TOPICS IN REGULARIZATION AND BOOSTING

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF STATISTICS

AND THE COMMITTEE ON GRADUATE STUDIES

OF STANFORD UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Saharon Rosset

August 2003

© Copyright 2003 by Saharon Rosset

All Rights Reserved

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Jerome Friedman
(Co-Advisor)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Trevor Hastie
(Co-Advisor)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Robert Tibshirani

Approved for the University Committee on Graduate Studies.

Abstract

Regularization is critical for successful statistical modeling of "modern" data, which is high-dimensional, sometimes noisy and often contains a lot of irrelevant predictors. It exists — implicitly or explicitly — at the heart of all successful methods. The two main challenges which we take on in this thesis are understanding its various aspects better and suggesting new regularization approaches, which are motivated by both theoretical and computational considerations.

This thesis comprises a collection of papers whose common thread is *regularization* in a vague, almost abstract, sense. Two main areas are covered in this context:

- **Boosting.** We analyze and illustrate its close relation to l_1-regularized optimization (chapter 2), discuss selection of robust loss functions for boosting (chapter 4) and apply boosting methodology to a new domain of density estimation (chapter 5).

- **Parametric regularization.** We discuss methods of solving the regularized optimization problem:

$$\hat{\beta}(\lambda) = \min_{\beta} L(\mathsf{y}, X\beta) + \lambda J(\beta)$$

  for whole ranges of the regularization parameter λ. In chapter 6 we characterize families of problems (characterized by choice of "loss" L and "penalty" J) where

the whole solution path $\{\hat{\beta}(\lambda) \ : \ 0 \leq \lambda \leq \infty\}$ can be calculated efficiently because it is "piecewise linear". We suggest and analyze a generic method for generating the solution path for any "well behaved" (loss, penalty) pair in chapter 7.

Acknowledgments

First and foremost I would like to extend my everlasting gratitude and love to my wife Tammy and my daughter Nogah, who make my life beautiful. I could not dream it up better.

My advisors Jerry Friedman and Trevor Hastie have been an inspiration to me and have shown great support throughout my years in the Ph.D. program. I have learned from them on many levels.

Ji Zhu has been a great collaborator, without whom my time at Stanford would have been longer and less stimulating and enjoyable. He has made contributions to many of the results described here, much more than I have made to his work. I truly hope our collaboration does not end at graduation.

Many people have helped and guided me — both professionally and personally — in the last four years. Rob Tibshirani has always been supportive and instructive. Brad Efron and Stephen Boyd were on my orals committee and have helped me greatly — a lot of this thesis is based on extensions of Brad's ideas, and Stephen's work in convex optimization supplies the tools. I am also indebted to Brad for his recommendation letter. I am grateful to Noureddine El Karoui, Giles Hooker, Eran Segal, Marc Coram, Carrie Grimes and all the people of the Hastie-Tibshirani research group for various combinations of help, support and inspiration.

David Steinberg of Tel Aviv University encouraged me to embark on this adventure, and I thank him for it.

Biss has been with me throughout these four years, always a loyal and loving companion. The happiness she brings far exceeds the trouble she got me into with some dog-unfriendly members of our department. While we are on the subject, Biss demands an opportunity to thank Pat for *always* having treats.

Finally, I would like to thank my parents, Aviva and Shmuel, who have fully supported me despite my continued absence, which deprived them of their son, and more importantly their granddaughter.

Contents

Abstract **iv**

Acknowledgments **vi**

1 Introduction **1**

1.1 Illustrative example 3

1.2 Boosting and regularized model fitting 5

1.3 Structure of this thesis 6

2 Boosting, regularized optimization and margin maximization **11**

2.1 Introduction and outline 11

2.1.1 Related work 15

2.2 Boosting as gradient descent 16

2.2.1 Practical implementation of boosting 18

2.2.2 Gradient-based boosting as a generic modeling tool 19

2.2.3 Loss functions 20

2.2.4 Line-search boosting vs. ϵ-boosting 21

2.3 l_p margins, support vector machines and boosting 22

2.3.1 The Euclidean margin and the support vector machine 22

2.3.2 The l_1 margin and its relation to boosting 23

2.4 Boosting as approximate l_1 constrained fitting 26
2.5 l_p-constrained classification loss functions 31
2.5.1 Implications of theorem 2.3 35
2.6 Examples 36
2.6.1 Spam dataset 36
2.6.2 Simulated data 37
2.7 Discussion 40
2.7.1 Regularized and non-regularized behavior of the loss functions 40
2.7.2 Boosting and SVMs as regularized optimization in high-dimensional predictor spaces 41
2.8 Summary and future work 46

3 Margin maximizing loss functions 48
3.1 Introduction 48
3.2 Sufficient condition for margin maximization 50
3.2.1 Necessity results 54
3.3 Examples 54
3.4 A multi-class generalization 56
3.4.1 Margin maximization in multi-class SVM and logistic regression 59
3.5 Discussion 60

4 Robust boosting and its relation to bagging 62
4.1 Introduction 62
4.2 Boosting, bagging and a connection 63
4.2.1 Robustness, convexity and boosting 67
4.2.2 Intermediate approaches 68
4.3 Boosting with weight decay 69
4.4 Experiments 71

4.5 Discussion 74

5 Boosting density estimation **76**

5.1 Introduction 76

5.2 A density estimation boosting algorithm 78

5.3 Training data performance 80

5.4 Boosting Bayesian Networks 83

5.5 Experimental Results 85

5.6 Discussion and future work 88

6 Adaptable, Efficient and Robust Methods for Regression and Classification Via Piecewise Linear Regularized Coefficient Paths **90**

6.1 Introduction 91

6.1.1 Illustrative example 93

6.2 A general theory and a useful *(loss, penalty)* family 96

6.2.1 Piecewise linear coefficient paths in twice differentiable loss and penalty models 96

6.2.2 "Almost quadratic" loss functions with Lasso penalty 98

6.2.3 Computational considerations 104

6.3 Robust methods for regression and classification 105

6.3.1 The Huberized Lasso for regression 106

6.3.2 The Huberized truncated squared loss for classification 109

6.4 Other piecewise linear models of interest 113

6.4.1 Using l_1 loss and its variants 113

6.4.2 The l_∞ penalty 114

6.4.3 Multiple penalty problems 117

6.5 Discussion: robustness and regularization 118

7 Approximate quadratic method for following curved solution paths 120

7.1 Algorithm 120

7.2 Data examples 122

7.3 Theoretical result 124

7.4 Some extensions 125

7.4.1 The Lasso and other non-twice-differentiable penalties 125

7.4.2 Example: logistic regression with lasso penalty 126

A Proofs of theorems 128

Bibliography 141

Chapter 1

Introduction

What is regularization?

My best definition is that regularization is any part of model building which takes into account — implicitly or explicitly — the finiteness and imperfection of the data and the limited information in it, which we can term "variance" in an abstract sense. By this definition, forms of regularization include but are not limited to:

1. Explicit constraints on model complexity, such as bounds on the l_p norm of a parameter or coefficient vector β which characterizes the model. This is the obvious sense in which "regularization" is used and it includes ridge regression and the lasso. This regularization has the form:

$$\min_{\beta} \text{loss}(\text{data}, \beta) \text{ s.t. } \|\beta\|_p^p \leq S \tag{1.1}$$

   It is interesting to note that constrained models can equivalently be cast as penalized models:

$$\min_{\beta} \text{loss}(\text{data}, \beta) + \lambda\|\beta\|_p^p \tag{1.2}$$

The two formulations are equivalent in the sense that for any value of S in (1.1) there exists a value of λ in (1.2) such that the solutions of the two problems are the same.

2. Implicit regularization through incremental building of the model. The prime example here is boosting (Freund & Schapire 1995), which iteratively builds a prediction model by combining "weak learners". Stopping the boosting iterations is implicitly selecting a complexity (or regularization) parameter for the model. One of the main results of this thesis (and other recent works) is to show that early stopping of boosting in fact corresponds — approximately — to parametric regularization of the form (1.1,1.2) with $p = 1$, i.e. the lasso penalty.

3. Choice of the empirical loss function for model fitting can also be affected by regularization considerations. The most obvious expression of this concept is equation (1.2) above, which can be interpreted as simply *changing the loss* to include a complexity penalty term.

Choice of robust loss functions is not strictly a regularization procedure, as robustness is related more to uncertainty in assumptions about the error than to variance. However, we know that for high-dimensional data every data point is likely to be an outlier. Thus for high-dimensional data regularization and robustness considerations are in fact confounded to a large extent, in the sense that any uncertainty in our assumptions about the error model will cause a large increase in variance. Hence we need to "regularize" the model by selecting a robust loss function. See chapter 6 for further discussion of this issue.

The main theme of this thesis is to discuss regularization in its many forms from theoretical, statistical and computational perspectives:

- What are good regularization approaches and why? What are good empirical loss functions?
- What connections can we draw between existing successful modeling approaches by examining the regularization schemes they employ?
- What can we say about the way a regularized model evolves as we change the regularization, in particular the regularization parameter in (1.2)?
- How can we calculate regularized solutions efficiently? For what problems? How can we calculate a "path" of regularized solutions to (1.2), parameterized by λ?

1.1 Illustrative example

The following simple simulated example (discussed in more detail in chapter 6) illustrates the importance of all three component of regularization for building good prediction models:

- An appropriate loss function, i.e. one that either matches data distribution or is robust enough to account for lack of knowledge of this distribution.
- An appropriate regularization scheme.
- Selection of an appropriate value for the regularization parameter(s).

Take $n = 100$ observations and $k = 80$ predictors, where all x_{ij} are i.i.d $N(0, 1)$ and the true model is:

$$y_i = 10 \cdot x_{i1} + \epsilon_i \tag{1.3}$$

$$\epsilon_i \overset{iid}{\sim} 0.9 \cdot N(0, 1) + 0.1 \cdot N(0, 100) \tag{1.4}$$

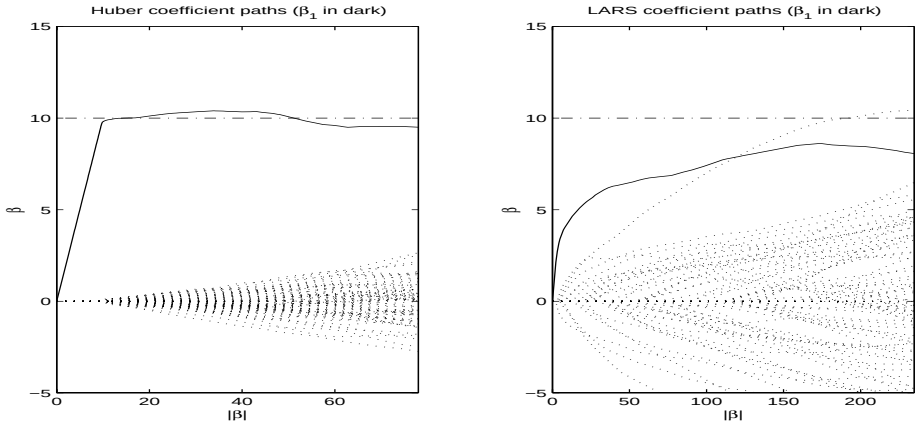


Figure 1.1: Coefficient paths for Huberized lasso (left) and lasso(right) for data example. β_1 is the full line, and the true model is $E(Y|x) = 10x_1$

So the normality of residuals, implicitly assumed by using squared error loss, is violated.

We solve two parametric regularization problems on this data:

$$\text{Lasso:} \qquad \hat{\beta}(\lambda) = \arg\min_{\beta} \sum_i (y_i - \beta' \mathbf{x}_i)^2 + \lambda \|\beta\|_1 \tag{1.5}$$

$$\text{“Huber” Lasso:} \qquad \hat{\beta}(\lambda) = \arg\min_{\beta} \sum_i L_h(y_i, \beta' \mathbf{x}_i) + \lambda \|\beta\|_1 \tag{1.6}$$

$$\text{with} \qquad L_h(y, \beta'\mathbf{x}) = \begin{cases} (y - \beta'\mathbf{x})^2 & \text{if } |y - \beta'\mathbf{x}| \le 1 \\ 1 + 2(|y - \beta'\mathbf{x}| - 1) & \text{otherwise} \end{cases}$$

The l_1(lasso) regularization approach is appropriate here, since we know the true model to be “sparse” with only one non-0 coefficient.

Figure 1.1 shows the optimal coefficient paths $\hat{\beta}(\lambda)$ for the lasso (1.5, right) and “huberized” lasso, (1.6, left). We observe that the lasso fails in identifying the correct model $E(Y|x) = 10x_1$ while the robust loss identifies it exactly, *if we choose the appropriate regularization parameter*. Figure 1.2 gives the “reducible” squared error loss of the models along the two regularized paths and illustrates the superiority of

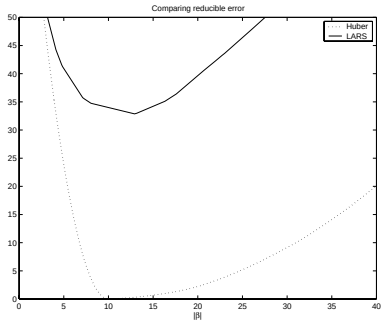


Figure 1.2: Reducible error for the models along the regularized paths

the robust loss in that respect as well.

Thus we see that in this simple example, the correct combination of loss, regularization method and regularization parameter choice allows us to identify the correct model exactly, while failing on any of these three components significantly degrades the performance.

1.2 Boosting and regularized model fitting

We focus special attention on boosting, a generic modeling approach which has attracted a lot of attention in the machine learning, data mining and statistics communities over the last five years. This interest can be attributed to its great empirical success in data modeling as well as its interesting theoretical properties. Chapters 2,4 and 5 of this thesis cover various aspects of boosting theory and methodology.

Boosting can most generally be described as a method for iteratively building an additive model

$$F_T(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_{j_t}(\mathbf{x}), \tag{1.7}$$

where $h_{j_t} \in \mathcal{H}$ — a large (or even infinite) dictionary of candidate predictors or "weak learners"; and h_{j_t} is the basis function selected as the "best candidate" to modify at

stage t. The essence of boosting is to design a way to "adaptively" select the next increment at each step, to improve the fit. The original boosting algorithm (Freund & Schapire 1995) and many of its successors described boosting in vague terms of "concentrating on difficult examples", game theory analogies and more. Our view of their algorithm — and of the essence of boosting — is that it is just a gradient descent algorithm (more specifically, a coordinate descent algorithm) in the high dimensional space spanned by the weak learners. In this view we follow Friedman (2001),Mason, Baxter, Bartlett & Frean (1999) and others. We build on this view to use boosting to solve a new problem in chapter 4 and to design a new family of boosting algorithms in chapter 5.

Taking this point of view, it is clear that boosting follows a "regularized path" of solutions in an abstract sense — the solutions get more complex and more "overfitted" as the boosting iterations proceed. However one of the main results of this thesis, presented in chapter 2, is that boosting can in fact be described as following — approximately — the path of l_1-regularized solutions to its loss criterion. Boosting doesn't merely trace a path of models of increasing complexity, it in fact approximately applies a specific, well researched and in many regards reasonable regularization scheme. We feel that this property is fundamental in boosting's success, and that understanding boosting's regularization behavior is key in improving boosting methodology and devising new algorithms.

1.3 Structure of this thesis

This thesis is organized as a collection of papers, each one presenting some aspect of boosting, regularized fitting and loss function selection. There are many connections and cross-references between the chapters, but I have tried to make them as

modular and self contained as possible, and construct each chapter around one central result. Chapters 2-5 are "machine learning" oriented and concentrate on issues surrounding boosting, support vector machines and logistic regression. Chapters 6-7 deal with more traditional statistical regularization approaches, and concentrate on their computational aspects.

Here is a brief exposition of the contents of the chapters which attempts to put them in context relative to the main theme and to the other chapters:

Chapter 2: Boosting, regularized optimization and margin maximization

In this chapter we study boosting methods from a new perspective. We build on recent work by Efron et al. to show that boosting approximately (and in some cases exactly) minimizes its loss criterion with an l_1 constraint on the coefficient vector. This helps understand the success of boosting with early stopping as regularized fitting of the loss criterion. For the two most commonly used criteria (exponential and binomial log-likelihood), we further show that as the constraint is relaxed — or equivalently as the boosting iterations proceed — the solution converges (in the separable case) to an "l_1-optimal" separating hyper-plane. We prove that this l_1-optimal separating hyper-plane has the property of maximizing the minimal l_1-margin of the training data, as defined in the boosting literature. An interesting fundamental similarity between boosting and kernel support vector machines emerges, as both can be described as methods for regularized optimization in high-dimensional predictor space, utilizing a computational trick to make the calculation practical, and converging to margin-maximizing solutions. While this statement describes SVMs exactly, it applies to boosting only approximately.

This chapter represents joint work with Ji Zhu and is based on Rosset, Zhu &

Hastie (2003).

Chapter 3: Margin maximizing loss functions

Margin maximizing properties play an important role in the analysis of classification models, such as boosting and support vector machines. Margin maximization is theoretically interesting because it facilitates generalization error analysis, and practically interesting because it presents a clear geometric interpretation of the models being built. In chapter 2 we show margin maximizing properties of the exponential and logistic loss functions in 2-class problems. In this chapter we extend the discussion to formulate and prove a sufficient condition for the solutions of regularized loss functions to converge to margin maximizing separators, as the regularization vanishes. This condition covers the hinge loss of SVM, as well the exponential loss of AdaBoost and logistic regression loss discussed in chapter 2. We also generalize this result to multi-class classification problems, and present margin maximizing multi-class versions of logistic regression and support vector machines.

This chapter represents joint work with Ji Zhu.

Chapter 4: Robust boosting and its relation to bagging

In the gradient-based view of boosting introduced in chapter 2, at each iteration observations are re-weighted using the gradient of the underlying loss function. This chapter presents an approach of weight decay for observation weights which is equivalent to "robustifying" the underlying loss function. At the extreme end of decay this approach converges to Bagging, which can be viewed as boosting with a linear underlying loss function. We illustrate the practical usefulness of weight decay for improving prediction performance and present an equivalence between one form of weight decay and "Huberizing" — a statistical method for making loss functions more

robust. Our examples show that "robustifying" the loss function in high dimensional data can be useful even when there are no outliers.

Chapter 5: Boosting density estimation

In this chapter, we apply gradient-based boosting methodology to the unsupervised learning problem of density estimation. We show convergence properties of the algorithm and prove that a "strength of weak learnability" property applies to this problem as well. We illustrate the potential of this approach through experiments with boosting Bayesian networks to learn density models. This chapter represents a slight deviation from the "regularization" theme as its main goal is to suggest and implement a practical boosting algorithm to a new "kind" of problem — that of density estimation.

This chapter represents joint work with Eran Segal and is based on Rosset & Segal (2002).

Chapter 6: Efficient, robust and adaptable methods for regression and classification via piecewise linear solution paths

We consider the generic regularized optimization problem

$$\hat{\beta}(\lambda) = \arg\min_{\beta} \sum_{i} L(y_i, \beta' x_i) + \lambda J(\beta)$$

Recently Efron, Hastie, Johnstone & Tibshirani (2002) have shown that for the Lasso — i.e. if L is squared error loss and $J(\beta) = \|\beta\|_1$ is the l_1 norm of β — the optimal coefficient path is piecewise linear, i.e. $\nabla\hat{\beta}(\lambda)$ is piecewise constant. We derive a general characterization of the properties of *(loss L, penalty J)* pairs which give piecewise linear coefficient paths. Such pairs allow for efficient generation of the

full regularized coefficient paths. We analyze in detail the solution for l_1-penalized Huber loss for regression, and l_1-penalized truncated Huber loss for classifications, and illustrate how we can use our results to generate robust, efficient and adaptable modeling tools.

This chapter represents joint work with Ji Zhu.

Chapter 7: Approximate methods for following curved regularized optimization solution paths

In this chapter I present a general approach for following "curved" regularized optimization solution paths, using a modified Newton-Raphson approach. This approach can be used when the "piecewise linearity" property discussed in chapter 6 does not hold. We prove that this approach is good in the sense that it guarantees staying "close" to the path of optimal solutions, and illustrate it on regularized logistic regression.

Chapter 2

Boosting, regularized optimization and margin maximization

2.1 Introduction and outline

Boosting is a method for iteratively building an additive model

$$F_T(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_{j_t}(\mathbf{x}), \tag{2.1}$$

where $h_{j_t} \in \mathcal{H}$ — a large (but we will assume finite) dictionary of candidate predictors or "weak learners"; and h_{j_t} is the basis function selected as the "best candidate" to modify at stage t. The model F_T can equivalently be represented by assigning a coefficient to each dictionary function $h \in \mathcal{H}$ rather than to the selected h_{j_t}'s only:

$$F_T(\mathbf{x}) = \sum_{j=1}^{J} h_j(x) \cdot \beta_j^{(T)}, \tag{2.2}$$

where $J = |\mathcal{H}|$ and $\beta_j^{(T)} = \sum_{j_t=j} \alpha_t$. The "$\beta$" representation allows us to interpret the coefficient vector $\beta^{(T)}$ as a vector in $\mathcal{R}^J$ or, equivalently, as the hyper-plane which has $\beta^{(T)}$ as its normal. This interpretation will play a key role in our exposition.

Some examples of common dictionaries are:

- The training variables themselves, in which case $h_j(\mathbf{x}) = x_j$. This leads to our "additive" model F_T being just a linear model in the original data. The number of dictionary functions will be $J = d$, the dimension of $\mathbf{x}$.
- Polynomial dictionary of degree p, in which case the number of dictionary functions will be $J = \begin{pmatrix} p+d \\ d \end{pmatrix}$
- Decision trees with up to k terminal nodes, if we limit the split points to data points (or mid-way between data points as CART does). The number of possible trees is bounded from above (trivially) by $J \leq (np)^k \cdot 2^{k^2}$. Note that regression trees do not fit into our framework, since they will give $J = \infty$.

The boosting idea was first introduced by Freund and Schapire (Freund & Schapire 1995), with their AdaBoost algorithm. AdaBoost and other boosting algorithms have attracted a lot of attention due to their great success in data modeling tasks, and the "mechanism" which makes them work has been presented and analyzed from several perspectives. Friedman, Hastie & Tibshirani (2000) develop a statistical perspective, which ultimately leads to viewing AdaBoost as a gradient-based incremental search for a good additive model (more specifically, it is a "coordinate descent" algorithm), using the exponential loss function $C(y, F) = \exp(-yF)$, where $y \in \{-1, 1\}$. The gradient boosting (Friedman 2001) and anyboost (Mason, Baxter, Bartlett & Frean 1999) generic algorithms have used this approach to generalize the boosting idea to wider families of problems and loss functions. In particular, Friedman, Hastie & Tibshirani (2000) have pointed out that the binomial log-likelihood loss $C(y, F) = \log(1 +$

$\exp(-yF)$) is a more natural loss for classification, and is more robust to outliers and misspecified data.

A different analysis of boosting, originating in the Machine Learning community, concentrates on the effect of boosting on the margins $y_i F(\mathbf{x}_i)$. For example, Schapire, Freund, Bartlett & Lee (1998) uses margin-based arguments to prove convergence of boosting to perfect classification performance on the training data under general conditions, and to derive bounds on the generalization error (on future, unseen data).

In this chapter we combine the two approaches, to conclude that gradient-based boosting can be described, in the separable case, as an approximate margin maximizing process. The view we develop of boosting as an approximate path of optimal solutions to regularized problems also justifies early stopping in boosting as specifying a value for "regularization parameter".

We consider the problem of minimizing convex loss functions (in particular the exponential and binomial log-likelihood loss functions) over the training data, with an l_1 bound on the model coefficients:

$$\hat{\beta}(c) = \arg\min_{\|\beta\|_1 \le c} \sum_i C(y_i, h(\mathbf{x}_i)'\beta). \tag{2.3}$$

Where $h(\mathbf{x}_i) = [h_1(\mathbf{x}_i), h_2(\mathbf{x}_i), \ldots, h_J(\mathbf{x}_i)]^T$ and $J = |\mathcal{H}|$.

Hastie, Tibshirani & Friedman (2001) (chapter 10) have observed that "slow" gradient-based boosting (i.e. we set $\alpha_t = \epsilon \, , \forall t$ in (2.1), with ϵ small) tends to follow the penalized path $\hat{\beta}(c)$ as a function of c, under some mild conditions on this path. In other words, using the notation of (2.2), (2.3), this implies that $\|\beta^{(c/\epsilon)} - \hat{\beta}(c)\|$ vanishes with ϵ, for all (or a wide range of) values of c. Figure 2.1 illustrates this equivalence between ϵ-boosting and the optimal solution of (2.3) on a real-life data set, using squared error loss as the loss function. In this chapter we demonstrate this equivalence further and formally state it as a conjecture. Some progress towards

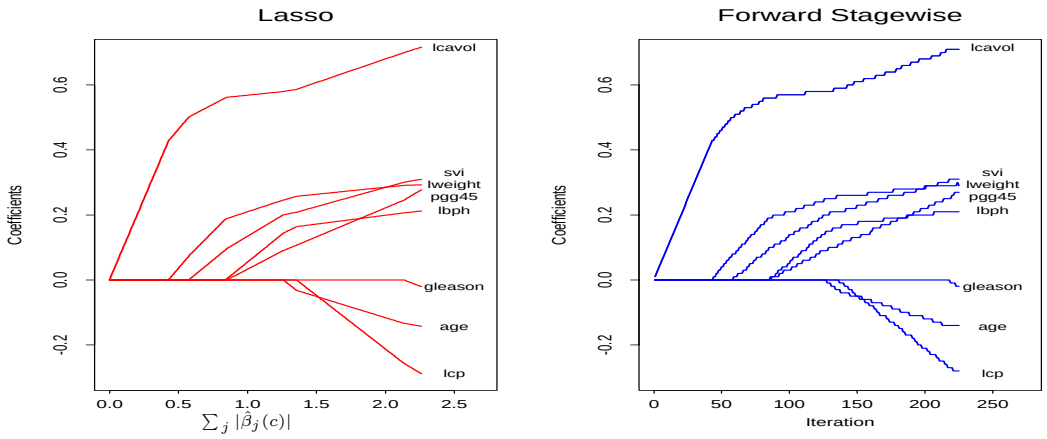


Figure 2.1: Exact coefficient paths(left) for l_1-constrained squared error regression and "boosting" coefficient paths (right) on the data from a prostate cancer study

proving this conjecture has been made by Efron et al. (Efron, Hastie, Johnstone & Tibshirani 2002) who prove a weaker "local" result for the case where C is squared error loss, under some mild conditions on the optimal path. We generalize their result to general convex loss functions.

Combining the empirical and theoretical evidence, we conclude that boosting can be viewed as an approximate incremental method for following the l_1-regularized path.

We then prove that in the separable case, for both the exponential and logistic log-likelihood loss functions, $\hat{\beta}(c)/c$ converges as $c \to \infty$ to an "optimal" separating hyper-plane $\hat{\beta}$ described by:

$$\hat{\beta} = \arg \max_{\|\beta\|_1=1} \min_i y_i \beta' h(\mathbf{x}_i) \tag{2.4}$$

In other words, $\hat{\beta}$ *maximizes the minimal margin* among all vectors with l_1-norm equal to 1. This result generalizes easily to other l_p-norm constraints. For example, if $p = 2$, then $\hat{\beta}$ describes the optimal separating hyper-plane in the Euclidean sense, i.e. the same one that a non-regularized support vector machine would find.

Combining our two main results, we get the following characterization of boosting:

> ϵ-Boosting can be described as a gradient-descent search, approximately following the path of l_1-constrained optimal solutions to its loss criterion, and converging, in the separable case, to a "margin maximizer" in the l_1 sense.

Note that boosting with a large dictionary $\mathcal{H}$ (in particular if $n < J = |\mathcal{H}|$) guarantees that the data will be separable (except for pathologies), hence separability is a very mild assumption here.

As in the case of support vector machines in high dimensional feature spaces, the non-regularized "optimal" separating hyper-plane is usually of theoretical interest only, since it typically represents a highly over-fitted model. Thus, we would want to choose a good regularized model. Our results indicate that Boosting gives a natural method for doing that, by "stopping early" in the boosting process. Furthermore, they point out the fundamental similarity between Boosting and SVMs: both approaches allow us to fit regularized models in high-dimensional predictor space, using a computational trick. They differ in the regularization approach they take – exact l_2 regularization for SVMs, approximate l_1 regularization for Boosting – and in the computational trick that facilitates fitting – the "kernel" trick for SVMs, coordinate descent for Boosting.

2.1.1 Related work

Schapire, Freund, Bartlett & Lee (1998) have identified the normalized margins as distance from an l_1-normed separating hyper-plane. Their results relate the boosting iterations' success to the minimal margin of the combined model. Rätsch, Onoda & Müller (2001) take this further using an asymptotic analysis of AdaBoost. They

prove that the "normalized" minimal margin, $\min_i y_i \sum_t \alpha_t h_t(\mathbf{x}_i) / \sum_t |\alpha_t|$, is asymptotically equal for both classes. In other words, they prove that the asymptotic separating hyper-plane is equally far away from the closest points on either side. This is a property of the margin maximizing separating hyper-plane as we define it. Both papers also illustrate the margin maximizing effects of AdaBoost through experimentation. However, they both stop short of proving the convergence to optimal (margin maximizing) solutions.

Motivated by our result, Rätsch & Warmuth (2002) have recently asserted the margin-maximizing properties of ϵ-AdaBoost, using a different approach than the one used in this chapter. Their results relate only to the asymptotic convergence of infinitesimal AdaBoost, compared to our analysis of the "regularized path" traced along the way and of a variety of boosting loss functions, which also leads to a convergence result on binomial log-likelihood loss.

The convergence of boosting to an "optimal" solution from a loss function perspective has been analyzed in several papers. Rätsch, Mika & Warmuth (2001) and Collins, Schapire & Singer (2000) give results and bounds on the convergence of training-set loss, $\sum_i C(y_i, \sum_t \alpha_t h_t(\mathbf{x}_i))$, to its minimum. However, in the separable case convergence of the loss to 0 is inherently different from convergence of the linear separator to the optimal separator. Any solution which separates the two classes perfectly can drive the exponential (or log-likelihood) loss to 0, simply by scaling coefficients up linearly.

2.2 Boosting as gradient descent

Generic gradient-based boosting algorithms such as Friedman (2001) and Mason, Baxter, Bartlett & Frean (1999) attempt to find a good linear combination of the members of some dictionary of basis functions to optimize a given loss function over

a sample. This is done by searching, at each iteration, for the basis function which gives the "steepest descent" in the loss, and changing its coefficient accordingly. In other words, this is a "coordinate descent" algorithm in $\mathcal{R}^J$, where we assign one dimension (or coordinate) for the coefficient of each dictionary function.

Assume we have data $\{\mathbf{x}_i, y_i\}_{i=1}^n$, a loss (or cost) function $C(y, F)$, and a set of basis functions $\{h_j(\mathbf{x})\} : \mathcal{R}^d \to \mathcal{R}$. Then all of these algorithms follow the same essential steps:

Algorithm 2.1 *Generic gradient-based boosting algorithm*

1. *Set* $\beta^{(0)} = 0$.
2. *For* $t = 1 : T$,
   (a) *Let* $F_i = \beta^{(t-1)'} h(\mathbf{x}_i),\ i = 1, \ldots, n$ *(the current fit).*
   (b) *Set* $w_i = \frac{\partial C(y_i, F_i)}{\partial F_i},\ i = 1, \ldots, n$.
   (c) *Identify* $j_t = \arg\max_j |\sum_i w_i h_j(\mathbf{x}_i)|$.
   (d) *Set* $\beta_{j_t}^{(t)} = \beta_{j_t}^{(t-1)} - \alpha_t$ *and* $\beta_k^{(t)} = \beta_k^{(t-1)}, k \neq j_t$.

Here $\beta^{(t)}$ is the "current" coefficient vector and $\alpha_t > 0$ is the current step size. Notice that $\sum_i w_i h_{j_t}(\mathbf{x}_i) = \frac{\partial \sum_i C(y_i, F_i)}{\partial \beta_{j_t}}$.

