)}) - [\rho_n(\boldsymbol{\alpha}^{(t)}) - \rho^{(t)}]_+\right) \sum_{r=1}^{t} \alpha_t \right\}$, with Z_t such that $\sum_{n=1}^{N} d_n^{(t+1)} = 1$.
4. **Break if** $\alpha_t \leq 0$
5. **Output:** Final hypothesis with weights $\boldsymbol{\alpha}^{(t)}$ as in Algorithm 1.1

6. **Function** $\mathrm{G}(\boldsymbol{\alpha})$
   (a) Compute ρ by solving (4.14) and set $\xi_n = [\rho_n(\boldsymbol{\alpha}) - \rho]_+$ for all $n = 1, \ldots, N$
   (b) Return $\sum_{n=1}^{N} \exp\left[-\left(\rho_n(\boldsymbol{\alpha}) + \xi_n - \frac{1}{\nu N} \sum_{n=1}^{N} \xi_n \right) \sum_{r=1}^{t} \alpha_t \right]$

The computational costs for determining the hypothesis weight α_t is considerably higher than for AdaBoost. The implementation given in Algorithm 4.6 needs $\mathcal{O}\left(N \log^2(1/\epsilon)\right)$ of basic operations to find α_t with accuracy ϵ (using a binary search). This might be a problem, if the base hypothesis is very simple and computing α_t takes a relatively high fraction of the computing time. To avoid this problem, one may use similar approximation approaches as e.g. used in Schapire and Singer [1999].

4.3.3 A Barrier Algorithm

From an optimization point of view, both algorithms proposed so far are not satisfying. Although in practice they perform well (as we will show in Section 4.4.1), they might actually not be able to find the optimal solution to the corresponding optimization problem. We therefore propose an algorithm that uses the barrier optimization technique which we already found to be related to AdaBoost and Arc-GV. For this algorithm we can actually

show the convergence to the optimal solution of (4.12). For a brief introduction to barrier optimization in the context of boosting see Section 2.5, Appendix B.2 and Rätsch et al. [2000c]. For more details on the optimization method see e.g. Frisch [1955], Cominetti and Dussault [1994], Bertsekas [1995], Nash and Sofer [1996].

For the sake of simplicity let us slightly reformulate the linear program (4.12):

$$\begin{aligned} \min_{\hat{\boldsymbol{\alpha}},\boldsymbol{\xi}} \quad & C\sum_{j=1}^{J}\hat{\alpha}_j + \sum_{n=1}^{N}\xi_n \\ \text{with} \quad & y_n f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}_n) \geq 1-\xi_n \quad \text{for all } n=1,\ldots,N \\ & \xi_n, \hat{\alpha}_j \geq 0 \quad \text{for all } j=1,\ldots,J \text{ and } n=1,\ldots,N \end{aligned} \tag{4.17}$$

One can in fact show that (4.17) has the same solution as (4.12): for any given ν one can find a C such that both problems have the same solution (up to scaling) [cf. Demiriz et al., 2001b, Theorem 3.1]. However, unfortunately one loses the ν-property. The following derivation could in principle also be done without this transformation (using similar techniques as proposed in Rätsch et al. [2002], see also Zhang [2002]), but it makes the presentation considerably more difficult.

The barrier minimization objective for the problem (4.17) using the exponential barrier can be written as:

$$\mathrm{F}_\beta(\hat{\boldsymbol{\alpha}},\boldsymbol{\xi}) = C\sum_{j=1}^{J}\hat{\alpha}_j + \sum_{n=1}^{N}\xi_n + \beta\sum_{n=1}^{N}\left[\exp\left(-\frac{\xi_n}{\beta}\right) + \exp\left(-\frac{\xi_n + y_n f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}_n) - 1}{\beta}\right)\right]. \tag{4.18}$$

where β is the barrier parameter and we have omitted the constraints $\hat{\boldsymbol{\alpha}} \geq 0$. They will be satisfied outside the barrier optimization. The first two terms in (4.18) are the objective of (4.17), the first term in the sum corresponds to the constraints $\xi_n \geq 0$ and the last term implements the constraints $y_n f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}_n) \geq 1-\xi_n$. By setting $\nabla_{\boldsymbol{\xi}}\mathrm{F}_\beta = \mathbf{0}$, we can find the minimizing *slack variables* $\boldsymbol{\xi}$ of (4.18) for given β and $\hat{\boldsymbol{\alpha}}$:

$$\xi_n(\hat{\boldsymbol{\alpha}},\beta) = \beta\log\left[1+\exp\left(\frac{1-y_n f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}_n)}{\beta}\right)\right] \tag{4.19}$$

Thus, the problem of minimizing (4.18) is greatly simplified, as there are N variables less to optimize. Plugging-in (4.19) into (4.18) yields

$$\mathrm{F}_\beta(\hat{\boldsymbol{\alpha}}) = C\sum_{j=1}^{J}\hat{\alpha}_j + \beta\sum_{n=1}^{N}\log\left[1+\exp\left(\frac{1-y_n f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}_n)}{\beta}\right)\right] + \beta N \tag{4.20}$$

Remark 4.2. *If we set $\beta = 1$ and $C = 0$ (i.e. if we do not regularize), then we almost*[7] *obtain the logistic loss as in Section 3.1 (cf. (3.2)). The current approach can indeed be understood as a leveraging algorithm like in Chapter 3* with regularization. *Furthermore, if we let β go to zero, then the loss in (4.20) converges to the so-called* soft-margin loss *[Bennett and Mangasarian, 1992] (also* hinge loss *[Gentile and Warmuth, 1999]).*

7. There is an offset by 1. Roughly speaking, this offset is responsible for large margins.

We propose an algorithm (cf. Algorithm 4.7) that iteratively minimizes (4.20) for a fixed β up to a certain precision. Then β is reduced and the optimization restarts. In order to minimize (4.20) for fixed β we follow the leveraging scheme as in Chapter 3: In each iteration t of the algorithm, one selects a hypothesis and then minimizes with respect to the corresponding variable. There are several issues that need to be discussed, in particular:

1. Are there different assumptions on the base learner necessary for convergence?
2. How should one handle the positivity constraints?
3. When and how should β be decreased?

Questions 1 and 2 are thoroughly answered in Rätsch et al. [2002]. Here, we give a discussion of the results only.

Algorithm 4.7 The Barrier Algorithm for Classification

1. **Input:** N examples $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, Number of iterations T, Regularization constant C
2. **Initialize:** $f^{(0)} \equiv 0$, $\beta^{(1)} = \beta_{\text{start}}$, $d_n^{(1)} = [\exp(1/\beta^{(1)}) + 1]^{-1}$, $n = 1, \ldots, N$.
3. **Do for** $t = 1, \ldots, T$
   (a) Train classifier with respect to the weighted sample set $\{S, \mathbf{d}^{(t)}\}$ and obtain hypothesis $h_t : \boldsymbol{x} \mapsto [-1, 1]$
   (b) Let $\hat{\gamma}_r = \sum_{n=1}^{N} y_n d_n^{(t)} h_r(\boldsymbol{x}_n)$ and $\underline{\alpha}_r = \sum_{s=1}^{t} \alpha_r \mathbf{I}(h_s = h_r)$ for $r = 1, \ldots, t$
   (c) $r^* = \operatorname{argmin}_{i \in \mathcal{I}} \hat{\gamma}_s$, where $\mathcal{I} = \{i \mid i \in \{1, \ldots, t-1\} \text{ and } \underline{\alpha}_i > 0\}$.
   (d) **if** $\hat{\gamma}_t - C < C - \gamma_{r^*}$ **then** $h_t = h_{r^*}$ and $\underline{\alpha}_t = \underline{\alpha}_{r^*}$ **else** $\underline{\alpha}_t = 0$
   (e) Find hypothesis coefficient by solving $\alpha_t = \operatorname{argmin}_{\alpha_t \geq -\underline{\alpha}_t} \mathrm{F}_{\beta^{(t)}}(\boldsymbol{\alpha}^{(t)})$
   (f) Let $f^{(t)} = f^{(t-1)} + \alpha_t h_t$.
   (g) Update weights $d_n^{(t+1)} = \dfrac{\exp([1 - y_n f^{(t)}(\boldsymbol{x}_n)]/\beta^{(t)})}{1 + \exp([1 - y_n f^{(t)}(\boldsymbol{x}_n)]/\beta^{(t)})}$.
   (h) **if** $\left|\sum_{n=1}^{N} d_n^{(t)} h_t(\boldsymbol{x}_n) - C\right| < \beta^{(t)}$, **then** $\beta^{(t+1)} := \texttt{next}(\beta^{(t)})$ **else** $\beta^{(t+1)} = \beta^{(t)}$
4. **Output:** Final hypothesis $f^{(T)}$

To answer the first question, suppose one would use a base learner that returns hypotheses with (approximately) largest edge (cf. Definition 3.5). Then one would do positive updates only. Therefore, also the coefficients of previously selected hypothesis need to be reconsidered. One has to check whether *not* selecting them, could violate the δ-optimality condition needed for Proposition 3.3 (for the negative direction). If this is the case, then our algorithm selects such a hypothesis for the next iteration instead of using the hypothesis returned by the base learning algorithm (cf. Algorithm 4.7, step 3d). One therefore has to use a *wrapper* around the base learning algorithm (cf. Algorithm 4.7, steps 3a–3d) to

satisfy the conditions of Proposition 3.3. In condition 1 of Definition 3.4 one makes use of the vector $\boldsymbol{\gamma}$ used in the objective (3.24). In the last chapter it was zero, whereas here we set $\boldsymbol{\gamma} = C\mathbf{1}$. This makes the requirements on the base learner indeed more restricting. For instance, for very large C there might exist only one element $h_j \in \mathsf{H}$ that satisfies $\hat{\gamma}_j \geq C$ and the base learner has to return this particular hypothesis.

Furthermore, one has to ensure that the hypothesis coefficients stay positive over all iterations. Note that simply adding a constraint $\alpha_t \geq 0$ (e.g. in step 3b of Algorithm 3.4) is undue, since then one could never reduce a potentially too large coefficient (cf. discussion above). We therefore add the constraint $\alpha_t \geq -\underline{\alpha}_t$ to the minimization with respect to α_t (cf. Algorithm 4.7, step 3e). The lower bound $-\underline{\alpha}_t$ is zero for all hypotheses directly returned by the base learning algorithm and can be negative if the wrapper selects a previous hypothesis (cf. step 3b). Note that a hypothesis coefficient can only be too large, if the corresponding hypothesis has been selected in a previous iteration (cf. definition of $\mathcal{I}$ in step 3c). This discussion leads to the first part of Algorithm 4.7, for which we can show linear convergence for fixed β.

This algorithm is one instance of a family of algorithms with regularized cost functions, for which we have extended the convergence analysis presented in Section 3.4 to the case of minimizing $\mathrm{F}[f_{\hat{\boldsymbol{\alpha}}}] = \mathrm{G}(H\hat{\boldsymbol{\alpha}}) + C\|\hat{\boldsymbol{\alpha}}\|_1$. To obtain the corresponding algorithm one has to omit step 3h from Algorithm 4.7 and computes the example weighting $\mathbf{d}$ as in Algorithm 3.4 [for details see Rätsch et al., 2002]. Then holds:

Proposition 4.4 (Rätsch et al. [2002]). *Given a regularization constant $C > 0$, a training sample S and a finite hypothesis space H. Assume that each $h_j(\boldsymbol{x}_n)$ and y_n is finite. Assume the loss function G is defined on $\mathbb{R}^N$, lower bounded and strongly convex on any bounded subset of $\mathbb{R}^N$. Suppose the algorithm described above generates a sequence of combined hypotheses $f_{\hat{\boldsymbol{\alpha}}^{(0)}}, f_{\hat{\boldsymbol{\alpha}}^{(1)}}, \ldots$ using a δ-optimal base learner. Then the sequence of coefficients $\{\hat{\boldsymbol{\alpha}}^{(t)}\}$ of the combined hypotheses converges* linearly *to a minimizer of the ℓ_1-norm regularized cost function.*

The proof is given in Appendix A.3.2 on page 143.

Proposition 4.4 shows that Algorithm 4.7 converges for fixed $\beta > 0$. This proposition is indeed a nice auxiliary result of this section. It essentially shows that any leveraging algorithm as considered in Chapter 3 can be extended easily to a regularized version. The LASSO algorithm for regression [Tibshirani, 1994], the PBVM algorithm (using the ℓ_1-norm) for classification [Singer, 2000] and the sparse (kernel) Fisher discriminant (SKFD) algorithm [Mika et al., 2001] are good examples, where one could apply this result [see also Chen et al., 1995, Bradley et al., 1998]. A more detailed discussion is given in Rätsch et al. [2002].

It is left to answer the third question regarding the decrement of β. The answer is readily found in a proposition proven in Cominetti and Dussault [1994] (cf. Proposition B.1 in Appendix B.2). Roughly speaking one has to ensure that $\beta \to 0$ *and* $\|\nabla_{\hat{\boldsymbol{\alpha}} \geq 0} \mathrm{F}_\beta\| \to 0$ to achieve the desired convergence.[8] We take the following approach: We reduce β, if the

8. Here $\nabla_{\hat{\boldsymbol{\alpha}} \geq 0} \mathrm{F}_\beta$ is the projected gradient. It is actually the sub-gradient of an extended function. See proof of

(sub-)gradient with respect to $\hat{\boldsymbol{\alpha}}$ is small enough, say smaller than $\mathcal{O}(\beta)$. This is implemented in Algorithm 4.7. We can show that β is decreased only if $\beta \geq \delta \| \nabla_{\hat{\boldsymbol{\alpha}} \geq 0} \mathrm{F}_\beta \|_\infty$.

Finally, we need to answer the question how β should be decreased (at which rate). One has to ensure that if β is reduced, the gradient $\nabla_{\hat{\boldsymbol{\alpha}} \geq 0} \mathrm{F}_\beta$ cannot become arbitrarily large. Otherwise the iterates may not converge. For instance, for $\texttt{next}(\beta) = c\beta$, where $c \in (0, 1)$, we can show that the gradient changes only slightly and, obviously, β and, as we will show in the proof, the gradient will go to zero.

By this discussion we can show the desired convergence of Algorithm 4.7:

Theorem 4.3. *Assume* H *is finite, the base learner* $\mathfrak{L}$ *is* δ*-optimal and* $C > 0$*. Let* $\texttt{next}(\beta) = c\beta$ *for some fixed* $c \in (0, 1)$*. Then for* $T \to \infty$ *the output of the algorithm converges to a global solution of (4.17).*

The complete proof is given in Appendix A.3.3 on page 145.

Let us briefly discuss the relation between both algorithms we have proposed in this section. Different from the ν-Arc, we explicitly allow to do steps to reduce some hypothesis coefficients $\hat{\boldsymbol{\alpha}}$ in the barrier algorithm. In ν-Arc the length of the $\hat{\boldsymbol{\alpha}}$'s can only increase. Since the above-mentioned reduction of coefficients is essential for the convergence proof of the barrier algorithm, we conjecture that ν-Arc may not converge in all cases. Also, in the barrier algorithm we have decoupled the barrier parameter β from the length of the hypothesis weight vector (cf. Section 2.5). In the separable case it was shown that the sum of the $\hat{\boldsymbol{\alpha}}$'s grows to infinity, however, for the non-separable case, they may stay finite and the barrier parameter would not reduce to 0. This is leads to another problem in showing the convergence of ν-Arc. Despite these theoretical problems, ν-Arc has been one of the first algorithms approaching the problem of overfitting in boosting and yields state-of-the-art results. The barrier algorithm was developed later in order to solve the theoretical convergence problems connected with ν-Arc.

4.3.4 An Illustrating Toy Experiment

In a first study, we show a set of toy experiments to illustrate the general behavior of ν-Arc. The barrier algorithm behaves similar, but does not have the ν-property. As base hypothesis class H we use RBF networks (see Appendix B.6.1 and Müller et al. [1999]), and as data a two-class problem generated from several 2D Gauss blobs [Rätsch, 1998], where we randomly flipped $p = 0\%, 10\%, 20\%, 25\%$ of the labels.

We obtained the following results [cf. Rätsch et al., 2000a]:

- ν-Arc leads to approximately νN examples that are effectively used in the training of the base learner: Figure 4.7 [left] shows the fraction of examples that have high average weights during the learning process (here $\sum_{t=1}^{T} d_n^{(t)} > 1/2N$). We find that the number of the latter increases (almost) linearly with ν (see also Figure 4.6).
- It leads to the fraction ν of margin errors (cf. dashed line in Figure 4.7) exactly as stated in Proposition 4.1.

Theorem 4.3.

- The ν algorithm is more robust against label noise than Arc-GV, which we recover for $\nu = 0$, and also AdaBoost [for details see Rätsch et al., 2000a]. As illustrated in Figure 4.8, also for increasing label noise p the minimum around the optimal ν stays reasonably flat. This coincides with the interpretation of Proposition 4.2 that the optimal ν should increase with the noise level.
- The (estimated) test error (averaged over ten training sets) exhibits a rather flat minimum in ν (Figure 4.7 [right]). This indicates that just as for ν-SVMs [Schölkopf et al., 2000], ν is a well-behaved parameter in the sense that a slight misadjustment it is not harmful.
- Finally, a good parameter of ν can already be inferred from prior knowledge of the expected error. Setting it to a value similar to the latter provides a good starting point for further optimization (cf. Proposition 4.2).

Note that for $\nu = 1$ we recover the Bagging algorithm (if we use bootstrap samples), as the weights of all examples will be the same ($d_n^{(t)} = 1/N$ for all $n = 1, \ldots, N$) and also the hypothesis weights will be constant ($\hat{\alpha}_t \sim 1/T$ for all $t = 1, \ldots, T$). This can be seen by setting $\rho^{(\nu)} = 1$ and α_t to an arbitrary positive constant, since $\mathrm{G}(\boldsymbol{\alpha})$ does in this case not depend on α_t. However, then the algorithm does not "boost" the performance of the base learner enough and one is likely to underfit the data. Conversely, if $\nu \approx 0$, one obtains Arc-GV and is overfitting the data. Roughly speaking, by choosing the parameter ν appropriately, one uses the right mixture between Bagging and Boosting. This behavior is in good agreement with the bound on the generalization error (cf. Proposition 4.2).

4.3.5 Summarizing Remarks

We proposed a modification of the margin-LP problem by introducing *slack variables* to the margin constraints as similarly proposed for SVMs [Cortes and Vapnik, 1995]. The resulting ν-LP problem is shown to be robust to local movements of the examples in hypothesis feature space. We developed two algorithms that are supposed to solve the ν-LP problem. First ν-Arc, which is appealing due to its simplicity and the ν property. Then a barrier algorithm, for which we have shown convergence to a solution of the ν-LP problem. Both algorithms are in fact very similar. However, the barrier algorithm does not have the

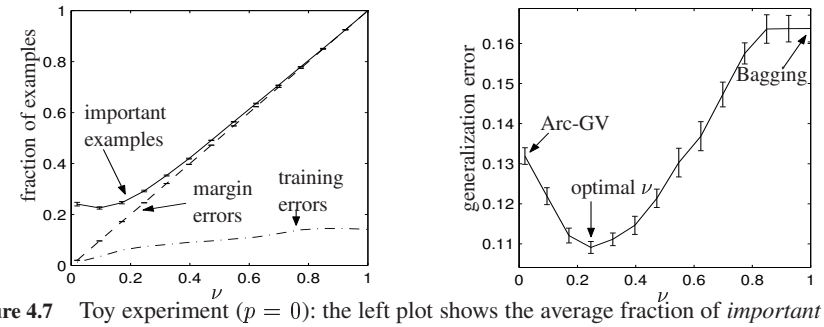

Figure 4.7 Toy experiment ($p = 0$): the left plot shows the average fraction of *important* examples (see main text), the av. fraction of margin errors and the av. training error for different values of the regularization constant ν for ν-Arc. The right plot show the corresponding generalization error. In both cases the parameter ν allows us to reduce the test errors to values much lower than for the hard margin algorithm (for $\nu = 0$ we recover AdaBoost and for $\nu = 1$ we get Bagging.)

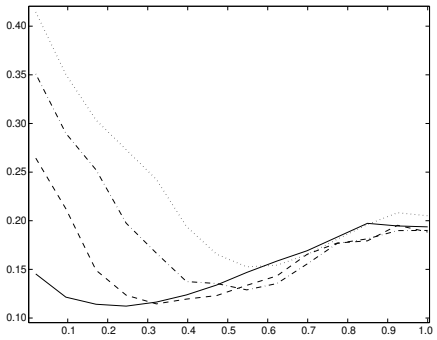

Figure 4.8 Illustration robustness of ν-algorithms: Depicted is the generalization error over ν for different label noise levels (on the training set; solid=0%, dashed=10%, dot-dashed=20%, and dotted=25%). Also, the minimal generalization error is better than for plain AdaBoost, which achieved 12.3, 13.8, 15.8 and 17.7 respectively. In particular for higher noise levels the ν algorithm performed relatively better (best results 11.2%, 11.5%, 12.9%, and 15.3%, respectively).

ν property. Using the techniques proposed in Rätsch et al. [2002] for an unsupervised learning problem, one can keep the ν-property (see also Zhang [2002]).

The barrier algorithm is related to regularized logistic regression. For proving the convergence of our algorithm, we extended the analysis presented in Chapter 3 to the case with ℓ_1-norm regularization. This builds a clean basis for developing other regularized leveraging methods [e.g. Rätsch et al., 2002].

Recently, a different algorithm for solving (4.12) has been proposed by Demiriz et al. [2001b]. It uses the *Column Generation (CG) method* known from numerical optimization [e.g. Nash and Sofer, 1996, Section 7.4]. The basic idea of column generation is to iteratively construct the *optimal* ensemble for a *restricted subset* of the hypothesis space, which is iteratively extended. This approach has indeed some advantages over the presented approaches, in particular: (i) well defined stopping criteria and (ii) finite termination at the optimal solution [cf. Demiriz et al., 2001b]. However, it requires to solve (or update) a linear program exactly in each iteration, which makes the implementation considerably more difficult. The barrier approach combines the merits of ν-Arc and the CG approach, as it is a relatively simple algorithm with good convergence properties.

In Chapter 5 we consider regression algorithms. Here the quality of the approximation of the global solution is of higher importance and we will in fact use the CG technique to derive a regression algorithm.

4.4 Evaluation and an Application

This section summarizes our efforts to show that the proposed algorithms indeed lead to significant improvements over the original AdaBoost algorithm. In Section 4.4.1 it is shown that the our new techniques compare favorably to other algorithms and perform as well as SVMs. These results place regularized boosting techniques into the standard toolbox of data analysis techniques. In Section 4.4.2, we consider an application in a real world setting and can solve an interesting problem for the electric power industry.

4.4.1 Evaluation on Benchmark Data Sets

In order to evaluate the performance of our new algorithms, we perform large scale simulations on benchmark datasets. On one side we would like to know, how they compare with traditional techniques like K-Nearest Neighbor classifiers (KNN) [e.g. Cover and Hart, 1967], decision trees [Quinlan, 1992] and neural networks (like the RBF nets, which we have used in previous sections). On the other side we illustrate that our efforts to improve the original AdaBoost algorithm have been fruitful. For completeness we also compare our methods with another state-of-the-art algorithm – the Support Vector Machine.

4.4.1.1 Experimental setup

We evaluate eight different algorithms: KNN, decision trees, neural networks, AdaBoost, AdaBoost$_{\text{Reg}}$, ν-Arc, the barrier algorithm, and SVMs. The comparison is performed on ten artificial and real world data sets from the UCI, DELVE and STATLOG benchmark repositories: banana (the toy data set used in the previous sections), breast cancer,[9] diabetes, german, heart, ringnorm, flare solar, new-thyroid, titanic, and waveform.[10] This collection is a well-balanced mixture of different learning tasks, which contains high and low noise problems with just a few examples or with many. Some of the problems are originally not binary classification problems, hence a random partition into two classes is used.[11] At first we generate 100 random realizations of training and test set (mostly $\approx 60\% : 40\%$). On each realization we train a classifier and then compute its test error.

There are several model parameters to be found for each method (discussed below). To find them we employ 10-fold cross validation (CV) [e.g. Bishop, 1995]. We start with some initial candidates for the model parameters and compute the generalization error estimates using CV. This is independently done for each of the ten data sets on the first five realizations. This leads to five estimates of optimal parameters for each data set.[12] The CV method is known to exhibit rather high variance in estimating the generalization error. To make our comparison more reliable, we select the model parameters for one particular data set to be the median of the five estimates of the optimal parameters. The median estimate compensates the high variance of the CV estimates [cf. discussion in Müller

9. The b. cancer domain was obtained from the Univ. Med. Center, Inst. of Oncology, Ljubljana. Thanks go to M. Zwitter and M. Soklic the data.

10. For the sake of brevity we omit a detailed description of the data sets. We preprocessed them and made them available on `http://ida.first.gmd.de/~raetsch/data/benchmarks.htm`(including a description, the splits into the 100 realizations and details of the simulation results). In the meantime this new repository has been frequently used by other researchers in the field [e.g. Weston, 1999, Herbrich and Weston, 1999, Chapelle et al., 2000, Pérez-Cruz et al., 2001], since it is a good basis for comparative studies [cf. Müller et al., 2001].

11. A random partition generates a mapping $\mathbf{m}$ of n to two classes. For this a random ± 1 vector $\mathbf{m}$ of length n is generated. The positive classes (and the negative respectively) are then concatenated. This partition is fixed afterwards.

12. The parameters selected by the cross validation are only close to optimal. Only 15-25 values for each parameter are tested in two stages: first a global search (i.e. over a wide range of the parameter space) was done to find a good guess of the parameter, which becomes more precise in the second stage.

et al., 2001, Section VII.B]. Finally, the model parameters used throughout the training and testing on all 100 realizations of one data set. This way of estimating the parameters is computationally highly expensive, but makes our comparison more robust and the results more reliable.

Let us briefly consider each of the eight techniques we use in this comparison to summarize the model parameters needed to be determined by the model selection procedure discussed above:

1. For the KNN algorithm one needs to find the best number of neighbors K. We considered 50 different values from the interval $[1, \ldots, N]$ (uniformly in logarithm), where N is the sample size of the particular data set.
2. As decision tree algorithm we use C4.5 [Quinlan, 1992], which actually has many parameters. The most important one controls the minimal number of examples per node. We considered 25 different values for this parameter on the interval $[1, \ldots, N]$ (uniformly in logarithmic space).
3. As a typical neural network algorithm we use RBF nets with adaptive centers as described in Appendix B.6.1. For selecting the best RBF model we optimize the number of centers (hidden units) and the number of iterations for adapting the centers and RBF widths. We considered up to 30 centers and up to 10 iterations of optimization. The regularization parameter was fixed to 10^{-6}, which is a reasonable choice [cf. Bishop, 1995, Orr, 1996, Müller et al., 1999].
4. For AdaBoost as well as for all other ensemble algorithms, we combine 200 hypotheses. Clearly, this number of hypotheses is somewhat arbitrary and may not be optimal. In an earlier study [Rätsch, 1998], we verified that AdaBoost with some optimal early stopping [e.g. Bishop, 1995] is not significantly better than AdaBoost with 200 iterations in our setting.[13] However, since we use a fixed number of iterations for all leveraging algorithms, this comparison should be fair.
5. AdaBoost$_{\text{Reg}}$ has the regularization parameter C to be found. Here we consider an interval from $[1, 10^6]$ (uniformly in logarithmic space). Since it is so large, we need two refinements of the search intervals [cf. footnote 12 and Rätsch et al., 2001].
6. Finding the parameter of ν-Arc is easier, since one only needs to find $\nu \in [0, 1]$.
7. For the barrier algorithm we need to find the regularization parameter C. A search on the interval $[0.1, 300]$ turned out out to be sufficient (uniformly in logarithmic space).
8. Finally for SVMs we decided to use an RBF kernel $k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{\sigma^2}\right)$ [e.g. Schölkopf et al., 1997], which has one parameter (the RBF width σ). Furthermore, we need to find the regularization constant C determining the complexity trade-off. We performed a search over the intervals $[0.1, 100]$ and $[0.1, 1000]$, respectively.

As base hypotheses for the ensemble algorithms we use RBF nets. We did not particularly

13. It is most of the time worse than any of the proposed soft margin algorithms (just as AdaBoost with 200 iterations; see results and cf. [Rätsch, 1998]). In neural networks early stopping is frequently used [e.g. Bishop, 1995, Orr and Müller, 1998]. However, it has been argued that shrinkage (regularization) often leads to better results than early stopping [cf. Copas, 1983, Friedman, 1999].

Table 4.1 Comparison among the eight methods: K-Nearest Neighbor Classifier (KNN), Decision trees (C4.5), Single RBF classifier, AdaBoost (AB), AdaBoost$_{\text{Reg}}$ (AB$_R$), ν-Arc, the barrier algorithm (C-Bar), and SVMs: Estimation of generalization error in % on ten data sets. The results of the best method and of all other methods with no significant difference (99% t-test) are set in bold face.

	KNN	C4.5	RBF	AB	AB$_R$	ν-Arc	C-Bar	SVM
Banana	15.0±1.0	16.1±2.8	**10.8±0.6**	12.3±0.7	**10.9±0.4**	**10.8±0.5**	**10.9±0.5**	11.5±0.7
B.Cancer	28.4±4.4	**24.6±4.5**	27.6±4.7	30.4±4.7	26.5±4.5	**25.8±4.6**	**25.9±4.4**	**26.0±4.7**
Diabetes	28.9±2.4	26.0±2.4	24.3±1.9	26.5±2.3	**23.8±1.8**	**23.7±2.0**	**23.7±1.8**	**23.5±1.7**
German	28.9±1.9	28.1±2.4	24.7±2.4	27.5±2.5	**24.3±2.1**	24.4±2.2	**24.3±2.4**	**23.6±2.1**
Heart	**15.8±3.3**	20.4±4.6	17.6±3.3	20.3±3.4	**16.5±3.5**	**16.5±3.6**	**17.0±3.4**	**16.0±3.3**
Ringnorm	35.9±1.3	15.3±1.5	1.7±0.2	1.9±0.3	**1.6±0.1**	1.7±0.2	1.7±0.2	1.7±0.1
F.Solar	37.8±2.8	33.2±1.9	34.4±2.0	35.7±1.8	34.2±2.2	34.4±1.9	33.7±1.9	**32.4±1.8**
Thyroid	5.8±2.8	8.7±3.3	**4.5±2.1**	**4.4±2.2**	**4.6±2.2**	**4.4±2.2**	**4.5±2.2**	**4.8±2.2**
Titanic	25.5±3.8	22.9±1.5	23.3±1.3	**22.6±1.2**	**22.6±1.2**	23.0±1.4	**22.4±1.1**	**22.4±1.0**
Waveform	11.4±0.8	17.8±1.0	10.7±1.1	10.8±0.6	**9.8±0.8**	10.0±0.7	**9.7±0.5**	9.9±0.4
Mean%	2400±6800	1200±2700	5.8±3.7	13.4±9.2	2.7±2.5	3.3±2.5	3.0±2.9	2.9±3.5

optimize the model parameters of the base learner for the ensemble algorithms. For simplicity we took about the best model parameters for single RBF networks (as found in step 3). We decided to reduce the number of centers and optimization iterations slightly, to avoid that the RBF network is able to overfit the data – then leveraging would not make much sense.

Note, to perform the simulations in this setup we had to train more than 3×10^6 adaptive RBF nets and to solve more than 10^5 linear or quadratic programming problems – a task that would have taken altogether 2 years of computing time on a single Ultra-SPARC machine, if we had not distributed it over about 30 computers.

4.4.1.2 Experimental Results and Discussion

Using the model parameters found as described in the last section, we computed the test set errors of all 100 realizations for each of the ten data sets. In Table 4.1 the average generalization performance estimates (with standard deviation) are given. The last line in Table 4.1 showing 'Mean%', is computed as follows: For each data set the average error rates of all classifier types are divided by the minimum error rate for this data set and 1 is subtracted. These resulting numbers are averaged over the ten data sets and the variance is computed.

From our experimental results (cf. Table 4.1) we observe the following:

- The classical techniques KNN and C4.5 that are so often used in data analysis perform very bad (except in one case for KNN and one for C4.5). On some data sets they show an up to a factor of 20 times higher test set errors than the other methods.
- The single RBF network performs already very well. This was also found in Müller et al. [1999] for regression problems. It can easily outperform KNN and C4.5.
- The results of AdaBoost are in seven cases significantly *worse* than the single RBF classifier (according to a 99% t-test). In one case AdaBoost is significantly better. This

comparably bad performance is clearly due to the overfitting of AdaBoost as discussed before.

- The proposed algorithms AdaBoost$_{\text{Reg}}$, ν-Arc and the barrier algorithm are able to improve to good results of RBF networks significantly in six cases and just slightly in other two cases. In no case the single RBF network performs significantly better than any of our three regularized leveraging algorithms.
- Compared with AdaBoost they perform significantly better in eight cases and in no case significantly worse This shows that the regularization is effective and improves the performance.
- The three regularized leveraging methods perform similar, whereas ν-Arc is slightly worse than the AdaBoost$_{\text{Reg}}$ and the barrier algorithm. In eight out of ten cases, AdaBoost$_{\text{Reg}}$ and the barrier algorithm perform best or not significantly worse than the best method. ν-Arc performs only five times among the best. This can be partially explained by some convergence problems, which we could not expel theoretically.
- Compared with the results of Support Vector Machines, AdaBoost$_{\text{Reg}}$ and the barrier algorithm perform very competitive. There is no significant difference between the performances of these methods. AdaBoost$_{\text{Reg}}$ might perform slightly better and the barrier algorithm slightly worse than SVMs. To be able to measure significant differences, larger simulations are necessary.

Summarizing, our three methods perform very well. In most of the cases they perform significantly better than all other methods. Among them AdaBoost$_{\text{Reg}}$ and the barrier algorithm perform slightly better than ν-Arc and yield competitive results compared to SVMs. This demonstrates the noise robustness of the proposed algorithm.

Since we did not optimize the base learner for the ensemble learning, we conjecture that these results can be slightly improved to finally perform better than SVMs. Note that we have optimized the kernel parameter for SVMs, which roughly equivalent to optimizing the model parameters of the base learner for the ensemble algorithms.

Summarizing, the original AdaBoost algorithm is useful for *low* noise data sets (e.g. ringnorm and thyroid), where the classes are relatively easily separable [also as shown for OCR problems in Schwenk and Bengio, 1997, LeCun et al., 1995]. Our regularized versions of AdaBoost extend the applicability of boosting methods to non-separable cases and should be preferably applied if the data is noisy.

4.4.2 An Application to a Non-intrusive Power Monitoring System

In this section we consider an application of one of our techniques in a *real world problem*. To underline its significance, we give some additional background information.

4.4.2.1 Motivation

The most difficult problem for power companies is to handle *short-term peak loads* for which additional power plants need to be built to give security against a peak load instigated

power failure. Alternatively one could also think of the following future scenario, where the power companies would be able to (remotely) control new generation household appliances. At the time of peak demand, the power companies would remotely down-regulate household appliances, e.g. air conditioners, and every individual appliance would lose e.g. 0.5% efficiency, without being noticeable by the home owners. This proposed technique would be ideally suited for the power companies to smooth the peak demands.

A prerequisite for controlling the electric energy demand, however, is the ability to correctly and non-intrusively detect and classify the operating status of electric appliances of individual households. Our goal is to develop such a non-intrusive measuring system for assessing the status of electric appliances. This is a hard problem in particular for appliances with inverter systems,[14] whereas non-intrusive measuring systems have already been developed for conventional on/off (non-inverter) operating electric equipments [cf. Hart, 1992, Carmichael, 1990]. Figure 4.9 illustrates electric load curves of an air conditioner with inverter and a refrigerator without inverter.

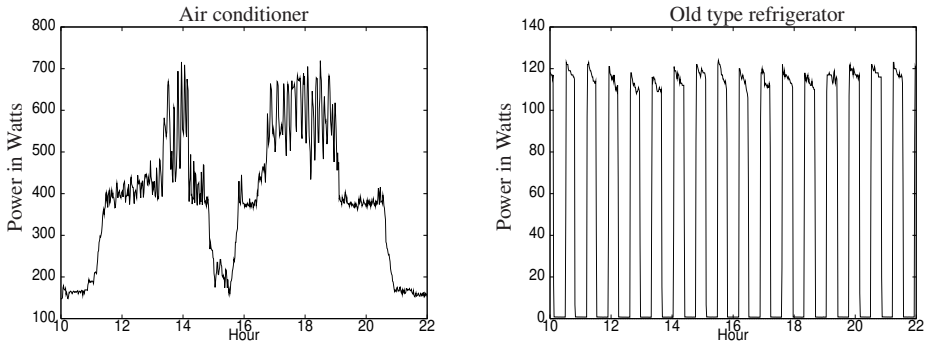

Figure 4.9 Load Curves of an air conditioner with inverter [left] and an old-type refrigerator [right]. Clearly, the load curve of the air conditioner is more complex than that of the refrigerator. The operating behavior of the refrigerator is characterized by repeated on/off operation, which is a comparatively simple pattern, whereas the operating behavior of the air conditioner depends on the modulation of the inverter system and has a more complicated electric load curve.

Here we present a first evaluation of machine learning techniques to classify the operating status of electric appliances (with and without inverter) for the purpose of constructing a non-intrusive monitoring system. In this study, we compare RBF networks, K-nearest neighbor classifiers (KNNs) [e.g. Cover and Hart, 1967], SVMs (cf. Section 1.3.2) and ν-Arc (cf. Section 4.3.2).[15]

4.4.2.2 Experimental Setup and Results

For the sake of brevity we omit a detailed description of the data measurement system [see Yoshimoto and Nakano, 1999, Onoda et al., 2000]. We used six appliances: air conditioner,

14. An inverter system controls e.g. the rotation speed of a motor (as in air-conditioners) by changing the frequency of the electric current.
15. We have chosen ν-Arc, since the parameter ν was easiest to control.

fridge (both with inverter), old type fridge (cf. Figure 4.9), incandescent light, fluorescence light and a television set. For 36 different on/off states of the appliances, the odd-order harmonic currents up to the 13-th order and respective the phase angles have been measured by a special purpose hardware device at a stable load point. All measurements and the control of the individual appliances were done manually, which makes the process of gathering data expensive and gives an explanation why there are so few data points. A larger data set is currently collected and analyzed [Onoda et al., 2001b]. Also, we have applied for a patent.

