Universidad Politécnica de Madrid
Facultad de Informática

Regularization for sparsity in statistical analysis and machine learning

Tesis Doctoral

Diego Vidaurre
Ingeniero en Informática

2012

Departamento de Inteligencia Artificial
Facultad de Informática

Regularization for sparsity in statistical analysis and machine learning

Diego Vidaurre
Ingeniero en Informática

Directores:

Concha Bielza
Doctora en Informática

Pedro Larrañaga
Doctor en Informática

Tribunal

Presidente: Serafín Moral

Vocales: Antonio Salmerón
Robert Castelo
Vincenç Gómez i Cerdá

Secretario: Rubén Armañanzas

Suplentes: Juan Antonio Fernández del Pozo
Iñaki Inza

Agradecimientos

Tras varios años de trabajo invertidos en el desarrollo de este trabajo, han sido varias las personas que merecen un agradecimiento:

Concha Bielza y Pedro Larrañaga, mis directores de tesis, por su apoyo y su consejo.
Mis padres, por generosas dosis de paciencia y cariño.
Maya, por estar ahí siempre.
Mis compañeros del laboratorio, por una atmósfera de trabajo valiosa. Mis amigos, por alguna otra razón.

Quiero mencionar a Tom Heskes, Marcel van Gerven y la gente del Machine Learning Group de Universidad de Radboud por su hospitalidad en los tres meses que pasé en Holanda trabajando y aprendiendo de ellos.

También, quiero agradecer al proyecto Cajal Blue Brain, y a los proyectos TIN2010-20900-C04-04 y Consolider Ingenio 2010-CSD2007-00018, su soporte económico, sin el cual el trabajo no hubiera sido posible.

Finalmente, quiero dedicar este trabajo a Jorge Muruzábal, la persona que me guió en mi primer contacto con el mundo de la investigación y al que debo mucho más que eso. Nunca dejaré de tener presente a Jorge.

Acknowledgements

After some years i have been working in this thesis, there are a number of people that deserve a grateful acknowledge.

Concha Bielza and Pedro Larrañaga, my supervisors, for their support and advices.
My parents, for generous amounts of patience and love.
Maya, for being there, always.
My workmates, for a friendly work atmosphere.
My friends, for some other reasons.

I want to mention Tom Heskes, Marcel van Gerven and the people from the Machine Learning Group, in the Radboud University, for their hospitality during the three months i spent in the Netherlands, working with and learning from them.

I am grateful to Cajal Blue Brain project and TIN2010-20900-C04-04 and Consolider Ingenio 2010-CSD2007-00018 projects, because without their finantial support this work could not been done.

Finally, i want to dedicate this thesis to Jorge Muruzábal, the person who guided my first contact with research and whom i owe much more than this. Jorge will always be present to me.

CONTENTS

SUMMARY

Pragmatism is the leading motivation of regularization. We can understand regularization as a modification of the maximum-likelihood estimator so that a reasonable answer could be given in an unstable or ill-posed situation. To mention some typical examples, this happens when fitting parametric or non-parametric models with more parameters than data or when estimating large covariance matrices. Regularization is usually used, in addition, to improve the bias-variance tradeoff of an estimation. Then, the definition of regularization is quite general, and, although the introduction of a penalty is probably the most popular type, it is just one out of multiple forms of regularization.

In this dissertation, we focus on the applications of regularization for obtaining sparse or parsimonious representations, where only a subset of the inputs is used. A particular form of regularization, L_1-regularization, plays a key role for reaching sparsity. Most of the contributions presented here revolve around L_1-regularization, although other forms of regularization are explored (also pursuing sparsity in some sense). In addition to present a compact review of L_1-regularization and its applications in statistical and machine learning, we devise methodology for regression, supervised classification and structure induction of graphical models. Within the regression paradigm, we focus on kernel smoothing learning, proposing techniques for kernel design that are suitable for high dimensional settings and sparse regression functions. We also present an application of regularized regression techniques for modeling the response of biological neurons. Supervised classification advances deal, on the one hand, with the application of regularization for obtaining a naïve Bayes classifier and, on the other hand, with a novel algorithm for brain-computer interface design that uses group regularization in an efficient manner. Finally, we present a heuristic for inducing structures of Gaussian Bayesian networks using L_1-regularization as a filter.