As we mentioned, algorithm 1 can be interpreted simply as a coordinate descent algorithm in "weak learner" space. Implementation details include the dictionary $\mathcal{H}$ of "weak learners", the loss function $C(y, F)$, the method of searching for the optimal j_t and the way in which $|\alpha_t|$ is determined (the sign of α_t will always be $-sign(\sum_i w_i h_{j_t}(\mathbf{x}_i))$, since we want the loss to be reduced. In most cases, the dictionary $\mathcal{H}$ is negation closed, and so it is assumed WLOG that $\alpha_t > 0$). For example, the original AdaBoost algorithm uses this scheme with the exponential loss $C(y, F) = \exp(-yF)$, and an implicit line search to find the best α_t once a "direction"

j has been chosen (see Hastie, Tibshirani & Friedman (2001), chapter 10 and Mason, Baxter, Bartlett & Frean (1999) for details). The dictionary used by AdaBoost in this formulation would be a set of candidate classifiers, i.e. $h_j(\mathbf{x}_i) \in \{-1, +1\}$ — usually decision trees are used in practice.

2.2.1 Practical implementation of boosting

The dictionaries used for boosting are typically very large – practically infinite – and therefore the generic boosting algorithm we have presented cannot be implemented verbatim. In particular, it is not practical to exhaustively search for the maximizer in step 2(c). Instead, an approximate, usually greedy search is conducted to find a "good" candidate weak learner h_{j_t} which makes the first order decline in the loss large (even if not maximal among all possible models).

In the common case that the dictionary of weak learners is comprised of decision trees with up to k nodes, the way AdaBoost and other boosting algorithms solve stage 2(c) is by building a decision tree to a re-weighted version of the data, with the weights $|w_i|$ (w_i as defined above). Thus they first replace step 2(c) with minimization of:

$$\sum_i |w_i| 1\{y_i \neq h_{j_t}(\mathbf{x}_i)\}$$

which is easily shown to be equivalent to the original step 2(c). They then use a greedy decision-tree building algorithm such as CART or C5 to build a decision tree which minimizes this quantity, i.e. achieves low "weighted misclassification error" on the weighted data. Since the tree is built greedily — one split at a time — it will not be the global minimizer of weighted misclassification error among all k-node decision trees. However, it will be a good fit for the re-weighted data, and can be considered an approximation to the optimal tree.

This use of approximate optimization techniques is critical, since much of the

strength of the boosting approach comes from its ability to build additive models in *very* high-dimensional predictor spaces. In such spaces, standard exact optimization techniques are impractical: any approach which requires calculation and inversion of Hessian matrices is completely out of the question, and even approaches which require only first derivatives, such as coordinate descent, can only be implemented approximately.

2.2.2 Gradient-based boosting as a generic modeling tool

As Friedman (2001) and Mason, Baxter, Bartlett & Frean (1999) mention, this view of boosting as gradient descent allows us to devise boosting algorithms for any function estimation problem — all we need is an appropriate loss and an appropriate dictionary of "weak learners". For example, Friedman, Hastie & Tibshirani (2000) suggested using the binomial log-likelihood loss instead of the exponential loss of AdaBoost for binary classification, resulting in the LogitBoost algorithm. However, there is no need to limit boosting algorithms to classification — Friedman (2001) applied this methodology to regression estimation, using squared error loss and regression trees, and Rosset & Segal (2002) applied it to density estimation, using the log-likelihood criterion and Bayesian networks as weak learners. Their experiments and those of others illustrate that the practical usefulness of this approach — coordinate descent in high dimensional predictor space — carries beyond classification, and even beyond supervised learning.

The view we present in this paper, of coordinate-descent boosting as approximate l_1-regularized fitting, offers some insight into why this approach would be good in general: it allows us to fit regularized models directly in high dimensional predictor space. In this it bears a conceptual similarity to support vector machines, which exactly fit an l_2 regularized model in high dimensional (RKH) predictor space.

2.2.3 Loss functions

The two most commonly used loss functions for boosting classification models are the exponential and the (minus) binomial log-likelihood:

$$\begin{aligned} Exponential: &\quad C_e(y,F) = \exp(-yF) \\ Loglikelihood: &\quad C_l(y,F) = \log(1+\exp(-yF)) \end{aligned}$$

These two loss functions bear some important similarities to each other. As Friedman,

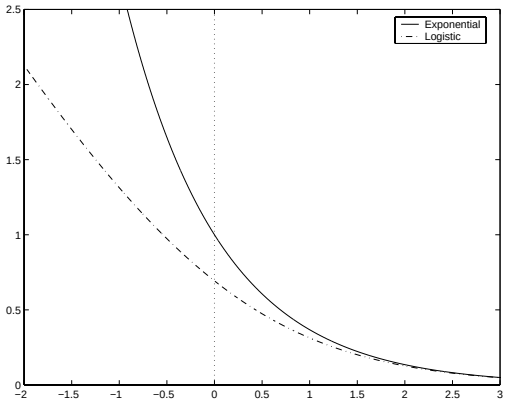


Figure 2.2: The two classification loss functions

Hastie & Tibshirani (2000) show, the population minimizer of expected loss at point $\mathbf{x}$ is similar for both loss functions and is given by:

$$\hat{F}(\mathbf{x}) = c \cdot log\left[\frac{P(y=1|\mathbf{x})}{P(y=-1|\mathbf{x})}\right]$$

where $c_e = 1/2$ for exponential loss and $c_l = 1$ for binomial loss.

More importantly for our purpose, we have the following simple proposition, which illustrates the strong similarity between the two loss functions for positive margins (i.e. correct classifications):

Proposition 2.1

$$yF \geq 0 \Rightarrow 0.5C_e(y,F) \leq C_l(y,F) \leq C_e(y,F) \tag{2.5}$$

In other words, the two losses become similar if the margins are positive, and both behave like exponentials.

Proof: Consider the functions $f_1(z) = z$ and $f_2(z) = log(1+z)$ for $z \in [0,1]$. Then $f_1(0) = f_2(0) = 0$, and:

$$\frac{\partial f_1(z)}{\partial z} \equiv 1$$

$$\frac{1}{2} \leq \frac{\partial f_2(z)}{\partial z} = \frac{1}{1+z} \leq 1$$

Thus we can conclude $0.5f_1(z) \leq f_2(z) \leq f_1(z)$. Now set $z = exp(-yf)$ and we get the desired result. □

For negative margins the behaviors of C_e and C_l are very different, as Friedman, Hastie & Tibshirani (2000) have noted. In particular, C_l is more robust against outliers and misspecified data.

2.2.4 Line-search boosting vs. ϵ-boosting

As mentioned above, AdaBoost determines α_t using a line search. In our notation for algorithm 1 this would be:

$$\alpha_t = \arg\min_{\alpha} \sum_i C(y_i, F_i + \alpha h_{j_t}(\mathbf{x_i}))$$

The alternative approach, suggested by Friedman (Friedman 2001, Hastie, Tibshirani & Friedman 2001), is to "shrink" all α_t to a single small value ϵ. This may slow down learning considerably (depending on how small ϵ is), but is attractive theoretically:

the first-order theory underlying gradient boosting implies that the weak learner chosen is the best increment only "locally". It can also be argued that this approach is "stronger" than line search, as we can keep selecting the same h_{j_t} repeatedly if it remains optimal and so ϵ-boosting "dominates" line-search boosting in terms of training error. In practice, this approach of "slowing the learning rate" usually performs better than line-search in terms of prediction error as well (see Friedman (2001)). For our purposes, we will mostly assume ϵ is infinitesimally small, so the theoretical boosting algorithm which results is the "limit" of a series of boosting algorithms with shrinking ϵ.

In regression terminology, the line-search version is equivalent to forward stagewise modeling, infamous in the statistics literature for being too greedy and highly unstable (See Friedman (2001)). This is intuitively obvious, since by increasing the coefficient until it "saturates" we are destroying "signal" which may help us select other good predictors.

2.3 l_p margins, support vector machines and boosting

We now introduce the concept of margins as a geometric interpretation of a binary classification model. In the context of boosting, this view offers a different understanding of AdaBoost from the gradient descent view presented above. In the following sections we connect the two views.

2.3.1 The Euclidean margin and the support vector machine

Consider a classification model in high dimensional predictor space: $F(\mathbf{x}) = \sum_j h_j(\mathbf{x})\beta_j$. We say that the model *separates* the training data $\{\mathbf{x}_i, y_i\}_{i=1}^n$ if $sign(F(\mathbf{x}_i)) = y_i, \ \ \forall i$.

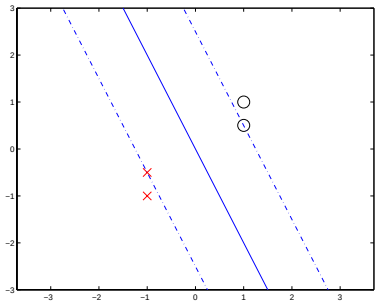


Figure 2.3: A simple data example, with two observations from class "O" and two observations from class "X". The full line is the Euclidean margin-maximizing separating hyper-plane.

From a geometrical perspective this means that the hyper-plane defined by $F(\mathbf{x}) = 0$ is a separating hyper-plane for this data, and we define its (Euclidean) margin as:

$$m_2(\beta) = \min_i \frac{y_i F(\mathbf{x}_i)}{\|\beta\|_2} \tag{2.6}$$

The margin-maximizing separating hyper-plane for this data would be defined by β which maximizes $m_2(\beta)$. Figure 2.3 shows a simple example of separable data in 2 dimensions, with its margin-maximizing separating hyper-plane. The Euclidean margin-maximizing separating hyper-plane is the (non regularized) support vector machine solution. Its margin maximizing properties play a central role in deriving generalization error bounds for these models, and form the basis for a rich literature.

2.3.2 The l_1 margin and its relation to boosting

Instead of considering the Euclidean margin as in (2.6) we can define an "l_p margin" concept as

$$m_p(\beta) = \min_i \frac{y_i F(\mathbf{x}_i)}{\|\beta\|_p} \tag{2.7}$$

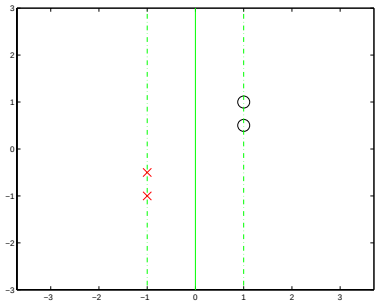


Figure 2.4: l_1 margin maximizing separating hyper-plane for the same data set as figure 2.3. The difference between the diagonal Euclidean optimal separator and the vertical l_1 optimal separator illustrates the "sparsity" effect of optimal l_1 separation

Of particular interest to us is the case $p = 1$. Figure 2.4 shows the l_1 margin maximizing separating hyper-plane for the same simple example as figure 2.3. Note the fundamental difference between the two solutions: the l_2-optimal separator is diagonal, while the l_1-optimal one is vertical. To understand why this is so we can relate the two margin definitions to each other as:

$$\frac{yF(\mathbf{x})}{\|\beta\|_1} = \frac{yF(\mathbf{x})}{\|\beta\|_2} \cdot \frac{\|\beta\|_2}{\|\beta\|_1} \tag{2.8}$$

From this representation we can observe that the l_1 margin will tend to be big if the ratio $\frac{\|\beta\|_2}{\|\beta\|_1}$ is big. This ratio will generally be big if β is sparse. To see this, consider fixing the l_1 norm of the vector and then comparing the l_2 norm of two candidates: one with many small components and the other — a sparse one — with a few large components and many zero components. It is easy to see that the second vector will have bigger l_2 norm, and hence (if the l_2 margin for both vectors is equal) a bigger l_1 margin.

A different perspective on the difference between the optimal solutions is given by a theorem due to Mangasarian (Mangasarian 1999), which states that the l_p margin maximizing separating hyper plane maximizes the l_q distance from the closest points

to the separating hyper-plane, with $\frac{1}{p} + \frac{1}{q} = 1$. Thus the Euclidean optimal separator ($p = 2$) also maximizes Euclidean distance between the points and the hyper-plane, while the l_1 optimal separator maximizes l_∞ distance. This interesting result gives another intuition why l_1 optimal separating hyper-planes tend to be coordinate-oriented (i.e. have sparse representations): since l_∞ projection considers only the largest coordinate distance, some coordinate distances may be 0 at no cost of increased l_∞ distance.

Schapire, Freund, Bartlett & Lee (1998) have pointed out the relation between AdaBoost and the l_1 margin. They prove that, in the case of separable data, the boosting iterations increase the "boosting" margin of the model, defined as:

$$\min_i \frac{y_i F(\mathbf{x})}{\|\alpha\|_1} \tag{2.9}$$

In other words, this is the l_1 margin of the model, except that it uses the α incremental representation rather than the β "geometric" representation for the model. The two representations give the same l_1 norm if there is sign consistency, or "monotonicity" in the coefficient paths traced by the model, i.e. if at every iteration t of the boosting algorithm:

$$\beta_{j_t} \neq 0 \Rightarrow sign(\alpha_t) = sign(\beta_{j_t}) \tag{2.10}$$

As we will see later, this monotonicity condition will play an important role in the equivalence between boosting and l_1 regularization.

The l_1-margin maximization view of AdaBoost presented by Schapire, Freund, Bartlett & Lee (1998) — and a whole plethora of papers that followed — is important for the analysis of boosting algorithms for two distinct reasons:

- It gives an intuitive, geometric interpretation of the model that AdaBoost is

looking for — a model which separates the data well in this l_1-margin sense. Note that the view of boosting as gradient descent in a loss criterion doesn't really give the same kind of intuition: if the data is separable, then any model which separates the training data will drive the exponential or binomial loss to 0 when scaled up:

$$m_1(\beta) > 0 \implies \sum_i C(y_i, d\beta'\mathbf{x}_i) \to 0 \ \ as \ \ d \to \infty$$

- The l_1-margin behavior of a classification model on its training data facilitates generation of generalization (or prediction) error bounds, similar to those that exist for support vector machines.

From a statistical perspective, however, we should be suspicious of margin-maximization as a method for building good prediction models in high dimensional predictor space. Margin maximization is by nature a *non-regularized* objective, and solving it in high dimensional space is likely to lead to over-fitting and bad prediction performance.This has been observed in practice by many authors, in particular Breiman (Breiman 1999). In section 2.7 we return to discuss these issues in more detail.

2.4 Boosting as approximate l_1 constrained fitting

In this section we introduce an interpretation of the generic coordinate-descent boosting algorithm as tracking a path of approximate solutions to l_1-constrained (or equivalently, regularized) versions of its loss criterion. This view serves our understanding of what boosting does, in particular the connection between early stopping in boosting and regularization. We will also use this view to get a result about the asymptotic margin-maximization of regularized classification models, and by analogy of classification boosting. We build on ideas first presented by Hastie, Tibshirani & Friedman

(2001) (chapter 10) and Efron, Hastie, Johnstone & Tibshirani (2002).

Given a loss criterion $C(\cdot,\cdot)$, consider the 1-dimensional path of optimal solutions to l_1 constrained optimization problems over the training data:

$$\hat{\beta}(c) = \arg\min_{\|\beta\|_1 \leq c} \sum_i C(y_i, h(\mathbf{x}_i)'\beta). \tag{2.11}$$

As c varies, we get that $\hat{\beta}(c)$ traces a 1-dimensional "optimal curve" through $\mathcal{R}^J$. If an optimal solution for the non-constrained problem exists and has finite l_1 norm c_0, then obviously $\hat{\beta}(c) = \hat{\beta}(c_0) = \hat{\beta}$, $\forall c > c_0$. Note that in the case of separable 2-class data, using either C_e or C_l, there is no finite-norm optimal solution. Rather, the constrained solution will always have $\|\hat{\beta}(c)\|_1 = c$.

A different way of building a solution which has l_1 norm c, is to run our ϵ-boosting algorithm for c/ϵ iterations. This will give an $\alpha^{(c/\epsilon)}$ vector which has l_1 norm exactly c. For the norm of the geometric representation $\beta^{(c/\epsilon)}$ to also be equal to c, we need the monotonicity condition (2.10) to hold as well. This condition will play a key role in our exposition.

We are going to argue that the two solution paths $\hat{\beta}(c)$ and $\beta^{(c/\epsilon)}$ are very similar for ϵ "small". Let us start by observing this similarity in practice. Figure 2.1 in the introduction shows an example of this similarity for squared error loss fitting with l_1 (lasso) penalty. Figure 2.5 shows another example in the same mold, taken from Efron, Hastie, Johnstone & Tibshirani (2002). The data is a diabetes study and the "dictionary" used is just the original 10 variables. The panel on the left shows the path of optimal l_1-constrained solutions $\hat{\beta}(c)$ and the panel on the right shows the ϵ-boosting path with the 10-dimensional dictionary (the total number of boosting iterations is about 6000). The 1-dimensional path through $\mathcal{R}^{10}$ is described by 10 coordinate curves, corresponding to each one of the variables. The interesting phenomenon we observe is that the two coefficient traces are not completely identical.

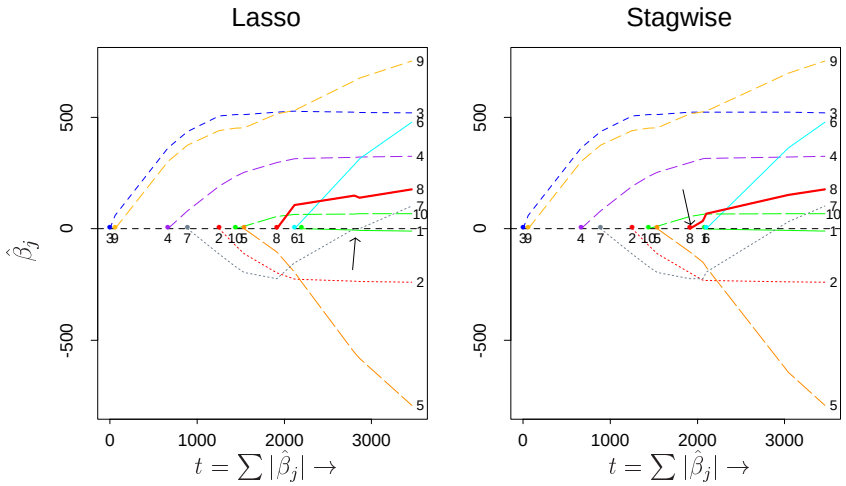


Figure 2.5: Another example of the equivalence between the Lasso optimal solution path (left) and ϵ-boosting with squared error loss. Note that the equivalence breaks down when the path of variable 7 becomes non-monotone

Rather, they agree up to the point where variable 7 coefficient path becomes *non monotone*, i.e. it violates (2.10) (this point is where variable 8 comes into the model, see the arrow on the right panel). This example illustrates that the monotonicity condition — and its implication that $\|\alpha\|_1 = \|\beta\|_1$ — is critical for the equivalence between ϵ-boosting and l_1-constrained optimization.

The two examples we have seen so far have used squared error loss, and we should ask ourselves whether this equivalence stretches beyond this loss. Figure 2.6 shows a similar result, but this time for the binomial log-likelihood loss, C_l. We used the "spam" dataset, taken from the UCI repository (Blake & Merz 1998). We chose only 5 predictors of the 57 to make the plots more interpretable and the computations more accommodating. We see that there is a perfect equivalence between the exact constrained solution (i.e. regularized logistic regression) and ϵ-boosting in this case, since the paths are fully monotone.

To justify why this observed equivalence is not surprising, let us consider the following "l_1-locally optimal monotone direction" problem of finding the best monotone

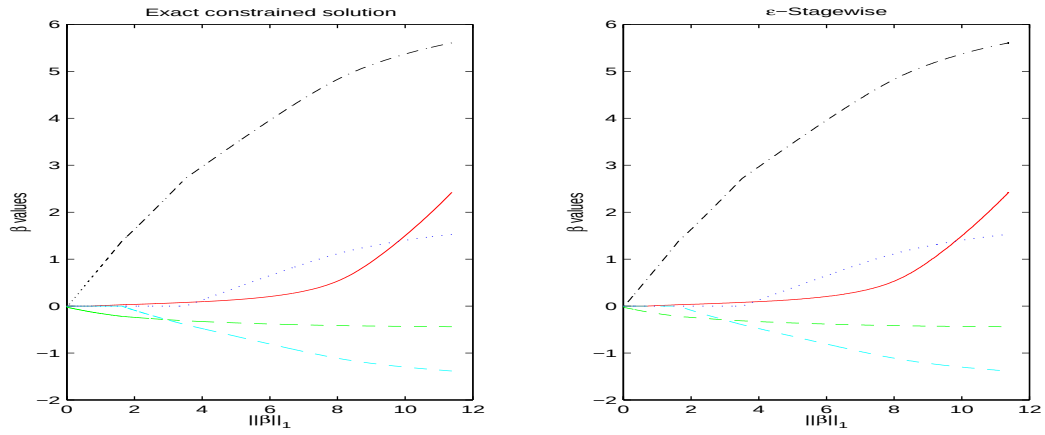


Figure 2.6: Exact coefficient paths (left) for l_1-constrained logistic regression and boosting coefficient paths (right) with binomial log-likelihood loss on five variables from the "spam" dataset. The boosting path was generated using $\epsilon = 0.003$ and 7000 iterations.

ϵ increment to a given model β_0.:

$$\begin{aligned} &\min C(\beta) && (2.12)\\ &s.t. \qquad \|\beta\|_1 - \|\beta_0\|_1 \leq \epsilon \\ &\qquad\quad |\beta| \succeq |\beta_0| \end{aligned}$$

Here we use $C(\beta)$ as shorthand for $\sum_i C(y_i, h(\mathbf{x}_i)^t\beta)$. A first order Taylor expansion gives us:

$$C(\beta) = C(\beta_0) + \nabla C(\beta_0)^t(\beta - \beta_0) + O(\epsilon^2)$$

And given the l_1 constraint on the increase in $\|\beta\|_1$, it is easy to see that a first-order optimal solution (and therefore an optimal solution as $\epsilon \to 0$) will make a "coordinate descent" step, i.e.:

$$\beta_j \neq \beta_{0,j} \quad \Rightarrow \quad \nabla C(\beta_0)_j = \max_k |\nabla C(\beta_0)_k|$$

assuming the signs match (i.e. $sign(\beta_{0j}) = -sign(\nabla C(\beta_0)_j)$).

So we get that if the optimal solution to (2.12) *without* the monotonicity constraint happens to be monotone, then it is equivalent to a coordinate descent step. And so it is reasonable to expect that if the optimal l_1 regularized path is monotone (as it indeed is in figures 2.1,2.6), then an "infinitesimal" ϵ-boosting algorithm would follow the same path of solutions. Furthermore, even if the optimal path is not monotone, we can still use the formulation (2.12) to argue that ϵ-boosting would tend to follow an approximate l_1-regularized path. The main difference between the ϵ-boosting path and the true optimal path is that it will tend to "delay" becoming non-monotone, as we observe for variable 7 in figure 2.5. To understand this specific phenomenon would require analysis of the true optimal path, which falls outside the scope of our discussion — Efron, Hastie, Johnstone & Tibshirani (2002) cover the subject for squared error loss, and their discussion applies to any continuously differentiable convex loss, using second-order approximations.

We can utilize this understanding of the relationship between boosting and l_1-regularization to construct l_p boosting algorithms by changing the coordinate-selection criterion in the coordinate descent algorithm. We will get back to this point in section 2.7, where we design an "l_2 boosting" algorithm.

The experimental evidence and heuristic discussion we have presented lead us to the following conjecture which connects slow boosting and l_1-regularized optimization:

Conjecture 2.2 *Consider applying the ϵ-boosting algorithm to any convex loss function, generating a path of solutions $\beta^{(\epsilon)}(t)$. Then if the optimal coefficient paths are monotone $\forall c < c_0$, i.e. if $\forall j,\ |\hat{\beta}(c)_j|$ is non-decreasing in the range $c < c_0$, then:*

$$\lim_{\epsilon \to 0} \beta^{(\epsilon)}(c_0/\epsilon) = \hat{\beta}(c_0)$$

Efron, Hastie, Johnstone & Tibshirani (2002) (theorem 2) prove a weaker "local"

result for the case of squared error loss only. We generalize their result for any convex loss. However this result still does not prove the "global" convergence which the conjecture claims, and the empirical evidence implies. For the sake of brevity and readability, we defer this proof, together with concise mathematical definition of the different types of convergence, to appendix A.

In the context of "real-life" boosting, where the number of basis functions is usually very large, and making ϵ small enough for the theory to apply would require running the algorithm forever, these results should not be considered directly applicable. Instead, they should be taken as an intuitive indication that boosting — especially the ϵ version — is, indeed, approximating optimal solutions to the constrained problems it encounters along the way.

2.5 l_p-constrained classification loss functions

Having established the relation between boosting and l_1 regularization, we are going to turn our attention to the regularized optimization problem. By analogy, our results will apply to boosting as well. We concentrate on C_e and C_l, the two classification losses defined above, and the solution paths of their l_p constrained versions:

$$\hat{\beta}^{(p)}(c) = \arg\min_{\|\beta\|_p \leq c} \sum_i C(y_i, \beta' h(\mathbf{x_i})) \tag{2.13}$$

where C is either C_e or C_l. As we discussed below equation (2.11), if the training data is separable in $span(\mathcal{H})$, then we have $\|\hat{\beta}^{(p)}(c)\|_p = c$ for *all* values of c. Consequently:

$$\|\frac{\hat{\beta}^{(p)}(c)}{c}\|_p = 1$$

We may ask what are the convergence points of this sequence as $c \to \infty$. The following theorem shows that these convergence points describe "l_p-margin maximizing" separating hyper-planes.

Theorem 2.3 *Assume the data is separable, i.e.* $\exists \beta \; s.t. \forall i, \; y_i \beta' h(\mathbf{x}_i) > 0$.
Then for both C_e *and* C_l, $\frac{\hat{\beta}^{(p)}(c)}{c}$ *converges to the* l_p*-margin-maximizing separating hyper-plane (if it is unique) in the following sense:*

$$\hat{\beta}^{(p)} = \lim_{c \to \infty} \frac{\hat{\beta}^{(p)}(c)}{c} = \arg \max_{\|\beta\|_p = 1} \min_i y_i \beta' h(\mathbf{x}_i) \tag{2.14}$$

If the l_p*-margin-maximizing separating hyper-plane is not unique, then* $\frac{\hat{\beta}(c)}{c}$ *may have multiple convergence points, but they will all represent* l_p*-margin-maximizing separating hyper-planes.*

Proof:
This proof applies for both C_e and C_l, given the property in (2.5). Consider two separating candidates β_1 and β_2 such that $\|\beta_1\|_p = \|\beta_2\|_p = 1$. Assume that β_1 separates better, i.e.:

$$m_1 := \min_i y_i \beta_1' h(\mathbf{x}_i) > m_2 := \min_i y_i \beta_2' h(\mathbf{x}_i) > 0$$

Then we have the following simple lemma:

Lemma 2.4 *There exists some* $D = D(m_1, m_2)$ *such that* $\forall d > D$, $d\beta_1$ *incurs smaller loss than* $d\beta_2$, *in other words:*

$$\sum_i C(y_i, d\beta_1' h(\mathbf{x}_i)) < \sum_i C(y_i, d\beta_2' h(\mathbf{x}_i))$$

Given this lemma, we can now prove that any convergence point of $\frac{\hat{\beta}^{(p)}(c)}{c}$ must be an l_p-margin maximizing separator. Assume β^* is a convergence point of $\frac{\hat{\beta}^{(p)}(c)}{c}$. Denote

its minimal margin on the data by m^*. If the data is separable, clearly $m^* > 0$ (since otherwise the loss of $d\beta^*$ does not even converge to 0 as $d \to \infty$).

Now, assume some $\tilde{\beta}$ with $\|\tilde{\beta}\|_p = 1$ has bigger minimal margin $\tilde{m} > m^*$. By continuity of the minimal margin in β, there exists some open neighborhood of β^*:

$$N_{\beta^*} = \{\beta : \|\beta - \beta^*\|_2 < \delta\}$$

and an $\epsilon > 0$, such that:

$$min_i y_i \beta' h(\mathbf{x_i}) < \tilde{m} - \epsilon, \;\; \forall \beta \in N_{\beta^*}$$

Now by the lemma we get that there exists some $D = D(\tilde{m}, \tilde{m} - \epsilon)$ such that $d\tilde{\beta}$ incurs smaller loss than $d\beta$ for any $d > D, \; \beta \in N_{\beta^*}$. Therefore β^* *cannot be a convergence point of* $\frac{\hat{\beta}^{(p)}(c)}{c}$.

We conclude that any convergence point of the sequence $\frac{\hat{\beta}^{(p)}(c)}{c}$ must be an l_p-margin maximizing separator. If the margin maximizing separator is unique then it is the only possible convergence point, and therefore:

$$\hat{\beta}^{(p)} = \lim_{c \to \infty} \frac{\hat{\beta}^{(p)}(c)}{c} = \arg \max_{\|\beta\|_p = 1} \min_i y_i \beta' h(\mathbf{x}_i)$$

□

Proof of Lemma:

Using (2.5) and the definition of C_e, we get for both loss functions:

$$\sum_i C(y_i, d\beta_1' h(\mathbf{x_i})) \leq n \exp(-d \cdot m_1)$$

Now, since β_1 separates better, we can find our desired

$$D = D(m_1, m_2) = \frac{logn + log2}{m_1 - m_2}$$

such that:

$$\forall d > D, \;\; n \exp(-d \cdot m_1) < 0.5 \exp(-d \cdot m_2)$$

And using (2.5) and the definition of C_e again we can write:

$$0.5 \exp(-d \cdot m_2) \leq \sum_i C(y_i, d\beta_2' h(\mathbf{x_i}))$$

Combining these three inequalities we get our desired result:

$$\forall d > D, \;\; \sum_i C(y_i, d\beta_1' h(\mathbf{x_i})) \leq \sum_i C(y_i, d\beta_2' h(\mathbf{x_i}))$$

□

We thus conclude that if the l_p-margin maximizing separating hyper-plane is unique, the normalized constrained solution converges to it. In the case that the margin maximizing separating hyper-plane is not unique, this theorem can easily be generalized to characterize a unique solution by defining tie-breakers: if the minimal margin is the same, then the second minimal margin determines which model separates better, and so on. Only in the case that the whole order statistics of the l_p margins is common to many solutions can there really be more than one convergence point for $\frac{\hat{\beta}^{(p)}(c)}{c}$.

2.5.1 Implications of theorem 2.3

Boosting implications

Combined with our results from section 2.4, theorem 2.3 indicates that the normalized boosting path $\frac{\beta^{(t)}}{\sum_{u\leq t}\alpha_u}$ — with either C_e or C_l used as loss — "approximately" converges to a separating hyper-plane $\hat{\beta}$, which attains:

$$\max_{\|\beta\|_1=1} \min_i y_i \beta' h(\mathbf{x_i}) = \max_{\|\beta\|_1=1} \min_i y_i d_i \|\beta\|_2, \tag{2.15}$$

where d_i is the Euclidean distance from the training point i to the separating hyper-plane. In other words, it maximizes Euclidean distance scaled by an l_2 norm. As we have mentioned already, this implies that the *asymptotic* boosting solution will tend to be sparse in representation, due to the fact that for fixed l_1 norm, the l_2 norm of vectors that have many 0 entries will generally be larger. We conjecture that this asymptotic solution $\hat{\beta} = \lim_{c\to\infty} \hat{\beta}^{(1)}(c)/c$, will have *at most* n (the number of observations) non-zero coefficients. This in fact holds for squared error loss, where there *always* exists a finite optimal solution $\hat{\beta}^{(1)}$ with at most n non-zero coefficients (see Efron, Hastie, Johnstone & Tibshirani (2002)).

Logistic regression implications

Recall, that the logistic regression (maximum likelihood) solution is undefined if the data is separable in the Euclidean space spanned by the predictors. Theorem 3 allows us to define a logistic regression solution for separable data, as follows:

1. Set a high constraint value c_{max}
2. Find $\hat{\beta}^{(p)}(c_{max})$, the solution to the logistic regression problem subject to the constraint $\|\beta\|_p \leq c_{max}$. The problem is convex for any $p \geq 1$ and differentiable for any $p > 1$, so interior point methods can be used to solve this problem.

3. Now you have (approximately) the l_p-margin maximizing solution for this data, described by

$$\frac{\hat{\beta}^{(p)}(c_{max})}{c_{max}}$$

This is a solution to the original problem in the sense that it is, approximately, the convergence point of the normalized l_p-constrained solutions, as the constraint is relaxed.

