High-dimensional regression problems with special structure

Author(s):
Meier, Lukas D.

Publication Date:
2008

Permanent Link:
https://doi.org/10.3929/ethz-a-005784712

Rights / License:
In Copyright - Non-Commercial Use Permitted
High-Dimensional Regression Problems with Special Structure

A dissertation submitted to
ETH ZURICH

for the degree of
Doctor of Sciences

presented by
LUKAS DIETER MEIER
Dipl. Math. ETH
born August 9, 1979
citizen of Unterkulm AG

accepted on the recommendation of
Prof. Dr. Peter Bühlmann, examiner
Prof. Dr. Sara van de Geer, co-examiner

2008
Acknowledgements

First, a very big thank you to Prof. Peter Bühlmann for supervising this thesis. He followed my work with great interest and was always there for help, advice, motivation and support of all kind while giving me the freedom to try my own steps in the world of statistical research. Not only in the office, but also while having black forest cake at Oberwolfach or while drinking a Singapore Sling at 1 a.m. at the waterside of Singapore river, I could always count on his support and interest.

I would also like to thank him for encouraging me to travel. Be it an arctic mission to Harvard, a late summer stay in Oberwolfach or the attendance of many international conferences, I had the chance to meet various researchers around the world and learned a lot, also about myself.

I also thank Prof. Sara van de Geer for acting as co-examiner, for the nice and fruitful collaborations and for her support.

Further thanks go to my friends and colleagues at the Seminar für Statistik for the very nice working atmosphere and for always providing an expert for the most exotic problems. We also had a lot of fun in our leisure time: Be it kayaking the Everglades with alligators, diving with sharks (or sea snakes), riding the world’s fastest and highest roller coaster, living in a very dangerous district of Philadelphia (which we missed by a good margin...), enjoying a chocolate fountain high-above Manhattan, relaxing on a nice bed at an altitude of 30’000 feet or simply climbing on a volcano during the night: We did it all!

Finally, I would like to thank my family (including a little big dog named “chica”), my friends and especially Jeannine for the unconditional support and for the patience during all the weeks when I was not around.
Contents

Abstract xi

Zusammenfassung xv

1 Introduction 1

1.1 $\ell_1$-Penalized Estimation 2

1.1.1 The Lasso Estimator 2

1.1.2 Algorithms 4

1.2 Group Lasso 5

1.3 Thesis Overview 7

2 The Group Lasso for Logistic Regression 9

2.1 Introduction 10

2.2 Logistic Group Lasso 12

2.2.1 Model Setup 12

2.2.2 Algorithms for the Logistic Group Lasso 13

2.2.3 Comparison with Other Algorithms 19
2.2.4 Consistency .............................................. 22
2.3 Logistic Group Lasso/Ridge Hybrid ......................... 25
  2.3.1 General Case ......................................... 25
  2.3.2 Restriction to Hierarchical Models .................... 26
2.4 Simulation .................................................. 27
2.5 Application to Splice Site Detection ......................... 29
  2.5.1 Dataset ................................................ 29
  2.5.2 Procedure ............................................ 31
  2.5.3 Results ................................................ 32
2.6 Discussion ................................................. 32
2.A Appendix .................................................... 34
  2.A.1 Proof of Lemma 2.1 .................................... 34
  2.A.2 Proof of Proposition 2.1 ................................ 34
  2.A.3 Proof of Proposition 2.2 ................................ 35
  2.A.4 Outline of the Proof of the Consistency Result ... 35

3 Smoothing $\ell_1$-Penalized Estimators for High-Dimensional Time-Course Data 37
  3.1 Introduction .............................................. 38
  3.2 Smoothed Lasso .......................................... 39
    3.2.1 Definition .......................................... 39
    3.2.2 Orthogonal Case ..................................... 41
    3.2.3 General Case ......................................... 42
  3.3 Smoothed Adaptive Lasso .................................. 44
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3.1 Choice of Initial Estimator</td>
<td>45</td>
</tr>
<tr>
<td>3.4 Simulations</td>
<td>47</td>
</tr>
<tr>
<td>3.4.1 Design</td>
<td>47</td>
</tr>
<tr>
<td>3.4.2 Performance Measures</td>
<td>49</td>
</tr>
<tr>
<td>3.4.3 Results</td>
<td>49</td>
</tr>
<tr>
<td>3.5 Real Data: Motif Finding in DNA Sequences</td>
<td>50</td>
</tr>
<tr>
<td>3.6 Discussion</td>
<td>53</td>
</tr>
<tr>
<td>3.7 Appendix</td>
<td>54</td>
</tr>
<tr>
<td>3.7.1 Regularity Conditions</td>
<td>54</td>
</tr>
<tr>
<td>3.7.2 Proofs</td>
<td>55</td>
</tr>
<tr>
<td>4 High-Dimensional Additive Modeling</td>
<td>61</td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>61</td>
</tr>
<tr>
<td>4.2 Penalized Maximum Likelihood for Additive Models</td>
<td>62</td>
</tr>
<tr>
<td>4.2.1 The Sparsity-Smoothness Penalty</td>
<td>63</td>
</tr>
<tr>
<td>4.2.2 Computational Algorithm</td>
<td>64</td>
</tr>
<tr>
<td>4.2.3 Oracle Results</td>
<td>66</td>
</tr>
<tr>
<td>4.3 Numerical Examples</td>
<td>72</td>
</tr>
<tr>
<td>4.3.1 Simulations</td>
<td>72</td>
</tr>
<tr>
<td>4.3.2 Real Data</td>
<td>76</td>
</tr>
<tr>
<td>4.4 Extensions</td>
<td>78</td>
</tr>
<tr>
<td>4.4.1 Generalized Additive Models</td>
<td>78</td>
</tr>
<tr>
<td>4.4.2 Adaptivity</td>
<td>80</td>
</tr>
<tr>
<td>4.5 Conclusions</td>
<td>81</td>
</tr>
</tbody>
</table>
4.A Appendix .................................................. 82
    4.A.1 Proof of Proposition 4.1 .......................... 82
    4.A.2 Proof of Proposition 4.2 .......................... 82

5 P-Values for High-Dimensional Regression 85
    5.1 Introduction .......................................... 86
    5.2 Sample-Splitting and High-Dimensional Variable Selection 87
        5.2.1 The Single-Split Method .......................... 87
        5.2.2 The New Multi-Split Method ....................... 88
    5.3 Error Control and Consistency ........................ 91
        5.3.1 Assumptions ...................................... 91
        5.3.2 Error Control ..................................... 92
        5.3.3 Model Selection Consistency ........................ 93
    5.4 Numerical Results ..................................... 94
        5.4.1 Simulations ....................................... 94
        5.4.2 Comparisons with Adaptive Lasso ................. 96
        5.4.3 Real Data ........................................ 99
    5.5 Extensions ............................................ 99
    5.6 Discussion ........................................... 101
    5.A Proofs ................................................ 101

6 Discussion of “One-step Sparse Estimates in Nonconcave Penalized Likelihood Models (H. Zou and R. Li)” 105
    6.1 Non-Convex Functions and Multi-Step Convex Optimization .......................... 106
6.1.1 Connection to Adaptive Lasso for Type 1 Penalty Functions ...................................... 107

6.2 Variable Selection in the High-Dimensional Case ........................................ 108

6.3 Beyond the One-Step Estimator ................................................................. 110

6.3.1 Small Simulation Study ............................................................................ 112

6.3.2 Real Data Example from Biology ......................................................... 113

Bibliography ........................................................................................................ 115

Curriculum Vitae .............................................................................................. 123
Abstract

High-dimensional variable selection has received tremendous attention in the last decade. Sparse estimators like the Lasso (Tibshirani, 1996) have proven to be powerful for high-dimensional settings, both from a theoretical and a practical point of view. In this thesis we propose several extensions of the Lasso (and of variants thereof) when either additional structure is available in the data or when more flexibility with respect to sparsity or functional modeling is of interest.

First, we consider the Group Lasso (Yuan and Lin, 2006). The Group Lasso is an extension of the Lasso to do variable selection on (predefined) groups of variables in linear regression models. The estimates have the attractive property of being invariant under groupwise orthogonal reparametrizations. We extend the Group Lasso to logistic regression models and present an efficient algorithm, especially suitable for high-dimensional problems, which can also be applied to generalized linear models to solve the corresponding convex optimization problem. The Group Lasso estimator for logistic regression is shown to be statistically consistent even if the number of predictors is much larger than sample size but with sparse true underlying structure. We further use a two-stage procedure which aims for sparser models than the Group Lasso, leading to improved prediction performance for some cases. Moreover, due to the two-stage nature, the estimates can be constructed to be hierarchical. The methods are used on simulated and real datasets about splice site detection in DNA sequences.

When a series of (related) linear models has to be estimated it is often appropriate to combine the different data-sets to construct more efficient estimators. We use $\ell_1$-penalized estimators like the Lasso or
the Adaptive Lasso (Zou, 2006) which can simultaneously do parameter estimation and model selection. We show that for a time-course of high-dimensional linear models the convergence rates of the Lasso and of the Adaptive Lasso can be improved by combining the different time-points in a suitable way. Moreover, the Adaptive Lasso still enjoys oracle properties and consistent variable selection. The finite sample properties of the proposed methods are illustrated on simulated data and on a real problem of motif finding in DNA sequences.

Often, the assumption of a linear relationship between the response and the predictors is too restrictive. We propose a new sparsity-smoothness penalty for high-dimensional generalized additive models. The combination of sparsity and smoothness is crucial for mathematical theory as well as performance for finite-sample data. We present a computationally efficient algorithm, with provable numerical convergence properties, for optimizing the penalized likelihood. Furthermore, we provide oracle results which yield asymptotic optimality of our estimator for high-dimensional but sparse additive models. Finally, an adaptive version of our sparsity-smoothness penalized approach yields large additional performance gains.

Assigning significance in high-dimensional regression is challenging. Most computationally efficient selection algorithms cannot guard against inclusion of noise variables. Asymptotically valid p-values are not available. An exception is a recent proposal by Wasserman and Roeder (2008) which splits the data into two parts. The number of variables is then reduced to a manageable size using the first split, while classical variable selection techniques can be applied to the remaining variables, using the data from the second split. This yields asymptotic error control under minimal conditions. It involves, however, a one-time random split of the data. Results are sensitive to this arbitrary choice: it amounts to a “p-value lottery” and makes it difficult to reproduce results. Here, we show that inference across multiple random splits can be aggregated, while keeping asymptotic control over the inclusion of noise variables. In addition, the proposed aggregation is shown to improve power, while reducing the number of falsely selected variables substantially.

The Adaptive Lasso (Zou, 2006) is a two-stage procedure. Based on an initial estimator, a weighted penalty function is constructed. The main idea is to penalize the more important variables less to remove
bias. We discuss the idea of successively applying such a weighted penalization scheme in order to get sparser solutions without losing a lot of prediction performance.
Zusammenfassung


P-Werte sind oft nicht verfügbar bei Verfahren für die Variablenwahl in hochdimensionalen Regressionsproblemen. In Wasserman and Roeder (2008) wird vorgeschlagen, dass man die Daten in zwei disjunkte Mengen aufteilen soll. Zuerst wird mit der ersten Teilmenge die Anzahl Variablen reduziert. Basierend auf der zweiten Teilmenge kann dann mit klassischen Verfahren die Variablenwahl verfeinert werden. Im Speziellen kann man für die im ersten Schritt gewählten Variablen p-Werte angeben und asymptotisch die Wahrscheinlichkeit kontrollieren, dass mindestens eine Variable fälschlicherweise ausgewählt wird, obwohl sie gar nicht im Modell ist. Um die Abhängigkeit von einer einzelnen Datenaufteilung zu reduzieren, führen wir obiges Verfahren mehrmals durch. Wir zeigen, dass man die daraus resultierenden p-Werte mittels Quantilen aggregieren kann, so dass die asymptotische Kontrolle über
die Fehlerrate gewährleistet bleibt.

Chapter 1

Introduction

Many scientific projects deal with the problem of analyzing the relationship between a response value \( Y \) and explanatory variables \( x^{(1)}, x^{(2)}, \ldots, x^{(p)} \). We can for example take the concentration of microparticles in the air at a location in Zurich as response \( Y \) and use wind speed as \( x^{(1)} \), temperature as \( x^{(2)} \) etc. Using a suitable model, we try to learn or estimate the relationship between the response \( Y \) and the predictors \( x^{(1)}, \ldots, x^{(p)} \) based on a training set of \( n \) observations. An estimator should have good prediction properties. For new data, the predicted values should be "close" to the true value. Often it is also of interest not to use all \( p \) predictors but only a small subset of them. The goal is to perform model or variable selection. Ideally, one would like to have a model which is as sparse as possible (without losing prediction capabilities). This principle is also called "Occam’s razor". Sparser models are much easier to interpret.

Nowadays, many data-sets are high-dimensional, i.e. the number of predictors \( p \) (greatly) exceeds the number of observations \( (p \gg n) \). For cancer research, the expression of thousands of genes of some dozen patients with different cancer types can be easily measured. In DNA sequence analysis where the relationship between some special words ("motifs") and the binding intensity of proteins to the DNA is of interest, the possible feature space is very large. If we consider 10 letter words (each letter is from the set \( \{A, C, G, T\} \)), there are already
$4^{10} = 1'048'576$ possible combinations.

For such kind of problems, classical approaches are either not applicable or not suitable because of the problem of over-fitting.

For regression models, the Lasso (Tibshirani, 1996) is a very powerful method which addresses the above mentioned problems. It can deal with the $p \gg n$ situation and it performs parameter estimation and model selection at the same time, leading to sparse and hence interpretable results.

### 1.1 $\ell_1$-Penalized Estimation

We would like to start with a short introduction about $\ell_1$-penalized estimators, their properties and the corresponding optimization algorithms.

#### 1.1.1 The Lasso Estimator

Consider the linear regression model

$$Y_i = \sum_{j=1}^{p} x_{i}^{(j)} \beta_j + \varepsilon_i, \ i = 1, \ldots, n,$$

where $\varepsilon_i$ are i.i.d. errors with mean zero and finite variance $\sigma^2$.

For suitably centered and scaled predictors, the Lasso estimator is defined as

$$\hat{\beta}_\lambda = \arg \min_{\beta} \|Y - X\beta\|^2_2 + \lambda \sum_{j=1}^{p} |\beta_j|, \quad (1.1.1)$$

where $Y$ is the (centered) vector of responses, $X$ is the $n \times p$ design matrix, $\beta$ is the parameter vector and $\|u\|^2_2 = \sum_{i=1}^{n} u_i^2$ for $u \in \mathbb{R}^n$. The tuning parameter $\lambda \geq 0$ controls the amount of penalization.

More generally, for any model with a log-likelihood function $\ell(\cdot)$ we can define

$$\hat{\beta}_\lambda = \arg \min_{\beta} -\ell(\beta) + \lambda \sum_{j=1}^{p} |\beta_j| \quad (1.1.2)$$
1.1. $\ell_1$-Penalized Estimation

This is in contrast to the Ridge regression estimator which is given by
\[
\hat{\beta}_\lambda = \arg \min_\beta -\ell(\beta) + \lambda \sum_{j=1}^{p} \beta_j^2.
\] (1.1.3)

Both Ridge and Lasso are shrinkage estimators. The estimates $\hat{\beta}_{\lambda,j}$ are shrunken towards zero. However, for Ridge regression, the estimates will typically not reach zero even for very large values of $\lambda$. This means that, in contrast to the Lasso estimator, no variable selection takes place. From a Bayesian point of view, the Ridge estimator is the MAP estimator when independent normal distributions are used for the $\beta_j$’s. The Lasso is the MAP estimator when independent double-exponential distributions are used as priors. The double-exponential distribution has much more mass around zero than the normal distribution.

Indeed, it is because of the special $\ell_1$-geometry, or more specifically because of the non-differentiability of the absolute value function at the origin, that the Lasso does variable selection, i.e. $\hat{\beta}_{\lambda,j} = 0$ for some large enough value of $\lambda$. If we define
\[
\lambda_{\text{max}} = \max_j \{ |\nabla_{\beta_j} \ell(\hat{\beta}_\lambda) ; \beta = 0 | \},
\]

it holds that $\hat{\beta}_{\lambda,j} = 0$ for all $j = 1, \ldots, p$ for any $\lambda \geq \lambda_{\text{max}}$.

More specifically, necessary and sufficient conditions for $\hat{\beta}_\lambda$ to be a solution of optimization problem (1.1.2) are
\[
-\nabla_{\beta_j} \ell(\hat{\beta}_\lambda) + \lambda \frac{\hat{\beta}_{\lambda,j}}{|\hat{\beta}_{\lambda,j}|} = 0 \text{ for } \hat{\beta}_{\lambda,j} \neq 0 \quad (1.1.4)
\]
\[
|\nabla_{\beta_j} \ell(\hat{\beta}_\lambda)| \leq \lambda \text{ for } \hat{\beta}_{\lambda,j} = 0. \quad (1.1.5)
\]

The Lasso has one tuning parameter: the penalization parameter $\lambda$. For any value of $\lambda \geq 0$ we get a parameter estimate $\hat{\beta}_\lambda$. Hence, we can draw a path (as a function of $\lambda$) for each of the $p$ coefficients. An example of the solution paths for a logistic regression model with 10 predictors (and an unpenalized intercept) is shown in Figure 1.1. Cross-validation is typically used to select a prediction optimal value of $\lambda$. 
Figure 1.1: *Lasso solution paths* $\{\hat{\beta}_{\lambda,j}\}_{\lambda \geq 0}$ for a logistic regression model with 10 predictors and an unpenalized intercept.

1.1.2 Algorithms

To compute the Lasso estimator for a given value of $\lambda$, one has to solve a convex optimization problem. For the special case of a linear regression model (1.1.1), the LARS-algorithm (Efron et al., 2004; Osborne et al., 2000a) computes the solutions for all values of $\lambda$ efficiently because of the piecewise linearity of the solution paths.

However, already for generalized linear models, there are no exact “path-following” algorithms. Approximate path-following approaches can be found in Rosset (2005) and Park and Hastie (2007).

Another approach is to use a predefined grid of values $\lambda_{\max} \geq \lambda_1 > \lambda_2 > \ldots > \lambda_K > 0$ (usually on the log-scale) and to solve the optimization problem (1.1.2) for every $\lambda_k, k = 1, \ldots, K$ using the solution corresponding to the previous grid value $\lambda_{k-1}$ as warm-start.

Recently, the focus has come back to coordinate-wise approaches to
solve the Lasso problem for a given value of \( \lambda \). Already at the end of the 1990’s, Fu (1998) has used such an approach for model (1.1.1). The idea is simple, very generic and powerful for sparse estimators. A coordinate-wise algorithm cycles through all coordinates, optimizing the current coordinate while keeping the remaining coordinates fixed. For model (1.1.1), the coordinate-updates are “easy” because closed form solutions are available (Fu, 1998). Theoretical results for numerical convergence for a broad variety of problems are e.g. available in Tseng (2001). Coordinate-descent is also possible for problem (1.1.2); however, no closed-form solutions exist in general.

1.2 Group Lasso

Another estimator which we will use later is the Group Lasso (Yuan and Lin, 2006; Bakin, 1999). It is useful if the predictors come “in groups” such as with factors. An example would be a high-dimensional ANOVA problem. Instead of selecting single (dummy) variables, the aim is to do variable selection on the “group-level”. Assume that we have \( G \) groups each having \( df_g \) degrees of freedom such that we can rewrite \( \beta = (\beta_1^T, \beta_2^T, \ldots, \beta_G^T)^T \), where \( \beta_g \in \mathbb{R}^{df_g} \) is the parameter vector of the \( g \)th group, \( g = 1, \ldots, G \). The Group Lasso estimator is then defined as

\[
\hat{\beta}_\lambda = \arg \min_{\beta} -\ell(\beta) + \lambda \sum_{g=1}^{G} \|\beta_g\|_2. \tag{1.2.1}
\]

If \( df_g = 1 \) for all \( g \), (1.2.1) is equivalent to (1.1.2). On the other side, if \( G = 1 \), the Group Lasso estimator (1.2.1) is equivalent to the Ridge estimator (1.1.3), but with a different penalty parameter \( \lambda \).

Analogously as in (1.1.4) and (1.1.5) we have the following (necessary and sufficient) optimality conditions for the Group Lasso estimator \( \hat{\beta}_\lambda \), see also Yuan and Lin (2006),

\[
-\nabla_{\beta_g} \ell(\hat{\beta}_\lambda) + \lambda \frac{\hat{\beta}_{\lambda,g}}{\|\hat{\beta}_{\lambda,g}\|_2} = 0 \quad \text{for} \quad \hat{\beta}_{\lambda,g} \neq 0 \tag{1.2.2}
\]

\[
\|\nabla_{\beta_g} \ell(\hat{\beta}_\lambda)\|_2 \leq \lambda \quad \text{for} \quad \hat{\beta}_{\lambda,g} = 0. \tag{1.2.3}
\]

These conditions can be derived using subdifferential calculus (Bertsekas, 2003). The subdifferential of the function \( f : \mathbb{R}^k \rightarrow \mathbb{R}, x \mapsto \|x\|_2 \)
at its only non-differentiable point $x = 0$ is the set \( \{ e \in \mathbb{R}^k, \| e \|_2 \leq 1 \} \).

As with the Lasso, one gets solution paths when varying $\lambda$. However, all coefficients of a group enter or leave the path at the same value of $\lambda$. An example of a solution path for a logistic regression model with 3 groups having size 2, 3 and 4 is depicted in Figure 1.2.

![Group Lasso solution paths](image)

**Figure 1.2:** Group Lasso solution paths \( \{ \hat{\beta}_{\lambda,j} \}_{\lambda \geq 0} \) for a logistic regression model with 3 predictors having 2, 3 and 4 degrees of freedom and an unpenalized intercept.

There are no exact path-following algorithms for the Group Lasso because the paths are not piecewise linear. But similar to the Lasso, (block) coordinate-wise approaches can be used to solve the optimization problem (1.2.1), cf. Yuan and Lin (2006) and Meier et al. (2008a).

We refer to Hesterberg et al. (2008) for a review of several Lasso and Group Lasso properties, extensions and their theoretical properties.
1.3 Thesis Overview

In this thesis we develop extensions of $\ell_1$-penalized regression when additional structure (such as grouping of the predictors or multiple response values) is available in the data. Often, more efficient estimators can be constructed when making use of the special data structure. Moreover, we propose new approaches which allow for more flexibility regarding functional modeling, sparsity and error control.

Chapter 2 is about the extension of the Group Lasso estimator to logistic regression models. We use a block-coordinate descent method to solve the corresponding convex optimization problem. Moreover, we propose a Group Lasso/Ridge hybrid method which aims for sparser models than the Group Lasso, leading to improved prediction performance for some cases. Chapter 2 is published in Meier et al. (2008a).

Chapter 3 is the paper Meier and Bühlmann (2007) which extends the Lasso to time-course data where several (related) response variables are available. More efficient estimators are constructed by using a weighted combination of the response values at neighbouring time-points.

Chapter 4 is the paper Meier et al. (2008b) about high-dimensional (generalized) additive modeling. In contrast to the Lasso estimator, we do not restrict ourselves to linear models, but use an additive modeling approach instead and do variable selection on the “function level”. We introduce a sparsity-smoothness penalty which can deal with a large number of basis functions in a way such that the resulting function estimates will be smooth.

In Chapter 5 we use multiple sample-splits to construct reproducible p-values in high-dimensional regression problems. The p-values corresponding to the different sample-splits are summarized by quantiles such that the family-wise error rate can be controlled at the desired level. The corresponding technical-report is Meinshausen et al. (2008).

Chapter 6 is an invited discussion of the paper Zou and Li (2008). It discusses the idea of successively applying an $\ell_1$-penalty with weights based on the previous estimator to get sparser solutions. The paper is published in Bühlmann and Meier (2008).
The Group Lasso for Logistic Regression

The Group Lasso is an extension of the Lasso to do variable selection on (predefined) groups of variables in linear regression models. The estimates have the attractive property of being invariant under group-wise orthogonal reparametrizations. We extend the Group Lasso to logistic regression models and present an efficient algorithm, especially suitable for high-dimensional problems, which can also be applied to generalized linear models to solve the corresponding convex optimization problem. The Group Lasso estimator for logistic regression is shown to be statistically consistent even if the number of predictors is much larger than sample size but with sparse true underlying structure. We further use a two-stage procedure which aims for sparser models than the Group Lasso, leading to improved prediction performance for some cases. Moreover, due to the two-stage nature, the estimates can be constructed to be hierarchical. The methods are used on simulated and real datasets about splice site detection in DNA sequences.
2.1 Introduction

The Lasso (Tibshirani, 1996), originally proposed for linear regression models, has become a popular model selection and shrinkage estimation method. In the usual linear regression setup we have a continuous response $Y \in \mathbb{R}^n$, an $n \times p$ design matrix $X$ and a parameter vector $\beta \in \mathbb{R}^p$. The Lasso estimator is then defined as

$$\hat{\beta}_\lambda = \arg\min_\beta \| Y - X\beta \|_2^2 + \lambda \sum_{j=1}^p |\beta_j|,$$

where $\|u\|_2^2 = \sum_{i=1}^n u_i^2$ for a vector $u \in \mathbb{R}^n$. For large values of the penalty parameter $\lambda$, some components of $\hat{\beta}_\lambda$ are set exactly to zero. The $\ell_1$-type penalty of the Lasso can also be applied to other models as for example Cox regression (Tibshirani, 1997), logistic regression (Lokhorst, 1999; Roth, 2004; Shevade and Keerthi, 2003; Genkin et al., 2007) or multinomial logistic regression (Krishnapuram et al., 2005) by replacing the residual sum of squares by the corresponding negative log-likelihood function.

Already for the special case in linear regression when not only continuous but also categorical predictors (factors) are present, the Lasso solution is not satisfactory as it only selects individual dummy variables instead of whole factors. Moreover, the Lasso solution depends on how the dummy variables are encoded. Choosing different contrasts for a categorical predictor will produce different solutions in general. The Group Lasso (Yuan and Lin, 2006; Bakin, 1999; Cai, 2001; Antoniadis and Fan, 2001) overcomes these problems by introducing a suitable extension of the Lasso penalty. The estimator is defined as

$$\hat{\beta}_\lambda = \arg\min_\beta \| Y - X\beta \|_2^2 + \lambda \sum_{g=1}^G \| \beta_{I_g} \|_2,$$

where $I_g$ is the index set belonging to the $g$th group of variables, $g = 1, \ldots, G$. This penalty can be viewed as an intermediate between the $\ell_1$- and $\ell_2$-type penalty. It has the attractive property that it does variable selection at the group level and is invariant under (groupwise) orthogonal transformations like Ridge regression (Yuan and Lin, 2006).

This chapter deals with the Group Lasso penalty for logistic regression models. The logistic case calls for new computational algorithms.
Kim et al. (2006) first studied the Group Lasso for logistic regression models and proposed a gradient descent algorithm to solve the corresponding constrained problem. We present methods which allow us to work directly on the penalized problem and whose convergence property does not depend on unknown constants as in Kim et al. (2006). Our algorithms are efficient in the sense that they can handle problems where $p$ and $n$ are large. Furthermore, they are also applicable for generalized linear models, beyond the case of logistic regression. We do not aim for an (approximate) path-following algorithm (Rosset, 2005; Zhao and Yu, 2007; Park and Hastie, 2007, 2006) but our approaches are fast enough for computing a whole range of solutions for varying penalty parameters on a (fixed) grid. Our approach is related to Genkin et al. (2007) which presents an impressively fast implementation (“the fastest”) for large-scale logistic regression with the Lasso; in fact, we can also deal with dimensionality $p$ in the tenths, but now for the Group Lasso. Moreover, we present an asymptotic consistency theory for the Group Lasso in high-dimensional problems where the predictor dimension is much larger than sample size. This has neither been developed for linear nor logistic regression. High-dimensionality of the predictor space arises in many applications, in particular with higher-order interaction terms or basis expansions for logistic additive models where the groups correspond to the basis functions for individual continuous covariates. Our application about the detection of splice sites, the regions between coding (exons) and non-coding (introns) DNA segments involves the categorical predictor space $\{A, C, G, T\}^7$ which has cardinality 16'384.

The rest of this chapter is organized as follows: In Section 2.2 we restate in more detail the idea of the Group Lasso for logistic regression models, present two efficient algorithms which are proven to solve the corresponding convex optimization problem and compare them with other optimization methods. Furthermore, we show that the Group Lasso estimator is statistically consistent for high-dimensional, sparse problems. In Section 2.3 we outline a two-stage procedure which often produces more adequate models both in terms of model size and prediction performance. Simulations follow in Section 2.4 and an application of the modeling of functional DNA sites can be found in Section 2.5. Section 2.6 contains the discussion. All proofs are given in the appendix.
2.2 Logistic Group Lasso

2.2.1 Model Setup

Assume we have i.i.d. observations \((x_i, y_i), i = 1, \ldots, n\) of a \(p\)-dimensional vector \(x_i \in \mathbb{R}^p\) of \(G\) predictors and a binary response variable \(y_i \in \{0, 1\}\). Both categorical and continuous predictors are allowed. We denote by \(d_{fg}\) the degrees of freedom of the \(g\)th predictor and can thus rewrite \(x_i = (x_{i,1}, \ldots, x_{i,G})^T\) with the group of variables \(x_{i,g} \in \mathbb{R}^{d_{fg}}, g = 1, \ldots, G\). For example, the main effect of a factor with 4 levels has \(df = 3\) while a continuous predictor involves \(df = 1\) only.