Since the data set is very small, one has to be very careful with finding good model parameters. At the same time the learning machine needs to be representative enough, but not overly complex. To get reliable results, also the experimental setup and evaluation procedure have to be quite sophisticated [cf. Onoda et al., 2000]. We start with summarizing which model parameters have to be chosen (the range tested is given in brackets):

1. The RBF network optimizes a regularized mean-squared error function. Model parameters are the number of RBF centers $[3, \ldots, 10]$, the regularization parameter $[10^{-8} \ldots 1]$ and the number of iterations for optimization $[0 \ldots 15]$.
2. For ν-Arc we use RBF networks as base hypotheses. As model parameters we have to find the regularization parameter ν $[0 \ldots 1]$, which controls the number of margin errors, and the optimal number T of boosting iterations $[1, \ldots, 30]$.
3. The SVM has the regularization parameter C $[10^{-1} \ldots 10^{5}]$ (cf. Section 1.3.2). In our experiments we use a RBF kernel $k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{\sigma^2}\right)$ that has an additional parameter σ $[10^{-3} \ldots 10^{3}]$.
4. To use the K-nearest neighbor algorithm, one needs to specify the number $[1, 3, \ldots]$ of neighbors.

For each method we have to find up to three model parameters. Since the number of observations is quite small, it is difficult to get estimates for good model parameters. We use *leave-one-out* cross-validation [e.g. Bishop, 1995], which is known to be rather reliable [Luntz and Brailowsky, 1969], but computationally expensive.

We setup an experiment that uses random splits of the whole data set: First, we generate 20 realizations of training and test data. We split the 36 examples into 30 training and 6 test examples. Then we do the model selection procedure on each single data set and compute the test error for the selected model. Finally, we have 20 estimates of the generalization error (each on 6 test examples). These estimates are averaged to get a more reliable estimate and a confidence interval was computed.

We observe from the Table 4.2 that the best average performance is achieved by ν-Arc (significant according to a 99% t-test) followed by the SVM. Clearly, the classical KNN yields the worst classification rate. ν-Arc improves the performance of the single RBF networks considerably.

We believe that the slightly worse performance of SVMs compared to ν-Arc is due to the fixed kernel width. The RBF networks can detect multi-scaling information in the data, but optimize a suboptimal loss function (mean squared error). ν-Arc can combine both virtues – maximizing the margin and "looking" at different scales of the data.

	KNN	RBF	ν-Arc	SVM
Air Cond.	8.0±1.2	3.7±2.0	**1.9±0.5**	2.5±0.7
Fridge	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
old Fridge	35.7±1.4	26.9±2.5	**19.0±2.2**	24.7±1.3
incand. L.	21.0±2.1	24.4±4.2	**11.5±1.1**	12.8±0.9
fluor. L.	35.8±1.5	16.4±3.6	12.9±1.1	**10.2±0.9**
TV	3.3±0.5	2.8±1.4	1.3±0.5	**0.5±0.3**
Av. Perf.	17.3±1.1	12.2±2.3	**7.8±0.9**	8.5±0.6

Table 4.2 Test set errors (incl. confidence intervals) in % averaged over 20 random splits into train and test data for K-Nearest Neighbor classifier (KNN), RBF network, ν-Arc, and SVMs with RBF-kernel. The model parameters are found by leave-one-out cross-validation on each realization of the training data separately.

4.4.2.3 Summarizing Remarks

Summarizing, we proposed a construction of a non-intrusive load monitoring system based on measuring harmonic waves, and evaluated the applicability of RBF networks, KNN, SVM and ν-Arc. At this point our claims are (a) that our careful leave-one-out strategy already reveals the potential of the classifiers on the task (in a controlled statistical setting) and (b) from the application point of view, the classifier precision is already reliable enough to assess the system state of the electric appliances, in particular of appliances with inverter systems. This is a novelty as previous techniques have only been able to assess the status of non-inverter type appliances.

Our results suggest that the future vision of a control system to balance load peaks – a large progress from the the environmental point of view – might not at all belong in the realm of "science fiction" anymore.

4.5 Discussion and Summary

We discussed in detail that AdaBoost-type algorithms and hard margin classifiers in general are noise sensitive and prone to overfit. We introduced two regularization strategies for AdaBoost to alleviate the overfitting problem: first a direct incorporation of the regularization term into AdaBoost's loss function leading to $\text{AdaBoost}_{\text{Reg}}$ and second by introducing slack variables to relax the hard margin constraint leading to the ν-LP problem. The essence of our proposed algorithms is to achieve a soft margin in contrast to the hard margin classification used before. The soft-margin approach allows to control how much we "trust" in each single example and we are permitted to ignore noisy examples (e.g. outliers), which would otherwise have spoiled our classification.

This generalization of boosting algorithm is very much in the spirit of Support Vector Machines that also tradeoff the maximization of the margin and the minimization of an upper bound of the classification errors by introducing slack variables. Whereas soft margins are used since 1995 for SVMs, in boosting they are not yet commonly used. We found theoretical and practical clear reasons why one should use them. The development of regularized boosting methods was therefore mandatory.

To show the convergence of one of our algorithms, we extended the analysis of Leveraging algorithms as for ℓ_1-norm regularized cost functions. As an important auxiliary result we could show that any leveraging algorithm can be regularized, while keeping the good convergence properties shown in Chapter 3. The proposed barrier algorithm illustrates that these results are directly applicable to design new algorithms that are eventually much more noise robust.

In our experimental evaluation on noisy data, the proposed regularized versions of AdaBoost show a significantly improved performance compared to original AdaBoost (and the base learner). Furthermore, they exhibit a better overall generalization performance than all other algorithms and are very competitive with SVMs. To illustrate the usefulness of our approach we reported results on a real world problem. Our results show that boosting-type algorithms belong to the small set of state-of-the-art methods in machine learning, if one properly regularizes the ensembles.

One problem, however, common for all ensemble learning algorithms is the relatively high computational complexity. In our experiments we often used hundreds of base hypotheses, which we combine. To make these algorithms practical, a fast base learning algorithm appropriate for the particular domain has to be available. This is two-fold. First, one may have a specialized algorithm for the problem at hand and, hence, can use domain specific knowledge to improve the performance of the base learner and also of the ensemble algorithm. Second, if such algorithm is not available, one may use some standard techniques like neural networks that usually perform well. However, then ensemble learning can become rather expensive.

In this chapter we considered classification problems only. In the next chapter we extend our view to regression problems and show how most of the developed techniques can be transferred to leveraging algorithms for regression.

5 Ensembles for Regression

Motivated by the success of boosting methods in the classification setting, it has been attempted by several researchers to transfer the boosting techniques to regression [e.g. Freund and Schapire, 1994, Fisher, 1997, Bertoni et al., 1997, Friedman, 1999, Ridgeway et al., 1999, Duffy and Helmbold, 2000a, Rätsch et al., 2000c, Shawe-Taylor and Karakoulas, 2000, Zemel and Pitassi, 2001]. However, it has been found that this transformation is rather difficult and for long no convincing boosting-like algorithm has been found for regression that is both theoretically motivated and works well in practice.

The regression problem is indeed more difficult also from a statistical point of view [cf. Vapnik, 1995]. For instance (1), in classification one only needs predict correctly near the class boundary, whereas in regression one needs to predict well on the whole domain. Furthermore (2), if one assumes that there is a reasonable large margin between the classes, then the *exact* estimation of the classication boundary is often not neccessary, whereas in regression one has often to find a function that fits all the training examples almost exactly (depending on the problem). And finally (3), the chance to overfit the training data in regression (e.g. if just a few examples are availabe) is by far higher than in classification. The last two issues concern the empirical accuracy of the estimated function and the control of capacity of the function space. These problems make the transfer of a classification technique to a useful regression algorithm complicated and partially explain why boosting techniques for regression are not yet as successful as the classification pendant.

Most of the proposed boosting-like regression algorithms have the problem of finding a function that fits the training data well enough – in particular the propositions in Freund and Schapire [1994], Bertoni et al. [1997], Shawe-Taylor and Karakoulas [2000], Zemel and Pitassi [2001]. Some of these approaches [see also Duffy and Helmbold, 2000a] try to minimize the maximal loss as opposed to some sum of losses. On noisy data, such approach often fails as a single example can spoil the estimate (cf. discussion in Section 4.1). Other approaches can fit the training data well, however, often have problems controlling the trade-off between empirical error and capacity of the function class. This particularly holds for the LS-Boost algorithm [Friedman, 1999] described in Section 3.1, but also for the approaches described in Ridgeway et al. [1999]. For a more detailed review of existing methods see e.g. Rätsch et al. [2002].

In this chapter we will propose two algorithms that do not suffer from these problems. We show that they are able to fit the training data well. Furthermore, they use a well defined regularization strategy that makes the algorithms fast to train and the further analysis of the generalization performance easier.

One major difficulty is rigorously defining the regression problem for infinite hypothesis sets. For classification assuming each hypothesis has a finite set of possible outputs, the hypothesis set is always finite since there are only a finite number of ways to label any finite training set. For regression, even relatively simple base hypotheses, such as linear functions constructed using weighted least squares, consist of an uncountable infinite set of hypotheses. It is not *a priori* clear how to even express a regression problem in an infinite hypothesis set from an optimization point of view. Clearly we can only practically consider ensemble functions that are a linear combination of some small subset of the set of possible hypotheses.

In this chapter, we study directly the issue of infinite hypothesis sets. In Section 5.1, we review a mathematical programming (MP) approach to sparse regression and show how it can be applied to build regression ensembles from finite hypothesis sets. In Section 5.3 we investigate the dual of this optimization problem for ensemble regression and propose a semi-infinite linear program formulation for "boosting" of infinite hypothesis sets: first in the dual and then in the primal space. The dual problem is called semi-infinite because it has an infinite number of constraints and a finite number of variables. An important sparseness property of the semi-infinite regression problem is that it has a solution consisting of a small number of hypotheses. In Section 5.4, we propose two different algorithms for efficiently computing ensembles for regression. The exact implementation of these algorithms is dependent on the choice of base learning algorithms. In Section 5.4 we investigate three possible base learning algorithms that result in both infinite and finite hypothesis sets. Simulation results are presented in Section 5.5.

5.1 Optimization Problems and Loss Functions for Regression

In this section we consider mathematical programming formulations for regression. We start with briefly reviewing the standard regression setting and then discuss regression with ensembles, which leads us – as in classification – to a linear regression problem in the feature space $\mathbb{F}$, which can be solved by solving a convex optimization problem. Finally we discuss some loss function and state some desirable properties for the ε-insensitive loss, which will be mainly used in the subsequent sections.

5.1.1 Problem definition and Preliminaries

Consider estimating a function based on i.i.d. examples

$$(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N) \in \mathbb{X} \times \mathbb{R}. \tag{5.1}$$

The goal of the learning process is to find a function f with a small expected risk (or test error)

$$R[f] = \int_{\mathbb{X}} \mathrm{g}(y, f(\boldsymbol{x}))\, dp(\boldsymbol{x}, y), \tag{5.2}$$

where p is the probability measure, which is assumed to be responsible for the generation of the observations (5.1), and g is a loss function like $\mathrm{g}(y, f(\boldsymbol{x})) = (y - f(\boldsymbol{x}))^2$ depending on the specific regression estimation problem at hand. The problem, however, is like in classification that we cannot minimize (5.2) directly, since we do not know p. Instead, the sample (5.1) is given and one tries to obtain a small risk by minimizing the empirical risk functional over a given set of functions F.

The regression problem is often stated as finding a function $f^* \in \mathsf{F} = \{f : \mathbb{X} \to \mathbb{R}\}$ that minimizes the regularized risk functional

$$R[f] := C\,\mathbf{P}[f] + \sum_{n=1}^{N} \mathrm{g}(y_n, f(\boldsymbol{x}_n)), \quad \text{with} \quad f^* = \operatorname*{argmin}_{f \in \mathsf{F}} R[f], \tag{5.3}$$

where $\mathbf{P}[\cdot]$ is a regularization operator, and C the regularization parameter, determining the trade-off between loss and complexity. Under rather mild conditions, it has been shown that C implicitly determines the size of the function class: minimizing the regularized risk functional over the complete function class is equivalent to empirical risk minimization over a restricted class $\mathsf{F}' = \{f \mid f \in \mathsf{F}, \mathbf{P}[f] \leq C'\}$ for some C', which depends on C [Smola, 1998]. The regularization therefore defines a nested structure of function classes, which is required for the analysis in the framework of *structural risk minimization* [cf. Section 1.2.3 and e.g. Vapnik, 1995].

For simplicity, we assume for the rest of this chapter that the loss function is of the form $\mathrm{g}(y, f(\boldsymbol{x})) = \mathrm{g}(y - f(\boldsymbol{x}))$. Moreover, the loss is computed for each example independently, i.e. the loss of the N-sample is the sum of the losses of the examples. This assumption is natural, if the data is generated i.i.d. from some probability distribution. However, most derivations can be generalized to other cases.

The loss and the regularizer are assumed to be non-negative and convex. The latter assumption is indeed rather restrictive. There might exist other and better suited loss functions and regularizers that are not convex. However, finding the global minimizer of a general non-convex loss function is NP-hard [e.g. Pardalos and Vavasis, 1992]. Nevertheless, there exist several quite successful approaches like Neural Networks that often find a *good local minimizer*. Whereas the convexity assumption is not absolutely necessary for use in practice, the theoretical and empirical analysis of algorithms that solve convex optimization problems is much easier. Moreover, note that it is possible to find reasonable proxies for most real world loss functions [e.g. Wahba, 1999].

Throughout this chapter we always assume that the labels y_n and the outputs $h(\boldsymbol{x}_n)$ of the base hypotheses $h \in \mathsf{H}$ are finite. The base hypothesis space H is assumed to be closed under complementation ($h \in \mathsf{H} \;\Rightarrow\; -h \in \mathsf{H}$). Moreover, it is reasonable to assume that the loss function and the regularization operator are such that (5.3) has a finite solution. In this and the next section we assume that the base hypothesis set H contains only a finite number of hypotheses. This will be generalized to infinite hypothesis sets in Section 5.3.

5.1.2 Linear Regression in Feature Spaces

Just as in the classification case, in regression one is looking for a linear combination of hypothesis from some base hypothesis set. However, here we do not threshold the combined hypothesis at 0, but are interested in its actual value. It is often advantageous to include a bias b in the regression function, as otherwise simple offsetting the labels can change the solution drastically. Hence, we define our function space F as the space of linear combinations of *base hypotheses* of H including a bias, i.e.

$$\mathsf{F} := \left\{ f_{\hat{\boldsymbol{\alpha}},b} \;\middle|\; f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}) = b + \sum_{j=1}^{J} \hat{\alpha}_j h_j(\boldsymbol{x}),\ b \in \mathbb{R}, 0 \leq \hat{\alpha}_j \in \mathbb{R}, j = 1, \ldots, J \right\}. \quad (5.4)$$

For simplicity, we restrict the hypothesis coefficients to be non-negative. But since we assumed the base hypothesis set to be closed under complementation, this constraint does not matter.

In classification we have seen that one finds a hyperplane in *feature space* $\mathbb{F}$ spanned by the hypotheses of the base hypothesis set (cf. Section 1.3.3). In regression, though, one finds a hyperplane that describes the data well. If $C = 0$ and $\mathrm{g}(y, f(\boldsymbol{x})) = (y - f(\boldsymbol{x}))^2$, then the minimizer of the empirical risk is just the ordinary least squares solution – just in a high dimensional feature space.

Let us consider a formulation of the regularized risk minimization (cf. (5.3)) as a mathematical programming problem. We first define the matrix $H_{n,j} = h_j(\boldsymbol{x}_n)$ and can write the optimization problem as:

$$\begin{array}{ll} \min\limits_{\hat{\boldsymbol{\alpha}},b,\boldsymbol{\xi}} & C\,\mathbf{P}[f_{\hat{\boldsymbol{\alpha}}}] + \sum\limits_{n=1}^{N} \mathrm{g}(\xi_n) \\ \text{with} & y_n - H\hat{\boldsymbol{\alpha}} - b = \xi_n \qquad\qquad n = 1, \ldots, N \\ & \hat{\boldsymbol{\alpha}} \in \mathbb{R}_+^J, b \in \mathbb{R}, \boldsymbol{\xi} \in \mathbb{R}^N, \end{array} \quad (5.5)$$

Here we introduced the variables ξ_n, which are the *residual* for the n-th training example. In fact, the formulation using the residuals and the linear constraints explicitly makes the subsequent analysis easier.

5.1.3 Loss Functions

In order to construct algorithms from the optimization problems analyzed so far, one needs to specify a loss function. In this section we will present some common choices for loss functions. One of them is in particular well-suited for our purposes, which additionally has desirable theoretical properties.

5.1.3.1 The Maximum Likelihood Approach

We start with a connection between the probability density p responsible for generating the data and the asymptotically best loss function g. It is found by maximizing the likelihood of the sample (5.1):

Remark 5.1 (Smola [1998]). *Under the assumption that the examples (5.1) were generated by an underlying functional dependency f_{true} plus additive noise, i.e. $y_n = f_{true}(x_n) + \xi_n$, with density $p(\xi)$, the asymptotically optimal loss function in a maximum likelihood sense is*

$$\mathrm{g}(y, f(\boldsymbol{x})) = -\log p(y - f(\boldsymbol{x})). \tag{5.6}$$

The loss function obtained from this approach is not necessarily convex. However, since we would like to design efficient algorithms, we have to restrict ourselves to convex loss functions. Therefore, in practice one has to find a good convex approximation for the actual loss.

Of course, another problem is that the type of noise is often not known. Therefore one should use a loss function that is likely to approximate the true loss function well and, even more important, which does not change the estimated function drastically, if it is wrongly specified [cf. Huber, 1981, Amari and Kawanabe, 1997].

Moreover, the loss function obtained from (5.6) does not necessarily have to coincide with the empirical loss which is minimized in our algorithms. In particular in the small sample case, *robust loss functions* have to be considered, as the empirical risk may deviate strongly from the actual risk. It has been shown that if the slope of the loss function is bounded, then the empirical risk minimizer is robust (cf. Huber [1981]). Otherwise, as e.g. for the squared loss and the maximum loss function, relatively small distortions of the labels may result in large deviations of the estimated function. Seen in the context of algorithmic stability [Bousquet and Elisseeff, 2001b] this a property that has to be avoided.

5.1.3.2 Common Loss Functions

Let us consider some particular choices for the loss function: squared loss, Huber's loss, a strictly convex approximation of Huber's loss and the ε-insensitive loss. These loss functions are illustrated in Figure 5.1.

The most frequently used loss function is the square loss

$$\mathrm{g}_2(y - f(\boldsymbol{x})) := \frac{1}{2}(y - f(\boldsymbol{x}))^2. \tag{5.7}$$

According to Remark 5.1, the squared loss corresponds to the assumption of Gaussian noise on the labels. In the small sample case, however, the squared loss is not always the best choice, as the assumption of Gaussianity is often not fulfilled exactly (e.g. for "long tails"). Roughly speaking, this is due to the fact that only a few examples with great loss, i.e. $p(y - f(\boldsymbol{x}))$ very small (cf. Remark 5.1), can spoil the estimate. For those cases robust loss functions are more appropriate.

The first robust loss function we consider is Huber's robust loss [Huber, 1981]:

$$\mathrm{g}_H(\xi) := \begin{cases} \frac{1}{2\sigma}\xi^2 & \text{if } |\xi| \leq \sigma \\ |\xi| - \frac{\sigma}{2} & \text{otherwise.} \end{cases} \tag{5.8}$$

By continuing the quadratic loss with the linear loss, it becomes robust against outliers.

Huber's loss is closely approximated by

$$\mathrm{g}_c(\xi) := \varphi \log\left(\cosh\left(\frac{\xi}{\varphi}\right)\right). \tag{5.9}$$

for $\varphi = \frac{\sigma}{2\log(2)}$ (cf. Figure 5.1). This loss is strictly convex and twice continuously differentiable, which is useful for algorithms that require smoothness of the first derivatives (cf. Chapter 3).

Finally let us consider the ε-insensitive loss [Vapnik, 1995, Schölkopf et al., 1999a]:

$$\mathrm{g}_\varepsilon(y - f(\boldsymbol{x})) := |y - f(\boldsymbol{x})|_\varepsilon = \max(0, |y - f(\boldsymbol{x})| - \varepsilon) \tag{5.10}$$

This does not penalize errors below some $\varepsilon \geq 0$, chosen *a priori*. For the ε-insensitive loss and ℓ_1-norm regularization (cf. Section 5.2), the optimization problem (5.5) reduces to a particularly simple form, which can be expressed by a linear program. Such optimization problems can indeed be solved very efficiently, which in turn will be exploited in our algorithms.

5.1.3.3 Properties of the ε-insensitive Loss

The ε-insensitive loss has been shown to have nice properties: (i) the robustness of the fit, (ii) the applicability of the ν-trick as in classification and (iii) sparseness of the dual variables.

Let us first rewrite (5.5) for the ε-insensitive loss:

$$\begin{aligned} \min_{\hat{\boldsymbol{\alpha}}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*} \quad & C\,\mathbf{P}[f_{\hat{\boldsymbol{\alpha}}}] + \frac{1}{N}\sum_{n=1}^{N}(\xi_n + \xi_n^*) \\ \text{with} \quad & y_n - f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}_n) \leq \varepsilon + \xi_n \quad n = 1, \ldots, N \\ & f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}_n) - y_n \leq \varepsilon + \xi_n^* \quad n = 1, \ldots, N \\ & \hat{\boldsymbol{\alpha}} \in \mathbb{R}_+^J, \boldsymbol{\xi}, \boldsymbol{\xi}^* \in \mathbb{R}_+^N, b \in \mathbb{R}, \end{aligned} \tag{5.11}$$

where $f_{\hat{\boldsymbol{\alpha}}, b}$ is defined as in (5.4) and we have split-up the equality constraints into two inequality constraints. At optimality, the variables ξ_n are positive, if the function is underestimated by more than ε and otherwise zero, whereas ξ_n^* are positive if it is overestimated by more than ε.

We start with the robustness of the solution with respect to small changes of the training data:

Proposition 5.1 (Smola et al. [1999]). *Using (5.11), local movements of target values y_n of points inside and outside (i.e. not on the edge of) the ε-tube do not influence the regression.*

For completeness we give a proof along the lines of Smola et al. [1999] in Appendix A.4.1 on page 146.

Note that this proposition holds for any admissible regularization operator. For classification we have shown that one can also distort the example itself in feature space (not only

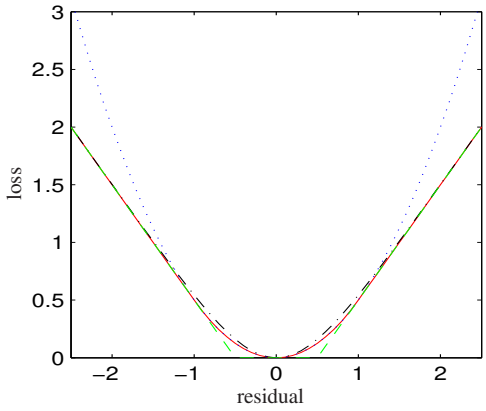

Figure 5.1 Four different loss function: squared loss (dotted), Huber's loss for $\sigma = 1$ (solid), an approximation of Huber's loss for $\varphi = 1/(2\log 2)$ (dash-dotted), and the ε-insensitive loss for $\varepsilon = 1/2$ (dashed). Near zero Huber's robust loss and its approximation are almost identical, whereas there are small deviations on the range $(0, \sigma)$. In the limit, the robust loss functions have the same linear behavior. This property makes them robust against distortions in the labels.

the label). However, this kind of robustness depends on the regularization operator. If one uses the ℓ_1-norm (cf. Section 5.2), then one can prove a similar result as for classification.

The parameter ε is usually difficult to control [Müller et al., 1999, Schölkopf et al., 2000], as one does not know beforehand how accurately one is able to fit the curve. Furthermore, the optimal ε depends on the parameter C [cf. Müller et al., 1999]. This problem is partially solved in the following optimization problem[1] for $\nu \in [\frac{1}{N}, 1]$:

$$\begin{aligned} \min_{\hat{\boldsymbol{\alpha}}, b, \varepsilon, \boldsymbol{\xi}, \boldsymbol{\xi}^*} \quad & C\,\mathbf{P}[f_{\hat{\boldsymbol{\alpha}}}] + \frac{1}{N}\sum_{n=1}^{N}(\xi_n + \xi_n^*) + \nu\varepsilon \\ \text{with} \quad & y_n - f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}_n) \leq \varepsilon + \xi_n \qquad n = 1, \ldots, N \\ & f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}_n) - y_n \leq \varepsilon + \xi_n^* \qquad n = 1, \ldots, N \\ & \hat{\boldsymbol{\alpha}} \in \mathbb{R}_+^J, \varepsilon \in \mathbb{R}_+, \boldsymbol{\xi}, \boldsymbol{\xi}^* \in \mathbb{R}_+^N, b \in \mathbb{R} \end{aligned} \tag{5.12}$$

The difference between (5.11) and (5.12) lies in the fact that ε has become a positively constrained variable of the optimization problem itself. The core aspect of (5.12) can be captured in the proposition stated below.

Proposition 5.2 (Smola et al. [1999]). *Consider the solutions of (5.12). Assume $\varepsilon > 0$. Then holds:*

(i) ν is an upper bound on the fraction of errors (i.e. points outside the ε tube).

(ii) ν is a lower bound on the fraction of points not inside (i.e. outside or on the edge of) the ε tube.

(iii) Suppose the base hypothesis set H is finite and the examples were drawn i.i.d. from a distribution $P(\boldsymbol{x}, y) = P(\boldsymbol{x})P(y|\boldsymbol{x})$ with $P(y|\boldsymbol{x})$ continuous. With probability 1, asymptotically, ν equals both the fraction of points not inside the tube and the fraction of errors.

A proof following Smola et al. [1999] and Schölkopf et al. [2000] can be found in Appendix A.4.2 on page 147. Note that the third statement can also be proven for infinite hypothesis sets, if they have well-behaved covering-numbers, $C > 0$ and the regularization operator is a ℓ_p-norm with $p \in (0, 2]$ [cf. Barnes, 1999, Williamson et al., 1999, 1998].

1. We have worked out a similar formulation for SVMs in Smola et al. [1999] and Schölkopf et al. [2000].

The optimization problem (5.12) has two parameters: (i) the regularization parameter C, which controls the size of the function space and therefore the "complexity" of the regression estimate, and (ii) the *tube-parameter* ν, which controls the fraction of examples outside the ε-tube and indirectly controls the size of the ε-tube. The parameter ν is usually easier to adjust in practice. Also, in [Smola et al., 1999, Proposition 3] the asymptotical optimal choice for ν has been derived for a certain classes of noise models [see also Schölkopf et al., 2000, Smola et al., 1998a, Smola, 1998].

In classification we only had the parameter ν which controls the size of the margin area (cf. Section 4.3), since the thresholded function is invariant under scaling. In regression however, one is interested in the outputs of the functions and also needs to control the "length" of $\hat{\boldsymbol{\alpha}}$. Therefore, one needs the parameter C.

When analyzing (5.12) one finds that all examples in the ε-tube could be removed from the training set without changing the solution. This is a unique property of linear losses.[2] There are at least two ways to exploit this fact and to prove that the regression estimate generalizes well, i.e. has small expected risk. The first is based on the leave-one-out error. It is known that the leave-one-out error is an almost unbiased estimator of the generalization error [cf. Luntz and Brailowsky, 1969]. Since, one can remove roughly νN examples from the training set, one can compute an upper bound on the leave-one-out error (assuming the loss is bounded). The second argument is based on compression schemes [Littlestone and Warmuth, 1986] originally proposed for classification. One can derive an upper bound (cf. Appendix B.5 for a sketch of proof) on the expected risk *only* depending on the fraction of examples inside the tube. It does not depend on the size or structure of the base hypothesis space. However, this generality has some drawbacks: The bound starts giving non-trivial results for ν smaller than about 15%, whereas for classification a similar bound starts at already about 50% [Floyd and Warmuth, 1995]. This gives an additional hint that regression estimation is indeed more difficult than classification. However, the above-mentioned bounds are very loose and it is hard to draw conclusions from them.

5.2 Sparseness induced by Regularization

We will now discuss two main classes of regularizers that have a drastic influence on the properties of the solution of (5.5). Some of the results might already be well-known, however, we have found no explicit treatment of the sparseness induced by the regularization for both cases in the literature.

Let us consider regularization operators of the form

$$\mathbf{P}[f_{\hat{\boldsymbol{\alpha}},b}] = \sum_{j=1}^{J} \mathbf{P}_j(\hat{\alpha}_j) \tag{5.13}$$

where $\mathbf{P}_j : \mathbb{R}_+ \to \mathbb{R}_+$ are differentiable and monotonically increasing functions[3] with $\mathbf{P}_j(0) = 0$ for all $j = 1, \ldots, J$. The analysis can be generalized to other lower bounded regularization operators.

2. It is unique among convex loss functions. See results in Section 5.2 and apply them for the dual problem.
3. Monotonicity makes sense for regularization operators to define nested subsets. Differentiability is generally not necessary, but makes the analysis much easier.

We say a hypothesis coefficient is sparse, if it contains only $\mathcal{O}(N)$ non-zero elements. Its not sparse if it is e.g. $\mathcal{O}(J)$, since the feature space is assumed to be much higher dimensional than N. We show: (i) if the regularization functions are strictly convex, then the solution to (5.5) is generally not sparse (under mild assumptions) and (ii) if the functions $\mathbf{P}_j$ are concave there exists an optimal hypothesis coefficient vector $\hat{\boldsymbol{\alpha}}$ that has at most N non-zero entries. The main result of this section is that the ℓ_1-norm regularization, which is exactly on the border between both cases, is particularly well-suited for ensemble learning.

5.2.1 Strictly Convex vs. Concave Regularization

In neural networks, SVMs, matching pursuit [e.g. Mallat and Zhang, 1993] and many other algorithms, one uses the ℓ_2-norm as regularization operator. The next proposition shows that if the regularization operator is strictly convex, the solution of (5.5) with (5.13) is likely to be not sparse:

Proposition 5.3 (Non-sparseness of strict convex regularization). *Let* $\mathsf{H} = \{h_1, \ldots, h_J\}$ *be a finite hypothesis set,* S *a training sample of size* N *and* $\mathbf{P}_j(\cdot)$ *be strictly convex regularization functions as described above. Furthermore assume* $\mathrm{g}(0) = 0$*,* $\mathrm{g}(z) \geq 0$ *and* $C \in (0, \infty)$*. If* $\hat{\boldsymbol{\alpha}}^* \neq 0$*, then the solution* $\hat{\boldsymbol{\alpha}}^*$ *of (5.5) has at most a fraction of* $\sqrt[J]{p_0}$ *zero entries, where* p_0 *is the probability that* J *randomly chosen columns of* H *with replacement do not have rank* N*.*

The proof can be found in Appendix A.4.3 on page page 148.

First note, that the result is independent of the (proper) loss function and the regularization constant $C > 0$. Our bound only depends on p_0 determined by the training data and the hypothesis set. The probability p_0 can be interpreted as a measure how volatile the solution with respect to changes of the hypothesis set is. For *non-volatile* matrices H, p_0 converges very fast to zero when increasing J. This is for instance the case, if vectors $H_{*,j}$ ($j = 1, \ldots, J$), interpreted as points in an N dimensional space are in *general position*, i.e. if any N points span the full N dimensional space (for instance random points). For the case $J \gg N$, one can show[4] that $p_0 \approx \left(\frac{N}{J-N}\right)^J$. Thus, there is about a fraction of $\frac{N}{J-N}$ coefficients that are zero. Hence, for large J there are most coefficients non-zero and the solution is not sparse.

Concave regularization operators have been used to generate particularly sparse solutions [cf. Bradley et al., 1998, Bradley and Mangasarian, 1998]. For completeness we show the following:

Proposition 5.4 (Sparseness of concave regularization). *Let* H*,* S *and* g *as in Proposition 5.3,* $C \in [0, \infty)$ *and* $\mathbf{P}_j$ *be concave functions as described above. Then there exists a solution* $\hat{\boldsymbol{\alpha}}^*$ *of (5.5) that has at most* N *non-zero entries. Moreover, if all regularization functions are strictly concave, then any solution has at most* N *non-zero entries.*

The proof can be found in Appendix A.4.4 on page page 149.

4. Many thanks to Wolfgang Thumser, Fred Galvin and Ortwin Gasper for help with this question.

We can conclude that the shape of the regularizer determines whether there exists an optimal hypothesis coefficient vector that is sparse or not. However, (strictly) concave minimization is considered to be a hard problem (but see also Bradley and Mangasarian [1998]). In practice one is only able to solve *convex* optimization problems exactly. Fortunately there is one function that is *concave and convex* – the linear function. By employing linear regularization (also called ℓ_1-norm regularization), one can obtain sparse solutions.

5.2.2 Convex Sparseness Regularization

The ℓ_1-norm regularization is in fact frequently used in sparsity favoring approaches, e.g. basis pursuit [Chen et al., 1995], parsimonious least norm approximation [Bradley et al., 1998] and many other techniques [e.g. Tibshirani, 1994, Bennett, 1999]. Roughly speaking, a reason for the induced sparseness is the fact that vectors far from the coordinate axes are "larger" with respect to the ℓ_1-norm than with respect to p-norms with $p > 1$. For example, consider the vectors $(1, 0)$ and $(1/\sqrt{2}, 1/\sqrt{2})$. For the two norm, $\|(1, 0)\|_2 = \|(1/\sqrt{2}, 1/\sqrt{2})\|_2 = 1$, but for the ℓ_1-norm, $1 = \|(1, 0)\|_1 < \|(1/\sqrt{2}, 1/\sqrt{2})\|_2 = \sqrt{2}$. For strictly concave regularizers this effect is even stronger.

For $\mathbf{P}[f_{\hat{\boldsymbol{\alpha}}}] = \|\hat{\boldsymbol{\alpha}}\|_1$ (using $\hat{\boldsymbol{\alpha}} \geq 0$), the minimization of (5.3) can be done by solving[5]

$$\begin{array}{ll} \min\limits_{\hat{\boldsymbol{\alpha}}, b, \boldsymbol{\xi}} & C \sum\limits_{j=1}^{J} \hat{\alpha}_j + \sum\limits_{n=1}^{N} \mathrm{g}(\xi_n) \\ \text{with} & y_n - H\hat{\boldsymbol{\alpha}} - b \;\; = \xi_n \qquad n = 1, \ldots, N \\ & \hat{\boldsymbol{\alpha}} \in \mathbb{R}_+^J, \boldsymbol{\xi} \in \mathbb{R}^N, b \in \mathbb{R}. \end{array} \tag{5.14}$$

The solution of (5.14) is always a vertex solution (or can be expressed as such) and tends to be very sparse. In the analysis above it is only shown that the number of non-zero coefficients is smaller than N. In the following discussion we give some more reasons why the solution is sparse and how sparse it is.

Let us analyze the the first order optimality conditions for $\hat{\boldsymbol{\alpha}}$. Substituting $\xi_n \equiv y_n - f_{\hat{\boldsymbol{\alpha}}, b}(\boldsymbol{x}_n)$ in (5.14), one arrives at the Lagrangian $L(\hat{\boldsymbol{\alpha}}, b, \boldsymbol{\gamma}) = C \sum_{j=1}^{J} \hat{\alpha}_j + \sum_{n=1}^{N} \mathrm{g}\left(y_n - H\hat{\boldsymbol{\alpha}} - b\right) - \boldsymbol{\gamma}^\top \hat{\boldsymbol{\alpha}}$, where the $\boldsymbol{\gamma}$'s are the Lagrange multipliers for the constraints $\hat{\boldsymbol{\alpha}} \geq \mathbf{0}$ with $\boldsymbol{\gamma} \in \mathbb{R}_+^J$. At optimal $\hat{\boldsymbol{\alpha}}$, we have

$$\frac{\partial L}{\partial \hat{\alpha}_j} = C - \sum_{n=1}^{N} h_j(\boldsymbol{x}_n)\, \mathrm{g}'\left(y_n - H\hat{\boldsymbol{\alpha}} - b\right) - \boldsymbol{\gamma} = \mathbf{0}. \tag{5.15}$$

From the KKT conditions $\hat{\alpha}_j \gamma_j = 0$ we get:

Lemma 5.1. *if* $\langle h_j, \mathrm{g}'(\boldsymbol{\xi}) \rangle < C$, *then* $\hat{\alpha}_j = 0$.

Roughly speaking, if the outputs of a hypothesis on the training set are not enough correlated with the $\mathrm{g}'(\cdot)$-transformed residue vector $\boldsymbol{\xi}$, then its weight is zero. For some fixed[6] $\boldsymbol{\xi}$, increasing C would lead to a sparser vector $\hat{\boldsymbol{\alpha}}$. For $C = \infty$, all elements of $\hat{\boldsymbol{\alpha}}$ are zero and for $C \to 0$, potentially any element could be non-zero (cf. Proposition 5.3).

5. For simplicity we have absorbed $\frac{1}{N}$ into the loss function.
6. Note that $\boldsymbol{\xi}$ usually changes, if C is changed. This construction is done for illustration only.

Also, we observe that the objective of the dual optimization problem (cf. Section 5.3.1 and Appendix A.4.6) includes terms $\mathrm{g}_*(\mathrm{g}'(\xi_n))$ that are minimized, where g_* is a convex function (the conjugate of g, cf. Section 5.3.1) with $\mathrm{g}_*(0) = 0$. Hence, at optimality the $\mathrm{g}'(\xi_n)$'s are likely to be small and therefore there are just a few functions h_j satisfying $\langle h_j, \mathrm{g}'(\boldsymbol{\xi})\rangle \geq C$, which may then have non-zero coefficients.

5.2.3 Boosting vs. SVMs again

Let us come back to discuss another relation of SVMs and boosting-like algorithms (see also Section 1.3.4). It is a common folklore statement and has already been pointed out e.g. in Schapire et al. [1998], Freund and Schapire [1999b] that boosting and SVMs are "essentially the same" except for the way they measure the margin or the way they optimize their weight vector: SVMs use the ℓ_2-norm and boosting employs an ℓ_1-norm. One might think that this solely influences the imposed regularization.

For SVMs and kernel methods in general it was shown that the optimal hyperplane with normal vector $\boldsymbol{w}^*$ can always be expressed as a linear combination of the examples in feature space $\mathbb{F}_\Phi$ implied by the mapping Φ:

Theorem 5.2 (Representer Theorem, Wahba [1990], Schölkopf et al. [2000]). *Let* g *be an arbitrary loss function and* r *be a strictly monotonically increasing function on* $[0, \infty)$.

$$\min_{\boldsymbol{w}\in\mathbb{F}_\Phi} \sum_{n=1}^{N} \mathrm{g}(y_n, \langle \boldsymbol{w}, \Phi(\boldsymbol{x}_n)\rangle) + r(\|\boldsymbol{w}\|_2) \tag{5.16}$$

Then for any N *examples* $(\boldsymbol{x}_1, y_1), \ldots, (x_N, y_N)$ *and any mapping* Φ, any solution $\boldsymbol{w}^*$ *of* (5.16) *can be expressed by* $\boldsymbol{w}^* = \sum_{n=1}^{N} \hat{\alpha}_n \Phi(\boldsymbol{x}_n)$ *for some* $\hat{\alpha}_n \in \mathbb{R}$, $n = 1, \ldots, N$.