Of course, with our result from theorem 3 it would probably make more sense to simply find the optimal separating hyper-plane directly — this is a linear programming problem for l_1 separation and a quadratic programming problem for l_2 separation. We can then consider this optimal separator as a logistic regression solution for the separable data.

2.6 Examples

2.6.1 Spam dataset

We now know if the data are separable and we let boosting run forever, we will approach the same "optimal" separator for both C_e and C_l. However if we stop early — or if the data is not separable — the behavior of the two loss functions may differ significantly, since C_e weighs negative margins exponentially, while C_l is approximately linear in the margin for large negative margins (see Friedman, Hastie & Tibshirani (2000) for detailed discussion). Consequently, we can expect C_e to concentrate more on the "hard" training data, in particular in the non-separable case. Figure 2.7 illustrates the behavior of ϵ-boosting with both loss functions, as well as that of AdaBoost, on the spam dataset (57 predictors, binary response). We used 10 node trees and $\epsilon = 0.1$. The left plot shows the minimal margin as a function of the l_1 norm of the coefficient vector $\|\beta\|_1$. Binomial loss creates a

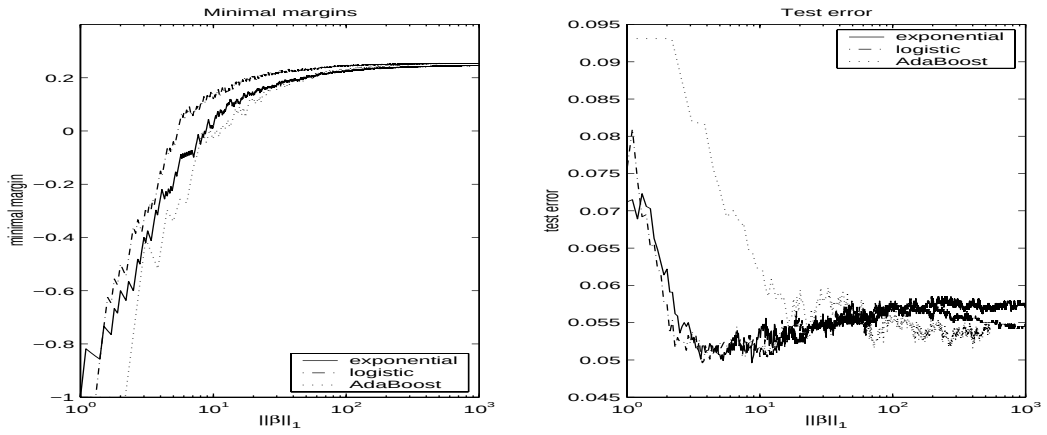


Figure 2.7: Behavior of boosting with the two loss functions on spam dataset

bigger minimal margin initially, but the minimal margins for both loss functions are converging asymptotically. AdaBoost initially lags behind but catches up nicely and reaches the same minimal margin asymptotically. The right plot shows the test error as the iterations proceed, illustrating that both ϵ-methods indeed seem to over-fit eventually, even as their "separation" (minimal margin) is still improving. AdaBoost did not significantly over-fit in the 1000 iterations it was allowed to run, but it obviously would have if it were allowed to run on.

2.6.2 Simulated data

To make a more educated comparison and more compelling visualization, we have constructed an example of separation of 2-dimensional data using a 8-th degree polynomial dictionary (45 functions). The data consists of 50 observations of each class, drawn from a mixture of Gaussians, and presented in figure 2.8. Also presented, in the solid line, is the optimal l_1 separator for this data in this dictionary (easily calculated as a linear programming problem - note the difference from the l_2 optimal decision boundary, presented in section 2.7.2, figure 2.11). The optimal l_1 separator has only 12 non-zero coefficients out of 45.

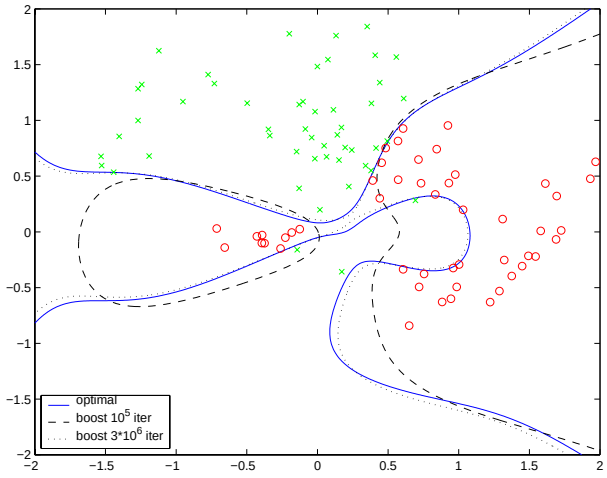


Figure 2.8: Artificial data set with l_1-margin maximizing separator (solid), and boosting models after 10^5 iterations (dashed) and 10^6 iterations (dotted) using $\epsilon = 0.001$. We observe the convergence of the boosting separator to the optimal separator

We ran an ϵ-boosting algorithm on this dataset, using the logistic log-likelihood loss C_l, with $\epsilon = 0.001$, and figure 2.8 shows two of the models generated after 10^5 and $3 \cdot 10^6$ iterations. We see that the models seem to converge to the optimal separator. A different view of this convergence is given in figure 2.9, where we see two measures of convergence — the minimal margin (left) and the l_1-norm distance between the normalized models (right), given by:

$$\sum_i \left| \hat{\beta}_j - \frac{\beta_j^{(t)}}{\|\beta^{(t)}\|_1} \right|$$

where $\hat{\beta}$ is the optimal separator with l_1 norm 1 and $\beta^{(t)}$ is the boosting model after t iterations.

We can conclude that on this simple artificial example we get nice convergence of the logistic-boosting model path to the l_1-margin maximizing separating hyper-plane.

We can also use this example to illustrate the similarity between the boosted path and the path of l_1 optimal solutions, as we have discussed in section 2.4.

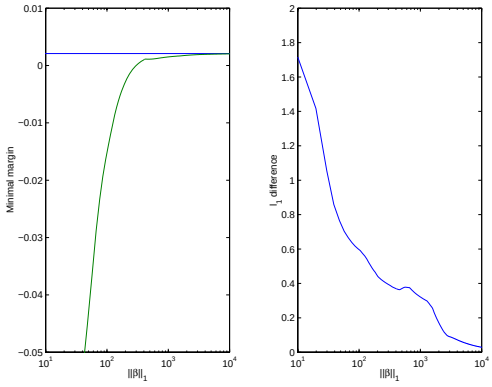


Figure 2.9: Two measures of convergence of boosting model path to optimal l_1 separator: minimal margin (left) and l_1 distance between the normalized boosting coefficient vector and the optimal model (right)

Figure 2.10 shows the class decision boundaries for 4 models generated along the boosting, compared to the optimal solutions to the constrained "logistic regression" problem with the same bound on the l_1 norm of the coefficient vector. We observe the clear similarities in the way the solutions evolve and converge to the optimal l_1 separator. The fact that they differ (in some cases significantly) is not surprising if we recall the monotonicity condition presented in section 2.4 for exact correspondence between the two model paths. In this case if we look at the coefficient paths (not shown), we observe that the monotonicity condition is consistently violated in the low norm ranges, and hence we can expect the paths to be similar in spirit but not identical.

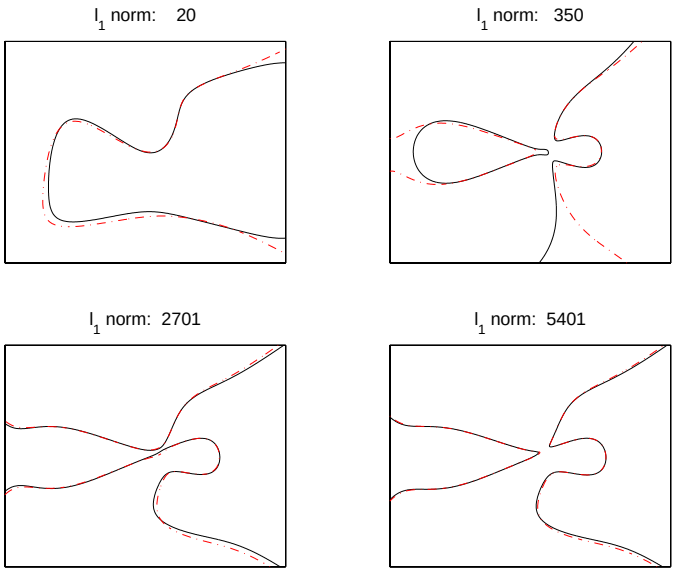


Figure 2.10: Comparison of decision boundary of boosting models (broken) and of optimal constrained solutions with same norm (full)

2.7 Discussion

2.7.1 Regularized and non-regularized behavior of the loss functions

We can now summarize what we have learned about boosting from the previous sections:

- Boosting approximately follows the path of l_1-regularized models for its loss criterion
- If the loss criterion is the exponential loss of AdaBoost or the binomial log-likelihood loss of logistic regression, then the l_1 regularized model converges to an l_1-margin maximizing separating hyper-plane, if the data are separable in the span of the weak learners

We may ask, which of these two points is the key to the success of boosting approaches. One empirical clue to answering this question, can be found in The work of Leo Breiman (Breiman 1999), who programmed an algorithm to *directly* maximize the margins. His results were that his algorithm consistently got significantly higher minimal margins than AdaBoost on many data sets, but had slightly worse prediction performance. His conclusion was that margin maximization is *not* the key to AdaBoost's success. From a statistical perspective we can embrace this conclusion, as reflecting the notion that non-regularized models in high-dimensional predictor space are bound to be over-fitted. "Margin maximization" is a non-regularized objective, both intuitively and more rigorously by our results from the previous section. Thus we would expect the margin maximizing solutions to perform *worse* than regularized models — in the case of boosting regularization would correspond to "early stopping" of the boosting algorithm.

2.7.2 Boosting and SVMs as regularized optimization in high-dimensional predictor spaces

Our exposition has led us to view boosting as an approximate way to solve the regularized optimization problem:

$$\min_{\beta} \sum_{i} C(y_i, \beta' h(\mathbf{x}_i)) + \lambda \|\beta\|_1 \tag{2.16}$$

which converges as $\lambda \to 0$ to $\hat{\beta}^{(1)}$, if our loss is C_e or C_l. In general, the loss C can be any convex differentiable loss and should be defined to match the problem domain.

Support vector machines can be described as solving the regularized optimization

problem (see for example Friedman, Hastie & Tibshirani (2000), chapter 12):

$$\min_{\beta} \sum_{i} (1 - y_i \beta' h(\mathbf{x}_i))_+ + \lambda \|\beta\|_2^2 \tag{2.17}$$

which "converges" as $\lambda \to 0$ to the non-regularized support vector machine solution, i.e. the optimal Euclidean separator, which we denoted by $\hat{\beta}^{(2)}$.

An interesting connection exists between these two approaches, in that they allow us to solve the regularized optimization problem in high dimensional predictor space:

- We are able to solve the l_1- regularized problem approximately in very high dimension via boosting by applying the "approximate coordinate descent" trick of building a decision tree (or otherwise greedily selecting a weak learner) based on re-weighted versions of the data.

- Support vector machines facilitate a different trick for solving the regularized optimization problem in high dimensional predictor space: the "kernel trick". If our dictionary $\mathcal{H}$ spans a Reproducing Kernel Hilbert Space, then RKHS theory tells us we can find the regularized solutions by solving an n-dimensional problem, in the space spanned by the kernel representers $\{K(\mathbf{x}_i, \mathbf{x})\}$. This fact is by no means limited to the hinge loss of (2.17), and applies to any convex loss. We concentrate our discussion on SVM (and hence hinge loss) only since it is by far the most common and well-known application of this result.

So we can view both boosting and SVM as methods that allow us to fit regularized models in high dimensional predictor space using a computational "shortcut". The complexity of the model built is controlled by the regularization. In that they are distinctly different than traditional statistical approaches for building models in high dimension, which start by reducing the dimensionality of the problem so that standard tools (e.g. Newton's method) can be applied to it, and also to make over-fitting less

of a concern. While the merits of regularization without dimensionality reduction — like Ridge regression or the Lasso — are well documented in statistics, computational issues make it impractical for the size of problems typically solved via boosting or SVM, without computational tricks.

We believe that this difference may be a significant reason for the enduring success of boosting and SVM in data modeling, i.e.:

> working in high dimension and regularizing is statistically preferable to a two-step procedure of first reducing the dimension, then fitting a model in the reduced space.

It is also interesting to consider the differences between the two approaches, in the loss (flexible vs. hinge loss), the penalty (l_1 vs. l_2), and the type of dictionary used (usually trees vs. RKHS). These differences indicate that the two approaches will be useful for different situations. For example, if the true model has a sparse representation in the chosen dictionary, then l_1 regularization may be warranted; if the form of the true model facilitates description of the class probabilities via a logistic-linear model, then the logistic loss C_l is the best loss to use, and so on.

The computational tricks for both SVM and boosting limit the kind of regularization that can be used for fitting in high dimensional space. However, the problems can still be formulated and solved for different regularization approaches, as long as the dimensionality is low enough:

- Support vector machines can be fitted with an l_1 penalty, by solving the *1-norm* version of the SVM problem, equivalent to replacing the l_2 penalty in (2.17) with an l_1 penalty. In fact, the 1-norm SVM is used quite widely, because it is more easily solved in the "linear", non-RKHS, situation (as a linear program, compared to the standard SVM which is a quadratic program) and tends to give sparser solutions in the primal domain.

- Similarly, we describe below an approach for developing a "boosting" algorithm for fitting approximate l_2 regularized models.

Both of these methods are interesting and potentially useful. However they lack what is arguably the most attractive property of the "standard" boosting and SVM algorithms: a computational trick to allow fitting in high dimensions.

An l_2 boosting algorithm

We can use our understanding of the relation of boosting to regularization and theorem 3 to formulate l_p-boosting algorithms, which will approximately follow the path of l_p-regularized solutions and converge to the corresponding l_p-margin maximizing separating hyper-planes. Of particular interest is the l_2 case, since theorem 3 implies that l_2-constrained fitting using C_l or C_e will build a regularized path to the optimal separating hyper-plane in the Euclidean (or SVM) sense.

To construct an l_2 boosting algorithm, consider the "equivalent" optimization problem (2.12), and change the step-size constraint to an l_2 constraint:

$$\|\beta\|_2 - \|\beta_0\|_2 \leq \epsilon$$

It is easy to see that the first order to this solution entails selecting for modification the coordinate which maximizes:

$$\frac{\nabla C(\beta_0)_k}{\beta_{0,k}}$$

and that subject to monotonicity, this will lead to a correspondence to the locally l_2-optimal direction.

Following this intuition, we can construct an l_2 boosting algorithm by changing only step 2(c) of our generic boosting algorithm of section 2 to:

2(c)* Identify j_t which maximizes $\frac{|\sum_i w_i h_{j_t}(\mathbf{x}_i)|}{|\beta_{j_t}|}$

Note that the need to consider the current coefficient (in the denominator) makes the l_2 algorithm appropriate for toy examples only. In situations where the dictionary of weak learner is prohibitively large, we will need to figure out a trick like the one we presented in section 2.1, to allow us to make an approximate search for the optimizer of step 2(c)*.

Another problem in applying this algorithm to large problems is that we never choose the same dictionary function twice, until all have non-0 coefficients. This is due to the use of the l_2 penalty, which implies current coefficient value affects the rate at which the penalty term is increasing. In particular, if $\beta_j = 0$ then increasing it cause the penalty term $\|\beta\|_2$ to increase at rate 0, to first order (which is all the algorithm is considering).

The convergence of our l_2 boosting algorithm on the artificial dataset of section 2.6.2 is illustrated in figure 2.11. We observe that the l_2 boosting models do indeed

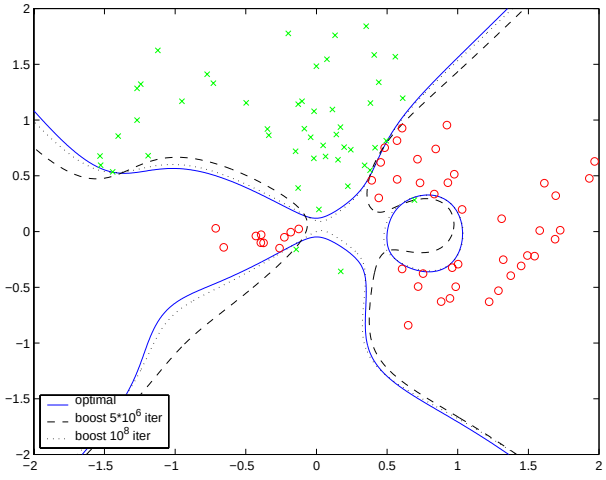


Figure 2.11: Artificial data set with l_2-margin maximizing separator (solid), and l_2-boosting models after $5 * 10^6$ iterations (dashed) and 10^8 iterations (dotted) using $\epsilon = 0.0001$. We observe the convergence of the boosting separator to the optimal separator

approach the optimal l_2 separator. It is interesting to note the significant difference between the optimal l_2 separator as presented in figure 2.11 and the optimal l_1 separator presented in section 2.6.2 (figure 2.8).

2.8 Summary and future work

In this chapter we have introduced a new view of boosting in general, and two-class boosting in particular, comprised of two main points:

- Following Efron, Hastie, Johnstone & Tibshirani (2002), Hastie, Tibshirani & Friedman (2001), we have described boosting as approximate l_1-regularized optimization.
- We have shown that the exact l_1-regularized solutions converge to an l_1-margin maximizing separating hyper-plane.

We hope our results will help in better understanding how and why boosting works. It is an interesting and challenging task to separate the effects of the different components of a boosting algorithm:

- Loss criterion
- Dictionary and greedy learning method
- Line search / slow learning

and relate them to its success in different scenario. The implicit l_1 regularization in boosting may also contribute to its success - it has been shown that in some situations l_1 regularization is inherently superior to others (see Donoho, Johnstone, Kerkyacharian & Picard (1995)).

An important issue when analyzing boosting is over-fitting in the noisy data case. To deal with over-fitting, Rätsch, Onoda & Müller (2001) propose several regularization methods and generalizations of the original AdaBoost algorithm to achieve a *soft margin* by introducing slack variables. Our results indicate that the models along the boosting path can be regarded as l_1 regularized versions of the optimal separator, hence regularization can be done more directly and naturally by stopping the boosting iterations early. It is essentially a choice of the constraint parameter c.

Many other questions arise from our view of boosting. Among the issues to be considered:

- Is there a similar "separator" view of multi-class boosting? In chapter 3 we answer this question positively — we develop and prove multi-class "margin maximizing" versions of logistic regression and SVM.

- Can the constrained optimization view of boosting help in producing generalization error bounds for boosting that would be more tight than the current existing ones?

Chapter 3

Margin maximizing loss functions

3.1 Introduction

Assume we have a classification "learning" sample $\{\mathbf{x}_i, y_i\}_{i=1}^n$ with $y_i \in \{-1, +1\}$. We wish to build a model $F(\mathbf{x})$ for this data by minimizing (exactly or approximately) a loss criterion $\sum_i C(y_i, F(\mathbf{x}_i)) = \sum_i C(y_i F(\mathbf{x}_i))$ which is a function of the *margins* $y_i F(\mathbf{x}_i)$ of this model on this data. Most common classification modeling approaches can be cast in this framework: logistic regression, support vector machines, boosting and more. The model $F(\mathbf{x})$ which these methods actually build is a *linear combination* of *dictionary functions* coming from a dictionary $\mathcal{H}$ which can be large or even infinite:

$$F(\mathbf{x}) = \sum_{h_j \in \mathcal{H}} \beta_j h_j(\mathbf{x})$$

and our prediction at point $\mathbf{x}$ based on this model is $\text{sgn}F(\mathbf{x})$.

When $|\mathcal{H}|$ is large, as is the case in most boosting or kernel SVM applications, some regularization is needed to control the "complexity" of the model $F(\mathbf{x})$ and the resulting over-fitting. Thus, it is common that the quantity actually minimized on

the data is a regularized version of the loss function:

$$\hat{\beta}(\lambda) = \min_{\beta} \sum_{i} C(y_i \beta' h(\mathbf{x}_i)) + \lambda \|\beta\|_p^p \tag{3.1}$$

where the second term penalizes for the l_p norm of the coefficient vector β ($p \geq 1$ for convexity, and in practice usually $p \in \{1, 2\}$), and $\lambda \geq 0$ is a tuning regularization parameter. The 1- and 2-norm support vector machine training problems with slack can be cast in this form (Hastie, Tibshirani & Friedman (2001), chapter 12). In chapter 2 we have shown that boosting approximately follows the "path" of regularized solutions traced by (3.1) as the regularization parameter λ varies, with the appropriate loss and an l_1 penalty.

The main question that we answer in this chapter is: for what loss functions does $\hat{\beta}(\lambda)$ converge to an "optimal" separator as $\lambda \to 0$? The definition of "optimal" which we will use depends on the l_p norm used for regularization, and we will term it the "l_p-margin maximizing separating hyper-plane". More concisely, we will investigate for which loss functions and under which conditions we have:

$$\lim_{\lambda \to 0} \frac{\hat{\beta}(\lambda)}{\|\hat{\beta}(\lambda)\|} = \arg \max_{\|\beta\|_p = 1} \min_{i} y_i \beta' h(\mathbf{x}_i) \tag{3.2}$$

This margin maximizing property is interesting for two distinct reasons. First, it gives us a geometric interpretation of the "limiting" model as we relax the regularization. It tells us that this loss seeks to optimally separate the data by maximizing a distance between a separating hyper-plane and the "closest" points. A theorem by Mangasarian (Mangasarian 1999) allows us to interpret l_p margin maximization as l_q distance maximization, with $1/p + 1/q = 1$, and hence make a clear geometric interpretation. Second, from a learning theory perspective large margins are an important

quantity — generalization error bounds that depend on the margins have been generated for support vector machines (Vapnik (1995) — using l_2 margins) and boosting (Schapire, Freund, Bartlett & Lee (1998) — using l_1 margins). Thus, showing that a loss function is "margin maximizing" in this sense is useful and promising information regarding this loss function's potential for generating good prediction models.

Our main result is a sufficient condition on the loss function, which guarantees that (3.2) holds, *if the data is separable*, i.e. if the maximum on the RHS of (3.2) is positive. This condition is presented and proven in section 3.2. It covers the hinge loss of support vector machines, the logistic log-likelihood loss of logistic regression, and the exponential loss, most notably used in boosting. We discuss these and other examples in section 3.3. Our result generalizes elegantly to multi-class models and loss functions. We present the resulting margin-maximizing versions of SVMs and logistic regression in section 3.4.

3.2 Sufficient condition for margin maximization

The following theorem shows that if the loss function vanishes "quickly" enough, then it will be margin-maximizing as the regularization vanishes. It provides us with a unified margin-maximization theory, covering SVMs, logistic regression and boosting.

Theorem 3.1 *Assume the data* $\{\mathbf{x}_i, y_i\}_{i=1}^n$ *is separable, i.e.* $\exists \beta$ *s.t.* $\min_i y_i \beta' h(\mathbf{x}_i) > 0$.

Let $C(y, f) = C(yf)$ *be a monotone non-increasing loss function depending on the margin only.*

If $\exists T > 0$ *(possibly* $T = \infty$ *) such that:*

$$\lim_{t \nearrow T} \frac{C(t \cdot [1-\epsilon])}{C(t)} = \infty, \ \forall \epsilon > 0 \tag{3.3}$$

Then C *is a margin maximizing loss function in the sense that* any *convergence point of the normalized solutions* $\frac{\hat{\beta}(\lambda)}{\|\hat{\beta}(\lambda)\|_p}$ *to the regularized problems (3.1) as* $\lambda \to 0$ *is an* l_p *margin-maximizing separating hyper-plane. Consequently, if this margin-maximizing hyper-plane is* unique*, then the solutions converge to it:*

$$\lim_{\lambda \to 0} \frac{\hat{\beta}(\lambda)}{\|\hat{\beta}(\lambda)\|_p} = \arg \max_{\|\beta\|_p = 1} \min_i y_i \beta' h(\mathbf{x}_i) \tag{3.4}$$

Proof We prove the result separately for $T = \infty$ and $T < \infty$.

a. $T = \infty$**:**

Lemma 3.2 $\|\hat{\beta}(\lambda)\|_p \xrightarrow{\lambda \to 0} \infty$

Proof Since $T = \infty$ then $C(m) > 0 \ \forall m > 0$, and $\lim_{m \to \infty} C(m) = 0$. Therefore, for loss+penalty to vanish as $\lambda \to 0$, $\|\hat{\beta}(\lambda)\|_p$ must diverge, to allow the margins to diverge.

Lemma 3.3 *Assume* β_1, β_2 *are two separating models, with* $\|\beta_1\|_p = \|\beta_2\|_p = 1$*, and* β_1 *separates the data better, i.e.:* $0 < m_2 = \min_i y_i h(\mathbf{x}_i)' \beta_2 < m_1 = \min_i y_i h(\mathbf{x}_i)' \beta_1$*. Then* $\exists U = U(m_1, m_2)$ *such that*

$$\forall t > U, \ \sum_i C(y_i h(\mathbf{x}_i)'(t\beta_1)) < \sum_i C(y_i h(\mathbf{x}_i)'(t\beta_2))$$

In words, if β_1 *separates better than* β_2 *then scaled-up versions of* β_1 *will incur smaller loss than scaled-up versions of* β_2*, if the scaling factor is large enough.*

Proof Since condition (3.3) holds with $T = \infty$, there exists U such that $\forall t > U$, $\frac{C(tm_2)}{C(tm_1)} > n$. Thus from C being non-increasing we immediately get:

$$\forall t > U,\ \sum_i C(y_i h(\mathbf{x}_i)'(t\beta_1)) \leq n \cdot C(tm_1) < C(tm_2) < \sum_i C(y_i h(\mathbf{x}_i)'(t\beta_2))$$

Proof of case a.: Assume β^* is a convergence point of $\frac{\hat{\beta}(\lambda)}{\|\hat{\beta}(\lambda)\|_p}$ as $\lambda \to 0$, with $\|\beta^*\|_p = 1$. Now assume by contradiction $\tilde{\beta}$ has $\|\tilde{\beta}\|_p = 1$ and bigger minimal l_p margin. Denote the minimal margins for the two models by m^* and $\tilde{m}$, respectively, with $m^* < \tilde{m}$.

By continuity of the minimal margin in β, there exists some open neighborhood of β^* on the l_p sphere:

$$N_{\beta^*} = \{\beta : \|\beta\|_p = 1,\ \|\beta - \beta^*\|_2 < \delta\}$$

and an $\epsilon > 0$, such that:

$$\min_i y_i \beta' h(\mathbf{x_i}) < \tilde{m} - \epsilon,\ \ \forall \beta \in N_{\beta^*}$$

Now by lemma 3.3 we get that exists $U = U(\tilde{m}, \tilde{m} - \epsilon)$ such that $t\tilde{\beta}$ incurs smaller loss than $t\beta$ for any $t > U$, $\beta \in N_{\beta^*}$. Therefore β^* *cannot be a convergence point of* $\frac{\hat{\beta}(\lambda)}{\|\hat{\beta}(\lambda)\|_p}$.

b. $T < \infty$

Lemma 3.4 $C(T) = 0$ *and* $C(T - \delta) > 0,\ \forall \delta > 0$.

Proof From condition (3.3), $\frac{C(T - T\epsilon)}{C(T)} = \infty$. Both results follow immediately, with $\delta = T\epsilon$.

Lemma 3.5 $\lim_{\lambda \to 0} \min_i y_i \hat{\beta}(\lambda)' h(\mathbf{x}_i) = T$

Proof Assume by contradiction that there is a sequence $\lambda_1, \lambda_2, ... \searrow 0$ and $\epsilon > 0$ s.t. $\forall j,\ \min_i y_i \hat{\beta}(\lambda_j)' h(\mathbf{x}_i) \leq T - \epsilon$.

Pick any separating normalized model $\tilde{\beta}$ i.e. $\|\tilde{\beta}\|_p = 1$ and $\tilde{m} := \min_i y_i \tilde{\beta}' h(\mathbf{x}_i) > 0$. Then for *any* $\lambda < \tilde{m}^p \frac{C(T-\epsilon)}{T^p}$ we get:

$$\sum_i C(y_i \frac{T}{\tilde{m}} \tilde{\beta}' h(\mathbf{x}_i)) + \lambda \| \frac{T}{\tilde{m}} \tilde{\beta} \|_p^p < C(T - \epsilon)$$

since the first term (loss) is 0 and the penalty is smaller than $C(T-\epsilon)$ by condition on λ.

But $\exists j_0$ s.t. $\lambda_{j_0} < \tilde{m}^p \frac{C(T-\epsilon)}{T^p}$ and so we get a contradiction to optimality of $\hat{\beta}(\lambda_{j_0})$, since we assumed $\min_i y_i \hat{\beta}(\lambda_{j_0})' h(\mathbf{x}_i) \leq T - \epsilon$ and thus:

$$\sum_i C(y_i \hat{\beta}(\lambda_{j_0})' h(\mathbf{x}_i)) \geq C(T - \epsilon)$$

We have thus proven that $\liminf_{\lambda \to 0} \min_i y_i \hat{\beta}(\lambda)' h(\mathbf{x}_i) \geq T$. It remains to prove equality. Assume by contradiction that for *some* value of λ we have $m := \min_i y_i \hat{\beta}(\lambda)' h(\mathbf{x}_i) > T$. Then the re-scaled model $\frac{T}{m} \hat{\beta}(\lambda)$ has the same zero loss as $\hat{\beta}(\lambda)$, but a smaller penalty, since $\| \frac{T}{m} \hat{\beta}(\lambda) \| = \frac{T}{m} \| \hat{\beta}(\lambda) \| < \| \hat{\beta}(\lambda) \|$. So we get a contradiction to optimality of $\hat{\beta}(\lambda)$.

Proof of case b.: Assume β^* is a convergence point of $\frac{\hat{\beta}(\lambda)}{\|\hat{\beta}(\lambda)\|_p}$ as $\lambda \to 0$, with $\|\beta^*\|_p = 1$. Now assume by contradiction $\tilde{\beta}$ has $\|\tilde{\beta}\|_p = 1$ and bigger minimal margin. Denote the minimal margins for the two models by m^* and $\tilde{m}$, respectively, with $m^* < \tilde{m}$.

Let $\lambda_1, \lambda_2, ... \searrow 0$ be a sequence along which $\frac{\hat{\beta}(\lambda_j)}{\|\hat{\beta}(\lambda_j)\|_p} \to \beta^*$. By lemma 3.5 and our assumption, $\|\hat{\beta}(\lambda_j)\|_p \to \frac{T}{m^*} > \frac{T}{\tilde{m}}$. Thus, $\exists j_0$ such that $\forall j > j_0,\ \|\hat{\beta}(\lambda_j)\|_p > \frac{T}{\tilde{m}}$ and

consequently:

$$\sum_i C(y_i\hat{\beta}(\lambda_j)'h(\mathbf{x}_i)) + \lambda\|\hat{\beta}(\lambda_j)\|_p^p > \lambda(\frac{T}{\tilde{m}})^p = \sum_i C(y_i\frac{T}{\tilde{m}}\tilde{\beta}h(\mathbf{x}_i)) + \lambda\|\frac{T}{\tilde{m}}\tilde{\beta}\|_p^p$$

So we get a contradiction to optimality of $\hat{\beta}(\lambda_j)$.

Thus we conclude for both cases **a.** and **b.** that any convergence point of $\frac{\hat{\beta}(\lambda)}{\|\hat{\beta}(\lambda)\|_p}$ must maximize the l_p margin. Since $\|\frac{\hat{\beta}(\lambda)}{\|\hat{\beta}(\lambda)\|_p}\|_p = 1$, such convergence points obviously exist. If the l_p-margin-maximizing separating hyper-plane is unique, then we can conclude:

$$\frac{\hat{\beta}(\lambda)}{\|\hat{\beta}(\lambda)\|_p} \to \hat{\beta} := \arg\max_{\|\beta\|_p=1} \min_i y_i\beta'h(\mathbf{x}_i)$$

3.2.1 Necessity results

A necessity result for margin maximization *on any separable data* seems to require either additional assumptions on the loss or a relaxation of condition (3.3). We conjecture that if we also require that the loss is convex and vanishing (i.e. $lim_{m\to\infty}C(m) = 0$) then condition (3.3) is sufficient *and necessary*. However this is still a subject for future research.