Linear logistic regression models the conditional probability \(p_{\beta}(x_i) = \mathbb{P}_{\beta}[Y = 1 \mid x_i]\) by

\[
\log \left\{ \frac{p_{\beta}(x_i)}{1 - p_{\beta}(x_i)} \right\} = \eta_{\beta}(x_i),
\]

with

\[
\eta_{\beta}(x_i) = \beta_0 + \sum_{g=1}^{G} x_{i,g} \beta_g,
\]

where \(\beta_0\) is the intercept and \(\beta_g \in \mathbb{R}^{d_{fg}}\) is the parameter vector corresponding to the \(g\)th predictor. We denote by \(\beta \in \mathbb{R}^{p+1}\) the whole parameter vector, i.e. \(\beta = (\beta_0, \beta_1^T, \ldots, \beta_G^T)^T\).

The Logistic Group Lasso estimator \(\hat{\beta}_\lambda\) is given by the minimizer of the convex function

\[
S_\lambda(\beta) = -\ell(\beta) + \lambda \sum_{g=1}^{G} s(d_{fg}) \|\beta_g\|_2,
\]

where \(\ell(\cdot)\) is the log-likelihood function, i.e.

\[
\ell(\beta) = \sum_{i=1}^{n} y_i \eta_{\beta}(x_i) - \log[1 + \exp\{\eta_{\beta}(x_i)\}].
\]

The tuning parameter \(\lambda \geq 0\) controls the amount of penalization. Note that we do not penalize the intercept. However, as shown in Lemma 2.1, the minimum in (2.2.2) is attained. The function \(s(\cdot)\) is used to
rescale the penalty with respect to the dimensionality of the parameter vector $\beta_g$. Unless stated otherwise, we use $s(df_g) = \sqrt{df_g}$ to ensure that the penalty term is of the order of the number of parameters $df_g$. The same rescaling is used in Yuan and Lin (2006).

**Lemma 2.1.** Assume that $0 < \sum_{i=1}^{n} y_i < n$. For $\lambda > 0$ and $s(d) > 0$ for all $d \in \mathbb{N}$, the minimum in optimization problem (2.2.2) is attained.

The first condition in Lemma 2.1 is a minimal requirement for the observed data. If the design matrix $X$ has full rank, the minimizer of $S_\lambda(\cdot)$ is unique. Otherwise, the set of minimizers is a convex set whose elements correspond to the same minimum value of $S_\lambda(\cdot)$.

The “groupwise” $\ell_2$-norm in (2.2.2) is an intermediate between the Lasso and the Ridge penalty function. It encourages that in general either $\hat{\beta}_g = 0$ or $\hat{\beta}_{g,j} \neq 0$ for all $j \in \{1, \ldots, df_g\}$, where we have omitted the index $\lambda$ for easier notation. A geometrical interpretation of this special sparsity property is given in Yuan and Lin (2006). An example of a solution path $\{\hat{\beta}_\lambda\}_{\lambda \geq 0}$ for a model consisting of an intercept and two factors having 3 degrees of freedom each is depicted in Figure 2.1.

Let the $n \times df_g$ matrix $X_g$ be the columns of the design matrix corresponding to the $g$th predictor. If we assume that the block matrices $X_g$ are of full rank, we can perform a (blockwise) orthonormalization – e.g. by a QR-decomposition – to get $X_g^T X_g = I_{df_g}$, $g = 1, \ldots, G$. Using such a design matrix, the Group Lasso estimator does not depend on the encoding scheme of the dummy variables. We choose a rescaled version $X_g^T X_g = n \cdot I_{df_g}$ to ensure that the parameter estimates are on the same scale when varying the sample size $n$. After parameter estimation, the estimates have to be transformed back in order to correspond to the original encoding.

### 2.2.2 Algorithms for the Logistic Group Lasso

**Block Coordinate Descent**

Parameter estimation is computationally more demanding than for linear regression models. The algorithm presented in Yuan and Lin (2006) sequentially solves a system of (necessary and sufficient) non-linear
Chapter 2. The Group Lasso for Logistic Regression

Figure 2.1: Solution path $\{\hat{\beta}_\lambda\}_{\lambda \geq 0}$ for a model consisting of an intercept (dotted line) and two factors having 3 degrees of freedom each (dashed and solid lines, respectively). $\lambda_{\text{max}}$ is the value of the penalty parameter $\lambda$ such that no penalized group is active in the model.

equations which corresponds to a groupwise minimization of the penalized residual sum of squares. Hence, the algorithm is a special case of a block coordinate descent algorithm. No result on numerical convergence is given in Yuan and Lin (2006).

For the more difficult case of logistic regression, we can also use a block coordinate descent algorithm and we prove numerical convergence by using the results of Tseng (2001) as shown in Proposition 2.1. The key lies in the separable structure of the non-differentiable part in $S_\lambda(\cdot)$. These properties of course also apply for the Group Lasso for linear and generalized linear regression models.

We cycle through the parameter groups and minimize the objective function $S_\lambda(\cdot)$, keeping all but the current parameter group fixed. This leads us to the algorithm presented in Table 2.1, where we denote by $\beta_{-g}$
2.2. Logistic Group Lasso

<table>
<thead>
<tr>
<th>Block Coordinate Descent Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Let $\beta \in \mathbb{R}^{p+1}$ be an initial parameter vector.</td>
</tr>
<tr>
<td>(2) $\beta_0 \leftarrow \text{arg min}<em>{\beta_0} S</em>\lambda(\beta)$</td>
</tr>
<tr>
<td>(3) for $g = 1, \ldots, G$</td>
</tr>
<tr>
<td>\quad if $|X_g^T(y - p\beta_{-g})|_2 \leq \lambda s(df_g)$</td>
</tr>
<tr>
<td>\quad $\beta_g \leftarrow 0$</td>
</tr>
<tr>
<td>\quad else</td>
</tr>
<tr>
<td>\quad $\beta_g \leftarrow \text{arg min}<em>{\beta_g} S</em>\lambda(\beta)$</td>
</tr>
<tr>
<td>\quad end</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>(4) Repeat steps (2)–(3) until some convergence criterion is met.</td>
</tr>
</tbody>
</table>

Table 2.1: Logistic Group Lasso Algorithm using Block Coordinate Descent Minimization.

The parameter vector $\beta$ when setting $\beta_g$ to 0 while all other components remain unchanged. In step (3) we first check whether the minimum is at the non-differentiable point $\beta_g = 0$. If not, we can use a standard numerical minimizer, e.g. a Newton type algorithm, to find the optimal solution with respect to $\beta_g$. In such a case the values of the last iteration can be used as starting values to save computing time. If the group was not in the model in the last iteration, we first go a small step in the opposite direction of the gradient of the negative log-likelihood function to ensure that we start at a differentiable point.

**Proposition 2.1.** Steps (2) and (3) of the block coordinate descent algorithm perform groupwise minimizations of $S_\lambda(\cdot)$ and are well defined in the sense that the corresponding minima are attained. Furthermore, if we denote by $\hat{\beta}^{(t)}$ the parameter vector after $t$ block updates, then every limit point of the sequence $\{\hat{\beta}^{(t)}\}_{t \geq 0}$ is a minimum point of $S_\lambda(\cdot)$.

Because the iterates can be shown to stay in a compact set, the existence of a limit point is guaranteed.

The main drawback of such an algorithm is that the blockwise minimizations of the active groups have to be performed numerically. However, for small and moderate sized problems in the dimension $p$ and the group sizes $df_g$ this turns out to be sufficiently fast. For large-scale applications it would be attractive to have a closed form solution for a
block update as in Yuan and Lin (2006). This will be discussed in the next subsection.

### Block Coordinate Gradient Descent

The key idea of the block coordinate gradient descent method of Tseng and Yun (2008) is to combine a quadratic approximation of the log-likelihood with an additional line search. Using a second order Taylor expansion at $\hat{\beta}^{(t)}$ and replacing the Hessian of the log-likelihood function $\ell(\cdot)$ by a suitable matrix $H^{(t)}$ we define

$$
M^{(t)}_\lambda(d) = -\left\{ \ell(\hat{\beta}^{(t)}) + d^T \nabla \ell(\hat{\beta}^{(t)}) + \frac{1}{2} d^T H^{(t)} d \right\} + \lambda \sum_{g=1}^G s(d_g) \| \hat{\beta}_g^{(t)} + d_g \|_2 
\approx S_\lambda(\hat{\beta}^{(t)} + d),
$$

where $d \in \mathbb{R}^{p+1}$. Now we consider the minimization of $M^{(t)}_\lambda(\cdot)$ with respect to the $g$th penalized parameter group. This means that we restrict ourselves to vectors $d$ with $d_k = 0$ for $k \neq g$. Moreover, we assume that the corresponding $df_g \times df_g$ submatrix $H_{gg}^{(t)}$ is of the form $H_{gg}^{(t)} = h_g^{(t)} \cdot I_{df_g}$ for some scalar $h_g^{(t)} \in \mathbb{R}$.

If $\| \nabla \ell(\hat{\beta}^{(t)})_g - h_g^{(t)} \hat{\beta}_g^{(t)} \|_2 \leq \lambda s(d_g)$, the minimizer of (2.2.3) is

$$
d_g^{(t)} = -\hat{\beta}_g^{(t)}.
$$

Otherwise

$$
d_g^{(t)} = -\frac{1}{h_g^{(t)}} \left\{ \nabla \ell(\hat{\beta}^{(t)})_g - \lambda s(d_g) \frac{\nabla \ell(\hat{\beta}^{(t)})_g - h_g^{(t)} \hat{\beta}_g^{(t)}}{\| \nabla \ell(\hat{\beta}^{(t)})_g - h_g^{(t)} \hat{\beta}_g^{(t)} \|_2} \right\}.
$$

If $d^{(t)} \neq 0$, an inexact line search using the Armijo rule has to be performed: Let $\alpha^{(t)}$ be the largest value in $\{\alpha_0 \delta^l\}_{l \geq 0}$ such that

$$
S_\lambda(\hat{\beta}^{(t)} + \alpha^{(t)} d^{(t)}) - S_\lambda(\hat{\beta}^{(t)}) \leq \alpha^{(t)} \sigma \Delta^{(t)},
$$

where $0 < \delta < 1$, $0 < \sigma < 1$, $\alpha_0 > 0$, and $\Delta^{(t)}$ is the improvement in the objective function $S_\lambda(\cdot)$ when using a linear approximation for the
log-likelihood, i.e.

\[ \Delta^{(t)} = -(d^{(t)})^T \nabla \ell(\hat{\beta}^{(t)}) + \lambda_s(df_g)\|\hat{\beta}_g^{(t)} + d_g^{(t)}\|_2 - \lambda_s(df_g)\|\hat{\beta}_g^{(t)}\|_2. \]

Finally, we define

\[ \hat{\beta}^{(t+1)} = \hat{\beta}^{(t)} + \alpha^{(t)}d^{(t)}. \]

The algorithm is outlined in Table 2.2. When minimizing \( M_\lambda^{(t)}(\cdot) \) with respect to a penalized group, we first have to check whether the minimum is at a non-differentiable point as outlined above. For the (unpenalized) intercept this is not necessary and the solution can be directly computed

\[ d^{(t)}_0 = -\frac{1}{h^{(t)}_0} \nabla \ell(\hat{\beta}^{(t)})_0. \]

For a general matrix \( H^{(t)} \) the minimization with respect to the \( g \)th parameter group depends on \( H^{(t)} \) only through the corresponding submatrix \( H_{gg}^{(t)} \). To ensure a reasonable quadratic approximation in (2.2.3), \( H_{gg}^{(t)} \) is ideally chosen to be close to the corresponding submatrix of the Hessian of the log-likelihood function. Restricting ourselves to matrices of the form \( H_{gg}^{(t)} = h^{(t)}_g \cdot I_{df_g} \), a possible choice is (Tseng and Yun, 2008)

\[ h^{(t)}_g = -\max \left[ \text{diag} \left\{ -\nabla^2 \ell(\hat{\beta}^{(t)})_{gg} \right\}, c_* \right], \quad (2.2.4) \]

where \( c_* > 0 \) is a lower bound to ensure convergence (see Proposition 2.2). The matrix \( H^{(t)} \) does not necessarily have to be recomputed in each iteration. Under some mild conditions on \( H^{(t)} \) convergence of the algorithm is assured as can be seen from Tseng and Yun (2008) and from the proof of Proposition 2.2.

Standard choices for the tuning parameters are for example \( \alpha_0 = 1 \), \( \delta = 0.5 \), \( \sigma = 0.1 \) (Bertsekas, 2003; Tseng and Yun, 2008). Other definitions of \( \Delta^{(t)} \) as for example to include the quadratic part of the improvement are also possible. We refer the reader to Tseng and Yun (2008) for more details and proofs that \( \Delta^{(t)} < 0 \) for \( d^{(t)} \neq 0 \) and that the line search can always be performed.

**Proposition 2.2.** If \( H_{gg}^{(t)} \) is chosen according to (2.2.4), then every limit point of the sequence \( \{\hat{\beta}^{(t)}\}_{t \geq 0} \) is a minimum point of \( S_\lambda(\cdot) \).

**Remark 2.1.** When cycling through the coordinate blocks, we could restrict ourselves to the current active set and visit the remaining blocks.
Chapter 2. The Group Lasso for Logistic Regression

Block Coordinate Gradient Descent Algorithm

(1) Let $\beta \in \mathbb{R}^{p+1}$ be an initial parameter vector.
(2) for $g = 0, \ldots, G$
   \[ H_{gg} \leftarrow h_g(\beta) \cdot I_{d_fg} \]
   \[ d \leftarrow \arg \min_{d} |d_k = 0, k \neq g M_\lambda(d) \]
   if $d \neq 0$
      \[ \alpha \leftarrow \text{Line search} \]
      \[ \beta \leftarrow \beta + \alpha \cdot d \]
   end
end
(3) Repeat step (2) until some convergence criterion is met.

Table 2.2: Logistic Group Lasso Algorithm using Block Coordinate Gradient Descent Minimization.

e.g. every 10th iteration to update the active set. This is especially useful for very high-dimensional settings and it easily allows for $p \approx 10^4 - 10^5$. For the high-dimensional example in Section 2.2.3, this modification decreases the computational times by about 40% of what is reported in Figure 2.2. Moreover, it is also possible to update the coordinate blocks in a non-cyclic manner or all at the same time which would allow for a parallelizable approach with the convergence result still holding.

Remark 2.2. The block coordinate gradient descent algorithm can also be applied to the Group Lasso in other models. For example, any generalized linear model where the response $y$ has a distribution from the exponential family falls into this class. This is available in our R-package grplasso.

A related algorithm is found in Krishnapuram et al. (2005), where a global upper bound on the Hessian is used to solve the Lasso problem for multinomial logistic regression. This approach can also be used with the Group Lasso penalty resulting in a closed form solution for a block update. However, the upper bound is not tight enough for moderate and small values of $\lambda$ which leads to too slow convergence in general. Genkin et al. (2007) overcomes this problem by working with an updated local bound on the second derivative and by restricting the change of the current parameter to a local neighbourhood.
For linear models, the LARS-algorithm (Efron et al., 2004; Osborne et al., 2000a) is very efficient for computing the path of Lasso solutions \( \{ \hat{\beta}_\lambda \}_{\lambda \geq 0} \). For logistic regression, approximate path following algorithms have been proposed (Rosset, 2005; Zhao and Yu, 2007; Park and Hastie, 2007). But with the Group Lasso penalty, some of them are not applicable (Rosset, 2005) or do not necessarily converge to a minimum point of \( S_\lambda(\cdot) \) (Zhao and Yu, 2007), and all of them do not seem to be computationally faster than working iteratively on a fixed grid of penalty parameters \( \lambda \). The latter has been observed as well by Genkin et al. (2007) for logistic regression with the Lasso in large-scale applications.

To calculate the solutions \( \hat{\beta}_\lambda \) on a grid of the penalty parameter \( 0 \leq \lambda_K < \ldots < \lambda_1 \leq \lambda_{\text{max}} \) we can for example start at

\[
\lambda_{\text{max}} = \max_{g \in \{1, \ldots, G\}} \frac{1}{s(df_g)} \| X^T_g (y - \bar{y}) \|_2,
\]

where only the intercept is in the model. We then use \( \hat{\beta}_{\lambda_k} \) as a starting value for \( \hat{\beta}_{\lambda_{k+1}} \) and proceed iteratively until \( \hat{\beta}_{\lambda_K} \) with \( \lambda_K \) equal or close to zero. Instead of updating the approximation of the Hessian \( H^{(t)} \) in each iteration, we can use a constant matrix based on the previous parameter estimates \( \hat{\beta}_{\lambda_k} \) to save computing time, i.e.

\[
H^{(t)}_{gg} = h_g(\hat{\beta}_{\lambda_k}) I_{df_g},
\]

for the estimation of \( \hat{\beta}_{\lambda_{k+1}} \). Some cross-validation can then be used for choosing the parameter \( \lambda \). Most often, we aim for minimal test-sample negative log-likelihood score.

### 2.2.3 Comparison with Other Algorithms

In this subsection we compare the Block Coordinate Gradient Descent algorithm (BCGD) with the algorithm of Kim et al. (2006) (BSR, standing for Blockwise Sparse Regression). After an earlier version of this manuscript, Park and Hastie (2006) (PH) also applied their methodology of Park and Hastie (2007) to Group Lasso models which we also include in our comparison.

We emphasize that BSR is a method which requires the specification of an algorithmic tuning parameter, denoted by \( s \). It is shown in
Kim et al. (2006) that numerical convergence of BSR only holds if $s$ is chosen sufficiently small (depending on the unknown Lipschitz constant of the gradient). Moreover, a small parameter $s$ slows down the computational speed of BSR, and vice-versa for a large $s$. Thus, we are in a situation of trading-off numerical convergence versus computational speed. Our BCGD method does not require the specification of an algorithmic tuning parameter to ensure convergence, and we view this as a very substantial advantage for practical use.

For comparing the different algorithms, we use a random design matrix where the predictors are simulated according to a centered multivariate normal distribution with covariance matrix $\Sigma_{i,j} = \rho^{|i-j|}$. If not stated otherwise, $\rho = 0.5$ is used. For the penalty parameter $\lambda$ multiplicative grids between $\lambda_{\text{max}}$ and $\lambda_{\text{max}}/100$ are used.

For BCGD we use the R package `grplasso` and for BSR our own implementation in R. As BSR works with a constraint instead of a penalty, we use the result of BCGD as constraint value. We use an equivalent stopping criterion as in the package `grplasso`, i.e. the relative function improvement and the relative change of the parameter vector have to be small enough. Although this slowed down the algorithms, it is necessary in order to identify the correct active set of the solution. For both algorithms we make use of the preceding solution of the path as starting value for the next grid point. For BCGD we update the Hessian at each 5th grid point and we use an “ordinary” cycling through the coordinate blocks. For the path-following algorithm PH we use the corresponding Matlab implementation available at [http://www.stanford.edu/~mypark/glasso.htm](http://www.stanford.edu/~mypark/glasso.htm). As recommended, the step length on the $\lambda$-scale is chosen adaptively. However, we were able to run PH with reasonable computing time on very small datasets only.

One of them is motivated by the user guide of PH. It consists of $n = 200$ observations of $G = 3$ groups each having $df = 3$, i.e. $p = 10$ (with intercept). For the design matrix we use $\rho = 0$ and the whole parameter vector is set to zero, i.e. there is no signal. 20 grid points are used for $\lambda$. The corresponding cpu times (in seconds) based on 20 simulation runs are 0.093 (0.01), 0.041 (0.0054), 5.96 (1.23) for BCGD, BSR and PH respectively. Standard deviations are given in parentheses. We used the tuning parameter $s = 0.01$ for BSR. Already for such a simple, low-dimensional problem, BCGD and BSR were substantially
faster than PH. As mentioned above, we could not run PH for larger problems (this is probably due to implementation, but we also think that an optimized implementation of PH, involving potentially large active sets, would be slower than BCGD or BSR).

As a second example, we use a higher-dimensional setting with \( n = 100 \) and \( G = 250 \) groups each having \( df = 4 \) (\( p = 1001 \)). The first 10 groups are active with coefficient 0.2, resulting in a Bayes risk of approximately 0.2. The computing times based on 20 simulation runs are depicted in Figure 2.2, where we have used 100 grid points for \( \lambda \). The boxplot for \( s_0 = 0.025 \) is dashed because the probability for numerical convergence was only 20%. BSR with \( s \) suitably chosen is not faster in this example. The success for numerical convergence depends heavily on the choice of \( s \) and additional time is needed to find an optimal value for \( s \). For some single \( \lambda \) values, BSR sometimes turned out to be faster, but the difference between the computing times when calculating the

![Figure 2.2: CPU times for BCGD (left) and for BSR with different values of the parameter \( s \). See the text for more details.](image-url)
whole path is much smaller due to good starting values and the fact that BSR slows down for small values of $\lambda$.

We also applied BSR to the splice site dataset in Section 2.5. The running times are reported in Table 2.3. We summarize that BCGD is

<table>
<thead>
<tr>
<th>$s$</th>
<th>948</th>
<th>2737</th>
<th>4273</th>
<th>6688</th>
<th>10581</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s = 5 \cdot 10^{-4}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s = 2.5 \cdot 10^{-4}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s = 1.25 \cdot 10^{-4}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s = 6.125 \cdot 10^{-5}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2.3: CPU times [s] on the splice site dataset for BCGD (left) and for BSR with different values for the parameter $s$. The algorithm did not converge for larger values of $s$.

often almost as fast as or even faster than (as for the real splice site dataset) BSR with the optimal algorithmic tuning parameter $s$. This tuning parameter varies very much from problem to problem, and it is highly unrealistic to have reasonable a-priori knowledge about a good parameter. Thus, the user needs to do some trial-and-error first which can be very unpleasant. In contrast, BCGD runs fully automatic and is proved to converge, as described in Section 2.2.2.

Due to implementational issues it can be difficult to compare different algorithms. But the fact that coordinate-wise approaches for sparse models are efficient for high-dimensional data has also been noticed by Genkin et al. (2007) or Balakrishnan and Madigan (2008). They have successfully applied related algorithms for the Lasso even when the number of variables was in the hundreds of thousands. For the coordinate-wise approaches in general, already after a few sweeps through all variables, both the objective function and the number of selected variables is close to the optimal solution.

### 2.2.4 Consistency

A reasonable choice of the tuning parameter $\lambda$ will depend on the sample size $n$, as well as on the number of groups $G$, and the degrees of freedom within each group. Assuming that the degrees of freedom per group are kept fixed, the smoothing parameter $\lambda$ can be taken of order $\log(G)$. Then the Group Lasso can be shown to be globally consistent under some further regularity and sparseness conditions. This section gives more details on this asymptotic result.
Let us consider the data (before rescaling) \((x_i, y_i)\) as independent copies of the population variable \((x, y)\). The negative log-likelihood function is used as loss function which we denote for easier notation by

\[
\gamma_\beta(x, y) = -(y \eta_\beta(x) - \log[1 + \exp\{\eta_\beta(x)\}]).
\]

The theoretical risk is defined as

\[
R(\beta) = \mathbb{E}[\gamma_\beta(x, y)],
\]

and the empirical counterpart as

\[
R_n(\beta) = \frac{1}{n} \sum_{i=1}^n \gamma_\beta(x_i, y_i).
\]

With this notation, the Logistic Group Lasso estimator \(\hat{\beta}_\lambda\) is the minimizer of

\[
\frac{S_\lambda(\beta)}{n} = R_n(\beta) + \frac{\lambda}{n} \sum_{g=1}^G s(df_g) \|\beta_g\|_2.
\]

Let us consider a minimizer

\[
\beta^0 \in \arg \min_{\beta} R(\beta).
\]

Note that if the model is well-specified it holds that

\[
\mathbb{E}[y \mid x] = p_{\beta^0}(x).
\]

There are various ways to measure the quality of the estimation procedure. We will use the global measure

\[
d^2(\eta_{\hat{\beta}_\lambda}, \eta_{\beta^0}) = \mathbb{E}\left[\left|\eta_{\hat{\beta}_\lambda}(x) - \eta_{\beta^0}(x)\right|^2\right].
\]

The following assumptions are made:

(A1) We will suppose that for some constant \(0 < \epsilon \leq 1/2\)

\[
\epsilon \leq p_{\beta^0}(x) \leq 1 - \epsilon
\]

for all \(x\).
(A2) We will require that the matrix
\[ \Sigma = \mathbb{E} [xx^T] \]
is non-singular. We denote the smallest eigenvalue of \( \Sigma \) by \( \nu^2 \).

(A3) Let \( x_g \) denote the \( g \)th predictor in \( x \). We normalize \( x_g \) such that it has identity inner product matrix \( \mathbb{E} [x_g x_g^T] = I_{df} \). With this normalization, we assume in addition that, for some constant \( L_n \),
\[ \max_{g} \max_{x} x_g^T x_g \leq n L_n^2. \]
The smallest possible order for \( L_n^2 \) is \( L_n^2 = O(1/n) \), since we use the normalization \( \mathbb{E} [x_g x_g^T] = I_{df_g} \). For categorical predictors, \( L_n^2 = O(1/n) \) corresponds to the balanced case where in each category the probability of finding an individual in that category is bounded away from zero and one.

One can then show consistency in the following sense. Let \( \beta_g^0 \) denote the elements in the vector \( \beta^0 \) corresponding to the \( g \)th group. Let \( N_0 \) be the number of non-zero group effects, i.e. the number of vectors \( \beta_g^0 \) satisfying \( \| \beta_g^0 \|_2 \neq 0 \). Then there exist universal constants \( C_1, C_2, C_3 \) and \( C_4 \), and constants \( c_1 \) and \( c_2 \) depending on \( \epsilon, \nu \) and \( \max_g df_g \), such that whenever
\[ C_1 (1 + N_0^2) L_n^2 \log G \leq c_1 \text{ and } C_1 \log G \leq \lambda \leq \frac{c_1}{(1 + N_0^2) L_n^2}, \]
the following probability inequality holds:
\[ \mathbb{P} \left[ d^2(\eta_{\beta_{\lambda}}, \eta_{\beta^0}) \geq c_2 \frac{(1 + N_0) \lambda}{n} \right] \leq C_2 \left\{ \log(n) \exp \left( -\frac{\lambda}{C_3} \right) + \exp \left( -\frac{1}{C_4 L_n^2} \right) \right\}. \]
This result follows from arguments similar to the ones used in van de Geer (2003) and Tarigan and van de Geer (2006). An outline of the proof is given in the appendix.

For the asymptotic implications, let us assume that \( \epsilon, \nu \) and \( \max_g df_g \) are kept fixed as \( n \to \infty \) and that \( G \gg \log n \). Take \( \lambda \propto \log G \), i.e. \( \lambda \)
is of the order $\log G$. When for example the number of non-zero group effects $N_0$ satisfies $N_0 = O(1)$, and when $L_n^2 = O(1/\log G)$, then we find the, almost parametric, rate

$$d^2(\eta\beta_\lambda, \eta\beta^0) = O_P\left(\frac{\log G}{n}\right).$$

With $L_n^2 = O(1/n)$, the maximal rate for $N_0$ is $N_0 = O(\sqrt{n}/\log G)$, and when $N_0$ is exactly of this order we arrive at the rate

$$d^2(\eta\beta_\lambda, \eta\beta^0) = O_P\left(\frac{\sqrt{\log G}}{n}\right).$$

**Remark 2.3.** We may improve the result by replacing $N_0$ by the number of non-zero coefficients of “the best” approximation of $\eta\beta^0$, which is the approximation that balances estimation error and approximation error.

**Remark 2.4.** A similar consistency result can be obtained for the Group Lasso for Gaussian regression.

## 2.3 Logistic Group Lasso/Ridge Hybrid

### 2.3.1 General Case

As can be observed in the simulation study in Yuan and Lin (2006), the models selected by the Group Lasso are large compared with the underlying true models. For the ordinary Lasso, smaller models with good prediction performance can be obtained using Lasso with relaxation (Meinshausen, 2007). This idea can also be incorporated into the (logistic) Group Lasso approach and our proposal will also allow us to fit hierarchical models.

Denote by $\hat{I}_\lambda \subseteq \{0, \ldots, G\}$ the index set of predictors that are selected by the Group Lasso with penalty parameter $\lambda$ and by $\hat{M}_\lambda = \{\beta \in \mathbb{R}^{p+1} | \beta_g = 0 \text{ for } g \notin \hat{I}_\lambda\}$ the set of possible parameter vectors of the corresponding submodel. The Group Lasso Ridge Hybrid estimator is defined as

$$\hat{\beta}_{\lambda, \kappa} = \arg\min_{\beta \in \hat{M}_\lambda} -\ell(\beta) + \kappa \sum_{g=1}^G \frac{s(df_g)}{\sqrt{df_g}} \|\beta_g\|_2^2$$  \hspace{1cm} (2.3.1)
for \( \lambda, \kappa \geq 0 \). The penalty in (2.3.1) is rescaled with \( 1/\sqrt{d}\bar{g} \) to ensure that it is of the same order as the Group Lasso penalty. The special case \( \kappa = 0 \) is analogous to the LARS/OLS hybrid in Efron et al. (2004) and is denoted by Group Lasso/MLE hybrid. In this case, we only need the Group Lasso to select a candidate model \( \hat{\mathcal{M}}_\lambda \) and estimate its parameters with the (unconstrained) maximum likelihood estimator. Optimization problem (2.3.1) can be solved with a Newton type algorithm. For large scale applications coordinatewise approaches as used in Genkin et al. (2007) may be more appropriate. The reason why we choose a Ridge type penalty follows in the next subsection.