RESUMEN

El pragmatismo es la principal motivación de la regularización. Podemos entender la regularización como una modificación del estimador de máxima verosimilitud, de tal manera que se pueda dar una respuesta cuando la configuración del problema es inestable. A modo de ejemplo, podemos mencionar el ajuste de modelos paramétricos o no paramétricos cuando hay más parámetros que casos en el conjunto de datos, o la estimación de grandes matrices de covarianzas. Se suele recurrir a la regularización, además, para mejorar el compromiso sesgo-varianza en una estimación. Por tanto, la definición de regularización es muy general y, aunque la introducción de una función de penalización es probablemente el método más popular, éste es sólo uno de entre varias posibilidades.

En esta tesis se ha trabajado en aplicaciones de regularización para obtener representaciones dispersas, donde sólo se usa un subconjunto de las entradas. En particular, la regularización L_1 juega un papel clave en la búsqueda de dicha dispersión. La mayor parte de las contribuciones presentadas en la tesis giran alrededor de la regularización L_1, aunque también se exploran otras formas de regularización (que igualmente persiguen un modelo disperso). Además de presentar una revisión de la regularización L_1 y sus aplicaciones en estadística y aprendizaje de máquina, se ha desarrollado metodología para regresión, clasificación supervisada y aprendizaje de estructura en modelos gráficos. Dentro de la regresión, se ha trabajado principalmente en métodos de regresión local, proponiendo técnicas de diseño del kernel que sean adecuadas a configuraciones de alta dimensionalidad y funciones de regresión dispersas. También se presenta una aplicación de las técnicas de regresión regularizada para modelar la respuesta de neuronas reales. Los avances en clasificación supervisada tratan, por una parte, con el uso de regularización para obtener un clasificador naive Bayes y, por otra parte, con el desarrollo de un algoritmo que usa regularización por grupos de una manera eficiente y que se ha aplicado al diseño de interfaces cerebro-máquina. Finalmente, se presenta una heurística para inducir la estructura de redes Bayesianas Gaussianas usando regularización L_1 a modo de filtro.

L_1-regularization, or regularization by an L_1-penalty, has become very popular in the last two decades and is of special interest for achieving sparse solutions. We basically add an L_1-penalty, defined as the sum of absolute values of the model parameters, to the original target function that we want to optimize. In general terms, a sparse solution is a solution with reduced complexity. However, it is usual to define sparsity as the selection of important variables, discarding the rest from the model, so that L_1-regularization is linked to the feature selection problem. This is the definition that we use in this dissertation.

Although the L_1-penalty idea dates back further, the major impact of this type of L_1-regularization materialized after Tibshirani (1996) proposed the lasso. The lasso is just the application of the L_1-penalty to linear regression, where the response to predict is continuous. The emergence of the LARS algorithm (Efron et al., 2004), which efficiently solves the optimization problem underlying the lasso, was another key to the rapid spread of the lasso within the statistics and machine learning communities.

Some researchers view variable selection in regression as one of the most important problems in modern statistics. Traditionally, model selection was built on methods like forward stepwise regression, all subsets regression or prefiltering approaches. Some of these approaches are based on univariate measures and are thus seriously biased. Others, like all subset regression (if the cardinality of the set of features is not high), are not computationally affordable for data sets of moderate or big size in the number of variables. The main advantage of the lasso and related methods is that they offer interpretable, stable models and an accurate prediction (not exempt, however, from some bias) at a reasonable cost. For example, the lasso/LARS approach has the computational cost of a least-squares estimation. Also, it provides a data-driven and very simple estimation of the optimal level of model complexity, i.e., of how much the model should be regularized.

Although we pay special attention to the lasso, and most of the contributions discussed in this thesis are about the L_1-regularization idea (Vidaurre et al., 2010, 2011a,c,d,e,f), we also deal with other types of regularization. In particular, (Vidaurre et al., 2012) is related to L_2-regularization (ridge regression within the statistical literature), being the L_2-penalty the sum of the squared parameters. L_2-regularization was formally introduced by Hoerl and Kennard (1970) almost 30 years later than the birth of the original idea, due to Tikhonov (1943), who approximated the solution of a set of unsolvable integral equations using this concept. Another paper (Vidaurre et al., 2011b) applies penalty-free regularization for obtaining a naïve Bayes classifier. Vidaurre et al. (2011g) also make certain use of regularization different from L_1-regularization within a boosting trees approach.

The main contributions of this dissertation are presented within four blocks:

- **A survey on L_1-regularization for statistics and machine learning**, where we present a review of the concept and application of L_1-regularization. It is not our aim to present a comprehensive list of the utilities of the L_1-penalty. Rather we focus on what, we believe, is the set of most representative uses of this regularization technique, which we describe in some detail. Thus, we deal with a number of L_1-regularized methods for linear and nonlinear regression, supervised classification, unsupervised classification, induction of graphical models and feature extraction. Although this review targets practice rather than theory,

we do give some theoretical details about L_1-penalized linear regression. This section is based on (Vidaurre et al., 2011f).