3.3 Examples

Support vector machines

Support vector machines (linear or kernel) can be described as a regularized problem:

$$\min_\beta \sum_i [1 - y_i\beta'h(\mathbf{x}_i)]_+ + \lambda\|\beta\|_p^p \tag{3.5}$$

where $p = 2$ for the standard ("2-norm") SVM and $p = 1$ for the 1-norm SVM. This formulation is equivalent to the better known "norm minimization" SVM formulation in the sense that they have the same set of solutions as the regularization parameter λ varies in (3.5) or the slack bound varies in the norm minimization formulation.

The loss in (3.5) is termed "hinge loss" since it's linear for margins less than 1, then fixed at 0 (see figure 1). The theorem obviously holds for $T = 1$, and it verifies our knowledge that the non-regularized SVM solution, which is the limit of the regularized solutions, maximizes the appropriate margin (Euclidean for standard SVM, l_1 for 1-norm SVM).

Note that our theorem indicates that the squared hinge loss (AKA truncated squared loss):

$$C(y_i, F(\mathbf{x}_i)) = [1 - y_i F(\mathbf{x}_i)]_+^2$$

is also a margin-maximizing loss.

Logistic regression and boosting

The two loss functions we consider in this context are:

$$Exponential: \qquad C_e(m) = \exp(-m) \tag{3.6}$$

$$Log\ likelihood: \qquad C_l(m) = \log(1 + \exp(-m)) \tag{3.7}$$

These two loss functions are of great interest in the context of two class classification: C_l is used in logistic regression and more recently for boosting (Friedman, Hastie & Tibshirani 2000), while C_e is the implicit loss function used by AdaBoost - the original and most famous boosting algorithm (Freund & Schapire 1995) .

In chapter 2 we showed that boosting approximately follows the regularized path of solutions $\hat{\beta}(\lambda)$ using these loss functions and l_1 regularization. We also proved that the two loss functions are very similar for positive margins, and that their regularized

solutions converge to margin-maximizing separators. Theorem 3.1 provides a new proof of this result, since the theorem's condition holds with $T = \infty$ for both loss functions.

Some interesting non-examples

Commonly used classification loss functions which are not margin-maximizing include *any* polynomial loss function: $C(m) = \frac{1}{m}$, $C(m) = m^2$, etc. do not guarantee convergence of regularized solutions to margin maximizing solutions.

Another interesting method in this context is linear discriminant analysis. Although it does not correspond to the loss+penalty formulation we have described, it does find a "decision hyper-plane" in the predictor space.

For both polynomial loss functions and linear discriminant analysis it is easy to find examples which show that they are not necessarily margin maximizing on separable data.

3.4 A multi-class generalization

Our main result can be elegantly extended to versions of multi-class logistic regression and support vector machines, as follows. Assume the response is now multi-class, with $K \geq 2$ possible values i.e. $y_i \in \{c_1, ..., c_K\}$. Our model consists of a "prediction" for each class:

$$F_k(\mathbf{x}) = \sum_{h_j \in \mathcal{H}} \beta_j^{(k)} h_j(\mathbf{x})$$

with the obvious prediction rule at $\mathbf{x}$ being $\arg\max_k F_k(\mathbf{x})$.

This gives rise to a $K-1$ dimensional "margin" for each observation. For $y = c_k$,

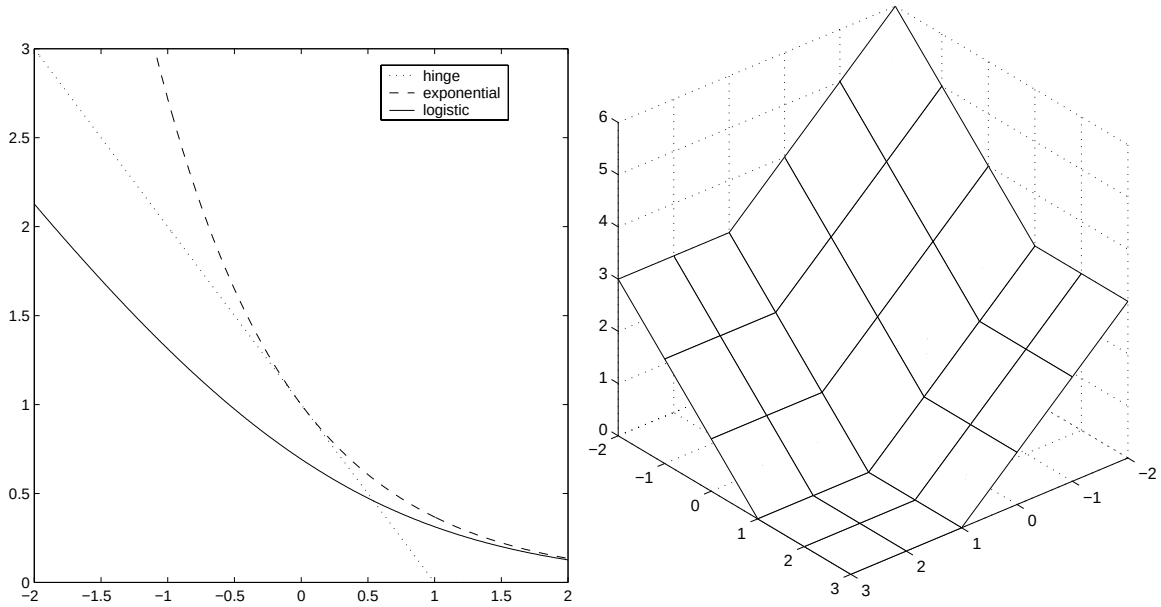


Figure 3.1: Margin maximizing loss functions for 2-class problems (left) and the SVM 3-class loss function of section 3.4.1 (right)

define the margin vector as:

$$\mathbf{m}(c_k, f_1, ..., f_K) = (f_k - f_1, ..., f_k - f_{k-1}, f_k - f_{k+1}, ..., f_k - f_K)' \tag{3.8}$$

And our loss is a function of this $K-1$ dimensional margin:

$$C(y, f_1, ..., f_K) = \sum_k I\{y = c_k\} C(\mathbf{m}(c_k, f_1, ..., f_K))$$

The l_p-regularized problem is now:

$$\hat{\beta}(\lambda) = \arg \min_{\beta^{(1)},...,\beta^{(K)}} \sum_i C(y_i, h(\mathbf{x}_i)'\beta^{(1)}, ..., h(\mathbf{x}_i)'\beta^{(K)}) + \lambda \sum_k \|\beta^{(k)}\|_p^p \tag{3.9}$$

Where $\hat{\beta}(\lambda) = (\hat{\beta}^{(1)}(\lambda), ..., \hat{\beta}^{(K)}(\lambda))' \in \mathcal{R}^{K \cdot |\mathcal{H}|}$.

In this formulation, the concept of margin maximization corresponds to maximizing the minimal of all $n \cdot (K-1)$ normalized l_p-margins generated by the data:

$$\max_{\|\beta^{(1)}\|_p^p + \ldots + \|\beta^{(K)}\|_p^p = 1} \min_i \min_{y_i \neq c_k} h(\mathbf{x}_i)'(\beta^{(y_i)} - \beta^{(k)}) \tag{3.10}$$

Note that this margin maximization problem still has a natural geometric interpretation, as $h(\mathbf{x}_i)'(\beta^{(y_i)} - \beta^{(k)}) > 0 \; \forall i, k \neq y_i$ implies that the hyper-plane $h(\mathbf{x})'(\beta^{(j)} - \beta^{(k)}) = 0$ successfully separates classes j and k for any two classes.

Here is a generalization of the optimal separation theorem 3.1 to multi-class models:

Theorem 3.6 *Assume $C(\mathbf{m})$ is commutative and decreasing in each coordinate, then if $\exists T > 0$ (possibly $T = \infty$) such that:*

$$\begin{aligned} \lim_{t \nearrow T} \quad & \frac{C(t[1-\epsilon], tu_1, \ldots tu_{K-2})}{C(t, tv_1, \ldots, tv_{K-2})} = \infty, \\ & \forall \epsilon > 0, \; u_1 \geq 1, \ldots, u_{K-2} \geq 1, v_1 \geq 1, \ldots v_{K-2} \geq 1 \end{aligned} \tag{3.11}$$

Then C is a margin-maximizing loss function for multi-class models, in the sense that any convergence point of the normalized solutions to (3.9), $\frac{\hat{\beta}(\lambda)}{\|\hat{\beta}(\lambda)\|_p}$, attains the optimal separation as defined in (3.10)

Idea of proof The proof is essentially identical to the two class case, now considering the $n \cdot (K-1)$ margins on which the loss depends. The condition (3.11) implies that as the regularization vanishes the model is determined by the minimal margin, and so an optimal model puts the emphasis on maximizing that margin.

Corollary 3.7 *In the 2-class case, theorem 3.6 reduces to theorem 3.1.*

Proof The loss depends on $\beta^{(1)} - \beta^{(2)}$, the penalty on $\|\beta^{(1)}\|_p^p + \|\beta^{(2)}\|_p^p$. An optimal solution to the regularized problem must thus have $\beta^{(1)} + \beta^{(2)} = 0$, since by

transforming:

$$\beta^{(1)} \rightarrow \beta^{(1)} - \frac{\beta^{(1)} + \beta^{(2)}}{2}\,, \quad \beta^{(2)} \rightarrow \beta^{(2)} - \frac{\beta^{(1)} + \beta^{(2)}}{2}$$

we are not changing the loss, but reducing the penalty, by Jensen's inequality:

$$\|\beta^{(1)} - \frac{\beta^{(1)} + \beta^{(2)}}{2}\|_p^p + \|\beta^{(2)} - \frac{\beta^{(1)} + \beta^{(2)}}{2}\|_p^p = 2\|\frac{\beta^{(1)} - \beta^{(2)}}{2}\|_p^p \leq \|\beta^{(1)}\|_p^p + \|\beta^{(2)}\|_p^p$$

So we can conclude that $\hat{\beta}^{(1)}(\lambda) = -\hat{\beta}^{(2)}(\lambda)$ and consequently that the two margin maximization tasks (3.2), (3.10) are equivalent.

3.4.1 Margin maximization in multi-class SVM and logistic regression

Here we apply theorem 3.6 to versions of multi-class logistic regression and SVM.

For logistic regression, we use a slightly different formulation than the "standard" logistic regression models, which uses class K as a "reference" class, i.e. assumes that $\beta^{(K)} = 0$. This is required for non-regularized fitting, since without it the solution is not uniquely defined. However, using regularization as in (3.9) guarantees that the solution will be unique and consequently we can "symmetrize" the model — which allows us to apply theorem 3.6. So the loss function we use is (assume $y = c_k$ belongs to class k):

$$\begin{aligned} C(y, f_1, ..., f_K) &= -\log \frac{e^{f_k}}{e^{f_1} + ... + e^{f_K}} = \\ &= \log(e^{f_1 - f_k} + ... + e^{f_{k-1} - f_k} + 1 + e^{f_{k+1} - f_k} + ... + e^{f_K - f_k}) \end{aligned} \tag{3.12}$$

with the linear model: $f_j(\mathbf{x}_i) = h(\mathbf{x}_i)'\beta^{(j)}$. It is not difficult to verify that condition (3.11) holds for this loss function with $T = \infty$, using the fact that $log(1 + \epsilon) =$

$\epsilon + O(\epsilon^2)$. The sum of exponentials which results from applying this first-order approximation satisfies (3.11), and as $\epsilon \to 0$, the second order term can be ignored.

For support vector machines, consider a multi-class loss which is a natural generalization of the two-class loss:

$$C(\mathbf{m}) = \sum_{j=1}^{K-1} [1 - m_j]_+ \tag{3.13}$$

Where m_j is the j'th component of the multi-margin $\mathbf{m}$ as in (3.8). Figure 1 shows this loss for $K = 3$ classes as a function of the two margins. The loss+penalty formulation using 3.13 is equivalent to a standard optimization formulation of multi-class SVM (e.g. Weston & Watkins (1998)):

$$\begin{aligned} \max \quad & c \\ \text{s.t.} \quad & h(\mathbf{x}_i)'(\beta^{(y_i)} - \beta^{(k)}) \geq c(1 - \xi_{ik}), \;\; i \in \{1, ...n\}, \; k \in \{1, ..., K\}, \; c_k \neq y_i \\ & \xi_{ik} \geq 0 \; , \quad \sum_{i,k} \xi_{ik} \leq B \; , \quad \sum_k \|\beta^{(k)}\|_p^p = 1 \end{aligned}$$

As both theorem 3.6 (using $T = 1$) and the optimization formulation indicate, the regularized solutions to this problem converge to the l_p margin maximizing multi-class solution.

3.5 Discussion

What are the properties we would like to have in a classification loss function? Recently there has been a lot of interest in Bayes-consistency of loss functions and algorithms (Bartlett, Jordan & McAuliffe (2003) and references therein), as the data size increases. It turns out that practically all "reasonable" loss functions are consistent in that sense, although convergence rates and other measures of "degree of

consistency" may vary.

Margin maximization, on the other hand, is a finite sample optimality property of loss functions, which is potentially of decreasing interest as sample size grows, since the training data-set is less likely to be separable. Note, however, that in very high dimensional predictor spaces, such as those typically used by boosting or kernel SVM, separability of any finite-size data-set is a mild assumption, which is violated only in pathological cases.

We have shown that the margin maximizing property is shared by some popular loss functions used in logistic regression, support vector machines and boosting. Knowing that these algorithms "converge", as regularization vanishes, to the same model (provided they use the same regularization) is an interesting insight. So, for example, we can conclude that 1-norm support vector machines, exponential boosting and l_1-regularized logistic regression all facilitate the *same* non-regularized solution, which is an l_1-margin maximizing separating hyper-plane. From Mangasarian's theorem (Mangasarian 1999) we know that this hyper-plane maximizes the l_∞ distance from the closest points on either side.

The most interesting statistical question which arises is: *are these "optimal" separating models really good for prediction*, or should we expect regularized models to do better in practice? Statistical common sense, as well as the empirical evidence, seem to support the latter, for example see the margin-maximizing experiments by Breiman (Breiman 1999) and Grove and Schuurmans (Grove & Schuurmans 1998). Accepting this view implies that margin-dependent generalization error bounds cannot capture the essential elements of predicting future performance, and that the theoretical efforts should rather be targeted at the loss functions themselves.

Chapter 4

Robust boosting and its relation to bagging

4.1 Introduction

Boosting (Freund & Schapire 1995, Freund & Scahpire n.d.) and Bagging (Breiman 1996) are two approaches to combining "weak" models in order to build prediction models that are significantly better. Much has been written about the empirical success of these approaches in creating prediction models in actual modeling tasks (Breiman 1998, Bauer & Kohavi 1999). The theoretical discussions of these algorithms (Breiman 1998, Friedman, Hastie & Tibshirani 2000, Friedman & Hall 2000, Buhlmann & Yu 2003, Mason, Baxter, Bartlett & Frean 1999) have viewed them from various perspectives. The general theoretical and practical consensus, however, is that the weak learners for boosting should be really weak, while the "weak learners" for bagging should actually be strong. In tree terminology, one should use small trees when boosting and big trees for bagging. In intuitive "bias-variance" terms, we can say that bagging is mainly a variance reduction (or stabilization) operation, while boosting, in the way it flexibly combines models, is also a bias reduction

operation, i.e. it adds flexibility to the representation beyond that of a single learner. In this chapter we present a view of boosting and bagging which allows us to connect them in a natural way, and in fact view bagging as a form of boosting. This view is interesting because it facilitates creation of families of "intermediate" algorithms, which offer a range of "degrees of bias reduction" between boosting and bagging. Our experiments indicate that in many cases these intermediate approaches perform better than either boosting or bagging.

Our exposition concentrates on 2-class classification, this being the most common application for both boosting and bagging, but the results mostly carry through to other learning domains where boosting and bagging have been used, such as multi-class classification, regression (Friedman 2001) and density estimation (Rosset & Segal 2002).

4.2 Boosting, bagging and a connection

A view has emerged in the last few years of boosting as a gradient-based search for a good model in a large implicit predictor space (Mason, Baxter, Bartlett & Frean 1999, Friedman 2001). More specifically, the components of a 2-class classification boosting algorithm are:

- A data sample $\{\mathbf{x}_i, y_i\}_{i=1}^n$, with $\mathbf{x}_i \in \mathcal{R}^p$ and $y_i \in \{-1, +1\}$
- A loss function $C(y, f)$. The two most commonly used are the exponential loss of AdaBoost (Freund & Schapire 1995) and the logistic log-likelihood of LogitBoost (Friedman, Hastie & Tibshirani 2000). We will assume throughout that $C(y, f)$ is a *monotone decreasing and convex* function of the margin $m = yf$, and so we will sometimes write it as $C(m)$.
- A dictionary $\mathcal{H}$ of "weak learners", where each $h \in \mathcal{H}$ is a classification model:

$h : \mathcal{R}^p \to \{-1, +1\}$.

The dictionary most commonly used in classification boosting is classification trees, i.e. $\mathcal{H}$ is the set of all trees with up to k splits.

Given these components, a boosting algorithm incrementally builds a "good" linear combination of the dictionary functions:

$$F(\mathbf{x}) = \sum_{h \in \mathcal{H}} \beta_h h(\mathbf{x})$$

Where "good" is defined as making the empirical loss small:

$$\sum_i C(y_i F(\mathbf{x}_i))$$

The actual incremental algorithm is an exact or approximate coordinate descent algorithm. At iteration t we have the "current" fit F_t, and we look for the weak learner h_t which maximizes the first order decrease in the loss, i.e. h_t maximizes

$$-\sum_i \frac{\partial C(y_i, F(\mathbf{x}_i))}{\partial \beta_h}|_{F=F_t}$$

or equivalently and more clearly it maximizes

$$-\sum_i \frac{\partial C(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}|_{F=F_t} \cdot h(\mathbf{x}_i)$$

which in the case of two-class classification is easily shown to be equivalent to minimizing

$$\sum_i w_i I\{y_i \neq h(\mathbf{x}_i)\} \tag{4.1}$$

Where $w_i = |\frac{\partial C(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}|_{F=F_t}|$. So we seek to find a weak learner which minimizes *weighted* error rate, with the weights being the gradient of the loss. If we use the exponential loss:

$$C(y, f) = \exp(-yf) \tag{4.2}$$

then it can be shown (e.g. Hastie, Tibshirani & Friedman (2001)) that (4.1) is the exact classification task which AdaBoost (Freund & Schapire 1995), the original and most famous boosting algorithm, solves for finding the next weak learner. In their original AdaBoost implementation (Freund & Scahpire n.d.) suggested solving (4.1) on a "new" training data set of size n at each iteration, by sampling from the training dataset "with return" with probabilities proportional to w_i. This facilitates the use of methods for solving non-weighted classification problems for approximately solving (4.1). We will term this approach the "sampling" boosting algorithm. Friedman (2001) has argued that sampling actually improves the performance of boosting algorithms, by adding much needed randomness (his approach is to solve the *weighted* version of (4.1) on a sub-sample, but the basic motivation applies to "sampling boosting" as well). Here is a formal description of a sampling boosting algorithm, given the inputs described above:

Algorithm 4.1 *Generic gradient-based sampling boosting algorithm*

1. *Set* $\beta_0 = 0$.
2. *For* $t = 1 : T$,
   (a) *Let* $F_i = \beta'_{t-1} h(\mathbf{x}_i),\ i = 1, \ldots, n$ *(the current fit).*
   (b) *Set* $w_i = |\frac{\partial C(y_i, F_i)}{\partial F_i}|,\ i = 1, \ldots, n$.
   (c) *Draw a sample* $\{\mathbf{x}_i^*, y_i^*\}_{i=1}^n$ *of size* n *by re-sampling with return from* $\{\mathbf{x}_i, y_i\}_{i=1}^n$ *with probabilities proportional to* w_i

(d) Identify $j_t = \arg\min_j \sum_i I\{y_i^* \neq h_j(\mathbf{x}^*{}_i)\}$.

(e) Set $\beta_{t,j_t} = \beta_{t-1,j_t} + \epsilon$ *and* $\beta_{t,k} = \beta_{t-1,k}, k \neq j_t$.

Comments:

1. Implementation details include the determination of T (or other stopping criterion) and the approach for finding the minimum in step 2(d).

2. We have fixed the step size to ϵ at each iteration (step 2(e)) . While AdaBoost uses a line search to determine step size, it can be argued that a fixed (usually "small") ϵ step is theoretically preferable (see Friedman (2001) and chapter 2 for details).

3. An important issue in designing boosting algorithms is the selection of the loss function $C(\cdot, \cdot)$. Extreme loss functions, such as the exponential loss of AdaBoost, are not robust against outliers and misspecified data, as they assign overwhelming weight to observations which have the smallest margins. Friedman, Hastie & Tibshirani (2000) have thus suggested replacing the exponential loss with the logistic log likelihood loss, but in many practical situations, in particular when the two classes in the training data are separable in $\text{sp}(\mathcal{H})$, this loss can also be non-robust (in chapter 2 we show that for separable data, it resembles the exponential loss as the boosting iterations proceed).

4. The algorithm as described here is not affected by positive affine transformations of the loss function, i.e. running algorithm 4.1 ,using a loss function $C(m)$ is exactly the same as using $C^*(m) = aC(m) + b$ as long as $a > 0$.

Bagging for classification (Breiman 1996) is a different "model combining" approach. It searches, in each iteration, for the member of the dictionary $\mathcal{H}$ which best classifies a *bootstrap sample* of the original training data, and then averages the discovered models to get a final "bagged" model. So, in fact, we can say that:

Proposition 1 *Bagging implements algorithm 1, using a linear loss, $C(y, f) = -yf$ (or any positive affine transformation of it)*

Proof: From the definition of the loss we get:

$$\forall F_t, \ \ \forall i, \ \ |\frac{\partial - yF(\mathbf{x}_i)}{\partial F(\mathbf{x}_i)}|_{F=F_t} \equiv 1$$

So no matter what our "current model" is, all the gradients are always equal, hence the weights will be equal if we apply a "sampling boosting" iteration using the linear loss. In the case that all weights are equal, the boosting sampling procedure described above reduces to bootstrap sampling. Hence the bagging algorithm is a "sampling boosting" algorithm with a linear loss. □

Thus, we can consider Bagging as a boosting algorithm utilizing a very robust (linear) loss. This loss is so robust that it requires no "message passing" between iterations through re-weighting, since the gradient of the loss *does not* depend on the current margins.

4.2.1 Robustness, convexity and boosting

The gradient of a linear loss does not emphasize low-margin observations over high-margin ("well predicted") ones. In fact, it is the most robust convex loss possible, in the following sense:

Proposition 2 *Any loss which is a differentiable, convex and decreasing function of the margin has the property:*

$$m_1 < m_2 \Rightarrow |C'(m_1)| \geq |C'(m_2)|$$

And a linear loss is the only one which attains equality $\forall m_1, m_2$.

Proof: immediate from convexity and monotonicity. □

Mason, Baxter, Bartlett & Frean (1999) have used arguments about generalization error bounds to argue that a good loss for boosting would be even more robust than the linear loss, and consequently non-convex. In particular, they argue that both high-margin and low-margin observations should have low weight, leading to a sigmoid-shaped loss function. Non-convex loss functions present a significant computational challenge, which Mason, Baxter, Bartlett & Frean (1999) have solved for small dictionary examples. Although the idea of such "outlier tolerant" loss functions is appealing, we limit our discussion to convex loss functions, which facilitate the use of standard fitting methodology, in particular boosting.

Our view of bagging as boosting with linear loss allows us to interpret the similarity — and difference — between the algorithms by looking at the loss functions they are "boosting". The linear loss of bagging implies it is not emphasizing the badly predicted observations, but rather treats all data "equally". Thus it is more robust against outliers and more stable, but less "adaptable" to the data than boosting with an exponential or logistic loss.

4.2.2 Intermediate approaches

This view of bagging as a boosting algorithm, opens the door to creating boosting-bagging hybrids, by "robustifying" the loss functions used for boosting. These hybrids may combine the advantages of boosting and bagging to give us new and useful algorithms.

There are two ways to go about creating these intermediate algorithms:

- Define a series of loss functions starting from a "boosting loss" – the exponential or logistic – and converging to the linear loss of bagging.

- Implicitly define the intermediate loss functions by decaying the weights w_i

given by the boosting algorithm using a boosting loss. The loss implied will be the one whose gradient corresponds to the decayed weights.

The two approaches are obviously equivalent through a differentiation or integration operation. We will adopt the "weight decay" approach, but will discuss the loss function implications of the different decay schemes.

4.3 Boosting with weight decay

We would like to change the loss $C(\cdot, \cdot)$ to be more robust, by first decaying the (gradient) weights w_i, then considering the implicit effect of the decay on the loss. In general, we assume that we have a decay function $v(p, w)$ which depends on a decay parameter $p \in [0, 1]$ and the observation weight $w \geq 0$. We require:

- $v(1, w) = w$, i.e. no decay
- $v(0, w) = 1$, i.e. no weighting, which implicitly assumes a linear loss when considered as a boosting algorithm and thus corresponds to Bagging
- Monotonicity: $w_1 < w_2 \rightarrow v(p, w_1) < v(p, w_2) \ \ \forall p$.
- Continuity in both p and w.

And once we specify a decay parameter p, the problem we solve in each boosting iteration, instead of (4.1), is to find the dictionary function h to minimize:

$$\sum_i v(p, w_i) I\{y_i \neq h(\mathbf{x}_i)\} \tag{4.3}$$

which we solve approximately as a non-weighted problem, using the "sampling boosting" approach of algorithm 1.

For clarity and brevity, we concentrate in this chapter on one decay function — the bounding or "Windsorising" operator:

$$v(p,w) = \begin{cases} \frac{1}{1-p} & \text{if } w > \frac{p}{1-p} \\ \frac{w}{p} & \text{otherwise} \end{cases} \tag{4.4}$$

Since w is the absolute gradient of the loss, then bounding w means that we are "huberizing" the loss function, i.e. continuing it linearly beyond some point. The loss function corresponding to the decayed weights is:

$$C^{(p)}(m) = \begin{cases} \frac{C(m)}{p} & \text{if } m > m^*(p) \\ \frac{C(m^*(p))}{p} - \frac{1}{1-p} \cdot (m - m^*(p)) & \text{otherwise} \end{cases} \tag{4.5}$$

where $m^*(p)$ is such that $C'(m^*(p)) = -\frac{p}{1-p}$ (unique because the loss is convex and monotone decreasing).

As we have mentioned, for the purpose of the sampling boosting algorithm, only

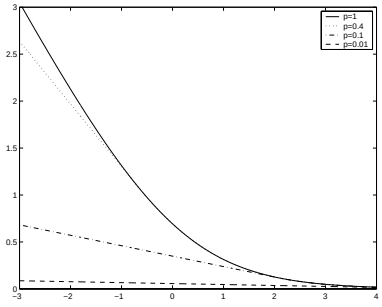


Figure 4.1: "Huberized" logistic loss, for different values of p

the relative sizes of the weights are important, and thus multiplication of the loss function by a constant of $1/p$ — which we have done to achieve the desired properties of $v(p,w)$ — does not affect the algorithm for fixed p. Figure 4.1 illustrates the effect of bounding on the logistic log likelihood loss function, for several values of p (for presentation, this plot uses the non-scaled version, i.e. (4.5) multiplied by p).

Thus, applying this transformation to the original loss function gives a modified loss which is linear for small margins, then as the margin increases starts behaving like the original loss. Thus we can interpret boosting with the huberized loss function as "bagging for a while", until the margins become big enough and reach the non-huberized region. If we boost long enough with this loss, we will have most of the data margins in the non-huberized region, except for outliers, and the resulting weighting scheme will emphasize the "problematic" data points, i.e. those with small margins.

There are many other interesting decay functions, such as the power transformation:

$$v(p, w) = w^p$$

This transformation is attractive because it does not entail the arbitrary "threshold" determination, but rather decays all weights, with the bigger ones decayed more. However, it is less interpretable in terms of its effect on the underlying loss. For the exponential loss $C(m) = \exp(-m)$, the power transformation does not change the loss, since $\exp(-m)^p = \exp(-mp)$. So the effect of this transformation is simply to slow the learning rate (exactly equivalent to decreasing ϵ in algorithm 1).

4.4 Experiments

We now discuss briefly some experiments to examine the usefulness of weight decay and the situations in which it may be beneficial. We use three datasets: the "spam" and "waveform" datasets, available from the UCI repository (Blake & Merz 1998); and the "digits" handwritten digits recognition dataset, discussed in LeCun, Boser, Denker, Henderson, Howard, Hubbard & Jackel (1990). These were chosen since they are reasonably large and represent very different problem domains. Since we have limited the discussion here to 2-class models, we selected only two classes from the multi-class datasets: waveforms 1 and 2 from "waveform" and the digits "2" and "3"

from "digits" (these were selected to make the problem as challenging as possible). The resulting 2-class data-sets are reasonably large — ranging in size between about 2000 and over 5000. In all three cases we used only 25% of the data for training and 75% for evaluation, as our main goal is not to excel on the learning task, but rather to make it difficult and expose the differences between the models which the different algorithms build.

Our experiments consisted of running algorithm 1 using various loss functions, all obtained by decaying the observation weights given by the exponential loss function (4.2). We used "windsorising" decay as in (4.4) and thus the decayed versions correspond to "huberized" versions of (4.2). In all our experiments, bagging performed significantly worse than all versions of boosting, which is consistent with observations made by various researchers, that well-implemented boosting algorithms almost invariably dominate bagging (see for example Breiman's own experiments in Breiman (1998)). It should be clear, however, that this fact does not contradict the view that bagging has some desirable properties, in particular a greater stabilizing effect than boosting.

We present in figure 4.2 the results of running algorithm 1 with $p = 1$ (i.e. using the non-decayed loss function (4.2)), and with $p = 0.0001$ (which corresponds to "huberizing" the loss, as in (4.5), at around $m = 9$) on the three data-sets. We use two settings for the "weak learner": 10-node and 100-node trees. The "learning rate" parameter ϵ is fixed at 0.1. The results in figure 4.2 represent averages over 20 random train-test splits, with estimated 2-standard deviations confidence bounds.

These plots expose a few interesting observations. First, note that weight decay leads to a slower learning rate. From an intuitive perspective this is to be expected, as a more robust loss corresponds to less aggressive learning, by putting less emphasis on the hardest cases.

Second, weight decay is more useful when the "weak learners" are not weak,

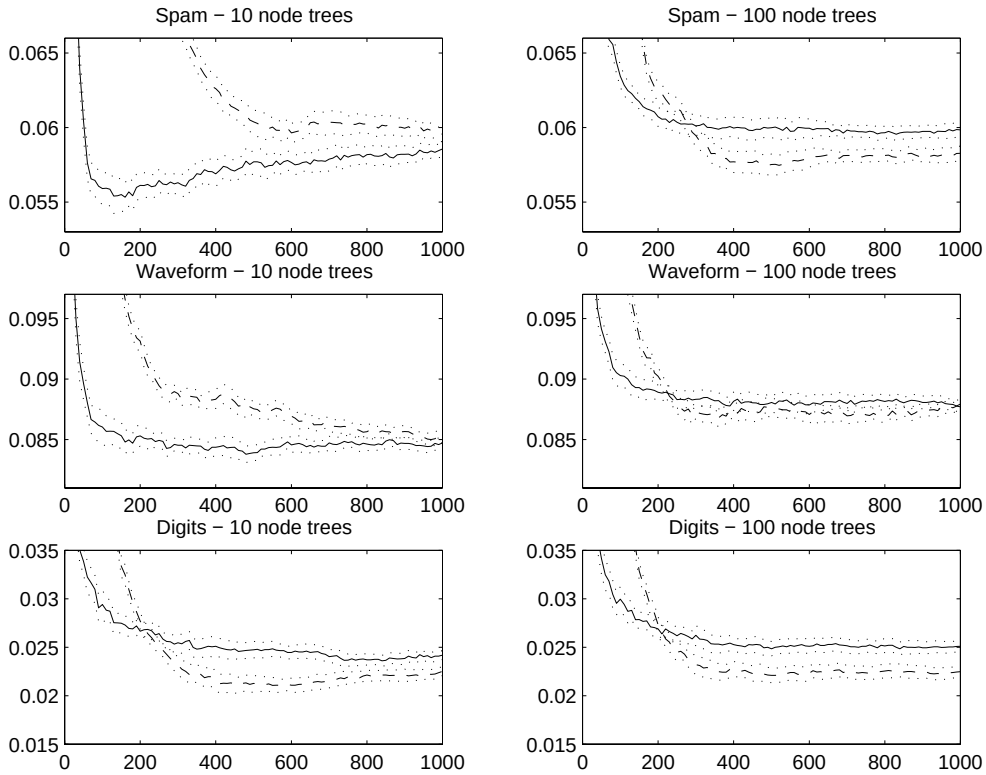


Figure 4.2: Results of running "sampling boosting" with $p = 1$ (solid) and $p = 0.0001$ (dashed) on the three datasets. with 10-node and 100-node trees as the weak learners. The x-axis is the number of iterations, and the y-axis is the mean test-set error over 20 random training-test assignments. The dotted lines are 2-sd confidence bounds.

but rather strong, like a 100-node tree. This is particularly evident in the "spam" dataset, where the performance of the non-decayed exponential boosting deteriorates significantly when we move from 10-node to 100-node learners, while that of the decayed version actually improves significantly. This phenomenon is also as we expect, given the more robust "Huberized" loss implied by the decay.