### 2.3.2 Restriction to Hierarchical Models

When working with interactions between predictors (e.g. factors), the Group Lasso solutions are not necessarily hierarchical. An interaction may be present even though (some) corresponding main-effects are missing. In most applications hierarchical models are preferred because of their interpretability. The above two stage procedure (2.3.1) can be used to produce hierarchical models by expanding the model class \( \hat{\mathcal{M}}_\lambda \) to \( \hat{\mathcal{M}}_\lambda^{\text{hier}} \), where \( \hat{\mathcal{M}}_\lambda^{\text{hier}} \) is the hierarchical model class induced by \( \hat{\mathcal{M}}_\lambda \):

\[
\hat{\beta}_{\lambda,\kappa}^{\text{hier}} \text{ is then defined as in (2.3.1), but with the minimum taken over } \hat{\mathcal{M}}_\lambda^{\text{hier}}.
\]

Instead of using a Ridge type penalty, we could have also used again the Group Lasso penalty and proceed exactly as in Meinshausen (2007), using a “relaxed” penalty \( \kappa \leq \lambda \) in the second stage. While this works well for the general (non-hierarchical) case, there are problems if we restrict ourselves to hierarchical models. Even if we choose \( \kappa \leq \lambda \) the solutions may not be hierarchical due to the expansion of \( \hat{\mathcal{M}}_\lambda \) to \( \hat{\mathcal{M}}_\lambda^{\text{hier}} \). In other words: some variable selection may happen in addition in the second stage. Using a Ridge type penalty, we prevent any further model selection and just do shrinkage.
2.4 Simulation

We use a simulation scheme similar to that of Yuan and Lin (2006) but with larger models. In each simulation run we first sample $n_{\text{train}}$ instances of a 9 dimensional multivariate normal distribution $(T_1, \ldots, T_9)^T$ with mean vector $0$ and covariance matrix $\Sigma_{i,j} = \rho^{|i-j|}$. Each component $T_k$, $k = 1, \ldots, 9$, is subsequently transformed into a four-valued categorical random variable using the quartiles of the standard normal distribution. For the main effects, this results in (non-orthogonalized) predictors $x_i = (x_{i,1}^T, \ldots, x_{i,9}^T)^T \in \mathbb{R}^{9 \times 3}$, $i = 1, \ldots, n_{\text{train}}$. We use the sum-constraint as encoding scheme for the dummy variables, i.e. the coefficients have to add up to zero. The entire predictor space has dimension $4^9 = 262'144$. The corresponding responses $y_i$ are simulated according to a Bernoulli distribution with model based probabilities.

The parameter vector $\beta_g$ of a predictor with $df_g$ degrees of freedom is set up as follows to conform to the encoding scheme. We simulate $df_g + 1$ independent standard normal distributions resulting in $\tilde{\beta}_{g,1}, \ldots, \tilde{\beta}_{g,df_g+1}$ and define

$$\beta_{g,j} = \tilde{\beta}_{g,j} - \frac{1}{df_g + 1} \sum_{k=1}^{df_g+1} \tilde{\beta}_{g,k}$$

for $j \in \{1, \ldots, df_g\}$. The intercept is set to zero. The whole parameter vector $\beta$ is finally rescaled to adjust the empirical Bayes risk $r$ at the desired level, where

$$r = \frac{1}{n} \sum_{i=1}^{n} \min\{p_\beta(x_i), 1 - p_\beta(x_i)\}$$

for some large $n$. For all simulation runs for a given setting of $\rho$ and $r$, the same parameter vector is reused.

The four different cases studied are as follows.

(A) The main effects and the 2-way interaction between the first two factors $x_{i,1}$, $x_{i,2}$ build the true model which has a total of 4 terms or 16 parameters. $n_{\text{train}} = 500$ observations are used in each simulation run.

(B) The underlying true model consists of all main effects and 2-way
interactions between the first 5 factors $x_{i,1}, \ldots, x_{i,5}$ resulting in 16 terms or 106 parameters. $n_{\text{train}} = 500$.

(C) As case (B) but with $n_{\text{train}} = 1000$.

(D) All main effects and 2-way interactions between $x_{i,k}$ and $x_{i,l}$, $|k - l| = 1$ are active. In addition the 2-way interaction between $x_{i,1}$, $x_{i,5}$ and $x_{i,3}$, $x_{i,9}$ are present. This makes a total of 20 terms or 118 parameters. $n_{\text{train}} = 1000$.

For estimation, the candidate models used for the Logistic Group Lasso and its variants consist always of all main effects and all 2-way interactions from the 9 factors, which makes a total of 46 terms or $p = 352$ parameters. We use the restriction for hierarchical model fitting for the two-stage procedures.

In each simulation run, all models are fitted on a training dataset of size $n_{\text{train}}$. A grid of the form $\{\lambda_{\text{max}}, 0.96 \cdot \lambda_{\text{max}}, \ldots, 0.96^{148} \cdot \lambda_{\text{max}}, 0\}$ is used for the penalty parameter $\lambda$ of the Logistic Group Lasso. For the Logistic Group Lasso/Ridge hybrid, we consider values $\kappa \in \{1.5^{11}, 1.5^{10}, \ldots, 1.5^{-5}, 0\}$. The penalty parameters $\lambda$ and $\kappa$ are selected according to the (unpenalized) log-likelihood score on an independent validation set of size $n_{\text{train}}/2$. Finally, the models are evaluated on an additional test set of size $n_{\text{train}}$.

Table 2.4 reports the average test-set negative log-likelihood and the average number of selected terms based on 100 simulation runs for each setting. The corresponding standard deviations are given in parentheses. The Group Lasso produces the largest models followed by Group Lasso/Ridge hybrid and Group Lasso/MLE hybrid. Compared to the underlying true models, Group Lasso seems to select unnecessarily large models with many noise variables resulting in a low true discovery rate (not shown), which is defined as the ratio of the number of correctly selected terms and the total number of selected terms. On the other side, Group Lasso/MLE hybrid is very conservative in selecting terms resulting in a large true discovery rate at the cost of a low true positive rate (not shown). Group Lasso/Ridge hybrid seems to be the best compromise.

The prediction performance measured in terms of the (unpenalized) log-likelihood score on the test set is in most cases best for the
Group Lasso/Ridge hybrid, followed by the Group Lasso and the Group Lasso/MLE hybrid. Group Lasso/Ridge hybrid seems to be able to benefit from the good prediction performance of the Group Lasso with the advantage of producing reasonably sized models.

2.5 Application to Splice Site Detection

The prediction of short DNA motifs plays an important role in many areas of computational biology. Gene finding algorithms such as GEN-NIE (Burge and Karlin, 1997) often rely on the prediction of splice sites. Splice sites are the regions between coding (exons) and non-coding (introns) DNA segments. The 5’ end of an intron is called a donor splice site and the 3’ end an acceptor splice site. A donor site whose first two intron positions are the letters “GT” is called canonical, whereas an acceptor site is called canonical if the corresponding intron ends with “AG”. An overview of the splicing process and of some models that are used for detecting splice sites can be found in Burge (1998).

2.5.1 Dataset

MEMset Donor. This dataset consists of a training set of 8’415 true (encoded as $Y = 1$) and 179’438 false (encoded as $Y = 0$) human donor sites. An additional test set contains 4’208 true and 89’717 false donor sites. A sequence of a real splice site consists of the last 3 bases of the exon and the first 6 bases of the intron. False splice sites are sequences on the DNA which match the consensus sequence at position four and five. Removing the consensus “GT” results in a sequence length of 7 with values in $\{A,C,G,T\}^7$: thus, the predictor variables are 7 factors, each having 4 levels. The data are available at http://genes.mit.edu/burgelab/maxent/ssdata/. A more detailed description can be found in Yeo and Burge (2004).

The original training dataset is used to build a smaller balanced training dataset (5’610 true, 5’610 false donor sites) and an unbalanced validation set (2’805 true, 59’804 false donor sites). All sites are chosen randomly without replacement such that the two sets are disjoint. The
Table 2.4: Average test-set negative log-likelihood and average number of selected terms based on 100 simulation runs. Standard deviations are given in parentheses. Methods: Group Lasso (GL), Group Lasso/Ridge hybrid (GL/R) and Group Lasso/MLE hybrid (GL/MLE).

<table>
<thead>
<tr>
<th>$\rho$</th>
<th>$r$</th>
<th>Test-set negative log-likelihood</th>
<th>Number of terms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>GL</td>
<td>GL/R</td>
</tr>
<tr>
<td>Case A</td>
<td>0.00</td>
<td>0.15</td>
<td>185.57</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(10.85)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(8.28)</td>
</tr>
<tr>
<td></td>
<td>0.20</td>
<td>0.15</td>
<td>185.85</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(11.62)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(8.48)</td>
</tr>
<tr>
<td></td>
<td>0.50</td>
<td>0.15</td>
<td>194.67</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(12.49)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(9.37)</td>
</tr>
<tr>
<td>Case B</td>
<td>0.00</td>
<td>0.15</td>
<td>233.84</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(12.76)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(9.50)</td>
</tr>
<tr>
<td></td>
<td>0.20</td>
<td>0.15</td>
<td>235.29</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(12.71)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(9.40)</td>
</tr>
<tr>
<td></td>
<td>0.50</td>
<td>0.15</td>
<td>232.63</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(11.35)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(7.78)</td>
</tr>
<tr>
<td>Case C</td>
<td>0.00</td>
<td>0.15</td>
<td>408.63</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(16.55)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(14.13)</td>
</tr>
<tr>
<td></td>
<td>0.20</td>
<td>0.15</td>
<td>408.27</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(18.53)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(15.17)</td>
</tr>
<tr>
<td></td>
<td>0.50</td>
<td>0.15</td>
<td>397.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(19.83)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(15.57)</td>
</tr>
<tr>
<td>Case D</td>
<td>0.00</td>
<td>0.15</td>
<td>419.61</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(18.02)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(13.78)</td>
</tr>
<tr>
<td></td>
<td>0.20</td>
<td>0.15</td>
<td>418.67</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(17.91)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(14.50)</td>
</tr>
<tr>
<td></td>
<td>0.50</td>
<td>0.15</td>
<td>416.96</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(18.02)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(15.09)</td>
</tr>
</tbody>
</table>
additional test set remains unchanged. Note that the ratio of true to false sites are equal for the validation and the test set.

2.5.2 Procedure

All models are fitted on the balanced training dataset. As the ratio of true to false splice sites strongly differs from the training to the validation and the test set, the intercept is corrected as follows (King and Zeng, 2001):

$$\hat{\beta}_0^{corr} = \hat{\beta}_0 - \log\left(\frac{\bar{y}}{1-\bar{y}}\right) + \log\left(\frac{\pi}{1-\pi}\right),$$

where $\pi$ is the proportion of true sites in the validation set.

Penalty parameters $\lambda$ and $\kappa$ are selected according to the (unpenalized) log-likelihood score on the validation set using the corrected intercept estimate.

For a threshold $\tau \in (0,1)$ we assign observation $i$ to class 1 if $p_\hat{\beta}(x_i) > \tau$ and to class 0 otherwise. Note that the class assignment can also be constructed without intercept correction by using a different threshold.

The correlation coefficient $\rho_\tau$ corresponding to a threshold $\tau$ is defined as the pearson correlation between the binary random variable of the true class membership and the binary random variable of the predicted class membership. In Yeo and Burge (2004) the maximal correlation coefficient

$$\rho_{max} = \max\{\rho_\tau \mid \tau \in (0,1)\}$$

is used as a goodness of fit statistics on the test set.

The candidate model that was used for the Logistic Group Lasso consists of all 3-way and lower order interactions involving 64 terms or $p = 1156$ parameters. In addition to the standard Logistic Group Lasso estimator, the hierarchical Group Lasso/Ridge hybrid and Group Lasso/MLE hybrid estimators are considered.
2.5.3 Results

The best model with respect to the log-likelihood score on the validation set is the Group Lasso estimator. It is followed by the Group Lasso/Ridge hybrid and the Group Lasso/MLE hybrid. The corresponding values of $\rho_{\text{max}}$ on the test set are 0.6593, 0.6569 and 0.6541, respectively. They are all competitive with the results from Yeo and Burge (2004) whose best $\rho_{\text{max}}$ equals 0.6589. While the Group Lasso solution has some active 3-way interactions, the Group Lasso/Ridge hybrid and the Group Lasso/MLE hybrid only contain 2-way interactions. Figure 2.3 shows the $\ell_2$-norm of each parameter group for the three estimators. The 3-way interactions of the Group Lasso solution seem to be very weak. Considering also the non-hierarchical models for the two-stage procedures yields the same selected terms. Decreasing the candidate model size to only contain 2-way interactions gives similar results.

In summary, the prediction performance of the Group Lasso and its variants is competitive with the Maximum Entropy models that were used in Yeo and Burge (2004) which have been viewed as (among) the best for short motif modeling and splice site prediction. Advantages of the Group Lasso and variants thereof include selection of terms. In addition, other (possibly continuous) predictor variables as for example global sequence information could be naturally included in the Group Lasso approach to improve the rather low correlation coefficients (Yeo and Burge, 2004).

2.6 Discussion

We study the Group Lasso for logistic regression. We present efficient algorithms, especially suitable for very high-dimensional problems, for solving the corresponding convex optimization problem which is inherently more difficult than $\ell_1$-penalized logistic regression. The algorithms rely on recent theory and developments for block coordinate and block coordinate gradient descent (Tseng, 2001; Tseng and Yun, 2008). In contrast with the algorithm in Kim et al. (2006), our procedure is fully automatic and does not require the specification of an algorithmic tuning parameter to ensure convergence. Moreover, our algorithm is much
2.6. Discussion

Figure 2.3: $\ell_2$-norms $\| \hat{\beta}_g \|_2$, $g \in \{1, \ldots, G\}$ of the parameter groups with respect to the blockwise orthonormalized design matrix when using a candidate model with all 3-way interactions. $i:j:k$ denotes the 3-way interaction between the $i$th, $j$th and $k$th sequence position. The same scheme applies to the 2-way interactions and the main effects. Active 3-way interactions are additionally marked with vertical lines.

faster than the recent proposal from Park and Hastie (2006). An implementation can be found in our R-package grplasso. Additionally, we present a statistical consistency theory for the setting where the predictor dimension is potentially much larger than sample size but assuming the true underlying logistic regression model is sparse. The algorithms with the supporting mathematical optimization theory as well as the statistical consistency theory also apply directly to the Group Lasso in other generalized linear models.

Furthermore, we propose the Group Lasso/Ridge hybrid method
which often yields better predictions and better variable selection than the Group Lasso. In addition, our Group Lasso/Ridge hybrid allows for hierarchical model fitting.

Finally, we apply the Group Lasso and its variants to short DNA motif modeling and splice site detection. Our general methodology performs very well in comparison to the maximum entropy method which is considered to be among the best for this task.

2. Appendix

2.A.1 Proof of Lemma 2.1

We will first eliminate the intercept. Let $\beta_1, \ldots, \beta_G$ be fixed. To get the estimate for the intercept we have to minimize a function of the form

$$g(\beta_0) = -\sum_{i=1}^{n} [y_i \{\beta_0 + c_i\} - \log\{1 + \exp(\beta_0 + c_i)\}]$$

with derivative

$$g'(\beta_0) = -\sum_{i=1}^{n} \left\{ y_i - \frac{\exp(\beta_0 + c_i)}{1 + \exp(\beta_0 + c_i)} \right\},$$

where $c_i = \sum_{g=1}^{G} x_{i,g}^T \beta_g$ is a constant. It holds that $\lim_{\beta_0 \to \infty} g'(\beta_0) = n - \sum_{i=1}^{n} y_i > 0$ and $\lim_{\beta_0 \to -\infty} g'(\beta_0) = -\sum_{i=1}^{n} y_i < 0$. Furthermore $g'(\cdot)$ is continuous and strictly increasing. Therefore there exists a unique $\beta_0^* \in \mathbb{R}$ such that $g'(\beta_0^*) = 0$. By the implicit function theorem the corresponding function $\beta_0^*(\beta_1, \ldots, \beta_G)$ is continuously differentiable. Hence, we can replace $\beta_0$ in $S_\lambda(\beta)$ by the continuous function $\beta_0^*(\beta_1, \ldots, \beta_G)$. Because $S_\lambda(\beta) \to \infty$ if $\|(\beta_1, \ldots, \beta_G)\|_2 \to \infty$ the minimum is attained.

2.A.2 Proof of Proposition 2.1

The fact that the groupwise minima are attained follows from the same arguments as in the proof of Lemma 2.1. We now show that step
(3) minimizes the convex function $S_\lambda(\beta_g)$ for $g \geq 1$. Since $S_\lambda(\beta_g)$ is not differentiable everywhere, we invoke subdifferential calculus (Bertsekas, 2003). The subdifferential of $S_\lambda(\cdot)$ with respect to $\beta_g$ is the set $\partial S_\lambda(\beta_g) = \{-X^T(y - p\beta) + \lambda e, e \in E(\beta_g)\}$, $E(\beta_g) = \{e \in \mathbb{R}^{df_g} : e = s(df_g)\beta_g / \|\beta_g\|^2\}$ if $\beta_g \neq 0$ and $\|e\|_2 \leq s(df_g)$ if $\beta_g = 0$. The parameter vector $\beta_g$ minimizes $S_\lambda(\beta_g)$ if and only if $0 \in \partial S_\lambda(\beta_g)$ which is equivalent to the formulation of step 3. Furthermore conditions (A1), (B1) - (B3) and (C2) in Tseng (2001) hold. By Lemma 3.1 and Proposition 5.1 in Tseng (2001) every limit point of the sequence $\{\hat{\beta}^{(t)}\}_{t \geq 0}$ is a stationary point of the convex function $S_\lambda(\cdot)$, hence a minimum point. 

\[\square\]

2.A.3 Proof of Proposition 2.2

The proposition directly follows from Theorem 1(e) in Section 4 of Tseng and Yun (2008). We have to show that $-H^{(t)}$ is bounded by above and away from zero. The Hessian of the negative log-likelihood function is $N = \sum_{i=1}^n p_\beta(x_i)\{1 - p_\beta(x_i)\}x_ix_i^T \preceq \frac{1}{4}X^TX$ in the sense that $N - \frac{1}{4}X^TX$ is negative semidefinite. For the blockmatrix $N_{gg}$ corresponding to the $g$th predictor it follows from the blockwise orthonormalization that $N_{gg} \preceq \frac{n}{4}I_{df_g}$ and hence max$\{\text{diag}(N_{gg})\} \leq \frac{n}{4}$. An upper bound on $-H^{(t)}$ is therefore always guaranteed. The lower bound is enforced by the choice of $H_{gg}^{(t)}$. By the choice of the line search we ensure that $\alpha^{(t)}$ is bounded by above and therefore Theorem 1(e) in Section 4 of Tseng and Yun (2008) can be applied. 

\[\square\]

2.A.4 Outline of the Proof of the Consistency Result

The proof follows the arguments that were used in Tarigan and van de Geer (2006). The latter consider hinge loss instead of logistic loss, but, as they point out, a large part of their results can be easily extended because only the Lipschitz property of the loss is used there. Furthermore, under (A1), logistic loss has the usual “quadratic” behaviour near its overall minimum. This means that it does not share the problem of unknown margin behaviour with hinge loss, i.e. the situation is in that
Chapter 2. The Group Lasso for Logistic Regression

respect simpler than in Tarigan and van de Geer (2006).

The Group Lasso reduces to the \( \ell_1 \)-penalty (the usual Lasso) when there is only one degree of freedom in each group. The extension of consistency results to more degrees of freedom is straightforward, provided \( \max_g df_g \) does not depend on \( n \). We furthermore note that the Group Lasso uses a normalization involving the design matrix of the observed predictors. For the consistency result, one needs to prove that this empirical normalization is uniformly close to the theoretical one. This boils down to showing that empirical and theoretical eigenvalues of the design matrix per group cannot be too far away from each other, uniformly over the groups. Here, we invoke that assumption (A4) implies that \( L_n^2 \) is no larger than \( c_1/(C_1 \log G) \). We then apply (A3) in Bernstein’s inequality to bound the difference in eigenvalues.

A technical extension as compared to Tarigan and van de Geer (2006) is that we do not assume an a priori bound on the functions \( \eta_\beta(\cdot) \). This is now handled by using convexity arguments (similar to van de Geer (2003)), and again part of the assumption (A4), namely that \( \lambda \) is smaller than \( c_1/((1 + N_0^2)L_n^2) \). This assumption ensures that for all \( n \), with high probability, the difference between \( \eta_\beta^0 \) and the estimated regression \( \eta_\beta^\lambda \) is bounded by a constant independent of \( n \). For similar results see van de Geer (2008). \( \square \)
Chapter 3

Smoothing $\ell_1$-Penalized Estimators for High-Dimensional Time-Course Data

When a series of (related) linear models has to be estimated it is often appropriate to combine the different data-sets to construct more efficient estimators. We use $\ell_1$-penalized estimators like the Lasso or the Adaptive Lasso which can simultaneously do parameter estimation and model selection. We show that for a time-course of high-dimensional linear models the convergence rates of the Lasso and of the Adaptive Lasso can be improved by combining the different time-points in a suitable way. Moreover, the Adaptive Lasso still enjoys oracle properties and consistent variable selection. The finite sample properties of the proposed methods are illustrated on simulated data and on a real problem of motif finding in DNA sequences.
3.1 Introduction

The Lasso (Tibshirani, 1996) has attracted a lot of attention for prediction and variable selection in linear regression models, including high-dimensional settings where the number of covariates is much larger than sample size (Greenshtein and Ritov, 2004; Meinshausen and Bühlmann, 2006; Bunea et al., 2007; van de Geer, 2008; Meinshausen and Yu, 2006; Zhang and Huang, 2008). Not only has the idea of \( \ell_1 \)-penalization shown its success in other models (Tibshirani, 1997; Lokhorst, 1999; Park and Hastie, 2007), but also many extensions of the Lasso in linear regression models have been proposed, among them are the Fused Lasso (Tibshirani et al., 2005), the Adaptive Lasso (Zou, 2006) and the Relaxed Lasso (Meinshausen, 2007).

Also for multivariate regression, penalization estimators have been shown to be successful (Turlach et al., 2005; Similä and Tikka, 2007). In many problems there is a natural ordering of the response space: our new methodology and theory are exploiting this fact. If we think of time-course data where we observe a response variable over certain time-points, the relationship between “neighbouring” time-points is expected to be stronger than between more distant time-points. Instead of separately estimating a parameter vector for each time-point, it is often a better strategy to combine information across different time-points. By putting an appropriate constraint on the parameter vector, we can control certain characteristics, e.g. the smoothness over time. As an advantage, we may get a more efficient estimator: By pooling of information, we reduce the variance, typically at the cost of some bias, to achieve a lower mean squared error. For multivariate regression, Breiman and Friedman (1997) use the correlation between the responses to construct an estimator with lower mean squared prediction error. In the discussion of Breiman and Friedman (1997), the idea of relevance weighted likelihood (Hu and Zidek, 2002) is mentioned (Zidek, 1997). We use this idea for \( \ell_1 \)-penalized estimators. By using an estimator which also fits well for neighbouring time-points, we can not only get a smoother behaviour of the parameter vector over time, but also profit from more efficiency, both in estimation accuracy and in variable selection.

The rest of this chapter is organized as follows. In Section 3.2 we introduce the Smoothed Lasso estimator and show that it asymptot-
ically reduces the bound on the mean squared error compared to the univariate Lasso estimator. In Section 3.3 we apply the smoothing idea to the Adaptive Lasso and variants thereof and show that it can consistently select the correct model and has a faster convergence rate than the univariate estimator. Simulations follow in Section 3.4 and a real data analysis for motif search in DNA sequences in Section 3.5. Section 3.6 contains the discussion. All proofs can be found in the appendix.

3.2 Smoothed Lasso

We first start with the definition of the Smoothed Lasso estimator and then study its theoretical properties.

3.2.1 Definition

Assume that we observe data at $N$ different time-points and that at each time-point $t_r, r = 1,\ldots, N$, we have a linear regression problem of the form

$$y(t_r) = X\beta(t_r) + \varepsilon(t_r),$$

where $X$ is the $n \times p$ design matrix, $y(t_r) \in \mathbb{R}^n$ is the response vector, $\beta(t_r) \in \mathbb{R}^p$ is the parameter vector corresponding to time-point $t_r$ and $\varepsilon(t_r) \in \mathbb{R}^n$ is the corresponding error vector: $\varepsilon(t_r), r = 1,\ldots, N$ are assumed to be i.i.d. random vectors with i.i.d. components having mean zero and finite variance $\sigma^2$. Note that the design matrix $X$ does not depend on $t_r$ in our setup (but it could), and hence we consider a multivariate linear model. As commonly used for penalized estimation, we assume that the columns of $X$ are centered and scaled to have empirical column means 0 and column variances 1.

**Remark 3.1.** Generalizations of the methodology and theory include that the design matrix $X$ depends on $t_r$, i.e. $X(t_r)$, and that the errors have correlated components $\text{Cov}(\varepsilon(t_r)) = \Sigma$ or arise from a dependent, stationary process with respect to the time-points.

The idea of the Smoothed Lasso is to use an $\ell_1$-penalty and to suitably combine or smooth the information of the different time-points. It
is defined as

\[ \hat{\beta}_{\lambda_n,w}(t_r) = \arg \min_{\beta} \sum_{s=1}^{N} w(t_s, t_r) \| y(t_s) - X \beta \|_2^2 + \lambda_n \sum_{j=1}^{p} |\beta_j|, \]  

(3.2.1)

where \( w(t_s, t_r) \) are weights satisfying \( \sum_{s=1}^{N} w(t_s, t_r) = 1 \). A typical choice is

\[ w(t_s, t_r) \propto K \left( \frac{t_s - t_r}{h} \right), \]

where \( K(\cdot) \) is a univariate kernel, i.e. \( K(x) \geq 0, K(x) = K(-x), \int_{-\infty}^{\infty} K(x) dx = 1 \), and \( h \) is a bandwidth parameter. Thus, the Smoothed Lasso is \( \hat{\beta}(t_r) = \hat{\beta}_{\lambda_n,h}(t_r) \), depending on two tuning parameters.

We can rewrite the weighted optimization problem (3.2.1) as an ordinary Lasso problem

\[ \hat{\beta}_{\lambda_n,h}(t_r) = \arg \min_{\beta} \| \tilde{y}(t_r) - X \beta \|_2^2 + \lambda_n \sum_{j=1}^{p} |\beta_j|, \]  

(3.2.2)

where

\[ \tilde{y}(t_r) = \sum_{s=1}^{N} w(t_s, t_r) y(t_s). \]

Hence, any algorithm to solve a standard Lasso problem can be used to calculate the Smoothed Lasso estimator for a given bandwidth \( h \).

By forcing the estimate \( \hat{\beta}(t_r) \) to fit also well for “neighbouring” time-points, a smooth (non-parametric) trend of \( \hat{\beta}(t_r) \) as a function of time is usually inherited (if \( p < n \) this is always true because of strict convexity with respect to \( \beta \) and continuity with respect to \( t_r \) of the criterion in (3.2.2)).

**Remark 3.2.** Another approach would be to use a Fused Lasso penalty which also penalizes the absolute value of the differences between neighbouring time-points, i.e. \( |\beta_j(t_r) - \beta_j(t_r-1)| \). Such an approach has two drawbacks: First, we have to model all time-points simultaneously, i.e. fit a model with \( Np \) parameters. Moreover, the Fused Lasso problem is more difficult to solve than the Lasso problem. In our approach we fit \( N \) Lasso problems with \( p \) parameters each. Second, if we want to mimick the behaviour of a bandwidth which is locally adaptive to the underlying
3.2. Smoothed Lasso

ture parameter function $\beta(t)$, we have to introduce a lot of tuning parameters for the Fused Lasso and search over a high-dimensional grid when doing cross-validation.

**Remark 3.3.** We do not assume that the active set (the set of predictors with nonzero coefficients) stays the same over all time-points. Our methodology allows for the fact that some predictors enter or leave the active set along the time-course.

In the next section we first consider the special case of an orthogonal design and indeed, we show that the mean squared error is decreased asymptotically.

### 3.2.2 Orthogonal Case

We consider the situation where the number of parameters equals the number of observations and the design matrix is orthogonal, i.e. $X^TX = nI_n$, and where the errors $\varepsilon(t_r)$ are Gaussian. We focus on a single time-point of interest and therefore omit the time-index for notational simplicity. In Donoho and Johnstone (1994, Theorem 1) it is shown that the (univariate) soft-threshold estimator $\hat{\beta}_{ST}$ (with threshold parameter $\sigma \sqrt{2 \log(n)/n}$), which is equivalent to the Lasso in the orthogonal case (with penalty parameter $\lambda = 2\sigma \sqrt{2 \log(n)n}$), satisfies

$$E[\|\hat{\beta}_{ST} - \beta\|_2^2] \leq (2 \log(n) + 1) \left\{ \frac{\sigma^2}{n} + \sum_{i=1}^{n} \min \left( \frac{\beta_i^2}{\sigma^2}, \frac{\sigma^2}{n} \right) \right\}$$

for all $\beta \in \mathbb{R}^n$ and that this bound is asymptotically sharp in a minimax-sense (Donoho and Johnstone, 1994, Theorem 3). If the non-zero $\beta_i$'s are not of too low order (i.e. $|\beta_i| \gg n^{-1/2}$), the order of this bound is $\log(n) |\mathcal{A}_n|/n$, where $\mathcal{A}_n = \{i \mid \beta_i \neq 0\}$ denotes the active set of the time-point of interest. Even though we restrict ourselves to a class of parameter vectors which stay out of the $n^{-1/2}$-range, the order of the bound in (3.2.3) is sharp because the maximal risk is attained for an element of this class (see the proof of Theorem 3 in Donoho and Johnstone (1994)).

The order $\log(n) |\mathcal{A}_n|/n$ can be decreased by the smoothed estimator as shown in Proposition 3.1.
Chapter 3. Smoothing $\ell_1$-Penalized Estimators

**Proposition 3.1.** Assume Gaussian errors $\varepsilon(t_r)$ and the regularity conditions (RC 1) – (RC 4) described in Appendix 3.A.1. For $h = h_n \asymp \log(n)^{1/5} n^{-1/5} N^{-1/5}$ and $N = N_n$ such that $Nh \to \infty (n \to \infty)$ the risk of the Smoothed Lasso from (3.2.2) asymptotically satisfies: for

$$\lambda_n = 2\sigma(Nh)^{-1/2} \sqrt{2 \log(n)n}$$

$$\mathbb{E}[\|\hat{\beta}_{\lambda_n,h_n} - \beta\|_2^2] \leq C \log(n) |A_n| / (nNh)$$

$$\asymp \log(n)^{4/5} |A_n| / (n^{4/5} N^{4/5}), n \to \infty,$$

for all $\beta \in \mathbb{R}^n$ and some constant $C$.