- **Regularization in nonparametric regression**, where we discuss some contributions for nonparametric continuous response estimation. First, we propose a heuristic that combines the lasso with locally weighted regression to achieve sparse models. The core of this idea is that the kernel estimation in local regression should account for the importance of each variable. In other words, if a variable is absolutely irrelevant for the regression function, or noisy, it should be precluded from the kernel design. We call this method *lazy lasso*. This idea appeared in (Vidaurre et al., 2011d).

  Second, also within the local linear regression framework, we intend to give a Bayesian formulation of regularized local linear regression, where a nearly optimal smoothing parameter or bandwidth is estimated. We show how to give a maximum posterior joint estimation of both the kernel and the regression function. Also, we develop a full Bayesian treatment, based on the data augmentation algorithm, for finding both the parameter distribution and the predictive distribution. These contributions are compiled in (Vidaurre et al., 2012).

  Finally, we apply ideas from nonparametric regression for estimating the instantaneous firing rate of biological neurons, showing how different levels of nonlinearity influence the instantaneous firing rate estimation. Nonlinearity can be achieved in several ways. Besides local estimation, we can enrich the predictor set with basis expansions of the input variables, enlarging the number of inputs. Spline-based models are popular within this category. Our goal is to demonstrate that appropriate nonlinearity treatment can greatly improve the results. We test our hypothesis on both synthetic data and real neuronal recordings in cat primary visual cortex, giving a plausible explanation of the results from a biological perspective (Vidaurre et al., 2011e).

- **Regularization for supervised classification**, where we present three contributions of the application of regularization within the supervised classification paradigm. First, we focus on the naïve Bayes model and propose the application of regularization techniques to learn a naïve Bayes classifier. The main contribution is a stagewise version of the selective naïve Bayes, which can be considered a regularized version of the naïve Bayes model. We call it *forward stagewise naïve Bayes*. For comparison's sake, we also introduce an explicitly regularized formulation of the naïve Bayes model, where conditional independence (absence of arcs) is promoted via an L_1/L_2-group penalty on the parameters that define the conditional probability distributions. Although already published in the literature, this idea has only been applied for continuous predictors. We extend this formulation to discrete predictors and propose a modification that yields an adaptive penalization. We show that, whereas the L_1/L_2 group penalty formulation only discards irrelevant predictors, the forward stagewise naïve Bayes can discard both irrelevant and redundant predictors, which are known to be harmful for the naïve Bayes classifier. This work is described in detail in (Vidaurre et al., 2011b).

  Second, we study how to apply a regularization technique to learn a computationally efficient classifier that is inspired by naïve Bayes. The proposed formulation, combined with an L_1-penalty, is capable of discarding harmful, redun-

dant predictors. A modification of the LARS algorithm is devised to solve this problem. We tackle both real-valued and discrete predictors, assuring that our method is applicable to a wide range of data. Vidaurre et al. (2011c) details this approach.

Eventually, we present a signal classification framework that can be used for brain-computer interface design. The actual classification is performed on sparse autoregressive features. It can use any well-known classification algorithm, such as discriminant analysis, linear logistic regression and support vector machines. The autoregressive coefficients of all signals and channels are simultaneously estimated by the group lasso, and the estimation is guided by the classification performance. Thanks to the variable selection capability of the group lasso, the framework can drop both individual autoregressive coefficients and entire channels that are useless in the prediction stage. Also, the framework is relatively insensitive to the chosen autoregressive order. We devise an efficient algorithm to solve this problem. This contribution is presented in (Vidaurre et al., 2011a)

- **A regularized estimation of Gaussian Bayesian networks**. Learning the structure of a graphical model from data is a common task in a wide range of practical applications. Here, we focus on Gaussian Bayesian networks, that is, on continuous data and directed acyclic graphs with a joint probability density of all variables given by a Gaussian density. We propose to work in an equivalence class search space, specifically using the k-greedy equivalent search algorithm. This, combined with regularization techniques to guide the structure search, can learn sparse networks close to the one that generated the data. This work is presented in (Vidaurre et al., 2010).