Third, there is no clear winner between the non-decayed and the decayed versions. For the "Spam" and "Waveform" datasets it seems that if we choose the "best" settings, we would choose the non-decayed loss with small trees. For the "Digits" data, the decayed loss seems to produce consistently better prediction models.

Note that in our examples we did not have big "robustness" issues as they pertain to extreme or highly prevalent outliers in the predictors or the response. Rather, we examine the "bias-variance" tradeoff in employing more "robust" loss functions. Since real-life data usually has "difficult" cases, the sensitivity of the loss and the resulting algorithms to these data can still be considered a "robustness" issue, as we interpret this concept. However, our experiments do not necessarily represent the extreme situations — of many and/or big outliers — where the advantage of using robust loss functions is more pronounced.

4.5 Discussion

The gradient descent view of boosting allows us to design boosting algorithms for a variety of problems, and choose the loss functions which we deem most appropriate. In chapter 2 we show that gradient boosting approximately follows a path of l_1-regularized solutions to the chosen loss function. Thus selection of an appropriate loss is a critical issue in building useful algorithms.

In this chapter I have shown that the gradient boosting paradigm covers bagging as well, and used it as rationale for considering new families of loss functions —

hybrids of standard boosting loss functions and the bagging linear loss function — as candidates for gradient-based boosting. The results seem promising. There are some natural extensions to this concept, which may be even more promising, in particular the idea that the loss function does not have to remain fixed throughout the boosting iterations. Thus, we can design "dynamic" loss functions which change as the boosting iterations proceed, either as a function of the performance of the model or independently of it. It seems that there are a lot of interesting theoretical and practical questions that should come into consideration when we design such an algorithm, such as:

- Should the loss function become more or less robust as the boosting iterations proceed?
- Should the loss function become more or less robust if there are problematic data points?

Chapter 5

Boosting density estimation

5.1 Introduction

Boosting is a method for incrementally building linear combinations of "weak" models, to generate a "strong" predictive model. Given data $\{z_i\}_{i=1}^n$, a basis (or dictionary) of weak learners $\mathcal{H}$ and a loss function L, a boosting algorithm sequentially finds models $h_1, h_2, ... \in \mathcal{H}$ and constants $\alpha_1, \alpha_2, ... \in \mathcal{R}$ to minimize $\sum_i L(z_i, \sum_j \alpha_j h_j(z_i))$. AdaBoost (Freund & Schapire 1995), the original boosting algorithm, was specifically devised for the task of classification, where $z_i = (x_i, y_i)$ with $y_i \in \{-1, 1\}$ and $L = L(y_i, \sum_j \alpha_j h_j(x_i))$. AdaBoost sequentially fits weak learners on re-weighted versions of the data, where the weights are determined according to the performance of the model so far, emphasizing the more "challenging" examples. Its inventors attribute its success to the "boosting" effect which the linear combination of weak learners achieves, when compared to their individual performance. This effect manifests itself both in training data performance, where the boosted model can be shown to converge, under mild conditions, to ideal training classification, and in generalization error, where the success of boosting has been attributed to its "separating" — or margin maximizing — properties (Schapire, Freund, Bartlett & Lee 1998).

It has been shown (Friedman, Hastie & Tibshirani 2000, Mason, Baxter, Bartlett & Frean 1999) that AdaBoost can be described as a gradient descent algorithm, where the weights in each step of the algorithm correspond to the gradient of an exponential loss function at the "current" fit. In chapter 2 we show that the margin maximizing properties of AdaBoost can be derived in this framework as well. This view of boosting as gradient descent has allowed several authors (Friedman 2001, Mason, Baxter, Bartlett & Frean 1999, Zemel & Pitassi 2001) to suggest "gradient boosting machines" which apply to a wider class of supervised learning problems and loss functions than the original AdaBoost. Their results have been very promising.

In this chapter we apply gradient boosting methodology to the unsupervised learning problem of density estimation, using the negative log likelihood loss criterion $L(z, \sum_j \alpha_j h_j(z)) = -\log(\sum_j \alpha_j h_j(z))$. The density estimation problem has been studied extensively in many contexts using various parametric and non-parametric approaches (Bishop 1995, Duda & Hart 1973). A particular framework which has recently gained much popularity is that of Bayesian networks (Heckerman 1998), whose main strength stems from their graphical representation, allowing for highly interpretable models. More recently, researchers have developed methods for learning Bayesian networks from data including learning in the context of incomplete data. We use Bayesian networks as our choice of weak learners, combining the models using the boosting methodology. We note that several researchers have considered learning weighted mixtures of networks (Meila & Jaakkola 2000), or ensembles of Bayesian networks combined by model averaging (Friedman & Koller 2002, Thiesson, Meek & Heckerman 1997).

We describe a generic density estimation boosting algorithm, following the approach of Friedman (2001),Mason, Baxter, Bartlett & Frean (1999). The main idea is to identify, at each boosting iteration, the basis function $h \in \mathcal{H}$ which gives the largest "local" improvement in the loss at the current fit. Intuitively, h assigns higher

probability to instances that received low probability by the current model. A line search is then used to find an appropriate coefficient for the newly selected h function, and it is added to the current model.

We provide a theoretical analysis of our density estimation boosting algorithm, showing an explicit condition, which if satisfied, guarantees that adding a weak learner to the model improves the training set loss. We also prove a "strength of weak learnability" theorem which gives lower bounds on overall training loss improvement as a function of the individual weak learners' performance on re-weighted versions of the training data.

We describe the instantiation of our generic boosting algorithm for the case of using Bayesian networks as our basis of weak learners $\mathcal{H}$ and provide experimental results on two distinct data sets, showing that our algorithm achieves higher generalization on unseen data as compared to a single Bayesian network and one particular ensemble of Bayesian networks. We also show that our theoretical criterion for a weak learner to improve the overall model applies well in practice.

5.2 A density estimation boosting algorithm

At each step t in a boosting algorithm, the model built so far is: $F_{t-1}(z) = \sum_{j<t} \alpha_j h_j(z)$. If we now choose a weak learner $h \in \mathcal{H}$ and add it to our model with a small coefficient ϵ, then developing the training loss of the new model $G = F_{t-1} + \epsilon h$ in a Taylor series around the loss at F_{t-1} gives

$$\sum_i L(z_i, G(z_i)) = \sum_i L(z_i, F_{t-1}(z_i)) + \epsilon \sum_i \frac{\partial L(z_i, F_{t-1}(z_i))}{\partial F_{t-1}(z_i)} h(z_i) + O(\epsilon^2),$$

which in the case of negative log-likelihood loss can be written as

$$\sum_i -log(G(z_i)) = \sum_i -log(F_{t-1}(z_i)) - \epsilon \sum_i \frac{1}{F_{t-1}(z_i)} h(z_i) + O(\epsilon^2).$$

Since ϵ is small, we can ignore the second order term and choose the next boosting step h_t to maximize $\sum_i \frac{1}{F_{t-1}(z_i)} h(z_i)$. We are thus finding the first order optimal weak learner, which gives the "steepest descent" in the loss at the current model predictions. However, we should note that once ϵ becomes non-infinitesimal, no "optimality" property can be claimed for this selected h_t.

The main idea of gradient-based generic boosting algorithms, such as AnyBoost (Mason, Baxter, Bartlett & Frean 1999) and GradientBoost (Friedman 2001), is to utilize this first order approach to find, at each step, the weak learner which gives good improvement in the loss and then follow the "direction" of this weak learner to augment the current model. The step size α_t is determined in various ways in the different algorithms, the most popular choice being line-search, which we adopt here.

When we consider applying this methodology to density estimation, where the basis $\mathcal{H}$ is comprised of probability distributions and the overall model F_t is a probability distribution as well, we cannot simply augment the model, since $F_{t-1} + \alpha_t h_t$ will no longer be a probability distribution. Rather, we consider a step of the form $F_t = (1 - \alpha_t)F_{t-1} + \alpha_t h_t$, where $0 \leq \alpha_t \leq 1$. It is easy to see that the first order theory of gradient boosting and the line search solution apply to this formulation as well.

If at some stage t, the current F_{t-1} cannot be improved by adding any of the weak learners as above, the algorithm terminates, and we have reached a global minimum. This can only happen if the derivative of the loss at the current model with respect

to the coefficient of each weak learner is non-negative:

$$\forall h \in \mathcal{H}, \quad \frac{\partial \sum_i -\log((1-\alpha)F_{t-1}(z_i) + \alpha h(z_i))}{\partial \alpha}|_{\alpha=0} = n - \sum_i \frac{1}{F_{t-1}} h(z_i) \geq 0.$$

Thus, the algorithm terminates if no $h \in \mathcal{H}$ gives $\sum_i \frac{1}{F_{t-1}} h(z_i) > n$ (see section 5.3 for proof and discussion).

The resulting generic gradient boosting algorithm for density estimation can be seen in Fig. 5.1. Implementation details for this algorithm include the choice of the family of weak learners $\mathcal{H}$, and the method for searching for h_t at each boosting iteration. We address these details in Section 5.4.

1. Set $F_0(z)$ to uniform on the domain of z
2. For t = 1 to T
   (a) Set $w_i = 1/F_{t-1}(z_i)$
   (b) Find $h_t \in \mathcal{H}$ to maximize $\sum_i w_i h_t(z_i)$
   (c) If $\sum_i w_i h_t(z_i) \leq n$ break.
   (d) Find $\alpha_t = \arg\min_\alpha \sum_i -\log((1-\alpha)F_{t-1}(z_i) + \alpha h_t(z_i))$
   (e) Set $F_t = (1-\alpha_t)F_{t-1} + \alpha_t h_t$
3. Output the final model F_T

Figure 5.1: Boosting density estimation algorithm

5.3 Training data performance

The concept of "strength of weak learnability" (Freund & Schapire 1995, Schapire, Freund, Bartlett & Lee 1998) has been developed in the context of boosting classification models. Conceptually, this property can be described as follows: "if for any weighting of the training data $\{w_i\}_{i=1}^n$, there is a weak learner $h \in \mathcal{H}$ which achieves weighted training error slightly better than random guessing on the *re*-weighted version of the data using these weights, then the combined boosted learner will have vanishing error on the training data".

In classification, this concept is realized elegantly. At each step in the algorithm, the weighted error of the previous model, using the new weights is exactly 0.5. Thus, the new weak learner doing "better than random" on the re-weighted data means it can improve the previous weak learner's performance at the current fit, by achieving weighted classification error better than 0.5. In fact it is easy to show that the weak learnability condition of at least one weak learner attaining classification error less than 0.5 on the re-weighted data does not hold *only* if the current combined model is the optimal solution in the space of linear combinations of weak learners.

We now derive a similar formulation for our density estimation boosting algorithm. We start with a quantitative description of the performance of the previous weak learner h_{t-1} at the combined model F_{t-1}, given in the following lemma:

Lemma 5.1 *Using the algorithm of section 5.2 we get:* $\forall t$, $\sum_i \frac{h_t(z_i)}{F_t(z_i)} = n$, *where* n *is the number of training examples.*

Proof: The line search (step 2(c) in the algorithm) implies:

$$0 \quad = \quad \frac{\partial \sum_i -\log((1-\alpha)F_{t-1}(z_i) + \alpha h_t(z_i))}{\partial \alpha}|_{\alpha=\alpha_t} = \frac{1}{1-\alpha_t}(n - \sum_i \frac{h_t(z_i)}{F_t(z_i)}).$$

Lemma 5.1 allows us to derive the following stopping criterion (or optimality condition) for the boosting algorithm, illustrating that in order to improve training set loss, the new weak learner only has to exceed the previous one's performance at the current fit.

Theorem 5.2 *If there does not exist a weak learner* $h \in \mathcal{H}$ *such that* $\sum_i \frac{1}{F_t(z_i)} h(z_i) > n$, *then* F_t *is the global minimum in the domain of normalized linear combinations of* $\mathcal{H}$*:* $F_t = \arg\min_\gamma \sum_i -log(\sum_j \gamma_j h_j(z_i))$ *s.t.* $\gamma \geq 0$ *and* $\sum_j \gamma_j = 1$

Proof: This is a direct result of the optimality conditions for a convex function (in this case $-\log$) in a compact domain. □

So unless we have reached the global optimum in the simplex within $span\{\mathcal{H}\}$ (which will generally happen quickly only if $\mathcal{H}$ is very small, i.e. the "weak" learners are very weak), we will have some weak learners doing better than "random" and attaining $\sum_i \frac{h(z_i)}{F_t(z_i)} > n$. If this is indeed the case, we can derive an explicit lower bound for training set loss improvement as a function of the new weak learner's performance at the current model:

Theorem 5.3 *Assume:*

1. *The sequence of selected weak learners in the algorithm of section 5.2 has:*

$$\sum_i \frac{1}{F_{t-1}(z_i)} h_t(z_i) = n + \lambda_t$$

2. $\forall t \;\; min_i(F_{t-1}(z_i), h_t(z_i)) \geq \epsilon_t$

Then we get: $-\sum_i \log(F_t(z_i)) \leq -\sum_i \log(F_{t-1}(z_i)) - \frac{\lambda_t^2 \epsilon_t^2}{2n}$

Proof:

$$\frac{\partial \sum_i -\log((1-\alpha)F_{t-1}(z_i) + \alpha h_t(z_i))}{\partial \alpha}|_{\alpha=0} = n - \sum_i \frac{h_t(z_i)}{F_{t-1}(z_i)} = -\lambda_t$$

$$\frac{\partial^2 \sum_i -\log((1-\alpha)F_{t-1}(z_i) + \alpha h_t(z_i))}{\partial \alpha^2} = \sum_i \frac{[F_{t-1}(z_i) - h_t(z_i)]^2}{[(1-\alpha)F_{t-1}(z_i) + \alpha h_t(z_i)]^2} \leq \frac{n}{\epsilon_t^2}$$

Combining these two gives: $\frac{\partial \sum_i -\log((1-\alpha)F_{t-1}(z_i) + \alpha h_t(z_i))}{\partial \alpha n} \leq -\lambda_t + \frac{\alpha n}{\epsilon_t^2}$, which implies: $\sum_i \log(F_t(z_i)) - \sum_i \log(F_{t-1}(z_i)) \leq -\int_0^{\frac{\lambda_t \epsilon_t^2}{n}} \lambda_t - \frac{xn}{\epsilon_t^2} dx = -\frac{\lambda_t^2 \epsilon_t^2}{n} + \frac{\lambda_t^2 \epsilon_t^2}{2n} = -\frac{\lambda_t^2 \epsilon_t^2}{2n}$ □

The second assumption of theorem 5.3 may not seem obvious but it is actually quite mild. With a bit more notation we could get rid of the need to lower bound h_t completely. For F_t, we can see intuitively that a boosting algorithm will not let any observation have exceptionally low probability over time since that would cause this observation to have overwhelming weight in the next boosting iteration and hence

the next selected h_t is certain to give it high probability. Thus, after some iterations we can assume that we would actually have a threshold ϵ independent of the iteration number and hence the loss would decrease at least as the sum of squares of the "weak learnability" quantities λ_t.

5.4 Boosting Bayesian Networks

We now focus our attention on a specific application of the boosting methodology for density estimation, using Bayesian networks as the weak learners. A Bayesian network is a graphical model for describing a joint distribution over a set of random variables. Recently, there has been much work on developing algorithms for learning Bayesian networks (both network structure and parameters) from data for the task of density estimation and hence they seem appropriate as our choice of weak learners. Another advantage of Bayesian networks in our context, is the ability to tune the strength of the weak learners using parameters such as number of edges and strength of prior.

Assume we have categorical data $\{z_i\}_{i=1}^n$ in a domain $\mathcal{Z}$ where each of the n observations contains assignments to k variables. We rewrite step 2(b) of the boosting algorithm as:

$$2(b) \text{ Find } h_t \in \mathcal{H} \text{ to maximize } \sum_{z \in \mathcal{Z}} v_z h(z), \text{ where } v_z = \sum_{z_i = z} w_i$$

In this formulation, all possible values of z have weights, some of which may be 0.

As mentioned above, the two main implementation-specific details in the generic density estimation algorithm are the set $\mathcal{H}$ of weak models and the method for searching for the "optimal" weak model h_t at each boosting iteration. When boosting Bayesian networks, a natural way of limiting the "strength" of weak learners in $\mathcal{H}$ is to limit the complexity of the network structure in $\mathcal{H}$. This can be done, for instance, by bounding the number of edges in each "weak density estimator" learned during

the boosting iterations.

The problem of finding an "optimal" weak model at each boosting iteration (step 2(b) of the algorithm) is trickier. We first note that if we only impose an L_1 constraint on the norm of h_t (specifically, the PDF constraint $\sum_z h(z) = 1$), then step 2(b) has a trivial solution, concentrating all the probability at the value of z with the highest "weight": $h^*(z) = 1\{z = \arg\max_{u \in \mathcal{Z}} v_u\}$. This phenomenon is *not* limited to the density estimation case and would appear in boosting for classification if the set of weak learners $\mathcal{H}$ had fixed L_1 norm, rather than the fixed L_∞ norm, implicitly imposed by limiting $\mathcal{H}$ to contain classifiers. This consequence of limiting $\mathcal{H}$ to contain probability distributions is particularly problematic when boosting Bayesian networks, since h^* can be represented with a fully disconnected network. Thus, limiting $\mathcal{H}$ to "simple" structures by itself does not amend this problem.

However, the boosting algorithm does not explicitly require $\mathcal{H}$ to include only probability distributions. Let us consider instead a somewhat different family of candidate models, with an implicit L_2 size constraint, rather than L_1 as in the case of probability distributions (note that using an L_∞ constraint as in Adaboost is not possible, since the trivial optimal solution would be $h^* \equiv 1$). For the unconstrained "distribution" case (corresponding to a fully connected Bayesian network), this leads to re-writing step 2(b) of the boosting algorithm as:

$2(b)^1$ Find h to maximize $\sum_{z \in \mathcal{Z}} v_z h(z)$, subject to $\sum_{z \in \mathcal{Z}} h(z)^2 = 1$

By considering the Lagrange multiplier version of this problem it is easy to see that the optimal solution is $\hat{h}_{L_2}(z) = \frac{v_z}{\sqrt{\sum_{u \in \mathcal{Z}} v_u^2}}$ and is proportional to the optimal solution to the log-likelihood maximization problem:

$2(b)^2$ Find h to maximize $\sum_{z \in \mathcal{Z}} v_z \log(h(z))$, subject to $\sum_{z \in \mathcal{Z}} h(z) = 1$

given by $\hat{h}_{log}(z) = \frac{v_z}{\sum_{u \in \mathcal{Z}} v_u}$. This fact points to an interesting correspondence between solutions to L_2-constrained linear optimization problems and L_1-constrained log optimization problems and leads us to believe that good solutions to step $2(b)^1$ of the

boosting algorithm can be approximated by solving step $2(b)^2$ instead.

The formulation given in $2(b)^2$ presents us with a problem that is natural for Bayesian network learning, that of maximizing the log-likelihood (or in this case the weighted log-likelihood $\sum_z v_z \log h(z)$) of the data given the structure.

Our implementation of the boosting algorithm, therefore, does indeed limit $\mathcal{H}$ to include probability distributions only, in this case those that can be represented by "simple" Bayesian networks. It solves a constrained version of step $2(b)^2$ instead of the original version $2(b)$. Note that this use of "surrogate" optimization tasks is not alien to other boosting applications as well. For example, Adaboost calls for optimizing a re-weighted classification problem at each step; Decision trees, the most popular boosting weak learners, search for "optimal" solutions using surrogate loss functions - such as the Gini index for CART (Breiman, Friedman, Olshen & Stone 1984) or information gain for C4.5 (Quinlan 1993).

5.5 Experimental Results

We evaluated the performance of our algorithms on two distinct datasets: a genomic expression dataset and a US census dataset. In gene expression data, the level of mRNA transcript of every gene in the cell is measured simultaneously, using DNA microarray technology, allowing researchers to detect functionally related genes based on the correlation of their expression profiles across the various experiments. We combined three yeast expression data sets (Gasch, Spellman, Kao, O.Carmel-Harel, Eisen, G.Storz, D.Botstein & Brown 2000, Hughes & al. 2000, Spellman & al. 1998) for a total of 550 expression experiments. To test our methods on a set of correlated variables, we selected 56 genes associated with the oxidative phosphorlylation pathway in the KEGG database (*KEGG: Kyoto Encyclopedia of Genes and Genomes* 1998). We discretized the expression measurements of

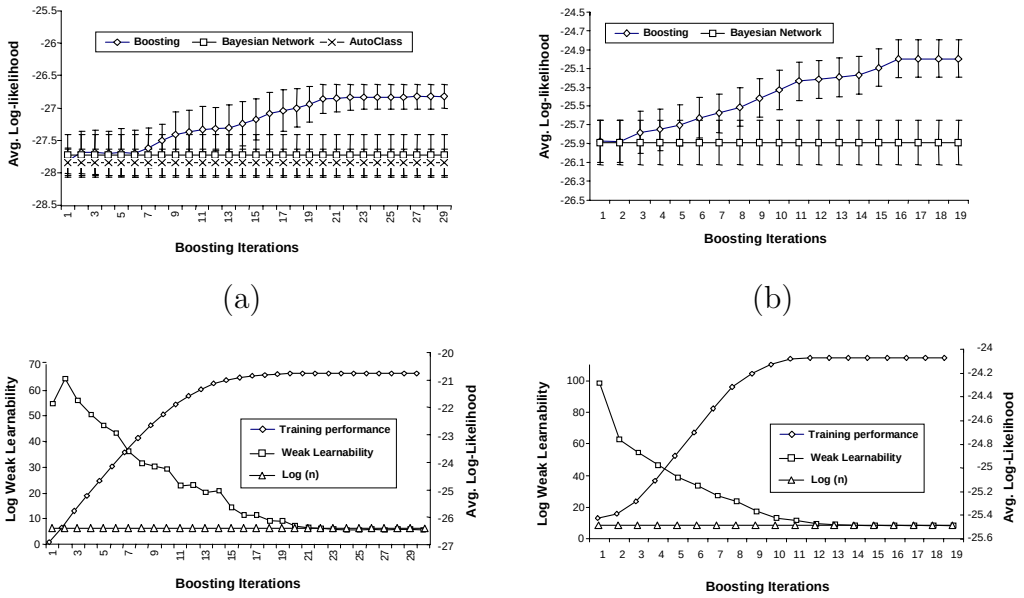


Figure 5.2: (a) Comparison of boosting, single Bayesian network and AutoClass performance on the genomic expression dataset. The average log-likelihood for each test set instance is plotted. (b) Same as (a) for the census dataset. Results for AutoClass were omitted as they were not competitive in this domain (see text). (c) The weak learnability condition is plotted along with training data performance for the genomic expression dataset. The plot is in log-scale and also includes $\log(N)$ as a reference where N is the number of training instances (d) Same as (c) for the census dataset.

each gene into three levels (down, same, up) as in Pe'er, Regev, Elidan & Friedman (2001). We obtained the 1990 US census data set from the UC Irvine data repository (http://kdd.ics.uci.edu/databases/census1990/USCensus1990.html). The data set includes 68 discretized attributes such as age, income, occupation, work status, etc. We randomly selected 5k entries from the 2.5M available entries in the entire data set.

Each of the data sets was randomly partitioned into 5 equally sized sets and our boosting algorithm was learned from each of the 5 possible combinations of 4 partitions. The performance of each boosting model was evaluated by measuring the log-likelihood achieved on the data instances in the left out partition. We compared the performance achieved to that of a single Bayesian network learned using standard

techniques (see Heckerman (1998) and references therein). To test whether our boosting approach gains its performance primarily by using an ensemble of Bayesian networks, we also compared the performance to that achieved by an ensemble of Bayesian networks learned using AutoClass (Cheeseman & Stutz 1995), varying the number of classes from 2 to 100. We report results for the setting of AutoClass achieving the best performance. The results are reported as the average log-likelihood measured for each instance in the test data and summarized in Fig. 5.2(a,b). We omit the results of AutoClass for the census data as it was not comparable to boosting and a single Bayesian network, achieving an average test instance log-likelihood of -31.49 ± 0.28. As can be seen, our boosting algorithm performs significantly better, rendering each instance in the test data roughly 3 and 2.8 times more likely than it is using other approaches in the genomic and census datasets, respectively.

To illustrate the theoretical concepts discussed in Section 5.3, we recorded the performance of our boosting algorithm on the training set for both data sets. As shown in Section 5.3, if $\sum_i \frac{1}{F_t(z_i)} h(z_i) > n$, then adding h to the model is guaranteed to improve our training set performance. Theorem 2 relates the magnitude of this difference to the amount of improvement in training set performance. Fig. 5.2(c,d) plots the weak learnability quantity $\sum_i \frac{1}{F_t(z_i)} h(z_i)$, the training set log-likelihood and the threshold n for both data sets on a log scale. As can be seen, the theory matches nicely, as the improvement is large when the weak learnability condition is large and stops entirely once it asymptotes to n.

Finally, boosting theory tells us that the effect of boosting is more pronounced for "weaker" weak learners. To that extent, we experimented (data not shown) with various strength parameters for the family of weak learners $\mathcal{H}$ (number of allowed edges in each Bayesian network, strength of prior). As expected, the overall effect of boosting was much stronger for weaker learners.

5.6 Discussion and future work

In this chapter we extended the boosting methodology to the domain of density estimation and demonstrated its practical performance on real world datasets. We believe that this direction shows promise and hope that our work will lead to other boosting implementations in density estimation as well as other function estimation domains.

Our theoretical results include an exposition of the training data performance of the generic algorithm, proving analogous results to those in the case of boosting for classification. Of particular interest is theorem 5.2, implying that the idealized algorithm converges, asymptotically, to the global minimum. This result is interesting, as it implies that the greedy boosting algorithm converges to the exhaustive solution. However, this global minimum is usually *not* a good solution in terms of test-set performance as it will tend to overfit (especially if $\mathcal{H}$ is not very small). Boosting can be described as generating a regularized path to this optimal solution — see chapter 2 — and thus we can assume that points along the path will usually have better generalization performance than the non-regularized optimum.

In Section 5.4 we described the theoretical and practical difficulties in solving the optimization step of the boosting iterations (step 2(b)). We suggested replacing it with a more easily solvable log-optimization problem, a replacement that can be partly justified by theoretical arguments. However, it will be interesting to formulate other cases where the original problem has non-trivial solutions - for instance, by not limiting $\mathcal{H}$ to probability distributions only and using non-density estimation algorithms to generate the "weak" models h_t.

The popularity of Bayesian networks as density estimators stems from their intuitive interpretation as describing causal relations in data. However, when learning the network structure from data, a major issue is assigning confidence to the learned

features. A potential use of boosting could be in improving interpretability and reducing instability in structure learning. If the weak models in $\mathcal{H}$ are limited to a small number of edges, we can collect and interpret the "total influence" of edges in the combined model. This seems like a promising avenue for future research, which we intend to pursue.

Chapter 6

Adaptable, Efficient and Robust Methods for Regression and Classification Via Piecewise Linear Regularized Coefficient Paths

We consider the generic regularized optimization problem $\hat{\beta}(\lambda) = \arg\min_\beta L(\mathsf{y}, X\beta) + \lambda J(\beta)$. Recently Efron, Hastie, Johnstone & Tibshirani (2002) have shown that for the Lasso — i.e. if L is squared error loss and $J(\beta) = \|\beta\|_1$ is the l_1 norm of β — the optimal coefficient path is piecewise linear, i.e. $\nabla\hat{\beta}(\lambda)$ is piecewise constant. We derive a general characterization of the properties of (loss L, penalty J) pairs which give piecewise linear coefficient paths. Such pairs allow for efficient generation of the full regularized coefficient paths. We analyze in detail the solution for l_1-penalized Huber loss for regression, and l_1-penalized truncated Huber loss for classifications, and illustrate how we can use our results to generate robust, efficient and adaptable modeling tools.

6.1 Introduction

Regularization is an essential component in modern data analysis, in particular when the number of predictors is large, possibly larger than the number of observations, and non-regularized fitting is guaranteed to give badly over-fitted and useless models.

In this chapter we consider the generic regularized optimization problem. The inputs we have are:

- A training data sample $X = (x_1, ..., x_n)^t$, $\mathsf{y} = (y_1, ..., y_n)^t$, $x_i \in \mathcal{R}^p$ and $y_i \in \mathcal{R}$ for regression, $y_i \in \{\pm 1\}$ for 2-class classification.
- A convex non-negative loss functional $L : \mathcal{R}^n \times \mathcal{R}^n \to \mathcal{R}$
- A convex non-negative penalty functional $J : \mathcal{R}^p \to \mathcal{R}$, with $J(0) = 0$. We will almost exclusively use $J(\beta) = \|\beta\|_q^q$ in this chapter, i.e. penalizing the l_q norm of the coefficient vector.

We want to find:

$$\hat{\beta}(\lambda) = \arg\min_{\beta \in \mathcal{R}^p} L(\mathsf{y}, X\beta) + \lambda J(\beta) \tag{6.1}$$

Where $\lambda \geq 0$ is the regularization parameter: $\lambda = 0$ corresponds to no regularization (hence $\hat{\beta}(0) = \arg\min_\beta L(\mathsf{y}, X\beta)$), while $lim_{\lambda\to\infty}\hat{\beta}(\lambda) = \mathsf{0}$. There is not much theory on choosing the "right" value of λ and the most common approach is to solve (6.1) for a "representative" set of λ values and choose among these models, using cross validation, generalized cross validation or similar methods.

We concentrate our attention on (loss L, penalty J) pairings where the optimal path $\hat{\beta}(\lambda)$ is *piecewise linear* as a function of λ, i.e. $\exists \lambda_0 = 0 < \lambda_1 < \ldots < \lambda_m = \infty$ and $\gamma_0, \gamma_1, \ldots, \gamma_{m-1} \in \mathcal{R}^p$ such that $\hat{\beta}(\lambda) = \hat{\beta}(\lambda_k) + (\lambda - \lambda_k)\gamma_k$ where k is such that $\lambda_k \leq \lambda \leq \lambda_{k+1}$. Such models are attractive because they allow us to generate

the whole regularized path $\hat{\beta}(\lambda),\ 0 \leq \lambda \leq \infty$ simply by calculating the "directions" $\gamma_1, \ldots, \gamma_{m-1}$. Our discussion will concentrate on $(L,\ J)$ pairs which allow efficient generation of the whole path and give useful models.

A canonical example is the lasso Tibshirani (1996):

$$L(\mathsf{y}, \hat{\mathsf{y}}) = \|\mathsf{y} - \hat{\mathsf{y}}\|_2^2 \tag{6.2}$$

$$J(\beta) = \|\beta\|_1 = \sum_j |\beta_j| \tag{6.3}$$

Recently Efron, Hastie, Johnstone & Tibshirani (2002) have shown that the piecewise linear coefficient paths property holds for the lasso. Their results show that the number of linear pieces in the lasso path is approximately the number of the variables in X, and the complexity of generating the whole coefficient path, for all values of λ, is approximately equal to one least square calculation on the full sample.