In that way SVMs can be very efficient in dealing with possibly infinite dimensional feature spaces. However, SVMs need to use the ℓ_2-norm to implicitly compute scalar products in feature space with the help of the kernel trick. No other norm can be expressed in terms of scalar products.

Boosting, in contrast, performs the computation explicitely in feature space. This is well-known to be prohibitive if the solution $\boldsymbol{w}$ is not sparse, as the feature space might be very high or even infinite dimensional, depending on the size of the base hypothesis set H. We have the following result:

Theorem 5.3. *Let* g *be an arbitrary loss function and* r *be a monotonically increasing function on* $[0, \infty)$.

$$\inf_{f_{\boldsymbol{w}}\in\mathsf{F}} \sum_{n=1}^{N} \mathrm{g}(y_n, f_{\boldsymbol{w}}(\boldsymbol{x}_n)) + r(\|\boldsymbol{w}\|_1), \tag{5.17}$$

where F *is the set of functions that are linear combinations of a finite number of functions of* H. *Then for any* N *examples* $(\boldsymbol{x}_1, y_1), \ldots, (x_N, y_N)$ *and any compact hypothesis set* H, *there* exist *a solution* f^* *of* (5.17) *that can be expressed by*

$$f^*(\boldsymbol{x}) = \sum_{n=1}^{N} \hat{\alpha}_n h_n(\boldsymbol{x}).$$

The above theorem is a direct consequence of Proposition 5.4 for the finite case and an implication of the later presented Theorem 5.5. The additional possibly non-linear transformation of the regularization term does not matter, since it can be reduced to the case of multiplying with a properly chosen constant.[7]

Thus, the sparseness property turns out to be very useful for ensemble learning, as one only needs to combine few hypotheses to reach an optimal solution. Otherwise, one would need to consider too many hypotheses from H to compute the prediction of an optimally combined hypothesis. In particular, if H is huge or even infinite, this would be impossible. Note that there is one important difference between Theorems 5.2 and 5.3: for kernel methods we have that any solution can be expressed as a small linear combination, whereas in boosting one has that there exists a sparse solution. If the regularizer is strictly concave, then any solution would be sparse, but the optimization problem would be hard to solve.

Hence, both algorithms work in very high-dimensional feature spaces, but they differ, however, in how they deal with the algorithmic problems that this can cause. They lead to sparse solutions, either in sample (coefficient) space (SVMs), or in feature space (boosting), and both methods are adapted to algorithmically exploit the form of sparsity they produce.

The sparseness property opens the door to develop efficient ensemble algorithms for solving the regression problem [cf. Section 5.4 and Rätsch et al., 2002]. Hence, for the rest of this chapter we consider the ℓ_1-norm regularization operator only and exploit its properties.[8]

5.3 Infinite Hypothesis Sets and Semi-Infinite Programming

In this section we consider infinite hypothesis sets. We start with the dual problem of (5.14), which has a finite number of constraints. Then we extend our view to infinite hypothesis sets leading to an infinite number of constraints. For this case we state results showing that they can be reduced to a small finite number of active constraints. Finally we state the primal regression problem.

5.3.1 Dual formulation

Let $\mathrm{g}(\cdot) : \mathbb{R} \to \mathbb{R}_+$ be a convex loss function with $\mathrm{g}(0) = 0$, $\mathrm{g}(z) \geq 0$ for $z \in \mathbb{R}$. Then the dual optimization problem of (5.14) for finite H is:

$$\begin{array}{lll} \max\limits_{\mathbf{d}\in\mathbb{R}^n} & Y^\top \mathbf{d} - \sum_{n=1}^N \mathrm{g}_*(d_n) & \\ \text{with} & \sum_{n=1}^N h_j(\boldsymbol{x}_n) d_n & \leq C \quad \text{for } j = 1, \ldots, J \\ & \sum_{n=1}^N d_n & = 0. \end{array} \tag{5.18}$$

7. This constant is given by the derivative of r at the solution; see Smola [1998], Schölkopf et al. [2000] for a more detailed discussion of this issue.

8. Note that we have already used the ℓ_1-norm regularization also for the large margin classification algorithms.

The derivation for differentiable loss functions can be found in Appendix A.4.6. In (5.18), there is one equality constraint and $J = |\mathsf{H}|$ inequality constraints, one for each hypothesis $h \in \mathsf{H}$. The function g_* is the conjugate function of g. If g is differentiable this is the Legendre-conjugate function [e.g. Rockafellar, 1970, page 256]:

$$\mathrm{g}_*(z) = z(\mathrm{g}')^{-1}(z) - \mathrm{g}((\mathrm{g}')^{-1}(z)). \tag{5.19}$$

The conjugate function g_* of an arbitrary convex function g can be defined as [e.g. Rockafellar, 1970, Corollary 12.2.2]

$$\mathrm{g}_*(y) = \sup_{x \in \mathrm{ri}(\mathrm{dom}\,\mathrm{g})} \langle x, y \rangle - \mathrm{g}(x). \tag{5.20}$$

At optimality, the quantity d_n defines an error residual: by differentiating the Lagrangian of (5.18) (cf. (A.20)) with respect to $\mathbf{d}$, one gets $\mathrm{g}'_*(d_n) = y_n - f_{\hat{\boldsymbol{\alpha}},b}(\boldsymbol{x}_n)$. Thus, if the point is underestimated, $f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}_n) < y_n$, then $\mathrm{g}'_*(d_n) \geq 0$. Likewise, if the point is overestimated, $f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}_n) > y_n$, then $\mathrm{g}'_*(d_n) \leq 0$.[9] The magnitude of d_n reflects the sensitivity of the objective. The larger the change in error, the larger d_n. The quantity in the constraints $\sum_n h_j(\boldsymbol{x}_n) d_n$ reflects how well the hypothesis h_j addressed the residual errors. If $\sum_n h_j(\boldsymbol{x}_n) d_n$ is positive and large in size then the hypothesis will be likely to improve the ensemble (cf. Lemma 5.1).

Furthermore, its easy to show that the Lagrange multipliers are bounded:

Lemma 5.2. *Let* $\mathrm{g} : \mathbb{R} \to \mathbb{R}_+$ *be a convex and differentiable function with* $\mathrm{dom}\,\mathrm{g} = \mathbb{R}$*. For any solution* $\mathbf{d}$ *of (5.18) holds* $\|\mathbf{d}\|_\infty \leq \max(|\,\mathrm{g}'(\|Y\|_1)|, |\,\mathrm{g}'(-\|Y\|_1)|) =: d_{\max} < \infty$.

The proof can be found in Appendix A.4.7 on page 151. By Lemma 5.2 one could in principle introduce additional constraints $|d_n| \leq d_{\max}$ to (5.18) without changing the solution.

In the finite case it is straightforward to show the strong duality:

Lemma 5.3. *Let* g *be a convex (loss) function with* $0 \leq \mathrm{g}(z) < \infty$ *for* $z < \infty$*. If* $\|Y\|_\infty < \infty$*, then there exist optimal solutions to (5.14) and (5.18), such that the objective values are the same, i.e. there is no duality gap (strong duality holds).*

The proof can be found in Appendix A.4.8 on page 152. Its generalization to the infinite case is more difficult and is considered in the next sections.

5.3.2 The Dual Problem for Infinite Hypothesis Sets

Consider now the case where there is an infinite set of possible hypotheses H. Say we select any finite subset H_1 of H, then the primal and dual regression MPs on H_1 are well defined. Now say we increase the subset size and define $\mathsf{H}_2 \supset \mathsf{H}_1$ of H. What is the relationship between the optimal ensembles created on the two subsets? A solution of the smaller H_1-problem is always primal feasible for the larger H_2-MP. If the H_1-solution is

9. If $\mathrm{g}(z) > 0$ for $z \neq 0$, this simplifies to $d_n \geq 0$ and $d_n \leq 0$, respectively.

dual feasible for the larger H_2-MP, then the solution is also optimal for the H_2-problem. So dual feasibility is the key issue. Let us define the maximal constraint violation for some fixed $\mathbf{d}$:

$$h_{\mathbf{d}} := \operatorname*{argmax}_{h \in \mathsf{H}} \sum_{n=1}^{N} h(\boldsymbol{x}_n) d_n, \tag{5.21}$$

where it is sufficient to assume that the set

$$\mathsf{H}(X) := \left\{ \begin{bmatrix} h(\boldsymbol{x}_1) \\ \vdots \\ h(\boldsymbol{x}_N) \end{bmatrix} \;\middle|\; h \in \mathsf{H} \right\} \tag{5.22}$$

is compact.[10] Otherwise the maximum in (5.21) might not exist. Eq. (5.21) is indeed very similar to the δ-optimality condition with $\delta = 1$ for base learners as used in classification. However, here we use it for a different purpose – to extend the dual problem to the infinite case.

If $\sum_{n=1}^{N} h_{\mathbf{d}}(\boldsymbol{x}_n) d_n > C$, then dual feasibility is violated; $h_{\mathbf{d}}$ is a good hypothesis that should be added to the ensemble, and the solution may not be optimal (cf. Lemma 5.1).

By thinking of $h_{\boldsymbol{p}}$ as a function of $\mathbf{d}$ as in (5.21), we can extend the dual problem (5.18) to the infinite hypotheses case. The set of dual feasible values of $\mathbf{d}$ is equivalent to the following compact polyhedron:

$$\mathcal{P} = \left\{ \boldsymbol{p} \;\middle|\; \sum_{n=1}^{N} p_n = 0,\ |p_n| \le d_{\max}, n = 1, \ldots, N \right\}, \tag{5.23}$$

where $d_{\max}$ is given in Lemma 5.2. The *dual semi-infinite-regression problem* (SIP) is

$$\begin{aligned} \Omega(D_R) = \max_{\mathbf{d}} \quad & \sum_{n=1}^{N} y_n d_n - \sum_{n=1}^{N} \mathrm{g}_*(d_n) \\ \text{with} \quad & \textstyle\sum_{n=1}^{N} h_{\boldsymbol{p}}(\boldsymbol{x}_n) d_n \le C, \quad \forall \boldsymbol{p} \in \mathcal{P} \\ & \textstyle\sum_{n=1}^{N} d_n = 0 \\ & -d_{\max} \le d_n \le d_{\max},\ n = 1, \ldots, N, \end{aligned} \tag{5.24}$$

which is an example out of a class of problems that has been extensively studied in the mathematical programming literature. The problem is called semi-infinite because it has an infinite number of constraints and a finite number of variables. The set $\mathcal{P}$ is known as the index set. If the set of hypotheses (producible by the base learner) is finite, then the problem is exactly equivalent to regression problem (5.18).

Remark 5.4. *Let $\mathsf{H}_X = \{h_{\boldsymbol{p}} \mid \boldsymbol{p} \in \mathcal{P}\}$. Note that $\mathsf{H}_X \subseteq \mathsf{H}$, but by (5.21) it is obvious that (5.24) with all constraints produced by $h \in \mathsf{H}$ (instead of H_X) yields the same solution.*

We will establish several facts about this semi-infinite programming problem using the results for general semi-infinite programs summarized in the excellent review paper [Hettich and Kortanek, 1993]. To simplify the presentation, we straightforwardly extended

10. For simplicity, we say that H is compact if $\mathsf{H}(X)$ is compact.

the results in Hettich and Kortanek [1993] to the case of SIP with an additional set of finite linear constraints. The results presented can be easily derived from Hettich and Kortanek [1993] through a change in notation and by increasing the index set to include the additional finite set of traditional linear constraints. To be consistent with our derivation of the SIP-regression problem, we will refer to the problem with infinitely many constraints as the dual problem and the problem with infinitely many variables as the primal problem.

We define the *generic dual SIP* D_G as

$$\begin{aligned} \Omega(D_G) = \max_{\boldsymbol{z}\in\mathbb{R}^N} \; & -\mathrm{G}_*(\boldsymbol{z}) \\ \text{with } & \langle \boldsymbol{u}(\boldsymbol{p}), \boldsymbol{z}\rangle \le v(\boldsymbol{p}) \quad \forall \boldsymbol{p} \in \mathcal{P}_G \\ & Q\boldsymbol{z} \le \boldsymbol{q}, \end{aligned} \tag{5.25}$$

where $\mathrm{G}_*(\cdot)$ is a convex function from $\mathbb{R}^N$ to $\mathbb{R}$, $Q \in \mathbb{R}^{r\times N}$, $\boldsymbol{q} \in \mathbb{R}^r$, $\mathcal{P}_G$ is a compact subset of $\mathbb{R}^N$, $\boldsymbol{u}(\cdot)$ is a function from $\mathcal{P}_G$ to $\mathbb{R}^N$, and $v(\cdot)$ is a function from $\mathcal{P}_g$ to $\mathbb{R}$. We will make the additional assumption that the problem is always feasible and that the feasible region is compact. Then the maximum value is always obtained since we are maximizing a continuous function over a compact set.

Ideally, we would like the solution of a finite problem to correspond to the optimal solution of the semi-infinite problem. We now define a sufficient condition for the existence of a finite mathematical program whose optimal solution also solves the semi-infinite program. We will denote the generic dual SIP restricted to a finite subset $\mathcal{P}_N = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_N\} \subset \mathcal{P}_G$ as $D_G(\mathcal{P}_N)$ and its objective at optimality by $\Omega(D_G(\mathcal{P}_N))$. This is a standard mathematical program since it has a finite number of constraints.

The first theorem gives sufficient conditions for the optimal solution of a generic dual SIP to be equivalent to the solution of a finite program (Borwein [1983b], Theorem 2.1(c); Hettich and Kortanek [1993], Theorem 4.2):

Theorem 5.5 (Sufficient condition for finite solution, Hettich and Kortanek [1993]). *Assume the following Slater condition holds: For every set of $N+1$ points $\hat{\boldsymbol{p}}_0, \hat{\boldsymbol{p}}_1, \ldots, \hat{\boldsymbol{p}}_N$, there exists $\hat{\boldsymbol{z}}$ such that $\langle \boldsymbol{u}(\hat{\boldsymbol{p}}_n), \hat{\boldsymbol{z}}\rangle < v(\hat{\boldsymbol{p}}_n)$, $n = 0, \ldots, N$, and $Q\hat{\boldsymbol{z}} < \boldsymbol{q}$. Then there exists $\mathcal{P}_N = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_N\} \subset \mathcal{P}_G$ such that*

1. $\Omega(D_G) = \Omega(D_G(\mathcal{P}_N))$*;*
2. *There exist multipliers $\mu_n \ge 0$, $n = 1, \ldots, N$, such that*

$$\Omega(D_G) = \sup_{\boldsymbol{z}\in\mathbb{R}^N} \left\{ -\mathrm{G}_*(\boldsymbol{z}) - \sum_{n=1}^{N} \mu_n(\boldsymbol{u}(\boldsymbol{p}_n) - v(\boldsymbol{p}_n)) \;\middle|\; Q\boldsymbol{z} \le \boldsymbol{q} \right\}. \tag{5.26}$$

The proof uses a Helly-type theorem [cf. Borwein, 1983a]. This result immediately applies to dual SIP regression:

Corollary 5.1 (Finite solution of regression ensemble, Rätsch et al. [2002]). *Assume $\mathsf{H}(X)$ is compact. For Problem $\Omega(D_G)$ as in (5.24) with $C > 0$, there exists $\mathcal{P}_N = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_N\} \subset \mathcal{P}$ such that $\Omega(D_R) = \Omega(D_R(\mathcal{P}_N))$.*

The proof can be found in Appendix A.4.5 on page 150.

The significance of this result is that there exists an optimal ensemble that consists of at most N hypotheses where N is the number of training examples. Although, we have shown this for finite hypothesis sets already in Section 5.2.2, it is satisfying that this property does not change if the set of possible hypotheses is infinite.

5.3.3 Primal Regression SIP

Next we look at the corresponding primal problem for the semi-infinite case. We would like our semi-infinite dual problem to be equivalent to a meaningful primal problem that simplifies to the original primal for the finite hypothesis case.

Let $M^+(\mathcal{P}_G)$ be the set of nonnegative Borel measures on $\mathcal{P}_G$. The subset

$$\mathbb{R}_+^{(\mathcal{P}_G)} := \{\mu \in M^+(\mathcal{P}_G) \mid \operatorname{supp}(\mu) \text{ finite}\} \tag{5.27}$$

denotes the set of nonnegative generalized finite sequences. The primal problem of the generic SIP as in (5.25) is [cf. Hettich and Kortanek, 1993]

$$\begin{aligned} \Omega(P_G) = \inf_{\mu,\boldsymbol{\xi},\boldsymbol{\kappa}} & \sum_{\boldsymbol{p}\in\mathcal{P}_G} v(\boldsymbol{p})\mu(\boldsymbol{p}) + \mathrm{G}(\boldsymbol{\xi}) - \langle \boldsymbol{\kappa}, \boldsymbol{q}\rangle \\ \text{with } & Q\boldsymbol{\kappa} - \sum_{\boldsymbol{p}\in\mathcal{P}_G} \boldsymbol{u}(\boldsymbol{p})\mu(\boldsymbol{p}) = \boldsymbol{\xi} \\ & \mu \in \mathbb{R}_+^{(\mathcal{P}_G)}, \boldsymbol{\kappa} \in \mathbb{R}_+^r, \boldsymbol{\xi} \in \mathbb{R}^N, \end{aligned} \tag{5.28}$$

where G is the convex conjugate function of G_* and $\boldsymbol{q}, \boldsymbol{u}, v$ are as in (5.25). From this general primal SIP one can straightforwardly derive the primal regression semi-infinite problems. For completeness we give the primal problem for (5.24) with convex loss function and ℓ_1-norm regularization:

$$\begin{aligned} \inf_{\hat{\boldsymbol{\alpha}},b,\boldsymbol{\xi}} \quad & C \sum_{\boldsymbol{p}\in\mathcal{P}_R} \hat{\alpha}_{\boldsymbol{p}} + \sum_{n=1}^{N} \mathrm{g}(\xi_n) \\ \text{with} \quad & y_n - b - \sum_{\boldsymbol{p}\in\mathcal{P}_R} \hat{\alpha}_{\boldsymbol{p}} h_{\boldsymbol{p}}(\boldsymbol{x}_n) \quad = \xi_n \quad \text{for } n = 1,\ldots,N \\ & \hat{\boldsymbol{\alpha}} \in \mathbb{R}_+^{(\mathcal{P}_R)}, \boldsymbol{\xi} \in \mathbb{R}^N, b \in \mathbb{R}, \end{aligned} \tag{5.29}$$

where we have used (5.28), while setting $\boldsymbol{q} = \left[\begin{smallmatrix}0\\0\end{smallmatrix}\right]$ and $Q = \left[\begin{smallmatrix}1,\ldots,1\\-1,\ldots,-1\end{smallmatrix}\right]$ and $\mathbb{R}_+^{(\mathcal{P}_R)}$ is as defined in (5.27). For convenience we write $\hat{\alpha}_{\boldsymbol{p}}$ instead of $\hat{\alpha}(\boldsymbol{p})$.

Analogously we can derive the semi-infinite version of (5.12) using the ϵ-insensitive loss (see also (5.33)):

$$\begin{aligned} \inf_{\hat{\boldsymbol{\alpha}},b,\boldsymbol{\xi},\boldsymbol{\xi}^*,\epsilon} \quad & C \sum_{\boldsymbol{p}\in\mathcal{P}_R} \hat{\alpha}_{\boldsymbol{p}} + \sum_{n=1}^{N} (\xi_n + \xi_n^*) + \nu\epsilon \\ \text{with} \quad & y_n - b - \sum_{\boldsymbol{p}\in\mathcal{P}_R} \hat{\alpha}_{\boldsymbol{p}} h_{\boldsymbol{p}}(\boldsymbol{x}_n) \quad \le \epsilon + \xi_n \qquad \text{for } n = 1,\ldots,N \\ & b + \sum_{\boldsymbol{p}\in\mathcal{P}_R} \hat{\alpha}_{\boldsymbol{p}} h_{\boldsymbol{p}}(\boldsymbol{x}_n) - y_n \quad \le \epsilon + \xi_n \qquad \text{for } n = 1,\ldots,N \\ & \hat{\boldsymbol{\alpha}} \in \mathbb{R}_+^{(\mathcal{P}_R)}, \boldsymbol{\xi}, \boldsymbol{\xi}^* \in \mathbb{R}_+^N, b \in \mathbb{R}, \epsilon \in \mathbb{R}_+, \end{aligned} \tag{5.30}$$

where we only needed to double the constraints and had to introduce a new variable ϵ.

In our finite regression problems, the optimal objective values of the primal and dual problems are always equal (cf. Lemma 5.3). This might not always be true for the semi-infinite case. *Weak duality* always holds, that is, $\Omega(P_G) \leq \Omega(D_G)$. We must ensure that there is no duality gap, i.e., that $\Omega(P_G) = \Omega(D_G)$. Furthermore, one needs that the minimum exists (and not only the infimum, cf. (5.28)).

For simplicity we will restrict our attention in the following to (piecewise) linear and convex loss functions, like the ϵ-insensitive loss. For non-linear loss functions, however, most of these result can be transfered easily [cf. Borwein, 1983b, Hettich and Kortanek, 1993]. From Hettich and Kortanek [1993] (Theorem 6.5) we have the following:

Theorem 5.6 (Sufficient conditions for no duality gap, Hettich and Kortanek [1993]). *Let the convex cone*

$$\begin{aligned} M_{N+1} &= co\left(\left\{ \begin{pmatrix} \boldsymbol{u}(\boldsymbol{p}) \\ v(\boldsymbol{p}) \end{pmatrix} \;\middle|\; \boldsymbol{p} \in \mathcal{P}_G \right\}\right) \subseteq \mathbb{R}^{N+1} \\ &= \left\{ \sum_{\boldsymbol{p}\in\mathcal{P}_G} \lambda(\boldsymbol{p}) \begin{pmatrix} \boldsymbol{u}(\boldsymbol{p}) \\ v(\boldsymbol{p}) \end{pmatrix} \;\middle|\; \lambda \in \mathbb{R}_+^{(\mathcal{P}_G)} \right\} \subseteq \mathbb{R}^{N+1}. \end{aligned} \tag{5.31}$$

be closed, then $\Omega(P_G) = \Omega(D_G)$ and primal minimum is attained.

Let us consider regression problems with piecewise-linear, convex loss functions. Then the resulting problem can be expressed as a linear program. We use $\boldsymbol{u}(\boldsymbol{p}) = h_{\boldsymbol{p}}(X) = [h_{\boldsymbol{p}}(\boldsymbol{x}_1), \ldots, h_{\boldsymbol{p}}(\boldsymbol{x}_N)]^\top$ and $v(\boldsymbol{p}) = C$. Using a result found in Glashoff and Gustafson [1978], one can easily show the following:

Corollary 5.2. *Assume each y_n is finite, $\mathsf{H}(X)$ is compact and $C > 0$. Consider the regression problems (5.29) and (5.25), called P_R and D_R, respectively. If g is a piecewise-linear, convex loss function, then (i) $\Omega(P_R) = \Omega(D_R)$, (ii) primal minimum and (iii) the dual maximum are attained.*

The proof can be found in Appendix A.4.9 on page 152. The corollary requires that $\mathsf{H}(X)$ is compact. This is the case if

- the (vector-valued) function $h_{\boldsymbol{p}}(X)$ as defined in (5.21) is continuous with respect to $\boldsymbol{p}$, or
- the set of distinct hypotheses is finite, i.e. $\{h(X) \mid h = \mathfrak{L}(X, \boldsymbol{p}),\ \boldsymbol{p} \in \mathcal{P}\}$ is finite.

These two conditions are sufficient to cover all the base hypotheses sets considered in this work except decision trees (other, more refined conditions are possible). For decision trees one has the problem, that $h_{\boldsymbol{p}}(X)$ might change drastically, when $\boldsymbol{p}$ changes only very little, since another coordinate might be chosen somewhere in the tree. However, there is a simple trick to avoid this problem: Roughly speaking, at each point with discontinuity $\hat{\boldsymbol{p}}$, one adds all hypotheses to H that are limit points of $\{h_{\boldsymbol{p}^s}\}$, where $\{\boldsymbol{p}^s\}_{s=1}^{\infty}$ is an arbitrary sequence converging to $\hat{\boldsymbol{p}}$. If the base learner is asked to return a hypothesis at some discontinuity, then it might at random pick one of these hypotheses. A similar strategy could be used for other open, bounded hypothesis sets.

5.4 Optimization Algorithms

In this section we develop three regression algorithms, where we focus to apply them for infinite hypothesis sets. The first algorithm is for the ε-insensitive loss and based on the Column Generation (CG) technique discussed for classification in Section 4.3.5 [cf. Demiriz et al., 2001b]. The second algorithm extends the leveraging scheme for arbitrary strictly convex loss functions (cf. Section 3.1 and Section 4.3.3) to infinite hypothesis sets. It is easier to implement than the CG algorithm, but requires *strictly* convex loss functions. Therefore, we outline a third algorithm, that exemplary (on the ε-insensitive loss) shows how barrier techniques can be used to generate sequences of strictly convex loss function to approximate the convex loss. Then the above-mentioned leveraging approach can be used to minimize the barrier objective.

As in classification, we have to assume properties on the base learner to be able to prove convergence. Here we will require τ_1-optimality similar to Definition 3.4:

Definition 5.1 (τ_1-Optimality). *Let S be a sample, H complementation closed and compact hypothesis space. Furthermore, let $\tau_1 : \mathbb{R}_+ \to \mathbb{R}_+$ be a strictly monotonically increasing and continuous function with $\tau_1(0) = 0$. A base learning algorithm $\mathfrak{L}$ is called* τ_1-optimal *for H, S and C, if it always returns a hypothesis $h \in \mathsf{H}$ that satisfies*

$$\gamma_h - C \geq \tau\left(\max_{h' \in \mathsf{H}} (\gamma_{h'} - C)\right) \tag{5.32}$$

for any weighting $\mathbf{d} \in \mathbb{R}^N$, where $\gamma_h = \sum_{n=1}^{N} d_n h(\boldsymbol{x}_n)$.

Note that the maximum in (5.32) always exists, since we assume that H is compact. This is an assumption always inherent in the following without explicitly being stated. Since the hypothesis set is infinite, we will state asymptotic convergence results only. Hence, the τ_1-optimality is sufficient for our purpose (see also Section 3.2).

The following three algorithms are presented in an order to make the presentation easiest. We therefore first consider the ε-insensitive loss, then general loss functions and then we come back to the ε-insensitive loss.

5.4.1 Column Generation Method for the ε-insensitive Loss

In this section we consider a column generation linear programming approach for the ε-insensitive loss. The dual of (5.14) for this loss and infinite hypothesis sets can be rewritten:

$$\begin{array}{lll} \max\limits_{\mathbf{d}^+,\mathbf{d}^-} & \sum\limits_{n=1}^{N} y_n(d_n^+ - d_n^-) & \\ \text{with} & \sum_n d_n^+ - d_n^- & = 0 \\ & \sum_n d_n^+ + d_n^- & \leq \nu \\ & \sum_n h_{\boldsymbol{p}}(\boldsymbol{x}_n)(d_n^+ - d_n^-) & \leq C \quad \boldsymbol{p} \in \mathcal{P} \\ & 0 \leq d_n^+, d_n^- \leq 1/N & \quad n = 1, \ldots, N, \end{array} \tag{5.33}$$

The basic idea of column generation is to construct an *optimal* ensemble for a *restricted subset* of the hypothesis space [Nash and Sofer, 1996]. The linear problem (5.33) is solved for a finite subset of hypotheses only. It is called the *restricted master problem*. Then the base learner is called to generate a hypothesis $h_t = \mathcal{L}(X, \mathbf{d})$ where $\mathbf{d} = \mathbf{d}^+ - \mathbf{d}^-$. Assuming the base learner is τ_1-optimal, holds: if $\sum_n h_t(\boldsymbol{x}_n)(\mathbf{d}_n^+ - \mathbf{d}_n^-) \leq C$, then the current ensemble is optimal as all constraints are fulfilled. If not, the hypothesis is added to the problem, which is then re-solved. This corresponds to generating a *column* in the primal LP (5.12) or SILP (5.30) or a *row* of the dual LP or SILP (5.33). The resulting algorithm is outlined in Algorithm 5.8. Here we use $y_{\max} < \infty$, which is satisfied since H is compact.

Algorithm 5.8 The CG-Regression algorithm.

1. **Input:** N examples $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, Number of iterations T, $y_{\max} < \infty$
   Tube parameter $\nu \in (\frac{1}{N}, 1)$, Regularization constant C
2. **Do for** $t = 1, \ldots, T$
   (a) Let $(\mathbf{d}^{(t),+}, \mathbf{d}^{(t),-})$ be the solution of (5.33) using $t-1$ hypotheses
   (b) Call weak learner with weighted sample set $\{S, \mathbf{d}^{(t)}\}$ and obtain hypothesis $h_t : \boldsymbol{x} \mapsto [-y_{\max}, y_{\max}]$
3. **Break if** $\sum_{n=1}^N \left(d_n^{(t),+} - d_n^{(t),-}\right) h_t(\boldsymbol{x}_n) \leq C$
4. Let $(\hat{\boldsymbol{\alpha}}, b)$ be the dual solution to $\left(\mathbf{d}^{(t),+}, \mathbf{d}^{(t),-}\right)$, i.e. a solution to (5.12) with $\mathbf{P}[f_{\hat{\boldsymbol{\alpha}}}] = \|\hat{\boldsymbol{\alpha}}\|_1$
5. **Output:** Final hypothesis $f = b + \sum_{q=1}^{t-1} \hat{\alpha}_q h_q$

Algorithm 5.8 is a special case of the set of SILP algorithms known as *exchange methods*. These methods are known to converge. Clearly if the set of hypotheses is finite, then the method will converge in a finite number of iterations since no constraints are ever dropped. But one can also prove that it converges for the semi-infinite case:

Theorem 5.7 (Convergence of Algorithm 5.8). *Assume the base learner is τ_1 optimal. Then Algorithm 5.8 stops after a finite number of steps with a solution to the dual regression SILP (5.33) or the sequence of intermediate solutions $(\mathbf{d}^{(t),+}, \mathbf{d}^{(t),-})$ has at least one accumulation point and each of these solves (5.33).*

A proof adapted from Theorem 7.2 in Hettich and Kortanek [1993] can be found in Appendix A.4.10 on page 152.

In practice, we found that the column generation algorithm stops at an optimal solution in a small number of iterations for both LP and SILP regression problems. Note that this theorem is assumed to hold for a more general set of exchange methods than Algorithm 5.8 [Hettich and Kortanek, 1993]. For example, it is possible to add or drop multiple constraints at each iteration.

Clearly, since there is no duality gap as H is assumed to be compact, one has also solved the primal optimization problem (cf. Theorem 5.6). It holds:

Corollary 5.3. *Under the conditions of Theorem 5.7, a dual solution* $(\hat{\boldsymbol{\alpha}}, b)$ *(with finite support) to any limit point of Algorithm 5.8 (cf. step 4) solves (5.30).*

Algorithm 5.8 requires that one solves a linear programming problem *exactly* in each iteration. Some linear programming packages such as CPLEX [CPL, 1994] have a so-called *hot-start* feature to add additional constraints to a LP and then to re-solve the problem quite efficiently. However, this particular feature is applicable for the ε-insensitive loss only. For other loss functions one has to solve the non-linear optimization problem in each iteration exactly, which can be very costly.

5.4.2 A Regularized Leveraging Approach

In this section we consider the case of strictly convex loss functions, where one only needs to solve a optimization problem approximately (e.g. using the coordinate descent techniques used earlier). This exploits the fact that it is indeed not necessary to find an exact minimizer of the approximate problem – the restricted master problem. As a result of our analysis, one can trade-off the computational effort for the base learner with the effort for finding approximate solutions of the non-linear optimization problem. This is useful, as combining different types of base learners will usually result in different requirements on the accuracy of the solution of the optimization problem. For instance, if the base learner is very "weak" one has "increase the complexity" through appropriately combining them. Then one would require more accurate solutions. On the other hand, if they are quite complex, the combined functions might already work well considerable well separately. Then a very exact solution might not be necessary.

The problem (5.14) for a restricted hypothesis set $\mathsf{H}^{(t)}$ with t elements can be written as:

$$\begin{aligned} &\min_{\hat{\boldsymbol{\alpha}},b} \; E(\hat{\boldsymbol{\alpha}}, b) := C \sum_{r=1}^{t} \hat{\alpha}_r + \sum_{n=1}^{N} \mathrm{g}(y_n - f_{\hat{\boldsymbol{\alpha}},b}(\boldsymbol{x}_n)), \\ &\text{with } \hat{\boldsymbol{\alpha}} \in \mathbb{R}_+^t, b \in \mathbb{R} \end{aligned} \tag{5.34}$$

where the constraints $y_n - f_{\hat{\boldsymbol{\alpha}},b}(\boldsymbol{x}_n) = \xi_n$ are fulfilled by plugging $y_n - f_{\hat{\boldsymbol{\alpha}},b}(\boldsymbol{x}_n)$ into the loss function directly. The dual problem of (5.34) is (5.18).

The important question to answer is, whether one can adapt Theorem 5.7 to the case where one does not solve the restricted master problem exactly as done in the last section. Let us first define the notion of a φ-minimizer for our problem. The first order optimality condition for $\hat{\alpha}_r$ is

$$\frac{\partial E(\hat{\boldsymbol{\alpha}}, b)}{\partial \hat{\alpha}_r} - \gamma_r = C - \sum_{n=1}^{N} h_r(\boldsymbol{x}_n)\, \mathrm{g}'(y_n - f_{\hat{\boldsymbol{\alpha}},b}(\boldsymbol{x}_n)) - \gamma_r = 0,$$

where $\gamma_r \geq 0$ is a Lagrange multiplier for the constraint $\hat{\alpha}_r \geq 0$. From the KKT conditions we have $\gamma_r \hat{\alpha}_r = 0$. Hence, if $\hat{\alpha}_r = 0$, the gradient can be positive at optimality. Otherwise it must be zero.

We therefore call a pair $(\hat{\boldsymbol{\alpha}}, b)$ a φ-minimizer for (5.34), if

1. $\frac{\partial E(\hat{\boldsymbol{\alpha}},b)}{\partial \hat{\alpha}_r} \geq -\varphi$, for all $\hat{\alpha}_r = 0$,
2. $\left|\frac{\partial E(\hat{\boldsymbol{\alpha}},b)}{\partial \hat{\alpha}_r}\right| \leq \varphi$ for all $\hat{\alpha}_r > 0$ and
3. $\left|\frac{\partial E(\hat{\boldsymbol{\alpha}},b)}{\partial b}\right| \leq \varphi$.

Such approximate solutions can be easily obtained by early stopping usual optimization tools. One could for instance use easy implementable coordinate descent methods and stop if the conditions above are satisfied. For small hypothesis sets such methods will converge quite fast to a φ-optimal solution (in $\mathcal{O}(\log(1/\varphi))$ iterations; cf. Section 3.3).

Let us now come to our algorithm, which is designed to take advantage of the tolerated inaccuracy. As the algorithm proceeds, one is more and more likely to have enough hypotheses selected. Hence, one reduces φ_t iteratively. How φ_t is reduced does not matter for our convergence result, however, in practice one has to find a good trade-off between the number of calls of the base learner and the effort to compute approximate solutions.

Our (conceptional) algorithm is outlined in Algorithm 5.9, and is in fact quite similar to the generic leveraging algorithm proposed in Section 3.1. Furthermore, it is related to the regularized leveraging algorithm proposed in Section 4.3.3, with the only difference that we require here φ_t-optimality. This could e.g. be implemented using the coordinate descent techniques (cf. Section 4.3.3 and Section 3.3), or by other optimization methods.

Algorithm 5.9 The Leveraging Algorithm for Regression in Infinite Hypothesis Sets

1. **Input:** N examples $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, Regul. constant C, $y_{\max} < \infty$, Number of iterations T, a sequence $\varphi_t \to 0$
2. **Do for** $t = 1, \ldots, T$
   (a) Let $(\hat{\boldsymbol{\alpha}}^{(t)}, b_t)$ be a φ_t-minimizer of (5.34) using $t-1$ hypotheses
   (b) Let $d_n^{(t)} = \mathrm{g}'(y_n - f_{\hat{\boldsymbol{\alpha}}^{(t)}, b_t})$
   (c) Call weak learner with weighted sample set $\{S, \mathbf{d}^{(t)}\}$ and obtain hypothesis $h_t : \boldsymbol{x} \mapsto [-y_{\max}, y_{\max}]$
3. **Break if** $\sum_{n=1}^{N} d_n^{(t)} h_t(\boldsymbol{x}_n) \leq C$ and $\varphi_t = 0$
4. **Output:** Final hypothesis $f = b + \sum_{q=1}^{t-1} \hat{\alpha}_q h_q$

Using the φ-optimality, we can show that, if $\varphi \to 0$, then it will eventually converge to the optimum:

Theorem 5.8. *Let S be a sample, $C > 0$, $\mathcal{L}$ be a τ_1-optimal base learner on some compact hypothesis set H, g be a strictly convex, symmetric and continously differentiable loss function, and φ_t be a sequence with $\varphi_t \to 0$. Furthermore, let $(\hat{\boldsymbol{\alpha}}^{(t)}, b_t)$ be the result of Algorithm 5.9 after t iterations. Assume there is no duality gap and the number of non-zeros in $\hat{\boldsymbol{\alpha}}^{(t)}$ stays bounded for $t \to \infty$. Then any limit point of $(\hat{\boldsymbol{\alpha}}_t, b_t)$ for $t \to \infty$ is an optimal solution of (5.29).*

The proof is given in Appendix A.4.11 on page 153.

There are some assumptions, which we explain in the following. First, we obviously need $C > 0$, as otherwise the regularization is not effective and we lose some desirable properties. Furthermore, the loss has to be *strictly* convex. This turned out to be necessary to bound the deviation of the approximate dual variables from the optimal solution. Finally we need to assume, that the φ-minimizer has asymptotically a finite support, i.e. the sparseness property as stated in Corollary 5.1 is effective. Otherwise, one would not solve (5.29), since the support could be infinite. But also note, that the loss function only depends on the hypothesis outputs and the sum of the $\hat{\boldsymbol{\alpha}}$'s. Hence, one can always express the same solution with a bounded number of non-zero $\hat{\boldsymbol{\alpha}}$'s (cf. Corollary 5.1).

A question, however, is how fast the sequence φ_t should go to zero. Our proof allows to choose φ_t to be dependent of the result of previous iterations, but it has to go to zero. We think it would be useful to adapt it to how the base learner performs. Hence, in practice we suggest to set φ_t depending on the result of the previous call of the base learner, i.e.

$$\varphi_t = \Phi \left(\sum_{n=1}^{N} d_n^{(t-1)} h_t(\boldsymbol{x}_n) - C \right) \tag{5.35}$$

for some small enough $\Phi \in (0,1)0$. Empirical results indicate that e.g. $\Phi = \frac{1}{2}$ is sufficient. From an theoretical side, however, it bears the problem to show that φ_t converges to zero. Here, we do not have an answer and it is further research to refine our results.