A proof is given in Appendix 3.A.2.

For a faster convergence rate than $\log(n) |A_n| / n$ (for the unsmoothed Lasso) we require $Nh$ to converge to infinity which implies that

$$N = N_n \gg \left( \frac{n}{\log(n)} \right)^{1/4},$$

i.e. $N$ can be of much lower order than $n$ for achieving a faster convergence rate for the minimax bound.

### 3.2.3 General Case

Let us now consider the general case, i.e. we do not restrict ourselves to an orthogonal design matrix. In particular, we allow for high-dimensional situations where $p = p_n \gg n$ is increasing very fast as $n \to \infty$.

Using the results in Meinshausen and Yu (2006) for a fixed design, the univariate Lasso estimator satisfies under regularity conditions on the design matrix

$$\|\hat{\beta}_{\text{Lasso},\lambda_n} - \beta\|_2^2 \leq O_P \left( \sigma^2 \frac{m_{\lambda_n}}{\log(p_n)} \right) + O \left( \frac{|A_n|}{m_{\lambda_n}} \right), n \to \infty,$$

where $m_{\lambda_n} = Cn^2 / \lambda_n^2$ for some constant $C$. In a certain sense, this bound is tight, see Meinshausen and Yu (2006, Remark 1). We can choose

$$\lambda_n \asymp \sigma^{1/2} n^{3/4} \log(p_n)^{1/4} |A_n|^{-1/4}$$
and arrive at the optimal rate
\[
\|\hat{\beta}_{\text{Lasso}, \lambda_n} - \beta\|_2^2 \leq O_P \left( \sigma n^{-1/2} \log(p_n)^{1/2} |A_n|^{1/2} \right), \quad n \to \infty. \tag{3.2.4}
\]

For proving such a result, assumptions on the design matrix are crucial: Various authors use different conditions, cf. Meinshausen and Yu (2006), Bunea et al. (2007), van de Geer (2008) and Zhang and Huang (2008). We refer the reader to Meinshausen and Yu (2006) for a detailed description of the regularity conditions for (3.2.4).

**Proposition 3.2.** Assume that the univariate Lasso satisfies (3.2.4) and denote the bound on the right hand side of (3.2.4) by \(a_n\). Furthermore, assume the regularity conditions (RC1) – (RC4) described in Appendix 3.A.1. Then, if \(N = N_n \gg |A_n|^{1/4} a_n^{-1/4}\) and for some suitable \(\lambda_n\) and \(h = h_n\):

\[
\|\hat{\beta}_{\lambda_n, h_n} - \beta\|_2^2 = o_P(a_n),
\]

i.e. the Smoothed Lasso has a faster convergence rate than the (tight) bound in (3.2.4) for the Lasso.

A proof is given in Appendix 3.A.2. Suitable choices for \(\lambda_n\) and \(h_n\) in Proposition 3.2 are

\[
h_n \asymp N^{-1/9} |A_n|^{-2/9} a_n^{2/9}
\]

and

\[
\lambda_n \asymp \sigma^{1/2} (Nh)^{-1/4} n^{3/4} \log(p_n)^{1/4} |A_n|^{-1/4}.
\]

Using the notation that is introduced at the beginning of the proof of Proposition 3.1, one can derive other asymptotic properties by linking known results for the Lasso (Greenshtein and Ritov, 2004; Bunea et al., 2007; van de Geer, 2008) with the smoothed model

\[
\tilde{y} = X\tilde{\beta} + \tilde{\epsilon}
\]

and an analysis of the bias term \(\|\hat{\beta} - \beta\|_q\) for \(q \in \{1, 2\}\) as in (3.A.3).

Up to now we only considered the estimation error for \(\beta\) and no variable selection properties. The smoothing reduces the variance and thus it can be expected that the Smoothed Lasso selects more (noise) variables than its univariate counterpart. Empirical evidence of this
property is given in Section 3.4. This problem can be overcome by a second stage which removes many of the coefficients whose estimates are close to zero. In fact, already the case with a univariate response often requires such a second stage for consistent variable selection (Zou, 2006). We will treat a special case in the next section.

### 3.3 Smoothed Adaptive Lasso

The Adaptive Lasso (Zou, 2006) weights the penalty for the different coefficients using an initial estimator $\hat{\beta}_{\text{init}}$, i.e.

$$
\hat{\beta}_{\lambda_n}^{(\hat{\beta}_{\text{init}})} = \arg\min_{\beta} \|y - X\beta\|_2^2 + \lambda_n \sum_{j=1}^{p} \hat{\tau}_j |\beta_j|,
$$

where $\hat{\tau}_j = 1/|\hat{\beta}_{\text{init},j}|^\gamma$ for some $\gamma > 0$ are weights based on the initial estimator $\hat{\beta}_{\text{init}}$. For simplicity we will restrict ourselves to $\gamma = 1$. In Zou (2006), the ordinary least squares (OLS) estimator is used for $\hat{\beta}_{\text{init}}$: here, we will mainly use the Lasso and versions thereof. Through a re-scaling of the columns of the design matrix, the Adaptive Lasso estimator can be formulated as an ordinary Lasso problem, see Zou (2006).

We can also apply the smoothing technique of Section 3.2 to the Adaptive Lasso. In the smoothed case we again replace the residual sum of squares in the objective function with its smoothed counterpart in (3.2.2), i.e.

$$
\hat{\beta}_{\lambda_n,h}^{(\hat{\beta}_{\text{init}},t_r)} = \arg\min_{\beta} \|\hat{y}(t_r) - X\beta\|_2^2 + \lambda_n \sum_{j=1}^{p} \hat{\tau}_j |\beta_j|.
$$

In Zou (2006), an asymptotic oracle result for the Adaptive Lasso is given for fixed dimension $p$. We show that the Smoothed Adaptive Lasso has a faster convergence rate. Again, as we focus on a single time-point, we omit the time-index for notational simplicity.

We will consider the situation where the number of variables $p$ is kept fixed as $n \to \infty$. As before, let $\mathcal{A}$ be the active set of the true parameter vector at the current time-point and $\hat{\mathcal{A}}_n$ be its empirical counterpart.
Theorem 3.1. Assume a fixed design with \( \lim_{n \to \infty} \frac{1}{n} X^T X = C \) for some positive definite matrix \( C \). If \( \hat{\beta}_{\text{init}} - \beta = O_p(a_n^{-1}) \) for some sequence \( a_n \to \infty \), \( \lambda_n \sqrt{N_n h_n}/\sqrt{n} \to 0 \), \( \lambda_n a_n \sqrt{N_n h_n}/\sqrt{n} \to \infty \), \( h_n = o(n^{-1/5} N_n^{-1/5}) \) and \( N_n h_n \to \infty \) \((n \to \infty)\), then the Smoothed Adaptive Lasso in (3.3.1) satisfies under the regularity conditions (RC1) – (RC4) described in Appendix 3.A.1:

\[
\lim_{n \to \infty} \Pr(\hat{A}_n = A) = 1
\]

and

\[
\sqrt{nN_n h_n} (\hat{\beta}_{\lambda_n,h_n,A} - \beta_A) \xrightarrow{d} N(0, \sigma_*^2 (C_{AA})^{-1}), \ n \to \infty,
\]

where \( \sigma_*^2 = \sigma^2 \int_{-\infty}^{\infty} K^2(x) dx \), \( \hat{\beta}_A, \beta_A \) are the sub-vectors of \( \hat{\beta}, \beta \) and \( C_{AA} \) is the sub-matrix of \( C \) corresponding to the active set \( A \).

A proof is given in Appendix 3.A.2. Thus, if the initial estimator is consistent, we can find a sequence \( \lambda_n \) such that the Smoothed Adaptive Lasso has the property of consistent model selection and asymptotic normality on the active set \( A \). Furthermore, if \( N = N_n \gg n^{1/4} \), we can choose \( h = h_n = o(n^{-1/5} N_n^{-1/5}) \) such that \( Nh \to \infty \). Thus, as already pointed out in Section 3.2.2, a relatively small value of \( N = N_n \) is sufficient for achieving an improved convergence rate.

Remark 3.4. The optimal convergence rate \( n^{-2/5} N^{-2/5} \) in Theorem 3.1 can be achieved using \( h_n \propto n^{-1/5} N_n^{-1/5} \). Then, the limiting normal distribution becomes \( N(B_A, \sigma_*^2 (C_{AA})^{-1}) \) for some vector \( B_A \) with \( 0 \leq |B_{A,j}| < \infty \) for all \( j \). This is the same distribution as when using local least squares (with kernel \( K \)). Hence, the Smoothed Adaptive Lasso has an oracle property saying that it is asymptotically as good as local least squares with the true underlying active set \( A \) known beforehand.

3.3.1 Choice of Initial Estimator

The choice of the initial estimator will influence the final estimator. In particular, the sparsity of the final estimator can be maximized by making an appropriate choice, as discussed below.

We will first focus on univariate estimators, i.e. on estimators which only use the data of the current time-point. In view of Theorem 3.1, the
basic assumption for the initial estimator is consistency. The ordinary least squares (OLS) method is a possible choice for low-dimensional problems with fixed dimension \( p \) as it is \( \sqrt{n} \)-consistent. The Lasso is consistent in an \( \ell_2 \)-sense, even in the high-dimensional setting, see Section 3.2.3. Finally, the Adaptive Lasso is \( \sqrt{n} \)-consistent for fixed \( p \) (Zou, 2006) and consistent under suitable regularity conditions for \( p \gg n \) (Huang et al., 2008). For high-dimensional problems the OLS estimator is not appropriate because it is unstable or even not defined in a \( p > n \) situation. The Lasso or Adaptive Lasso are more appropriate choices.

If the initial estimator is doing variable selection, i.e. some of the coefficients \( \hat{\beta}_{\text{init},j} = 0 \), the smoothed estimator is at least as sparse as the initial estimator: a zero-coefficient in the initial estimator, i.e. \( \hat{\beta}_{\text{init},j} = 0 \), results in an infinite penalty for that component, i.e. \( \hat{\tau}_j = \infty \), forcing the smoothed estimate to be zero, i.e. \( \hat{\beta}_j(t_r) = 0 \). This reduces the computational complexity for the smoothing stage since some or even many predictors can be excluded from the model.

For the case that the initial estimator has a tuning parameter, as with the Lasso and the Adaptive Lasso, one would in practice tune it to be prediction optimal. For the Lasso, this produces too large models, i.e. many noise variables are included in the selected model (Meinshausen and Bühlmann, 2006). However, noise variables tend to have small coefficients and will therefore be heavily penalized in the second smoothing step of the Smoothed Adaptive Lasso.

It is of course also possible to use a smoothed estimator as initial estimator, e.g. the Smoothed Lasso. In terms of the number of selected variables, as we will see in Section 3.4, this is often worse than directly using the univariate counterpart. Due to the reduced variance, the smoothed initial estimator tends to select too many variables and not all of them will be eliminated in the second stage of the Smoothed Adaptive Lasso.

In view of some empirical results in Section 3.4, we advocate the following: the initial estimator for the Smoothed Adaptive Lasso is the univariate Adaptive Lasso; the latter itself uses the univariate Lasso as initial estimator. This amounts to be a three-stage procedure where all of the estimations are tuned to be prediction optimal using e.g. some cross-validation scheme. There is substantial agreement by now that
two or more stages are needed to achieve good regularization properties in high-dimensional settings (Zou, 2006; Meinshausen, 2007; Zou and Li, 2008; Meinshausen and Yu, 2006; Wasserman and Roeder, 2008; Bühlmann and Meier, 2008). As a novelty here, our third stage involves an additional smoothing operation.

3.4 Simulations

In this section we want to evaluate the finite sample properties of the proposed estimators.

3.4.1 Design

We consider the following models, similar to Zou (2006):

Model 1: Some large effects

\[ \beta(t) = (0.45t, 3 \sin(t), 3 \cos(t - 3), 0, \ldots, 0) \]

Model 2: Many small effects

\[
\beta(t) = (0.85 + 0.5 \sin(t), 0.85 + 0.5 \cos(t), 0.85 + 0.5 \sin(t - 1), 0.85 + 0.5 \cos(t - 1), \ldots, 0.85 + 0.5 \sin(t - 3), 0.85 + 0.5 \cos(t - 3), 0, \ldots, 0)
\]

Figure 3.1 illustrates the two parameter vectors as a function of time \( t \). We use an equidistant grid on the interval \([0, 2\pi]\), i.e. \( t_r = (r - 1) \frac{2\pi}{N-1}, r \in \{1, \ldots, N\} \), where \( N = 18 \). The design matrix \( X \) is simulated from a multivariate normal distribution with mean zero and covariance matrix \( \Sigma_{i,j} = 0.5^{|i-j|} \). The standard deviation of the error term is chosen from \( \sigma \in \{2, 4\} \) which corresponds to a signal-to-noise ratio (averaged over \( N \)) of approximately \( \{2.7, 0.7\} \) and \( \{3.8, 0.9\} \) for model 1 and model 2, respectively. We use both a “classical” setup with \( n = 50, p = 8 \) and a high-dimensional setup with \( n = 100, p = 1000 \).
Chapter 3. Smoothing $\ell_1$-Penalized Estimators

The best combination of bandwidth $h$ and penalization parameter $\lambda$ is being searched on a two-dimensional grid using an independent validation set of half the size of the training set. This is done independently for each time-point which means that we allow for a locally varying bandwidth. The density of the standard normal distribution is used as kernel function $K(\cdot)$ for the weight function $w(\cdot, \cdot)$, see Section 3.2.1.

For the (Smoothed) Adaptive Lasso with (Smoothed) Lasso as initial estimator, we first determine the optimal penalization parameter for the initial estimator and keep it fixed when searching for the optimal penalization parameter and bandwidth for the final estimator.

All estimators which we compare are listed in Table 3.1.

Table 3.1: Different estimators

<table>
<thead>
<tr>
<th>Univariate Estimators</th>
<th>Smoothed Estimators</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Lasso</td>
<td>4. Lasso</td>
</tr>
<tr>
<td>2. Adapt. Lasso with OLS</td>
<td>5. Adapt. Lasso with smoothed OLS</td>
</tr>
</tbody>
</table>
3.4.2 Performance Measures

To measure the goodness of fit and the ability to pick the model of the correct size we define the following performance measures.

For the mean squared error we use

\[ \text{MSE}_\beta = \frac{1}{N} \sum_{r=1}^{N} \| \hat{\beta}(t_r) - \beta(t_r) \|_2^2. \]

Moreover, we also report the mean squared prediction error for the regression function \( x^T \beta(t_r) \)

\[ \text{MSE}_P = \frac{1}{N} \left( \sum_{r=1}^{N} (\hat{\beta}_0(t_r) - \beta_0(t_r))^2 + (\hat{\beta}(t_r) - \beta(t_r))^T \Sigma (\hat{\beta}(t_r) - \beta(t_r)) \right), \]

where \( \hat{\beta}_0(t_r) \) is the intercept term (and \( \beta_0(t_r) = 0 \) for our simulations) and \( \Sigma \) is the covariance matrix of the covariates.

For the number of variables we define the mean model size

\[ \text{MSize} = \frac{1}{N} \sum_{r=1}^{N} | \hat{A}(t_r)| \]

and the mean number of false positives

\[ \text{FP} = \frac{1}{N} \sum_{r=1}^{N} \sum_{j=1}^{p} 1[\hat{\beta}_j(t_r) \neq 0] \cdot 1[\beta_j(t_r) = 0]. \]

In applied sciences where (possibly expensive) experiments are conducted to verify the selected variables (e.g. in biology), the number of false positives is a crucial quantity one wants to minimize in order to keep the costs low.

3.4.3 Results

The results can be found in Table 3.2. For the high-dimensional setting we did not consider OLS initial estimators. Several conclusions can be
made. Let us first focus on the Lasso estimator. In all simulation settings, smoothing improves the $MSE_\beta$ score substantially. The downside for the Smoothed Lasso estimator is that due to the decreased variance, more noise variables tend to enter the model which results in larger selected models (with more false positives) than for the univariate Lasso estimator. However, in practice one would assign a variable importance score to each coefficient and therefore concentrate first on those with the largest contributions, whereas many of the false positives have small importance scores only.

Also for the Adaptive Lasso, the $MSE_\beta$ scores get decreased by smoothing in all simulation settings. Using a smoothed initial estimator leads to too large models. Take for example Adaptive Lasso with Smoothed Lasso as initial estimator, i.e. proposal 6 in Table 3.1. As we have described above, the Smoothed Lasso tends to select a too large initial model. Although the Adaptive Lasso can eliminate most noise variables in the second stage due to their large weights from small coefficients of the initial estimator, the resulting models still get a bit too large. However, the estimator is very competitive with respect to prediction performance.

Using a univariate initial estimator, i.e. our proposal 7 in Table 3.1, to get more reasonably sized models seems to be a good compromise. It does not only produce the sparsest models but is often also competitive with respect to $MSE_\beta$ and $MSE_P$.

### 3.5 Real Data: Motif Finding in DNA Sequences

We apply the smoothing methodology to a problem of motif regression (Conlon et al., 2003). A motif (typically a 5–15 letter word consisting of letters A, C, G, and T) is a candidate of a binding site of some functional element, e.g. a transcription factor (a protein which regulates gene expression). In Beer and Tavazoie (2004) a collection of various gene expression time-course experiments and a set of candidate motifs for yeast is provided. Gene expression values for a total of 2587 genes are available and $p = 666$ motif candidates are used to build the motif scores for each gene. These measure how well the motifs are represented in the
Table 3.2: Mean values of the different performance measures based on 100 simulation runs for \( n = 50 \), \( p = 8 \) (top) and \( n = 100 \), \( p = 1000 \) (bottom). Standard deviations are given in parentheses. The low-dimensional case of model 2 can’t have false positives because all variables are active.

<table>
<thead>
<tr>
<th>Model 1</th>
<th>( \sigma = 2 )</th>
<th>( \sigma = 4 )</th>
<th>Model 2</th>
<th>( \sigma = 2 )</th>
<th>( \sigma = 4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>0.83 (0.17)</td>
<td>0.65 (0.10)</td>
<td>5.67 (0.50)</td>
<td>2.96 (0.47)</td>
<td>3.14 (0.67)</td>
</tr>
<tr>
<td>2.</td>
<td>0.71 (0.17)</td>
<td>0.57 (0.10)</td>
<td>4.39 (0.50)</td>
<td>1.82 (0.46)</td>
<td>3.03 (0.71)</td>
</tr>
<tr>
<td>3.</td>
<td>0.69 (0.15)</td>
<td>0.56 (0.09)</td>
<td>4.30 (0.48)</td>
<td>1.72 (0.44)</td>
<td>3.05 (0.72)</td>
</tr>
<tr>
<td>4.</td>
<td>0.54 (0.14)</td>
<td>0.43 (0.09)</td>
<td>6.26 (0.49)</td>
<td>3.48 (0.48)</td>
<td>1.93 (0.50)</td>
</tr>
<tr>
<td>5.</td>
<td>0.46 (0.13)</td>
<td>0.38 (0.08)</td>
<td>4.85 (0.50)</td>
<td>2.15 (0.48)</td>
<td>1.75 (0.46)</td>
</tr>
<tr>
<td>6.</td>
<td>0.47 (0.13)</td>
<td>0.38 (0.09)</td>
<td>4.80 (0.51)</td>
<td>2.11 (0.49)</td>
<td>1.80 (0.48)</td>
</tr>
<tr>
<td>7.</td>
<td>0.51 (0.14)</td>
<td>0.41 (0.09)</td>
<td>4.06 (0.47)</td>
<td>1.50 (0.44)</td>
<td>2.29 (0.68)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model 1</th>
<th>( \sigma = 2 )</th>
<th>( \sigma = 4 )</th>
<th>Model 2</th>
<th>( \sigma = 2 )</th>
<th>( \sigma = 4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>1.10 (0.19)</td>
<td>0.86 (0.13)</td>
<td>7.60 (0.17)</td>
<td>–</td>
<td>3.22 (0.42)</td>
</tr>
<tr>
<td>2.</td>
<td>1.33 (0.23)</td>
<td>0.97 (0.15)</td>
<td>7.35 (0.25)</td>
<td>–</td>
<td>4.10 (0.56)</td>
</tr>
<tr>
<td>3.</td>
<td>1.30 (0.22)</td>
<td>0.95 (0.15)</td>
<td>7.26 (0.24)</td>
<td>–</td>
<td>3.99 (0.53)</td>
</tr>
<tr>
<td>4.</td>
<td>0.51 (0.12)</td>
<td>0.42 (0.08)</td>
<td>7.92 (0.08)</td>
<td>–</td>
<td>1.42 (0.34)</td>
</tr>
<tr>
<td>5.</td>
<td>0.60 (0.14)</td>
<td>0.47 (0.10)</td>
<td>7.64 (0.21)</td>
<td>–</td>
<td>1.72 (0.43)</td>
</tr>
<tr>
<td>6.</td>
<td>0.64 (0.14)</td>
<td>0.49 (0.10)</td>
<td>7.60 (0.21)</td>
<td>–</td>
<td>1.84 (0.45)</td>
</tr>
<tr>
<td>7.</td>
<td>0.84 (0.20)</td>
<td>0.61 (0.15)</td>
<td>7.14 (0.28)</td>
<td>–</td>
<td>2.94 (0.57)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model 1</th>
<th>( \sigma = 2 )</th>
<th>( \sigma = 4 )</th>
<th>Model 2</th>
<th>( \sigma = 2 )</th>
<th>( \sigma = 4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>2.16 (0.36)</td>
<td>1.48 (0.23)</td>
<td>27.35 (5.13)</td>
<td>24.92 (5.11)</td>
<td>5.67 (0.70)</td>
</tr>
<tr>
<td>3.</td>
<td>1.07 (0.30)</td>
<td>0.76 (0.18)</td>
<td>7.11 (1.93)</td>
<td>4.79 (1.89)</td>
<td>4.73 (0.84)</td>
</tr>
<tr>
<td>4.</td>
<td>1.05 (0.18)</td>
<td>0.78 (0.12)</td>
<td>41.51 (6.62)</td>
<td>38.87 (6.61)</td>
<td>3.12 (0.50)</td>
</tr>
<tr>
<td>6.</td>
<td>0.48 (0.11)</td>
<td>0.40 (0.09)</td>
<td>12.74 (3.44)</td>
<td>10.14 (3.42)</td>
<td>2.25 (0.50)</td>
</tr>
<tr>
<td>7.</td>
<td>0.74 (0.29)</td>
<td>0.49 (0.14)</td>
<td>5.10 (1.32)</td>
<td>2.79 (1.28)</td>
<td>3.80 (0.90)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model 2</th>
<th>( \sigma = 2 )</th>
<th>( \sigma = 4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>1.37 (0.23)</td>
<td>1.69 (0.34)</td>
</tr>
<tr>
<td>3.</td>
<td>1.52 (0.22)</td>
<td>1.22 (0.23)</td>
</tr>
<tr>
<td>4.</td>
<td>0.50 (0.08)</td>
<td>0.51 (0.09)</td>
</tr>
<tr>
<td>6.</td>
<td>0.51 (0.09)</td>
<td>0.44 (0.08)</td>
</tr>
<tr>
<td>7.</td>
<td>1.00 (0.19)</td>
<td>0.67 (0.14)</td>
</tr>
</tbody>
</table>
upstream regions of the genes. We focus on a time-course experiment spanning \( N = 12 \) different time-points. In summary we have 2587 observations of a 666 dimensional predictor (the motif scores) and a one-dimensional response (the gene expression value) at each of the 12 time-points. Thus, each row of the design matrix \( X \) corresponds to a gene and each column to a motif score. The element \( x_{i,j} \) measures how well the \( j \)th motif score is represented in the upstream region of the \( i \)th gene.

To illustrate the smoothing methods and the effect of different sizes for the training set, we use random subsets of different sizes as training set. An independent validation set is used to determine the prediction optimal tuning parameters. The size of the validation set is half the size of the training set. The remaining data is used as test-set.

The results for a training set of size 1300 is given in Table 3.3. In terms of prediction error, there is not much gain when smoothing the estimators for this data-set, especially for the Adaptive Lasso. A reason for this may be the large variance of the error term. Note that for a new test observation \((x_{\text{test}}, y_{\text{test}})\) we have

\[
\mathbb{E}_{x_{\text{test}}, y_{\text{test}}}[\hat{y} - y_{\text{test}}]^2 = \mathbb{E}[(x^T \hat{\beta} - x^T \beta)^2] + \sigma^2.
\]

The error variance \( \sigma^2 \) is likely to be the dominating quantity since motif regression is known to be very noisy. In terms of variable selection,

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>0.07</td>
<td>0.13</td>
<td>0.44</td>
<td>2.71</td>
<td>1.52</td>
<td>1.66</td>
<td>1.98</td>
<td>2.55</td>
<td>3.17</td>
<td>2.77</td>
<td>3.02</td>
<td>2.92</td>
</tr>
<tr>
<td>3.</td>
<td>0.07</td>
<td>0.13</td>
<td>0.45</td>
<td>2.82</td>
<td>1.59</td>
<td>1.75</td>
<td>2.05</td>
<td>2.64</td>
<td>3.26</td>
<td>2.81</td>
<td>3.09</td>
<td>3.04</td>
</tr>
<tr>
<td>4.</td>
<td>0.07</td>
<td>0.13</td>
<td>0.44</td>
<td>2.71</td>
<td>1.52</td>
<td>1.66</td>
<td>1.99</td>
<td>2.52</td>
<td>3.17</td>
<td>2.78</td>
<td>3.02</td>
<td>2.91</td>
</tr>
<tr>
<td>7.</td>
<td>0.07</td>
<td>0.13</td>
<td>0.45</td>
<td>2.86</td>
<td>1.59</td>
<td>1.76</td>
<td>2.06</td>
<td>2.65</td>
<td>3.27</td>
<td>2.85</td>
<td>3.10</td>
<td>3.06</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>35</td>
<td>28</td>
<td>74</td>
<td>155</td>
<td>133</td>
<td>157</td>
<td>154</td>
<td>141</td>
<td>124</td>
<td>106</td>
<td>123</td>
<td>109</td>
</tr>
<tr>
<td>3.</td>
<td>10</td>
<td>6</td>
<td>29</td>
<td>80</td>
<td>85</td>
<td>35</td>
<td>47</td>
<td>31</td>
<td>40</td>
<td>30</td>
<td>42</td>
<td>49</td>
</tr>
<tr>
<td>4.</td>
<td>35</td>
<td>29</td>
<td>70</td>
<td>155</td>
<td>123</td>
<td>157</td>
<td>177</td>
<td>178</td>
<td>124</td>
<td>98</td>
<td>126</td>
<td>112</td>
</tr>
<tr>
<td>7.</td>
<td>4</td>
<td>5</td>
<td>21</td>
<td>71</td>
<td>46</td>
<td>29</td>
<td>34</td>
<td>22</td>
<td>29</td>
<td>22</td>
<td>35</td>
<td>43</td>
</tr>
</tbody>
</table>

the smoothing step decreases the model size for the Adaptive Lasso estimator and is potentially reducing the number of false positives: In particular for time-points \( t_r = 1, 3, 5, 7, 8, 9, 10 \) the Smoothed Adaptive Lasso yields much sparser model fits. For the Lasso estimator, the
smoothing has a tendency to increase the number of selected variables resulting in rather large models. This coincides with our findings in Section 3.4. If we decrease the training sample size to 200 we see some small improvement with respect to the mean squared prediction error (not shown).

3.6 Discussion

We propose smoothing techniques for $\ell_1$-penalized (Lasso-type) estimators for a time-course of high-dimensional linear models. We show theoretically that for the Lasso and the Adaptive Lasso, better estimates in terms of the mean squared error can be obtained by combining the responses of different time-points in a suitably weighted way. Empirically, the Smoothed Adaptive Lasso estimator yields the sparsest models with competitive mean squared error performance when using the univariate Adaptive Lasso as initial estimator. The Smoothed Lasso estimator has very good performance with respect to the mean squared error but selects too many noise variables in general. An additional thresholding stage would be necessary if the primary interest is in variable selection.

The smoothing methodology can also be applied to generalized linear models (GLM). The main difference is that we can’t rewrite the smoothed estimator as an ordinary lasso problem as in (3.2.2). This implies that the computational burden increases: In the worst case (depending on the support of the kernel and the bandwidth $h$), by stacking the response variables and design matrices of the different time-points, the total sample size is $Nn$, which can be substantially larger than $n$ in (3.2.2), while the dimensionality is still $p$.

Our methodology applies to more general problems than time-course settings. For example, we can directly treat the situation of different (heterogeneous) data-sets $(y(t), X)$, $t = 1, \ldots, N$ (or $(y(t), X(t))$, $t = 1, \ldots, N$) with $n(t) \times 1$ response vectors and $n(t) \times p$ design matrices, where $t$ is the index for the various data-sets. All we need is a suitable pseudo-distance $d(t, s)$ among the different data-sets indexed by $t$ and $s$. The weights in (3.2.2) are then of the form

$$w(t, s) \propto K\left(\frac{d(t, s)}{h}\right).$$
Chapter 3. Smoothing $\ell_1$- Penalized Estimators

The pseudo-distance $d(\cdot, \cdot)$ could be learned from the data, e.g. based on some pseudo-metrics for clustering different data-sets.

Whether the multivariate view over different time-points (or different data-sets) pays off for a particular problem is not clear a-priori. However, our methodology encompasses the univariate Lasso methods, by choosing the bandwidth $h = 0$. Hence, using some cross-validation scheme enables to find out whether pooling information over different time-points (or data-sets) is worthwhile and if so, the Smoothed (Adaptive) Lasso from the multivariate approach renders more accurate estimates.

3.A Appendix

3.A.1 Regularity Conditions

We denote by $\beta(t) \in \mathbb{R}^p$ the true underlying parameter vector as a function of time $t$.