A simple example to illustrate the piecewise linear property can be seen in figure 6.1, where we show the lasso optimal solution paths for a simple 4-variable synthetic dataset. The plot shows the optimal lasso solutions $\hat{\beta}(\lambda)$ as a function of the regularization parameter λ. Each line represents one coefficient and gives its values at the optimal solution for the range of λ values. We observe that between every two "+" signs the lines are straight, i.e. the coefficient paths are piecewise-linear, as a function of λ, and the 1-dimensional curve $\hat{\beta}(\lambda)$ is piecewise linear in $\mathcal{R}^4$.

The structure of this chapter is as follows: In section 6.2 we present a general formulation of *(loss, penalty)* pairs which give piecewise linear coefficient paths. Our main result is that the loss is required to be piecewise quadratic and the penalty is required to be piecewise linear for the coefficient paths to be piecewise linear. We utilize this understanding to define a family of "almost quadratic" loss functions whose lasso-penalized solution paths are piecewise linear, and formulate and prove an algorithm for generating these solution paths.

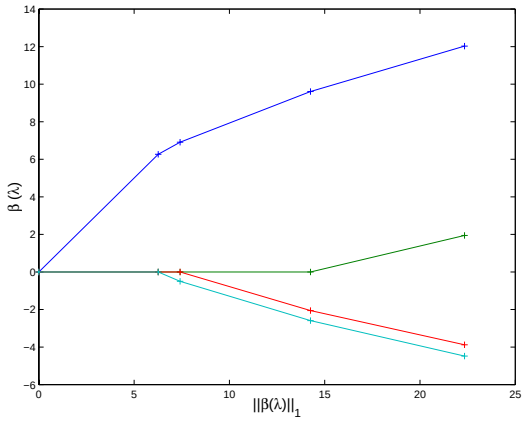


Figure 6.1: Piecewise linear solution paths for the lasso on a simple 4-variable example

We then focus in section 6.3 on some interesting examples which use robust loss functions, yielding modeling tools for both regression and classification which are robust (because of the loss function), adaptable (because we can calculate the whole regularized path and choose a good regularization parameter) and efficient (because we can calculate the path with a relatively small computational burden). In section 6.3.1 we present the l_1 (lasso)-penalized Huber loss for regression, and in section 6.3.2 we discuss the l_1-penalized truncated Huber loss for classification.

In section 6.4 we list further examples of *(loss, penalty)* pairs which give piecewise linear coefficient paths, but do not belong to the family of *(almost quadratic loss, lasso penalty)* problems describes in section 6.3. These include linear loss functions , the l_∞ penalty and multiple-penalty problems. We discuss the practical uses and computational feasibility of calculating the full coefficient paths for these examples.

6.1.1 Illustrative example

Before we delve into the technical details, let us consider an artificial, but realistic, example, which illustrates the importance of both robust loss functions and adaptable regularization.

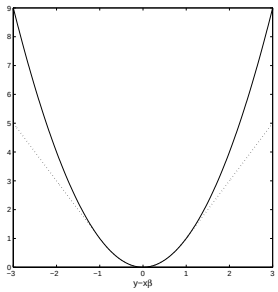


Figure 6.2: Squared error loss and Huber's loss with knot at 1

Our example has $n = 100$ observations and $p = 80$ predictors. All x_{ij} are i.i.d $N(0, 1)$ and the true model is:

$$y_i = 10 \cdot x_{i1} + \epsilon_i \tag{6.4}$$

$$\epsilon_i \overset{iid}{\sim} 0.9 \cdot N(0, 1) + 0.1 \cdot N(0, 100) \tag{6.5}$$

We consider two regularized coefficient paths:

- For the l_1-penalized "Huber's loss" with knot at 1:

$$\hat{\beta}(\lambda) = \arg\min_{\beta} \sum_{|y_i - \beta' x_i| \leq 1} (y_i - \beta' x_i)^2 + 2 \sum_{|y_i - \beta' x_i| > 1} (|y_i - \beta' x_i| - 0.5) + \\ + \lambda \sum_j |\beta_j| \tag{6.6}$$

- For the Lasso ((6.2), (6.3))

Figure 6.3 shows the two regularized paths $\hat{\beta}(\lambda)$ as a function of $\|\beta\|_1$. The "true" coefficient β_1 is the solid line, all other coefficient paths are the dotted lines. For the "Huberized" loss (on the left) we observe that β_1 is the only non-0 coefficient until it reaches its true value of 10. At this point the regularized model is *exactly* the correct model! Only then do any of the other coefficients become non-0, and as $\|\beta\|_1$ increases (or equivalently as the regularization parameter λ decreases) the

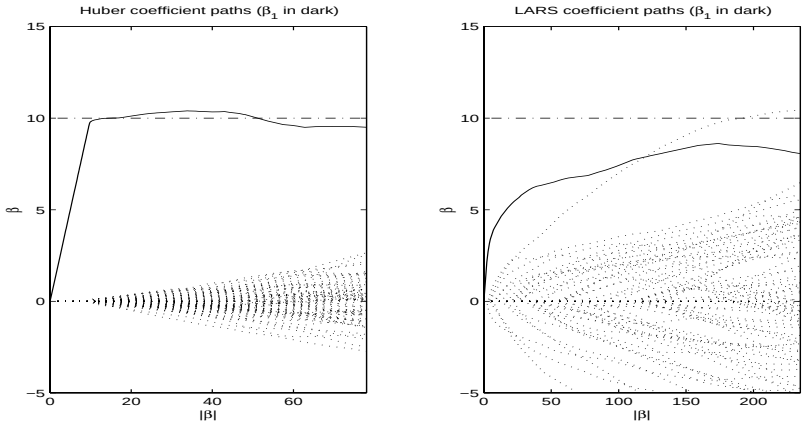


Figure 6.3: Coefficient paths for Huberized lasso(left) and lasso(right) for simulated data example. β_1 is the unbroken line, and the true model is $Ey = 10x_1$

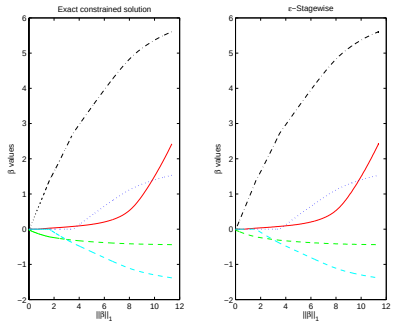


Figure 6.4: Reducible error for the models along the regularized paths

model becomes less and less adequate. The lasso (on the right) does not do as well - the lack of robustness of the loss function (i.e. the implicit normality assumption of squared error loss) causes the regularized path to miss the true model badly at all λ values.

Figure 6.4 shows the reducible "future" squared loss ($E\|\hat{\mathbf{y}} - E\mathbf{y}\|_2^2$) as a function of $\|\beta\|_1$ for the Huberized-lasso and Lasso coefficient paths we got, and it re-iterates our observation that the Huberized version hits *exactly* 0 reducible loss at $\|\beta\|_1 = 10$, and the model then degrades significantly as $\|\beta\|_1$ increases, while the lasso does not do particularly well at any point in the regularized path in terms of reducible loss.

6.2 A general theory and a useful *(loss, penalty)* family

In this section, we first develop a general criterion for piecewise linear solution paths in the case that the loss and penalty are both twice differentiable. This will serve us as an intuitive guide to identify regularized models where we can expect piecewise linearity. It will also prove useful as a tool in asserting piecewise linearity for non-twice differentiable functions. We then build on this result to define a family of "almost quadratic" loss functions whose lasso-regularized solution paths are piecewise linear. We present a general algorithm for generating the solution paths. This family will supply us with modeling tools for regression and classification which we will analyze in detail in the next sections.

6.2.1 Piecewise linear coefficient paths in twice differentiable loss and penalty models

For the coefficient paths to be piecewise linear, we require that $\frac{\partial \hat{\beta}(\lambda)}{\partial \lambda}$ is "piecewise proportional" (i.e. piecewise constant vector, up to multiplication by scalar) as a function of λ, or in other words that over ranges of λ values:

$$\lim_{\epsilon \to 0} \frac{\hat{\beta}(\lambda + \epsilon) - \hat{\beta}(\lambda)}{\epsilon} \propto \text{const vector} \tag{6.7}$$

To get a condition, let us start by considering only β values where L, J are both twice differentiable, with bounded second derivatives in the relevant region. Throughout this section we are going to use the notation $L(\beta)$ in the obvious way, i.e. we make the dependence on the data X, y implicit, since we are dealing with optimization problems in the coefficients β only, and assuming the data is fixed.

We can now write the normal equations for (6.1) at λ and $\lambda + \epsilon$:

$$\nabla L(\hat{\beta}(\lambda)) + \lambda \nabla J(\hat{\beta}(\lambda)) = 0 \tag{6.8}$$

$$\nabla L(\hat{\beta}(\lambda + \epsilon)) + (\lambda + \epsilon)\nabla J(\hat{\beta}(\lambda + \epsilon)) = 0 \tag{6.9}$$

Developing (6.9) in Taylor series around $\nabla L(\hat{\beta}(\lambda))$ and $\nabla J(\hat{\beta}(\lambda))$ we get:

$$\begin{aligned} \nabla L(\hat{\beta}(\lambda)) &+ \nabla^2 L(\hat{\beta}(\lambda))[\hat{\beta}(\lambda + \epsilon) - \hat{\beta}(\lambda)] + \\ &+ \lambda \nabla J(\hat{\beta}(\lambda)) + \lambda \nabla^2 J(\hat{\beta}(\lambda))[\hat{\beta}(\lambda + \epsilon) - \hat{\beta}(\lambda)] + \epsilon \nabla J(\hat{\beta}(\lambda)) + \\ &+ O(\epsilon^2) = 0 \end{aligned} \tag{6.10}$$

From (6.8) and (6.10) we easily get:

$$\frac{\hat{\beta}(\lambda + \epsilon) - \hat{\beta}(\lambda)}{\epsilon} = -[\nabla^2 L(\hat{\beta}(\lambda) + \lambda \nabla^2 J(\hat{\beta}(\lambda))]^{-1} \nabla J(\hat{\beta}(\lambda)) + O(\epsilon) \tag{6.11}$$

And from (6.7) and (6.11) we get that:

Lemma 6.1 *For twice differentiable loss and penalty, a sufficient and necessary condition for piecewise linear coefficient paths is that the direction*

$$-(\nabla^2 L(\hat{\beta}(\lambda) + \lambda \nabla^2 J(\hat{\beta}(\lambda)))^{-1} \nabla J(\hat{\beta}(\lambda)) \tag{6.12}$$

is a piecewise proportional (i.e. constant up to multiplication by scalar) vector in $\mathcal{R}^p$ as a function of λ.

The obvious situation in which lemma 6.1 holds and we get piecewise linear coefficient paths is:

- L is piecewise quadratic as a function of β along the optimal path $\hat{\beta}(\lambda)$, when

X, y are assumed constant at their sample values

- J is piecewise linear as a function of β along this path.

We would generally not expect the lemma to apply in any other situation, since (6.12) will follow a "curved" path. Technically we cannot rule out the existence of (L, J) pairs which would yield piecewise linear coefficient paths for a *specific sample* X, y, as the requirements of the lemma amount to a functional differential equation. However if we require the same (L,J) choice to apply for *all* the sample space and give piecewise linear coefficient paths, it seems that the *piecewise quadratic L - piecewise linear J* is the only feasible solution.

So we can conclude that the search for regularized problems with piecewise linear coefficient paths should concentrate on losses L which are "almost everywhere" quadratic and penalties J which are "almost everywhere" linear. We next define a family of such loss and penalty pairs for which a generic algorithm exists for generating the whole coefficient paths. Members of this family are going to be the focus of our later sections.

6.2.2 "Almost quadratic" loss functions with Lasso penalty

We now concentrate our attention on a family of models which give piecewise linear coefficient paths - a family we consider to be most useful and practical. We fix the penalty to be the l_1 (lasso) penalty:

$$J(\beta) = \|\beta\|_1 = \sum_j |\beta_j| \tag{6.13}$$

And the loss is required to be differentiable and piecewise quadratic in a fixed function of the sample response and the "prediction" $\beta' x$:

$$L(\mathsf{y}, X\beta) = \sum_i l(y_i, \beta' x_i) \tag{6.14}$$

$$l(y_i, \beta' x) = a(r(y, \beta' x))r(y, \beta' x)^2 + b(r(y, \beta' x))r(y, \beta' x) + c(r(y, \beta' x)) \tag{6.15}$$

Where $r(y, \beta' x) = (y - \beta' x)$ is the residual for regression and $r(y, \beta' x) = (y\beta' x)$ is the margin for classification; and $a(\cdot), b(\cdot), c(\cdot)$ are piecewise constant functions, with a finite (usually small) number of pieces, defined so as to make the function l differentiable *everywhere*.

Some examples from this family are:

- The lasso: $l(y, \beta' x) = (y - \beta' x_i)^2$, i.e. $a \equiv 1, b \equiv 0, c \equiv 0$.
- The Huber loss function with *fixed* knot t (define $r = y - \beta' \mathbf{x}$ the residual):

$$l(r) = \begin{cases} r^2 & \text{if } |r| \le t \\ 2t|r| - t^2 & \text{otherwise} \end{cases} \tag{6.16}$$

- Truncated squared error loss for classification (define $r = y\beta' \mathbf{x}$ the margin):

$$l(r) = \begin{cases} (r-1)^2 & \text{if } r \le 1 \\ 0 & \text{otherwise} \end{cases} \tag{6.17}$$

Theorem 6.2 *All regularized problems of the form (6.1) using (6.13), (6.15) (with r being either the residual or the margin) generate piecewise linear optimal coefficient paths $\hat{\beta}(\lambda)$ as the regularization parameter λ varies.*

Proof Since we do not assume twice-differentiability of either the loss or the penalty

(but we do assume differentiability of the loss), we use lemma 6.1 as an intuitive argument only, and prove the theorem formally using the Karush-Kuhn-Tucker (KKT) formulation of the optimization problem implied.

We will code $\beta_j = \beta_j^+ - \beta_j^-$, with $\beta_j^+ \geq 0, \beta_j^- \geq 0$, and start from the constrained version of the regularized optimization problem:

$$\begin{aligned} \min_{\beta^+,\beta^-} \quad & \sum_i l(y_i, (\beta^+ - \beta^-)' x_i) \\ \text{s.t.} \quad & \sum_j \beta_j^+ + \beta_j^- \leq S, \\ & \beta_j^+ \geq 0,\ \beta_j^- \geq 0, \quad \forall j \end{aligned} \tag{6.18}$$

Then we can write the Lagrange dual function of our minimization problem as:

$$\sum_i l(y_i, (\beta^+ - \beta^-)' x_i) + \lambda([\sum_j \beta_j^+ + \beta_j^-] - S) - \sum_j \lambda_j^+ \beta_j^+ - \sum_j \lambda_j^- \beta_j^- \tag{6.19}$$

And the corresponding KKT optimality conditions:

$$(\nabla L(\beta))_j + \lambda - \lambda_j^+ = 0 \tag{6.20}$$

$$-(\nabla L(\beta))_j + \lambda - \lambda_j^- = 0 \tag{6.21}$$

$$\lambda(\sum_j \beta_j^+ + \beta_j^- - S) = 0 \tag{6.22}$$

$$\lambda_j^+ \beta_j^+ = 0 \tag{6.23}$$

$$\lambda_j^- \beta_j^- = 0 \tag{6.24}$$

Using these we can figure that at the optimal solution for fixed S (and hence fixed

λ in (6.1)) the following scenarios could hold:

$$\lambda = 0 \quad \Rightarrow \ (\nabla L(\beta))_j = 0 \ \forall j \text{ (optimal solution)} \tag{6.25}$$

$$\beta_j^+ > 0, \lambda > 0 \quad \Rightarrow \ \lambda_j^+ = 0 \ \Rightarrow \ (\nabla L(\beta))_j = -\lambda < 0 \ \Rightarrow \tag{6.26}$$

$$\Rightarrow \ \lambda_j^- > 0 \ \Rightarrow \ \beta_j^- = 0$$

$$\beta_j^- > 0, \lambda > 0 \quad \Rightarrow \ \lambda_j^- = 0 \ \Rightarrow \ (\nabla L(\beta))_j = \lambda > 0 \ \Rightarrow \tag{6.27}$$

$$\Rightarrow \ \lambda_j^+ > 0 \ \Rightarrow \ \beta_j^+ = 0$$

$$|(\nabla L(\beta))_j| > \lambda \quad \Rightarrow \text{ contradiction} \tag{6.28}$$

Based on these possible scenarios we can see that:

- Variables can have non-0 coefficients at $\hat{\beta}(\lambda)$ only if their "generalized absolute correlation" $|\nabla L(\hat{\beta}(\lambda))_j|$ is equal to λ. Thus, for every value of λ we have a set of "active" variables $\mathcal{A} = \{j \in \{1, ..., m\} : \beta_j \neq 0\}$ such that:

$$j \in \mathcal{A} \quad \Rightarrow \ |\nabla L(\hat{\beta}(\lambda))_j| = \lambda, \ \text{sgn}(\nabla L(\hat{\beta}(\lambda))_j) = -\text{sgn}(\beta_j) \tag{6.29}$$

$$j \notin \mathcal{A} \quad \Rightarrow \ |\nabla L(\hat{\beta}(\lambda))_j| \leq \lambda$$

- The direction in which the fit is moving $\frac{\partial \hat{\beta}(\lambda)}{\partial \lambda}$ should be such that it maintains the conditions $|\nabla L(\hat{\beta}(\lambda))_{\mathcal{A}}| = \lambda, \ |\nabla L(\hat{\beta}(\lambda))_{\mathcal{A}^C}| \leq \lambda$.

So, if we know what the "active" set $\mathcal{A}$ is we can use a similar argument to the one we used for proving lemma 6.1 in (6.8)- (6.12) to argue that, as long as we are in a region where the loss and penalty are both right differentiable, we will have

$$\frac{\partial \hat{\beta}(\lambda)_{\mathcal{A}}}{\partial \lambda} = -(\nabla^2 L(\hat{\beta}(\lambda))_{\mathcal{A}})^{-1} \text{sgn}(\beta_{\mathcal{A}}) \tag{6.30}$$

which is just a version of (6.12), limited to only the active variables and substituting

the lasso penalty for J.

For the family of almost quadratic loss functions we can derive $\nabla^2 L(\hat{\beta}(\lambda))_{\mathcal{A}}$ explicitly, since the second derivative of the constant and linear parts in (6.15) is 0. Thus we easily see:

$$\nabla^2 L(\hat{\beta}(\lambda))_{\mathcal{A}} = \sum_i a(r(y_i, \hat{\beta}(\lambda)'_{\mathcal{A}}\mathbf{x}_{\mathcal{A}i}))\mathbf{x}_{\mathcal{A}i}\mathbf{x}'_{\mathcal{A}i} \tag{6.31}$$

Twice differentiability is violated and hence the direction in (6.30) changes when one of the following events occur:

- A new variable should join $\mathcal{A}$, i.e. we reach a point where $|L(\hat{\beta}(\lambda))_{\mathcal{A}^C}| \leq \lambda$ would cease to hold if we keep moving in the same direction
- A coefficient in $\mathcal{A}$ hits 0. This is possible, just as in the lasso (Efron, Hastie, Johnstone & Tibshirani 2002). In that case, we reach a non-differentiability point in the penalty and we can see that $\text{sgn}(\nabla L(\hat{\beta}(\lambda))_{\mathcal{A}}) = -\text{sgn}(\beta_{\mathcal{A}})$ would cease to hold if we continue in the same direction. Thus we need to drop the coefficient hitting 0 from $\mathcal{A}$.
- A "generalized residual" $r(y_i\hat{\beta}(\lambda)'\mathbf{x}_i)$ hits a non-twice differentiability point (a "knot") in L, for example the "huberizing" point t in (6.16), or the truncation point in (6.17).

And so we conclude that the path $\hat{\beta}(\lambda)$ will be piecewise linear, with the direction given by (6.30) and direction changes occurring whenever one of the three events above happens. When it happens, we need to update $\mathcal{A}$ to get a legal scenario and re-calculate the direction using (6.30). □

Based on this theorem and the arguments in its proof we can derive a generic algorithm to generate coefficient paths for all members of the "almost quadratic" family

of loss functions with Lasso penalty. The LARS-Lasso algorithm (Efron, Hastie, Johnstone & Tibshirani 2002) is a simplified version of this algorithm since "knot crossing" events do not occur in the lasso (as the loss is twice differentiable). Our algorithm starts at $\lambda = \infty$ and follows the linear pieces, while identifying the "events" and re-calculating the direction when they occur. Notice that the algorithm does not calculate λ explicitly, since the direction calculations and step-length calculations do not require to track the λ values.

Algorithm 6.1 *An algorithm for "almost quadratic" loss with Lasso penalty*

1. *Initialize:* $\beta = 0,\ \mathcal{A} = \arg\max_j(\nabla L(\beta))_j,\ \gamma_{\mathcal{A}} = -sgn(\nabla L(\beta))_{\mathcal{A}},\ \gamma_{\mathcal{A}^C} = 0$
2. *While* $(max|\nabla L(\beta)| > 0)$
   - *(a)* $d_1 = \min\{d \geq 0 : \min_{j \notin \mathcal{A}} |\nabla L(\beta + d\gamma)_j| = |\nabla L(\beta + d\gamma)_{\mathcal{A}}|\}$
   - *(b)* $d_2 = \min\{d \geq 0 : \min_{j \in \mathcal{A}}(\beta + d\gamma)_j = 0\}$ *(hit 0)*
   - *(c)* $d_3 = \min\{d \geq 0 : \min_i [r(y_i, (\beta + d\gamma)' x_i)$ *hits a "knot"*$]\}$
   - *(d) set* $d = \min(d_1, d_2, d_3)$
   - *(e) If* $d = d_1$ *then add variable attaining equality at* d *to* $\mathcal{A}$.
   - *(f) If* $d = d_2$ *then remove variable attaining* 0 *at* d *from* $\mathcal{A}$.
   - *(g)* $\beta \leftarrow \beta + d\gamma$
   - *(h)* $C = \sum_i a(r(y_i, \beta' x_i)) x_{\mathcal{A},i} x'_{\mathcal{A},i}$
   - *(i)* $\gamma_{\mathcal{A}} = C^{-1}(-sgn(\beta_{\mathcal{A}}))$
   - *(j)* $\gamma_{\mathcal{A}^C} = 0$

It should be noted that our formulation here of the "almost quadratic" family with lasso penalty has ignored the existence of a non-penalized intercept. This has been

done for simplicity of exposition, however incorporating a non-penalized intercept into the algorithm is straight forward (alas not as straight forward as it is for squared error loss, where centering the predictors allows us to fit the intercept separately). The functions we use for numerical examples in the next sections allow for the inclusion of a non-penalized intercept.

6.2.3 Computational considerations

What is the computational complexity of running algorithm 6.1 on a data-set with n observations and p variables? Let us start by considering the simpler question of the complexity of the LARS-lasso algorithm , i.e. algorithm 6.1 when the loss is squared error loss. As Efron, Hastie, Johnstone & Tibshirani (2002) discuss, the complexity of LARS-lasso is "approximately" that of one least squares calculation, i.e. $O(p^3+np^2) = O(np^2)$ when $n > p$. The qualification refers to the fact that "drop" events in algorithm 6.1 — that is, if $d = d_2$ in step $2(d)$ — are not accounted for, and there is no simple theoretical bound on how many of these events can occur. It stands to reason that "on average" these events are rare (see Efron, Hastie, Johnstone & Tibshirani (2002) for discussion) but that cannot be guaranteed.

In the same sense we can say that in the "average" case, the complexity of algorithm 6.1 is $O(n^2p)$ when $n > p$. The additional complexity incurred by this algorithm is twofold:

- Figuring out the step size requires considering all possible $O(n)$ "knot crossing" events, which cannot occur in the lasso
- Knot crossing events increase the number of steps of the algorithm. The rarity assumption on drop events made the number of steps of the lasso $O(p)$. Here we add the assumption that "knot crossing" events occur only $O(n)$ times. This second assumption is very reasonable in the "average" case, since the residuals

tend to move monotonically towards from 0 as the regularization parameter λ decreases.

Thus overall we assume we have $O(n+p)$ steps, each requiring:

- $O((n+p)p) = O(np)$ calculations to figure out the step length (steps 2(a)-2(d) of algorithm 6.1)
- $O(p^2)$ calculations to calculate the new direction (step 2(i)) using the Sherman-Morrison-Woodbury lemma and the fact that we are either adding or dropping a single variable or a single observation.

which gives us an overall complexity of $O((n+p)^2p+(n+p)p^2) = O(n^2p)$ when $n > p$, given our (reasonable, but not guaranteed) assumptions on the number of steps.

An interesting case, especially in micro-array data analysis, is when $p >> n$. In that case, a least squares calculation will incur a complexity of only $O(n^3)$. Using similar assumptions to the ones above (i.e. that the number of "drop" events is small, i.e. $O(1)$, and the number of "know crossing" events is $O(n)$), we get a computational complexity of $O(pn^2)$ for both the lasso and algorithm 6.1

In both cases, the complexity is dominated by the step-length calculations, and the direction "least squares" calculations are less costly.

6.3 Robust methods for regression and classification

We now concentrate on a detailed statistical and computational analysis of two members of the "almost quadratic" family of loss functions — one for regression and one for classification. We choose loss functions which are robust, i.e. linear for large residuals (or margins). We will build on the previous section, in particular algorithm

6.1, to generate the solution paths to the regularized problems $\hat{\beta}(\lambda)$ for the lasso (l_1) regularized versions of these fitting problems.

The primary motivation for use of lasso regularization as presented here is computational, since the resulting problems have piecewise linear solution paths. However there are also statistical reasons why l_1 regularization may be a good idea, and in many cases preferable to the ridge (l_2) regularized solutions. Using l_1 regularization results in "sparse" solutions with a relatively small fraction of non-0 coefficients, as opposed to ridge regularization which forces all non-0 coefficients. Thus, in situations where the number of relevant variables is small and there are a lot of irrelevant "noise" variables, l_1 regularization may prove much superior to l_2 regularization from a prediction error perspective. From an inference/interpretation perspective, l_1 regularization gives "smooth" variable selection and more compact models than l_2 regularization. For detailed discussion of these and other advantages of l_1 regularization see Tibshirani (1996),Friedman, Hastie, Rosset, Tibshirani & Zhu (2003),Efron, Hastie, Johnstone & Tibshirani (2002),Donoho, Johnstone, Kerkyacharian & Picard (1995).

6.3.1 The Huberized Lasso for regression

The loss is given by (6.16). It is obviously "almost quadratic" as defined in section 6.2.2, and so theorem 6.2 and algorithm 6.1 apply to its l_1 regularized solution paths.

In section 6.1.1 we have illustrated on an artificial example the robustness of the huberized lasso against incorrect distributional assumption. In (Huber 1964), Huber has shown that "huberizing" has some asymptotic optimality properties in protecting against "contamination" of the assumed normal errors. For our (very practical) purpose the main motivating property of the huberized loss is that it protects us against both extreme "outliers" and long tailed error distributions.

The one open issue is how to select the "knot" t in (6.16). Huber (1964) suggests

$t = 1.345\sigma$, where σ^2 is the variance of the non-contaminated normal, and describes an iterative algorithm for fitting the data and selecting t, when the variance is unknown. Since our algorithm 6.1 does not naturally admit changing the knot as it advances, we need to iterate the whole algorithm to facilitate adaptive selection of t. This is certainly possible and practical, however for clarity and brevity we use a fixed-knot approach in our examples.

Prostate cancer dataset

Consider the "prostate cancer" dataset, used in the original lasso paper (Tibshirani 1996), and available from: *http://www-stat.stanford.edu/~tibs/ElemStatLearn/data.html*. We use this dataset to compare the prediction performance of the "huberized" lasso to that of the lasso on the original data and after we artificially "contaminate" the data by adding large constants to a small number of responses.

We use the training-test configuration as in Hastie, Tibshirani & Friedman (2001), page 48. The training set consists of 67 observations and the test set of 30 observations. We ran the lasso and the huberized lasso with "knot" at $t = 1$ on the original dataset, and on the "contaminated" dataset where 5 has been added to the response of 6 observations, and 5 was subtracted from the response of 6 other observations. This contamination is extreme in the sense that the original responses vary between -0.4 and 5.6, so their range is more than doubled by the contamination, and practically all contaminated observations become outliers.

Figure 6.5 shows the mean squared error on the 30 test set observations for the four resulting regularized solution paths from solving the lasso and huberized lasso for all possible values of λ on the two data sets. We observe that on the non-contaminated data, the lasso (full) and huberized lasso (dashed) perform quite similarly. When we add contamination, the huberized lasso (dash-dotted) does not seem to suffer from it at all, in that its best test set performance is comparable to that of both

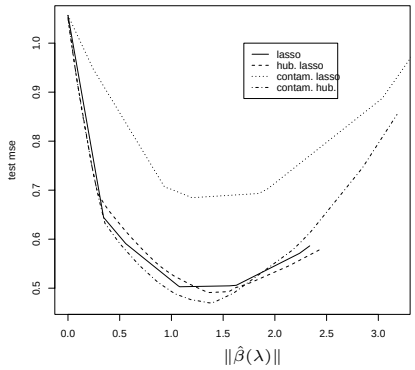


Figure 6.5: Mean test squared error of the models along the regularized path for the lasso (full), huberized lasso (dashed), lasso on contaminated training data (dotted) and huberized lasso on contaminated data (dash-dotted). We observe that the huberized lasso is not affected by contamination at all, while the lasso performance deteriorates significantly.

regularized models on the non-contaminated data. The prediction performance of the non-huberized lasso (dotted), on the other hand, deteriorates significantly when contamination is added, illustrating the lack of robustness of squared error loss. A naive paired t-test of the difference in test set MSE between the best lasso model and the best huberized lasso model on the contaminated data gives a marginally significant p-value of 4.5%. However the difference in performance seems quite striking.

The two lasso solutions contain 9 pieces each, while the Huber-lasso path for the non-contaminated data contains 41 pieces, and the one for the contaminated data contains 39 pieces.

6.3.2 The Huberized truncated squared loss for classification

For classification we would like to have a loss which is a function of the margin: $r(y, \beta' x) = (y\beta' x)$. This is true of all loss functions typically used for regression:

$$\begin{aligned} \text{Logistic regression:} \qquad & l(y, \beta' x) = log(1 + \exp(-y\beta' x)) \\ \text{Support vector machines:} \qquad & l(y, \beta' x) = (1 - y\beta' x)_+ \\ \text{Exponential loss (boosting):} \qquad & l(y, \beta' x) = \exp(-y\beta' x) \end{aligned}$$

The properties we would like from our classification loss are:

- We would like it to be "almost quadratic", so we can apply algorithm 6.1.
- We would like it to be robust, i.e. linear for large absolute value negative margins, so that outliers would have a small effect on the fit. This property is shared by the loss functions used for logistic regression and support vector machines. The truncated squared loss (AKA squared hinge loss) (6.17) and more so the exponential loss are non-robust in this sense.

This leads us to suggesting for classification the "Huberized truncated squared loss", i.e. (6.17) "Huberized" at $t < 1$:

$$l(r) = \begin{cases} (1-t)^2 + 2(1-t)(t-r) & \text{if } r \le t \\ (1-r)^2 & \text{if } r \le 1 \\ 0 & \text{otherwise} \end{cases} \tag{6.32}$$

Figure 6.6 compares some of these classification loss functions: the logistic, exponential, truncated squared loss and our suggested loss function (6.32), with $t = -1$. The exponential and logistic are scaled up by 4 to make comparison easier. We can see the non-robustness of the squared and more so the exponential in the way they "diverge" as the margins become negative.