5.4.3 A Barrier Approach for the ε-insensitive Loss

In Section 5.4.1 we have proposed a column generation algorithm for the ε-insensitive loss, where one is required to solve a LP in each iteration exactly. This can be practical if good commercial optimizers such as CPLEX are available. In the last section, we considered a leveraging algorithm that only uses approximate solutions, that are considerably easier to obtain. However, the loss function needs to be strictly convex. In this section we consider a leveraging algorithm for the ε-insensitive loss using the barrier optimization techniques used earlier. It combines the merits of the ε-insensitive loss as discussed in Section 5.1.3.3 and of leveraging techniques as in the last section.

From an optimization point of view, however, the algorithm that we are going to propose has a theoretical problem when using infinite hypothesis sets: Barrier optimization techniques are usually applied for finite mathematical programs only and the theory for an infinite number of constraints is not well developed. We think the extension of the theory of semi-infinite barrier optimization goes beyond the scope of this thesis. Here, we would like to note that in Kaliski et al. [1999] a similar barrier algorithm using the $\log$-barrier has been used [cf. also Mosheyev and Zibulevsky, 1999]. It is future work to rigorously prove that this algorithm also converges to the optimal solution when the hypothesis set is infinite. We conjecture this is possible.

Since we have already developed a barrier algorithm for classification in Section 4.3.3, we will omit some details and also the proofs in this presentation. They can easily be adapted from the results in Section 4.3.3. Here, we will give only an outline of the algorithm [for more details and proofs see Rätsch et al., 2002].

The ε-insensitive loss can be understood as a two-sided soft-margin loss (hinge loss). For the soft-margin loss we have shown that the barrier algorithm for classification replaces it by a sequence of smooth approximations (similar to logistic loss), where the approximation accuracy is controlled by the barrier parameter β (cf. Section 4.3.3). The main idea also of this section is to replace the ε-insensitive loss by some "double-logistic" approximation.

The barrier minimization objective for problem (5.12) with $\mathbf{P}[f_{\hat{\boldsymbol{\alpha}}}] = \|\hat{\boldsymbol{\alpha}}\|_1$ using the exponential barrier can be written as (cf. Appendix B.2):

$$\begin{aligned} &E_\beta(\hat{\boldsymbol{\alpha}}, b, \boldsymbol{\xi}, \epsilon) = \\ &C\sum_{j=1}^{J}\hat{\alpha}_j + \frac{1}{N}\sum_{n=1}^{N}(\xi_n + \xi_n^*) + \nu\varepsilon + \\ &+\beta\sum_{n=1}^{N}\left[\exp\left(-\frac{\xi_n}{\beta}\right) + \exp\left(-\frac{\xi_n^*}{\beta}\right)\right] + \beta\exp\left(-\frac{\varepsilon}{\beta}\right) + \\ &+\beta\sum_{n=1}^{N}\left[\exp\left(\frac{y_n - f_{\hat{\boldsymbol{\alpha}},b}(\boldsymbol{x}_n) - \xi_n - \varepsilon}{\beta}\right) + \exp\left(\frac{f_{\hat{\boldsymbol{\alpha}},b}(\boldsymbol{x}_n) - y_n - \xi_n^* - \varepsilon}{\beta}\right)\right], \end{aligned} \tag{5.36}$$

where the constraints $\hat{\boldsymbol{\alpha}} \geq \mathbf{0}$ are omitted here. As in classification, they will be considered separately. The first line in (5.36) is the objective of (5.12), the second line implements the constraints $\epsilon > 0$ and $\xi_n \geq 0$ for $n = 1, \ldots, N$, and the third line $y_n - f_{\hat{\boldsymbol{\alpha}},b}(\boldsymbol{x}_n) \leq \varepsilon + \xi_n$ and $f_{\hat{\boldsymbol{\alpha}},b}(\boldsymbol{x}_n) - y_n \leq \varepsilon + \xi_n^*$, for $n = 1, \ldots, N$.

As in classification we can set the derivatives with respect to $\boldsymbol{\xi}$ and $\boldsymbol{\xi}^*$ to zero and obtain a simplified barrier objective (cf. Section 4.3.3):

$$\begin{aligned} &E_\beta(\hat{\boldsymbol{\alpha}}, b, \epsilon) = \\ &C\sum_{j=1}^{J}\hat{\alpha}_j + \frac{\beta}{N}\sum_{n=1}^{N}\left[\log\left(1 + \exp\left(\frac{\delta_n - \varepsilon}{\beta}\right)\right) + \log\left(1 + \exp\left(\frac{-\delta_n - \varepsilon}{\beta}\right)\right)\right] \\ &\quad +\nu\varepsilon + \beta\exp\left(-\frac{\varepsilon}{\beta}\right) \end{aligned} \tag{5.37}$$

where $\delta_n = y_n - f_{\hat{\boldsymbol{\alpha}},b}(\boldsymbol{x}_n)$. It can easily verified that (5.37) corresponds to a strictly convex loss function for $\beta > 0$ (the "double-logistic" loss). Hence, we may either use one of the leveraging schemes (cf. Section 4.3.3) or other techniques to iteratively minimize E_β. We decided to leave this issue open and take the generic approach of φ-optimal solutions of the last section. Then, in fact at least the minimization of E_β works provably also for infinite hypothesis sets.

As in classification, if the edge of the returned hypothesis is small enough, the barrier parameter β is reduced by some factor $c \in (0, 1)$ (cf. step 3d in Algorithm 5.10). If the hypothesis set is finite, then one can show that it is reduced only if the gradients with respect to all variables are small enough. Hence, one can apply the barrier convergence results (cf. Proposition B.1) to show convergence. This argumentation unfortunately only holds, if the hypothesis set is finite. Since the proof for the finite case is a straightforward extension of the proof given for the barrier algorithm for classification (cf. Theorem 4.3), we omit further details [see Rätsch et al., 2002]. The pseudo-code of our proposition is

given in Algorithm 5.10.[11] The computation of the example weighting in step 3b directly follows from step 2b in Algorithm 5.9 and our definition of the loss in (5.37).

Algorithm 5.10 The Barrier-Regression algorithm [Rätsch et al., 2002]

1. **Input:** N examples $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, Regularization constant C, $y_{\max} < \infty$, $c \in (0,1)$

   Number of iterations T
2. **Initialize:** $\beta = \beta_{\text{start}}$
3. **Do for** $t = 1, \ldots, T$
   (a) Let $(\hat{\boldsymbol{\alpha}}^{(t)}, b_t, \varepsilon_t)$ be a β-minimizer of (5.37) with $\hat{\boldsymbol{\alpha}} \geq \mathbf{0}$ using $t-1$ hypotheses
   (b) Let $d_n^{(t)} = \left[1 + \exp\left(-\frac{f_{\hat{\boldsymbol{\alpha}}^{(t)}, b_t}(\boldsymbol{x}_n) - y_n - \varepsilon_t}{\beta}\right)\right]^{-1} - \left[1 + \exp\left(-\frac{y_n - f_{\hat{\boldsymbol{\alpha}}^{(t)}, b_t}(\boldsymbol{x}_n) - \varepsilon_t}{\beta}\right)\right]^{-1}$
   (c) Call weak learner with weighted sample set $\{S, \mathbf{d}^{(t)}\}$ and obtain hypothesis $h_t : \boldsymbol{x} \mapsto [-y_{\max}, y_{\max}]$
   (d) **If** $\sum_{n=1}^{N} d_n^{(t)} h_t(\boldsymbol{x}_n) - C \leq \beta$ **then** $\beta = c\beta$
4. **Output:** Final hypothesis $f = b + \sum_{q=1}^{t-1} \hat{\alpha}_q h_q$

5.5 Evaluation and an Application

In this section we present some results indicating the feasibility of our approaches. We will start in Section 5.5.1 with showing some basic properties of the column generation (CG) and barrier algorithms for regression. We show that both algorithms are able to produce excellent fits on a noiseless and several noisy toy problems. The regularized leveraging algorithm as proposed in Section 5.4.2 works very similar to the barrier algorithm, but one can choose different loss functions. To show the competitiveness of our algorithms we performed a benchmark comparison in Section 5.5.2 on time-series prediction problems that have been extensively studied in the past. Moreover, we give an interesting application to a problem derived from computer-aided drug-design in Section 5.5.3.

To use our algorithms we need to specify base learning algorithms. We use three different base learning algorithms. The details are given in Appendix B.6.2–B.6.4. Here we give a brief description only. We use RBF kernel functions

$$h_n(\boldsymbol{x}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{\mu}^{(n)}\|^2}{\sigma_n^2}\right),$$

either with centers fixed at the training points position (cf. Appendix B.6.2) or adapted to maximize the edge (cf. Appendix B.6.3). We call the latter approach *active kernels*.[12] As a third base learner, we use a particular simple one, related to the Tree-Boost algorithm

11. Its convergence for finite hypothesis sets has been proven in Rätsch et al. [2002].
12. This idea is also known as *moving centers* proposed in T.Poggio and Girosi [1990].

[Friedman, 1999]. We will call it *sparse classification functions*, as they actually implement a classification algorithm that only tries to fit the sign to the weighting $\mathbf{d}$ (note that it can be positive and negative). It solves a linear program in the input space leading to sparse weight-vectors and, hence, to a feature selection capability (cf. Appendix B.6.4 for details). In particular for the drug-design problem this base learner turned out to be very well suited (high dimensional data with a few examples).

We will denote by CG-k, CG-ak and CG-LP, the CG algorithm using RBF kernels (cf. Appendix B.6.2), active RBF kernels (Appendix B.6.3) and classification functions (Appendix B.6.4) as base learners, respectively. Likewise for Bar-k, Bar-ak and Bar-LP using the barrier algorithm. Furthermore, we use the prefix "Lev" for the regularized leveraging algorithm. Not all of these possible combinations have been implemented, since we found that the CG and the barrier algorithm perform very similar.

5.5.1 An Experiment on toy data

To illustrate (i) that the proposed regression algorithm converges to the optimal (i.e. zero error) solution and (ii) is capable of finding a good fit to noisy data (signal:noise=2:1) we applied it to a toy example – the frequently used sinc function ($\mathrm{sinc}(x) = \sin(\pi x)/(\pi x)$) in the range $[-2\pi, 2\pi]$. For our demonstration (cf. Figure 5.2) we used two base hypothesis sets: (i) RBF kernels in the way described in Appendix B.6.2, i.e.

$$\mathsf{H} = \{h_n(\boldsymbol{x}) = \exp(-\|\boldsymbol{x} - \boldsymbol{x}_n\|^2/\sigma^2) \mid n = 1, \ldots, N\}$$

with $\sigma^2 = 1/2$ and (ii) classification functions as described in Appendix B.6.4. In the first case we used the CG and the Barrier approach – leading to the algorithms CG-k and Bar-k. The latter case is included for demonstration purposes only, the CG-LP is designed for high-dimensional data sets and does not perform well in low dimensions due to the severely restricted nature of the base hypothesis set.

To keep the results comparable between different data sets we use a normalized measure of error – the Q^2-error (also called normalized mean squared error), which is defined as:

$$Q^2 = \frac{\sum_{n=1}^{N}(y_n - f(\boldsymbol{x}_n))^2}{\sum_{n=1}^{N}(y_n - \frac{1}{N}\sum_i y_i)^2}. \tag{5.38}$$

A Q^2-value greater than one is meaningless, since simply predicting the mean value will result in a Q^2-value of one.

Let us first consider the case of RBF-kernels. In the noise-free case (left panel of Figure 5.2) we observe – as expected from Proposition 5.2 – that the (automatically determined) tube size ε is very small (0.0014), while it is kept large (0.12) for the high noise case (right panel). Using the right tube size, one gets an almost perfect fit ($Q^2 = 4 \cdot 10^{-5}$) in the noise-free case and an excellent fit in the noisy case ($Q^2 = 0.12$) – without re-tuning parameters.

The CG-LP produced a piecewise-constant function based on only two classification functions. The same solution of $Q^2 = 0.4$ was produced in both the noisy and noise-free cases. Interestingly in the noisy case it produces almost an identical function. Because the

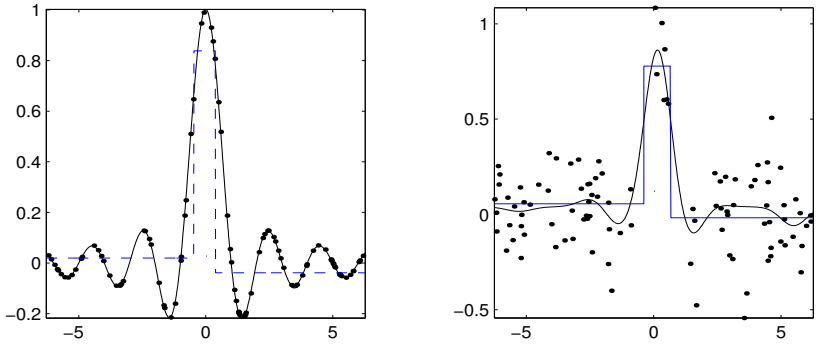

Figure 5.2 Toy example: The left panel shows the fit of the sinc function without noise using RBF-kernels (solid) and classification functions (dashed). The solid fit is almost perfect ($Q^2 = 4 \cdot 10^{-5}$), while the dashed function is too simple ($Q^2 = 0.4$). The right panel shows a fit using RBF-kernels ($Q^2 = 0.12$) on noisy data (signal:noise=2:1, $C = 100$). The tube size is automatically adapted by the algorithm ($\varepsilon = 0.0014$ (left) and $\varepsilon = 0.12$ (right)), such that a half of the patterns lie inside the tube ($\nu = 1/2$).

hypothesis sets only consists of *linear* classification functions constructed by LP (B.24) (cf. Appendix B.6.4), the set of base hypothesis is extremely restricted. Thus high bias, but low variance behavior can be expected. We will see later than on high dimensional datasets the CG-LP can perform quite well.

Let us now compare the convergence speed of CG- and Barrier-Regression in the controlled setting of this toy example. For this we run both algorithms and record the objective values of the restricted master problem. In each iteration the barrier algorithm has to find the minimizing or almost minimizing parameters $(\hat{\boldsymbol{\alpha}}, \varepsilon, b)$ of the barrier function E_β for the restricted master problem. In our implementation we use an iterative gradient descent method, where the number of gradient steps is a parameter of the algorithm. The result is shown in Figure 5.3. One observes that both algorithms converge rather fast to the optimal objective value (dotted line). The CG algorithm converges faster than the barrier algorithm, as the barrier parameter β usually decreases not quickly enough to compete with the very efficient Simplex method. However, if the number of gradient descent steps is large enough (e.g. 20), the barrier algorithm produces comparable results in the same number of iterations. Note that these gradient steps are computationally usually much cheaper than solving a linear programming problem (as e.g. for the CG-algorithm).

5.5.2 Time Series Prediction

In this section we would like to compare our new methods to SVMs and RBF networks. For this we choose two well-known data sets that have been frequently used as benchmarks on time-series prediction: (i) the Mackey-Glass chaotic time series [Mackey and Glass, 1977] and (ii) data set D from the Santa Fe competition [Weigend and Gershenfeld, 1994]. We fix the following experimental setup for our comparison. We use seven different models for our comparison: three models that have been used in Müller et al. [1999] (RBF nets and SVM-Regression (SVR) with linear and Huber loss) and four new models: CG-k, CG-ak, Bar-k and Bar-ak.

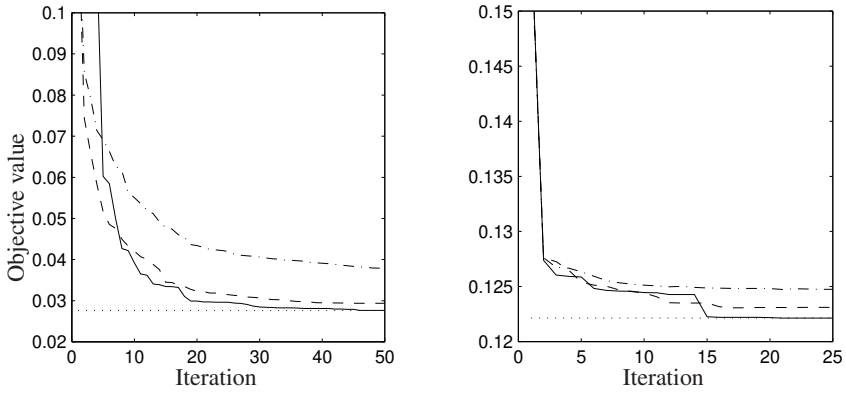

Figure 5.3 Convergence on the toy example: The convergence of the objective function $\|\hat{\boldsymbol{\alpha}}\|_1 + \|\boldsymbol{\xi}\|_1/N + C\nu\varepsilon$ in CG-Regression (solid) and Barrier-Regression to the optimal value (dotted) over the number of iterations. Left for no noise and right for large normal noise (signal:noise=2 : 1). For Barrier-Regression we did 1 (dash-dotted) and 20 (dashed) gradient descent steps in each iteration, respectively. We used $\nu = 1/2, C = 100$ and RBF-kernels with $\sigma^2 = 1/2$. We got $\|\hat{\boldsymbol{\alpha}}\|_1 = 2.7$, $\varepsilon = 0.0014$ (left) and $\|\hat{\boldsymbol{\alpha}}\|_1 = 1.3$, $\varepsilon = 0.12$ (right).

All models are trained using a simple cross validation technique. We choose the model with the minimum prediction error measured on a randomly chosen validation set [originally taken from Müller et al., 1999]. The data including our experimental results can be obtained from `http://ida.first.gmd.de/~raetsch/data/ts`.

5.5.2.1 *Mackey Glass Equation*

Our first application is a high-dimensional chaotic system generated by the Mackey-Glass delay differential equation

$$\frac{dx(t)}{dt} = -0.1x(t) + \frac{0.2x(t - t_\Delta)}{1 + x(t - t_\Delta)^{10}}, \tag{5.39}$$

with delay $t_\Delta = 17$. Eq. (5.39) was originally introduced as a model of blood cell regulation [Mackey and Glass, 1977] and became quite common as an artificial forecasting benchmark. After integrating (5.39), we added noise to the time series. We obtained training (1000 patterns) and validation (the following 194 patterns) sets using an embedding dimension $d = 6$ and a step size $\tau = 6$. The test set (1000 patterns) is noiseless to measure the true prediction error. We conducted experiments for different signal to noise ratios[13] (SNR) using uniform noise.

In Table 5.1 we state the results given in the original paper [Müller et al., 1999] for SVMs using ε-insensitive loss and Huber's robust loss (quadratic/linear) and RBF networks. Moreover, we give the results for the CG and the barrier algorithm using RBF kernels and active RBF-kernels.[14] We also applied the CG algorithm using classification functions

13. We define the SNR in this experiment as the ratio between the variance of the noise and the variance of the data.

14. On the entries set as italic, the model selection failed completely. In this case we selected the model manually

(CG-LP), but the algorithm performed very poorly ($Q^2 \approx 0.16$), because it could not generate complex enough functions. From Table 5.1 we observe that all four algorithms perform on average as good as the best of the other algorithms (in 11 cases better and in 13 cases worse). The 100 step prediction at low noise levels is rather poor compared to SVMs, but it is great on the higher noise levels.

Note that the CG and the barrier algorithm do not perform significantly different (CG is in 5 cases better and in 7 cases worse). This shows that the simple barrier implementation given in Algorithm 5.10 achieves a high enough accuracy to compete with a sophisticated simplex implementation used in the CG-algorithms.

SNR	6.2%		12.4%		18.6%	
test error	1S	100S	1S	100S	1S	100S
CG-k	0.0011	0.0804	0.0035	0.0838	0.0031	0.0882
CG-ak	0.0010	0.0749	0.0035	0.0854	0.0065	0.0998
BAR-k	*0.0013*	*0.0900*	0.0032	0.0590	0.0051	0.0661
BAR-ak	0.0012	0.0893	0.0027	0.0621	0.0066	0.0821
SVM ε-ins.	0.0007	0.0158	0.0028	0.0988	0.0057	0.4065
SVM Huber	0.0013	0.0339	0.0038	0.0852	0.0071	1.0297
RBF-NN	0.0016	0.0775	0.0038	0.1389	0.0154	1.6032

Table 5.1 1S denotes the 1-step prediction error (Q^2) on the test set. 100S is the 100-step iterated autonomous prediction. "SNR" is the ratio between the variance of the respective noise and the underlying time series.

5.5.2.2 Data Set D from the Santa Fe Competition

Data set D from the Santa Fe competition is artificial data generated from a nine-dimensional periodically driven dissipative dynamical system with an asymmetrical four-well potential and a slight drift on the parameters [Weigend and Gershenfeld, 1994]. The system has the property of operating in one well for some time and then switching to another well with a different dynamical behavior. Therefore, we first segment the time series into regimes of approximately stationary dynamics. This is accomplished by applying the *Annealed Competition of Experts* (ACE) method described in Pawelzik et al. [1996], Müller et al. [1995] (no assumption about the number of stationary subsystems was made). Moreover, in order to reduce the effect of the continuous drift, only the last 2000 data points of the training set are used for segmentation. After applying the ACE algorithm, the data points are individually assigned to classes of different dynamical modes. We then select the particular class of data that includes the data points at the end of Data Set D as the training set.[15]

This allows us to train our models on quasi-stationary data and we avoid having to predict the average over all dynamical modes hidden in the full training set [for further discussion see Pawelzik et al., 1996]. However, at the same time we are left with a rather small training set requiring careful regularization, since there are only 327 patterns in the extracted training set. As in the previous section we use a validation set (50 patterns of

by choosing the model on the 10th percentile of the test errors over all tested models.

15. Hereby we assume that the class of data that generated the last points in the training set is the one that is also responsible for the first couple of steps of the iterated continuation that we aim to predict.

the extracted quasi-stationary data) to determine the model parameters of SVMs, RBF networks and CG-Regression. The embedding parameters used, $d = 20$ and $\tau = 1$, are the same for all the methods compared in Table 5.2.

Table 5.2 shows the errors (Q^2-value) for the 25 step iterated prediction.[16] In the previous result of Müller et al. [1999] the Support vector machine with ε-ins. loss is 30% better than the one achieved by Pawelzik et al. [1996]. This is the current record on this dataset. Given that it is quite hard to beat this record, our methods perform quite well. CG-ak improves the result in Pawelzik et al. [1996] by 28%, while CG-k is 26% better.[17] This is very close to the previous result. The model-selection is a crucial issue for this benchmark competition. The model, which is selected on the basis of the best prediction on the 50 validation patterns, turns out to be rather suboptimal. Thus, more sophisticated model selection methods are needed here to obtain more reliable results.

CG		SVM		Neural Net	
CG-k	CG-ak	ε-ins.	Huber	RBF	PKM
0.036	0.035	0.032	0.033	0.060	0.066

Table 5.2 Comparison (under competition conditions) of 25 step iterated predictions (Q^2-value) on Data set D. A prior segmentation of the data according to Müller et al. [1995], Pawelzik et al. [1996] was done as preprocessing.

5.5.3 Experiments on Drug data

This data set is taken from computer-aided drug design. The goal is to predict bio-reactivity of molecules based on molecular structure through the creation of Quantitative Structure-Activity Relationship (QSAR) models. Once a predictive model has been constructed, large databases can be screened cost effectively for desirable chemical properties. Then this small subset of molecules can be tested further using traditional laboratory techniques. The target of this dataset LCCKA is the logarithm of the concentration of each compound that is required to produce 50 percent inhibition of site "A" of the Cholecystokinin (CCK) molecule. These CCK and CCK-like molecules serve important roles as neuro-transmitters and/or neuro-modulators. 66 compounds (examples) were taken from the Merck CCK inhibitor data set. The dataset[18] originally consisted of 323 descriptors (dimensions) taken from a combination of "traditional" 2D, 3D, and topological properties and electron density derived TAE (Transferable Atomic Equivalent) molecular descriptors derived using wavelets [for details see Brenema et al., 2000]. All data was scaled to be between 0 and 1.

It is well known that appropriate feature selection on this dataset and others is essential for good performance of QSAR models due to the small amount of available data with known bio-reactivity and the large number of potential descriptors [see for example Embrechts et al., 1998]. In an unrelated study [Demiriz et al., 2001a] feature selection was

16. Iterated prediction means that based on the past predictions (and not on the original data) the new prediction is computed.
17. We have not performed experiments with the barrier algorithm on this data, since the performance is expected to be similar.
18. The data can be obtained from `http://www.rpi.edu/~bennek`.

done by constructing a ℓ_1-norm linear support vector regression machine (like in equation (5.12) with fixed ε and ℓ_1-norm regularization and where the features are the input dimensions) to produce a sparse weighting of the descriptors. Only the descriptors with positive weights were retained. We take the reduced set of 39 descriptors as given. We refer to the full data set as LCCKA and the reduced dataset as LCCKA-R.

The typical performance measured used to evaluate QSAR data is the average sum squared error between the predicted and true target values divided by the true target variance. This is Q^2 as defined in (5.38). A Q^2 of less than 0.3 is considered very good. To measure the performance, 6-fold cross validation was performed. We report the out-of-sample Q^2 averaged over the 6 folds. In this first study, model-selection using parameter selection techniques was not performed. As models we consider CG-LP (CG with classification functions) and CG-k (CG with non-active kernels) described in Appendix B.6.4 and B.6.2. For CG-k, we used only three different values for the regularization constant C, the tube-parameter ν and the parameter of the base learner σ (kernel-width) and δ (complexity parameter in (B.24)), respectively. Thus, we examined 27 different parameter combinations. For CG-LP, we used parameter values found to work well on a reduced dataset in Demiriz et al. [2001a] and then choose C and δ such that the number of hypotheses and attributes per hypothesis were similar on the training data. Research is in progress to repeat these studies using a more appropriate model selection technique – leave-one-out cross validation. Model selection is critical for performance of these methods, thus efficient model selection techniques is an important open question that needs to be further addressed.

First we tried CG-k and Lev-k (the regularized leveraging algorithm with the approximated Huber loss (5.9), on the full data set LCCKA, but it failed to achieve good performance ($Q^2 = 0.48$ and $Q^2 = 0.49$), while the simple approach CG-LP already performed quite well with $Q^2 = 0.33$. This is because CG-LP is able to select the discriminative features based on subsets of the attributes, while the kernel-approaches apparently get confused by the nuisance features. For the reduced set LCCKA-R, where the features are already pre-selected, the kernel approaches CG-k and Lev-K improve significantly ($Q^2 = 0.27$ and $Q^2 = 0.28$) and is not significantly different than CG-LP ($Q^2 = 0.25$).

CG-k and CG-LP produced very sparse ensembles. On the full dataset, using parameters $C = 8$, $\nu = 0.8$, and δ = 6, CG-LP used on average ensembles containing 22 hypotheses consisting of, on average, 10.1 of the possible 323 attributes, while CG-k with RBF-kernel ($\sigma = 30$) and $\nu = 0.1$ used 45 hypotheses and all attributes. On the reduced dataset, using parameters $C = 15$, $\nu = 0.8$, and δ = 10, CG-LP used on average ensembles containing 23.5 hypotheses consisting of, on average, 10.7 attributes, while the CG-k approach ($\sigma = 10$) used on average 30.3 hypotheses ($\nu = 0.1$). The slight difference between CG-LP and CG-k might be explained again by the presence of uninformative features. The regularized leveraging algorithm Lev-k produced ensembles with more hypotheses (about 40). This is due to the choice of the loss function. For our choice, the dual variables are not sparse and therefore there are also more non-zero primal variables. However, in the analyzed case the performance is very similar to the CG-k algorithm.[19]

19. In other cases, e.g. if the noise distribution is given, one might design a particular loss function, from which Lev-k can profit.

Summarizing, the CG-LP approach seems to be a very robust method to learn simple regression functions in high-dimensional spaces with automatic feature selection. The CG and the leveraging approach with RBF kernels can perform well, if the data does not contain too many nuisance features.

5.6 Discussion and Summary

In this work we examined mathematical programming formulations for constructing regression ensembles based on ℓ_1-norm regularized loss functions. We used the dual formulation of the finite regression LP: (i) to rigorously define a proper extension to the infinite hypothesis case and (ii) to derive two efficient algorithms for solving the problems with the ε-insensitive loss and one for strictly convex loss functions. It is shown theoretically and empirically that even if the hypothesis space is infinite, only a small, finite set of the hypotheses is needed to express the optimal solution (cf. Corollary 5.1). This sparseness is possible due to the use of the ℓ_1-norm of the hypothesis coefficient vector, which acts as a sparsity-regularizer.

We proposed three different algorithms to efficiently compute an optimal finite ensembles. Here, the base-learner acts as an oracle to find the constraints in the dual semi-infinite problem that are violated, or alternatively, to find the coordinate in the primal domain that needs to be optimized. For the first two algorithms (the CG algorithm and the leveraging algorithm for regression), we proved the convergence for the infinite case (cf. Theorem 5.7). The third algorithm – the Barrier algorithm for Regression – is based on an exponential barrier method that has connections to the original AdaBoost method for classification [cf. Rätsch et al., 2000c]. This algorithm converges for finite hypothesis classes. Using recent results in the mathematical programming literature [e.g. Mosheyev and Zibulevsky, 1999, Kaliski et al., 1999] we claim that it is possible to generalize it to the infinite case. Computationally the first and third algorithm find a provably optimal solution in a small number of iterations.

We examined three types of base learning algorithms. One, based on boosting kernel functions chosen from a finite dictionary of kernels, is an example of a finite hypothesis set. We also consider active kernel methods where the kernel basis is selected from an infinite dictionary of kernels. Finally we consider the case using the finite set of linear classification functions constructed using an LP. This is a very limited hypothesis space that is specifically designed to work on under-determined high-dimensional problems such as the drug design data discussed in this work.

Our simulations on toy and real world data showed that the proposed algorithms behave very well in both finite and infinite cases. In a benchmark comparison on time-series prediction problems our algorithms perform as well as the current state of the art regression methods such as support vector machines for regression. In the case of “Data set D” of the Santa Fe competition we obtained results that are competitive with the current record (by SVM) on this dataset. The LP classification-based approach worked extremely well on the high-dimensional drug design datasets, since the algorithm inherently performs feature selection essential for success on such datasets.

The primary contribution of this chapter has been a theoretical and conceptual study of ℓ_1-norm regularized ensemble regression algorithms in finite and infinite hypothesis spaces. For future work we plan a more rigorous investigation of the computational aspects of our approach. One open question is how to best perform selection of the model parameters. Another open question involves the best algorithmic approaches for solving the semi-infinite linear program. While they work well in practice, the column generation and barrier interior-point methods described here are not the current state of the art for semi-infinite linear programming. A primal-dual interior point algorithm may perform even better both theoretically and empirically especially on very large datasets. Lastly, the ability to handle infinite hypothesis sets opens up the possibility of many other possible types of base learning algorithms.

6 Synopsis

In this thesis we have considered leveraging methods for classification and regression problems. One of our major efforts was to work out *connections between leveraging methods* and *convex optimization*. This understanding enabled us to transfer knowledge from the theory of convex optimization to ensemble learning. We have shown whether, under which conditions and how fast leveraging algorithms converge to the solution of a convex optimization problem. Particularly, we found that AdaBoost can be used to efficiently maximize the margin in classification, i.e. to approximate the solution of a very large linear programming problem (Chapter 2). Furthermore, we considered a quite general family of algorithms to solve large non-linear convex optimization problems. We identified sufficient conditions for linear and asymptotic convergence if the hypothesis set is finite (Chapter 3). Finally, we have been able to drop the assumption of finiteness of the hypothesis set and presented algorithms and theory for regularized ensemble learning with infinitely many hypotheses (Chapter 5). Such algorithms can still be efficient, if one uses the ℓ_1-norm regularization on the combined hypothesis. This turns out to be for similar reasons as for learning with kernels: the solution can always be expressed by a small linear combination of hypotheses. Our work closes a central gap between existing algorithms and their theoretical understanding in terms of convergence. These results are indeed very important for understanding leveraging algorithms, since it would be hard to give guarantees for algorithms whose outputs are not converging or not characterizable.

We discussed in detail that AdaBoost-type algorithms and hard margin classifiers in general are noise sensitive and prone to overfit (Chapter 4). We introduced two *regularization strategies* for AdaBoost to alleviate the overfitting problem: first a direct incorporation of a regularization term into AdaBoost's loss function leading to $\text{AdaBoost}_{\text{Reg}}$ and second by introducing slack variables to relax the hard margin constraint leading to a *regularized linear optimization problem*. The essence of our proposed algorithms is to achieve a *soft margin* in contrast to the hard margin classification used before – a practice that is already used successfully in support vector learning. We found clear theoretical and practical reasons why one should use the proposed regularized leveraging algorithms when analyzing noisy data. Our improvements were indeed necessary and place Boosting back into the standard toolbox of machine learning techniques.

In the last part of this work (Chapter 5) we developed extensions of the boosting idea for *regression tasks*. For this we studied mathematical programming formulations for regression using different cost functions and regularization operators. The ϵ-insensitive loss and an approximation of Huber's robust loss turned out to have desirable properties for our purposes. Using the previously developed understanding of leveraging methods, we

proposed *practical algorithms* for constructing ℓ_1-norm regularized regression ensembles. Our work extends the practical applicability of boosting to regression tasks, and thus solves a problem for which for long no convincing solution was available.

In simulations we showed that the proposed regularized algorithms for classification and regression behave very well and achieve excellent results. In benchmark comparisons on ten data sets of the IDA repository and on two time-series prediction problems they perform competitive to state of the art classification and regression methods such as support vector machines. We expect that these simple-to-implement algorithms will be frequently used in the near future.

To further illustrate the usefulness of our algorithms in a real-world setting, we used them in two interesting industrial applications: a non-intrusive power monitoring system and in the drug discovery process. In both applications only a few examples are available and learning, i.e. generalization to unseen data, is a difficult problem. We have been able to achieve generalization performances that are of great value for these applications. For the power monitoring system we have applied for a patent.

Summarizing, the main contribution of this work is to understand boosting methods from a learning theoretical *and* optimization point of view and exploit this understanding to develop *practical and ready-to-use* algorithms for classification. We conjecture that our results can be used to analyze many other variants of boosting type algorithms proposed recently in the literature (cf. `http://www.boosting.org`). One of the most difficult and challenging open question is how to best perform selection of the model parameters. Moreover, we have several other applications of our methods in genome analysis, drug discovery and fraud detection in mind, which will be worked out in the future.

References

H. Akaike. A new look at the statistical model identification. *IEEE Trans. Automat. Control*, 19(6):716–723, 1974.

S. Amari and M. Kawanabe. Information geometry of estimating functions in semiparametric statistical models. *Bernoulli*, 3:29–54, 1997.

M. Anthony and N. Biggs. *Computational Learning Theory*, volume 30 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1997.

J.A. Aslam. Improving algorithms for boosting. In *Proc. COLT*, San Francisco, 2000. Morgan Kaufmann.

R. Avnimelech and N. Intrator. Boosted mixture of experts: An ensemble learning scheme. *Neural Computation*, 11:483–497, 1999a.

R. Avnimelech and N. Intrator. Boosting regression estimators. *Neural Computation*, 11: 499–520, 1999b.

J.P. Barnes. Capacity control in boosting using a p-convex hull. Master's thesis, Australian National University, 1999. supervised by R.C. Williamson.

P.L. Bartlett. The sample complexity of pattern classification with neural networks: The size of the weight is more important than the size of the network. *IEEE Transactions on Information Geometry*, 44(2):525–536, 1998.

E.B. Baum and D. Haussler. What size net gives valid generalization? *Neural Computation*, 1:151–160, 1989.

H.H. Bauschke and J.M. Borwein. Legendre functions and the method of random bregman projections. *Journal of Convex Analysis*, 4:27–67, 1997.

R.E. Bellman. *Adaptive Control Processes: A guided Tour*. Princeton Univ. Press, 1961.

K. Bennett. Combining support vector and mathematical programming methods for induction. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in Kernel Methods - SV Learning*, pages 307–326, Cambridge, MA, 1999. MIT Press.

K.P. Bennett, A. Demiriz, and J. Shawe-Taylor. A column generation algorithm for boosting. In P. Langley, editor, *Proceedings, 17th ICML*, pages 65–72, San Francisco, 2000. Morgan Kaufmann.

K.P. Bennett and O.L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.

A. Bertoni, P. Campadelli, and M. Parodi. A boosting algorithm for regression. In W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, editors, *Proceedings ICANN'97, Int. Conf. on Artificial Neural Networks*, volume V of *LNCS*, pages 343–348, Berlin, 1997. Springer.

D.P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1995.

C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

G. Blanchard. *Methodes de melange et d'aggregation d'estimateurs en reconnaissance de formes. Applications aux arbres de decision*. PhD thesis, University Paris 13, 2001.

A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, 1987.

A. Blumer, A. Ehrenfeucht, D. Haussler, and M.K. Warmuth. Learnability and the vapnik chervonenkis dimension. *Journal of the ACM*, 36(4):925–965, October 1989.

J.M. Borwein. Adjoint process duality. *Math. of Oper. Res.*, 8, 403–434 1983a.

J.M. Borwein. Semi-infinite programming: how special is it? In A.V. Fiacco and K.O. Kortanek, editors, *Semi-Infinite Programming and Applications*, number 215 in Lecture notes in Econom. and Math. Systems, pages 10–36. Springer, Berlin, 1983b.

B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, 1992.

O. Bousquet and A. Elisseeff. Algorithmic stability and generalization performance. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13, pages 196–202. MIT Press, 2001a.

O. Bousquet and A. Elisseeff. Stability and generalization. Submitted to Journal of Machine Learning Research, 2001b.

P. Bradley, O. Mangasarian, and J. Rosen. Parsimonious least norm approximation. *Computational Optimization and Applications*, 11(1):5–21, 1998.

P.S. Bradley and O.L. Mangasarian. Feature selection via concave minimization and support vector machines. In *Proc. 15th International Conf. on Machine Learning*, pages 82–90. Morgan Kaufmann, San Francisco, CA, 1998.

L.M. Bregman. The relaxation method for finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Math. and Math. Physics*, 7:200–127, 1967.

L. Breiman. Bagging predictors. *Machine Learning*, 26(2):123–140, 1996.

L. Breiman. Prediction games and arcing algorithms. *Neural Computation*, 11(7):1493–1518, 1999. Also Technical Report 504, Statistics Department, UC Berkeley.

C. Brenema, N. Sukumar, K.P. Bennett, M.J. Embrechts, M. Sundling, and L. Lockwood. Wavelet representations of molecular electronic properties: Applications in adme, qspr, and qsar. Presentation, QSAR in Cells Symposium of the Computers in Chemistry Division's 220th American Chemistry Society National Meeting, August 2000.

J. Carmichael. Non-intrusive appliance load monitoring system. Epri journal, Electric Power Research Institute, 1990.

Y. Censor and S.A. Zenios. *Parallel Optimization: Theory, Algorithms and Application*. Numerical Mathematics and Scientific Computation. Oxford University Press, 1997.