(RC1) Curvature of underlying function

$\beta_j(t)$ is twice continuously differentiable with $\sup_j \left| \beta''_j(t) \right| \leq C < \infty$ for all $t$ and some constant $C$.

(RC2) Equidistant grid

For the asymptotic implications we assume that we have an equidistant grid around the time-point of interest $t_r$ of the form

$$t_s = t_r + \frac{s}{N},$$

where $s = -[N/2], \ldots, [N/2]$. Note that we enumerate using negative values of $s$ as well.

(RC3) Sampling Points

For the time-point of interest $t_r$ we assume that if $\beta_j(t_r) = 0$ it follows that there is an open neighbourhood $U_j \ni t_r$, such that $\beta_j(u) = 0 \forall u \in U_j$. Moreover, we require $\inf_j \text{diam}(U_j) > \delta$ for some $\delta > 0$. I.e. for the time-point of interest no variable enters or leaves the active set.
(RC4) **Compact kernel**  
The kernel function $K(\cdot)$ is assumed to have compact support on $[-1, 1]$.

### 3.A.2 Proofs

**Proof of Proposition 3.1**  
As we focus on a single time-point $t_r$ we omit the time index for notational simplicity, whenever possible. For the smoothed response $\tilde{y}$ we have the model

$$
\tilde{y} = X\tilde{\beta} + \tilde{\epsilon}_N,
$$

at the time-point of interest, where

$$
\tilde{\beta} = \sum_{s=-[Nh]}^{[Nh]} w(t_s, t_r) \beta(t_s)
$$

and

$$
\tilde{\epsilon}_N = \sum_{s=-[Nh]}^{[Nh]} w(t_s, t_r) \varepsilon(t_s).
$$

Note that $(\tilde{\epsilon}_N)_1, \ldots, (\tilde{\epsilon}_N)_n$ are i.i.d. with mean zero and variance given below.

We can now use the decomposition

$$
\|\hat{\beta} - \beta\|^2_2 \leq 2\|\hat{\beta} - \tilde{\beta}\|^2_2 + 2\|\tilde{\beta} - \beta\|^2_2. \tag{3.A.1}
$$

The first term is “classical”. We can use the theory of Donoho and Johnstone (1994) with respect to an error term with reduced variance.
For the asymptotic variance we have

\[
\text{Var}(\tilde{\varepsilon}_N) = \text{Var}\left( \sum_{s=-[Nh]}^{[Nh]} w(t_s, t_r) \varepsilon(t_s) \right) \\
= \sigma^2 \sum_{s=-[Nh]}^{[Nh]} w^2(t_r + \frac{s}{N}, t_r) \\
= \sigma^2 \left( \frac{1}{Nh} \right)^2 \sum_{s=-[Nh]}^{[Nh]} K^2\left( \frac{s/N}{h} \right) \left\{ \frac{1}{Nh} \sum_{s=-[Nh]}^{[Nh]} K\left( \frac{s/N}{h} \right) \right\}^2.
\]

Using a Riemann sum approximation, we arrive at

\[
\text{Var}(\tilde{\varepsilon}_N) \sim \frac{\sigma^2}{Nh} \int K^2(x)dx,
\]  

(3.A.2)

i.e. the error variance is of order \(1/(Nh)\).

Let us now consider the bias term. If \(\beta_i(t_r) = 0\) it follows with the compactness assumption of the kernel and (RC3) that for \(h = h_n\) small enough \(\tilde{\beta}_i(t_r) = 0\). If \(\beta_i(t_r) \neq 0\) we have

\[
\tilde{\beta}_i(t_r) = \sum_{s=-[Nh]}^{[Nh]} w(t_s, t_r) \beta_i(t_r + \frac{s}{N}) \\
= \sum_{s=-[Nh]}^{[Nh]} w(t_s, t_r) \left\{ \beta_i(t_r) + \beta'_i(t_r) \frac{s}{N} + \frac{1}{2} \beta''_i(\tau_s) \frac{s^2}{N^2} \right\} \\
= \beta_i(t_r) + \sum_{s=-[Nh]}^{[Nh]} w(t_s, t_r) \frac{1}{2} \beta''_i(\tau_s) \frac{s^2}{N^2},
\]

where \(|\tau_s - t_r| \leq \frac{s}{N}\).

Hence, by (RC1),

\[
\left| \tilde{\beta}_i(t_r) - \beta_i(t_r) \right| \leq \frac{(Nh)^2}{N^2} C \sum_{s=-[Nh]}^{[Nh]} w(t_s, t_r) = Ch^2.
\]
Therefore
\[ \| \hat{\beta} - \beta \|^2 \leq |A_n| C^2 h^4, \tag{3.A.3} \]
for \( h = h_n \) small enough.

If we choose \( h_n \approx \log(n)^{1/5} n^{-1/5} N^{-1/5} \), all terms in (3.A.1) are of the same order. \( \square \)

**Proof of Proposition 3.2**

We use the decomposition in (3.A.1). Since the variance in the smoothed case is of order \( 1/(Nh) \), see (3.A.2), we obtain
\[ \| \hat{\beta} - \tilde{\beta} \|^2 \leq O_P((Nh)^{-1/2} a_n). \tag{3.A.4} \]
On the other hand, we have by (3.A.3)
\[ \| \hat{\beta} - \beta \|^2 \approx |A_n| h^4. \tag{3.A.5} \]
The optimal rate for the bandwidth minimizing the terms in (3.A.4) and (3.A.5) is
\[ h_{opt} = N_n^{-1/9} |A_n|^{-2/9} a_n^{2/9} \to \infty, \ n \to \infty \]
and we obtain using (3.A.1), (3.A.4) and (3.A.5)
\[ \| \hat{\beta} - \beta \|^2 \leq O_P((Nh_{opt})^{-1/2} a_n). \tag{3.A.6} \]
Since
\[ Nh_{opt} \approx N_n^{8/9} |A_n|^{-2/9} a_n^{2/9} \to \infty, \ n \to \infty \]
because \( N_n \gg |A_n|^{1/4} a_n^{-1/4} \), we see from (3.A.6) that a faster convergence rate \( o_P(a_n) \) is achieved. \( \square \)

**Proof of Theorem 3.1**

As in the proof of Proposition 3.1, we have the model
\[ \tilde{y} = X\tilde{\beta} + \tilde{\varepsilon}_N \]
for the smoothed response \( \tilde{y} \). Multiplying both sides with \( \sqrt{Nh} \) results in
\[ \tilde{y} = X\tilde{\beta} + \tilde{\varepsilon}_N, \tag{3.A.7} \]
with \( \tilde{y} = \sqrt{Nh}\hat{y}, \tilde{X} = \sqrt{Nh}X \) and \( \tilde{\varepsilon}_N = \sqrt{Nh}\tilde{\varepsilon}_N \).

Note that the variance of the error term \( \tilde{\varepsilon}_N \) depends on \( N \). As can be seen from (3.A.2), we have for \( N \to \infty \)

\[
\text{Var}((\tilde{\varepsilon}_N)_i) \sim \sigma^2 \int K^2(x)dx.
\]

Using the rescaled model (3.A.7), we can now adapt the proof of Zou (2006).

Let us first focus on the problem on the original scale. We re-parameterize the parameter vector \( \beta \) as

\[
\beta = \tilde{\beta} + \frac{u}{\sqrt{nNh}},
\]

or \( u = \sqrt{nNh}(\beta - \tilde{\beta}) \in \mathbb{R}^p \). The quantity of interest is \( \hat{u} = \sqrt{nNh}(\tilde{\beta} - \beta) \), where

\[
\hat{u} = \arg \min_u \psi_n(u),
\]

with

\[
\psi_n(u) = \left\| \tilde{y} - \sum_{j=1}^p x_j \left( \tilde{\beta}_j + \frac{u_j}{\sqrt{nNh}} \right) \right\|^2_2 + \lambda_n \sum_{j=1}^p \tilde{w}_j \left| \tilde{\beta}_j + \frac{u_j}{\sqrt{nNh}} \right|.
\]

By multiplying \( \psi_n(u) \) with \( Nh \), we can rewrite \( \hat{u} = \arg \min_u \tilde{\psi}_n(u) \), where

\[
\tilde{\psi}_n(u) = \left\| \tilde{y} - \sum_{j=1}^p \tilde{x}_j \left( \tilde{\beta}_j + \frac{u_j}{\sqrt{Nh}} \right) \right\|^2_2 + \tilde{\lambda}_n \sum_{j=1}^p \tilde{w}_j \left| \tilde{\beta}_j + \frac{u_j}{\sqrt{Nh}} \right|,
\]

and \( \tilde{\lambda}_n = Nh\lambda_n \). Now we can follow the proof of Zou (2006). With slight changes, because of the non-constant variance of the error-term, we arrive at

\[
\sqrt{nNh}(\hat{\beta}_A - \tilde{\beta}_A) \xrightarrow{d} N(0, \sigma^2(C_A A)^{-1}),
\]

where \( A \) is the active set of the unsmoothed parameter vector, i.e. the parameter vector at the current time-point. Finally, observe that

\[
\sqrt{nNh}(\hat{\beta}_A - \beta_A) = \sqrt{nNh}(\hat{\beta}_A - \tilde{\beta}_A) + \sqrt{nNh}(\tilde{\beta}_A - \beta_A),
\]
and that we get for the second term analogously as in (3.A.3), using \( |\mathcal{A}_n| \leq p < \infty \),

\[
 nNh\|\tilde{\beta}_A - \beta_A\|_2^2 \leq CnNh^5
\]

for some constant \( C \). If we choose \( h = o(n^{-1/5}N^{-1/5}) \), the asymptotic normality part follows.

The proof of model selection consistency is analogous to Zou (2006). \(\square\)
Chapter 4

High-Dimensional Additive Modeling

We propose a new sparsity-smoothness penalty for high-dimensional generalized additive models. The combination of sparsity and smoothness is crucial for mathematical theory as well as performance for finite-sample data. We present a computationally efficient algorithm, with provable numerical convergence properties, for optimizing the penalized likelihood. Furthermore, we provide oracle results which yield asymptotic optimality of our estimator for high-dimensional but sparse additive models. Finally, an adaptive version of our sparsity-smoothness penalized approach yields large additional performance gains.

4.1 Introduction

Substantial progress has been achieved over the last years in estimating high-dimensional linear or generalized linear models where the number of covariates $p$ is much larger than sample size $n$. The theoretical properties of penalization approaches like the Lasso (Tibshirani, 1996) are now well understood (Greenshtein and Ritov, 2004; Meinshausen and Bühlmann, 2006; Zhang and Huang, 2008; Meinshausen and Yu, 2009; Bickel et al., 2008) and this knowledge has led to several extensions or
alternative approaches like Adaptive Lasso (Zou, 2006), Relaxed Lasso (Meinshausen, 2007), Sure Independence Screening (Fan and Lv, 2008) and graphical model based methods (Bühlmann et al., 2009). Moreover, with the fast growing amount of high-dimensional data in e.g. biology, imaging or astronomy, these methods have shown their success in a variety of practical problems. However, in many situations the conditional expectation of the response given the covariates may not be linear. While the most important effects may still be detected by a linear model, substantial improvements are sometimes possible by using a more flexible class of models. Recently, some progress has been made regarding high-dimensional additive model selection (Bühlmann and Yu, 2003; Lin and Zhang, 2006; Ravikumar et al., 2008) and some theoretical results are available (Ravikumar et al., 2008).

In this chapter, we consider the problem of estimating a high-dimensional generalized additive model where \( p \gg n \). An approach for high-dimensional additive modeling is described and analyzed in Ravikumar et al. (2008). We use an approach which penalizes both the sparsity and the roughness. This is particularly important if a large number of basis functions is used for modeling the additive components. This is similar to Ravikumar et al. (2008) where the smoothness and the sparsity is controlled in the backfitting step. In addition, our computational algorithm, which builds upon the idea of a Group Lasso problem, has rigorous convergence properties and thus, it is provably correct for finding the optimum of a penalized likelihood function. Moreover, we provide oracle results which establish asymptotic optimality of the procedure.

### 4.2 Penalized Maximum Likelihood for Additive Models

We consider high-dimensional additive regression models with a continuous response \( Y \in \mathbb{R}^n \) and \( p \) covariates \( x^{(1)}, \ldots, x^{(p)} \in \mathbb{R}^n \) connected through the model

\[
Y_i = c + \sum_{j=1}^{p} f_j(x_i^{(j)}) + \varepsilon_i, \quad i = 1, \ldots, n,
\]
where \( c \) is the intercept term, \( \varepsilon_i \) are i.i.d. random variables with mean zero and \( f_j : \mathbb{R} \to \mathbb{R} \) are smooth univariate functions. For identification purposes we assume that all \( f_j \) are centered, i.e.

\[
\sum_{i=1}^{n} f_j(x_i^{(j)}) = 0
\]

for \( j = 1, \ldots, p \). We consider the case of fixed design, i.e. we treat the predictors \( x^{(1)}, \ldots, x^{(p)} \) as non-random.

With some slight abuse of notation we also denote by \( f_j \) the \( n \)-dimensional vector \((f_j(x_1^{(j)}), \ldots, f_j(x_n^{(j)}))^T\). For a vector \( f \in \mathbb{R}^n \) we define \( \|f\|_n^2 = \frac{1}{n} \sum_{i=1}^{n} f_i^2 \).

### 4.2.1 The Sparsity-Smoothness Penalty

In order to construct an estimator which encourages sparsity at the function level, penalizing the norms \( \|f_j\|_n \) would be a suitable approach. Some theory for the case where a truncated orthogonal basis with \( O(n^{1/5}) \) basis functions for each component \( f_j \) is used has been developed in Ravikumar et al. (2008).

If we use a large number of basis functions, which is necessary to be able to capture some functions at high complexity, the resulting estimator will produce function estimates which are too wiggly if the underlying true functions are very smooth. Hence, we need some additional control or restrictions of the smoothness of the estimated functions. In order to get sparse and sufficiently smooth function estimates, we propose the sparsity-smoothness penalty

\[
J(f_j) = \lambda_1 \sqrt{\|f_j\|_n^2 + \lambda_2 I^2(f_j)},
\]

where

\[
I^2(f_j) = \int (f_j''(x))^2 \, dx
\]

measures the smoothness of \( f_j \). The two tuning parameters \( \lambda_1, \lambda_2 \geq 0 \) control the amount of penalization.
Our estimator is given by the following penalized least squares problem

\[
\hat{f}_1, \ldots, \hat{f}_p = \arg \min_{f_1, \ldots, f_p \in \mathcal{F}} \left\| Y - \sum_{j=1}^{p} f_j \right\|^2_n + \sum_{j=1}^{p} J(f_j),
\]

(4.2.1)

where \( \mathcal{F} \) is a suitable class of functions and \( Y = (Y_1, \ldots, Y_n)^T \) is the vector of responses. We assume the same level of regularity for each function \( f_j \). If \( Y \) is centered, we can omit an unpenalized intercept term and the nature of the objective function in (4.2.1) automatically forces the function estimates \( \hat{f}_1, \ldots, \hat{f}_p \) to be centered.

**Proposition 4.1.** Let \( a, b \in \mathbb{R} \) such that \( a < \min_{i,j}\{x_i^{(j)}\} \) and \( b > \max_{i,j}\{x_i^{(j)}\} \). Let \( \mathcal{F} \) be the space of functions that are twice continuously differentiable on \([a, b]\) and assume that there exist minimizers \( \hat{f}_j \in \mathcal{F} \) of (4.2.1). Then the \( \hat{f}_j \)'s are natural cubic splines with knots at \( x_i^{(j)} \), \( i = 1, \ldots, n \).

A proof is given in the appendix. Hence, we can restrict ourselves to the finite dimensional space of natural cubic splines instead of considering the infinite dimensional space of twice continuously differentiable functions.

In the following subsection we illustrate the existence and the computation of the estimator.

### 4.2.2 Computational Algorithm

For each function \( f_j \) we use a cubic B-spline parameterization with a reasonable amount of knots or basis functions. A typical choice would be to use \( K - 4 \approx \sqrt{n} \) interior knots that are placed at the empirical quantiles of \( x^{(j)} \). Hence, we parameterize

\[
f_j(x) = \sum_{k=1}^{K} \beta_{j,k} b_{j,k}(x),
\]

where \( b_{j,k} : \mathbb{R} \to \mathbb{R} \) is the \( k \)th B-spline basis function of the \( j \)th predictor and \( \beta_j = (\beta_{j,1}, \ldots, \beta_{j,K})^T \in \mathbb{R}^K \) is the parameter vector corresponding
4.2. Penalized Maximum Likelihood for Additive Models

to $f_j$. Based on the basis functions we can construct an $n \times pK$ design matrix $B = [B_1 \mid B_2 \mid \ldots \mid B_p]$, where $B_j$ is the $n \times K$ design matrix of the B-spline basis of the $j$th predictor, i.e. $B_{j,il} = b_{j,l}(x_i^{(j)})$.

For twice continuously differentiable functions, the optimization problem (4.2.1) can now be re-formulated as

$$\hat{\beta} = \arg \min_{\beta=(\beta_1,\ldots,\beta_p)} \|Y - B\beta\|_n^2 + \lambda_1 \sum_{j=1}^p \sqrt{\frac{1}{n} \beta_j^T B_j^T B_j \beta_j + \lambda_2 \beta_j^T \Omega_j \beta_j},$$

where the $K \times K$ matrix $\Omega_j$ contains the inner products of the second derivatives of the B-spline basis functions, i.e.

$$\Omega_{j,kl} = \int b_{j,k}''(x)b_{j,l}''(x) \, dx$$

for $k, l \in \{1, \ldots, K\}$.

Hence, (4.2.2) can be re-written as a general Group Lasso problem (Yuan and Lin, 2006)

$$\hat{\beta} = \arg \min_{\beta=(\beta_1,\ldots,\beta_p)} \|Y - B\beta\|_n^2 + \lambda_1 \sum_{j=1}^p \beta_j^T M_j \beta_j,$$

where $M_j = \frac{1}{n} B_j^T B_j + \lambda_2 \Omega_j$. By decomposing (e.g. using the Cholesky decomposition) $M_j = R_j^T R_j$ for some quadratic $K \times K$ matrix $R_j$ and by defining $\tilde{\beta}_j = R_j \beta_j$, $\tilde{B}_j = B_j R_j^{-1}$, (4.2.3) reduces to

$$\hat{\tilde{\beta}} = \arg \min_{\tilde{\beta}=(\tilde{\beta}_1,\ldots,\tilde{\beta}_p)} \|Y - \tilde{B}\tilde{\beta}\|_n^2 + \lambda_1 \sum_{j=1}^p \|\tilde{\beta}_j\|,$$

where $\|\tilde{\beta}_j\| = \sqrt{K} \|\tilde{\beta}_j\|_K$ is the Euclidean norm in $\mathbb{R}^K$. This is an ordinary Group Lasso problem for any fixed $\lambda_2$ and hence the existence of a solution is guaranteed. For $\lambda_1$ large enough, some of the coefficient groups $\beta_j \in \mathbb{R}^K$ will be estimated to be exactly zero. Hence, the corresponding function estimate will be zero. Moreover, there exists a value $\lambda_{1,\text{max}} < \infty$ such that $\hat{\tilde{\beta}}_1 = \ldots = \hat{\tilde{\beta}}_p = 0$ for $\lambda_1 \geq \lambda_{1,\text{max}}$. This is especially useful to construct a grid of $\lambda_1$ candidate values for cross-validation (usually on the log-scale).
Regarding the uniqueness of the identified components, we have equivalent results as for the Lasso. Define \( S(\hat{\beta}; \tilde{B}) = \|Y - \tilde{B}\hat{\beta}\|_n^2 \). Similar to Osborne et al. (2000b), we have the following proposition.

**Proposition 4.2.** If \( pK \leq n \) and if \( \tilde{B} \) has full rank, a unique solution of (4.2.4) exists. If \( pK > n \), there exists a convex set of solutions of (4.2.4). Moreover, if \( \|\nabla_{\hat{\beta}_j} S(\hat{\beta}; \tilde{B})\| < \lambda_1 \) then \( \hat{\beta}_j = 0 \) and all other solutions \( \hat{\beta}_{\text{other}} \) satisfy \( \hat{\beta}_{\text{other},j} = 0 \).

A proof can be found in the appendix.

By re-writing the original problem (4.2.1) in the form of (4.2.4), we can make use of already existing algorithms (Meier et al., 2008a; Kim et al., 2006; Yuan and Lin, 2006) to compute the estimator. Coordinate-wise approaches as in Meier et al. (2008a) and Yuan and Lin (2006) are efficient and have rigorous convergence properties. Thus, we are able to compute the estimator exactly, even if \( p \) is very large.

An example of estimated functions, from simulated data according to Example 1 in Section 4.3, is shown in Figure 4.1. For illustrational purposes we have also plotted the estimator which involves no smoothness penalty \((\lambda_2 = 0)\). The latter clearly shows that for this example, the function estimates are “too wiggly” compared to the true functions. As we will also see later, the smoothness penalty plays a key role for the theory.

**Remark 4.1.** Alternative possibilities of our penalty would be to use either (i) \( J(f_j) = \lambda_1 \|f_j\|_n + \lambda_2 I(f_j) \) or (ii) \( J(f_j) = \lambda_1 \|f_j\|_n + \lambda_2 I^2(f_j) \). Both approaches lead to a sparse estimator. While proposal (i) also enjoys nice theoretical properties (see Meier et al. (2008b)), it is computationally more demanding, because it leads to a second order cone programming problem. Proposal (ii) basically leads again to a Group Lasso problem but appears to have theoretical drawbacks, i.e. the term \( \lambda_2 I^2(f_j) \) is really needed within the square root.

### 4.2.3 Oracle Results

We present now an oracle inequality for the penalized estimator. The proofs can be found in Meier et al. (2008b).
For the theoretical analysis, we introduce an additional penalty parameter $\lambda_3 \geq 0$ for technical reasons. We consider here a penalty of the form

$$J(f_j) = \lambda_1 \sqrt{\|f_j\|^2_n + \lambda_2 I^2(f_j) + \lambda_3 I^2(f_j)}.$$

This penalty involves three smoothing parameters $\lambda_1$, $\lambda_2$ and $\lambda_3$. One may reduce this to a single smoothing parameter by choosing

$$\lambda_2 = \lambda_3 = \lambda_1^2,$$

(see Theorem 4.1 below). In the simulations however, the choice $\lambda_3 = 0$ turned out to provide slightly better results than the choice $\lambda_2 = \lambda_3$. With $\lambda_3 = 0$, the theory goes through provided the smoothness $I(f_j)$ remains bounded in an appropriate sense.

We let $f^0$ denote the “true” regression function (which is not necessarily additive), i.e., we suppose the regression model

$$Y_i = f^0(x_i) + \varepsilon_i,$$
where $x_i = (x_i^{(1)}, \ldots, x_i^{(p)})^T$ for $i = 1, \ldots, n$, and where $\varepsilon_1, \ldots, \varepsilon_n$ are independent random errors with $\mathbb{E}[\varepsilon_i] = 0$. Let $f^*$ be a (sparse) additive approximation of $f^0$ of the form
\[
f^*(x_i) = c^* + \sum_{j=1}^p f_j^*(x_i^{(j)}).
\]
where we take $c^* = \mathbb{E}[\bar{Y}], \bar{Y} = \sum_{i=1}^n Y_i/n$. The result of this subsection (Theorem 4.1) holds for any such $f^*$ satisfying the compatibility condition below. Thus, one may invoke the optimal additive predictor among such $f^*$, which we will call the “oracle”. For an additive function $f$, the squared distance $\|f - f^0\|_n^2$ can be decomposed into
\[
\|f - f^0\|_n^2 = \|f - f^0_{\text{add}}\|_n^2 + \|f^0_{\text{add}} - f^0\|_n^2,
\]
where $f^0_{\text{add}}$ is the projection of $f^0$ on the space of additive functions. Thus, when $f^0$ is itself not additive, the oracle can be seen as the best sparse approximation of the projection $f^0_{\text{add}}$ of $f^0$.

The active set is defined as
\[
\mathcal{A}_* = \{j : \|f_j^*\|_n \neq 0\}.
\]
We define, for $j = 1, \ldots, p$,
\[
\tau_n^2(f_j) = \|f_j\|_n^2 + \lambda^{2-\gamma} I^2(f_j).
\]
Moreover, we let $0 < \eta < 1$ be some fixed value. The constant $4/(1 - \eta)$ appearing below in the compatibility condition is stated in this form to facilitate reference, later in the proof of Theorem 4.1.

We will use a compatibility condition, in the spirit of the incoherence conditions used for proving oracle inequalities for the standard Lasso (see e.g. Bickel et al. (2008), Bunea et al. (2006), Bunea et al. (2007), Candes and Tao (2007) and van de Geer (2008)). To avoid digressions, we will not attempt to formulate the most general condition. A discussion can be found in Meier et al. (2008b).

**Compatibility condition** For some constants $0 < \eta < 1$ and $0 < \phi_{n,*} \leq 1$, and for all $\{f_j\}_{j=1}^p$ satisfying
\[
\sum_{j=1}^p \tau_n(f_j) \leq \frac{4}{1 - \eta} \sum_{j \in \mathcal{A}_*} \tau_n(f_j),
\]
the following inequality is met:

\[
\sum_{j \in A_*} ||f_j||_n^2 \leq \left( ||\sum_{j=1}^{p} f_j||_n^2 + \lambda^{2-\gamma} \sum_{j \in A_*} I^2(f_j) \right) / \phi_n^2.
\]

As before, we assume that \( f_j \in \mathcal{F} \) for \( j = 1, \ldots, p \). For practical applications, the compatibility condition can not be checked because the set \( A_* \) is unknown.

Consider the general case where \( I \) is some semi-norm, e.g. as in Section 4.2.1. For mathematical convenience we write

\[ f_j = g_j + h_j, \]  

(4.2.6)

with \( g_j \) and \( h_j \) centered and orthogonal functions, i.e.

\[
\sum_{i=1}^{n} g_{j,i} = \sum_{i=1}^{n} h_{j,i} = 0
\]

and

\[
\sum_{i=1}^{n} g_{j,i} h_{j,i} = 0,
\]

such that \( I(h_j) = 0 \) and \( I(g_j) = I(f_j) \). The functions \( h_j \) are assumed to lie in a \( d \)-dimensional space. The entropy of \( (\{g_j : I(g_j) = 1\}, ||\cdot||_n) \) is denoted by \( H_j(\cdot) \), see e.g. van de Geer (2000). We assume that for all \( j \),

\[ H_j(\delta) \leq A\delta^{-2(1-\alpha)}, \ \delta > 0, \]  

(4.2.7)

where \( 0 < \alpha < 1 \) and \( A > 0 \) are constants. When \( I^2(f_j) = \int \left( f_j''(x) \right)^2 dx \), the functions \( h_j \) are the linear part of \( f_j \), i.e. \( d = 1 \). Moreover, one then has \( \alpha = 3/4 \) (see e.g. van de Geer (2000), Lemma 3.9).

Finally, we assume sub-Gaussian tails for the errors: for some constants \( L \) and \( M \),

\[ \max_i \mathbb{E} \left[ \exp \left( \varepsilon_i^2 / L \right) \right] \leq M. \]  

(4.2.8)

The next lemma presents the behavior of the empirical process. We use the notation \( (\varepsilon, f)_n = \frac{1}{n} \sum_{i=1}^{n} \varepsilon_i f(x_i) \) for the inner product. Define

\[ S = S_1 \cap S_2 \cap S_3 \]  

(4.2.9)
where
\[ S_1 = \left\{ \max_j \sup_{g_j} \left( \frac{2|\varepsilon, g_j|_n}{\|g_j\|_n I^{1-\alpha}(g_j)} \right) \leq \xi_n \right\}, \]
\[ S_2 = \left\{ \max_j \sup_{h_j} \left( \frac{2|\varepsilon, h_j|_n}{\|h_j\|_n} \right) \leq \xi_n \right\}, \]
and
\[ S_3 = \{ \bar{\varepsilon} \leq \xi_n \}, \quad \bar{\varepsilon} = \frac{1}{n} \sum_{i=1}^{n} \varepsilon_i. \]
For an appropriate choice of \( \xi_n \), the set \( S \) has large probability.

**Lemma 4.1.** Assume (4.2.7) and (4.2.8). There exist constants \( c \) and \( C \) depending only on \( d, \alpha, A, L, \) and \( M \), such that for
\[ \xi_n \geq C \sqrt{\frac{\log p}{n}}, \]
one has
\[ \mathbb{P}(S) \geq 1 - c \exp\left[-n \xi_n^2/c^2\right]. \]

For \( \alpha \in (0, 1) \), we define its “conjugate” \( \gamma = 2(1-\alpha)/(2-\alpha) \). Recall that when \( I^2(f_j) = \int (f''_j(x))^2 \, dx \), one has \( \alpha = 3/4 \), and hence \( \gamma = 2/5 \).

We are now ready to state the oracle result for \( \hat{f} = \hat{c} + \sum_{j=1}^{p} \hat{f}_j \) as defined in (4.2.1), with \( \hat{c} = \bar{Y} \).

**Theorem 4.1.** Suppose the compatibility condition is met. Take for \( j = 1, \ldots, p \),
\[ J(f_j) = \lambda_1 \sqrt{\|f_j\|_n^2 + \lambda_2 I^2(f_j) + \lambda_3 I^2(f_j)}, \]
with \( \lambda_1 = \lambda^{2-\gamma} \) and \( \lambda_2 = \lambda_3 = \lambda_1^2 \), and with \( \xi_n \sqrt{2/\eta} \leq \lambda \leq 1 \). Then on the set \( S \) given in (4.2.9), it holds that
\[ \| \hat{f} - f^0_{add} \|_n^2 + 2(1-\eta)\lambda^{2-\gamma} \sum_{j=1}^{p} \tau_n(\hat{f}_j - f^*_j) + \lambda^{2-\gamma} \sum_{j=1}^{p} I^2(\hat{f}_j) \]
\[ \leq 3\| f^* - f^0_{add} \|_n^2 + 3\lambda^{2-\gamma} \sum_{j \in A_*} \left[ I^2(f_j^*) + \frac{8}{\phi_{n,*}^2} \right] + 2\xi_n^2. \]
The result of Theorem 4.1 does not depend on the number of knots (basis functions) which are used to build the functions $\hat{f}_j$, as long as $\hat{f}_j$ and $\hat{f}_j^*$ use the same basis functions.