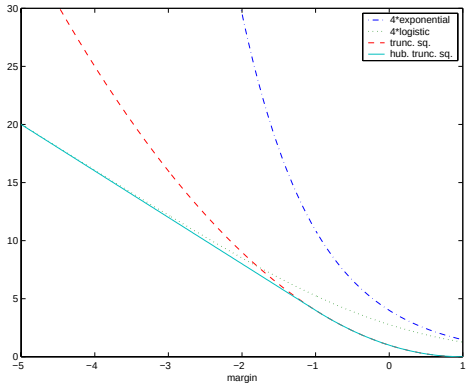


Figure 6.6: Classification loss functions

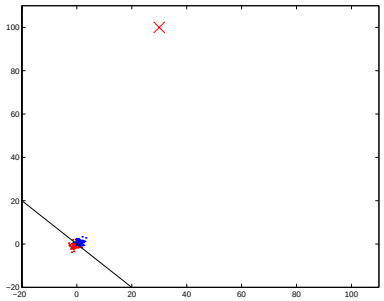


Figure 6.7: Simulated classification data

To illustrate the similarity between our loss (6.32) and the logistic loss, and their difference from the truncated squared loss (6.17), consider the following simple simulated example:$x \in \mathcal{R}^2$ with class centers at $(-1,-1)$ (class "-1") and $(1,1)$ (class "1") with one big outlier at $(30,100)$, belonging to the class "-1". The Bayes model, ignoring the outlier, is to classify to class "1" iff $x_1 + x_2 > 0$. The data and optimal separator can be seen in figure 6.7.

Figure 6.8 shows the regularized model paths and misclassification rate for this data using the logistic loss (left), the huberized truncated loss (6.32) (middle) and the truncated squared loss (6.17) (right). We observe that the logistic and huberized regularized model paths are similar and they are both less affected by the outlier than the non-huberized squared loss.

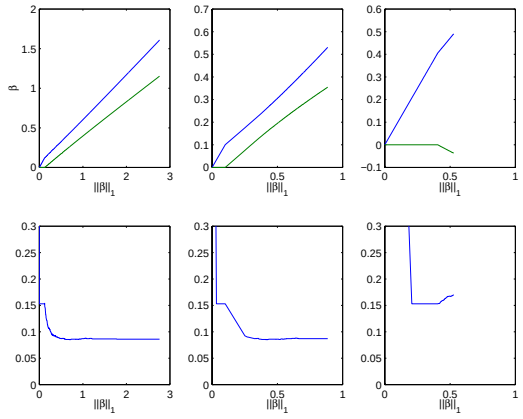


Figure 6.8: Regularized paths and prediction error for the logistic loss (left), huberized truncated squared loss (middle) and truncated squared loss (right)

Spam dataset

We illustrate the algorithm on the "Spam" dataset, available from the UCI repository (Blake & Merz 1998). The dataset contains 4601 observations and 57 variables. To make the problem more interesting from a regularization perspective we use only 1000 of the available data for model training, leaving the rest for evaluation. We want to test the effect of "contamination" on the robustness of the huberized and non-huberized truncated squared error loss. Label contamination in the 2-class classification case is limited to switching the class of the response — there are no real response "outliers" in that sense. We now describe our experiments, comparing the performance of $l1$-penalized truncated squared error (6.17) and its Huberized version (6.32) on this dataset, using $t = -.5$.

1. **Original data:** we ran 10 repetitions of algorithm 6.1 with random selections of 1000 observations for training. The left panel of figure 6.9 shows the test error of the two regularized model paths on one of these repetitions . As we can see, the behavior of the two loss functions is quite similar, probably due to the fact that few observations get to the "Huberized" section of the loss,

corresponding to large absolute value negative margins. The average difference in test error of the best prediction models in the regularized sequences was only 0.2%.

2. **Contaminated response:** we ran the same 10 training-test samples as in the first experiment, but this time we randomly "flipped" the labels of 10% of the training data. The right panel of figure 6.9 shows the test error of the two regularized paths on the same data as in the left panel, this time with the contamination. We observe that the performance of both model deteriorates, but that the advantage of the "Huberized" version is now slightly bigger. This effect is consistent throughout the 10 repetitions. The average difference in best performance between the robust (Huberized) regularized path and the non-robust one is now 0.5%, and a naive matched t-test over the 10 repetitions gave a significance level of just under 4%.

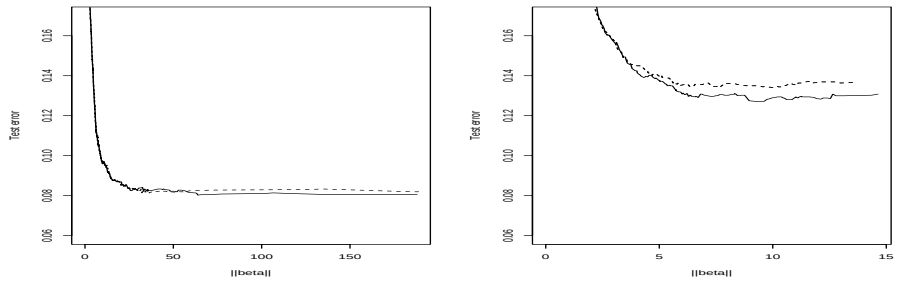


Figure 6.9: The prediction error of the resgularized models for the Spam data (left) and for the same data with label contamination (right), as a function of model norm. The *broken* line is for models created using the *truncated squared* loss and the *full* line is for the *Huberized truncated squared* loss

Our overall conclusion from these and other experiments is that robustness and regularization are indeed issues in classification as well as in regression. However it seems

that their effect in classification is lee pronounced, due to a variety of reasons, not all of which are clear to us at this point.

6.4 Other piecewise linear models of interest

6.4.1 Using l_1 loss and its variants

Our discussion above has concentrated on differentiable loss functions, with a quadratic component. It is also interesting to consider piecewise-linear non-differentiable loss functions, which appear in practice in both regression and classification problems. For regression, the absolute loss is quite popular:

$$l(y, \beta' x) = |y - \beta' x| \tag{6.33}$$

For classification, the hinge loss is of great importance, as it is the loss underlying support vector machines (Vapnik 1995):

$$l(y, \beta' x) = [1 - y\beta' x]_+ \tag{6.34}$$

The second derivatives of both the loss and the penalty vanish where defined so our analysis of the differentiable cases (6.12) cannot be applied directly, but it stands to reason that the solution paths to l_1-penalized versions of (6.33) and (6.34) are also piecewise linear. This is in fact the case, however proving it and developing an algorithm for following the regularized path require a slightly different approach from the one we presented here. In Zhu, Hastie & Tibshirani (2003) we present an algorithm for following the lasso-penalized solution path of (6.34). Figure 6.10, reproduced from (Zhu, Hastie & Tibshirani 2003), illustrates this algorithm on a simple 3-variable simulated example. As it turns out l_2 (ridge) penalized solution

paths of (6.33) and (6.34) also follow a piecewise linear solution path — this is a subject of active research.

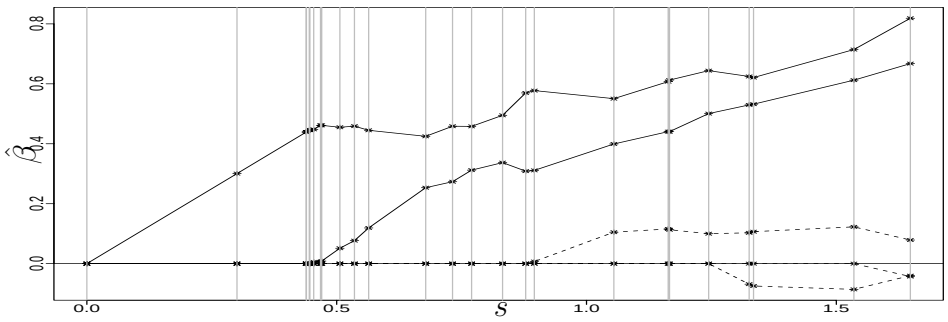


Figure 6.10: The solution path $\hat{\beta}(s)$ as a function of s.

Another interesting variant is to replace the l_1 loss with an l_∞ loss, which is also linear "almost everywhere" and non-differentiable. Similar properties hold for that case, i.e. the solution paths are piecewise-linear and similar algorithms can be used to generate them. However working out the details and implementing algorithms for this case remains as a future task (in section 6.4.2 we use l_∞ as the penalty and show the similarity between the resulting algorithms and the ones using the l_1 penalty. A similar approach can be applied on the loss side).

6.4.2 The l_∞ penalty

The l_∞ penalty $J(\beta) = \max_j |\beta_j|$ is also piecewise linear in $\mathcal{R}^k$. It has non differentiability points where the maximum coefficient is not unique. To demonstrate the piecewise linearity regularized solution paths using almost-quadratic loss functions and the l_∞ penalty, consider first the optimization formulation of the problem. As

before, we will code $\beta_j = \beta_j^+ - \beta_j^-$, with $\beta_j^+ \geq 0, \beta_j^- \geq 0$, and start from the constrained version of the regularized optimization problem:

$$\begin{aligned} \min_{\beta^+,\beta^-} \quad & \sum_i l(y_i, (\beta^+ - \beta^-)' x_i) \\ \text{s.t.} \quad & \beta_j^+ + \beta_j^- \leq M, \ \forall j \\ & \beta_j^+ \geq 0, \ \beta_j^- \geq 0, \ \ \forall j \end{aligned} \tag{6.35}$$

then we can write the Lagrange dual function of our minimization problem as:

$$\min_{\beta^+,\beta^-} \sum_i l(y_i, (\beta^+ - \beta^-)' x_i) + \sum_j \lambda_j (\beta_j^+ + \beta_j^- - M) - \lambda_j^+ \beta_j^+ - \lambda_j^- \beta_j^- \tag{6.36}$$

And the corresponding KKT optimality conditions:

$$(\nabla L(\beta))_j - \lambda_j^+ + \lambda_j = 0 \tag{6.37}$$

$$-(\nabla L(\beta))_j - \lambda_j^- + \lambda_j = 0 \tag{6.38}$$

$$\lambda_j (\beta_j^+ + \beta_j^- - M) = 0 \tag{6.39}$$

$$\lambda_j^+ \beta_j^+ = 0 \tag{6.40}$$

$$\lambda_j^- \beta_j^- = 0 \tag{6.41}$$

Using these we can figure out that at the optimum the possible scenarios are:

$$\beta_j^+ + \beta_j^- < M \ \Rightarrow \ \lambda_j = 0 \ \Leftrightarrow \ \lambda_j^+ = \lambda_j^- = (\nabla L(\beta))_j = 0 \tag{6.42}$$

$$(\nabla L(\beta))_j > 0 \ \Rightarrow \ \lambda_j^+ > 0, \lambda_j > 0 \ \Rightarrow \ \beta_j^+ = 0, \beta_j^- = M \tag{6.43}$$

$$(\nabla L(\beta))_j < 0 \ \Rightarrow \ \lambda_j^- > 0, \lambda_j > 0 \ \Rightarrow \ \beta_j^- = 0, \beta_j^+ = M \tag{6.44}$$

We can use these to construct an algorithm for solution paths for l_∞ regularized "almost quadratic" loss functions, in the spirit of algorithm 6.1 above. In the l_∞

version, all variables have the maximal absolute value coefficient, except the ones which have 0 "generalized correlation" (i.e. derivative of loss). Those are changing freely, in such a way as to keep the correlation at 0. Following is an algorithm for l_∞ penalized squared error loss regression — we present this rather than the more general "almost quadratic" version for clarity and brevity.

Algorithm 6.2 *An algorithm for squared error loss with l_∞ penalty*

1. *Initialize:* $\beta = 0,\ \mathcal{A} = \emptyset,\ \gamma = -sgn(\nabla L(\beta))$
2. *While* $(max|\nabla L(\beta)| > 0)$
   (a) $d_1 = \min\{d \geq 0 : \min_{j \notin \mathcal{A}} |\nabla L(\beta + d\gamma)_j| = 0\}$
   (b) $d_2 = \min\{d \geq 0 : \min_{j \in \mathcal{A}} |\beta_j + d\gamma_j| = |\beta_{\mathcal{A}^C}| + d\}$
   (c) *set* $d = min(d_1, d_2)$
   (d) *If* $d = d_1$ *then add variable attaining equality at* d *to* $\mathcal{A}$.
   (e) *If* $d = d_2$ *then remove variable attaining equality at* d *from* $\mathcal{A}$.
   (f) $\beta \leftarrow \beta + d\gamma$
   (g) $\gamma_{\mathcal{A}^C} = (-sgn(\nabla L(\beta))_{\mathcal{A}^C})$
   (h) $\gamma_{\mathcal{A}} = (X'_{\mathcal{A}} X_{\mathcal{A}})^{-1} X'_{\mathcal{A}} X_{\mathcal{A}^C} \gamma_{\mathcal{A}^C}$

Figure 6.11 illustrates the resulting coefficient paths when running algorithm 6.2 on the diabetes data-set, used for illustrating the lasso in Efron, Hastie, Johnstone & Tibshirani (2002). Notice how all coefficients start moving away from 0 together, then leave the "maximal" set one by one as their correlations hit zero.

The statistical motivation for using l_∞ penalty is not clear, and we have not encountered such examples in the statistics literature. It may actually be more adequate

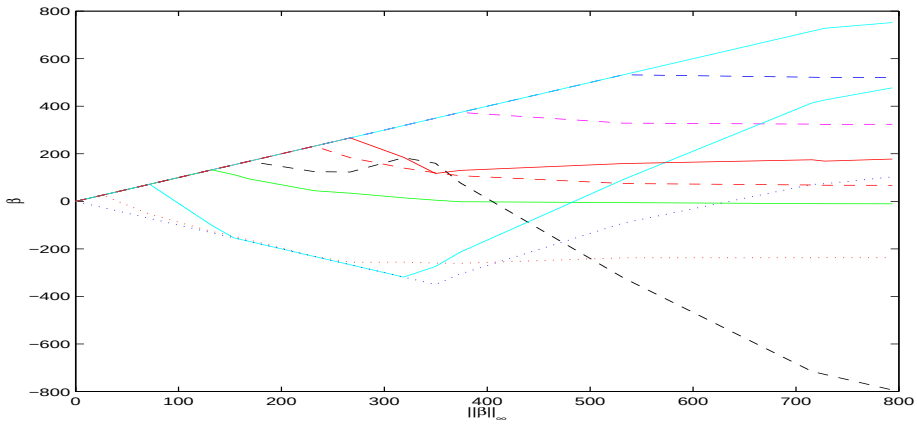


Figure 6.11: Solution paths for l_∞ penalized squared error loss, using the diabetes dataset

for operations research scenarios, where the coefficients represent costs, such as operating times of machines, and the overall cost depends on the maximal one. We feel it is still an open issue whether there are compelling examples to that effect, however it is worth noticing that the possibility exists.

6.4.3 Multiple penalty problems

Suppose now that we would like to penalize more than one function of the coefficients, i.e. solve:

$$\hat{\beta}(\lambda_1, \lambda_2) = \min_{\beta} L(\mathbf{y}, X\beta) + \lambda_1 J_1(\beta) + \lambda_2 J_2(\beta) \tag{6.45}$$

then we can easily see that the theory we have developed tells us that as long as *both* penalties and the loss follow the conditions we have defined for piecewise linearity, the solution $\hat{\beta}(\lambda_1, \lambda_2)$ will be a *piecewise affine* surface in $\mathcal{R}^p$. In particular, if we limit our interest to a 1-dimensional line in (λ_1, λ_2) space, i.e. $\lambda_1 = a + b\lambda_2$, we get a piecewise-linear solution path.

An example where the two penalty formulation is natural and of great practical

interest can be found in Tibshrani, Saunders, Rosset & Zhu (2003). The problem considered is that of protein mass spectroscopy, and the predictors correspond to a continuum of "time of flight" sites. Thus the predictors have an order and a distance between them, and there are scientific reasons why we would expect neighboring coefficients to be much more correlated than distant ones. We would like a method which strongly prefers to make neighboring coefficients similar, while still limiting the amount of non-0 coefficients. A natural way to achieve that is to include a separate penalty on magnitude of coefficients and on differences of neighboring coefficients:

$$\min_\beta \quad \|\mathsf{y} - X\beta\|^2 + \lambda_1 \sum_j |\beta_j| + \lambda_2 \sum_{j>1} |\beta_j - \beta_{j-1}| \tag{6.46}$$

In Tibshrani, Saunders, Rosset & Zhu (2003) we show how we can formulate an incremental algorithm for this problem which is a more involved version of the LARS algorithm of Efron, Hastie, Johnstone & Tibshirani (2002) and our algorithm 6.1.

6.5 Discussion: robustness and regularization

In this chapter we have tried to design methods which combine computational and statistical advantages. We emphasize the importance of both regularization and robust loss functions for successful practical modeling of data. From a statistical perspective, we can consider robustness and regularization as almost independent desirable properties dealing with different issues in predictive modeling:

- Robustness mainly protects us against wrong assumptions about our error (or noise) model. It does little or nothing to protect us against the uncertainty about our model structure which is inherent in the finiteness of our data. For example, if our errors really are normal, then squared error loss minimizes the asymptotic variance of the coefficients, no matter how little data we have or

how inappropriate our model is (Huber 1964). Using a robust loss in such a situation is *always* counter productive.

- Regularization deals mainly with the uncertainty about our model structure by limiting the model space. Note, in this context, the equivalence between the "penalized" formulation (6.1) and a "constrained" formulation:

$$\min_{\beta} L(\mathsf{y}, X\beta) \text{ s.t. } J(\beta) \leq s \tag{6.47}$$

The two formulations are equivalent in the sense that every solution $\hat{\beta}(s)$ to (6.47) is equal to a solution $\hat{\beta}(\lambda)$ to (6.1) for some value of λ and vice versa ((6.1) is in fact just a Lagrange multiplier version of (6.47)). Thus the main goal of regularization is to make the model estimation problem easier, and match it to the amount of information we have for estimating it, namely our training data.

We can also take a Bayesian approach, in which case the penalized optimization problem (6.1) corresponds to maximizing a posterior log-likelihood, if $J(\beta)$ is a log-prior on the model space and L is a log-likelihood (for example, the lasso maximizes posterior with i.i.d Gaussian error model and i.i.d exponential priors on the coefficients). The more vague Bayesian interpretation for using robust loss would be that it represents a reluctance to commit to a specific error model, electing instead to use a "non-informative" one which can accommodate a range of reasonable error models, in particular ones contaminated by outliers, without suffering significant degradation in performance.

Chapter 7

Approximate quadratic method for following curved solution paths

In this chapter we build on classical "newton"-like methods to approximately track the parametrized path of regularized solutions:

$$\hat{\beta}(\lambda) = \arg\min_{\beta} \sum_i L(y_i, \beta' x_i) + \lambda J(\beta) \tag{7.1}$$

Where we will assume that both L and J are convex, twice differentiable and "nice" (in an obvious sense of bounded derivatives, which we define rigorously later). The key to our method are the normal equations for (7.1):

$$\nabla L(\hat{\beta}(\lambda)) + \lambda \nabla J(\hat{\beta}(\lambda)) = 0 \tag{7.2}$$

7.1 Algorithm

Our algorithm constructs an approximate solution $\beta^{(\epsilon)}(t)$ by taking "small" ϵ steps in a Newton-Raphson direction trying to maintain (7.2). Our main result in this

chapter is to show, both empirically and theoretically, that for small ϵ, the difference $\|\beta^{(\epsilon)}(t) - \hat{\beta}(\epsilon \cdot t)$ is small, and thus that our method successfully tracks the path of optimal solutions to (7.1).

Algorithm 7.1 *Approximate incremental quadratic algorithm for regularized optimization*

1. *Set* $\beta^{(\epsilon)}(0) = \hat{\beta}(\lambda_0)$, *set* $t = 0$.
2. *While* $(\lambda_t < \lambda_{max})$
   (a) $\lambda_{t+1} = \lambda_t + \epsilon$
   (b)

$$\beta^{(\epsilon)}(t+1) \leftarrow \beta^{(\epsilon)}(t) - (\nabla^2 L(\beta^{(\epsilon)}(t)) + \lambda_{t+1}\nabla^2 J(\beta^{(\epsilon)}(t)))^{-1} \cdot [\nabla L(\beta^{(\epsilon)}(t)) + \lambda_{t+1}\nabla J(\beta^{(\epsilon)}(t))]$$

   (c) $t = t + 1$

Algorithm 7.1 gives a formal description of our quadratic tracking method. We start from a solution to (7.1) for some fixed λ_0 (e.g. $\hat{\beta}(0)$, the non-regularized solution). At each iteration we increase λ by ϵ and take a single Newton-Raphson step towards the solution to (7.2) with the current λ value in step 2.b.

We illustrate the empirical usefulness of this algorithm in section 7.2 and prove a theoretical result in section 7.3, which implies that under "regularity" conditions, the algorithm guarantees that

$$\forall c > 0, \quad \beta^{(\epsilon)}(c/\epsilon) \xrightarrow[\epsilon \to 0]{} \hat{\beta}(\lambda_0 + c) \tag{7.3}$$

7.2 Data examples

We illustrate our quadratic method on regularized logistic regression using a small subset of the "spam" dataset (Blake & Merz 1998).

We choose five variables and 300 observations and track the solution paths to two regularized optimization problems using this data:

$$\hat{\beta}(\lambda) = \arg\min_{\beta} L(y_i, \beta' x_i) + \lambda \|\beta\|_2^2 \tag{7.4}$$

$$\hat{\beta}(\lambda) = \arg\min_{\beta} L(y_i, \beta' x_i) + \lambda \|\beta\|_{1.1}^{1.1} \tag{7.5}$$

Where the loss is the logistic log-likelihood:

$$L(y, f) = (\frac{e^f}{e^f + 1})^y + (\frac{1}{e^f + 1})^{1-y}$$

The two solution paths we calculate differ in the regularization they are using. The l_2 penalty of (7.4) gives the standard "penalized logistic regression" method, popular in high-dimensional data modeling, where non-regularized solutions are not useful, or even undefined (probably the prime current application for this approach is in modeling gene microarrays). The $l_{1.1}$ penalty of (7.5) is devised to make the problem somewhat like using l_1 regularization, i.e. the fit to be dominated by a few large coefficients. In section 7.4.2 we discuss and illustrate an approach for applying our method to l_1-regularized models, which slightly deviate from our standard formulation since the penalty $\|\beta\|_1$ is not differentiable.

Figure 7.1 shows the solution paths $\beta^{(\epsilon)}(t)$ generated by running algorithm 7.1 on this data using $\epsilon = 0.02$ and srtrting at $\lambda = 0$, i.e. from the non-regularized logistic regression solution. The interesting graphs for our purpose are the ones on the right.

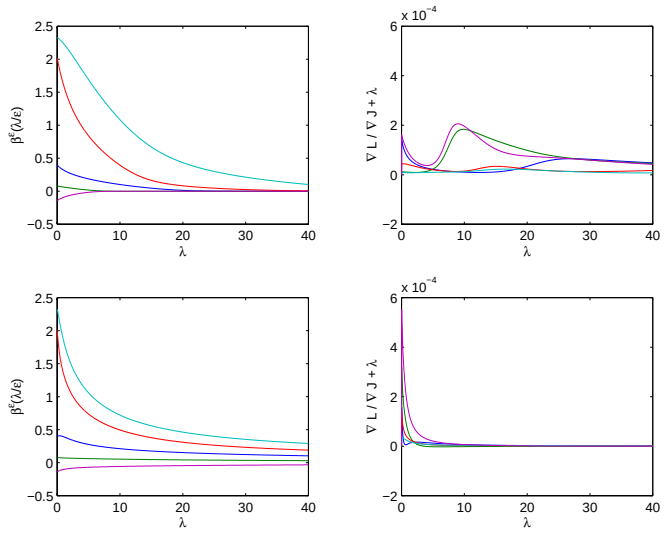


Figure 7.1: Solution paths (left) and optimality criterion (right) for $l_{1.1}$ penalized logistc regression (top) and l_2 penalized logistic regression (bottom). These result from sunning algorithm 7.1 using $\epsilon = 0.02$ and starting from the non-regularized logistic regression solution

They represent the "optimality criterion":

$$G(\beta^{(\epsilon)}(t)), t) = \frac{\nabla L(\beta^{(\epsilon)}(t))}{\nabla J(\beta^{(\epsilon)}(t))} + \epsilon \cdot t$$

where the division is done componentwise. Note that the optimal solution $\hat{\beta}(t\epsilon)$ is uniquely defined by the fact that (7.2) holds and therefore $G(\hat{\beta}(t\epsilon), t\epsilon) \equiv 0$ componentwise. By convexity and regularity of the loss and the penalty, there is a correspondence between small values of G and small distance $\|\beta^{(\epsilon)}(t) - \hat{\beta}(t\epsilon)$.

In our example we observe that the components of G seem to be bounded in a small region around 0 for both paths (note the small scale of the y axis in both plots - the maximal error is less than 10^{-3}).

Thus we conclude that on this simple example our method tracks the optimal solution paths nicely.

7.3 Theoretical result

Theorem 7.1 *Assume $\lambda_0 > 0$, then under regularity conditions on the derivatives of L and J,*

$$\forall c > 0 \ , \ \ \|\beta^{(\epsilon)}(c/\epsilon) - \hat{\beta}(\lambda_0 + c)\| = O(\epsilon^2)$$

So there is a uniform *bound $O(\epsilon^2)$ on the error which* does not *depend on c.*

Proof We give the details of the proof in appendix A. Here we give a brief review of the main steps.

Denote:

$$|(\frac{\nabla L(\beta^{(\epsilon)}(t))}{\nabla J(\beta^{(\epsilon)}(t))})_j + \lambda_t| = e_{tj} \tag{7.6}$$

And define a "regularity constant" M, which depends on λ_0 and the first, second and third derivatives of the loss and penalty.

The proof is presented as a succession of lemmas:

Lemma 7.2 *Assume we have m predictors. Let $u_1 = M \cdot m \cdot \epsilon^2$, $u_t = M(u_{t-1} + \sqrt{m} \cdot \epsilon)^2$, then: $\|\mathbf{e}_t\|_2 \leq u_t$*

This lemma gives a recursive expression bounding the error in the "normal equations" (7.6) as the algorithm proceeds.

Lemma 7.3 *If $\sqrt{m}\epsilon M \leq 1/4$ then $u_t \nearrow \frac{1}{2M} - \sqrt{m} \cdot \epsilon - \frac{\sqrt{1-4\sqrt{m}\cdot\epsilon M}}{2M} = O(\epsilon^2)$*

This lemma shows that the recursive bound translates to an absolute $O(\epsilon^2)$ bound, if ϵ is small enough.

Lemma 7.4 *Under regularity conditions on the functions and the solutions to (7.1), the $O(\epsilon^2)$ uniform bound of lemma 7.3 translates to an $O(\epsilon^2)$ uniform bound on $\|\beta^{(\epsilon)}(c/\epsilon) - \hat{\beta}(\lambda_0 + c)\|$*

7.4 Some extensions

In this section, we discuss some cases that do not strictly abide with the conditions of the theorem but are of great practical interest.

7.4.1 The Lasso and other non-twice-differentiable penalties

If $J(\beta)$ in (7.1) is not twice differentiable then we cannot apply our method directly, since the theory may not apply. In particular, consider using the lasso penalty $J(\beta) = \|\beta\|$. In chapter 6 we have shown that if the loss is "piecewise quadratic" then we can derive the lasso-penalized solution path $\hat{\beta}(\lambda)$ directly and exactly without reverting to the approximate quadratic method. However, if we want to use a non-quadratic loss function (such as the logistic log-likelihood we used in section 7.2) with the lasso penalty we need to utilize a quadratic approximation.

To understand how we can generalize algorithm 7.1 and theorem 7.1 to this situation, consider the optimization formulation for the lasso-penalized problem presented in chapter 6, in (6.18), and the conclusions drawn about the resulting optimal solution path in (6.25)–(6.28). These hold for any loss and tell us that at each point on the path we have a set $\mathcal{A}$ of non-0 coefficients which correponds to the variables whose current "generalized correlation" $|\nabla L(\hat{\beta}(\lambda))_j|$ is maximal and equal to λ. We can summarize the situation as:

$$|\nabla L(\hat{\beta}(\lambda))_j| < \lambda \;\Rightarrow\; \hat{\beta}(\lambda)_j = 0 \tag{7.7}$$

$$\hat{\beta}(\lambda)_j \neq 0 \;\Rightarrow\; |\nabla L(\hat{\beta}(\lambda))_j| = \lambda \tag{7.8}$$

Using these rules we can now adapt algorithm 7.1 to working with a lasso penalty, as follows:

Algorithm 7.2 *Approximate incremental quadratic algorithm for regularized optimization with lasso penalty*

1. *Set* $\beta_0^{(\epsilon)} = \hat{\beta}(\lambda_0)$, *set* $t = 0$, *set* $\mathcal{A} = \{j : \hat{\beta}(\lambda_0)_j \neq 0\}$.
2. *While* $(\lambda_t < \lambda_{max})$
   (a) $\lambda_{t+1} = \lambda_t + \epsilon$
   (b)
   $$\beta_{t+1}^{(\epsilon)} \leftarrow \beta_t^{(\epsilon)} - [\nabla^2 L(\beta_t^{(\epsilon)})_{\mathcal{A}}]^{-1} \cdot [\nabla L(\beta_t^{(\epsilon)})_{\mathcal{A}} + \lambda_{t+1} sgn(\beta_t^{(\epsilon)})_{\mathcal{A}}]$$
   (c) $\mathcal{A} = \mathcal{A} \cup \{j \notin \mathcal{A} : \nabla L(\beta_{t+1}^{(\epsilon)})_j > \lambda_{t+1}\}$
   (d) $\mathcal{A} = \mathcal{A} - \{j \in \mathcal{A} : |\beta_{t+1,j}^{(\epsilon)}| > \delta\}$
   (e) $t = t + 1$

Algorithm 7.2 employs the Newton approach of algorithm 7.1 for twice differentiable penalty, limited to the sub-space of "active" coefficients denoted by $\mathcal{A}$. In that sub-space the penalty is indeed twice differntiable since all the coefficients are non-0. It adds to algorithm 7.1 updates for the "add variable to active set" and "remove variable from active set" events, when equality in correlation and 0 coefficient are attained, respectively.

7.4.2 Example: logistic regression with lasso penalty

We continue the example of section 7.2, this time with a lasso penalty:

$$\hat{\beta}(\lambda) = \arg\min_{\beta} L(y_i, \beta' x_i) + \lambda \|\beta\|_1 \tag{7.9}$$

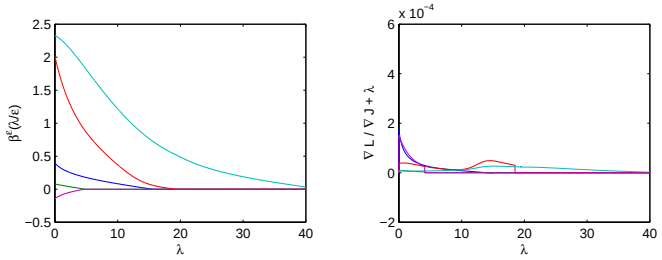


Figure 7.2: Solution path (left) and optimality criterion (right) for l_1 penalized logistc regression. These result from sunning algorithm 7.2 using $\epsilon = 0.02$ and starting from the non-regularized logistic regression solution

Figure 7.2 shows the result of applying algorithm 7.2 to the same 300 observations and 5 variables as in section 7.2. We observe that the coefficient paths seem to track the optimal solution nicely, by looking at the optimallity criterion in the right panel. We can also observe the variable selection effect of using the lasso penalty, in the left panel. For example, for all values of λ above 20 we have only a single non-0 coefficient in the regularized solution. By comparison, in the solution paths to the $l_{1.1}$ and l_2 regularized models, displayed in figure 7.1, we observe that all coefficients are non-0 for all value of the regularization parameter λ.

Appendix A

Proofs of theorems

Proof of local equivalence of ϵ-boosting and lasso from chapter 2

As before, we assume we have a set of training data $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots (\mathbf{x}_n, y_n)$, a smooth cost function $C(y, F)$, and a set of basis functions $(h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots h_J(\mathbf{x}))$.

We denote by $\hat{\beta}(s)$ be the optimal solution of the L_1-constrained optimization problem:

$$\min_{\beta} \quad \sum_{i=1}^{n} C(y_i, h(\mathbf{x}_i)^T \beta) \tag{A.1}$$

$$\text{subject to} \quad \|\beta\|_1 \leq s. \tag{A.2}$$

Suppose we initialize the ϵ-boosting version of algorithm 2.1, as described in section 2.2, at $\hat{\beta}(s)$ and run the algorithm for T steps. Let $\beta(T)$ denote the coefficients after T steps.