N. Cesa-Bianchi, A. Krogh, and M. Warmuth. Bounds on approximate steepest descent for likelihood maximization in exponential families. *IEEE Transaction on Information Theory*, 40(4):1215–1220, July 1994a.

N. Cesa-Bianchi, A. Krogh, and M. Warmuth. Bounds on approximate steepest descent for likelihood maximization in exponential families. *IEEE Trans. Inf. Th.*, 40(4):1215–1220, 1994b.

O. Chapelle, V. Vapnik, and J. Weston. Transductive inference for estimating values of functions. In S. Solla, T. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 421–428. MIT press, 2000.

S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. Technical Report 479, Department of Statistics, Stanford University, 1995.

M. Collins, R.E. Schapire, and Y. Singer. Logistic Regression, AdaBoost and Bregman distances. In *Proc. COLT*, pages 158–169, San Francisco, 2000. Morgan Kaufmann.

R. Cominetti and J.-P. Dussault. A stable exponential penalty algorithm with superlinear convergence. *J.O.T.A.*, 83(2), Nov 1994.

J. Copas. Regression, prediction and shrinkage. *J.R. Statist. Soc. B*, 45:311–354, 1983.

C. Cortes and V.N. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

R. Courant and D. Hilbert. *Methods of Mathematical Physics*, volume 1. Interscience Publishers, Inc, New York, 1953.

T.M. Cover and P.E. Hart. Nearest neighbor pattern classifications. *IEEE transaction on information theory*, 13(1):21—27, 1967.

D.D. Cox and F. O'Sullivan. Asymptotic analysis of penalized likelihood and related estimates. *The Annals of Statistics*, 18(4):1676–1695, 1990.

Using the CPLEX Callable Library. CPLEX Optimization Incorporated, Incline Village, Nevada, 1994.

S. Das. Filters, wrappers and a boosting-based hybrid for feature selection. In *Proceedings, 18th ICML*. Morgan Kaufmann, 2001.

S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, April 1997.

S. Della Pietra, V. Della Pietra, and J. Lafferty. Duality and auxiliary functions for bregman distances. Technical Report CMU-CS-01-109, School of Computer Science, Carnegie Mellon University, 2001.

A. Demiriz, K.P. Bennett, C. Breneman, and M. Embrechts. Support vector machine regression in chemometrics. In *Computer Science and Statistics: Proceedings of the conference on the 32 Symposium on the Interface*, 2001a.

A. Demiriz, K.P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Journal of Machine Learning Research*, 2001b. To appear in special issue on Support Vector Machines and Kernel Methods.

L. Devroye. Bounds for the uniform deviation of empirical measures. *Journal of Multivariate Analysis*, 12:72–79, 1982.

M. Doljansky and M. Teboulle. An interior proximal algorithm and the exponential multiplier method for semidefinite programming. *SIAM J. Optim.*, 9(1):1–13, 1998.

C. Domingo and O. Watanabe. A modification of AdaBoost. In *Proc. COLT*, San Francisco, 2000. Morgan Kaufmann.

H. Drucker, C. Cortes, L.D. Jackel, Y. LeCun, and V. Vapnik. Boosting and other ensemble methods. *Neural Computation*, 6, 1994.

H. Drucker, R. Schapire, and P.Y. Simard. Boosting performance in neural networks. *Int. J. of Pattern Recognition and Artificial Intelligence*, 7:705–719, 1993.

N. Duffy and D.P. Helmbold. A geometric approach to leveraging weak learners. In P. Fischer and H. U. Simon, editors, *Computational Learning Theory: 4th European Conference (EuroCOLT '99)*, pages 18–33, March 1999. Long version to appear in TCS.

N. Duffy and D.P. Helmbold. Leveraging for regression. In *Proc. COLT*, pages 208–219, San Francisco, 2000a. Morgan Kaufmann.

N. Duffy and D.P. Helmbold. Potential boosters? In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 258–264. MIT Press, 2000b.

M. Embrechts, R. Kewley, and C. Breneman. Computationally intelligent data mining for the automated design and discovery of novel pharmaceuticals. In C. Dagli et al., editor, *Intelligent Engineering Systems Through Artifical Neural Networks*, pages 397–403152–161. ASME Press, 1998.

D.H. Fisher, Jr., editor. *Improving regressors using boosting techniques*, Proceedings of the Fourteenth International Conference on Machine Learning, 1997.

S. Floyd and M. Warmuth. Sample compression, learnability, and the Vapnik-Chervonenkis dimension. *Machine Learning*, 21(3):269–304, 1995.

Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, September 1995.

Y. Freund and R. Schapire. Game theory, on-line prediction and boosting. In *Proc. COLT*, pages 325–332, New York, NY, 1996a. ACM Press.

Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *EuroCOLT: European Conference on Computational Learning Theory*. LNCS, 1994.

Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *Proc. 13th International Conference on Machine Learning*, pages 148–146. Morgan Kaufmann, 1996b.

Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

Y. Freund and R.E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29:79–103, 1999a.

Y. Freund and R.E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, September 1999b. Appeared in Japanese, translation by Naoki Abe.

J. Friedman, T. Hastie, and R.J. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 2:337–374, 2000, with discussion pp. 375–407. Also Technical Report at Department of Statistics, Sequoia Hall, Stanford University.

J.H. Friedman. Greedy function approximation. Technical report, Department of Statistics, Stanford University, February 1999.

K.R. Frisch. The logarithmic potential method of convex programming. Memorandum, University Institute of Economics, Oslo, May 13 1955.

C. Gentile and M.K. Warmuth. Linear hinge loss and average margin. In M.S. Kearns, S.A. Solla, and D.A. Cohn, editors, *Advances in Neural information processings systems*, volume 11, pages 225–231. MIT Press, 1999.

K. Glashoff. Duality theory of semi-infinite linear programming. *Semi-Infinite Programming*, pages 1–16, 1979.

K. Glashoff and S.A. Gustafson. *Einführung in die lineare Optimierung*. Wissenschaftliche Buchgesellschaft, Darmstadt, 1978.

T. Graepel, R. Herbrich, B. Schölkopf, A.J. Smola, P.L. Bartlett, K.-R. Müller, K. Obermayer, and R.C. Williamson. Classification on proximity data with LP-machines. In D. Willshaw and A. Murray, editors, *Proceedings of ICANN'99*, volume 1, pages 304–309. IEE Press, 1999.

Y. Grandvalet. Bagging can stabilize without reducing variance. In *ICANN'01*, Lecture Notes in Computer Science. Springer, 2001.

A.J. Grove and D. Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *Proceedings of the Fifteenth National Conference on Artifical Intelligence*, 1998.

J. B. Hampshire and A. Waibel. A novel objective function for improved phoneme recognition using time-delay neural networks. *IEEE Trans. Neural Networks*, 1:216–228, 1990.

W. Hart. Non-intrusive appliance load monitoring. *Proceedings of the IEEE*, 80(12), 1992.

D. Haussler, M. Kearns, N. Littlestone, and M.K. Warmuth. Equivalence of models for polynomial learnability. *Information and Computation*, 95(2):129–161, December 1991.

R. Herbrich. *Learning Linear Classifiers: Theory and Algorithms*, volume 7 of *Adaptive Computation and Machine Learning*. MIT Press, 2001. forthcoming.

R. Herbrich, T. Graepel, and J. Shawe-Taylor. Sparsity vs. large margins for linear classifiers. In *Proc. COLT*, pages 304–308, San Francisco, 2000. Morgan Kaufmann.

R. Herbrich and J. Weston. Adaptive margin support vector machines for classification. In *Proceedings of the Ninth International Conference on Artificial Neural Networks*, pages 880–885, 1999.

M. Herbster and M. Warmuth. Tracking the best linear prediction. *Journal of Machine Learning Research*, pages 281–309, September 01.

R. Hettich and K.O. Kortanek. Semi-infinite programming: Theory, methods and applications. *SIAM Review*, 3:380–429, September 1993.

A.J. Hoffmann. On approximate solutions of systems of linear inequalities. *Journal of Research of the National Bureau of Standards*, 49(4):263–265, October 1952.

P. J. Huber. *Robust Statistics*. John Wiley and Sons, New York, 1981.

T.S. Jaakkola and D. Haussler. Probabilistic kernel regression models. In *Proceedings of the 1999 Conference on AI and Statistics*, 1999.

W. Jiang. Does boosting overfit: Views from an exact solution. Technical Report 00-04, Department of Statistics, Northwestern University, September 2000a.

W. Jiang. Is regularization unnecessary for boosting? Technical Report 00-04, Department of Statistics, Northwestern University, November 2000b.

W. Jiang. On weak base hypotheses and their implications for boosting regression and classification. Technical Report 00-01, Department of Statistics, Northwestern University, October 2000c.

W. Jiang. Process consistency for AdaBoost. Technical Report 00-05, Department of Statistics, Northwestern University, November 2000d.

W. Jiang. Some results on weakly accurate base learners for boosting regression and classification. In *Proceedings of the First International Workshop on Multiple Classifier Systems, Cagliari, Italy, June 2000.*, volume 1857 of *Lecture Notes in Computer Science*, pages 87–96. Springer, 2000e.

W. Jiang. Some theoretical aspects of boosting in the presence of noisy data. Technical Report 01-01, Department of Statistics, Northwestern University, 2001. To appear in Proceedings: The Eighteenth International Conference on Machine Learning (ICML-2001), June 2001, Morgan Kaufmann.

J. Kaliski, D. Haglin, C.Roos, and T. Terlaky. Logarithmic barrier decomposition methods for semi-infinite programming. Submitted to Elsevier Science, April 1999.

M.J. Kearns and U.V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

G.S. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *J. Math. Anal. Applic.*, 33:82–95, 1971.

J. Kivinen and M. Warmuth. Boosting as entropy projection. In *Proc. 12th Annu. Conf. on Comput. Learning Theory*, pages 134–144. ACM Press, New York, NY, 1999.

J. Kivinen, M. Warmuth, and P. Auer. The perceptron algorithm vs. winnow: Linear vs. logarithmic mistake bounds when few input variables are relevant. *Special issue of Artificial Intelligence*, 97(1–2):325–343, 1997.

J. Kivinen and M.K. Warmuth. Additive versus exponentiated gradient updates for linear prediction. *Information and Computation*, 132(1):1–64, 1997.

K.C. Kiwiel. Relaxation methods for strictly convex regularizations of piecewise linear programs. *Applied Mathematics and Optimization*, 38:239–259, 1998.

J. Kohlmorgen, S. Lemm, G. Rätsch, and K.-R. Müller. Analysis of nonstationary time series by mixtures of self-organizing predictors. In *Neural Networks for Signal Processing X*, pages 85–94, Sydney, 2000. IEEE.

B.W. Kort and D.P. Bertsekas. Multiplier methods for convex programming. In *Proc 1073 IEEE Conf. Decision Control, San-Diego, Calif.*, pages 428–432, 1973.

A. Krieger, A. Wyner, and C. Long. Boosting noisy data. In *Proceedings, 18th ICML.* Morgan Kaufmann, 2001.

J. Lafferty. Additive models, boosting, and inference for generalized divergences. In *Proc. 12th Annu. Conf. on Comput. Learning Theory*, pages 125–133, New York, NY, 1999. ACM Press.

J.D. Lafferty, S. Della Pietra, and V. Della Pietra. Statistical learning algorithms based on bregman distances. In *Proc. of the Canadian Workshop on Information Theory*, pages 77–80, Fields Institute, Toronto, Canada, 1997.

A. Lazarevic and Z. Obradovic. Adaptive boosting techniques in heterogeneous and spatial databases. *Intelligent Data Analysis*, 2001. in press.

Y. LeCun, L.D. Jackel, L. Bottou, C. Cortes, J.S. Denker, H. Drucker, I.Guyon, U.A. Müller, E. Säckinger, P. Simard, and V. Vapnik. Learning algorithms for classification: A comparism on handwritten digit recognistion. *Neural Networks*, pages 261–276, 1995.

N. Littlestone, P.M. Long, and M.K. Warmuth. On-line learning of linear functions. *Journal of Computational Complexity*, 5:1–23, 1995. Earlier version is Technical Report CRL-91-29 at UC Santa Cruz.

N. Littlestone and M. Warmuth. Relating data compression and learnability. Technical report, University of California at Santa Cruz, USA, June 10 1986.

D.G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley Publishing Co., Reading, second edition, May 1984. ISBN 0-201-15794-2. Reprinted with corrections in May, 1989.

A. Luntz and V. Brailowsky. On estimation characters obtained in statistical procedure of recognition. *Technicheskaya Kibernetica*, 3, 1969. In russian.

Z.-Q. Luo and P. Tseng. On the convergence of coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72(1): 7–35, 1992.

M. C. Mackey and L. Glass. Oscillation and chaos in physiological control systems. *Science*, 197:287–289, 1977.

R. Maclin and D. Opitz. An empirical evaluation of bagging and boosting. In *Proceedings of the Fourteenth National Conference on AI*, pages 546–551, 1997.

S. Mallat and Z. Zhang. Matching Pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, December 1993.

O.L. Mangasarian. Linear and nonlinear separation of patterns by linear programming. *Operations Research*, 13:444–452, 1965.

O.L. Mangasarian. Arbitrary-norm separating plane. *Operation Research Letters*, 24(1): 15–23, 1999.

O.L. Mangasarian and T.H. Shiau. Lipschitz continuity of solutions of linear inequalities: Programs and complementarity problems. *SIAM Journal on Control and Optimization*, 25:583–595, 1987.

D.D. Margineantu and T.G. Dietterich. Pruning adaptive boosting. In D.H. Fisher, editor, *Proc. ICML'97*, pages 211–218. Morgan Kaufmann, 1997.

L. Mason, P.L. Bartlett, and J. Baxter. Improved generalization through explicit optimization of margins. Technical report, Department of Systems Engineering, Australian National University, 1998.

L. Mason, J. Baxter, P.L. Bartlett, and M. Frean. Functional gradient techniques for combining hypotheses. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Adv. in Large Margin Classifiers*, pages 221–247. MIT Press, Cambridge, 2000.

R. Meir, R. El-Yaniv, and Shai Ben-David. Localized boosting. In *Proc. COLT*, pages 190–199, San Francisco, 2000. Morgan Kaufmann.

J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Roy. Soc. London*, A 209:415–446, 1909.

S. Mika, G. Rätsch, and K.-R. Müller. A mathematical programming approach to the Kernel Fisher algorithm. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13, pages 591–597. MIT Press, 2001.

S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. In Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, editors, *Neural Networks for Signal Processing IX*, pages 41–48. IEEE, 1999a.

S. Mika, G. Rätsch, J. Weston, B. Schölkopf, A.J. Smola, and K.-R. Müller. Invariant feature extraction and classification in kernel spaces. In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 526–532. MIT Press, 2000a.

S. Mika, G. Rätsch, J. Weston, B. Schölkopf, A.J. Smola, and K.-R. Müller. Learning discriminative and invariant nonlinear features. Submitted to IEEE PAMI, March 2000b.

S. Mika, B. Schölkopf, A.J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. Kernel PCA and de–noising in feature spaces. In M.S. Kearns, S.A. Solla, and D.A. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11, pages 536–542. MIT Press, 1999b.

P. Moerland and E. Mayoraz. Dynaboost: Combining boosted hypotheses in a dynamic way. Technical Report IDIAP-RR99-09, IDIAP, 1999.

J. Moody. The *effective* number of parameters: An analysis of generalization and regularization in non-linear learning systems. In S. J. Hanson J. Moody and R. P. Lippman, editors, *Advances in Neural information processings systems*, volume 4, pages 847–854, San Mateo, CA, 1992. Morgan Kaufman.

J. Moody and C. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2):281–294, 1989.

L. Mosheyev and M. Zibulevsky. Penalty/barrier multiplier algorithm for semidefinite programming. *Optimization Methods and Software*, 1999.

K.-R. Müller, J. Kohlmorgen, and K. Pawelzik. Analysis of switching dynamics with competing neural networks. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E78–A(10):1306–1315, 1995.

K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201, 2001.

K.-R. Müller, A. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik. Predicting time series with support vector machines. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 243–254, Cambridge, MA, 1999. MIT Press. Short version appeared in ICANN'97, Springer Lecture Notes in Computer Science Vol. 1327.

N. Murata, S. Amari, and S. Yoshizawa. Network information criterion — determining the

number of hidden units for an artificial neural network model. *IEEE Transactions on Neural Networks*, 5:865–872, 1994.

S. Nash and A. Sofer. *Linear and Nonlinear Programming*. McGraw-Hill, New York, NY, 1996.

T. Onoda, G. Rätsch, and K.-R. Müller. An asymptotic analysis of AdaBoost in the binary classification case. In L. Niklasson, M. Bodén, and T. Ziemke, editors, *Proc. of the Int. Conf. on Artificial Neural Networks (ICANN'98)*, pages 195–200, March 1998.

T. Onoda, G. Rätsch, and K.-R. Müller. A non-intrusive monitoring system for household electric appliances with inverters. In H. Bothe and R. Rojas, editors, *Proc. of NC'2000*, Berlin, 2000. ICSC Academic Press Canada/Switzerland.

T. Onoda, G. Rätsch, and K.R. Müller. An arcing algorithm with an intuitive learning control parameter. *Journal of the Japanese Society for AI*, 16(5C):417–426, September 2001a. In Japanese.

T. Onoda, G. Rätsch, Y. Nakano, K. Yoshimoto, and K.-R. Müller. A non-intrusive monitoring system for household electric appliances with inverters. *IEEE Transactions on Power Systems*, 2001b. in preparation.

G. Orr and K.-R. Müller, editors. *Neural Networks: Tricks of the Trade*, volume 1524. Springer LNCS, 1998.

M.J.L. Orr. Introduction to radial basis function networks. Technical report, Centre for Neural Systems, Edinburgh University, 1996.

J. O'Sullivan, J. Langford, R. Caruana, and A. Blum. Featureboost: A meta-learning algorithm that improves model robustness. In *Proceedings, 17th ICML*. Morgan Kaufmann, 2000.

P.M. Pardalos and S.A. Vavasis. Quadratic programming with one negative eigenvalue is NP-hard. *Journal of Global Optimization*, 1:15–22, 1992.

K. Pawelzik, J. Kohlmorgen, and K.-R. Müller. Annealed competition of experts for a segmentation and classification of switching dynamics. *Neural Computation*, 8(2):342–358, 1996.

F. Pérez-Cruz, P.L. Alarcón-Diana, A. Navia-Vázquez, and A. Artés-Rodríguez. Fast training of support vector classifiers. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Inf. Proc. Systems*, volume 13, pages 734–740. MIT Press, 2001.

T. Poggio and F. Girosi. Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, 247:978–982, 1990.

W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, Cambridge, second edition, 1992.

J.R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1992.

J.R. Quinlan. Boosting first-order learning. *Lecture Notes in Computer Science*, 1160:143, 1996.

G. Rätsch. Ensemble learning methods for classification. Master's thesis, Dep. of Computer Science, University of Potsdam, April 1998. In German.

G. Rätsch, A. Demiriz, and K. Bennett. Sparse regression ensembles in infinite and finite hypothesis spaces. *Machine Learning*, 48(1-3):193–221, 2002. Special Issue on New

Methods for Model Selection and Model Combination. Also NeuroCOLT2 Technical Report NC-TR-2000-085.

G. Rätsch, S. Mika, B. Schölkopf, and K.-R. Müller. Constructing boosting algorithms from SVMs: an application to one-class classification. *IEEE PAMI*, 2002. In press. Earlier version is GMD TechReport No. 119, 2000.

G. Rätsch, S. Mika, and M.K. Warmuth. On the convergence of leveraging. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural information processings systems*, volume 14, 2002. In press. Longer version also NeuroCOLT Technical Report NC-TR-2001-098.

G. Rätsch, T. Onoda, and K.-R. Müller. An improvement of AdaBoost to avoid overfitting. In *Proc. of the Int. Conf. on Neural Information Processing (ICONIP)*, pages 506–509, Kitakyushu, Japan, May 1998.

G. Rätsch, T. Onoda, and K.-R. Müller. Regularizing AdaBoost. In M.S. Kearns, S.A. Solla, and D.A. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11, pages 564–570. MIT Press, 1999.

G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, March 2001. also NeuroCOLT Technical Report NC-TR-1998-021.

G. Rätsch, B. Schölkopf, A.J. Smola, S. Mika, T. Onoda, and K.-R. Müller. Robust ensemble learning. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 207–219. MIT Press, Cambridge, MA, 2000a.

G. Rätsch, B. Schölkopf, A.J. Smola, K.-R. Müller, T. Onoda, and S. Mika. ν-Arc: Ensemble learning in the presence of outliers. In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 561–567. MIT Press, 2000b.

G. Rätsch and M. Warmuth. Some primal and dual formulations. unpublished manuscript., July 2000.

G. Rätsch, M. Warmuth, S. Mika, T. Onoda, S. Lemm, and K.-R. Müller. Barrier boosting. In *Proc. COLT*, pages 170–179, San Francisco, 2000c. Morgan Kaufmann.

G. Rätsch and M.K. Warmuth. Marginal boosting. NeuroCOLT2 Technical Report 97, Royal Holloway College, London, July 2001.

G. Ridgeway, D. Madigan, and T. Richardson. Boosting methodology for regression problems. In D. Heckerman and J. Whittaker, editors, *Proceedings of Artificial Intelligence and Statistics '99*, pages 152–161, 1999.

S.M. Robinson. Bounds for errors in the solution set of a perturbed linear programm. *Linear Algebra and its applications*, 6:69–81, 1973.

R.T. Rockafellar. *Convex Analysis*. Princeton Landmarks in Mathemathics. Princeton University Press, New Jersey, 1970.

J. Rokui and H. Shimodaira. Improving the generalization performance of the minimum classification error learning and its application to neural networks. In *Proc. of the Int. Conf. on Neural Information Processing (ICONIP)*, pages 63–66, Kitakyushu, Japan, May 1998.

R.E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

R.E. Schapire. *The Design and Analysis of Efficient Learning Algorithms*. PhD thesis, MIT Press, 1992.

R.E. Schapire. A brief introduction to boosting. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999.

R.E. Schapire, Y. Freund, P.L. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5): 1651–1686, October 1998.

R.E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, December 1999. also Proceedings of the 14th Workshop on Computational Learning Theory 1998, pages 80–91.

B. Schölkopf. *Support vector learning*. Oldenbourg Verlag, Munich, 1997.

B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors. *Advances in Kernel Methods – Support Vector Learning*. MIT Press, 1999a.

B. Schölkopf, R. Herbrich, A.J. Smola, and R.C. Williamson. A generalized representer theorem. Technical Report 81, NeuroCOLT, 2000. Published in: Williamson and Helmboldt (eds.), Proceedings COLT'01, Springer Lecture Notes in Artificial Intelligence.

B. Schölkopf, S. Mika, C.J.C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A.J. Smola. Input space vs. feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5):1000–1017, September 1999b.

B. Schölkopf, S. Mika, A.J. Smola, G. Rätsch, and K.-R. Müller. Kernel PCA pattern reconstruction *via* approximate pre-images. In L. Niklasson, M. Bodén, and T. Ziemke, editors, *Proceedings of the 8th International Conference on Artificial Neural Networks*, Perspectives in Neural Computing, pages 147 – 152, Berlin, 1998. Springer Verlag.

B. Schölkopf, A. Smola, R.C. Williamson, and P.L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1207 – 1245, 2000. also NeuroCOLT Technical Report NC-TR-1998-031.

B. Schölkopf, K.-K. Sung, C.J.C. Burges, F. Girosi, P. Niyogi, and V.N. Vapnik. Comparing support vector machines with gaussian kernels to radial basis function classifiers. *IEEE Transactions on Signal Processing*, 45(11):2758–2765, 1997.

H. Schwenk and Y. Bengio. AdaBoosting neural networks. In W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, editors, *Proc. of the Int. Conf. on Artificial Neural Networks (ICANN'97)*, volume 1327 of *LNCS*, pages 967–972, Berlin, 1997. Springer.

D.W. Scott. *Multivariate Density Estimation*. Wiley, New York, 1992.

J. Shawe-Taylor, P.L. Bartlett, R.C. Williamson, and M. Anthony. A framework for structural risk minimization. In *Proc. COLT*. Morgan Kaufmann, 1996.

J. Shawe-Taylor and N. Cristianini. Robust bounds on generalization from the margin distribution. Technical Report NC-TR-98-029, NeuroCOLT2, October 1998.

J. Shawe-Taylor and G. Karakoulas. Towards a strategy for boosting regressors. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 247–258, Cambridge, MA, 2000. MIT Press.

Y. Singer. Leveraged vector machines. In S.A. Solla, T.K. Leen, and K.-R. Müller, editors,

Advances in Neural Information Processing Systems, volume 12, pages 610–616. MIT Press, 2000.

A.J. Smola. *Learning with Kernels*. PhD thesis, Technische Universität Berlin, 1998.

A.J. Smola, N. Murata, B. Schölkopf, and K.-R. Müller. Asymptotically optimal choice of ε-loss for support vector machines. In L. Niklasson, M. Bodén, and T. Ziemke, editors, *Proceedings of the 8th International Conference on Artificial Neural Networks*, Perspectives in Neural Computing, Berlin, 1998a. Springer Verlag.

A.J. Smola, B. Schölkopf, and K.-R. Müller. The connection between regularization operators and support vector kernels. *Neural Networks*, 11:637–649, 1998b.

A.J. Smola, B. Schölkopf, and K.-R. Müller. General cost functions for support vector regression. In *Proc. of the Ninth Australian Conf. on Neural Networks*, 1998c.

A.J. Smola, B. Schölkopf, and G. Rätsch. Linear programs for automatic accuracy control in regression. In *Proceedings ICANN'99, Int. Conf. on Artificial Neural Networks*, Berlin, 1999. Springer.

S. Sonnenburg, G. Rätsch, A. Jagota, and K.-R. Müller. New methods for splice-site recognition. submitted to ICANN'02, 2002.

R.J. Tibshirani. Regression selection and shrinkage via the LASSO. Technical report, Department of Statistics, University of Toronto, June 1994. ftp://utstat.toronto.edu/pub/tibs/lasso.ps.

A.N. Tikhonov and V.Y. Arsenin. *Solutions of Ill-posed Problems*. W.H. Winston, Washington, D.C., 1977.

T.Poggio and F. Girosi. Extensions of a theory of networks for approximation and learning: Dimensionality reduction and clustering. Technical Report AIM-1167, MIT-AILab, March 1990.

V. Tresp. Committee machines. In Y. Hu and J.-N. Hwang, editors, *Handbook on neural Network Signal Processing*. CRC Press, 2001.

K. Tsuda, M. Kawanabe, G. Rätsch, S. Sonnenburg, and K.R. Müller. A new discriminative kernel from probabilistic models. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural information processings systems*, volume 14, 2002. In press.

K. Tsuda, G. Rätsch, S. Mika, and K.-R. Müller. Learning to predict the leave-one-out error of kernel based classifiers. In G. Dorffner, H. Bischof, and K. Hornik, editors, *Artificial Neural Networks — ICANN'01*, pages 331–338. Springer Lecture Notes in Computer Science, Vol. 2130, 2001.

L.G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.

P. Vannerem, K.-R. Müller, A.J. Smola, B. Schölkopf, and S. Söldner-Rembold. Classifying lep data with support vector algorithms. In *Proceedings of AIHENP'99*, 1999.

V.N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, New York, 1995.

V.N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

V.N. Vapnik and A.Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probab. and its Applications*, 16(2):264–280, 1971.

V.N. Vapnik and A.Y. Chervonenkis. *Theory of Pattern Recognition [in Russian]*. Nauka, Moscow, 1974. (German Translation: W. Wapnik & A. Tscherwonenkis, *Theorie der Zeichenerkennung*, Akademie-Verlag, Berlin, 1979).

J. von Neumann. Zur Theorie der Gesellschaftsspiele. *Math. Ann.*, 100:295–320, 1928.

G. Wahba. *Splines Models for Observational Data*. Series in Applied Mathematics, Vol. 59, SIAM, Philadelphia, 1990.

G. Wahba. Support vector machines, reproducing hilbert spaces and the randomized gacv. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 69–88, Cambridge, MA, 1999. MIT Press.

M.K. Warmuth, G. Rätsch, M. Mathieson, J. Liao, and C. Lemmen. Active learning in the drug discovery process. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural information processings systems*, volume 14, 2002. In press.

A.S. Weigend and N.A. Gershenfeld, editors. *Time Series Prediction: Forecasting the Future and Understanding the Past*. Addison-Wesley, 1994. Santa Fe Institute Studies in the Sciences of Complexity.

J. Weston. LOO-Support Vector Machines. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Proc. IJCAI*, pages 727–733, 1999.

R.C. Williamson, A.J. Smola, and B. Schölkopf. Generalization performance of regularization networks and support vector machines via entropy numbers of compact operators. NeuroCOLT Technical Report NC-TR-98-019, Royal Holloway College, University of London, UK, 1998. To appear in *IEEE Transactions on Information Theory*.

R.C. Williamson, A.J. Smola, and B. Schölkopf. A maximum margin miscellany. Preprint, 1999.

K. Yoshimoto and Y. Nakano. Non-intrusive load monitoring system part i: Identification of inverter-driven appliances by a neual network. Technical report, Central Institute of Electric Power Industry, 1999. (in Japanese).

R. Zemel and T. Pitassi. A gradient-based boosting algorithm for regression problems. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13, pages 696–702. MIT Press, 2001.

T. Zhang. A general greedy approximation algorithm with applications. In *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2002. in press.

A. Zien, G. Rätsch, S. Mika, B. Schölkopf, T. Lengauer, and K.-R. Müller. Engineering Support Vector Machine Kernels That Recognize Translation Initiation Sites. *BioInformatics*, 16(9):799–807, September 2000.

Appendix A Proofs

A.1 Proofs from Chapter 2

This lemma will be used by the following proofs:

Lemma A.1 (Schapire et al. [1998], Schapire and Singer [1999]). *Let $\boldsymbol{\alpha}$ be the vector hypothesis coefficients satisfying $\alpha_t \geq 0$, $t = 1, \ldots, T$. Then holds*

$$\frac{1}{N}\sum_{n=1}^{N}\mathbf{I}\left(\rho_n(\boldsymbol{\alpha}) \leq \theta\right) \leq \left(\prod_{t=1}^{T} Z_t\right)\exp\left\{\sum_{t=1}^{T}\theta\alpha_t\right\} = \exp\left\{\sum_{t=1}^{T}(\theta\alpha_t + \log Z_t)\right\} \quad \text{(A.1)}$$

where Z_t is as defined in (1.9) and (2.5).

A.1.1 Proof of Lemma 2.1 from page 24

With (2.6), the smallest value for α_t is achieved for $\hat{\gamma}_t = \gamma + c$. Then we have $\alpha_t \geq \frac{1}{2}\log\frac{1+\gamma+c}{1-\gamma-c} - \frac{1}{2}\log\frac{1+\gamma}{1-\gamma}$. The minimizing γ for this expression is $\gamma = -\frac{1}{2}c$. Plugging in yields $\alpha_t \geq \frac{1}{2}\log\frac{1+c/2}{1-c/2} - \frac{1}{2}\log\frac{1-c/2}{1+c/2}$. Using the convexity and the Taylor expansion of the expression on the rhs. we can bound $\alpha_t \geq c$. ∎

A.1.2 Proof of Lemma 2.2 from page 23

Using $h_t(\boldsymbol{x}_n) \in \{-1, +1\}$, (2.4) is equivalent to (2.6). We plug-in $\hat{\gamma}_t \geq \varrho + b_t$ to (2.6), where $b_t = \frac{c}{t}$. The resulting function in b_t is convex for $\varrho + b_t \geq 0$. We can therefore use the Taylor expansion to lower bound $\hat{\alpha}_t$ by $\frac{1}{1-\varrho^2}b_t$. Plugging in the definition of b_t, yields $\hat{\alpha}_t \geq \frac{c}{t(1-\varrho^2)}$. Since $\sum_{t=1}^{\infty}\frac{1}{t}$ is divergent, we can conclude that $\sum_{r=1}^{t}\hat{\alpha}_r \overset{t\to\infty}{\longrightarrow} \infty$. ∎

A.1.3 Proof of Lemma 2.3 from page 24

Fix any $-1 < \theta < \varrho$. By Lemma A.1 holds

$$\begin{aligned}\frac{1}{N}\sum_{n=1}^{N}\mathbf{I}\left(\rho_n(\boldsymbol{\alpha}^T) \leq \theta\right) &\leq \exp\left\{\sum_{t=1}^{T}(\theta-\varrho)\alpha_t\right\}\exp\left\{\sum_{t=1}^{T}(\varrho\alpha_t + \log Z_t)\right\} \\ &\leq \exp\left\{\sum_{t=1}^{T}(\theta-\varrho)\alpha_t\right\}, \quad \text{(A.2)}\end{aligned}$$

where the last inequality holds follows from the fact that α_t is chosen in each iteration such that $\varrho\alpha_t + \log Z_t \leq 0$ is minimized. It can therefore not be worse than for $\hat{\alpha}^{(t)} = 0$, $t = 1, \ldots, T$, which would lead to an equality. Since $\sum_t \alpha_t \to \infty$ and $\theta < \varrho$ we can conclude that (A.2) converges to zero. Since it holds for any $\theta < \varrho$, there can asymptotically not exist an example with margin smaller than ϱ. Thus, $\lim_{t\to\infty} \rho_n(\boldsymbol{\alpha}^{(t)}) \geq \varrho$. ■

A.1.4 Proof of Lemma 2.4 from page 24

We use (A.5) for $\varrho = \theta$. Then holds

$$\begin{aligned}\frac{1}{N}\sum_{n=1}^{N} \mathbf{I}\left(y_n f(\boldsymbol{x}_n) \leq \theta\right) &\leq \exp\left\{-\sum_{t=1}^{T} \Delta_2(\varrho, \hat{\gamma}_t)\right\} \\ &\leq \exp\left\{-\sum_{t=1}^{T} \frac{1}{2}(\varrho - \hat{\gamma}_t)^2\right\} \leq \exp\left\{-\sum_{t=1}^{T} \frac{c^2}{2t}\right\} \leq 0,\end{aligned}$$

where we used a lower bound on the binary entropy, the assumption $\hat{\gamma}_t \geq \varrho + \frac{c}{\sqrt{t}}$ and that $\sum_{t=1}^{\infty} \frac{c}{t}$ is divergent for $c > 0$. ■

A.1.5 Proof of Theorem 2.2 from page 26

By Lemma A.1 holds

$$\frac{1}{N}\sum_{n=1}^{N} \mathbf{I}\left(\rho_n(\boldsymbol{\alpha}) \leq \theta\right) \leq \exp\left\{\sum_{t=1}^{T} (\theta - \varrho)\alpha_t + \varrho\alpha_t + \log Z_t\right\}. \tag{A.3}$$

We upper bound[1] Z_t by $\hat{Z}_t = \frac{1+\hat{\gamma}_t}{2}\exp(-\alpha_t) + \frac{1-\hat{\gamma}_t}{2}\exp(\alpha_t)$ [Schapire and Singer, 1999, Section 3.1]

$$(A.3) \leq \exp\left\{\sum_{t\in\mathcal{T}} (\theta - \varrho)\alpha_t + \varrho\alpha_t + \log \hat{Z}_t\right\}, \tag{A.4}$$

and let α_t be the minimizer of $\varrho\alpha_t + \log \hat{Z}_t$. That is $\alpha_t = \frac{1}{2}\log\frac{(1-\varrho)(1+\hat{\gamma}_t)}{(1+\varrho)(1-\hat{\gamma}_t)}$. Plugging in yields: $\varrho\alpha_t + \log \hat{Z}_t = -\Delta_2(\varrho, \hat{\gamma}_t)$, where $\Delta_2(\cdot,\cdot)$ is the the binary relative entropy defined

1. If the base hypothesis outputs are discrete this bound is exact. Note that this proof also holds for real valued outputs, if (2.6) is used instead of (2.4) to determine the hypothesis coefficients.

as $\Delta_2(\varrho, \hat{\gamma}_t) := \frac{1+\varrho}{2}\log\frac{1+\varrho}{1+\hat{\gamma}_t} + \frac{1-\varrho}{2}\log\frac{1-\varrho}{1-\hat{\gamma}_t}$. Hence,

$$
\begin{aligned}
(A.4) &= \exp\left\{\sum_{t=1}^{T}(\theta-\varrho)\alpha_t - \Delta_2(\varrho,\hat{\gamma}_t)\right\} \\
&= \exp\left\{\sum_{t=1}^{T}(\theta-\varrho)\frac{1}{2}\log\frac{(1-\varrho)(1+\hat{\gamma}_t)}{(1+\varrho)(1-\hat{\varrho}_t)} + \frac{1+\varrho}{2}\log\frac{1+\hat{\gamma}_t}{1+\varrho} + \frac{1-\varrho}{2}\log\frac{1-\hat{\gamma}_t}{1-\varrho}\right\} \\
&= \exp\left\{\sum_{t=1}^{T}\frac{1+\theta}{2}\log\frac{1+\hat{\gamma}_t}{1+\varrho} + \frac{1-\theta}{2}\log\frac{1-\hat{\gamma}_t}{1-\varrho}\right\}, \qquad (A.5)
\end{aligned}
$$

which proves the theorem. ∎

A.1.6 Proof of Corollary 2.2 from page 26

We start with (2.10) in Theorem 2.2. The maximum of $\left(\frac{1-\hat{\gamma}_t}{1-\varrho}\right)^{1-\theta}\left(\frac{1+\hat{\gamma}_t}{1+\varrho}\right)^{1+\theta}$ with respect to $\hat{\gamma}_t$ is obtained for $\hat{\gamma}_t = \theta$ and for $\theta \geq \hat{\gamma}_t \geq 1$ it is decreasing monotonically in $\hat{\gamma}_t$. We can therefore replace $\hat{\gamma}_t$ by $\hat{\gamma}$ in (2.10), i.e.

$$
\frac{1}{N}\sum_{n=1}^{N}\mathbf{I}\left(y_n f(\boldsymbol{x}_n) \leq \theta\right) \leq \left[\left(\frac{1-\hat{\gamma}}{1-\varrho}\right)^{1-\theta}\left(\frac{1+\hat{\gamma}}{1+\varrho}\right)^{1+\theta}\right]^{T/2}.
$$

If the basis on the right hand side is smaller than 1, then asymptotically we have $\frac{1}{N}\sum_{n=1}^{N}\mathbf{I}\left(y_n f(\boldsymbol{x}_n) \leq \theta\right) = 0$; this means that asymptotically, there is no example that has a smaller or equal margin than θ. The supremum $\theta_{\sup}$ over all θ such that the basis is less than 1 satisfies

$$
\left(\frac{1-\hat{\gamma}}{1-\varrho}\right)^{1-\theta_{\sup}}\left(\frac{1+\hat{\gamma}}{1+\varrho}\right)^{1+\theta_{\sup}} = 1\,.
$$

We can solve this equation to obtain $\theta_{\sup}$

$$
\theta_{\sup} = \frac{\log(1-\hat{\gamma}^2) - \log(1-\gamma^2)}{\log((1+\gamma)(1-\hat{\gamma})) - \log((1-\gamma)(1+\hat{\gamma}))}\,.
$$

Thus, for any $\theta \leq \theta^{\sup}$ holds $\frac{1}{N}\sum_{n=1}^{N}\mathbf{I}\left(y_n f(\boldsymbol{x}_n) \leq \theta\right) = 0$ (here we use $\hat{\gamma} > \varrho$). So there cannot exist an example with margin smaller that $\theta^{\sup}$. We have therefore shown that $\hat{\rho} \geq \theta_{\sup}$. ∎

A.1.7 Proof of Theorem 2.3 from page 31

See Algorithm 2.3 for definitions of $u_r, l_r, \hat{\gamma}_r, \hat{\varrho}_r$.