We would like to point out that the theory of Theorem 4.1 goes through with only two tuning parameters $\lambda_1$ and $\lambda_2$ but with the additional restriction that $I(\hat{f}_j)$ is appropriately bounded.

We also remark that we did not attempt to optimize the constants given in Theorem 4.1, but rather looked for a simple explicit bound.

Remark 4.2. Assume that $\phi_{n,*}$ is bounded away from zero. For example, this holds with large probability for a realization of a design with independent components (see Meier et al. (2008b)). In view of Lemma 4.1, one may take (under the conditions of this lemma) the smoothing parameter $\lambda$ of order $\sqrt{\log p/n}$. For $I^2(f_j) = \int (f_j''(x))^2 dx$, $\gamma = 2/5$ and this gives $\lambda^{2-\gamma}$ of order $(\log p/n)^{4/5}$, which is up to the log-term the usual rate for estimating a twice differentiable function. If the oracle $f^*$ has bounded smoothness $I(f_j^*)$ for all $j$, Theorem 4.1 yields the convergence rate $p_{\text{act}} (\log p/n)^{4/5}$, with $p_{\text{act}} = |A_*|$ being the number of active variables the oracle needs. This is, again up to the log-term, the same rate one would obtain if it was known beforehand which of the $p$ functions are relevant. For general $\phi_{n,*}$ we have the convergence rate $p_{\text{act}} \phi_{n,*}^{-2} (\log p/n)^{4/5}$.

Furthermore, the result implies that with large probability, the estimator selects a sup-set of the active functions, provided that the latter have enough signal (such kind of variable screening results have been established for the Lasso in linear and generalized linear models, cf. van de Geer (2008) and Meinshausen and Yu (2009). More precisely, we have the following Corollary.

Corollary 4.1. Let $A_0 = \{ j : \|f^0_{\text{add},j}\|_n \neq 0 \}$ be the active set of $f^0_{\text{add}}$. Assume the compatibility condition holds for $A_0$, with constant $\phi_{n,0}$. Suppose also that for $j \in A_0$, the smoothness is bounded, say $I(f^0_{\text{add},j}) \leq 1$. Choosing $f^* = f^0_{\text{add}}$ in Theorem 4.1, tells us that on $S$,

$$\sum_{j=1}^{p} \|\hat{f}_j - f^0_{\text{add},j}\|_n \leq C \lambda^{2-\gamma} |A_0|/\phi_{n,0}^2 + 2\xi_n^2$$
for some constant $C$. Hence, if
\[ \| f_{\text{add},j}^0 \|_n > C \lambda^{2-\gamma} |A_0| / \phi_{n,0}^2 + 2 \xi_n^2, \ j \in A_0, \]
we have (on $S$), that the estimated active set \{ $j : \| \hat{f}_j \|_n \neq 0$ \} contains $A_0$.

### 4.3 Numerical Examples

#### 4.3.1 Simulations

In this section we investigate the empirical properties of the proposed estimator. We compare our approach with the Boosting approach of Bühlmann and Yu (2003), where smoothing splines with low degrees of freedom are used as base learners; see also Bühlmann and Hothorn (2007). For $p = 1$, boosting with splines is known to be able to adapt to the smoothness of the underlying true function (Bühlmann and Yu, 2003). Generally, boosting is a very powerful machine learning method and a wide variety of software implementations are available, e.g. the R add-on package mboost.

We use a training set of $n$ samples to train the different methods. An independent validation set of size $\lfloor n/2 \rfloor$ is used to select the prediction optimal tuning parameters $\lambda_1$ and $\lambda_2$. We use grids (on the log-scale) for both $\lambda_1$ and $\lambda_2$, where the grid for $\lambda_1$ is of size 100 and the grid for $\lambda_2$ is typically of about size 15. For boosting, the number of boosting iterations is used as tuning parameter. The shrinkage factor $\nu$ and the degrees of freedom $df$ of the boosting procedure are set to their default values $\nu = 0.1$ and $df = 4$; see also Bühlmann and Hothorn (2007).

By SNR we denote the signal-to-noise ratio, which is defined as
\[ \text{SNR} = \frac{\text{Var}(f(X))}{\text{Var}(\varepsilon)}, \]
where $f = f^0 : \mathbb{R}^p \to \mathbb{R}$ is the true underlying function.

A total of 100 simulation runs are used for each of the following settings.
4.3. Numerical Examples

Models

We use the following simulation models.

**Example 1** \((n = 150, p = 200, p_{\text{act}} = 4, \text{SNR} \approx 15)\)
This example is similar to Example 1 in Ravikumar *et al.* (2008) and Härdle *et al.* (2004). The model is

\[
Y_i = f_1(x_i^{(1)}) + f_2(x_i^{(2)}) + f_3(x_i^{(3)}) + f_4(x_i^{(4)}) + \varepsilon_i, \ \varepsilon_i \text{ i.i.d. } N(0, 1),
\]

with

\[
f_1(x) = -\sin(2x), \ f_2(x) = x^2 - 25/12, \ f_3(x) = x, \ f_4(x) = e^{-x} - 2/5 \cdot \sinh(5/2).
\]

The covariates are simulated from independent Uniform\((-2.5, 2.5)\) distributions. The true and the estimated functions of a simulation run are illustrated in Figure 4.1.

**Example 2** \((n = 100, p = 1000, p_{\text{act}} = 4, \text{SNR} \approx 6.7)\)
As above but high-dimensional and correlated. The covariates are simulated according to a multivariate normal distribution with covariance matrix \(\Sigma_{ij} = 0.5^{|i-j|}; i, j = 1, \ldots, p.\)

**Example 3** \((n = 100, p = 80, p_{\text{act}} = 4, \text{SNR} \approx 9 (t = 0), \approx 7.9 (t = 1))\)
This is similar to Example 1 in Lin and Zhang (2006) but with more predictors. The model is

\[
Y_i = 5f_1(x_i^{(1)}) + 3f_2(x_i^{(2)}) + 4f_3(x_i^{(3)}) + 6f_4(x_i^{(4)}) + \varepsilon_i, \ \varepsilon_i \text{ i.i.d. } N(0, 1.74),
\]

with

\[
f_1(x) = x, \ f_2(x) = (2x - 1)^2, \ f_3(x) = \frac{\sin(2\pi x)}{2 - \sin(2\pi x)}
\]

and

\[
f_4(x) = 0.1 \sin(2\pi x) + 0.2 \cos(2\pi x) + 0.3 \sin^2(2\pi x) + 0.4 \cos^3(2\pi x) + 0.5 \sin^3(2\pi x).
\]

The covariates \(x = (x^{(1)}, \ldots, x^{(p)})^T\) are simulated according to

\[
x^{(j)} = \frac{W^{(j)} + tU}{1 + t}, \ j = 1, \ldots, p,
\]
where \( W^{(1)}, \ldots, W^{(p)} \) and \( U \) are i.i.d. Uniform(0, 1). For \( t = 0 \) this is the independent uniform case. The case \( t = 1 \) results in a design with correlation 0.5 between all covariates.

The true functions and the first 6 estimated functions of a simulation run with \( t = 0 \) are illustrated in Figure 4.2.

![Figure 4.2](image_url)

**Figure 4.2:** True functions \( f_j \) (solid) and estimated functions \( \hat{f}_j \) (dashed) for the first 6 components of a simulation run of Example 3 \((t = 0)\). Small vertical bars indicate original data and grey vertical lines knot positions. The dotted lines are the function estimates when no smoothness penalty is used, i.e. when setting \( \lambda_2 = 0 \).

Moreover, we also consider a “high-frequency” situation where we use \( f_3(8x) \) and \( f_4(4x) \) instead of \( f_3(x) \) and \( f_4(x) \). The corresponding signal-to-noise ratios for these models are \( \text{SNR} \approx 9 \) for \( t = 0 \) and \( \text{SNR} \approx 8.1 \) for \( t = 1 \).

**Example 4** \((n = 100, p = 60, p_{act} = 12, \text{SNR} \approx 9 \ (t = 0), \approx 11.25 \ (t = 1))\)

This is similar to Example 2 in Lin and Zhang (2006) but with fewer observations. We use the same functions as in Example 3. The model
is

\[ Y_i = f_1(x_i^{(1)}) + f_2(x_i^{(2)}) + f_3(x_i^{(3)}) + f_4(x_i^{(4)}) + \]
\[ 1.5f_1(x_i^{(5)}) + 1.5f_2(x_i^{(6)}) + 1.5f_3(x_i^{(7)}) + 1.5f_4(x_i^{(8)}) + \]
\[ 2f_1(x_i^{(9)}) + 2f_2(x_i^{(10)}) + 2f_3(x_i^{(11)}) + 2f_4(x_i^{(12)}) + \varepsilon_i, \]

with \( \varepsilon_i \) i.i.d. \( N(0, 0.5184) \). The covariates are simulated as in Example 3.

**Performance Measures**

In order to compare the prediction performances we use the mean squared prediction error

\[ PE = \mathbb{E}_X [(\hat{f}(X) - f(X))^2] \]

as performance measure. The above expectation is approximated by a sample of 10,000 points from the distribution of \( X \). In each simulation run we compute the ratio of the prediction performance of the two methods. Finally, we take the mean of the ratios over all simulation runs.

For variable selection properties we use the number of true positives (TP) and false positives (FP) at each simulation run. We report the average number over all runs to compare the different methods.

**Results**

The results are summarized in Table 4.1 and 4.2. The sparsity-smoothness penalty approach (SSP) has smaller prediction error than boosting, especially for the “high-frequency” situations. Because the weak learners of the boosting method only use 4 degrees of freedom, boosting tends to neglect or underestimate those components with higher oscillation. This can also be observed with respect to the number of true positives. By relaxing the smoothness penalty (i.e. choosing \( \lambda_2 \) small or setting \( \lambda_2 = 0 \)), SSP is able to handle the high-frequency situations, at the cost of too wiggly function estimates for the remaining components. Using a different amount of regularization for sparsity and smoothness, SSP
can work with a large amount of basis functions in order to be flexible enough to capture sophisticated functional relationships and, on the other side, to produce smooth estimates if the underlying functions are smooth.

With the exception of the high-frequency examples, the number of true positives (TP) is very similar for both methods. There is no clear trend with respect to the number of false positives (FP).

<table>
<thead>
<tr>
<th>Model</th>
<th>$PE_{SSP}/PE_{boost}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example 1</td>
<td>0.93 (0.13)</td>
</tr>
<tr>
<td>Example 2</td>
<td>0.96 (0.10)</td>
</tr>
<tr>
<td>Example 3 ($t = 0$)</td>
<td>0.81 (0.13)</td>
</tr>
<tr>
<td>Example 3 ($t = 1$)</td>
<td>0.90 (0.19)</td>
</tr>
<tr>
<td>Example 3 “high-freq” ($t = 0$)</td>
<td>0.65 (0.11)</td>
</tr>
<tr>
<td>Example 3 “high-freq” ($t = 1$)</td>
<td>0.57 (0.10)</td>
</tr>
<tr>
<td>Example 4 ($t = 0$)</td>
<td>0.89 (0.10)</td>
</tr>
<tr>
<td>Example 4 ($t = 1$)</td>
<td>0.88 (0.13)</td>
</tr>
</tbody>
</table>

Table 4.1: Results of the different simulation models. Reported is the mean of the ratio of the prediction error of the two methods. SSP: Sparsity-Smoothness Penalty approach, boost: Boosting with smoothing splines. Standard deviations are given in parentheses.

<table>
<thead>
<tr>
<th>Model</th>
<th>$TP_{SSP}$</th>
<th>$FP_{SSP}$</th>
<th>$TP_{boost}$</th>
<th>$FP_{boost}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ex 1</td>
<td>4.00 (0.00)</td>
<td>24.30 (14.11)</td>
<td>4.00 (0.00)</td>
<td>22.18 (12.75)</td>
</tr>
<tr>
<td>Ex. 2</td>
<td>3.47 (0.61)</td>
<td>34.37 (17.38)</td>
<td>3.60 (0.64)</td>
<td>28.76 (20.15)</td>
</tr>
<tr>
<td>Ex. 3 ($t = 0$)</td>
<td>4.00 (0.00)</td>
<td>20.20 (9.30)</td>
<td>4.00 (0.00)</td>
<td>21.61 (10.90)</td>
</tr>
<tr>
<td>Ex. 3 ($t = 1$)</td>
<td>3.93 (0.29)</td>
<td>19.28 (9.61)</td>
<td>3.92 (0.27)</td>
<td>18.65 (8.35)</td>
</tr>
<tr>
<td>Ex. 3 “high-freq” ($t = 0$)</td>
<td>2.80 (0.78)</td>
<td>12.26 (7.61)</td>
<td>2.16 (0.94)</td>
<td>9.23 (9.74)</td>
</tr>
<tr>
<td>Ex. 3 “high-freq” ($t = 1$)</td>
<td>2.46 (0.85)</td>
<td>11.17 (8.50)</td>
<td>1.59 (1.27)</td>
<td>13.24 (13.89)</td>
</tr>
<tr>
<td>Ex. 4 ($t = 0$)</td>
<td>11.69 (0.56)</td>
<td>21.23 (6.85)</td>
<td>11.68 (0.57)</td>
<td>25.91 (9.43)</td>
</tr>
<tr>
<td>Ex. 4 ($t = 1$)</td>
<td>10.64 (1.15)</td>
<td>19.78 (7.51)</td>
<td>10.67 (1.25)</td>
<td>23.76 (9.89)</td>
</tr>
</tbody>
</table>

Table 4.2: Average values of the number of true (TP) and false (FP) positives. Standard deviations are given in parentheses.

4.3.2 Real Data

In this section we would like to compare the different estimators on real datasets.
4.3. Numerical Examples

Tecator

The *meatspec* dataset contains data from the Tecator Infratec Food and Feed Analyzer. It is for example available in the R add-on package *faraway* and on StatLib. The $p = 100$ predictors are channel spectrum measurements and are therefore highly correlated. A total of $n = 215$ observations are available.

The data is split into a training set of size 100 and a validation set of size 50. The remaining data are used as test set. On the training dataset, the first 30 principal components are calculated, scaled to unit variance and used as covariates in additive modeling. Moreover, the validation and the test dataset are transformed to correspond to the principal component of the training dataset. We fit an additive model to predict the logarithm of the fat content. This is repeated 50 times. For each split into training and test data we compute the ratio of the prediction errors from the SSP and boosting method on the test data, as in Section 4.3.1. The mean of the ratio over the 50 splits is 0.86, the corresponding standard deviation is 0.46. This indicates superiority of our sparsity-smoothness penalty approach.

Motif Regression

In motif regression problems (Conlon et al., 2003), the aim is to predict gene expression levels or binding intensities based on information on the DNA sequence. For our specific dataset, from the Ricci lab at ETH Zurich, we have binding intensities $Y_i$ of a certain transcription factor (TF) at 287 regions on the DNA. Moreover, for each region $i$, motif scores $x_i^{(1)}, \ldots, x_i^{(p)}, p = 196$ are available. A motif is a candidate for the binding site of the TF on the DNA, typically a 5–15bp long DNA sequence. The score $x_i^{(j)}$ measures how well the $j$th motif is represented in the $i$th region. The candidate list of motifs and their corresponding scores were created with a variant of the MDScan algorithm (Liu et al., 2002). The main goal is here to find the relevant covariates.

We used 5 fold cross-validation to determine the prediction optimal tuning parameters, yielding 28 active functions. To assess the stability of the estimated model, we performed a nonparametric bootstrap analysis. At each of the 100 bootstrap samples, we fit the model with the
Chapter 4. High-Dimensional Additive Modeling

fixed optimal tuning parameters from above. The two functions which appear most often in the bootstrapped model estimates are depicted in Figure 4.3. While the left-hand side plot shows an approximate linear relationship, the effect of the other motif seems to diminish for larger values. Indeed, Motif.P1.6.26 is the true (known) binding site. A follow-up experiment showed that the TF does not directly bind to Motif.P1.6.23. Hence, this motif is a candidate for a binding site of a co-factor (another TF) and needs further experimental validation.

Figure 4.3: Estimated functions \( \hat{f}_j \) of the two most stable motifs. Small vertical bar indicate original data.

4.4 Extensions

4.4.1 Generalized Additive Models

Conceptually, we can also apply the sparsity-smoothness penalty from Section 4.2 to generalized linear models (GLM) by replacing the residual sum of squares \( \|Y - \sum_{j=1}^{p} f_j\|_n^2 \) by the corresponding negative log-likelihood function. We illustrate the method for logistic regression where \( Y \in \{0, 1\} \). The negative log-likelihood as a function of the lin-
ear predictor $\eta$ and the response vector $Y$ is

$$\ell(\eta, Y) = -\frac{1}{n} \sum_{i=1}^{n} [Y_i \eta_i - \log\{1 + \exp(\eta_i)\}],$$

where $\eta_i = c + \sum_{j=1}^{p} f_j(x_i^{(j)})$. The estimator is defined as

$$\hat{c}, \hat{f}_1, \ldots, \hat{f}_p = \arg\min_{c \in \mathbb{R}, f_1, \ldots, f_p \in \mathcal{F}} \ell\left(c + \sum_{j=1}^{p} f_j, Y\right) + \sum_{j=1}^{p} J(f_j). \quad (4.4.1)$$

This has a similar form as (4.2.1) with the exception that we have to explicitly include a (non-penalized) intercept term $c$. Using the same arguments as in Section 4.2 leads to the fact that for twice continuously differentiable functions, the solution can be represented as a natural cubic spline and that (4.4.1) leads again to a Group Lasso problem. This can for example be minimized with the algorithm of Meier et al. (2008a). We illustrate the performance of the estimator in a small simulation study.

**Small Simulation Study**

Denote by $f : \mathbb{R}^p \to \mathbb{R}$ the true function of Example 2 in Section 4.3. We simulate the linear predictor $\eta$ as

$$\eta(X) = 1.5 \cdot (2 + f(X)),$$

where $X \in \mathbb{R}^p$ has the same distribution as in Example 2. The binary response $Y$ is then generated according to a Bernoulli distribution with probability $1/(1 + \exp(-\eta(X)))$, which results in a Bayes risk of approximately 0.17. The sample size $n$ is set to 100. The results for various model sizes $p$ are reported in Table 4.3 and Table 4.4. The performance of the two methods is quite similar. SSP has a slightly lower prediction error. Regarding model selection properties, SSP has fewer false positives at the cost of slightly fewer true positives.
Table 4.3: Results of different model sizes \( p \). Reported is the mean of the ratio of the prediction error of the two methods. SSP: Sparsity-Smoothness Penalty approach, boost: Boosting with smoothing splines. Standard deviations are given in parentheses.

<table>
<thead>
<tr>
<th>( p )</th>
<th>( PE_{SSP}/PE_{boost} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>250</td>
<td>0.93 (0.06)</td>
</tr>
<tr>
<td>500</td>
<td>0.96 (0.07)</td>
</tr>
<tr>
<td>1000</td>
<td>0.98 (0.05)</td>
</tr>
</tbody>
</table>

Table 4.4: Average values of the number of true (TP) and false (FP) positives. Standard deviations are given in parentheses.

<table>
<thead>
<tr>
<th>( p )</th>
<th>( TP_{SSP} ) (TP)</th>
<th>( FP_{SSP} ) (FP)</th>
<th>( TP_{boost} ) (TP)</th>
<th>( FP_{boost} ) (FP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>250</td>
<td>2.94 (0.71)</td>
<td>22.81 (10.56)</td>
<td>3.09 (0.78)</td>
<td>29.67 (14.91)</td>
</tr>
<tr>
<td>500</td>
<td>2.56 (0.82)</td>
<td>24.92 (12.47)</td>
<td>2.80 (0.82)</td>
<td>31.41 (17.28)</td>
</tr>
<tr>
<td>1000</td>
<td>2.36 (0.84)</td>
<td>26.45 (14.88)</td>
<td>2.61 (0.71)</td>
<td>33.69 (19.54)</td>
</tr>
</tbody>
</table>

4.4.2 Adaptivity

Similar to the Adaptive Lasso Zou (2006), we can also use different penalties for the different components, i.e. use a penalty of the form

\[
J(f_j) = \lambda_1 \sqrt{w_{1,j}} \|f_j\|_n + \lambda_2 w_{2,j} I(f_j),
\]

where the weights \( w_{1,j} \) and \( w_{2,j} \) are ideally chosen in a data-adaptive way. If an initial estimator \( \hat{f}_{j,\text{init}} \) is available, a choice would be to use

\[
w_{1,j} = \frac{1}{\|\hat{f}_{j,\text{init}}\|_n^{\gamma}}, \quad w_{2,j} = \frac{1}{I(\hat{f}_{j,\text{init}})^\gamma},
\]

for some \( \gamma > 0 \). The estimator can then be computed similarly as described in Section 4.2.2. This allows for different degrees of smoothness for different components.

We have applied the adaptive estimator to the simulation models of Section 4.3. In each simulation run we use weights (with \( \gamma = 1 \)) based on the ordinary sparsity-smoothness estimator. For comparison, we compute the ratio of the prediction error of the adaptive and the ordinary sparsity-smoothness estimator at each simulation run. The results are summarized in Table 4.5. Both the prediction error and
4.5. Conclusions

We present an estimator and algorithm for fitting sparse, high-dimensional generalized additive models. The estimator is based on a penalized likelihood. The penalty is new, as it allows for different regularization of the sparsity and the smoothness of the additive functions. It is exactly this combination which allows to derive oracle results for high-dimensional additive models. We also argue empirically that the inclusion of a smoothness-part into the penalty function yields much better results than having the sparsity-term only. Furthermore, we show that the optimization of the penalized likelihood can be written as a Group Lasso problem and hence, efficient coordinate-wise algorithms can be used which have provable numerical convergence properties.

We illustrate some empirical results for simulated and real data. Our new approach with the sparsity and smoothness penalty is never worse and sometimes substantially better than $L_2$Boosting for generalized additive model fitting (Bühlmann and Yu, 2003; Bühlmann and Hothorn, 2007). Furthermore, with an adaptive sparsity-smoothness

<table>
<thead>
<tr>
<th>Model</th>
<th>$PE_{SSP;adapt}/PE_{SSP}$</th>
<th>$TP$</th>
<th>$FP$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ex. 1</td>
<td>0.47 (0.13)</td>
<td>4.00 (0.00)</td>
<td>4.09 (4.63)</td>
</tr>
<tr>
<td>Ex. 2</td>
<td>0.63 (0.10)</td>
<td>3.31 (0.71)</td>
<td>6.12 (5.14)</td>
</tr>
<tr>
<td>Ex. 3 ($t = 0$)</td>
<td>0.53 (0.13)</td>
<td>4.00 (0.00)</td>
<td>4.64 (4.52)</td>
</tr>
<tr>
<td>Ex. 3 ($t = 1$)</td>
<td>0.63 (0.19)</td>
<td>3.81 (0.46)</td>
<td>5.04 (4.82)</td>
</tr>
<tr>
<td>Ex. 3 “high-freq” ($t = 0$)</td>
<td>0.87 (0.11)</td>
<td>2.28 (0.78)</td>
<td>2.98 (2.76)</td>
</tr>
<tr>
<td>Ex. 3 “high-freq” ($t = 1$)</td>
<td>0.91 (0.10)</td>
<td>1.69 (0.73)</td>
<td>2.59 (3.30)</td>
</tr>
<tr>
<td>Ex. 4 ($t = 0$)</td>
<td>0.77 (0.10)</td>
<td>11.21 (0.84)</td>
<td>8.18 (5.04)</td>
</tr>
<tr>
<td>Ex. 4 ($t = 1$)</td>
<td>0.88 (0.13)</td>
<td>9.73 (1.29)</td>
<td>7.93 (5.35)</td>
</tr>
</tbody>
</table>

Table 4.5: Results of the different simulation models. Reported is the mean of the ratio of the prediction error of the two methods and the average values of the number of true (TP) and false (FP) positives. $SSP;adapt$: Adaptive Sparsity-Smoothness Penalty approach, $SSP$: Ordinary Sparsity-Smoothness Penalty approach. Standard deviations are given in parentheses.

the number of false positives can be decreased by a good margin in all examples. The number of true positives gets slightly decreased in some examples.

4.5 Conclusions
penalty method, large additional performance gains are achieved. With the real data about motif regression for finding DNA-sequence motifs, one among two selected “stable” variables is known to be true, i.e. it corresponds to a known binding site of a transcription factor.

4.A Appendix

4.A.1 Proof of Proposition 4.1

Proof. Because of the additive structure of $f$ and the penalty, it suffices to analyze each component $f_j$, $j = 1, \ldots, p$ independently. Let $\hat{f}_1, \ldots, \hat{f}_p$ be a solution of (4.2.1) and assume that some or all $\hat{f}_j$ are not natural cubic splines with knots at $x_i^{(j)}$, $i = 1, \ldots, n$. By Theorem 2.2 in Green and Silverman (1994) we can construct natural cubic splines $\hat{g}_j$ with knots at $x_i^{(j)}$, $i = 1, \ldots, n$ such that

$$\hat{g}_j(x_i^{(j)}) = \hat{f}_j(x_i^{(j)})$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, p$. Hence

$$\| Y - \sum_{j=1}^p \hat{g}_j \|_n^2 = \| Y - \sum_{j=1}^p \hat{f}_j \|_n^2$$

and

$$\| \hat{g}_j \|_n^2 = \| \hat{f}_j \|_n^2.$$ 

But by Theorem 2.3 in Green and Silverman (1994), $I^2(\hat{g}_j) \leq I^2(\hat{f}_j)$. Therefore, the value in the objective function (4.2.1) can be decreased. Hence, the minimizer of (4.2.1) must lie in the space of natural cubic splines. \hfill \Box

4.A.2 Proof of Proposition 4.2

Proof. The first part follows because of the strict convexity of the objective function. Consider now the case $pK > n$. The (necessary and
sufficient) conditions for $\hat{\beta}$ to be a solution of the Group-Lasso problem (4.2.4) are (Yuan and Lin, 2006)

$$
\|
\nabla \beta_j \| = \lambda_1 \text{ for } \hat{\beta}_j \neq 0
$$

$$
\|
\nabla \beta_j \| \leq \lambda_1 \text{ for } \hat{\beta}_j = 0.
$$

Assume that there exist two solutions $\hat{\beta}^{(1)}$ and $\hat{\beta}^{(2)}$ such that for a component $j$ we have $\hat{\beta}_j^{(1)} = 0$ with $\|
\nabla \beta_j \| < \lambda_1$ but $\hat{\beta}_j^{(2)} \neq 0$. Because the set of all solutions is convex,

$$
\hat{\beta}_\rho = (1 - \rho)\hat{\beta}^{(1)} + \rho\hat{\beta}^{(2)}
$$

is also a minimizer for all $\rho \in [0, 1]$. By assumption $\hat{\beta}_{\rho,j} \neq 0$ and hence $\|
\nabla \beta_j \| = \lambda_1$ for all $\rho \in (0, 1)$. Hence, it holds for $g(\rho) = \|
\nabla \beta_j \|$ that $g(0) < \lambda_1$ and $g(\rho) = \lambda_1$ for all $\rho \in (0, 1)$. But this is a contradiction to the fact that $\|
\nabla \beta_j \|$ is continuous. Hence, a non-active (i.e. zero) component $j$ with $\|
\nabla \beta_j \| < \lambda_1$ can not be active (i.e. non-zero) in any other solution. □
Assigning significance in high-dimensional regression models is challenging. Most computationally efficient selection algorithms cannot guard against inclusion of noise variables. Asymptotically valid p-values are not available. An exception is a recent proposal by Wasserman and Roeder (2008) which splits the data into two parts. The number of variables is then reduced to a manageable size using the first split, while classical variable selection techniques can be applied to the remaining variables, using the data from the second split. This yields asymptotic error control under minimal conditions. It involves, however, a one-time random split of the data. Results are sensitive to this arbitrary choice: it amounts to a “p-value lottery” and makes it difficult to reproduce results. Here, we show that inference across multiple random splits can be aggregated, while keeping asymptotic control over the inclusion of noise variables. In addition, the proposed aggregation is shown to improve power, while reducing the number of falsely selected variables substantially.
5.1 Introduction

The problem of high-dimensional variable selection has received tremendous attention in the last decade. Sparse estimators like the Lasso (Tibshirani, 1996) and extensions thereof (Zou, 2006; Meinshausen, 2007) have been shown to be very powerful because they are suitable for high-dimensional data sets and because they lead to sparse, interpretable results.

In the usual work-flow for high-dimensional variable selection problems, the user sets potential tuning parameters to their prediction optimal values and uses the resulting estimator as the final result. In the classical low-dimensional setup, some error control based on p-values is a widely used standard in all areas of sciences. So far, p-values were not available in high-dimensional situations, except for the proposal of Wasserman and Roeder (2008). An ad-hoc solution for assigning relevance is to use the bootstrap to analyze the stability of the selected predictors and to focus on those which are selected most often (or even always). Bach (2008) shows for the Lasso that this leads to a consistent model selection procedure under fewer restrictions than for the non-bootstrap case.

More recently, some progress has been achieved to obtain error control (Wasserman and Roeder, 2008; Meinshausen and Bühlmann, 2008). Here, we build upon the approach of Wasserman and Roeder (2008) and show that an extension of their “screen and clean” algorithm leads to a more powerful variable selection procedure. Moreover, desired error rates can be controlled, while Wasserman and Roeder (2008) focus on variable selection rather than assigning significance via p-values.