The "global convergence" conjecture 2.2 in section 2.4 implies that $\forall \Delta s > 0$:

$$\beta(\Delta s/\epsilon) \to \hat{\beta}(s + \Delta s) \ \text{ as } \ \epsilon \to 0$$

under some mild assumptions. Instead of proving this "global" result, we show here a "local" result by looking at the derivative of $\hat{\beta}(s)$. Our proof builds on the proof by Efron, Hastie, Johnstone & Tibshirani (2002) (theorem 2) of a similar result for the case that the cost is squared error loss $C(y, F) = (y - F)^2$. Theorem (A.1) below shows that if we start the ϵ-boosting algorithm at a solution $\hat{\beta}(s)$ of the L_1-constrained optimization problem (A.1) - (A.2), the "direction of change" of the ϵ-boosting solution will agree with that of the L_1-constrained optimizaton problem.

Theorem A.1 *Assume the optimal coefficient paths $\hat{\beta}_j(s)\ \forall j$ are monotone in s and the coefficient paths $\beta_j(T)\ \forall j$ are also monotone as ϵ-boosting proceeds, then*

$$\frac{\beta(T) - \hat{\beta}(s)}{T \cdot \epsilon} \to \nabla \hat{\beta}(s) \quad as \quad \epsilon \to 0, T \to \infty, T \cdot \epsilon \to 0.$$

Proof First we introduce some notations. Let

$$\mathbf{h}_j = (h_j(\mathbf{x}_1), \ldots h_j(\mathbf{x}_n))^T$$

be the jth basis function evaluated at the n training data.

Let

$$\mathbf{F} = (F(\mathbf{x}_1), \ldots F(\mathbf{x}_n))^T$$

be the vector of current fit.

Let

$$\mathbf{r} = \left(-\frac{\partial C(y_1, F_1)}{\partial F_1}, \ldots - \frac{\partial C(y_n, F_n)}{\partial F_n} \right)^T$$

be the current "generalized residual" vector as defined in (Friedman 2001).

Let

$$c_j = \mathbf{h}_j^T \mathbf{r}, \quad j = 1, \ldots J$$

be the current "correlation" between $\mathbf{h}_j$ and $\mathbf{r}$.

Let

$$\mathcal{A} = \{j : |c_j| = \max_j |c_j|\}$$

be the set of indices for the maximum absolute correlation.

For clarity, we re-write this ϵ-boosting algorithm, starting from $\hat{\beta}(s)$, as a special case of Algorithm 1, as follows:

(1) Initialize $\beta(0) = \hat{\beta}(s), \mathbf{F}_0 = \mathbf{F}, \mathbf{r}_0 = \mathbf{r}$.

(2) For $t = 1 : T$

(a) Find $j_t = \arg\max_j |\mathbf{h}_j^T \mathbf{r}_{t-1}|$.

(b) Update

$$\beta_{t,j_t} \leftarrow \beta_{t-1,j_t} + \epsilon \cdot sign(c_{j_t})$$

(c) Update $\mathbf{F}_t$ and $\mathbf{r}_t$.

Notice in the above algorithm, we start from $\hat{\beta}(s)$, rather than 0. As proposed in Efron, Hastie, Johnstone & Tibshirani (2002), we consider an idealized ϵ-boosting

case: $\epsilon \to 0$. As $\epsilon \to 0$, $T \to \infty$ and $T \cdot \epsilon \to 0$, under the monotone paths condition, (Efron, Hastie, Johnstone & Tibshirani 2002) showed

$$\frac{\mathbf{F}_T - \mathbf{F}_0}{T \cdot \epsilon} \to \mathbf{u}, \tag{A.3}$$

$$\frac{\mathbf{r}_T - \mathbf{r}_0}{T \cdot \epsilon} \to \mathbf{v}, \tag{A.4}$$

where $\mathbf{u}$ and $\mathbf{v}$ satisfy two constraints:

(**Constraint 1**) $\mathbf{u}$ is in the convex cone generated by $\{sign(c_j)\mathbf{h}_j : j \in \mathcal{A}\}$, i.e.

$$\mathbf{u} = \sum_{j \in \mathcal{A}} P_j sign(c_j)\mathbf{h}_j, P_j \geq 0.$$

(**Constraint 2**) $\mathbf{v}$ has equal "correlation" with $sign(c_j)\mathbf{h}_j, j \in \mathcal{A}$:

$$sign(c_j)\mathbf{h}_j^T \mathbf{v} = \lambda_{\mathcal{A}} \text{ for } j \in \mathcal{A}.$$

Efron, Hastie, Johnstone & Tibshirani (2002) further showed $\mathbf{u}$ and $\mathbf{v}$ satisfying constraints 1 – 2 are unique.

Now we consider the L_1-constrained optimization problem (A.1 – A.2). Let $\hat{\mathbf{F}}(s)$ be the fitted vector and $\hat{\mathbf{r}}(s)$ be the corresponding residual vector. Since $\hat{\mathbf{F}}(s)$ and $\hat{\mathbf{r}}(s)$ are smooth, define

$$\mathbf{u}^* \equiv \lim_{\Delta s \to 0} \frac{\hat{\mathbf{F}}(s + \Delta s) - \hat{\mathbf{F}}(s)}{\Delta s}, \tag{A.5}$$

$$\mathbf{v}^* \equiv \lim_{\Delta s \to 0} \frac{\hat{\mathbf{r}}(s + \Delta s) - \hat{\mathbf{r}}(s)}{\Delta s}. \tag{A.6}$$

Lemma A.2 *Under the monotone coefficient paths assumption,* $\mathbf{u}^*$ *and* $\mathbf{v}^*$ *also satisfy constraints 1 – 2.*

Proof Write the coefficient β_j as $\beta_j^+ - \beta_j^-$, where

$$\begin{cases} \beta_j^+ = \beta_j, \beta_j^- = 0 & \text{if} \quad \beta_j > 0 \\ \beta_j^+ = 0, \beta_j^- = \beta_j & \text{if} \quad \beta_j < 0 \end{cases}$$

The L_1-constrained optimization problem (A.1) – (A.2) is then equivalent to:

$$\min_{\beta^+, \beta^-} \quad \sum_{i=1}^{n} C\left(y_i, h(\mathbf{x}_i)^T(\beta^+ - \beta^-)\right), \tag{A.7}$$

$$\text{subject to} \quad \|\beta^+\|_1 + \|\beta^-\|_1 \leq s, \beta^+ \geq 0, \beta^- \geq 0. \tag{A.8}$$

The corresponding Lagrangian dual is:

$$L = \sum_{i=1}^{n} C\left(y_i, h(\mathbf{x}_i)^T(\beta^+ - \beta^-)\right) + \lambda \sum_{j=1}^{J}(\beta_j^+ + \beta_j^-) \tag{A.9}$$

$$-\lambda \cdot s - \sum_{j=1}^{J} \lambda_j^+ \beta_j^+ - \sum_{j=1}^{J} \lambda_j^- \beta_j^-, \tag{A.10}$$

where $\lambda \geq 0, \lambda_j^+ \geq 0, \lambda_j^- \geq 0$ are Lagrange multipliers.

By differentiating the Lagrangian dual, we get the solution of (A.7) – (A.8) need to satisfy the following Karush-Kuhn-Tucker conditions:

$$\frac{\partial L}{\partial \beta_j^+} = -\mathbf{h}_j^T \hat{\mathbf{r}} + \lambda - \lambda_j^+ = 0 \tag{A.11}$$

$$\frac{\partial L}{\partial \beta_j^-} = \mathbf{h}_j^T \hat{\mathbf{r}} + \lambda - \lambda_j^- = 0 \tag{A.12}$$

$$\lambda_j^+ \hat{\beta}_j^+ = 0 \tag{A.13}$$

$$\lambda_j^- \hat{\beta}_j^- = 0 \tag{A.14}$$

Let $c_j = \mathbf{h}_j^T \hat{\mathbf{r}}$ and $\mathcal{A} = \{j : |c_j| = \max_j |c_j|\}$. We can see the following facts from the Karush-Kuhn-Tucker conditions:

(**Fact 1**) Use (A.11), (A.12) and $\lambda \geq 0, \lambda_j^+, \lambda_j^- \geq 0$, we have $|c_j| \leq \lambda$.

(**Fact 2**) If $\hat{\beta}_j \neq 0$, then $|c_j| = \lambda$ and $j \in \mathcal{A}$. For example, suppose $\hat{\beta}_j^+ \neq 0$, then $\lambda_j^+ = 0$ and (A.11) implies $c_j = \lambda$.

(**Fact 3**) If $\hat{\beta}_j \neq 0$, $sign(\hat{\beta}_j) = sign(c_j)$.

Remark:

- $\hat{\beta}_j^+$ and $\hat{\beta}_j^-$ can not both be non-zero, otherwise $\lambda_j^+ = \lambda_j^- = 0$, (A.11) and (A.12) can not hold at the same time.
- It is possible that $\hat{\beta}_j = 0$ and $j \in \mathcal{A}$. This only happens for a finite number of s values, where basis $\mathbf{h}_j$ is about to enter the model.

For sufficiently small Δs, since the second derivative of the cost function $C(y, F)$ is finite, $\mathcal{A}$ will stay the same. Since $j \in \mathcal{A}$ if $\hat{\beta}_j \neq 0$, the change in the fitted vector is:

$$\hat{\mathbf{F}}(s + \Delta s) - \hat{\mathbf{F}}(s) = \sum_{j \in \mathcal{A}} Q_j \mathbf{h}_j.$$

Since $sign(\hat{\beta}_j) = sign(c_j)$ and the coefficients $\hat{\beta}_j$ change monotonically, $sign(Q_j)$ will agree with $sign(c_j)$. Hence we have

$$\hat{\mathbf{F}}(s + \Delta s) - \hat{\mathbf{F}}(s) = \sum_{j \in \mathcal{A}} P_j sign(c_j) \mathbf{h}_j \quad P_j = |Q_j| \geq 0. \tag{A.15}$$

This implies $\mathbf{u}^*$ satisfies constraint 1. The claim $\mathbf{v}^*$ satisfies constraint 2 follows directly from fact 2, since both $\hat{\mathbf{r}}(s + \Delta s)$ and $\hat{\mathbf{r}}(s)$ satisfy constraint 2. □

Completion of proof of theorem (A.1): By the uniqueness of $\mathbf{u}$ and $\mathbf{v}$, we then have

$$\mathbf{u} = \mathbf{u}^* \text{ and } \mathbf{v} = \mathbf{v}^*.$$

To translate the result into $\hat{\beta}(s)$ and $\beta(T)$, we notice $F(\mathbf{x}) = h(\mathbf{x})^T\beta$. (Efron, Hastie, Johnstone & Tibshirani 2002) showed that for $\nabla\hat{\beta}(s)$ to be well defined, $\mathcal{A}$ can have at most n elements, i.e. $|\mathcal{A}| \leq n$. This fact can also be seen from the Karush-Kuhn-Tucker conditions (A.11) - (A.12): suppose $\mathcal{A}$ has $n+1$ elements; without loss of generality, assume $\mathcal{A} = \{1, 2, \ldots n+1\}$ and $\mathbf{h}_j^T\hat{\mathbf{r}} = \lambda, j \in \mathcal{A}$. Recall each $\mathbf{h}_j$ is a n-dimensional vector, therefore $\mathbf{h}_1, \ldots \mathbf{h}_{n+1}$ must be linearly dependent. Let $\mathbf{h}_{n+1} = a_1\mathbf{h}_1 + \ldots + a_n\mathbf{h}_n, a_j \in \mathcal{R}$, then $\mathbf{h}_j^T\hat{\mathbf{r}} = \lambda, j \in \mathcal{A}$ implies $a_1 + \ldots + a_n = 1$, which is in general not true.

Now Let

$$H = \left(h_j(\mathbf{x}_i)\right), i = 1, \ldots n; j \in \mathcal{A}$$

be a $n \times |\mathcal{A}|$ matrix, which we assume is of rank $|\mathcal{A}|$. Then $\nabla\hat{\beta}(s)$ is given by

$$\nabla\hat{\beta}(s) = \left(H^TH\right)^{-1}H^T\mathbf{u}^*,$$

and

$$\frac{\beta(T) - \hat{\beta}(s)}{T \cdot \epsilon} \rightarrow \left(H^TH\right)^{-1}H^T\mathbf{u}.$$

Hence the theorem is proved. □

Proof of theorem 7.1

Recall that we have defined:

$$|(\frac{\nabla L(\beta^{(\epsilon)}(t))}{\nabla J(\beta^{(\epsilon)}(t))})_j + \lambda_t| = e_{tj} \tag{A.16}$$

Now we define two "regularity constants":

$$M_1 = p\frac{\max_{\beta,j} \|(\nabla^3 L(\beta))_{j,\cdot,\cdot}\| + \lambda_{max} \max_{\beta,j} \|\nabla^3 J(\beta))_{j,\cdot,\cdot}\|}{\lambda_0 \min_{\beta,j} |(\nabla J(\beta))_j|} \tag{A.17}$$

$$M_2 = \max_{\beta,j}(\nabla J(\beta))_j^2 \cdot \max_{\beta,\lambda} \|[\nabla^2 L(\beta) + \lambda \nabla^2 J(\beta)]^{-1}\|^2 \tag{A.18}$$

We also define α_{t+1} as the direction algorithm 7.1 takes in step t, i.e.

$$\alpha_{t+1} = (\nabla^2 L(\beta^{(\epsilon)}(t)) + \lambda_{t+1} \nabla^2 J(\beta^{(\epsilon)}(t)))^{-1} \cdot [\nabla L(\beta^{(\epsilon)}(t)) + \lambda_{t+1} \nabla J(\beta^{(\epsilon)}(t))] \tag{A.19}$$

Proof of lemma 7.2 For the sequence $\beta_t^{(\epsilon)}$, defined in algorithm 7.1, denote:

$$\begin{aligned}
\mathbf{d}(t) &= \{\frac{\partial L(\beta^{(\epsilon)}(t))}{\partial \beta_j}\}_{j=1}^p \\
D(t) &= \{\frac{\partial^2 L(\beta^{(\epsilon)}(t))}{\partial \beta_j \partial \beta_l}\}_{j,l=1}^p \\
\mathcal{D}(t) &= \{\frac{\partial^3 L(\beta^{(\epsilon)}(t))}{\partial \beta_j \partial \beta_l \partial \beta_m}\}_{j,l,m=1}^p \\
\mathbf{c}(t) &= \{\frac{\partial J(\beta^{(\epsilon)}(t))}{\partial \beta_j}\}_{j=1}^p \\
C(t) &= \{\frac{\partial^2 J(\beta^{(\epsilon)}(t))}{\partial \beta_j \partial \beta_l}\}_{j,l=1}^p \\
\mathcal{C}(t) &= \{\frac{\partial^3 J(\beta^{(\epsilon)}(t))}{\partial \beta_j \partial \beta_l \partial \beta_m}\}_{j,l,m=1}^p
\end{aligned}$$

Fix t and define

$$\mathbf{a} = \mathbf{d}(t-1) + D(t-1)_{\cdot,\cdot}\alpha_t$$
$$\mathbf{b} = \mathbf{c}(t-1) + C(t-1)_{\cdot,\cdot}\alpha_t$$

Then it is easy to verify that $\frac{a_j}{b_j} = -\lambda_t \ , \ \ \forall j$.

By building separate Taylor series for numerator and denominator we can write:

$$\begin{aligned} \frac{\mathbf{d}_j(t)}{\mathbf{c}_j(t)} &= \frac{\mathbf{d}_j(t-1) + D(t-1)_{j,\cdot}\alpha_t + \alpha_t^t \mathcal{D}(\beta^*)_{j,\cdot,\cdot}\alpha_t}{\mathbf{c}_j(t-1) + C(t-1)_{j,\cdot}\alpha_t + \alpha_t^t \mathcal{C}(\tilde{\beta})_{j,\cdot,\cdot}\alpha_t} = \\ &= \frac{a_j + \alpha_t^t \mathcal{D}(\beta^*)_{j,\cdot,\cdot}\alpha_t}{b_j + \alpha_t^t \mathcal{C}(\tilde{\beta})_{j,\cdot,\cdot}\alpha_t} \end{aligned} \tag{A.20}$$

Where β^* and $\tilde{\beta}$ are the "intermediate" values which give equality. Developing (A.20) further we get:

$$\begin{aligned} \frac{\mathbf{d}_j(t)}{\mathbf{c}_j(t)} &= \frac{a_j}{b_j}(1 + \frac{b_j \alpha_t^t \mathcal{D}(\beta^*)_{j,\cdot,\cdot}\alpha_t - a_j \alpha_t^t \mathcal{C}(\tilde{\beta})_{j,\cdot,\cdot}\alpha_t}{a_j b_j + a_j \alpha_t^t \mathcal{C}(\tilde{\beta})_{j,\cdot,\cdot}\alpha_t}) = \\ &= \frac{a_j}{b_j} + \frac{b_j \alpha_t^t \mathcal{D}(\beta^*)_{j,\cdot,\cdot}\alpha_t - a_j \alpha_t^t \mathcal{C}(\tilde{\beta})_{j,\cdot,\cdot}\alpha_t}{a_j(b_j + \alpha_t^t \mathcal{C}(\tilde{\beta})_{j,\cdot,\cdot}\alpha_t)} = \\ &= -\lambda_t - \frac{\alpha_t^t \mathcal{D}(\beta^*)_{j,\cdot,\cdot}\alpha_t + \lambda_t \alpha_t^t \mathcal{C}(\tilde{\beta})_{j,\cdot,\cdot}\alpha_t}{\lambda_t c_j(t)} = \\ &= -\lambda_t - \alpha_t^t \frac{\mathcal{D}(\beta^*)_{j,\cdot,\cdot} + \lambda_t \mathcal{C}(\tilde{\beta})_{j,\cdot,\cdot}}{\lambda_t c_j(t)} \alpha_t \end{aligned} \tag{A.21}$$

And from combining (A.21) and (A.17) we get:

$$\|\mathbf{e}_t\| \leq \|\alpha_t\|^2 M_1 \tag{A.22}$$

To complete the proof we need to bound $\|\alpha_t\|^2$. Note that in our notation:

$$\begin{aligned} \alpha_t &= (D(t-1) + \lambda_t C(t-1))^{-1}(\lambda_t \mathbf{c}(t-1) + \mathbf{d}(t-1) = \\ &= (D(t-1) + \lambda_t C(t-1))^{-1}\mathbf{c}(t-1)(\lambda_{t-1} + \frac{\mathbf{d}(t-1)}{\mathbf{c}(t-1)} + \epsilon) = \\ &= (D(t-1) + \lambda_t C(t-1))^{-1}\mathbf{c}(t-1)(\mathbf{e}_{t-1} + \epsilon) \end{aligned} \tag{A.23}$$

note that the notation $\mathbf{e}_{t-1} + \epsilon$ is used for adding ϵ to *each* element of $\mathbf{e}_{t-1}$.

Consequently:

$$\|\alpha_t\|^2 \leq \|e_{t-1} + \epsilon\|^2 M_2 \tag{A.24}$$

And from (A.22) and (A.24) we get that:

$$\|\mathbf{e}_t\| \leq \|\mathbf{e}_{t-1} + \epsilon\|^2 M_1 M_2 \tag{A.25}$$

Now we observe:

$$\begin{aligned} \|\mathbf{e}_{t-1} + \epsilon\|^2 &= \|e_{t-1}\|^2 + 2\epsilon \sum_i e_{t-1,i} + m\epsilon^2 \leq \quad \text{(Jensen ineq.)} \\ &\leq \|\mathbf{e}_{t-1}\|^2 + 2\sqrt{m} \cdot \epsilon \|\mathbf{e}_{t-1} + m\epsilon^2 = (\|\mathbf{e}_{t-1}\| + \sqrt{m} \cdot \epsilon)^2 \end{aligned} \tag{A.26}$$

Since $\mathbf{e}_0 = 0$ and using (A.25) and (A.26) we thus get (with $M = M_1 M_2$):

$$\begin{aligned} &\|\mathbf{e}_1\| \leq Mm\epsilon^2 = u_1 \\ &\|\mathbf{e}_t\| \leq M(\|\mathbf{e}_{t-1}\| + \sqrt{m} \cdot \epsilon)^2 \leq u_t \end{aligned} \tag{A.27}$$

Thus lemma 7.2 is proven. □

Proof of lemma 7.3: We have $u_1 = mM\epsilon^2$, $u_t = M(u_{t-1} + \sqrt{m} \cdot \epsilon)^2$. It's easy to verify that u_t is an increasing series. To find $lim_t u_t$ we just need to solve the fixed point equation:

$$u = M(u + \sqrt{m} \cdot \epsilon)^2 \tag{A.28}$$

and find the smallest solution that is bigger than $u_1 = Mm\epsilon$.

Denote $\delta = \sqrt{m} \cdot \epsilon$. Then solving (A.28) gives:

$$u = \frac{1}{2M} - \delta \pm \frac{\sqrt{1 - 4\delta M}}{2M} \tag{A.29}$$

the solution is real if $4\delta M < 1$, and we can write:

$$\sqrt{1 - 4\delta M} = 1 + 2\delta M - 8\delta^2 M^2 + O(\delta^3) \tag{A.30}$$

which gives us that

$$\frac{1}{2M} - \delta - \frac{\sqrt{1 - 4\delta M}}{2M} = 4M\delta^2 + O(\delta^3) \tag{A.31}$$

is the fixed point we are looking for, if δ (or equivalently $\epsilon = \delta/\sqrt{m}$ is small enough.

□

Proof of lemma 7.4: A sufficient condition for this lemma to hold is a "Lipshitz" condition:

$$\|\frac{\nabla L(\beta^*)}{\nabla J(\beta^*)} - \frac{\nabla L(\tilde{\beta})}{\nabla J(\tilde{\beta})}\| \leq \delta \;\Rightarrow\; \|\beta^* - \tilde{\beta}\| \leq C\delta \tag{A.32}$$

This condition doesn't have to hold for every β but rather only in the "neighborhood" of the optimal solution path $\hat{\beta}(\lambda)$. It can easily be translated to boundedness conditions on the second derivatives of L and J in this neighborhood. In particular, if $\|\nabla^2 L\|$ is bounded from above and $\|\nabla^2 J\|$ is bounded from below in the relevant neighborhood then condition (A.32) holds.

Since we know that $\frac{\nabla L(\hat{\beta}(\lambda))}{\nabla J(\hat{\beta}(\lambda))} = \lambda$ (eq 7.2) and using lemma 7.3, we get that (A.32) is equivalent to:

$$\|\mathbf{e}_t\| = O(\epsilon^2) \;\Rightarrow\; \|\beta^{(\epsilon)}(c/\epsilon) - \hat{\beta}(\lambda_0 + c)\| = O(\epsilon^2) \tag{A.33}$$

□

Required regularity conditions

For full rigorosity of theorem 7.1 we need to figure out the exact regularity conditions required for all "constants" to be finite and the implications about the loss functions, data and ranges of regularization parameter values, where theorem 7.1 holds.

Consider the two constants M_1, M_2 defined in (A.17), (A.18) respectively. First note that λ_0 of algorithm 7.1 appears in the denominator of M_1 and therefore our theorem does not hold is the algorithm starts from $\lambda_0 = 0$, the non-penalized solution. Rather, we should start by finding an exact regularized solution for some (potentially tiny) value of λ_0. In the examples in chapter 7, however, we started from $\lambda_0 = 0$ and still got the desired behavior.

Next, let us make the following very mild assumptions:

1. J is "reasonable" in the sense that it penalizes more for bigger $\|\beta\|$ in some weak sense. For example, assume $\exists c$ such that $J(\beta) \leq c\|\beta\|$.

2. L and J are convex 3-times continuously differentiable functions on the compact domain $J(\beta) \leq J(\hat{\beta})$

Given these assumptions we are guaranteed that $\hat{\beta}(\lambda)$ is in this compact domain and that *all* relevant derivatives are bounded from above. Thus if we look at the "numerator" quantities in (A.17), (A.18) they are all guranteed to be finite. This also applies to interesting non-differentiable cases, such as absolute or hinge loss, Huber's loss and the lasso penalty.

We also need to bound the "denominator" quantity $\min_{\beta,j} |(\nabla J(\beta))_j|$ from below. For any reasonable penalty the other side of assumption 1 above also holds, i.e. $\exists c$ s.t. $\|J\| > \epsilon \Rightarrow J(\beta) > c\epsilon$. Thus bounding $|(\nabla J(\beta))|$ from below is equivalent to bounding $\|\beta\|$ from below. In other words, we are on safe ground as long as we bound λ from above in algorithm 7.1 and consequently don't let $\beta_t^{(\epsilon)}$ get close to 0.

The other "denominator" quantity, $\max_{\beta,\lambda} \|[\nabla^2 L(\beta) + \lambda \nabla^2 J(\beta)]^{-1}\|^2$ is somewhat more problematic if we want a general theory. Instead, we consider here some popular choices for L and J and illustrate that this expression is bounded for these.

- Ridge- or lasso-penalized squared error loss. We have $\nabla^2 L = X^T X \succ 0$ and $\nabla^2 J = I$ for ridge, $\nabla J = 0$ for lasso, all independent of β. Since $\lambda \geq 0$ we get:

$$\max_{\beta,\lambda} \|[\nabla^2 L(\beta) + \lambda \nabla^2 J(\beta)]^{-1}\|^2 = \|(X^T X)^{-1}\|^2$$

- Logistic or exponential loss with lasso/ridge penalty. It is easy to see that as long as $\|\beta\|$ is bounded from above, then $\|\nabla^2 L(\beta)\|$ is bounded from below (the second derivatives converge to 0 as the margins become ∞ for exponential and as the margins become $\pm\infty$ for logistic. Thus by convexity and the same arguments as before, with either the lasso or ridge penalty we are guaranteed that $\|[\nabla^2 L(\beta) + \lambda \nabla^2 J(\beta)]^{-1}\|^2$ is bounded.

Bibliography

Bartlett, P., Jordan, M. & McAuliffe, J. (2003), Convexity, classification and risk bounds, Technical report, dept. of Statistics, UC Berkeley.

Bauer, E. & Kohavi, R. (1999), 'An empirical comparison of voting classification algorithms: bagging, boosting and variants', *Machine Learning* **36**(1/2), 105–139.

Bishop, C. M. (1995), *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford, U.K.

Blake, C. & Merz, C. (1998), Repository of machine learning databases. [http://www.ics.uci.edu/ mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

Breiman, L. (1996), 'Bagging predictors', *MLJ* **24**(2), 123–140.

Breiman, L. (1998), 'Arcing classifiers', *AS* **26**(3), 801:849.

Breiman, L. (1999), 'Prediction games and arcing algorithms', *Neural Computation* **11**(7), 1493–1517.

Breiman, L., Friedman, J., Olshen, R. & Stone, C. (1984), *Classification and Regression Trees*, Wardsworth International Group.

Buhlmann, P. & Yu, B. (2003), 'Analyzing bagging', *AS* .

Cheeseman, P. & Stutz, J. (1995), *Bayesian Classification (AutoClass): Theory and Results*, AAAI Press.

Collins, M., Schapire, R. E. & Singer, Y. (2000), Logistic regression, adaboost and bregman distances, *in* 'Computational Learing Theory', pp. 158–169.

Donoho, D. L., Johnstone, I. M., Kerkyacharian, G. & Picard, D. (1995), 'Wavelet shrinkage: Asymptopia?', *J. R. Statist. Soc. B.* **57**(2), 301–337.

Duda, R. O. & Hart, P. E. (1973), *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York.

Efron, B., Hastie, T., Johnstone, I. & Tibshirani, R. (2002), Least angle regression, Technical report, Department of Statistics, Stanford University.

Freund, Y. & Scahpire, R. (n.d.), Experiments with a new boosting algorithm, *in* 'ICML-96'.

Freund, Y. & Schapire, R. (1995), A decision-theoretic generalization of on-line learning and an application to boosting, *in* 'European Conference on Computational Learning Theory', pp. 23–37.

Friedman, H., Hastie, T., Rosset, S., Tibshirani, R. & Zhu, J. (2003), 'Discussion of 3 boosting papers', *AS* . To appear.

Friedman, J. (2001), 'Greedy function approximation: A gradient boosting machine', *Annals of Statistics* **29**(5).

Friedman, J. H. & Hall, P. (2000), On bagging and nonlinear estimation. preprint.

Friedman, J., Hastie, T. & Tibshirani, R. (2000), 'Additive logistic regression: a statistical view of boosting', *Annals of Statistics* **28**, 337–407.

Friedman, N. & Koller, D. (2002), 'Being bayesian about network structure: A bayesian approach to structure discovery in bayesian networks', *Machine Learning Journal* .

Gasch, A., Spellman, P., Kao, C., O.Carmel-Harel, Eisen, M., G.Storz, D.Botstein & Brown, P. (2000), 'Genomic expression program in the response of yeast cells to environmental changes', *Mol. Bio. Cell* **11**, 4241–4257.

Grove, A. & Schuurmans, D. (1998), Boosting in the limit: Maximizing the margin of learned ensembles, *in* 'Proc. of 15th National Conf. on AI'.

Hastie, T., Tibshirani, T. & Friedman, J. (2001), *Elements of Statistical Learning*, Springer-Verlag, New York.

Heckerman, D. (1998), A tutorial on learning with Bayesian networks, *in* M. I. Jordan, ed., 'Learning in Graphical Models', MIT Press, Cambridge, MA.

Huber, P. (1964), 'Robust estimation of a location parameter', *AMS* **35**(1), 73:101.

Hughes, T. & al. (2000), 'Functional discovery via a compendium of expression profiles', *Cell* **102**(1), 109–26.

KEGG: Kyoto Encyclopedia of Genes and Genomes (1998), *in* 'http://www.genome.ad.jp/kegg'.

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W. & Jackel, L. (1990), Handwritten digit recognition with a back-propagation network, *in* 'Advances in Neural Information Processing Systems 2'.

Mangasarian, O. (1999), 'Arbitrary-norm separating plane', *Operations Research Letters* **24(1–2)**, 15–23.

Mason, L., Baxter, J., Bartlett, P. & Frean, M. (1999), Boosting algorithms as gradient descent, *in* 'Neural Information Processing Systems', Vol. 12.

Meila, M. & Jaakkola, T. (2000), Tractable bayesian learning of tree belief networks, Technical Report CMU-RI-TR-00-15, Robotics institute, Carnegie Mellon University.

Pe'er, D., Regev, A., Elidan, G. & Friedman, N. (2001), Inferring subnetworks from perturbed expression profiles, *in* 'ISMB'01'.

Quinlan, J. (1993), *C4.5 - Programs for Machine Learning*, Morgan-Kaufmann.

Rätsch, G., Mika, S. & Warmuth, M. (2001), On the convergence of leveraging, NeuroCOLT2 Technical Report 98, Royal Holloway College, London.

Rätsch, G., Onoda, T. & Müller, K.-R. (2001), 'Soft margins for AdaBoost', *Machine Learning* **42**(3), 287–320.

Rätsch, G. & Warmuth, M. (2002), Efficient margin maximization with boosting. submitted to JMLR.

Rosset, S. & Segal, E. (2002), Boosting density estimation. NIPS-2002, to appear.

Rosset, S., Zhu, J. & Hastie, T. (2003), Boosting as a regularized path to a maximum margin classifier, Technical report, Department of Statistics, Stanford University.

Schapire, R. E., Freund, Y., Bartlett, P. & Lee, W. (1998), 'Boosting the margin: a new explanation for the effectiveness of voting methods', *Annals of Statistics* **26**, 1651–1686.

Spellman, P. T. & al. (1998), 'Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization', *Mol. Biol. Cell* **9**(12), 3273–97.

Thiesson, B., Meek, C. & Heckerman, D. (1997), Learning mixtures of dag models, Technical Report MSR-TR-98-12, Microsoft Research.

Tibshirani (1996), 'Regression shrinkage and selection via the lasso', *Journal of the Royal Statistical Society Series B* **58**(1), 267–288.

Tibshrani, R., Saunders, M., Rosset, S. & Zhu, J. (2003), Sparsity and smoothness via the fused lasso. In preparation.

Vapnik, V. (1995), *The Nature of Statistical Learning Theory*, Springer.

Weston, J. & Watkins, C. (1998), Multi-class support vector machines, Technical Report CSD-TR-98-04, dept of CS, Royal Holloway, University of London.

Zemel, R. & Pitassi, T. (2001), A gradient-based boosting algorithm for regression problems, *in* 'Proc. NIPS'.

Zhu, J. R. S., Hastie, T. & Tibshirani, R. (2003), 1-norm support vector machines. Submitted.