We claim that in any iteration $u_r \geq \varrho^* \geq l_r$. We show, if $u_{r-1} \geq \varrho^* \geq l_{r-1}$, then $u_r \geq \varrho^* \geq l_r$ for all $r = 1, \ldots, R$. It holds $l_1 \geq \varrho^* \geq u_1$ (induction start). By assumption $\hat{\gamma}_{t,r} \geq \varrho^*$ and we may set $u_{r+1} = \min_{r_0=1,\ldots,r}\min_{t=1,\ldots,T}\hat{\gamma}_{t,r}$. By Theorem 2.1 holds $\varrho^* \geq \hat{\varrho}_r$ for all $r = 1, 2, \ldots$ and, hence, we may set $l_{r+1} = \max_{r_0=1,\ldots,r}\hat{\varrho}_{r_0}$. We have to consider two cases. (a) $\hat{\varrho}_r \geq \varrho_r$ and (b) $\hat{\varrho}_r < \varrho_r$. In case (b) we have an additional term in

u_{r+1}, which follows from $\varrho_r + \varepsilon \geq \varrho^*$, justified by $\hat{\varrho}_r < \varrho_r$ and Theorem 2.2.

By construction, the length of interval $[l_r, u_r]$ is (almost) decreased in each iteration by a factor of two. We show $u_r - l_r \leq 2^{-r+1} + 2\varepsilon$. In case (a) the interval is reduced by at least a factor of two. The worst case is if always (b) happens:

$$\begin{aligned} u_r - l_r &\leq \varrho_r + \varepsilon - l_r \\ &= \frac{\varrho_{r-1} + \varepsilon + l_{r-1}}{2} + \varepsilon - l_r \\ &= \frac{\varrho_{r-1} + \varepsilon + l_{r-1} + 2\varepsilon - 2l_r}{2} \\ &= \frac{\varrho_{r-j} + \varepsilon \sum_{i=0}^{j} 2^j + \sum_{i=1}^{j} 2^{j-i} l_{r-i} - 2^j l_r}{2^j} \\ &= \frac{\varrho_{r-j} + (2^{j+1} - 1)\varepsilon + \sum_{i=1}^{j} 2^{j-i} l_{r-i} - 2^j l_r}{2^j} \\ &= \frac{\varrho_1 + (2^r - 1)\varepsilon + \sum_{i=1}^{r-1} 2^{r-1-i} l_{r-i} - 2^{r-1} l_r}{2^{r-1}}. \end{aligned}$$

Since l_r is non-decreasing, $\sum_{i=1}^{j} 2^{r-1-i} l_{r-i} - 2^{r-1} l_r$ is maximized for $l_r = \text{const}$, i.e. $\sum_{i=1}^{j} 2^{r-1-i} l_{r-i} - 2^{r-1} l_r \leq l_r \sum_{i=0}^{r-2} 2^i - 2^{r-1} l_r = l_r(2^{r-1} - 1 - 2^{r-1}) = -l_r$. We continue by using $\varrho_1 = 0$ and $\min_r l_r = -1$:

$$\begin{aligned} &\leq \frac{(2^r - 1)\varepsilon - l_r}{2^{r-1}} \\ &\leq \frac{1}{2^{r-1}} + 2\varepsilon. \end{aligned}$$

Thus after $R = \lceil \log_2(1/\varepsilon) \rceil$ steps we have $u_{R+1} - l_{R+1} \leq 3\varepsilon$ and $\varrho^* - l_{R+1} \leq 3\varepsilon$.

Now we run AdaBoost $_{l_{R+1} - \varepsilon}(S, 2\log(N)/\varepsilon^2)$ and achieve a margin of at least $l_{R+1} - \varepsilon$ by Corollary 2.3. This can only be 4ε away from ϱ^*.

We called $R + 1 = \lceil \log_2(1/\varepsilon) + 1 \rceil$ times Algorithm 2.2, each time calling $\lceil 2\log(N)/\varepsilon^2 + 1 \rceil$ times the base learning algorithm. Algorithm 2.3 returns only the last hypothesis, combining only $\lceil 2\log(N)/\varepsilon^2 + 1 \rceil$ base hypotheses. ■

A.2 Proofs from Chapter 3

A.2.1 Proof of Proposition 3.2 from page 50

We need the following:

Lemma A.2. *Let $q : S \to \mathbb{R}$ be a strictly convex, twice continuously differentiable function on an open, convex set $S \subseteq \mathbb{R}$. Assume q' is L-Lipschitz differentiable on S. Assume $x^* = \operatorname{argmin}_{x \in S} q(x) \in S$ exists. Then holds for any $x \in S$:*

$$q(x) - q(x^*) \geq \frac{|q'(x)|^2}{4L} \tag{A.6}$$

Proof Let $q'(x) = \frac{\partial q(x)}{\partial x}$. Since q is twice continuously differentiable and L-Lipschitz differentiable, it holds $|q'(x) - q'(y)| \leq L|x - y|$ for all $x, y \in S$.

Fix $x \in S$ and let $\beta^* = x^* - x$. We have $q'(x^*) = 0$ since x^* is optimal. Furthermore $q'(x)q'(x + \beta) \geq 0$ for all β satisfying $\beta\beta^* \geq 0$ and $|\beta| \leq |\beta^*|$. Thus

$$|q(x) - q(x + \beta)| = \left| \int_x^{x+\beta} q'(\tilde{x}) d\tilde{x} \right| = \operatorname{sign}(\beta) \int_x^{x+\beta} |q'(\tilde{x})| d\tilde{x}.$$

Since q is strictly convex, q' is strictly monotone. Hence

$$\begin{aligned} \operatorname{sign}(\beta) \int_x^{x+\beta} |q'(\tilde{x})| d\tilde{x} &\geq \operatorname{sign}(\beta) \int_x^{x+\beta} |q'(x + \beta)| d\tilde{x} \\ &= \beta \operatorname{sign}(\beta) |q'(x + \beta)| \\ &= |\beta| \, |q'(x + \beta)| \end{aligned}$$

Moreover, we have by optimality and assumption

$$|q'(x) - q'(x^*)| = |q'(x)| \leq L|\beta^*|.$$

Thus $|\beta^*| \geq \frac{|q'(x)|}{L}$. Let $\hat{\beta}$ such that $\hat{\beta}\beta^* \geq 0$ and $|\hat{\beta}| = \frac{|q'(x)|}{2L}$. From the Lipschitz differentiability holds

$$|q'(x) - q'(x + \hat{\beta})| \leq L|\hat{\beta}| = \frac{|q'(x)|}{2}$$

Since $q'(x)$ and $q'(x + \hat{\beta})$ have the same sign and $|q'(x)| \geq q'(x + \hat{\beta})$ holds

$$|q'(x + \hat{\beta})| \geq \frac{1}{2}|q'(x)|.$$

We now combine the results:

$$\begin{aligned} q(x) - q(x^*) &\geq q(x) - q(x + \hat{\beta}) \\ &= |q(x) - q(x + \hat{\beta})| \\ &\geq |\hat{\beta}| \, |q'(x + \hat{\beta})| \\ &\geq \frac{1}{2}|\hat{\beta}| \, |q'(x)| = \frac{|q'(x)|^2}{4L} \end{aligned}$$

■

We define the function $q^{(t)}(\hat{\alpha}_t) = \mathrm{G}\left(\tilde{\mathrm{g}}(\mathbf{d}^{(t)}) + \hat{\alpha}_t \mathbf{h}^{(t)}\right)$. It is strictly convex (if $\mathbf{h}^{(t)} \neq \mathbf{0}$) and continuously differentiable, since it is the conjugate of a Bregman function and $\mathbf{h}^{(t)}$ finite. By assumption on g and finiteness of $\mathbf{h}^{(t)}$, we have that q is twice continously differentiable. By zone consistency holds $|\hat{\boldsymbol{\alpha}}| < \infty$ (otherwise $\mathbf{d}^{(t+1)} \in \overline{\mathcal{S}} \setminus \mathcal{S}$. We can conclude that the minimizing $\hat{\alpha}_t$, say $\hat{\alpha}_t^*$, is unique and finite for any $\mathbf{d}^{(t)} \in \mathcal{S}$ and $\mathbf{h}^{(t)} \neq 0$. By convexity of q, the minimum depends continuously on $\mathbf{d}^{(t)}$.

We can assume that $\mathbf{h}^{(t)} \neq \mathbf{0}$. Otherwise we would have reached the optimum of (3.15) (by (3.23) and the properties of τ). Hence, without loss of generality we can assume in the sequel that $\|\mathbf{h}^{(t)}\|_2 = 1$. By scaling $\hat{\alpha}_t^*$ one obtains identical results for $\|\mathbf{h}^{(t)}\|_2 \neq 1$.

Let $\mathbf{h} \in \mathbb{R}^N$ be arbitrary with $\|\mathbf{h}\|_2 = 1$. Let $S^{\mathbf{h}} = (0 \pm \epsilon_{\mathbf{h}}, \hat{\alpha}^*_{t,\mathbf{h}} \pm \epsilon_{\mathbf{h}})$ be the open set that contains the interval $[0, \hat{\alpha}^*_{t,\mathbf{h}}]$, where $\hat{\alpha}^*_{t,\mathbf{h}}$ is the step size when starting from $\mathbf{d}^{(t)}$ going in direction $\mathbf{h}$ to the minimum in this direction. The parameter $\epsilon_{\mathbf{h}} > 0$ is chosen small enough such that $\{\mathrm{g}\left(\tilde{\mathrm{g}}(\mathbf{d}^{(t)}) + \hat{\alpha}\mathbf{h}^{(t)}\right) \mid \hat{\alpha} \in S\} \subset \mathcal{S}$. Such $\epsilon_{\mathbf{h}}$ exists since $\mathbf{d}^{(t+1)} \in \mathcal{S}$ by the assumed zone-consistency of $\mathbf{h}^t$.

For any such $\mathbf{h}$ there exists a constant $L^{\mathbf{h}}$ such that $q'^{\mathbf{h}}$ is $L^{\mathbf{h}}$-Lipschitz differentiable on $S^{\mathbf{h}}$. Note that $L^{\mathbf{h}}$ depends continously on $\mathbf{d}^{(t)}$, since $\hat{\alpha}^*_{t,\mathbf{h}}$ and therefore also $S^{\mathbf{h}}$ depend continuously on $\mathbf{d}^{(t)}$ and $q'^{\mathbf{h}}$ is continously differentiable.

We define $L(\mathbf{d}^{(t)}) := \max_{\mathbf{h}:\|\mathbf{h}\|_2=1} L^{\mathbf{h}}$. The maximum exists. Note that $L(\mathbf{d}^{(t)})$ continuously depends on $\mathbf{d}^{(t)}$, since taking the maximum of continuous functions yields a continuous function. Note that L is non-zero by strict monotonicity of q' and $S^{(t)} \neq \emptyset$ since $\epsilon_t > 0$.

Let $L^{(t)}$ be the Lipschitz constant for $q^{(t)}$ on $S^{\mathbf{h}^{(t)}}$. Then holds $L^{(t)} \leq L(\mathbf{d}^{(t)})$. We have $q'(0) = \langle \mathrm{g}(\tilde{\mathrm{g}}(\mathbf{d}^{(t)})), \mathbf{h}^{(t)} \rangle = \langle \mathbf{d}^{(t)}, \mathbf{h}^{(t)} \rangle$. By Lemma A.2 holds $q(0) - q(\hat{\alpha}^*_t) = \mathrm{G}\left(\tilde{\mathrm{g}}(\mathbf{d}^{(t)})\right) - \mathrm{G}\left(\tilde{\mathrm{g}}(\mathbf{d}^{(t)}) + \hat{\alpha}^*_t \mathbf{h}^{(t)}\right) \geq \frac{|\langle \mathbf{d}^{(t)}, \mathbf{h}^{(t)} \rangle|^2}{4L^{(t)}} \geq \frac{|\langle \mathbf{d}^{(t)}, \mathbf{h}^{(t)} \rangle|^2}{4L(\mathbf{d}^{(t)})}$. By (3.23) holds $|\langle \mathbf{d}^{(t)}, \mathbf{h}^{(t)} \rangle| \geq \tau(\mathcal{A}^\nabla(\mathbf{d}^{(t)}))$. Thus, by (3.19) we can conclude $\Delta_{\mathrm{G}}(\mathbf{0}, \mathbf{d}^{(t)}) - \Delta_{\mathrm{G}}(\mathbf{0}, \mathbf{d}^{(t+1)}) \geq \frac{\tau(\mathcal{A}^\nabla(\mathbf{d}^{(t)}))^2}{4L(\mathbf{d}^{(t)})}$.

Condition (3.17) can easily be verified by observing $L(\mathbf{d}^{(t)}) < \infty$. Since τ and A^∇ are continuous, $\tau(A^\nabla(\cdot))$ is also continuous. We have already shown that $\mathfrak{L}(\mathbf{d}^{(t)})$ is continuous and always greater than zero. Therefore $\frac{\tau(\mathcal{A}^\nabla(\mathbf{d}^{(t)}))}{4L(\mathbf{d}^{(t)})}$ is an auxiliary function for the sequence $\{\mathbf{d}^{(t)}\}$ and H. ∎

A.2.2 Proof of Proposition 3.3 from page 53

We first bound $\lambda^*_j := |\hat{\alpha}^{(t+1,j)}_j - \hat{\alpha}^{(t)}_j|$. By condition 3 we have

$$\left| \frac{\partial \mathrm{G}(H\hat{\boldsymbol{\alpha}}^{(t)})}{\partial \hat{\alpha}_j} - \frac{\partial \mathrm{G}(H\hat{\boldsymbol{\alpha}}^{(t)} + \lambda_j H_j)}{\partial \hat{\alpha}_j} \right| \geq |\lambda_j| \eta_l .$$

By (3.29) holds $\frac{\partial \mathrm{G}(H\hat{\boldsymbol{\alpha}}^{(t)} + \lambda^*_j H_j)}{\partial \hat{\alpha}_j} + \gamma_j = 0$ and we have $\lambda^*_j \leq \frac{1}{\eta_l} \left| \frac{\partial \mathrm{G}(H\hat{\boldsymbol{\alpha}}^{(t)})}{\partial \hat{\alpha}_j} + \gamma_j \right|$, $j = 1, \ldots, J$. By a symmetric argument we also have $\lambda^*_{j_t} \geq \frac{1}{\eta_u} \left| \frac{\partial \mathrm{G}(H\hat{\boldsymbol{\alpha}}^{(t)})}{\partial \hat{\alpha}_{j_t}} + \gamma_j \right|$, for all $j = 1, \ldots, J$. Condition 1 implies

$$\frac{|\hat{\alpha}^{(t+1,j_t)}_{j_t} - \hat{\alpha}^{(t)}_{j_t}|}{|\hat{\alpha}^{(t+1,j)}_j - \hat{\alpha}^{(t)}_j|} = \frac{|\lambda^*_{j_t}|}{|\lambda^*_j|} \geq \frac{\frac{1}{\eta_u} \left| \frac{\partial \mathrm{F}(\hat{\boldsymbol{\alpha}}^{(t)})}{\partial \hat{\alpha}_{j_t}} \right|}{\frac{1}{\eta_l} \left| \frac{\partial \mathrm{F}(\hat{\boldsymbol{\alpha}}^{(t)})}{\partial \hat{\alpha}_j} \right|} \geq \frac{\delta \eta_l}{\eta_u} \qquad \forall j = 1, \ldots, J.$$

Thus, (3.28) is satisfied for $\beta = \frac{\delta \eta_l}{\eta_u} > 0$ and we can apply Theorem 3.2. ∎

A.2.3 Proof of Proposition 3.4 from page 53

Let $\Delta_j = \mathrm{F}(\hat{\boldsymbol{\alpha}}^{(t)}) - \mathrm{F}(\hat{\boldsymbol{\alpha}}^{(t+1,j)}) \geq 0$ and $\partial_j = \left| \frac{\partial \mathrm{G}(H\hat{\boldsymbol{\alpha}}^{(t)})}{\partial \hat{\alpha}_j} + \gamma_j \right|$. From the proof of Proposition 3.3 we have $\lambda^*_j \leq \frac{1}{\eta_l} \partial_j$ and $\lambda^*_j \geq \frac{1}{\eta_u} \partial_j$ for all $j = 1, \ldots, J$.

We first bound Δ_j in terms of ∂_j in both directions:

1. By condition 3 we have $\left|\left|\frac{\partial \mathrm{G}(H\hat{\boldsymbol{\alpha}}^{(t)} + \lambda H_j)}{\partial \hat{\alpha}_j} + \gamma_j\right| - \partial_j\right| \leq |\lambda|\eta_u$ for all λ and $j = 1, \ldots, J$. Since the gradient does not change the sign for smaller steps than λ_j^*, holds $\left|\frac{\partial \mathrm{G}(H\hat{\boldsymbol{\alpha}}^{(t)} + \lambda H_j)}{\partial \hat{\alpha}_j} + \gamma_j\right| \geq \partial_j - \eta_u|\lambda|$ for all $|\lambda| \leq \lambda_j^*$.
It holds $\Delta_j \geq \left|\int_0^{\lambda_j} \left|\frac{\partial \mathrm{G}(H\hat{\boldsymbol{\alpha}}^{(t)} + \lambda H_j)}{\partial \lambda} + \gamma_j\right| \mathrm{d}\lambda\right|$ for all $|\lambda_j| \leq \lambda_j^*$ and with equality if $\lambda_j = \hat{\alpha}_j^{(t)} - \hat{\alpha}_j^{(t+1,j)}$. Thus

$$\Delta_j \geq \int_0^{|\lambda_j|} (\partial_j - \eta_u|\lambda|)\mathrm{d}\lambda = \partial_j|\lambda_j| - \frac{\eta_u}{2}|\lambda_j|^2 \tag{A.7}$$

for all $|\lambda| \leq \lambda_j^*$. Using $\lambda_j^* \geq \frac{1}{\eta_u}\partial_j$, we may set $|\lambda_j| = \frac{\partial_j}{\eta_u}$ and get $\Delta_j \geq \frac{\partial_j^2}{\eta_u} - \frac{\partial_j^2}{2\eta_u} = \frac{\partial_j^2}{2\eta_u}$ and therefore $\partial_j \leq \sqrt{2\Delta_j\eta_u}$.
2. Furthermore, by convexity holds $\Delta_j \leq \lambda_j^*\partial_j$ and therefore $\partial_j \geq \sqrt{\Delta_j\eta_l}$ for all $j = 1, \ldots, J$ – in particular for j_t.

Using both inequalities and assumption 1, we can bound for all $j = 1, \ldots, J$:

$$\frac{\partial_{j_t}}{\partial_j} \geq \frac{\sqrt{\Delta_{j_t}\eta_l}}{\sqrt{2\Delta_j\eta_u}} \geq \frac{\sqrt{\delta\eta_l}}{\sqrt{2\eta_u}}. \tag{A.8}$$

Hence, we can apply Proposition 3.3 and conclude that (3.28) is satisfied for $\beta = \sqrt{\delta}\frac{\eta_l^{3/2}}{\sqrt{2}\eta_u^{3/2}}$. ∎

A.2.4 Proof of Theorem 3.5 from page 56

Let $\mathbb{S}^*$ be the solution set of (3.8) and

$$\mathbb{S}^b = \mathrm{conv}\{\hat{\boldsymbol{\alpha}}^{(t,j)} \mid j = 1, \ldots, J, t = 1, 2, \ldots\}.$$

We claim $\mathbb{S}^b$ is bounded. By the assumption on the finiteness of the $\hat{\boldsymbol{\alpha}}^{(t)}$'s, the set $\mathrm{conv}\{\hat{\boldsymbol{\alpha}}^{(t)} \mid t = 1, 2, \ldots\}$ is bounded. We therefore need to show that for any finite $\hat{\boldsymbol{\alpha}}^{(t)}$, the vectors $\hat{\boldsymbol{\alpha}}^{(t+1,j)}$, $j = 1, \ldots, J$ are each finite. We consider loss functions of the form $\mathrm{G}(\mathbf{o}) = \sum_{n=1}^N g_{y_n}(o_n)$, where $\mathrm{g}_{y_n}(o)$ is strongly convex at finite o. There are two cases: g_{y_n} grows (1) in both directions to infinity, i.e. $g_{y_n}(\pm\infty) = \infty$, and (2) it grows only in one direction to infinity. The third case that it grows in neither direction to ∞ cannot happen, since g_{y_n} is strongly convex.

(1) If $\hat{\boldsymbol{\alpha}}$ is finite, $\mathrm{G}(H\hat{\boldsymbol{\alpha}})$ is finite. Denote by H_j the j-th column of H. A single step in one direction j cannot be optimal at infinity, since $H_j \neq \mathbf{0}$ by assumption and so $H\hat{\boldsymbol{\alpha}}$ and therefore $\mathrm{G}(H\hat{\boldsymbol{\alpha}})$ would be $+\infty$. Thus, the infinite step cannot be optimal.

(2) Let

$$\hat{y}_n := \begin{cases} +1 & g_{y_n}(-\infty) = \infty \\ -1 & g_{y_n}(+\infty) = \infty \end{cases},$$

then $\hat{\alpha}_j^{(t+1,j)}$ can only be infinite, if $H_{j,n}\hat{y}_n \geq 0$ for all $n = 1, \ldots, N$. Otherwise the objective would grow to infinity. Since g_{y_n} is monotonically decreasing in direction $\hat{y}_n \infty$, any solution $\hat{\boldsymbol{\alpha}}^*$ does not get worse when increasing $\hat{\alpha}_j^*$ to infinity. Thus there must exist a infinite solution, which is a contradiction to the assumption.

Since this holds for any finite $\hat{\boldsymbol{\alpha}}$, it also holds for any limit point of $\{\hat{\boldsymbol{\alpha}}^{(t)}\}$ and any solution. We can therefore conclude that $\mathcal{S}^b$ is bounded. Then holds condition 2 of Proposition 3.3.

Each element H_{nj} of H is finite. Thus the set $\mathcal{S}^o = \{H\hat{\boldsymbol{\alpha}} \mid \hat{\boldsymbol{\alpha}} \in \mathcal{S}^b\}$ is bounded. Since G is strongly convex on $\mathcal{S}^o$ and $\|\nabla^2 G(\mathbf{o})\|$ is bounded by assumption, condition 3 of Proposition 3.3 is satisfied.

By Definition 3.5 and using the complementation closeness of H, we may apply Proposition 3.3 or Proposition 3.4 and can conclude that the sequence $\hat{\boldsymbol{\alpha}}^0, \hat{\boldsymbol{\alpha}}^{(t)}, \ldots$ satisfies (3.28).

Since G is strongly convex on $\mathcal{S}^*$, the Hessian $\nabla^2 \mathrm{G}(H\hat{\boldsymbol{\alpha}}^*)$ is a strictly positive matrix for all $\hat{\boldsymbol{\alpha}}^* \in \mathcal{S}^*$. Thus, we can apply Theorems 3.2 & 3.3 and have proven the theorem. ■

A.2.5 Proof of Corollary 3.1 from page 56

We first need to show that the loss functions of AdaBoost, Logistic regression and LS-Boost are strongly convex on any bounded subset $\mathcal{S}^o$ of $\mathbb{R}^N$ and fulfill the conditions in Theorem 3.2: The loss functions can be written as $\mathrm{G}(\mathbf{o}) = \sum_{n=1}^N g(y_n, o_n)$. Therefore, the Hessian $\nabla^2 \mathrm{G}(\mathbf{o})$ is diagonal with elements $\frac{\partial^2}{\partial o_n^2} g(y_n, o_n)$, $n = 1, \ldots, N$. Since the second derivatives of the one dimensional functions (cf. Section 3.1.3) can each be lower bounded by a positive constant on a bounded subset of $\mathbb{R}$, each element of the diagonal and therefore each eigenvalue is bounded away from 0. This implies the strong convexity of G. Using similar arguments one can also uniformly upper bound $\|\nabla^2 \mathrm{G}(\mathbf{o})\|$.
Thus, we can fulfill the conditions of Theorem 3.5. ■

A.3 Proofs from Chapter 4

A.3.1 Proof of Proposition 4.3 from page 70

By Figure 4.4, one can conclude that ρ^* is never changed for any local movements of a point $\hat{\boldsymbol{x}}_n$ satisfying $\xi_n > 0$. Moreover it easily seen that d_n^* does not change (it will still be at the upper bound $1/(\nu N)$) [cf. Smola et al., 1999]. To show that $\hat{\boldsymbol{\alpha}}^*$ is not changing, consider the dual optimization problem of (4.12) [Rätsch and Warmuth, 2000, Problem

L3]:

$$\begin{array}{ll} \min\limits_{\mathbf{d},\gamma} & \gamma \\ \text{with} & U^\top \mathbf{d} \leq \gamma \mathbf{1} \\ & \mathbf{d} \geq \mathbf{0}, \sum_{n=1}^{N} d_n = 1, \\ & d_n \leq \frac{1}{\nu N} \qquad\qquad \text{for all } n = 1, \ldots, N \end{array} \tag{A.9}$$

where U is as defined in Section 1.3.3 (cf. (1.21)). We first proof statement 1a. Consider the set $\mathcal{J} \subseteq \{\mathbf{1}, \ldots, J\}$ of indices j, where the constraint $\langle U_j, \mathbf{d}^* \rangle = \langle H_j * \boldsymbol{y}, \mathbf{d}^* \rangle \leq \gamma$ is satisfied as equality ($*$ denotes the Hadamard product). If the movement is perpendicular to all coordinates $j \in \mathcal{J}$, there is no change in $\hat{h}_j(\boldsymbol{x}_n)$ for all $j \in \mathcal{J}$. If the step is small enough, then no other constraint becomes active (there are just a finite number of constraints). Thus the original solution is still valid. This proves statement 1a.

To prove statement 1b, consider the a translation in direction $\boldsymbol{\Delta}$: let $\overline{\boldsymbol{x}}_n = \hat{\boldsymbol{x}}_n + \epsilon \boldsymbol{\Delta}$ be the translated point ($\epsilon \in \mathbb{R}$), $\hat{U} = [y_1 \hat{\boldsymbol{x}}_1, \ldots, y_N \hat{\boldsymbol{x}}_N]$ and $\overline{U} = [y_1 \hat{\boldsymbol{x}}_1, \ldots, y_n \overline{\boldsymbol{x}}_n, \ldots, y_N \hat{\boldsymbol{x}}_N]$. Then we have: $\langle \overline{U}_j, \mathbf{d}^* \rangle = \langle \hat{U}_j, \mathbf{d}^* \rangle + \epsilon y_n^2 d_n^* = \langle \hat{U}_j, \mathbf{d}^* \rangle + \epsilon d_n^*$. If ϵ is small enough (but positive) the optimal $\mathbf{d}$ does not change, just γ^* changes by ϵd_n^*. Thus the dual objective is changed by $\frac{\epsilon}{\nu N}$. We claim that the primal variables $\hat{\boldsymbol{\alpha}}^*$ and ρ^* are still optimal after moving $\hat{\boldsymbol{x}}_n$. Since only the n-th row in U is changed, only ξ_n changes. Since $\sum_{j=1}^{N} \hat{\alpha}_j = 1$, the total change after movement is $-\epsilon$. Thus the primal objective changes by $\frac{\epsilon}{\nu N}$. Since, both the primal and dual objective have change by the same amount and the construction is still feasible in both domains, they are optimal. This proves statement 1b.

Note that both arguments can be applied independently and the claim holds for any combination of both translations. This proves the last claim in statement 1.

Now we prove statement 2 following the lines of reasoning in the proof of Proposition 5.1. By assumption we have $\xi_n > 0$, i.e. $y_n f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}_n) < \rho$. Changing the class membership of a point, i.e. $y_n' = -y_n$, does not change the status of $\boldsymbol{x}_n$ as being a point inside the margin area. By optimality of the old solution holds $y_n f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}_n) = \rho - \xi_n$. All we have to show that by changing ξ_n into $\xi_n' = \xi_n + 2 y_n f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}_n)$ and keeping all the other variables and therefore also the estimate f unchanged, we obtain an optimal solution again. By construction the new set of variables for the modified problem is still feasible and the same constraints are active as in the initial solution. Finally the gradients in both the objective function and the constraint with respect to each variable remain unchanged since ξ_n only appears in a linear fashion and all other variables did not change at all. Thus the new solution is optimal again, leading to the same estimate of $f_{\hat{\boldsymbol{\alpha}}}$ as before. This proves statement 2. ∎

A.3.2 Proof of Proposition 4.4 from page 75

We prove the convergence of Algorithm 4.7 with the modification described on page 75 (which we refer to as Algorithm 4.7') to the minimum of

$$\mathrm{F}[f_{\hat{\boldsymbol{\alpha}}}] = \mathrm{G}(H\hat{\boldsymbol{\alpha}}) + C \sum_{j=1}^{J} \hat{\alpha}_j \quad \text{with} \quad \hat{\boldsymbol{\alpha}} \geq \mathbf{0}. \tag{A.10}$$

We start with:

Lemma A.3. *Let C, S, G, H be as in Proposition 4.4. Then any solution to (A.10) is finite.*

Proof Suppose there would exists an infinite solution $\hat{\boldsymbol{\alpha}}^{\infty}$, then the objective of (A.10) must be $+\infty$ (we assumed $\gamma > 0$). Since the labels $\boldsymbol{y}_n$, $n = 1, \ldots, N$, are finite, also $\mathrm{G}(\mathbf{0})$ is finite. Furthermore, $\hat{\boldsymbol{\alpha}} = \mathbf{0}$ is feasible and $\hat{\boldsymbol{\alpha}}^{\infty}$ cannot be optimal. ∎

By the same arguments we can also conclude that the α_t's computed in Algorithm 4.7' and the $\hat{\alpha}_j^{(t,j)}$'s as in Theorem 3.2 are bounded. Therefore, the eigenvalues of the Hessian are lower bounded in the region of interest.

Extending the results of Section 3.3 for the GS scheme, we have the following result for coordinate descent on the positive quadrant:

Proposition A.1 (GS scheme on $\mathbb{R}_+^J$, Rätsch et al. [2002]). *Let $\mathcal{S} = \mathbb{R}_+^J := \{\hat{\boldsymbol{\alpha}} \mid \mathbf{0} \leq \hat{\boldsymbol{\alpha}} \in \mathbb{R}^J\}$ and $\mathcal{S}^b$ be a convex subset of $\mathcal{S}$. Then a coordinate selection $j_1, j_2, \ldots$ starting at $\hat{\boldsymbol{\alpha}}^0$ satisfies condition (3.28) of Theorem 3.2 for all iterations t, where (i) conditions 2–3 of Proposition 3.3 hold, (ii) $\hat{\alpha}_{j_t}^{(t)} \geq \frac{1}{\eta_u} \left| \frac{\partial \mathrm{F}(\hat{\boldsymbol{\alpha}}^{(t)})}{\partial \hat{\alpha}_{j_t}} \right|$ and (iii) the selection satisfies $j_t \in \mathcal{J}_t$ as well as the following inequality:*

$$\left| \frac{\partial \mathrm{F}(\hat{\boldsymbol{\alpha}}^{(t)})}{\partial \hat{\alpha}_{j_t}} \right| \geq \delta \max_{j \in \mathcal{J}_t} \left| \frac{\partial \mathrm{F}(\hat{\boldsymbol{\alpha}}^{(t)})}{\partial \hat{\alpha}_j} \right| \tag{A.11}$$

for some fixed $\delta \in (0, 1]$, where

$$\mathcal{J}_t = \left\{ j \;\middle|\; \left(\frac{\partial \mathrm{F}(\hat{\boldsymbol{\alpha}}^{(t)})}{\partial \hat{\alpha}_j} < 0 \right) \vee (\hat{\alpha}_j^{(t)} > 0) \right\}.$$

Sketch of Proof. The only difference to Proposition 3.3 is the additional constraint $\hat{\boldsymbol{\alpha}} \geq \mathbf{0}$. Thus, there can exist a dimension j with *strictly positive gradient* that is already optimal. We therefore must only take the positive gradients of those coefficients into account that are strictly positive (and therefore have been selected before – since we start at $\mathbf{0}$). Thus, in Proposition 3.3 we restrict the indices in (A.11) and can follow the proof of Proposition 3.3. However, if $\hat{\alpha}_{j_t}^{(t)} < \frac{1}{\eta_u} \frac{\partial \mathrm{F}(\hat{\boldsymbol{\alpha}}^{(t)})}{\partial \hat{\alpha}_{j_t}}$, then $\hat{\alpha}_{j_t}^{(t+1)} = 0$ and (3.28) might not be satisfied. But this case is excluded by assumption (ii). For more details see Rätsch et al. [2002]. ∎

Proposition A.2 (MI scheme on $\mathbb{R}_+^J$). *Let $\mathcal{S} = \mathbb{R}_+^J$ and $\mathcal{S}^b$ be a convex subset of $\mathcal{S}$. Then a coordinate selection $j_1, j_2, \ldots$ starting at $\hat{\boldsymbol{\alpha}}^0$ satisfies (3.28) for all iterations t, where (i) conditions 2–3 of Proposition 3.3 hold, (ii) $\hat{\alpha}_{j_t}^{(t)} \geq \frac{1}{\eta_u} \left| \frac{\partial \mathrm{F}(\hat{\boldsymbol{\alpha}}^{(t)})}{\partial \hat{\alpha}_{j_t}} \right|$, and (iii) the selection j_t satisfies*

$$\mathrm{F}(\hat{\boldsymbol{\alpha}}^{(t)}) - \mathrm{F}(\hat{\boldsymbol{\alpha}}^{(t+1,j_t)}) \geq \delta \max_{j=1,\ldots,J} \left(\mathrm{F}(\hat{\boldsymbol{\alpha}}^{(t)}) - \mathrm{F}(\hat{\boldsymbol{\alpha}}^{(t+1,j)}) \right) \tag{A.12}$$

for some fixed $\delta \in (0, 1]$.

Proof We follow the proof of Proposition 3.4. From the proof of Proposition 3.3 we have $\lambda_j^* \leq \frac{1}{\eta_l} \partial_j$ for all $j = 1, \ldots, J$. The property $\lambda_j^* \geq \frac{1}{\eta_u} \partial_j$ may not hold anymore, since one may hit the constraint $\hat{\alpha}_j \geq 0$. Item 1 in the proof of Proposition 3.4 holds also in our case

if $\lambda_j^* \geq \frac{1}{\eta_u}\partial_j$, whereas item 2 holds anyway.

We have by assumption $\lambda_{j_t}^* \geq \frac{\partial_j}{\eta_u}$. Hence $\Delta_{j_t} \geq \frac{\partial_{j_t}^2}{2\eta_u}$ by the first inequality (cf. item 1 for j_t). By item 2 holds $\Delta_j \leq \lambda_j^* \partial_j$ and therefore $\Delta_j \leq \frac{\partial_j^2}{\eta_l}$ for all $j = 1, \ldots, J$. Let j' such that $\lambda_{j'}^* < \frac{\partial_{j'}}{\eta_u}$. Then by (A.7) holds $\lambda_{j'} \leq \frac{2\Delta_{j'}}{\partial_{j'}}$ and by convexity we have $\Delta_{j'} \leq \frac{\partial_{j'}^2}{\eta_u}$. Hence,

$$\frac{\lambda_{j_t}^*}{\lambda_{j'}^*} \geq \frac{\Delta_{j_t}\partial_{j'}}{2\partial_{j_t}\Delta_{j'}} \geq \frac{\Delta_{j_t}\sqrt{\Delta_{j'}\eta_l}}{2\Delta_{j'}\sqrt{2\Delta_{j_t}\eta_u}} = \frac{\sqrt{\Delta_{j_t}\eta_l}}{2\sqrt{2\Delta_{j'}\eta_u}} \geq \frac{\sqrt{\delta\eta_l}}{2\sqrt{2\eta_u}},$$

for all j'. For all other indices (A.8) holds. Hence, we can conclude that (3.28) is satisfied for $\beta = \min\left(\sqrt{\delta\frac{\eta_l}{8\eta_u}}, \sqrt{\delta}\frac{\eta_l^{3/2}}{\sqrt{2}\eta_u^{3/2}}\right)$. ∎

Using the same arguments as in Lemma A.3, one can show that the sequences $\{\hat{\boldsymbol{\alpha}}^{(t,1)}\}, \ldots, \{\hat{\boldsymbol{\alpha}}^{(t,J)}\}$, $t = 1, 2, \ldots$, stay bounded. One may therefore define the bounded convex set $\mathcal{S}^o = \operatorname{conv}\{H\hat{\boldsymbol{\alpha}}^{(t,j)} \mid j = 1, \ldots, J, t = 1, 2, \ldots\}$. Thus, using that G is assumed to be strongly convex on $\mathcal{S}^o$, conditions 2–3 are satisfied.

Using the assumption on the base learning algorithm, we can apply Proposition A.1 or Proposition A.2, respectively. Theorem 3.3 bounds the progress in the objective function in each individual iteration (and at any finite starting point). Thus, if we can find a subsequence $t_1, t_2, \ldots$ of iterations in which (3.28) is satisfied, the algorithm converges to the optimal solution.

By Proposition A.1 and Proposition A.2, if $\hat{\alpha}_{j_t}^{(t)} \geq \frac{1}{\eta_u}\left|\frac{\partial\, \mathrm{F}(\hat{\boldsymbol{\alpha}}^{(t)}))}{\partial\hat{\alpha}_{j_t}}\right|$, then (3.28) is satisfied for $\beta = \frac{\delta\eta_l}{\eta_u}$ and (3.30) guarantees the progress. The maximal number of subsequent iterations where this condition is not satisfied is bounded by J, since then $\hat{\alpha}_{j_t}^{(t+1)} = 0$. Hence, at least every J-th iteration applies (3.30). This proves the linear convergence in the sense of [Luenberger, 1984, p. 128] [see also Luo and Tseng, 1992, p. 26]. ∎

A.3.3 Proof of Theorem 4.3 from page 76

Lemma A.4. *While running Algorithm 4.7 using a δ-optimal base learner, the barrier parameter β is decreased only if $\beta \geq \mathcal{O}(\delta)\|\nabla_{\hat{\boldsymbol{\alpha}}\geq 0}\,\mathrm{F}_\beta\,\|_\infty$.*

Proof One can exploit the δ-optimality of the base learner to upper-bound the gradient $\|\nabla_{\hat{\boldsymbol{\alpha}}\geq 0}\,\mathrm{F}_\beta\,\|_\infty$ at the current iterate. If the base learner returns a hypothesis $h = \mathcal{L}(S, \mathbf{d})$, then by Definition 3.5 there does not exist another hypothesis with an edge larger than by a factor of δ^{-1} and $\sqrt{2\eta_u}(\eta_l\delta)^{-1/2}$ (cf. (A.8)), respectively.