This chapter is organized as follows. We discuss the single-split method of Wasserman and Roeder (2008) briefly in Section 5.2, showing that results can strongly depend on the arbitrary choice of a random sample-splitting. We propose a multi-split method, removing this dependence. In Section 5.3 we prove error control of the multi-split method under identical conditions as in Wasserman and Roeder (2008), and we show in Section 5.4 numerically for simulated and real data sets that the method is more powerful than the single-split version while reducing substantially the number of false discoveries. Some possible extensions of the proposed methodology are outlined in Section 5.5.
5.2 Sample-Splitting and High-Dimensional Variable Selection

We consider the usual high-dimensional linear regression setup with a response vector $Y = (Y_1, \ldots, Y_n)$ and an $n \times p$ fixed design matrix $X$ such that

$$Y = X\beta + \varepsilon,$$

where $\varepsilon = (\varepsilon_1, \ldots \varepsilon_n)$ is a random error vector with $\varepsilon_i$ iid $\mathcal{N}(0, \sigma^2)$ and $\beta \in \mathbb{R}^p$ is the parameter vector. Extensions to other models are outlined in Section 5.5.

Denote by

$$S = \{j; \beta_j \neq 0\}$$

the set of active predictors and similarly by $N = S^c = \{j; \beta_j = 0\}$ the set of noise variables. Our goal is to assign p-values for the null-hypotheses $H_{0,j} : \beta_j = 0$ versus $H_{A,j} : \beta_j \neq 0$ and to infer the set $S$ from a set of $n$ observations $(X_i, Y_i)$, $i = 1, \ldots, n$. We allow for potentially high-dimensional designs, i.e. $p \gg n$. This makes statistical inference very challenging. An approach proposed by Wasserman and Roeder (2008) is to split the data into two parts, reducing the dimensionality of predictors on one part to a manageable size of predictors (keeping the important variables with high probability), and then to assign p-values and making a final selection on the second part of the data, using classical least squares estimation.

5.2.1 The Single-Split Method

The procedure of Wasserman and Roeder (2008) relies on sample-splitting, performing variable selection and dimensionality reduction on one part of the data and classical significance testing on the remaining part. The data are splitted randomly into two disjoint groups $D_{in} = (X_{in}, Y_{in})$ and $D_{out} = (X_{out}, Y_{out})$ of equal size. Let $\tilde{S}$ be a variable selection or screening procedure which estimates the set of active predictors. Abusing notation slightly, we also denote by $\tilde{S}$ the set of selected predictors. Then variable selection and dimensionality reduction is based on $D_{in}$, i.e. we apply $\tilde{S}$ only on $D_{in}$. This includes the selection of potential tuning parameters involved in $\tilde{S}$. The idea is to break down
the large number \( p \) of potential predictor variables to a smaller number \( k \ll p \) with \( k \) at most a fraction of \( n \) while keeping all relevant variables. The regression coefficients and the corresponding p-values \( \tilde{P}_1, \ldots, \tilde{P}_p \) of the selected predictors are determined based on \( D_{out} \) by using ordinary least squares estimation on the set \( \tilde{S} \) and setting \( \tilde{P}_j = 1 \) for all \( j \notin \tilde{S} \). If the selected model \( \tilde{S} \) contains the true model \( S \), i.e. \( \tilde{S} \supseteq S \), the p-values based on \( D_{out} \) are unbiased. Finally, each p-value \( \tilde{P}_j \) is adjusted by a factor \( |\tilde{S}| \) to correct for the multiplicity of the testing problem.

The selected model is given by all variables in \( \tilde{S} \) for which the adjusted p-value is below a cutoff \( \alpha \in (0, 1) \),

\[
\hat{S}_{single} = \left\{ j \in \tilde{S} : \tilde{P}_j |\tilde{S}| \leq \alpha \right\}.
\]

Under suitable assumptions discussed later, this yields asymptotic control against inclusion of variables in \( N \) (false positives) in the sense that

\[
\limsup_{n \to \infty} \mathbb{P} \left[ |N \cap \hat{S}_{single}| \geq 1 \right] \leq \alpha,
\]

i.e. control of the family-wise error rate. The method is easy to implement and yields the asymptotic control under weak assumptions. The single-split method relies, however, on an arbitrary split into \( D_{in} \) and \( D_{out} \). Results can change drastically if this split is chosen differently. This in itself is unsatisfactory since results are not reproducible.

### 5.2.2 The New Multi-Split Method

An obvious alternative to a single arbitrary split is to divide the sample repeatedly. For each split we end up with a set of p-values. It is not obvious, though, how to combine and aggregate the results. Here, we give a possible answer. We will later show empirically that, maybe unsurprisingly, the resulting procedure is more powerful than the single-split method. The multi-split method also makes results reproducible, at least approximately if the number of random splits is chosen to be very large.

The multi-split method uses the following procedure:

For \( b = 1, \ldots, B \):
1. Randomly split the original data into two disjoint groups $D^{(b)}_{in}$ and $D^{(b)}_{out}$ of equal size.

2. Using only $D^{(b)}_{in}$, estimate the set of active predictors $\tilde{S}^{(b)}$.

3. (a) Using only $D^{(b)}_{out}$, fit the selected variables in $\tilde{S}^{(b)}$ with ordinary least squares and calculate the corresponding p-values $\tilde{P}^{(b)}_j$ for $j \in \tilde{S}^{(b)}$.

(b) Set the remaining p-values to 1, i.e.

$$\tilde{P}^{(b)}_j = 1, \ j \notin \tilde{S}^{(b)}.$$ 

4. Define the adjusted (non-aggregated) p-values as

$$P^{(b)}_j = \min \left( \tilde{P}^{(b)}_j / |\tilde{S}^{(b)}|, 1 \right), \ j = 1, \ldots, p \quad (5.2.1)$$

Finally, aggregate over the $B$ p-values $P^{(b)}_j$, as discussed below.

The procedure leads to a total of $B$ p-values for each predictor $j = 1, \ldots, p$. It will turn out that suitable summary statistics are quantiles. For $\gamma \in (0, 1)$ define

$$Q_j(\gamma) = q_{\gamma} \left( \{ P^{(b)}_j / \gamma; \ b = 1, \ldots, B \} \right), \quad (5.2.2)$$

where $q_{\gamma}(\cdot)$ is the (empirical) $\gamma$-quantile function.

A p-value for each predictor $j = 1, \ldots, p$ is then given by $Q_j(\gamma)$, for any fixed $0 < \gamma < 1$. We will show in Section 5.3 that this is an asymptotically correct p-value, adjusted for multiplicity. A typical choice for $\gamma$ would be to use the median, i.e. $\gamma = 0.5$. The value $Q_j(0.5)$ corresponds to twice the median value among $P^{(b)}_j$, $b = 1, \ldots, B$. As we will see later, the choice $\gamma = 0.5$ can be too restrictive. A proper selection of $\gamma$ may be difficult. Error control is not guaranteed anymore if we search for the best value of $\gamma$.

We propose to use instead an adaptive version which selects a suitable value of the quantile based on the data. Let $\gamma_{\text{min}} \in (0, 1)$ be a lower bound for $\gamma$, typically 0.05, and define

$$P_j = (1 - \log \gamma_{\text{min}}) \inf_{\gamma \in (\gamma_{\text{min}}, 1)} Q_j(\gamma). \quad (5.2.3)$$
Figure 5.1: Left: a histogram of adjusted p-values $P_j^{(b)}$ for the selected variable in the motif regression data example of Section 5.4.3. The single split method picks randomly one of these p-values (a "p-value lottery") and rejects if it is below $\alpha$. For the multi-split method, we reject if and only if the empirical distribution function of the adjusted p-values crosses the broken line (which is $f(p) = \max\{0.05, (3.996/\alpha)p\}$) for some $p \in (0, 1)$. This bound is shown as a broken line for $\alpha = 0.05$. For the given example, the bound is indeed exceeded and the variable is thus selected.

The extra correction factor $1 - \log \gamma_{\min}$ ensures that the family-wise error rate remains controlled at level $\alpha$ despite of the adaptive search for the best quantile, see Section 5.3. For the recommended choice of $\gamma_{\min} = 0.05$, this factor is upper bounded by 4; in fact, $1 - \log(0.05) \approx 3.996$. The selected subset consists now of all variables whose p-value is below a specified significance level $\alpha \in (0, 1)$,

$$\hat{S}_{multi} = \{j \in \tilde{S} : P_j \leq \alpha\}.$$  

Figure 5.1 shows an example. The left panel contains the histogram of the adjusted p-values $P_j^{(b)}$ for $b = 1, \ldots, B$ of the selected variable in the real data example in Section 5.4.3. The single split method is equivalent to picking one of these p-values randomly and selecting the variable if this randomly picked p-value is sufficiently small. To avoid this "p-value lottery", the multi-split method computes the empirical distribution of all p-values $P_j^{(b)}$ for $b = 1, \ldots, B$ and rejects if the empirical distribution crosses the broken line in the right panel of Figure 5.1. A short derivation of the latter is as follows. Variable $j$ is selected if and only
if $P_j \leq \alpha$, which happens if and only if there exists some $\gamma \in (0.05, 1)$ such that $Q_j(\gamma) \leq \alpha/(1 - \log 0.05) \approx \alpha/3.996$. Equivalently, using definition (5.2.2), the $\gamma$-quantile of the adjusted p-values, $q_\gamma(P_j^{(b)})$, has to be smaller than or equal to $\alpha \gamma/3.996$. This in turn is equivalent to the event that the empirical distribution of the adjusted p-values $P_j^{(b)}$ for $b = 1, \ldots, B$ is crossing above the bound $f(p) = \max\{0.05, (3.996/\alpha)p\}$ for some $p \in (0, 1)$. This bound is shown as a broken line in the right panel of Figure 5.1.

Besides better reproducibility and asymptotic family-wise error control, the multi-split version is, maybe unsurprisingly, more powerful than the single-split selection method. Before showing numerical evidence, we show that the proposed method provides indeed the desired error control.

5.3 Error Control and Consistency

5.3.1 Assumptions

To achieve asymptotic error control, a few assumptions are made in Wasserman and Roeder (2008) regarding the crucial requirements for the variable selection procedure $\tilde{S}$.

(A1) \textit{Screening property}: $\lim_{n \to \infty} \mathbb{P}\left[\tilde{S} \supseteq S\right] = 1$.

(A2) \textit{Sparsity property}: $|\tilde{S}| < n/2$.

The \textit{Screening property} (A1) ensures that all relevant variables are retained. Irrelevant noise variables are allowed to be selected, too, as long as there are not too many as required by the \textit{Sparsity property} (A2). A violation of the sparsity property would make it impossible to apply classical tests on the retained variables.

The Lasso (Tibshirani, 1996) is an important example which satisfies (A1) and (A2) under appropriate conditions discussed in Meinshausen and Bühlmann (2006), Zhao and Yu (2006), van de Geer (2008), Meinshausen and Yu (2009) and Bickel \textit{et al.} (2008). The adaptive Lasso
(Zou, 2006; Zhang and Huang, 2008) satisfies (A1) and (A2) as well under suitable conditions. Other examples include, assuming appropriate conditions, $L_2$ Boosting (Friedman, 2001; Bühlmann, 2006), orthogonal matching pursuit (Tropp and Gilbert, 2007) or Sure Independence Screening (Fan and Lv, 2008).

We will typically use the Lasso (and extensions thereof) as screening method. Other algorithms would be possible. Wasserman and Roeder (2008) studied various scenarios under which these two properties are satisfied for the Lasso, depending on the choice of the regularization parameter. We refrain from repeating these and similar arguments, just working on the assumption that we have a selection procedure $\hat{S}$ at hand which satisfies both the Screening property and the Sparsity property.

### 5.3.2 Error Control

We proposed two versions for multiplicity-adjusted p-values. One is $Q_j(\gamma)$ as defined in (5.2.2) which relies on a choice of $\gamma \in (0,1)$. The second is the adaptive version $P_j$ defined in (5.2.3) which makes an adaptive choice of $\gamma$. We show that both quantities are multiplicity-adjusted p-values providing asymptotic error control.

**Theorem 5.1.** Assume (A1) and (A2). Let $\alpha, \gamma \in (0,1)$. If the null-hypothesis $H_{0,j} : \beta_j = 0$ gets rejected whenever $Q_j(\gamma) \leq \alpha$, the family-wise error rate is asymptotically controlled at level $\alpha$, i.e.

$$
\lim_{n \to \infty} \sup \mathbb{P} \left[ \min_{j \in N} Q_j(\gamma) \leq \alpha \right] \leq \alpha,
$$

where $\mathbb{P}$ is with respect to the data sample and the statement holds for any of the $B$ random sample-splits.

Theorem 5.1 is valid for any pre-defined value of the quantile $\gamma$. However, the adjusted p-values $Q_j(\gamma)$ involve the somehow arbitrary choice of $\gamma$ which might pose a problem for practical applications. We therefore proposed the adjusted p-values $P_j$ which search for the optimal value of $\gamma$ adaptively.
5.3. Error Control and Consistency

Theorem 5.2. Assume (A1) and (A2). Let $\alpha \in (0,1)$. If the null-hypothesis $H_{0,j}: \beta_j = 0$ gets rejected whenever $P_j \leq \alpha$, the family-wise error rate is asymptotically controlled at level $\alpha$, i.e.

$$\limsup_{n \to \infty} \mathbb{P} \left[ \min_{j \in N} P_j \leq \alpha \right] \leq \alpha,$$

where the probability $\mathbb{P}$ is as in Theorem 5.1.

We comment briefly on the relation between the proposed adjustment to false discovery rate (Benjamini and Hochberg, 1995; Benjamini and Yekutieli, 2001) or family-wise error (Holm, 1979) controlling procedures. While we provide a family-wise error control and as such use union bound corrections as in Holm (1979), the definition of the adjusted p-values (5.2.3) and its graphical representation in Figure 5.1 are vaguely reminiscent of the false discovery rate procedure, rejecting hypotheses if and only if the empirical distribution of p-values crosses a certain linear bound. The empirical distribution in (5.2.3) is only taken for one predictor variable, though, which is either in $S$ or $N$. This would correspond to a multiple testing situation where we are testing a single hypothesis with multiple statistics.

5.3.3 Model Selection Consistency

If we let level $\alpha = \alpha_n \to 0$ for $n \to \infty$, the probability of falsely including a noise variable vanishes because of the preceding results. In order to get the property of consistent model selection, we have to analyze the asymptotic behavior of the power. It turns out that this property is inherited from the single-split method.

Corollary 5.1. Let $\hat{S}_{\text{single}}$ be the selected model of the single-split method. Assume that $\alpha_n \to 0$ can be chosen for $n \to \infty$ at a rate such that $\lim_{n \to \infty} \mathbb{P} [ \hat{S}_{\text{single}} = S ] = 1$. Then, for any $\gamma_{\min}$ (see (5.2.3)), the multi-split method is also model selection consistent for a suitable sequence $\alpha_n$, i.e. for $\hat{S}_{\text{multi}} = \{ j \in \hat{S}; P_j \leq \alpha_n \}$ it holds that

$$\lim_{n \to \infty} \mathbb{P} [ \hat{S}_{\text{multi}} = S ] = 1.$$
Wasserman and Roeder (2008) discuss conditions which ensure that 
\( \lim_{n \to \infty} \mathbb{P}[\hat{S}_{\text{single}} = S] = 1 \) for various variable selection methods such as the Lasso or some forward variable selection scheme.

The reverse of the Corollary above is not necessarily true. The multi-split method can be consistent if the single-split method is not. One could imagine for example a scenario, where the p-values of a variable \( j \in S \) satisfy for some \( c \in (0, 1) \) that \( \limsup_{n \to \infty} \mathbb{P}[P_j^{(b)} \leq c] < 1 \), where the probability is with respect to both the data and the random split-point. In this case, the single-split method cannot be consistent, as there is a positive probability of variable \( j \in S \) not being selected when \( \alpha_n \to 0 \). On the other hand, some quantiles of the distribution of \( P_j^{(b)} \) under repeated random split-point selection can converge to 0, which would make the multi-split method consistent even though the single-split method is not. We refrain from going into more details here and rather show with numerical results that the multi-split method is indeed more powerful than the single-split analogue.

### 5.4 Numerical Results

In this section we compare the empirical performance of the different estimators on simulated and real data sets. Simulated data allow a thorough evaluation of the model selection properties. The real data set shows that we can find signals in data with our proposed method that would not be picked up by the single-split method. We use a default value of \( \alpha = 0.05 \) everywhere.

#### 5.4.1 Simulations

We use the following simulation settings:

(A) Simulated data set with \( n = 100, p = 100 \) and a design matrix coming from a centered multivariate normal distribution with covariance structure \( \text{Cov}(X_j, X_k) = 0.5^{|j-k|} \).

(B) As (A) but with \( n = 100 \) and \( p = 1000 \).
5.4. Numerical Results

Real data set with $n = 71$ and $p = 4088$ for the design matrix $X$ and artificial response $Y$.

The data set in (C) is from gene expression measurements in Bacillus Subtilis. The $p = 4088$ predictor variables are log-transformed gene expressions and there is a response measuring the logarithm of the production rate of riboflavin in Bacillus Subtilis. The data is kindly provided by DSM (Switzerland). As the true variables are not known, we consider a linear model with design matrix from real data and simulating a sparse parameter vector $\beta$ as follows. In each simulation run, a new parameter vector $\beta$ is created by either “uniform” or “varying-strength” sampling. Under “uniform” sampling, $|S|$ randomly chosen components of $\beta$ are set to 1 and the remaining $p - |S|$ components to 0. Under “varying-strength” sampling, $|S|$ randomly chosen components of $\beta$ are set to values $1, \ldots, |S|$. The error variance $\sigma^2$ is adjusted such that the signal to noise ratio (SNR) is maintained at a desired level at each simulation run. We perform 50 simulations for each setting. As in Wasserman and Roeder (2008) we use normal approximations for the calculation of p-values.

The sample-splitting is done such that the model is trained on a data set of size $\lfloor \frac{n}{2} \rfloor$ and the p-values are calculated on the remaining data set. This slightly unbalanced scheme prevents us from situations where the full model might be selected on the first data set. Calculations of p-values would not be possible on the remaining data in such a situation. We use a total of $B = 50$ sample-splits at each simulation run.

We compare the average number of true positives and the family-wise error rate (FWER) for the single- and multi-split methods for all three simulation settings (A)–(C) and vary in each the SNR to 0.25, 1, 4 and 16 (which corresponds to population $R^2$ values of 0.2, 0.5, 0.8 and 0.94, respectively). The number $|S|$ of relevant variables is either 5 or 10. As initial variable selection or screening method $\tilde{S}$ we use three approaches, which are all based on the Lasso (Tibshirani, 1996). The first one, denoted by $\tilde{S}_{\text{fixed}}$, uses the Lasso and selects those $\lfloor n/6 \rfloor$ variables which appear most often in the regularization path when varying the penalty parameter. The second method, $\tilde{S}_{\text{cv}}$, uses the Lasso with penalty parameter chosen by 10-fold cross-validation and selecting the variables whose corresponding estimated regression coefficients are different from zero. The third method, $\tilde{S}_{\text{adap}}$, is the adaptive Lasso of
Zou (2006) where regularization parameters are chosen based on 10-fold cross-validation with the Lasso solution used as initial estimator for the adaptive Lasso. The selected variables are again the ones whose corresponding estimated regression parameters are different from zero.

Results are shown in Figures 5.2 and 5.3 for both the single-split method and the multi-split method with the default setting $\gamma_{\text{min}} = 0.05$. Using the multi-split method, the average number of true positives (the variables in $S$ which are selected) is typically slightly increased while the FWER (the probability of including variables in $N$) is reduced sharply. The single-split method has often a FWER above the level $\alpha = 0.05$ at which it is asymptotically controlled while for the multi-split method the FWER is above the nominal level only in few scenarios. The asymptotic control seems to give a good control in finite sample settings with the multi-split method, yet not the single-split method. Even though the multi-split method is more conservative than the single-split method (having substantially lower FWER), the number of true discoveries is often increased. We note that for data (C), with $p = 4088$, and in general for low SNR, the number of true positives is low since we control the very stringent family-wise error criterion at $\alpha = 0.05$ significance level. As an alternative, controlling less conservative error measures would be possible and is discussed in Section 5.5.

We also experimented with using the value of $Q_j(\gamma)$ directly as an adjusted p-value, without the adaptive choice of $\gamma$ but using a fixed value $\gamma = 0.5$ instead, i.e. looking at twice the median value of all p-values across multiple data splits. The results were not as convincing as for the adaptive choice and we recommend the adaptive version with $\gamma_{\text{min}} = 0.05$ as a good default choice.

5.4.2 Comparisons with Adaptive Lasso

Next, we compare the multi-split selector with the adaptive Lasso (Zou, 2006). We have used the adaptive Lasso previously as a variable selection method in our proposed multi-split method. The adaptive Lasso is usually employed on its own. There are a few choices to make when using the adaptive Lasso. We use the same seemingly default choices as previously. The initial estimator is obtained as the Lasso solution with a 10-fold CV-choice of the penalty parameter. The adaptive Lasso
Increasing SNR is indicated by increasing symbol size.

**Figure 5.2:** Simulation results for setting (A) in the top and (B) in the bottom row. Average number of true positives vs. the family-wise error rate (FWER) for the single split method (“S”) against the multi-split version (“M”). FWER is controlled (asymptotically) at $\alpha = 0.05$ for both methods and this value is indicated by a broken vertical line. From left to right are results for $S_{\text{fixed}}$, $S_{\text{cv}}$ and $S_{\text{adap}}$. Results of a unique setting of SNR, sparsity and design are joined by a line, which is solid if the coefficients follow the “uniform” sampling and broken otherwise. Increasing SNR is indicated by increasing symbol size.

**Figure 5.3:** Results of simulation setting (C). See caption of Figure 5.2 for details.
penalty is also obtained by 10-fold CV.

Despite desirable asymptotic consistency properties (Huang et al., 2008), the adaptive Lasso does not offer error control in the same way as Theorem 5.1 does for the multi-split method. In fact, the FWER (the probability of selecting at least one noise variable) is very close to 1 with the adaptive Lasso in all the simulations we have seen. In contrast, our multi-split method offers asymptotic control, which was seen to be very well matched by the empirical FWER in the vicinity of \( \alpha = 0.05 \). Table 5.1 shows the simulation results for the multi-split method using \( \tilde{S}_{\text{adap}} \) and the adaptive Lasso on its own side by side for a simulation setting with \( n = 100 \), \( p = 200 \) and the same settings as in (A) and (B) otherwise. The adaptive Lasso selects roughly 20 noise variables (out of \( p = 200 \) variables), even though the number of truly relevant variables is just 5 or 10. The average number of false positives is at most 0.04 and often simply 0 with the proposed multi-split method.

<table>
<thead>
<tr>
<th>Uniform Sampling</th>
<th>SNR</th>
<th>E(True Positives)</th>
<th>E(False Positives)</th>
<th>P(False Positives &gt; 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Multi Split</td>
<td>Adaptive Lasso</td>
<td>Multi Split</td>
</tr>
<tr>
<td>NO</td>
<td>10</td>
<td>0.00</td>
<td>2.30</td>
<td>0</td>
</tr>
<tr>
<td>NO</td>
<td>10</td>
<td>0.58</td>
<td>6.32</td>
<td>0</td>
</tr>
<tr>
<td>NO</td>
<td>10</td>
<td>4.14</td>
<td>8.30</td>
<td>0</td>
</tr>
<tr>
<td>NO</td>
<td>10</td>
<td>7.20</td>
<td>9.42</td>
<td>0.02</td>
</tr>
<tr>
<td>YES</td>
<td>10</td>
<td>0.02</td>
<td>2.52</td>
<td>0</td>
</tr>
<tr>
<td>YES</td>
<td>10</td>
<td>0.10</td>
<td>7.46</td>
<td>0.02</td>
</tr>
<tr>
<td>YES</td>
<td>10</td>
<td>2.14</td>
<td>9.96</td>
<td>0</td>
</tr>
<tr>
<td>YES</td>
<td>10</td>
<td>9.92</td>
<td>10.00</td>
<td>0.04</td>
</tr>
<tr>
<td>NO</td>
<td>5</td>
<td>0.06</td>
<td>1.94</td>
<td>0</td>
</tr>
<tr>
<td>NO</td>
<td>5</td>
<td>1.50</td>
<td>3.86</td>
<td>0.02</td>
</tr>
<tr>
<td>NO</td>
<td>5</td>
<td>3.52</td>
<td>4.58</td>
<td>0.02</td>
</tr>
<tr>
<td>NO</td>
<td>5</td>
<td>4.40</td>
<td>4.98</td>
<td>0</td>
</tr>
<tr>
<td>YES</td>
<td>5</td>
<td>0.02</td>
<td>2.22</td>
<td>0</td>
</tr>
<tr>
<td>YES</td>
<td>5</td>
<td>0.82</td>
<td>4.64</td>
<td>0.02</td>
</tr>
<tr>
<td>YES</td>
<td>5</td>
<td>3.90</td>
<td>5.00</td>
<td>0</td>
</tr>
<tr>
<td>YES</td>
<td>5</td>
<td>5.00</td>
<td>5.00</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.1: Comparing the multi-split method with CV-Lasso selection, \( \tilde{S}_{\text{adap}} \), with the selection made when using the adaptive Lasso and a CV-choice of the involved penalty parameters for a setting with \( n = 100 \) and \( p = 200 \).

There is clearly a price to pay for controlling the family-wise error rate. Our proposed multi-split method detects on average less truly relevant variables than the adaptive Lasso. For very low SNR, the difference is most pronounced. The multi-split method selects in general neither correct nor wrong variables for SNR = 0.25, while the adaptive Lasso averages between 2 to 3 correct selections, among 9-12 wrong
selections. Depending on the objectives of the study, one would prefer either of the outcomes. For larger SNR, the multi-split method detects almost as many truly important variables as the adaptive Lasso, while still reducing the number of falsely selected variables from 20 or above to roughly 0.

The multi-split method seems hence beneficial in settings where the cost of making an erroneous selection is rather high. For example, expensive follow-up experiments are usually required to validate results in bio-medical applications and a stricter error control will place more of the available resources into experiments which are likely to be successful.

5.4.3 Real Data

We apply the multi-split method to a real data set about motif regression (Conlon et al., 2003). For a total of $n = 287$ DNA segments we have the binding intensity of a protein to each of the segments. These will be our response values $Y_1, \ldots, Y_n$. Moreover, for $p = 195$ candidate words (“motifs”) we have scores $x_{ij}$ which measure how well the $j$th motif is represented in the $i$th DNA sequence. The motifs are typically 5–15bp long candidates for the true binding site of the protein. The hope is that the true binding site is in the list of significant variables showing the strongest relationship between the motif score and the binding intensity. Using a linear model with $\tilde{S}_{\text{adap}}$, the multi-split method identifies one predictor variable at the 5% significance level. The single-split method is not able to identify a single significant predictor. In view of the asymptotic error control and the empirical results in Section 5.4 there is substantial evidence that the selected variable corresponds to a true binding site. For this specific application it seems desirable to pursue a conservative approach with low FWER. As mentioned above, we could control other, less conservative error measures as discussed in Section 5.5.

5.5 Extensions

Due to the generic nature of our proposed methodology, extensions to any situation where (asymptotically valid) p-values $\tilde{P}_j$ for hypotheses
$H_{0,j}$ ($j = 1, \ldots, p$) are available are straightforward. An important class of examples are generalized linear models (GLMs) or Gaussian Graphical Models. The dimensionality reduction step would typically involve some form of shrinkage estimation. An example for Gaussian Graphical Models would be the recently proposed “Graphical Lasso” (Friedman et al., 2008). The second step would rely on classical (e.g. likelihood ratio) tests applied to the selected submodel, analogous to the methodology proposed for linear regression.

We have shown above how the family-wise error rate (FWER) can be controlled in a multi-split approach. We could control other error rates. Take for example the false-discovery rate (FDR). By directly aggregating the p-values $\tilde{P}_j^{(b)}$ (i.e. without applying a Bonferroni correction for the size of the selected model $\tilde{S}^{(b)}$), we can define “unadjusted” p-values $Q_j(\gamma)$ and $P_j$ analogously to (5.2.2) and (5.2.3) with the property that

$$\limsup_{n \to \infty} \mathbb{P}[Q_j(\gamma) \leq \alpha] \leq \alpha,$$

and

$$\limsup_{n \to \infty} \mathbb{P}[P_j \leq \alpha] \leq \alpha.$$

Now, by using a multiplicity correction as in Benjamini and Hochberg (1995) or Benjamini and Yekutieli (2001) we can asymptotically control the FDR. However, the multiplicity correction involves here (for the smallest p-value) a factor $p$, which can be much larger than the average value of $|\tilde{S}^{(b)}|$ in high-dimensional settings, and it might be more powerful to work with the proposed FWER-controlling procedure.

In some settings, control of FWER at $\alpha = 0.05$ is too conservative. If the control of FDR, as alluded to above, is also too conservative, our method can easily be adjusted to control the expected number of false rejections. Take as an example the adjusted p-value $P_j$, defined in (5.2.3). Variable $j$ is rejected if and only if $P_j \leq \alpha$. (For the following, assume that adjusted p-values, as defined in (5.2.1), are not capped at 1. This is a technical detail only as it does not modify the proposed FWER-controlling procedure.) Rejecting variable $j$ if and only if $P_j \leq \alpha$ controls FWER at level $\alpha$. Instead, one can reject variables if and only if $P_j/K \leq \alpha$, where $K > 1$ is a correction factor. Call the number of falsely rejected variables $V$,

$$V = \sum_{j \in N} 1\{P_j/K \leq \alpha\}.$$
5.6 Discussion

Then the expected number of false positives is controlled at level $\alpha K$, i.e. \( \limsup_{n \to \infty} \mathbb{E}[V] \leq \alpha K \). A proof of this follows directly from the proof of Theorem 5.2. Of course, we can equivalently set \( k = \alpha K \) and obtain a control \( \limsup_{n \to \infty} \mathbb{E}[V] \leq k \). For example, setting \( k = 1 \) offers a much less conservative error control, if so desired, than control of the family-wise error rate.

5.6 Discussion

We proposed a multi-sample-split method for assigning statistical significance and constructing conservative p-values for hypothesis testing for high-dimensional problems where the number of predictor variables may be much larger than sample size. Our method is an extension of the single-split method of Wasserman and Roeder (2008). Combining the results of multiple data-splits, based on quantiles as summary statistics, improves reproducibility compared to the single-split method. The multi-split method shares with the single-split method the property of asymptotic error control and model selection consistency. We argue empirically that the multi-split method usually selects much fewer false positives than the single-split method while the number of true positives is slightly increased. The method is very generic and can be used for a broad spectrum of error controlling procedures in multiple testing, including linear and generalized linear models.

5.A Proofs

Proof of Theorem 5.1. For technical reasons we define

\[
K_j^{(b)} = P_j^{(b)} 1\{S \subseteq \tilde{S}^{(b)}\} + 1\{S \not\subseteq \tilde{S}^{(b)}\}.
\]

\( K_j^{(b)} \) are the adjusted p-values if the estimated active set contains the true active set. Otherwise, all p-values are set to 1. Because of assumption (A1) and for fixed \( B \), \( \mathbb{P}[K_j^{(b)} = P_j^{(b)} \text{ for all } b = 1, \ldots, B] \) on a set \( A_n \) with \( \mathbb{P}[A_n] \to 1 \). Therefore, we can define all the quantities involving \( P_j^{(b)} \) also with \( K_j^{(b)} \), and it is sufficient to show under this slightly
altered procedure that

\[ P[\min_{j \in N} Q_j(\gamma) \leq \alpha] \leq \alpha. \]

In particular we can omit here the limes superior.

Define

\[ \pi_j(\alpha) = \frac{1}{B} \sum_{b=1}^{B} 1\{K_j^{(b)} / \gamma \leq \alpha\}. \]

Note that the events \( \{Q_j(\gamma) \leq \alpha\} \) and \( \{\pi_j(\alpha) \geq \gamma\} \) are equivalent. Hence,

\[ P\left[ \min_{j \in N} Q_j(\gamma) \leq \alpha \right] \leq \sum_{j \in N} \mathbb{E}\left[ 1\{Q_j(\gamma) \leq \alpha\} \right] = \sum_{j \in N} \mathbb{E}\left[ 1\{\pi_j(\alpha) \geq \gamma\} \right]. \]

Using that \( x/y \geq 1\{x \geq y\} \) for all \( x, y > 0 \),

\[ \sum_{j \in N} \mathbb{E}\left[ 1\{\pi_j(\alpha) \geq \gamma\} \right] \leq \frac{1}{\gamma} \sum_{j \in N} \mathbb{E}[\pi_j(\alpha)]. \]

By definition of \( \pi_j(\cdot) \),

\[ \frac{1}{\gamma} \sum_{j \in N} \mathbb{E}[\pi_j(\alpha)] = \frac{1}{\gamma} \frac{1}{B} \sum_{b=1}^{B} \sum_{j \in N \cap \tilde{S}^{(b)}} \mathbb{E}\left[ 1\{K_j^{(b)} \leq \alpha \gamma\} \right]. \]

Moreover,

\[ \mathbb{E}\left[ 1\{K_j^{(b)} \leq \alpha \gamma\} \right] \leq \mathbb{P}\left[ P_j^{(b)} \leq \alpha \gamma \mid S \subseteq \tilde{S}^{(b)} \right] = \frac{\alpha \gamma}{|\tilde{S}(b)|}. \]

This is a consequence of the uniform distribution of \( \tilde{F}_j^{(b)} \) given \( S \subseteq \tilde{S}(b) \). Summarizing these results we get

\[ P\left[ \min_{j \in N} Q_j(\gamma) \leq \alpha \right] \leq \alpha. \]

Proof of Theorem 5.2. As in the proof of Theorem 5.1 we will work with \( K_j^{(b)} \) instead of \( P_j^{(b)} \). Analogously, instead of \( \tilde{F}_j^{(b)} \) we work with \( \tilde{K}_j^{(b)} \).

\[ \square \]
For any $\tilde{K}_j^{(b)}$ with $j \in N$ and $\alpha \in (0, 1)$,

$$
\mathbb{E} \left[ \frac{1 \{ \tilde{K}_j^{(b)} \leq \alpha \gamma \} }{\gamma} \right] \leq \alpha.
$$

Furthermore,

$$
\mathbb{E} \left[ \max_{j \in N} \frac{1 \{ K_j^{(b)} \leq \alpha \gamma \} }{\gamma} \right] \leq \mathbb{E} \left[ \sum_{j \in N \cap \tilde{S}^{(b)}} \frac{1 \{ \tilde{K}_j^{(b)} \leq \alpha \gamma \} }{\gamma} \right] \leq \sum_{j \in N \cap \tilde{S}^{(b)}} \frac{\alpha}{|\tilde{S}^{(b)}|} \leq \alpha.
$$

(5.A.1)

For a random variable $U$ taking values in $[0, 1]$,

$$
\sup_{\gamma \in (\gamma_{\min}, 1)} \frac{1 \{ U \leq \alpha \gamma \} }{\gamma} = \begin{cases} 
0 & U \geq \alpha, \\
\alpha / U & \alpha \gamma_{\min} \leq U < \alpha, \\
1 / \gamma_{\min} & U < \alpha \gamma_{\min}.
\end{cases}
$$

Moreover, if $U$ has a uniform distribution on $[0, 1],

$$
\mathbb{E} \left[ \sup_{\gamma \in (\gamma_{\min}, 1)} \frac{1 \{ U \leq \alpha \gamma \} }{\gamma} \right] = \int_0^{\alpha \gamma_{\min}} \gamma^{-1} dx + \int_{\alpha \gamma_{\min}}^{\alpha} x^{-1} dx = \alpha (1 - \log \gamma_{\min}).
$$

Hence, by using that $\tilde{K}_j^{(b)}$ has a uniform distribution on $[0, 1]$ for all $j \in N$, conditional on $S \subseteq \tilde{S}^{(b)},$

$$
\mathbb{E} \left[ \sup_{\gamma \in (\gamma_{\min}, 1)} \frac{1 \{ \tilde{K}_j^{(b)} \leq \alpha \gamma \} }{\gamma} \right] \leq \mathbb{E} \left[ \sup_{\gamma \in (\gamma_{\min}, 1)} \frac{1 \{ \tilde{K}_j^{(b)} \leq \alpha \gamma \} }{\gamma} \right] | S \subseteq \tilde{S}^{(b)} \right] = \alpha (1 - \log \gamma_{\min}).
$$

Analogously to (5.A.1),

$$
\mathbb{E} \left[ \max_{j \in N} \sup_{\gamma \in (\gamma_{\min}, 1)} \frac{1 \{ K_j^{(b)} \leq \alpha \gamma \} }{\gamma} \right] \leq \alpha (1 - \log \gamma_{\min}).
$$

Averaging over all bootstrap samples yields

$$
\mathbb{E} \left[ \max_{j \in N} \sup_{\gamma \in (\gamma_{\min}, 1)} \frac{1 \{ K_j^{(b)} \leq \alpha \gamma \} }{\gamma} \right] \leq \alpha (1 - \log \gamma_{\min}).
$$
Using that $x/y \geq 1\{x \geq y\}$ for all $x, y > 0$,
\[
\mathbb{E} \left[ \max_{j \in N} \sup_{\gamma \in (\gamma_{\min}, 1)} 1\{\pi_j(\alpha) \geq \gamma\} \right] \leq \alpha(1 - \log \gamma_{\min}),
\]
where we have used the same definition for $\pi_j(\cdot)$ as in the proof of Theorem 5.1.

Since the events $\{Q_j(\gamma) \leq \alpha\}$ and $\{\pi_j(\alpha) \geq \gamma\}$ are equivalent, it follows that
\[
\mathbb{P} \left[ \min_{j \in N} \inf_{\gamma \in (\gamma_{\min}, 1)} Q_j(\gamma) \leq \alpha \right] \leq \alpha(1 - \log \gamma_{\min}),
\]
implying that
\[
\mathbb{P} \left[ \min_{j \in N} \inf_{\gamma \in (\gamma_{\min}, 1)} Q_j(\gamma)(1 - \log \gamma_{\min}) \leq \alpha \right] \leq \alpha.
\]
Using the definition of $P_j$ in (5.2.3),
\[
\mathbb{P} \left[ \min_{j \in N} P_j \leq \alpha \right] \leq \alpha,
\]
which completes the proof. \(\Box\)

**Proof of Corollary 5.1.** Because the single-split method is model selection consistent, it must hold that $\mathbb{P}[\max_{j \in S} \hat{P}_j | \tilde{S}] \leq \alpha_n] - 1$ for $n \to \infty$. Using multiple data-splits, this property holds for each of the $B$ splits and hence $\mathbb{P}[\max_{j \in S} \max_b \hat{P}^{(b)}_j | \tilde{S}^{(b)}] \leq \alpha_n] - 1$, which implies that, with probability converging to 1 for $n \to \infty$, the quantile $\max_{j \in S} Q_j(1)$ is bounded from above by $\alpha_n$. The maximum over all $j \in S$ of the adjusted p-values $P_j = (1 - \log \gamma_{\min}) \inf_{\gamma \in (\gamma_{\min}, 1)} Q_j(\gamma)$ is thus bounded from above by $(1 - \log \gamma_{\min})\alpha_n$, again with probability converging to 1 for $n \to \infty$. \(\Box\)
Discussion of “One-step Sparse Estimates in Nonconcave Penalized Likelihood Models (H. Zou and R. Li)”

This chapter is an invited discussion of Zou and Li (2008). Our focus is mainly on high-dimensional problems where $p \gg n$: we will illustrate, empirically and by describing some theory, that many of the ideas in Zou and Li (2008) are very useful for the $p \gg n$ setting as well. Moreover, we propose a multi-step procedure which yields sparser solutions. In order to facilitate readability we often refer to Zou and Li (2008) as “the paper”.

105
6.1 Non-Convex Functions and Multi-Step Convex Optimization

The paper demonstrate a nice, and in a sense surprising, connection between difficult non-convex optimization and computationally efficient Lasso-type methodology which involves one- (or multi-) step convex optimization. The SCAD-penalty function (Fan and Li, 2001) has been often criticized from a computational point of view as it corresponds to a non-convex objective function which is difficult to minimize; mainly in situations with many covariates, optimizing SCAD-penalized likelihood becomes an awkward task.

The usual way to optimize a SCAD-penalized likelihood is to use a local quadratic approximation. Zou and Li (2008) show here what happens if one uses a local linear approximation instead. In 2001, when Fan and Li (2001) proposed the SCAD-penalty, it was probably easier to work with a quadratic approximation. Nowadays, and because of the contribution of Zou and Li (2008), a local linear approximation seems as easy to use thanks to the homotopy method (Osborne et al., 2000a) and the LARS algorithm (Efron et al., 2004). While the latter is suited for linear models, more sophisticated algorithms have been proposed for generalized linear models, cf. Genkin et al. (2007), Park and Hastie (2007) and Meier et al. (2008a).

In addition, and importantly, the local linear approximation yields sparse model fits where quite a few or even many of the coefficients in a linear or generalized linear model are zero, i.e. the method does variable selection. From this point of view, the local linear approximation is often to be preferred. In fact, it closely corresponds to the adaptive Lasso (Zou, 2006) which is in our view very useful for variable selection with Lasso-type technology. The rigorous convergence results in Section 2.3 of the paper, with a nice ascent property as for the EM-algorithm, are further reasons to favor the local linear approximation over the local quadratic versions with its heuristic rule for ad-hoc thresholding to zero (as described in the paragraph after formula (2.4) of the paper). Finally in Theorem 2 of the paper, the local linear approximation is shown to yield the best convex majorization of the penalty function.
6.1.1 Connection to Adaptive Lasso for Type 1 Penalty Functions

Section 4 in the paper distinguishes the cases where the regularization parameter can be separated from the penalty function or not. Type 1 penalty functions \( p_\lambda(t) = \lambda p(t) \) allow for separation, e.g. the Bridge penalties and the logarithm penalty, but excluding SCAD. From a computational point of view, the type 1 penalties are to be preferred because path-following algorithms can be used: this is Algorithm 1 in Section 4 of the paper, allowing to compute the whole regularization path very efficiently. In contrast, Algorithm 2, which can be used for the one-step SCAD estimator, seems much less efficient for approximating the entire regularization paths.

Question: Why should we use the one-step SCAD estimator? In particular, (at least some of the) one-step type 1 penalty estimators, e.g. the adaptive Lasso as discussed below, have the same asymptotic oracle properties, under the same conditions, as the one-step SCAD presented in Theorem 3.

Consider now type 1 penalty functions. Formulae (3.1) and (3.3) of the paper describe the one-step estimator based on the local linear approximation. In formula (3.1) corresponding to a linear model, we see that only the penalty function involves the initial estimator: the estimator can be written as

\[
\beta^{(1)} = \arg\min_\beta \frac{1}{2} \|y - X\beta\|^2 + n\lambda \sum_{j=1}^{p} w(|\beta_j^{(0)}|) |\beta_j|,
\]

where the penalty is based on re-weighting the \( \ell_1 \)-norm (or Lasso-penalty) with weights \( w_j = w(|\beta_j^{(0)}|) \) depending on the initial estimator. Of course, the weights also depend on the type 1 penalty function which we aim to approximate with the local linear approximation.

This is exactly the idea of the adaptive Lasso, recently proposed by Zou (2006). We think it is important to emphasize this connection (which is not mentioned at all in the paper) because: (i) it is a simple and very effective idea to reduce the bias of the Lasso, see below; (ii) the adaptive Lasso is theoretically supported (Zou, 2006) and enjoys the same oracle result as the one-step SCAD described in Theorem 3,
and there are theoretical results even for the high-dimensional situation where \( p \gg n \) (Huang et al., 2008).

Regarding issue (i): particularly in cases with many ineffective (or non-substantial) covariates, the prediction optimal (w.r.t. MSE) Lasso typically needs a large penalty parameter to get rid of these ineffective covariates. But a large penalty parameter implies substantial shrinkage to zero even for the coefficients corresponding to the important covariates. Solutions to improve such bias problems of the Lasso are based on two-stage procedures, e.g. the LARS-OLS hybrid (Efron et al., 2004), the adaptive Lasso (Zou, 2006) or the relaxed Lasso (Meinshausen, 2007).

### 6.2 Variable Selection in the High-Dimensional Case

For simplicity, consider a linear model

\[ y = X\beta + \varepsilon \quad (6.2.1) \]

as discussed in Section 3.1 of the paper. Our goal is variable selection (which is in many applications more relevant than prediction; we will present an example from biology in our Section 6.3.2). In high-dimensional problems where \( p \gg n \), computational aspects become crucial. Since there are \( 2^p \) sub-models, we cannot inspect all of them (even when using efficient branch-and-bound methods). Ad-hoc methods can be used but they may be very unstable yielding poor results (e.g. forward variable selection), see Breiman (1996). On the other hand, having provably correct algorithms or methods, as the one in the paper with provable properties, is much more desirable.

The Lasso and its modifications belong to the latter class of methods. Regarding the computational feasibility, for linear models, the complexity to compute all sub-models from Lasso is \( O(np \min(n,p)) \) which is linear in the dimensionality \( p \) if \( p \gg n \). An important question is whether such computationally efficient estimators have good, provable statistical properties. Meinshausen and Bühlmann (2006) showed consistency of the Lasso for variable selection in the high-dimensional setting where \( p \gg n \): there is one major assumption, the neighborhood
stability condition, which was shown to be sufficient and “almost” necessary (the wording “almost” refers to a numerical value which has to be $< 1$ for sufficiency and $\leq 1$ for necessity). Later, the irrepresentable condition has been worked out (Zou, 2006; Zhao and Yu, 2006) which is easier to interpret but is equivalent to the Meinshausen-Bühlmann assumption on neighborhood stability. The irrepresentable assumption is restrictive and easily fails to hold if the design matrix exhibits a too strong “degree of linear dependence” (of course, there is always linear dependence if $p \gg n$) or a too strong population absolute correlation among the covariates.

Since the irrepresentable (or neighborhood stability) condition is restrictive, one would like to understand Lasso’s behavior under weaker assumptions: recently, consistency results in terms of

$$
\| \hat{\beta}(\lambda) - \beta \|_q = o_P(1) \quad (n \to \infty), \quad q \in \{1, 2\}
$$

have been achieved, see van de Geer (2008), Zhang and Huang (2008) and Meinshausen and Yu (2009). The result in (6.2.2) has implications for variable selection. In case of fixed dimension $p$, (6.2.2) implies: if $\beta_j \neq 0$, then $\hat{\beta}_j(\lambda) \neq 0$ with probability tending to 1 (because otherwise, the convergence to zero in (6.2.2) would not hold). That is:

The Lasso yields a too large model which contains the true model with high probability (tending to 1 as $n \to \infty$).  

Under suitable conditions, the statement in (6.2.3) also holds in the high-dimensional case where $p \gg n$, see Meinshausen and Yu (2009). In addition, we point out that:

The prediction optimal tuned Lasso contains the true model with high probability (tending to 1 as $n \to \infty$).

This has been proved for simple cases in Meinshausen and Bühlmann (2006).

Putting the two facts (6.2.3) and (6.2.4) together, we can view the Lasso as an excellent and computationally efficient tool for “variable filtering”, in the sense that the true model is with high probability a subset of the Lasso-estimated model. To appreciate the value of such a result: imagine that we have $p \approx 10,000$ and $n \approx 50$ (e.g. from microarray data). As the size of the Lasso-estimated sub-model is bounded by
min(n, p) which equals n if $p \gg n$, we pursue an immense dimensionality reduction from $p \approx 10'000$ to something of the order 50.

When viewing the Lasso as a variable filtering method, it is clear that we want to do an additional step (similar to the one-step estimator in the paper) which aims to go from the Lasso-estimated model in the first stage to the true model in a second stage. We have already touched upon two-stage procedures for addressing the bias problem in Lasso. The main proposals are the LARS-OLS hybrid (Efron et al., 2004), the relaxed Lasso (Meinshausen, 2007) and the adaptive Lasso (Zou, 2006) with the Lasso as initial estimator: all of them reduce the bias and in fact, this is what will lead to consistency in variable selection. We think (based on empirical evidence) that the latter, which is essentially the one-step estimator in the paper when using the Lasso as initial estimator, is a very elegant way to address Lasso’s overestimation behavior. In addition, some theory for consistency in variable selection has been worked out for the high-dimensional case (Huang et al., 2008).

### 6.3 Beyond the One-Step Estimator

For regularization in high-dimensional spaces, we may want to use more than one or two regularization parameters. This can be achieved by pursuing more iterations where every iteration involves a separate tuning parameter (and as described below, those parameters are “algorithmically” constrained). We propose here the

Multi-Step Adaptive Lasso (MSA-LASSO)

1. Initialize the weights $w^{(0)}_j \equiv 1$ ($j = 1 \ldots, p$).

2. For $k = 1, 2, \ldots, M$:
   - Use the adaptive Lasso with penalty function
     $$
     \lambda^{(k)}_* \sum_{j=1}^{p} w^{(k-1)}_j |\beta_j|.
     $$

   where $\lambda^{(k)}_*$ is the regularization parameter leading to prediction optimality. Denote the estimator by $\beta^{(k)} = \beta^{(k)}(\lambda^{(k)}_*)$. In practice,
the value \( \lambda^{(k)}_* \) can be chosen via some cross-validation scheme.

Up-date the weights

\[
    w^{(k)}_j = \frac{1}{|\beta^{(k-1)}(\lambda^{(k-1)}_*)_j|}, \quad j = 1, \ldots, p.
\]

For \( k = 1 \), we do an ordinary Lasso fit and \( k = 2 \) corresponds to the adaptive Lasso. Note that what is termed “one-step” in the paper corresponds here to \( k = 2 \). Note that Zou and Li initialize with \( w_j^{(0)} \equiv 0 \) (in the terminology of MSA-LASSO), yielding the MLE (in step \( k = 1 \)).

We find it more natural, and actually essential in the high-dimensional case with \( p \gg n \), to initialize with the non-zero weights allowing for regularized fitting in step \( k = 1 \).

We will illustrate below the MSA-LASSO on a small simulated model and a real data set from molecular biology. Before, we describe some properties of the method which are straightforward to derive.

**Property 1:** MSA-LASSO increases the sparsity in every step in terms of the \( \ell_0 \)-norm, i.e. fewer selected variables in every step. As “heuristics”, which is derived from the Zou and Li paper, MSA-LASSO is related to approximating the non-convex optimization problem with the log-penalty \( \sum_{j=1}^p \log(|\beta_j|) \), see the formula appearing just before Proposition 2.

**Property 2:** MSA-LASSO can be computed using the LARS algorithm for every step. The computational complexity of MSA-LASSO is bounded by \( O(Mnp \min(n,p)) \); due to the increase of sparsity, a later step is faster to compute than an early one. The computational load is in sharp contrast to computing all solutions over all \( M \) steps when allowing for any \( \lambda \) for each Lasso path: this would require many more essential operations.

MSA-LASSO is different from the multi-step procedure as analyzed in Section 2.3 of the paper: there, the regularization parameter \( \lambda \) is fixed. We will also discuss below in Tables 6.1 and 6.2, for a simulated example, that the algorithmic restriction of choosing the regularization parameters in a sequentially optimal fashion seems very reasonable.
6.3.1 Small Simulation Study

To illustrate the proposed MSA-LASSO method, a small simulation study is carried out. We use a linear model as in (6.2.1) with covariates $X$ from a multivariate normal distribution with correlation matrix $\Sigma_{i,j} = \rho^{|i-j|}$ (for various values of $\rho$). The true underlying parameter vector is of the form $\beta = (c, \ldots, c, 0, \ldots, 0)^T$ with $p_{\text{act}}$ non-zero entries and $c$ such that the signal-to-noise ratio is 9 (which we find more relevant for practical applications than a signal-to-noise ratio of 21.25 as in Example 1 in the paper). The number of predictors is set to $p = 500$. We choose the number of active variables $p_{\text{act}} \in \{3, 25\}$. In each simulation run, a training set of size 100 and a validation set of size 50 is used to determine the prediction optimal estimator. A total of 100 simulation runs are carried out for each parameter setting.

As performance measures we use the squared error $\|\hat{\beta} - \beta\|_2^2$ and the number of false positives (FP) $\sum_{j=1}^p I(\hat{\beta}_j \neq 0, \beta_j = 0)$. Table 6.1 illustrates the results for the case $p_{\text{act}} = 3$. We denote by 1-Step ($k = 2$) the MSA-LASSO with $k = 2$: it equals the adaptive Lasso with Lasso as initial estimator and we use the terminology “1-Step” to be more consistent with the paper. Furthermore, 1-Step opt. is the adaptive Lasso with Lasso as initial estimator (as for 1-Step) but we optimize over a large 2-dimensional grid of the two regularization parameters which are involved in the initial Lasso and the adaptive Lasso step. Several conclusions can be made. The estimation error can be slightly

<table>
<thead>
<tr>
<th></th>
<th>$\rho = 0$</th>
<th>$\rho = 0.2$</th>
<th>$\rho = 0.5$</th>
<th>$\rho = 0.8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-Step ($k = 2$)</td>
<td>0.12 (0.16)</td>
<td>0.09 (0.09)</td>
<td>0.11 (0.11)</td>
<td>0.21 (0.23)</td>
</tr>
<tr>
<td>2-Step ($k = 3$)</td>
<td>0.08 (0.12)</td>
<td>0.08 (0.10)</td>
<td>0.09 (0.09)</td>
<td>0.23 (0.30)</td>
</tr>
<tr>
<td>1-Step opt.</td>
<td>0.08 (0.09)</td>
<td>0.09 (0.10)</td>
<td>0.10 (0.09)</td>
<td>0.19 (0.17)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$\rho = 0$</th>
<th>$\rho = 0.2$</th>
<th>$\rho = 0.5$</th>
<th>$\rho = 0.8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-Step ($k = 2$)</td>
<td>6.10 (12.07)</td>
<td>3.82 (6.59)</td>
<td>3.73 (5.00)</td>
<td>3.19 (6.23)</td>
</tr>
<tr>
<td>2-Step ($k = 3$)</td>
<td>2.93 (7.53)</td>
<td>2.19 (6.24)</td>
<td>1.65 (2.69)</td>
<td>2.19 (5.09)</td>
</tr>
<tr>
<td>1-Step opt.</td>
<td>2.92 (7.13)</td>
<td>3.08 (7.55)</td>
<td>2.88 (6.00)</td>
<td>3.36 (5.56)</td>
</tr>
</tbody>
</table>

Table 6.1: Results for $p_{\text{act}} = 3$ active variables. Average and standard deviations (in parentheses) of squared errors and of false positives (FP).
decreased by an additional step if the correlation is not too high. More importantly, the number of false positives gets reduced. The number of false negatives is zero for this setting (not shown), i.e. the true variables are always identified. The computational extra effort of the 1-Step opt. estimator does not pay off in this situation.

The results for \( p_{\text{act}} = 25 \) are given in Table 6.2. There is a slight loss

<table>
<thead>
<tr>
<th></th>
<th>( \rho = 0 )</th>
<th>( \rho = 0.2 )</th>
<th>( \rho = 0.5 )</th>
<th>( \rho = 0.8 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-Step (( k = 2 ))</td>
<td>6.15 (1.48)</td>
<td>3.23 (1.96)</td>
<td>1.78 (0.45)</td>
<td>1.37 (0.32)</td>
</tr>
<tr>
<td>2-Step (( k = 3 ))</td>
<td>6.25 (1.50)</td>
<td>3.28 (1.12)</td>
<td>1.92 (0.48)</td>
<td>1.73 (0.35)</td>
</tr>
<tr>
<td>1-Step opt.</td>
<td>6.03 (1.49)</td>
<td>2.95 (1.04)</td>
<td>1.48 (0.49)</td>
<td>1.25 (0.33)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>( \rho = 0 )</th>
<th>( \rho = 0.2 )</th>
<th>( \rho = 0.5 )</th>
<th>( \rho = 0.8 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-Step (( k = 2 ))</td>
<td>32.91 (23.66)</td>
<td>30.26 (18.16)</td>
<td>17.91 (16.37)</td>
<td>8.42 (9.01)</td>
</tr>
<tr>
<td>2-Step (( k = 3 ))</td>
<td>27.43 (23.54)</td>
<td>25.52 (17.44)</td>
<td>14.60 (15.76)</td>
<td>6.35 (6.73)</td>
</tr>
<tr>
<td>1-Step opt.</td>
<td>31.55 (22.11)</td>
<td>24.55 (15.55)</td>
<td>9.48 (11.37)</td>
<td>2.76 (5.61)</td>
</tr>
</tbody>
</table>

**Table 6.2:** Results for 25 active variables. Average and standard deviations (in parentheses) of squared errors and of false positives (FP).

in terms of mean squared error (MSE) for this setting when doing an additional step. Already the 1-Step estimator loses compared to the initial Lasso estimator (not shown), in terms of MSE; likewise, the 1-Step opt. estimator has worse performance than the initial Lasso estimator. However, the number of false positives (FP) gets reduced again due to the increased sparsity. In terms of FP, the 1-Step opt. estimator seems to perform better for moderate and large values of \( \rho \); but an additional step (\( k = 4 \)) in MSA-LASSO would improve performance with respect to FP as well.

### 6.3.2 Real Data Example from Biology

Reducing the number of false positives can be very desirable in biological applications since follow-up experiments can be costly and laborious. In our experience, it is often appropriate to do estimation on the conservative side with a low number of false positives because we still see more positives than what can be typically validated in a laboratory.

We illustrate the MSA-LASSO method on a problem of motif regression (Conlon et al., 2003) for finding transcription factor binding
sites in DNA sequences. Beer and Tavazoie (2004) contains a collection of microarray data and a collection of motif candidates for yeast. The idea is to predict the gene expression value of a gene based on the corresponding motif scores (the information based on the sequence data). The dataset which we consider consists of \( n = 2587 \) gene expression values of a heat-shock experiment and \( p = 666 \) motif scores. We use a training set of size 1300 and a validation set of size 650. The remaining data is used as a test-set.

The squared prediction error on the test-set \( \mathbb{E}[(\hat{y}_{new} - y_{new})^2] = (\hat{\beta} - \beta)^T \Sigma (\hat{\beta} - \beta) + \text{Var}(\varepsilon) \) remains essentially constant for all estimators (probably due to high noise, i.e. large value of \( \text{Var}(\varepsilon) \)): 0.6193, 0.6230, 0.6226 for the Lasso, 1-Step and 2-step estimator, respectively. But the number of selected variables decreases substantially:

<table>
<thead>
<tr>
<th></th>
<th>Lasso</th>
<th>1-Step</th>
<th>2-Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of selected variables</td>
<td>91</td>
<td>42</td>
<td>28</td>
</tr>
</tbody>
</table>

The list of top-candidate motifs gets slightly rearranged between the different estimators. The hope (and in part a verified fact) is that the 1- or 2-Step estimator yield more stable lists with fewer false positives.
Bibliography


Curriculum Vitae

I was born on August 9, 1979 in Brugg. After attending primary school from 1986 to 1991 and Bezirksschule from 1991 to 1995 in Brugg, I spent 4 years at the Kantonsschule Baden and obtained a “Matura, Typus C” in 1999.

I studied mathematics at ETH Zurich from October 1999 to March 2004, finishing with my diploma thesis on “Extremwertanalyse von Starkniederschlägen” under the guidance of Prof. H.R. Künsch.

In July 2004, I joined the statistical consulting team at the Seminar für Statistik at ETH Zurich. In May 2005, I enrolled as a Ph.D. student under the guidance of Prof. Peter Bühlmann and worked as a teaching assistant at the Seminar für Statistik.