If the wrapper selects an hypothesis, there exists no hypothesis with smaller edge, except those which are already at the boundary (i.e. $\hat{\alpha}_j = 0$). Combining both facts yields that there exists not other dimension with gradient larger than $\mathcal{O}(\delta^{-1})\left|\sum_{n=1}^N d_n^{(t)} h_t(\boldsymbol{x}_n) - C\right|$ as computed in step 3h. The lemma directly follows by the condition step 3h. ∎

If $\beta > 0$, the barrier function F_β is strictly convex at any finite point. Hence, we can apply Proposition 4.4. By Proposition B.1, one knows that any accumulation point of a sequence $\{\hat{\boldsymbol{\alpha}}^{(t)}\}_t$ satisfying $\|\nabla_{\hat{\boldsymbol{\alpha}}\geq 0}\,\mathrm{F}_{\beta_t}(\hat{\boldsymbol{\alpha}}^{(t)})\|_\infty \to 0$ for $\beta_t \to 0$ is a global solution of

(4.17). By Lemma A.4 we have that β is decreased only if $\beta > \mathcal{O}(\delta)\|\nabla_{\hat{\boldsymbol{\alpha}} \geq 0} \mathrm{F}_\beta(\hat{\boldsymbol{\alpha}})\|_\infty$. If β is not decreased, the gradient will be reduced in a finite number of iterations such that $\mathcal{O}(\delta)\|\nabla_{\hat{\boldsymbol{\alpha}} \geq 0} \mathrm{F}_\beta(\hat{\boldsymbol{\alpha}})\|_\infty < \beta$. Thus $\beta \to 0$ and $\liminf_{t\to\infty} \|\nabla_{\hat{\boldsymbol{\alpha}} \geq 0} \mathrm{F}_\beta(\hat{\boldsymbol{\alpha}})\|_\infty = 0$.

To show that $\lim_{t\to\infty} \|\nabla_{\hat{\boldsymbol{\alpha}} \geq 0} \mathrm{F}_\beta(\hat{\boldsymbol{\alpha}})\|_\infty$ exists and is zero, consider reducing β by a factor c, i.e. $\texttt{next}(\beta) = c\beta$. We claim that the gradient $\nabla_{\hat{\boldsymbol{\alpha}} \geq 0} \mathrm{F}_\beta(\hat{\boldsymbol{\alpha}})$ at the new β is bounded in terms of the gradient of the old β, if β is small enough. Consider the gradient with respect to a variable that is not at the bound or has negative gradient. Then its sub-gradient is given as in (3.31):

$$\frac{\partial \mathrm{F}_\beta(\hat{\boldsymbol{\alpha}})}{\partial \hat{\alpha}_j} = C - \sum_{n=1}^{N} y_n \hat{h}_j(\boldsymbol{x}_n) d_n = C - \sum_{n=1}^{N} y_n \hat{h}_j(\boldsymbol{x}_n) \left(1 + \exp\{(y_n f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}_n) - 1)/\beta\}\right)^{-1}$$

Furthermore, we have the following by Taylor expansion:

$$\frac{1}{1+\exp(a/(\beta+\Delta))} = \frac{1}{1+\exp(a/\beta)} + \frac{a\exp(a/\beta)}{\left(1+\exp(a/\beta)\right)^2 \beta^2}\Delta + \mathcal{O}(\Delta^2). \quad \text{(A.13)}$$

Thus, for small enough β, when reducing β by a factor of c we have that $d_n^{(\beta)}$ is changing only little

$$d_n^{(c\beta)} \approx d_n^{(\beta)} + \frac{c_1 a_n}{\beta} \frac{\exp(a_n/\beta)}{\left(1+\exp(a_n/\beta)\right)^2} \approx d_n^{(\beta)} + \frac{c_1 a_n}{\beta} \cdot \begin{cases} \frac{1}{1+\exp(a_n/\beta)} & a_n > 0 \\ 1/2 & a_n = 0 \\ \exp(a_n/\beta) & a_n < 0 \end{cases}$$

where $c_1 = 1 - c$ and $a_n = y_n f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}_n) - 1$. Since $\exp(-|a_n|/\beta)$ decreases much faster than β^{-1}, we have for small enough $\beta \ll |a_n|$: $d_n^{(c\beta)} \approx d_n^{(\beta)}$. Thus, asymptotically, if the gradient is small β, it is also for $c\beta$. We can conclude that $\lim_{t\to\infty} \|\nabla_{\hat{\boldsymbol{\alpha}} \geq 0} \mathrm{F}_\beta(\hat{\boldsymbol{\alpha}})\|_\infty = 0$. ∎

For simplicity, we did not define $\nabla_{\hat{\boldsymbol{\alpha}} \geq 0} \mathrm{F}_\beta(\hat{\boldsymbol{\alpha}})$ exactly (cf. footnote 11 on page 36). It can in fact be done, by continuing F_β for negative $\hat{\boldsymbol{\alpha}}$'s and using the ℓ_1-norm instead of the linear term. This leads to a non-differentiable but continuous function. Proposition B.1 also holds for this case. Then one needs to use sub-gradients.

A.4 Proofs from Chapter 5

A.4.1 Proof of Proposition 5.1 from page 92

Shifting y_n locally into y_n' does not change the status of $(\boldsymbol{x}_n, y_n)$ as being a point outside the tube. First assume without loss of generality[2] that $\xi_n > 0$, i.e. $y_n < f(\boldsymbol{x}_n) - \varepsilon$. All we have to show that by changing ξ_n into $\xi_n' = \xi_n + (y_n - y_n')$ and keeping all the other variables and therefore also the estimate f unchanged, we obtain an optimal solution again.

By construction the new set of variables for the modified problem is still feasible and

2. The case of $\xi_n^* > 0$ works in the same way, just with opposite signs.

the same constraints are active as in the initial solution. Finally the gradients in both the objective function and the constraint with respect to each variable remain unchanged since ξ_n only appears in a linear fashion and all other variables did not change at all. Thus the new solution is optimal again, leading to the same estimate of f as before.

Obviously, if y_n moved locally for a point inside the tube, it will keep being in the tube. Thus, also after moving the target the corresponding slack variable will be zero and the constraint will still be inactive. Hence, the old solution is still optimal. ■

A.4.2 Proof of Proposition 5.2 from page 93

Ad (i): Imagine increasing ε starting from 0. The second term in $\frac{1}{N}\sum_{n=1}^{N}(\xi_N + \xi_N^*) + \nu\varepsilon$ will increase proportionally to ν, while the first term will decrease proportionally to the fraction of points outside of the tube. Hence, ε will grow as long as the latter function is larger than ν. At the optimum, it therefore must be $\leq \nu$.

Ad (ii): Next, imagine decreasing ε starting from some large value. Again, the change in the second term is proportional to ν, but this time, the change in the first term is proportional to the fraction of points not inside the tube (even points on the edge will contribute). Hence, ε will shrink as long as the fraction of such points is smaller than ν, eventually leading the stated claim.

Ad (iii): The strategy of proof is to show that asymptotically, the probability of a point lying *on* the edge of the tube vanishes. The continuity of the conditional distribution $P(y|x)$ implies that for all f, and all $t \in \mathbb{R}$,

$$\lim_{\gamma \to 0} P(|f(x) - y + t| < \gamma) = 0.$$

Since the hypothesis space is finite, the regression estimates f has well-behaved covering numbers (e.g. Williamson et al. [1998]), we get uniform convergence, so for all $\gamma > 0$,

$$\sup_{f \in \mathsf{F}} |P(|f(x) - y + t| < \gamma) - \hat{P}_\ell(|f(x) - y + t| < \gamma)|$$

converges to zero in probability, where $\hat{P}_\ell$ is the sample-based estimate of P (that is, the proportion of points with $|f(x) - y + t| < \gamma$). But then for all $\hat{\alpha} > 0$,

$$\lim_{\gamma \to 0} \lim_{\ell \to \infty} P(\sup_{f \in \mathsf{F}} \hat{P}_\ell(|f(x) - y + t| < \gamma) > \hat{\alpha}) = 0.$$

Hence, $\sup_f \hat{P}_\ell(|f(x) - y + t| = 0)$ converges to zero in probability. Thus, the fraction of points with a fixed distance t (in particular, $t = \pm\varepsilon$) to f almost surely converges to 0. Combining (i) and (ii) then shows that both fractions converge almost surely to ν.

■

A.4.3 Proof of Proposition 5.3 from page 95

The dual problem of (5.5) with (5.13) (in short (5.5)) is:

$$\begin{array}{lll} \max\limits_{\mathbf{d},\boldsymbol{\kappa}} & C\sum\limits_{j=1}^{J}\mathbf{P}_*^j(\kappa_j/C)+\sum\limits_{n=1}^{N}\mathrm{g}_*(d_n) & \\ \text{with} & H^\top\mathbf{d} &= \boldsymbol{\kappa} \\ & \mathbf{d}^\top\mathbf{1} &= 0 \\ & \boldsymbol{\kappa}\in\mathbb{R}^J, \mathbf{d}\in\mathbb{R}^N, & \end{array} \tag{A.14}$$

where $\mathbf{P}_*^j(\cdot)$ is the nonnegative and strictly convex conjugate function to $\mathbf{P}^j(\cdot)$ (cf. (5.19)). This is seen by a straightforward extension of Appendix A.4.6.

Lemma A.5. *It holds*

$$\kappa_j\neq 0 \quad\Rightarrow\quad \hat\alpha_j\neq 0, \quad \textit{for} \quad j=1,\ldots,J. \tag{A.15}$$

Proof Suppose (A.15) does not hold for some j^+. Assume $\kappa_{j^+}\neq 0$, then $\hat h_{j^+}(\boldsymbol{x}_n)\neq \mathrm{const}$. Let $\hat{\boldsymbol{\alpha}}^*$ be the solution to (5.5) with $\hat\alpha^*_{j^+}=0$, i.e. the j^+-th constraint in (A.14) is not active. Then there must exist a subset $\mathcal{J}=\{j^1,\ldots,j^k\}$ of the set of all indices corresponding to active constraints, such that the constraints indexed by $\mathcal{J}$ imply $H_{j^+}^\top\mathbf{d}=\kappa_{j^+}$ for any feasible $\mathbf{d}$. Let $H_{J+1}=\mathbf{1}$, denoting the last equality constraint in (A.14) which corresponds to the primal variable b in (5.5). Hence

$$H_{j^+}=\sum_{j\in\mathcal{J}}\beta_j H_j, \quad j\in\mathcal{J}, \tag{A.16}$$

for some $\beta_j\geq\mathbf{0}$, $j\in\mathcal{J}$ (by complementation closeness). Otherwise the dual cannot be optimal. Since $\hat h_{j^+}(\boldsymbol{x}_n)\neq\mathrm{const}$, it holds $\mathcal{J}\neq\{J+1\}$. Note, the constraints $H_j^\top\mathbf{d}=\kappa_j$, $j\in\mathcal{J}$ are active, i.e. $\hat\alpha_j\neq 0$ for $j\in\mathcal{J}$. Let

$$\mathcal{S}^+=\left\{\hat{\boldsymbol{\alpha}}^s \;\middle|\; \hat{\boldsymbol{\alpha}}^s\in\mathbb{R}^J, \hat\alpha_j^s=\begin{cases} s & \text{for } j=j^+ \\ \hat\alpha_j^*-s\beta_j & \text{for } j\in\mathcal{J} \quad \text{with } s\in\mathbb{R} \\ \hat\alpha_j=\hat\alpha_j^* & \text{otherwise} \end{cases}\right\}.$$

By (A.16), $H(\hat{\boldsymbol{\alpha}}^*-\hat{\boldsymbol{\alpha}}^+)=0$ for $\hat{\boldsymbol{\alpha}}^+\in\mathcal{S}^+$ holds. Thus the second term in the objective of (5.5) is not changed for any $\hat{\boldsymbol{\alpha}}^+\in\mathcal{S}^+$. Consider the subproblem

$$\min_{\hat{\boldsymbol{\alpha}}^+\in\mathcal{S}^+}\sum_{j=1}^{J}\mathbf{P}^j(\hat\alpha_j). \tag{A.17}$$

The optimal solution $\hat{\boldsymbol{\alpha}}^*$ of (5.5) must also be optimal for (A.17), because all elements from $\mathcal{S}$ are admissible. Since $\mathbf{P}^j$, $j=1,\ldots,J$ are differentiable and strictly convex the derivatives are continuous and strictly increasing. Since $\mathbf{P}^j(0)=0$ and $\mathbf{P}^j(z)>0$ for $z\neq 0$ holds: $\frac{\partial\mathbf{P}^j(z)}{\partial z}\Big|_{z=0}=0$. By continuity of $\frac{\partial\mathbf{P}^j(z)}{\partial z}$ and for small enough $s>0$, simultaneously reducing $\hat\alpha_j^*$ by $s\beta_j$, $j\in\mathcal{J}\setminus\{J+1\}$, reduces the regularization term more

than it increases by increasing $\hat{\alpha}^*_{j^+}$ by s. Thus $\hat{\alpha}_{j^+} = 0$ cannot be optimal in (A.17) and therefore also not optimal in (5.5). This proves Lemma A.5. ∎

We are now going to upper bound the number of zero entries in $\boldsymbol{\kappa}$ and $\hat{\boldsymbol{\alpha}}$. Assume $\mathbf{d} \neq 0$ and H is of full row-rank, otherwise $p_0 = 1$ and the statement of Proposition 5.3 is trivially fulfilled. If $\mathbf{d} = 0$, then $\boldsymbol{\kappa} = 0$ and the dual objective is zero since $\mathrm{g}_*(0) = \min_z \mathrm{g}(z) = \mathrm{g}(0) = 0$ (cf. (5.20)). Thus by Lemma 5.3 the primal objective is zero and therefore $\hat{\boldsymbol{\alpha}}^* = 0$, which we have excluded by assumption.

Let I be the largest subset of indices of columns of H, such that the resulting sub-matrix H_I has rank $N-1$. Let $I^c = \{1, \ldots, J\} \setminus I$. For any vector $\mathbf{d} \neq 0$, the vector $\boldsymbol{\kappa} = H^\top \mathbf{d}$ must have at least $|I^c|$ non-zero entries, since I^c is minimal. This can be seen as follows: For any $A \in \mathbb{R}^{J \times N}$, the system $A\mathbf{d} = \mathbf{0}$ can only be fulfilled if and only if A is not of full rank. By Lemma A.5, I^c is also the smallest number of non-zero entries in $\hat{\boldsymbol{\alpha}}$. Therefore, $|I|$ is the largest number of zero entries in $\hat{\boldsymbol{\alpha}}$.

The probability that a random sample of columns (with replacement) of size J does not have full rank is

$$\begin{aligned} p_0 = P(\mathrm{rg}(H_S) < N) &= P(\mathrm{rg}(H_S) < N, I^c \cap S \neq \emptyset) + P(\mathrm{rg}(H_S) < N, I^c \cap S = \emptyset) \\ &\geq P(\mathrm{rg}(H_S) < N, I^c \cap S = \emptyset) \\ &= P(I^c \cap S = \emptyset) = (1 - |I^c|/J)^J, \end{aligned}$$

where S is the randomly chosen subset of $\{1, \ldots, J\}$. Thus, $\frac{|I|}{J} \leq \sqrt[J]{p_0}$. This proves the proposition. ∎

A.4.4 Proof of Proposition 5.4 from page 95

If $J \leq N$, Proposition 5.4 is trivially true. Assume $J > N$. Suppose $\hat{\boldsymbol{\alpha}}^*$ is a local minimizer of (5.5) and has $M > N$ nonzero entries. We are going to construct an vector $\hat{\boldsymbol{\alpha}}^+$ that uses one hypothesis less and has a lower or the same objective value. This argumentation applies as long as $M > N$ and, hence, shows that there exists a solution that uses at most N hypotheses.

Let $\mathcal{J}$ be the set of indices such that $\hat{\alpha}_j > 0$ for all $j \in \mathcal{J}$. With $M > N$, there exists a subset $\mathcal{J}^+ \subseteq \mathcal{J}$ with at least two elements such that for $j^+ \in \mathcal{J}^+$ holds:

$$H_{j^+} = \sum_{j \in \mathcal{J}^+ \setminus \{j^+\}} \beta_j^{j^+} H_j \tag{A.18}$$

for some $\beta_j^{j^+} \in \mathbb{R}$. This is because there can only be N linearly independent components. If one reduces $\hat{\alpha}^*_{j^+}$ by say $s > 0$, one can find an update Δ such that $H(\hat{\boldsymbol{\alpha}}^* - \hat{\boldsymbol{\alpha}}^+) = \mathbf{0}$, where $\hat{\boldsymbol{\alpha}}^+ = \hat{\boldsymbol{\alpha}}^* + \Delta$. The update can be computed using (A.18): $\Delta_j = s\beta_j^{j^+}$ for $j \in \mathcal{J}^+ \setminus \{j^+\}$ and $\Delta_{j^+} = -s$. Let us consider the difference of the objective function in (5.5) for some small $s > 0$ and any fixed $j^+ \in \mathcal{J}^+$. The second part $\sum_{n=1}^{N} \mathrm{g}(\xi_n)$ does not change, since $H(\hat{\boldsymbol{\alpha}}^* - \hat{\boldsymbol{\alpha}}^+) = \mathbf{0}$. Using (5.13) and the concaveness of $\mathbf{P}^j$ for all $j = 1, \ldots, J$, the first term is a concave function $\delta(s)$ in s, which is properly defined on

an interval (including $s = 0$) such that $\hat{\alpha}_j \geq 0$ for all $j \in \mathcal{J}^+$. Since δ is concave, there must be a minimum on the edge of the interval. Hence, there exists an $\hat{\boldsymbol{\alpha}}^+$, such that the first term in (5.5) does not become larger. Also, if δ is strictly concave, then a minimizer has to be on the edge and $\hat{\boldsymbol{\alpha}}^*$ cannot be optimal. This argumentation applies for any j^* and shows that the new point $\hat{\boldsymbol{\alpha}}^+$ is sparser and not worse in the regularization term than $\hat{\boldsymbol{\alpha}}^*$. ■

Note that the above-constructed $\hat{\boldsymbol{\alpha}}^+$ might not be the optimal choice. There might be one particular j^+ that leads to the largest improvement – but here it is sufficient to show that there is a vector that is sparser than $\hat{\boldsymbol{\alpha}}^*$, it is feasible and has no larger objective function value. Finally note that this also holds if $C = 0$, since the constant zero function is concave.

A.4.5 Proof of Corollary 5.1 from page 101

For the point $\hat{\boldsymbol{z}} = \mathbf{0}$ holds $-d_{\max} < \hat{\boldsymbol{z}} < d_{\max}$ and $\langle \hat{h}_{\boldsymbol{p}}(X), \hat{\boldsymbol{z}} \rangle = 0 < C$ for $C > 0$ and $d_{\max} > 0$.This proves the statement, if $d_{\max} > 0$. Assume $d_{\max} = |\,\mathrm{g}'(\|Y\|_1)| = |\,\mathrm{g}'(-\|Y\|_1)| = 0$. Since $\mathrm{g}(0) = 0$ and g is convex and non-negative, we have $\mathrm{g}'(z) = \mathrm{g}(z) = 0$ for $z \in [-\|Y\|_1, \|Y\|_1]$. Thus an optimal solution to (5.14) is $\hat{\boldsymbol{\alpha}} = \mathbf{0}, b = 0$ and $\boldsymbol{\xi} = Y$. Thus, $\Omega(D_G) = \Omega(D_G(\emptyset)) = 0$. This proves the statement. ■

A.4.6 Derivation of the dual Regression Problem with ℓ_1-norm regularization (page 98)

A.4.6.1 Primal Problem

Let $\mathrm{g}(\cdot)$ be a continuously differential cost function.

$$\begin{array}{ll} \min\limits_{\hat{\boldsymbol{\alpha}}, b, \boldsymbol{\xi}} & C \sum_{j=1}^{J} \hat{\alpha}_j + \sum\limits_{n=1}^{N} \mathrm{g}(\xi_n) \\ \text{with} & y - b - \sum_{j=1}^{J} \hat{\alpha}_j \hat{h}_j(\boldsymbol{x}_n) \quad = \xi_n \quad \forall n = 1, \ldots, N \\ & \mathbf{0} \leq \hat{\boldsymbol{\alpha}} \in \mathbb{R}^J, b \in \mathbb{R}, \boldsymbol{\xi} \in \mathbb{R}^N, \end{array} \tag{A.19}$$

A.4.6.2 Lagrangian

Let $H_{nj} = \hat{h}_j(\boldsymbol{x}_n)$ and $Y = [y_1, \ldots, y_n]^\top$.

$$L = C \sum_{j=1}^{J} \hat{\alpha}_j + \sum_{n=1}^{N} \mathrm{g}(\xi_n) + \mathbf{d}^\top (Y - b\mathbf{1} - H\hat{\boldsymbol{\alpha}} - \boldsymbol{\xi}) - \boldsymbol{\gamma}^\top \hat{\alpha} \tag{A.20}$$

with $\mathbf{d} \in \mathbb{R}^N$ and $\mathbf{0} \leq \boldsymbol{\gamma} \in \mathbb{R}^J$. Setting the derivatives to zero

$$\frac{\partial L}{\partial \hat{\boldsymbol{\alpha}}} = C\mathbf{1} - H^\top \mathbf{d} - \boldsymbol{\gamma} = \mathbf{0} \tag{A.21}$$

$$\frac{\partial L}{\partial b} = -\mathbf{1}^\top \mathbf{d} = 0 \tag{A.22}$$

$$\frac{\partial L}{\partial \boldsymbol{\xi}} = \mathrm{g}'(\boldsymbol{\xi}) - \mathbf{d} = \mathbf{0} \tag{A.23}$$

where $\mathrm{g}' := \frac{d\,\mathrm{g}(x)}{dx}$, yields $H^\top \mathbf{d} \leq C\mathbf{1}$ and $\sum_n d_n = 0$ as linear constraints. Plugging in (A.21) and (A.22):

$$\begin{aligned} L &= C \sum_{j=1}^{J} \hat{\alpha}_j + \sum_{n=1}^{N} \mathrm{g}(\xi_n) - \hat{\boldsymbol{\alpha}}^\top H^\top \mathbf{d} + Y^\top \mathbf{d} - \mathbf{d}^\top \boldsymbol{\xi} - \boldsymbol{\gamma}^\top \hat{\alpha} \\ &= Y^\top \mathbf{d} + \sum_{n=1}^{N} \mathrm{g}(\xi_n) - \xi_n d_n \end{aligned}$$

From (A.23) we get $\xi_n = (\mathrm{g}')^{-1}(d_n)$, $n = 1, \ldots, N$. Hence

$$L = Y^\top \mathbf{d} + \sum_{n=1}^{N} \mathrm{g}((\mathrm{g}')^{-1}(d_n)) - d_n (\mathrm{g}')^{-1}(d_n),$$

where $\mathrm{g}_*(z) = z(\mathrm{g}')^{-1}(z) - \mathrm{g}((\mathrm{g}')^{-1}(z))$ is the conjugate function to $\mathrm{g}(\cdot)$ and is concave, if $\mathrm{g}(\cdot)$ is convex.

$$L = Y^\top \mathbf{d} - \sum_{n=1}^{N} \mathrm{g}_*(d_n)$$

A.4.6.3 Dual Problem

Thus, the dual of (A.19) is:

$$\begin{array}{lll} \max\limits_{\mathbf{d} \in \mathbb{R}^n} & Y^\top \mathbf{d} - \sum_{n=1}^{N} \mathrm{g}_*(d_n) & \\ \text{with} & \sum_{n=1}^{N} \hat{h}_j(\boldsymbol{x}_n) d_n \leq C & \forall j = 1, \ldots, J \\ & \sum_{n=1}^{N} d_n = 0. & \end{array} \tag{A.24}$$

A.4.7 Proof of Lemma 5.2 from page 99

The first order optimality conditions for (5.18) gives: $y_n - \mathrm{g}'_*(d_n) - \sum_{j=1}^{n} \hat{\alpha}_j \hat{h}_j(\boldsymbol{x}_n) - b = 0$, and thus for any optimal $\mathbf{d}^{\mathrm{opt}}, \hat{\boldsymbol{\alpha}}^{\mathrm{opt}}, b^{\mathrm{opt}}$ must hold $\sum_n |y_n - f_{\hat{\boldsymbol{\alpha}}^{\mathrm{opt}}, b^{\mathrm{opt}}}(\boldsymbol{x}_n)| = \sum_n |\mathrm{g}'_*(d_n^{\mathrm{opt}})|$. A feasible solution for (5.14) is $\hat{\boldsymbol{\alpha}} = \mathbf{0}$, $b = 0$ and $\boldsymbol{\xi} = Y$. By optimality of $\hat{\boldsymbol{\alpha}}^{\mathrm{opt}}$ and b^{opt}, holds $\sum_n |y_n - f_{\hat{\boldsymbol{\alpha}}^{\mathrm{opt}}, b^{\mathrm{opt}}}(\boldsymbol{x}_n)| \leq \|Y\|_1$ and therefore also $\|q\|_\infty \leq \|q\|_1 \leq \|Y\|_1$ with $q_n := \mathrm{g}'_*(d_n^{\mathrm{opt}})$. Since $|\mathrm{g}'(q_n)| \leq \max(\mathrm{g}'(\|Y\|_1), \mathrm{g}'(-\|Y\|_1))$ for $|q_n| \leq \|Y\|_1$ (by convexity of g) and $\mathrm{g}'(q_n) = d_n^{\mathrm{opt}}$, we obtain $|d_n^{\mathrm{opt}}| \leq \max(\mathrm{g}'(\|Y\|_1), \mathrm{g}'(-\|Y\|_1))$. ∎

A.4.8 Proof of Lemma 5.3 from page 99

Consider a linearly constraint nonlinear optimization problem

$$\begin{array}{ll} \min\limits_{\boldsymbol{x}} & f(\boldsymbol{x}) \\ \text{with} & \boldsymbol{x} \in \mathbb{X} \subseteq \mathbb{R}^N \\ & A\boldsymbol{x} \leq \boldsymbol{b} \\ & B\boldsymbol{x} = \boldsymbol{q} \end{array} \tag{A.25}$$

Proposition A.3 (e.g. Bertsekas [1995], page 437). *Assume that the problem (A.25) is feasible and its optimal value $f(\boldsymbol{x}^*)$ finite. Let also f be convex over $\mathbb{R}^N$ and let $\mathbb{X}$ be polyhedral. Then there exists at least one [set of] Lagrange multiplier[s] and there is no duality gap.*

We need to show that the assumptions in Lemma 5.3 imply the assumptions in Proposition A.3. The problem (5.14) is always feasible (e.g. $\hat{\boldsymbol{\alpha}} = \mathbf{0}, b = 0, \boldsymbol{\xi} = Y$). Furthermore, the objective is bounded from below by 0 and upper bounded by $\sum_{n=1}^{N} \mathrm{g}(y_n)$. In the case of (5.14) $\mathbb{X} = \mathbb{R}$ and is therefore polyhedral. Thus we can apply Proposition A.3 and have shown the strong duality of (5.14) and (5.18). ∎

A.4.9 Proof of Corollary 5.2 from page 103

In Glashoff and Gustafson [1978], Glashoff [1979] one finds the the following:

Lemma A.6. *Under the following assumptions*

- *$\mathcal{P}$ is a compact set; the functions $\boldsymbol{u}(\cdot)$ and $v(\cdot)$ are continuous on $\mathcal{P}$ and*
- *there exists a $\mathbf{d} \in \mathbb{R}^N$ such that $\langle \boldsymbol{u}(\boldsymbol{p}), \mathbf{d} \rangle < v(\boldsymbol{p})$ for all $\boldsymbol{p} \in \mathcal{P}$,*

the cone M_{N+1} is closed.

By the assumptions on the finiteness of y_n and by Corollary 5.1, we can apply Lemma 5.2 to show that $d_{\max} < \infty$ (differentiability is given by transforming the piecewise-linear function to a linear one, while adding additional constraints). Hence $\mathcal{P}$ is a compact set. The functions $\boldsymbol{u}(\cdot)$ and $v(\cdot)$ are obviously continuous (since they are linear). Furthermore, since $C > 0$, there exists a $\mathbf{d}$ (e.g. $\mathbf{d} = \mathbf{0}$) such that $\langle \boldsymbol{u}(\boldsymbol{p}), \mathbf{d} \rangle < C$. Hence, we can conclude by Lemma A.6, that M_{N+1} is closed. Thus, $\Omega(P_R) = \Omega(D_R)$ and primal minimum is attained. By, Theorem 5.5 (2) and by compactness of $\mathcal{P}$, also the dual minimum is attained. ∎

A.4.10 Proof of Theorem 5.7 from page 105

As $(\mathbf{d}^{(t),+}, \mathbf{d}^{(t),-})$ is bounded, a point of accumulation always exists. Without loss of generality assume $\mathbf{d}^{(t)} := (\mathbf{d}^{(t),+}, \mathbf{d}^{(t),-}) \to (\tilde{\mathbf{d}}^{+}, \tilde{\mathbf{d}}^{-}) =: \tilde{\mathbf{d}}$. Denote by $\mathsf{H}^{(t)}$ the set of hypothesis that have been returned by the base learner until the t-th iteration and by $\Phi(H^{(t)})$ the value of the objective at optimality. Let $\mathcal{M}(\mathbf{d}) := \max_{\boldsymbol{p} \in \mathcal{P}} \langle \hat{h}_{\boldsymbol{p}}, \mathbf{d}, \rangle$. Note, $\mathcal{M}$

is continuous. Since $\mathsf{H}^{(t)} \subset \mathsf{H}$, holds $\Phi(\mathsf{H}^{(t)}) \geq \Phi(\mathsf{H})$. Therefore, it suffices to show that $\tilde{\mathbf{d}}$ is feasible, i.e. $\mathcal{M}(\tilde{\mathbf{d}}) \leq 0$. Assume to the contrary, that $\tilde{\mathbf{d}}$ is not feasible, i.e. $\mathcal{M}(\tilde{\mathbf{d}}) > 0$. We have by τ_1-optimality

$$\begin{aligned}\mathcal{M}(\tilde{\mathbf{d}}) &= \mathcal{M}(\mathbf{d}^{(t)}) + [\mathcal{M}(\tilde{\mathbf{d}}) - \mathcal{M}(\mathbf{d}^{(t)})] \\ &\leq \tau_1^{-1}(\langle \mathbf{d}^{(t)}, h_t \rangle) + [\mathcal{M}(\tilde{\mathbf{d}}) - \mathcal{M}(\mathbf{d}^{(t)})] \\ &\leq \tau_1^{-1}(\langle \mathbf{d}^{(t)}, h_t \rangle - \langle \tilde{\mathbf{d}}, h_t \rangle) + [\mathcal{M}(\tilde{\mathbf{d}}) - \mathcal{M}(\mathbf{d}^{(t)})],\end{aligned}$$

where we have used $\langle \tilde{\mathbf{d}}, h_t \rangle \leq 0$ by construction and the monotonicity of τ_1^{-1}. Since τ_1 and $\mathcal{M}$ are continuous, the rhs. must go to zero as $\mathbf{d}^{(t)} \to \tilde{\mathbf{d}}$. This is a contradiction. Hence $\mathcal{M}(\tilde{\mathbf{d}}) \leq 0$. ∎

A.4.11 Proof of Theorem 5.8 from page 107

As $\mathbf{d}^{(t)}$ is bounded by Lemma 5.2, a point of accumulation always exists. Without loss of generality assume $\mathbf{d}^{(t)} \to \tilde{\mathbf{d}}$. Since $C > 0$ and the loss is symmetric, $(\hat{\boldsymbol{\alpha}}^{(t)}, b_t)$ stays bounded and there must exist an corresponding point of accumulation $(\tilde{\hat{\boldsymbol{\alpha}}}, \tilde{b})$ with $\tilde{d}_n = \mathrm{g}'(y_n - f_{\tilde{\hat{\boldsymbol{\alpha}}}, \tilde{b}}(\boldsymbol{x}_n))$ (cf. step 2b in Algorithm 5.9). By assumption the support of $\tilde{\hat{\boldsymbol{\alpha}}}$ is finite.

Since $(\hat{\boldsymbol{\alpha}}^{(t)}, b_t)$ is a φ_t-minimizer of E_t and g is strictly convex, there exists an *optimal* solution $(\tilde{\hat{\boldsymbol{\alpha}}}^{(t)}, \tilde{b}_t)$ to (5.34), such that

$$|f_{\tilde{\hat{\boldsymbol{\alpha}}}^{(t)}, \tilde{b}_t} - f_{\hat{\boldsymbol{\alpha}}^{(t)}, b_t}| \leq \mathcal{O}(\varphi_t). \tag{A.26}$$

Furthermore, $d_n^{(t)} = \mathrm{g}'(y_n - f_{\hat{\boldsymbol{\alpha}}^{(t)}, b_t}(\boldsymbol{x}_n))$ as in step 2b is the optimal dual solution in iteration t for $(\hat{\boldsymbol{\alpha}}^{(t)}, b_t)$ (cf. Appendix A.4.6). Let $\tilde{d}_n^{(t)} = \mathrm{g}'(y_n - f_{\tilde{\hat{\boldsymbol{\alpha}}}^{(t)}, \tilde{b}_t})$. Since g' is continuous by assumption and (A.26), holds $|d_n^{(t)} - \tilde{d}_n^{(t)}| \leq \mathcal{O}(\varphi_t)$. Thus, it holds $|\langle \mathbf{d}^{(t)}, h_t \rangle) - \langle \tilde{\mathbf{d}}^{(t)}, h_t \rangle)| \leq y_{\max} N \mathcal{O}(\varphi_t) = \mathcal{O}(\varphi_t)$. Roughly speaking, if φ is small enough, it does not change the dual variables $\mathbf{d}$ drastically and the hypothesis returned by the base learner in iteration t for non-optimal $\mathbf{d}^{(t)}$ can still be useful.

Denote by $\mathsf{H}^{(t)}$ the set of hypothesis that have been returned by the base learner until the t-th iteration and by $\Phi(H^{(t)})$ the value of the objective at optimality. Let $\mathcal{M}(\mathbf{d}) := \max_{\boldsymbol{p} \in \mathcal{P}} \langle \hat{h}_{\boldsymbol{p}}, \mathbf{d}, \rangle$. Since $\mathsf{H}^{(t)} \subset \mathsf{H}$ and g_* is continuous, holds $\Phi(\mathsf{H}^{(t)}) + \mathcal{O}(\varphi_t) \geq \Phi(\mathsf{H})$. Since $\varphi_t \to 0$, it suffices to show that $\tilde{\mathbf{d}}$ is feasible, i.e. $\mathcal{M}(\tilde{\mathbf{d}}) \leq 0$. Assume to the contrary, that $\tilde{\mathbf{d}}$ is not feasible, i.e. $\mathcal{M}(\tilde{\mathbf{d}}) > 0$. We have by τ_1-optimality

$$\begin{aligned}\mathcal{M}(\tilde{\mathbf{d}}) &= \mathcal{M}(\mathbf{d}^{(t)}) + [\mathcal{M}(\tilde{\mathbf{d}}) - \mathcal{M}(\mathbf{d}^{(t)})] \\ &\leq \tau_1^{-1}(\langle \mathbf{d}^{(t)}, h_t \rangle) + [\mathcal{M}(\tilde{\mathbf{d}}) - \mathcal{M}(\mathbf{d}^{(t)})] \\ &\leq \tau_1^{-1}(\langle \tilde{\mathbf{d}}^{(t)}, h_t \rangle + \mathcal{O}(\varphi_t)) + [\mathcal{M}(\tilde{\mathbf{d}}) - \mathcal{M}(\mathbf{d}^{(t)})] \\ &\leq \tau_1^{-1}(\langle \tilde{\mathbf{d}}^{(t)}, h_t \rangle - \langle \tilde{\mathbf{d}}, h_t \rangle + \mathcal{O}(\varphi_t)) + [\mathcal{M}(\tilde{\mathbf{d}}) - \mathcal{M}(\mathbf{d}^{(t)})],\end{aligned}$$

where we have used $\langle \tilde{\mathbf{d}}, h_t \rangle \leq 0$ since $\varphi_t \to 0$ and the monotonicity of τ_1^{-1}. Since τ_1 and $\mathcal{M}$ are continuous and $\tilde{\mathbf{d}}^{(t)} \to \tilde{\mathbf{d}}$ since $\varphi_t \to 0$, the rhs. must go to zero as $\mathbf{d}^{(t)} \to \tilde{\mathbf{d}}$. This is a contradiction. Hence $\mathcal{M}(\tilde{\mathbf{d}}) \leq 0$. ∎

Appendix B Technical Addenda

B.1 Notation

The following table summarizes the notation that will be used throughout this work:

Symbol	Description	First used in
n, N	counter and number of examples	Section 1.2.1
j, J	counter and number of hypotheses if finite	Section 1.3.1
t, T	counter and number of iterations	Section 1.3.1
$\mathbb{X}, s$	input space, dimensionality of $\mathbb{X}$ (if $\mathbb{X} \subseteq \mathbb{R}^s$)	Section 1.2.1
$\mathbb{Y}$	label space with $\mathbb{Y} \subseteq \mathbb{R}$	Section 1.2.1
$\boldsymbol{x}, y, \boldsymbol{z}$	a training example, the label and the pair	Section 1.2.1
X, Y, S	training data: input, targets, both	Section 1.2.1
X, Y	Random variables on $\mathbb{X}$ and $\mathbb{Y}$	Section 1.2.1
$\mathbf{E}\{\mathsf{X}\}$	expectation of a random variable X	Section 1.2.1
$P(A)$	probability of an event A	Section 1.2.1
$p(\boldsymbol{x}, y)$	probability density on $\mathbb{X}$, $\mathbb{Y}$	Section 1.2.1
$R[f]$	Expected risk of a function f	Section 1.2.1
$R_{\text{emp}}[f, S]$	Empirical risk of a function f on a sample S	Section 1.2.1
F	set of functions – usually set of linear combinations of H	Section 1.2.1
ϑ	Vapnik-Chervonenkis dimension of a function set (usually F)	Section 1.2.2
ϵ, δ	Accuracy and Confidence parameter	Section 1.2.2
$\mathsf{H}, \hat{h}_j$	set of base hypotheses, and an element	Section 1.3.1
$U_{n,j}$	hypotheses margin matrix $U_{n,j} = y_n \hat{h}_j(\boldsymbol{x}_n)$	Section 1.3.3
$H_{n,j}$	hypotheses output matrix $H_{n,j} = \hat{h}_j(\boldsymbol{x}_n)$	Section 3.1.3
$\mathbb{F}$	a feature space	Section 1.3.2
$\hat{\boldsymbol{\alpha}}$	hypothesis weight vector	Section 1.3.1
$f_{\hat{\boldsymbol{\alpha}}}$	an element of F using the weighting $\hat{\boldsymbol{\alpha}}$	Section 1.3.1
$\mathbf{d}$	weighting on the training set	Section 1.3.1

Symbol	Description	First used in
$\boldsymbol{w}$	a weight vector for linear models	Section 1.3.2
$\mathbf{I}(\cdot)$	the indicator function: $\mathbf{I}(true) = 1$ and $\mathbf{I}(false) = 0$	Section 1.3.1
ρ_n, ϱ	the margin size of an example and of a hyperplane	Section 1.3.1
ε	the tube size	Section 5.1.3.2
ν	the quantile parameter	Section 4.3
C	the regularization (complexity) parameter	Section 1.3.2
ϵ	weighted classification error	Section 1.3.1
$\|\cdot\|_p$	the ℓ_p-norm, $p = [1, \infty]$	Section 1.3.3
$\langle\cdot,\cdot\rangle$	scalar product	Section 1.3.2
$k(\cdot,\cdot)$	scalar product in feature space	Section 1.3.2
$\mathcal{S}$	a convex set (of feasible solutions)	Section 2.5
$\operatorname{dom} \mathrm{g}$	effective domain of a convex function g [e.g. Rockafellar, 1970, page 23]	Section 3.3.2
$\operatorname{ri} \mathcal{S}$	relative interior of a set $\mathcal{S}$ [e.g. Rockafellar, 1970, Section 6]	Section 5.3
$\operatorname{range} \mathrm{g}$	range of a function g	Section 3.4
β	the barrier parameter	Section 2.3.2
$\mathcal{L}(\ldots)$	base learner	Section 1.2.2
$\mathbb{R}, \mathbb{R}_+$	space of real numbers and non-negative real numbers	Section 1.2.1
$\mathcal{P}^N$	N dimensional probability simplex	Section 2.1
$\operatorname{rg}(A)$	rank of a matrix A	Section 5.2.1
$\mathcal{O}$	asymptotic order	Section 1.3.1
$\Delta_2(x, y)$	binary relative entropy	Section 2.2
$\nabla G(\boldsymbol{x})$	gradient of G evaluated at $\boldsymbol{x}$	Section 3.1
$\nabla^2 G(\boldsymbol{x})$	Hessian of G evaluated at $\boldsymbol{x}$	Section 3.3.2

B.2 Barrier Optimization

We briefly review some basic statements and formulations about barrier optimization techniques. For details see e.g. Frisch [1955], Bertsekas [1995], Luenberger [1984]. Consider the convex optimization problem

$$\begin{array}{ll} \min & g(\boldsymbol{\theta}) \\ \text{with} & c_n(\boldsymbol{\theta}) \geq 0, \quad \forall n = 1 \ldots N \end{array} \tag{B.1}$$

We call $\mathcal{S}$ the convex set of feasible solutions described by

$$\mathcal{S} = \{\boldsymbol{\theta} \; : \; c_n(\boldsymbol{\theta}) \geq 0, \quad \forall \; n = 1, \ldots, N\}. \tag{B.2}$$

We assume $\mathcal{S}$ is non-empty, i.e. there exists a feasible solution. If the function $g(\boldsymbol{\theta})$ is strictly convex, then the solution of problem (B.1) is unique. If $g(\boldsymbol{\theta})$ is convex only, there exists a set $\mathcal{S}^*$ of global solutions $\boldsymbol{\theta}^*$. Note that from the convexity of $g(\boldsymbol{\theta})$ and $\mathcal{S}$ follows that any local minimum is a global minimum as well.

Problem (B.1) can be solved by finding a sequence of (unconstraint) minimizers of the so called barrier (or penalty) error function

$$\mathrm{F}_\beta(\boldsymbol{\theta}) = g(\boldsymbol{\theta}) + \sum_{n=1}^{N} \kappa_\beta(c_n(\boldsymbol{\theta})), \tag{B.3}$$

where κ_β is a barrier function and $\beta > 0$ is a penalty parameter. Common choices for κ_β can be found in Table B.2. Barrier algorithms use a suitably chosen sequence of β's that goes to 0. For each β, (B.3) is minimized starting from the $\boldsymbol{\theta}$ found in the previous iteration.

For the log-barrier and the entropic barrier the initial $\boldsymbol{\theta}$ has to be a feasible solutions. The barrier function assures that the sequence of $\boldsymbol{\theta}$'s corresponding to the decreasing sequence of β values remain feasible. Thus the final $\boldsymbol{\theta}^*$ is approached from the "interior" of the feasible region. Methods based on the log-barrier and the entropic barrier are therefore called interior point methods. Such methods are often used for finding solutions for SVMs.

Table B.2 Common barrier functions used in convex optimization

$\kappa_\beta(t)$	Name	References
$-\beta \log t$	Log-Barrier	Bertsekas [1995], Frisch [1955], Luenberger [1984]
$-\beta t \log t$	Entropic Barrier	Bertsekas [1995], Luenberger [1984]
$\beta \exp(-t/\beta)$	Exp-Barrier	Cominetti and Dussault [1994], Kort and Bertsekas [1973]

In the case of the exp-barrier, the initial solution does not have to be feasible [Cominetti and Dussault, 1994]. For the exp-barrier the minimizers of $\mathrm{F}_\beta(\boldsymbol{\theta})$ will become feasible automatically, when β becomes small enough. If a constraint is violated then this will lead to an exponential growth in the barrier objective (B.3). So the barrier has the effect that it keeps $\boldsymbol{\theta}$ in the feasible region or pulls it closer to a feasible solution.

Besides an intuitive reasoning for the barrier function converging to an optimal solution of (B.1) the following presents some more formal aspects. Let us define

$$\boldsymbol{\theta}_\beta := \operatorname*{argmin}_{\boldsymbol{\theta}} \mathrm{F}_\beta(\boldsymbol{\theta}) \tag{B.4}$$

as the optimal solution for some fixed β. Moreover, let $\boldsymbol{\theta}^*$ be the set of global solutions of (B.1). Then for any barrier function and any sequence $\beta_t \to 0$ holds:

$$\lim_{t\to\infty} \boldsymbol{\theta}_{\beta_t} \in \boldsymbol{\Theta}^* \tag{B.5}$$

However, for the $\exp$-barrier it it turns out to be unnecessary to *exactly* minimize F_{β_t} for each β_t. For other barrier functions there exist similar results. The following proposition shows how close an estimate $\boldsymbol{\theta}^{(t)}$ to $\boldsymbol{\theta}_{\beta_t}$ has to be, in order to achieve convergence:

Proposition B.1 (along the lines of Cominetti and Dussault [1994]). *Assume g and c_n are continuous convex functions. Let $\boldsymbol{\theta}^{(t)}$ be an $\delta_t \geq 0$ minimizer of*

$$\mathrm{F}_{\beta_t}(\boldsymbol{\theta}) = g(\boldsymbol{\theta}) + \beta_t \sum_{n=1}^{N} \exp(-c_n(\boldsymbol{\theta})/\beta_t), \tag{B.6}$$

i.e. $\|\nabla_{\boldsymbol{\theta}} \mathrm{F}_{\beta_t}(\boldsymbol{\theta}^{(t)})\| \leq \delta_t$ (for non-differentiable functions one has to use sub-gradients). Then, for $\delta_t, \beta_t \overset{t\to\infty}{\longrightarrow} 0$ every limit point of $\{\boldsymbol{\theta}^{(t)}\}_{t\in\mathbb{N}}$ is a global solution to (B.1).

In the sequel we will often use a simpler version of Proposition B.1, where we use $\delta_t = \beta_t$ and require $\|\nabla_{\boldsymbol{\theta}} \mathrm{F}_{\beta_t}(\boldsymbol{\theta}^{(t)})\| \leq \beta_t$.

B.3 Another proof for Arc-GV

Breiman [1999] proposed the idea to adaptively set ϱ_t as the achieved margin of the previous iteration, i.e.

$$\varrho_t = \min_{n=1,\ldots,N} \rho_n(\boldsymbol{\alpha}^{t-1}). \tag{B.7}$$

This lead to an algorithm called Arc-GV (**Arc**ing-**G**ame **V**alue). Assuming that the base learning algorithm always returns a hypothesis with the *largest* edge among all hypotheses (called *weighted minimization*), it has been shown that Arc-GV asymptotically generates a hypothesis maximizing the margin, i.e.:

Proposition B.2 (Breiman [1999], Theorem 9.2). *Suppose the maximum achievable margin of a linear combination of hypotheses from H is ρ^* (cf. Section 1.3.3). Assume the base learning algorithm returns in each iteration t a hypothesis with* largest edge*. Then Algorithm 2.2 with ϱ adaptively chosen according to (B.7) converges to a combined hypothesis with margin ρ^*.*

The proof of Proposition B.2 is rather complicated. We therefore provide a much simpler

proof for a slightly different choice of ϱ_t:

$$\varrho_t = \max\left(\varrho_{t-1}, \min_{n=1,\dots,N} \rho_n(\boldsymbol{\alpha}^{t-1})\right) \tag{B.8}$$

starting with $\varrho_1 = 0$. Here $\{\varrho_t\}$ is monotonically increasing. We have:

Proposition B.3. *Suppose the maximum achievable margin of a linear combination of hypotheses from* H *is* $\rho^* \geq 0$ *(cf. Section 1.3.3). Assume the base learning algorithm returns in each iteration* t *a hypothesis with edge* $\hat{\gamma}_t \geq \rho^*$*. Then Algorithm 2.2 with* ϱ *adaptively chosen according to (B.8) converges to a combined hypothesis with margin* ρ^**.*

Proof By Theorem 2.1 and (B.8) we have $\varrho_t \leq \rho^*$. By assumption holds $\hat{\gamma}_t \geq \rho^*$ and therefore $\hat{\gamma}_t \geq \varrho_t$. Thus $\hat{\alpha}_t \geq 0$, where the equality only holds if $\hat{\gamma}_t = \varrho_t$, i.e. when $\varrho_t = \rho^*$ (again by Theorem 2.1). Then the goal is achieved and the algorithm may stop.

Suppose the sequence $\{\varrho_t\}$ would converge to some value ϱ^* smaller than ρ^*. Let $\epsilon > 0$ and t^ϵ such that $\varrho_t \geq \varrho^* - \epsilon$ for all $t \geq t^\epsilon$. We can apply Lemma 2.3 and conclude that $\sum_{r=1}^{t} \hat{\alpha}_r \overset{t\to\infty}{\longrightarrow} \infty$. Therefore the first $t^\epsilon - 1$ steps are not affecting the asymptotical hypothesis. Hence we can adopt the proof Corollary 2.2 for $\varrho = \varrho^* - \epsilon$ and conclude that the combined hypothesis has asymptotically a margin of at least $\frac{\varrho^* - \epsilon + \rho^*}{2}$. Since $\rho^* > \varrho^*$ and $\epsilon > 0$ chosen arbitrarily, it leads to the contradiction $\varrho^* > \varrho^*$ for every $\varrho^* < \rho^*$. Thus $\{\varrho_t\}$ must converge to ρ^*. ∎

Whereas Proposition B.2 seems to hold for finite hypothesis sets only (or with countable many elements [Breiman, private communication]), Proposition B.3 holds for any hypothesis sets H. Furthermore, one can easily show that the algorithm using (B.8) has the PAC boosting property (cf. Section 1.3.1).

For both algorithms above, it is difficult to show how fast the margin is maximized. Therefore, we have proposed Marginal Boosting in Section 2.4 that achieves a margin of at least $\rho^* - \epsilon$, $\epsilon > 0$, in a finite number of iterations.

B.4 A Note on Infinite Hypothesis Spaces for Marginal Boosting

In Chapter 2 we have implicitly assumed that the hypothesis space is finite. Here, we will show that this assumption is (often) not necessary [cf. Rätsch and Warmuth, 2001]. If the output of the hypotheses is discrete, the hypothesis space is effectively finite Demiriz et al. [2001b], Rätsch et al. [2002]. For *infinite hypothesis spaces*, Theorem 2.1 can be restated in a weaker form as:

Theorem B.1 (Weak Min-Max).

$$\gamma^* := \min_{\mathbf{d}} \sup_{\hat{h}\in H} \sum_{n=1}^{N} y_n \hat{h}(\boldsymbol{x}_n) d_n \geq \sup_{\hat{\boldsymbol{\alpha}}} \min_{n=1,\dots,N} y_n \sum_{t:\hat{\alpha}_t \geq 0} \hat{\alpha}_t \hat{h}_t(\boldsymbol{x}_n) =: \varrho^*, \tag{B.9}$$

where $\mathbf{d} \in P^N$, $\hat{\boldsymbol{\alpha}} \in P^{|H|}$ *with finite support. We call* $\Gamma = \gamma^* - \varrho^*$ *the "duality gap".*

In particular for any $\mathbf{d} \in P^N$: $\sup_{\hat{h}\in H} \sum_{n=1}^{N} y_n \hat{h}(\boldsymbol{x}_n) d_n \geq \gamma^*$ and for any $\hat{\boldsymbol{\alpha}} \in P^{|H|}$

with finite support: $\min_{n=1,\ldots,N} y_n \sum_{t:\hat{\alpha}_t \geq 0} \hat{\alpha}_t \hat{h}_t(\boldsymbol{x}_n) \leq \varrho^*$.

In theory the duality gap exists. However, Theorem 2.2 and Theorem 2.3 do not assume finite hypothesis spaces and show that the margin will converge arbitrarily close to ϱ^*, as long as the base learning algorithm can return *a single* hypothesis that has an edge not smaller than ϱ^*.

In other words, the duality gap may result from the fact that the $\sup$ on the left side can not be replaced by a $\max$, i.e. there might not exists a *single* hypothesis h with edge larger or equal to ϱ^*. By assuming that the base learner is always able to pick good enough hypotheses ($\geq \varrho^*$) one automatically gets that $\Gamma = 0$ (by Theorem 2.3).

Under certain conditions on H this maximum always exists and strong duality holds (see Section 5.3 and e.g. Rätsch et al. [2002], Hettich and Kortanek [1993] for details):

Theorem B.2 (Strong Min-Max). *If* $\{[\hat{h}(\boldsymbol{x}_1), \ldots, \hat{h}(\boldsymbol{x}_N)]^\top \mid \hat{h} \in H\}$ *is compact, then* $\Gamma = 0$.

In general this requirement can be fulfilled by base learning algorithms whose outputs continuously depend on the distribution $\mathbf{d}$. Furthermore, the outputs of the hypotheses need to be bounded (cf. step 3a in Algorithm 2.2). The first requirement might be a problem with base learning algorithms such as some variants of decision stumps or decision trees. However, there is a simple trick to avoid this problem: Roughly speaking, at each point with discontinuity $\hat{\mathbf{d}}$, one adds all hypotheses to H that are limit points of $\mathcal{L}(S, \mathbf{d}^s)$, where $\{\mathbf{d}^s\}_{s=1}^{\infty}$ is an arbitrary sequence converging to $\hat{\mathbf{d}}$ and $\mathcal{L}(S, \mathbf{d})$ denotes the hypothesis returned by the base learning algorithm for weighting $\mathbf{d}$ and training sample S (cf. Chapter 5 for more details).

B.5 Bounds with Compression Schemes

The idea of compression schemes has been introduced in Littlestone and Warmuth [1986]. It is quite different from the ones presented in Sections 1.2.2–1.2.3. Here one is not required to know the VC dimension of the function class F, but assumes that the algorithm uses only a few examples to determine the function $f \in \mathsf{F}$ and given only these examples it returns the same function as on the complete training set. It therefore *compresses* the sample to a smaller subset. See Floyd and Warmuth [1995] for connections between VC dimension and the minimal size of the compression set.

To define compression schemes more formally, we mainly follow Herbrich [2001] [see also Littlestone and Warmuth, 1986, Floyd and Warmuth, 1995]. Given an input space $\mathbb{X}$, a finite output space $\mathbb{Y}$ and a function space F with functions $f : \mathbb{X} \to \mathbb{Y}$. Let $\mathbb{Z} = \mathbb{X} \times \mathbb{Y}$ and $\mathbb{Z}^N$ be the set of all training sets of size N sampled from $P_{\mathsf{X},\mathsf{Y}}$. Furthermore, let $I_q \subset \mathbb{N}^q$ be the set of all ordered index vectors of size exactly $q \in \mathbb{N}$, i.e. $I_q = \{(i_1, \ldots, i_q) \in \mathbb{N}^q \mid 1 \leq i_1 < \ldots < i_q\}$. A compression scheme $(\mathcal{C}_q, \mathcal{R}_q)$ of size q consists of a compression function $\mathcal{C}_q : \cup_{N=q}^{\infty} \mathbb{Z}^N \to I_q$ and a reconstruction function $\mathcal{R}_q : \cup_{N=q}^{\infty} \mathbb{Z}^N \times I_q \to \mathsf{F}$. The reconstruction function is required to use only the indexed training samples, i.e.

$$\forall N \geq q : \forall S \in \mathbb{Z}^N : \forall \mathbf{i} \in I_q : \mathcal{R}_q(Z, \mathbf{i}) = \mathcal{R}_q((z_{i_1}, \ldots, z_{i_q}), (1, \ldots, q)).$$

The algorithm $\mathcal{L} : \cup_{N=q}^{\infty} Z^N \to \mathsf{F}$ is said to be a compression scheme of size q if and only if for any training sample $S \in \mathbb{Z}^N$ there exists a compression function $\mathcal{C}_q$ and a reconstruction function $\mathcal{R}_q$ whose composition yields the same hypothesis as $\mathcal{L}(S)$, i.e.

$$\forall S \in \mathbb{Z}^N : \exists \mathcal{C}_q, \mathcal{R}_q : \mathcal{L}(S) = \mathcal{R}_q(S, \mathcal{C}_q(S))$$

Assume the learning algorithm produces a function $f \in \mathsf{F}$ that is consistent with the training set, then we have the following [Littlestone and Warmuth, 1986, Floyd and Warmuth, 1995, Herbrich, 2001]:

Theorem B.3 (PAC Compression Bound). *Suppose we are given a fixed learning algorithm $\mathcal{L} : \cup_{N=k}^{\infty} \mathbb{Z}^N \to \mathsf{F}$, which is a compression scheme. For any probability measure $P_{\mathsf{X},\mathsf{Y}}$ and any $\delta \in [0,1]$, with probability at least $1-\delta$ over the random draw of the training sample $S \in \mathbb{Z}^N$: If $R_{\mathrm{emp}}[\mathcal{L}(S), S] = 0$ and $\mathcal{L}(S)$ corresponds to a compression scheme of size q, the expected risk $R[\mathcal{L}(S)]$ is bounded from above by*

$$R[\mathcal{L}(S)] \leq \frac{1}{N-q}\left(\log_2\left(\binom{N}{q}\right) + \log_2(N) + \log_2\left(\frac{1}{\delta}\right)\right) \tag{B.10}$$

This bound is very similar to the one in Theorem 1.3. However, there is one important difference. In Theorem B.3 the number of q of examples is not known a-priori but depends on the training sample S, whereas the VC dimension in Theorem 1.2 is fixed beforehand.

Note that this bound is already non-trivial, if the sample S of size N is compressed to a subset of about $N/2$. For instance, if a sample of size 200 is compressed to a subset of 10 examples and the hypothesis is consistent, the expected risk is smaller than 25% with probability 95%.

The compression idea can also be applied for regression using Chernoff bounding. Fix $q \in \{1, \ldots, N\}$, a hypothesis space $\mathsf{F} = \{f \mid f : \mathbb{X} \to \mathbb{Y}$ and a learning algorithm

$$\mathcal{L} : \cup_{N=k}^{\infty} \mathbb{Z}^N \to \mathsf{F} \tag{B.11}$$

For any measure $P_{\mathsf{X},\mathsf{Y}}$, such that $P(|\mathsf{Y}| \leq \Delta_1) = 1$, the probability that N examples $S \in \mathbb{Z}^N$ drawn i.i.d. according to $P_{\mathsf{X},\mathsf{Y}}$ contain a subset $S_q \in \mathbb{Z}^q$ of exactly q examples, and the regression function $\mathcal{L}(S)$ satisfies $P(|f(\mathbb{X})| \leq \Delta_2) = 1$, and has an expected risk $R[\mathcal{L}(S)]$ larger than

$$\frac{N}{N-q} R_{\mathrm{emp}}[f] + \epsilon$$

is less than

$$\binom{N}{d} \exp\left(-\frac{2\epsilon^2(N-d)^2}{\mathrm{g}_{\max}^2}\right), \tag{B.12}$$

where $\mathrm{g}_{\max} = \max_{|y_1| \leq \Delta_1} \max_{|y_2| \leq \Delta_2} \mathrm{g}(y_1, y_2)$. The proof mainly follows Littlestone and Warmuth [1986] and is omitted here.

The above result can be used to obtain a similar bound as in Theorem B.3 using a strategy found in Herbrich [2001]. Fix $q \in \{1, \ldots, N\}$ and a learning algorithm $\mathcal{L}$ as in (B.11),

which is a compression scheme of size q. Let $P_{\mathsf{Z}} = P_{\mathsf{X}\times\mathsf{Y}}$ be any probability measure, such that $P(|\mathsf{Y}| \leq \Delta_1) = 1$. Given a N-sample S drawn i.i.d. according to P_{Z}. Assume L returns a regression function $f = \mathcal{L}(S)$ with $P(|f(\mathbb{X})| \leq \Delta_2) = 1$. Then holds with probability $1 - \delta$:

$$R[f] \leq \frac{N}{N-q} R_{\mathrm{emp}}[f] + \frac{\mathrm{g}_{\max}}{\sqrt{N-q}} \sqrt{2 \log \left(\binom{N}{q} \right) + 2 \log \left(\frac{1}{\delta} \right)} \tag{B.13}$$

For space limitations we have to omit the proof here, however, it is straightforward to adapt the ideas on proving compressions bounds for classification as presented in Herbrich [2001]. One may use the stratification as in Herbrich et al. [2000] to leave q unspecified a priori, which leads to a practical bound for regression.

When computing (B.13) for different sample sizes and compression sizes, one finds that one obtains non-trivial results for a fraction of used examples less than about 15%.

B.6 Examples for Base Learning Algorithms

In this section we briefly look at some base learners that could be used together with the classification and regression algorithms proposed in this work. Which of them should be used for solving a certain problem depends of course on the problem at hand. It is important to note that our approach is general enough to use specialized hypothesis classes designed for particular problems, such that either the performance or the interpretability is as good as desired.

B.6.1 RBF nets with adaptive centers

The RBF nets used in the experiments are an extension of the method of [Moody and Darken, 1989], since centers and variances are also adapted [see also Bishop, 1995, Müller et al., 1999]. The output of the network is computed as a linear superposition of K basis functions

$$f(\boldsymbol{x}) = \sum_{k=1}^{K} w_k g_k(\boldsymbol{x}), \tag{B.14}$$

where w_k, $k = 1, \ldots, K$, denotes the weights of the output layer. The Gaussian basis functions g_k are defined as

$$g_k(\boldsymbol{x}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{\mu}_k\|^2}{2\,\sigma_k^2} \right), \tag{B.15}$$

where $\boldsymbol{\mu}_k$ and σ_k^2 denote means and variances, respectively. In a first step, the means $\boldsymbol{\mu}_k$ are initialized with K-means clustering and the variances σ_k are determined as the distance between $\boldsymbol{\mu}_k$ and the closest $\boldsymbol{\mu}_i$ ($i \neq k, i \in \{1, \ldots, K\}$). Then in the following steps we

perform a gradient descent in the regularized error function (weight decay)

$$E = \frac{1}{2}\sum_{n=1}^{N}\left(y_n - f(\boldsymbol{x}_n)\right)^2 + \frac{\lambda}{2l}\sum_{k=1}^{K} w_k^2. \tag{B.16}$$

Taking the derivative of (B.16) with respect to RBF means $\mu_{\mathbf{k}}$ and variances σ_k we obtain

$$\frac{\partial E}{\partial \boldsymbol{\mu}_k} = \sum_{n=1}^{N}(f(\boldsymbol{x}_n) - y_n)\frac{\partial}{\partial \boldsymbol{\mu}_k} f(\boldsymbol{x}_n), \tag{B.17}$$

with $\frac{\partial}{\partial \boldsymbol{\mu}_k} f(\boldsymbol{x}_n) = w_k \frac{\boldsymbol{x}_n - \mu_k}{\sigma_k^2} g_k(\boldsymbol{x}_n)$ and

$$\frac{\partial E}{\partial \sigma_k} = \sum_{n=1}^{N}\left(f(\boldsymbol{x}_n) - y_n\right)\frac{\partial}{\partial \sigma_k} f(\boldsymbol{x}_n), \tag{B.18}$$

with $\frac{\partial}{\partial \sigma_k} f(\boldsymbol{x}_n) = w_k \frac{\|\mu_k - \boldsymbol{x}_n\|^2}{\sigma_k^3} g_k(\boldsymbol{x}_n)$. These two derivatives are employed in the minimization of (B.16) by a conjugate gradient descent with line search, where we always compute the optimal output weights in every evaluation of the error function during the line search. The optimal output weights $\boldsymbol{w} = [w_1, \ldots, w_K]^\top$ in matrix notation can be computed in closed form by

$$\boldsymbol{w} = \left(G^\top G + 2\frac{\lambda}{N}\mathbf{I}\right)^{-1} G^T Y, \quad \text{where} \quad G_{ik} = g_k(\boldsymbol{x}_n) \tag{B.19}$$

and $Y = [y_1, \ldots, y_n]^\top$ denotes the output vector, and $\mathbf{I}$ an identity matrix. For $\lambda = 0$, this corresponds to the calculation of a pseudo-inverse of G.

So, we simultaneously adjust the output weights and the RBF centers and variances (see Algorithm B.11 for pseudo-code of this algorithm). In this way, the network fine-tunes itself to the data after the initial clustering step, yet, of course, overfitting has to be avoided by careful tuning of the regularization parameter, the number of centers K and the number of iterations [cf. Bishop, 1995]. In our experiments we always used $\lambda = 10^{-6}$ and up to ten conjgate gradient iterations [e.g. Press et al., 1992].

B.6.2 Kernel functions

Suppose we wish to construct ensembles of functions that themselves are linear combinations of other functions (e.g. of kernel functions) using coefficient $\boldsymbol{\gamma}$, i.e. functions of the form $k_n(\cdot) \equiv k(\boldsymbol{x}_n, \cdot)$:

$$h^{\boldsymbol{\gamma}}(\boldsymbol{x}) := \sum_{n=1}^{N} \gamma_n k(\boldsymbol{x}_n, \boldsymbol{x}), \quad \boldsymbol{\gamma} \in \mathbb{R}^N. \tag{B.20}$$

The set $\{h^{\boldsymbol{\gamma}}\}$ is an infinite hypothesis set and is unbounded, if $\boldsymbol{\gamma}$ is unbounded. So, one has to restrict $\boldsymbol{\gamma}$ – here we consider bounding the ℓ_1-norm of $\boldsymbol{\gamma}$ by some constant, e.g. $\mathsf{H} := \{h^{\boldsymbol{\gamma}} \mid \|\boldsymbol{\gamma}\|_1 \leq 1, \boldsymbol{\gamma} \in \mathbb{R}^N\}$. Then the problem (5.32) (for $\tau(x) = x$) has a closed

Algorithm B.11 The RBF net algorithm [Müller et al., 1999]

1. **Input:** Training sample $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, Number of RBF centers K, Regularization constant λ, Number of iterations O
2. **Initialize:** Run K-means clustering to find initial values for $\boldsymbol{\mu}_k$
   Determine $\sigma_k, k = 1, \ldots, K$, as the distance between $\boldsymbol{\mu}_k$ and the closest $\boldsymbol{\mu}_i$ $(i \neq k)$.
3. **Do for** $o = 1, \ldots, O$,
   (a) Compute optimal output weights $\boldsymbol{w} = \left(G^\top G + 2\frac{\lambda}{N}\mathbf{I}\right)^{-1} G^\top Y$
   (b) Compute gradients $\frac{\partial E}{\partial \boldsymbol{\mu}_k}$ and $\frac{\partial E}{\partial \sigma_k}$ as in (B.18) and (B.17) with optimal $\boldsymbol{w}$ and form a gradient vector $\mathbf{v}$
   (c) Estimate the conjugate direction $\overline{\mathbf{v}}$ with CG-Method as proposed in Press et al. [1992]
   (d) Perform a line search to find the minimizing step size δ in direction $\overline{\mathbf{v}}$; in each evaluation of E recompute the optimal output weights $\mathbf{w}$ as in step (a)
   (e) Update $\boldsymbol{\mu}_k$ and σ_k with $\overline{\mathbf{v}}$ and δ
4. **Output:** Optimized RBF net

form solution: Let j^* be the maximum absolute sum of the kernel values weighted by $\boldsymbol{p}$:

$$j^* = \operatorname*{argmax}_{j=1,\ldots,N} \sum_{n=1}^{N} d_n k(\boldsymbol{x}_j, \boldsymbol{x}_n).$$

Then $h^{\boldsymbol{\gamma}}$ with $\boldsymbol{\gamma} = [0, \ldots, 0, \gamma_{j^*}, 0, \ldots, \mathbf{0}]$ is a solution to (5.32), where $\gamma_{j^*} = \operatorname{sign}\left(\sum_{n=1}^{N} d_n k(\boldsymbol{x}_{j^*}, \boldsymbol{x}_n)\right)$. This means, if we "boost" linear combinations of kernel functions bounded by the ℓ_1-norm of $\boldsymbol{\gamma}$, then we will be adding in exactly one kernel basis function $k(\boldsymbol{x}_{j^*}, \cdot)$ per iteration. The resulting problem will be exactly the same as if we were optimizing a SVM regression LP [e.g. Smola et al., 1999] in the first place. The only difference is that we have now defined an algorithm for optimizing the function by adding one kernel basis at a time. So while we posed this problem as a semi-infinite learning problem it is exactly equivalent to the finite SVM case where the set of hypotheses being boosted is the individual kernel functions $k(\boldsymbol{x}_n, \boldsymbol{x})$.

If the γ_i were bounded using different norms then this would no longer be true. We would be adding functions that were the sum of many kernel functions [Rätsch et al., 2002, for using the ℓ_2-norm, see]. Then one may solve

$$\boldsymbol{\gamma} = \operatorname*{argmin}_{\boldsymbol{\gamma}} \sum_{n=1}^{N} (d_n - h^{\boldsymbol{\gamma}}(\boldsymbol{x}_n))^2 - h^{\boldsymbol{\gamma}}(\boldsymbol{x}_n)^2 + C' \|\boldsymbol{\gamma}\|_2^2,$$

where we use $\|\boldsymbol{\gamma}\|_2^2$ as a regularizer that effectively bounds the ℓ_2-norm of $\boldsymbol{\gamma}$ [e.g. Smola, 1998]. The problem has a simple solution: $\gamma_k = \frac{1}{C'} \sum_{n=1}^{N} d_n g_k(\boldsymbol{x}_n)$. This case is in particular interesting when using neural networks which linearly combine several units for the output layer.

Likewise, if we performed an active kernel strategy, where the set of kernels is param-

eterized over some set then the algorithm would change. We consider this problem in the next section.

B.6.3 (Active) Kernel Functions

Now consider the case where we choose a set of basis functions (nonlinearly) parameterized by some vector $\boldsymbol{\gamma}$. In this case since the basis function is parameterized over a set of continuous values $\boldsymbol{\gamma}$, we will have an infinite set of hypothesis. Say for example we wish to pick the a RBF kernel function with parameters $\boldsymbol{\mu}$ (the center) and σ^2 (the variance), i.e. $\boldsymbol{\gamma} = [\boldsymbol{\mu}, \sigma]$. Then we choose the hypothesis function ($s = \dim(\mathbb{X})$)

$$h^{(\hat{\boldsymbol{\mu}},\hat{\sigma})}(\boldsymbol{x}) = \frac{1}{(2\pi\hat{\sigma}^2)^{s/2}} \exp\left(-\frac{\|\boldsymbol{x} - \hat{\boldsymbol{\mu}}\|_2^2}{2\hat{\sigma}^2}\right), \tag{B.21}$$

with parameters $(\hat{\boldsymbol{\mu}}, \hat{\sigma})$ that maximize the correlation between weight $\boldsymbol{p}$ and the output, i.e.

$$(\hat{\boldsymbol{\mu}}, \hat{\sigma}) = \operatorname*{argmax}_{\boldsymbol{\mu},\sigma} E(\boldsymbol{\mu}, \sigma) \quad \text{where} \quad E(\boldsymbol{\mu}, \sigma) := \sum_{n=1}^{N} p_n h^{(\boldsymbol{\mu},\sigma)}(\boldsymbol{x}_n), \tag{B.22}$$

With reasonable assumptions, this is a bounded function that is in $\boldsymbol{p}$ and all of the results for the semi-infinite case hold.

There are several ways to efficiently find $\hat{\boldsymbol{\mu}}$ and $\hat{\sigma}$. The straight-forward way is to employ some standard nonlinear optimization technique to maximize (B.22). However, for RBF kernels as in (B.21) with fixed variance σ^2 there is a fast and easy to implement EM-like strategy [Bishop, 1995]. By setting $\nabla_{\boldsymbol{\mu}} E(\boldsymbol{\mu}, \sigma) = \mathbf{0}$, we get $\boldsymbol{\mu} = \sum_{n=1}^{N} q_n \boldsymbol{x}_n$, where $q_n = Z p_n \exp\left(\frac{-\|\boldsymbol{x}-\boldsymbol{\mu}\|_2^2}{2\sigma^2}\right)$, and Z is a normalization factor such that $\sum_{n=1}^{N} q_n = 1$. By this update, we are computing the weighted center of the data, where the weights depend on $\boldsymbol{p}$. Note, for given vector $\mathbf{q}$, one can compute (M-step) the optimal center $\boldsymbol{\mu}$. However, $\mathbf{q}$ depends on $\boldsymbol{\mu}$ and one has to iteratively recompute $\mathbf{q}$ (E-step). The iteration can be stopped, if $\sum_{n=1}^{N} q_0 = 0$ or $\boldsymbol{\mu}$ does not change anymore. As the objective function has local minima, one may start at a random position, e.g. at a random training point.

B.6.4 Classification Functions

Here we consider the case of using a linear combination of classification functions whose output is ± 1 to form a regression function. An example of such an algorithm is the Tree-Boost algorithm as in Friedman [1999]. For absolute error functions, Tree-Boost constructs a classification tree where the class of each point is taken to be the sign of the residual of each point, i.e. points that are overestimated are assigned to class -1 and points that are underestimated are assigned to class 1. A decision tree is constructed, then based on a projected gradient descent technique with an exact line-search, each point falling in a leaf node is assigned the mean value of the dependent variables of the training data falling at that node. This corresponds to a different $\hat{\alpha}_t$ for each node of the decision tree. So at each iteration, the virtual number of hypotheses added in some sense corresponds to the number of leaf nodes of the decision tree.

Here we will take a more simplified view and consider one node decision trees where the decision trees are linear combinations of the data. Specifically our decision function at each node is $f(\boldsymbol{x}, \boldsymbol{w}, b) = \text{sign}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b)$. Thus at each iteration of the algorithm we want to

$$(\hat{\boldsymbol{w}}, \hat{b}) = \underset{\boldsymbol{w}, b}{\text{argmax}} \left\{ \sum_{n=1}^{N} d_n \, \text{sign}(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b) \right\} \tag{B.23}$$

Note that there are only finitely many ways to label N points so this is a finite set of hypotheses. There are infinitely many possible (w, b) but any that produce the same objective value are equivalent to the boosting algorithm.

The question is how to practically optimize such a problem. Clearly an upper bound on the best possible value of the above equation is obtained by any $(\boldsymbol{w}, b)$ solution satisfying $\text{sign}(f(\boldsymbol{x}_n, \boldsymbol{w}, b)) = \text{sign}(d_n)$. So in some sense, we can consider the $\text{sign}(d_n)$ to be the desired class of $\boldsymbol{x}_n$. Now it frequently may not be possible to construct such a f. Each $\boldsymbol{x}_n$ that is misclassified will be penalized by exactly $|d_n|$. Thus we can think of $|d_n|$ as the misclassification cost of $\boldsymbol{x}_n$. Given these classes, and misclassification weights, we can use any weight sensitive classification algorithm to construct a hypothesis.

In this study we used the following problem converted into LP form to construct f:

$$\begin{aligned} (\hat{w}, \hat{b}) = \quad & \underset{\boldsymbol{w}, b, \boldsymbol{\xi}}{\text{argmin}} \quad \textstyle\sum_{n=1}^{N} |d_n| \xi_n \\ & \text{with} \quad \text{sign}(d_n)(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b) \geq 1 - \xi_n, \; n = 1, \ldots, N \\ & \qquad \|\boldsymbol{w}\|_1 \leq \delta, \; \boldsymbol{\xi} \geq 0 \end{aligned} \tag{B.24}$$

where $\delta > 0$ becomes a parameter of the problem.

Some interesting facts about this formulation. The choice of δ controls the capacity of the base learners to fit the data. For a fixed choice of δ, classification functions using a relatively fixed number of w_d nonzero. So the user can determine based on experimentation on the training data, how δ effects the complexity of the base hypothesis. Then the user may fix δ according to the desired complexity of the base hypothesis. Alternatively, a weighted variation of ν-SVMs [Schölkopf et al., 2000] could be used to dynamically choose δ.

Like in TreeBoost, we would like to allow each side of the linear decision to have a different weight. We describe the changes required to Algorithm 1 to allow this. At each iteration, LP (B.24)) is solved to find a candidate hypothesis $(\hat{\boldsymbol{w}}, \hat{b})$. Then instead of adding a single column to the restricted master LP (12), two columns are added. The first column is $h_{t_+} = I(\langle \hat{\boldsymbol{w}}, \boldsymbol{x} \rangle + b > 0)$ and the second column is $h_{t_-} = -I(\langle \hat{\boldsymbol{w}}, \boldsymbol{x} \rangle + b < 0)$. The algorithms stop if both of these hypotheses do not meet the criteria given in the algorithm. The algorithm should terminate if $\sum_n h_{t_+}(\boldsymbol{x}_n) + h_{t_-}(\boldsymbol{x}_n)(\mathbf{d}_n - \mathbf{d}_n^*) < 2$. We call this variant of the algorithm CG-LP. This change has no effect on the convergence properties.