The dissertation is hence divided into five main parts and nine chapters. The first part, which includes the first chapter, is devoted to introducing the basic regularization concepts. First, we describe linear regression and the lasso. Then, we extend the discussion beyond linearity towards methods based on basis expansions and local nonparametric approaches. After, we deal with supervised classification, focusing on discriminant analysis, logistic regression and support vector machines. Also, we describe methods for unsupervised classification. Following, we give a brief summary of L_1-regularized methods for structure induction of both undirected and directed graphical models. Finally, we discuss how L_1-regularization can be applied for feature extraction, including principal component analysis and independent component analysis.

The second part discusses the contributions on nonparametric regression. Chapter 2 describes the lazy lasso approach. Chapter 3 presents the sparse Bayesian regularized local regression, where the estimation of the bandwidth is based on optimal methodologies. Chapter 4 discusses nonlinearity in neural encoding models applied to the primary visual cortex.

The third part is about supervised classification. Chapter 5 introduces the forward stagewise naïve Bayes. Chapter 6 details an L_1-regularized naïve Bayes-inspired classifier for discarding redundant predictors. Chapter 7 deals with classification of neural signals from sparse autoregressive features.

The fourth part, embracing chapter 8, discusses how to learn an L_1-regularized Gaussian Bayesian network in the equivalence class space.

Specific conclusions and future work ideas are exposed in each corresponding chapter. The fifth and last part gives an overview of such conclusions.

Part I

INTRODUCTION TO L_1-REGULARIZATION

1

INTRODUCTION TO L_1-REGULARIZATION

Generally speaking, regularization is a type of technique that, by introducing some constraint on the parameters, alters a maximum likelihood estimation that is unstable or cannot be obtained by a classical estimator. Consequently, the regularized solutions are conveniently less complex and, hence, more stable. In other words, regularization imposes certain restrictions on the optimization problem so as to trade a little bias in exchange for a larger reduction in variance, and hence avoids overfitting. Bickel and Li (2006) present an excellent general review of regularization in statistics.

In this part, we focus on some relevant practical methods that make use of L_1-regularization and describe them in some detail. Special attention is paid to the lasso. The chapter is organized as follows. Section 1.1 describes linear regression and includes the lasso. Section 1.2 extends Section 1.1 beyond linearity, discussing methods based on basis expansions and local nonparametric approaches. Section 1.3 deals with supervised classification and is centered on discriminant analysis, logistic regression and support vector machines. Section 1.4 describes methods for unsupervised classification. Section 1.5 gives a brief summary of L_1-regularized methods for structure induction of both undirected and directed graphical models. Section 1.6 discusses how L_1-regularization can be applied for feature extraction, including principal component analysis and independent component analysis. Section 1.7 describes some approaches using L_1-regularization for time series analysis. Table 1.1 lists the methods that are described throughout this chapter.

Some applications of L_1-regularization that, albeit interesting, are omitted from this review include quantile regression (Li and Zhu, 2008), Cox models for survival analysis (Porzelius et al., 2010), matrix completion (Candès and Tao, 2010), compressed sensing (Donoho, 2006), sparse canonical correlation analysis (Hardoon and Shawe-Taylor, 2011) or sparse coding (Sprechmann et al., 2010; Lee et al., 2007), among others.

This survey is based on the submitted article (Vidaurre et al., 2011f).

1.1 LINEAR REGRESSION

We first deal with the estimation of a continuous dependent variable from a set of independent (typically, but not necessarily, continuous) variables. Although this dissertation focuses on the practical uses of the L_1-penalty, we go into some theoretical detail about the lasso because of its unquestionable importance. Afterwards, we describe a number of extensions of the lasso.

The general linear regression problem is defined by following notation. We denote the set of p input variables as $\{X_1, ..., X_p\}$ and the scalar response variable as Y. Let $\boldsymbol{D} = \{(x_{i1}, ..., x_{ip}, y_i), i = 1, ..., N\}$ be the labeled data set containing N instances. In the simplest version of the problem, we assume $\boldsymbol{x}_i \in \mathbb{R}^p$. We denote the $N \times p$ predictor data matrix as $\boldsymbol{X}$, i.e.

Table 1.1: Outline of the review

Section	Subsection	Described methods
Linear regression	Lasso	
	Improving lasso's properties	Relaxed lasso, VISA, adaptive lasso, Dantzig selector, LAD-lasso, elastic net, bootstrapping lasso
	Adapting to particular problems	Group lasso, CAP, fused lasso, multiresponse, generalizations
Nonlinear regression	Basis expansions	L_1-regularized additive cubic smoothing splines, SpAM, adaptive functional group lasso, lasso-type spline method, COSSO, VANISH
	Kernel smoothing regression	Lazy lasso, rodeo, geographically weighted lasso, smoothed lasso, L_1-regularized varying coefficient model
Supervised classification	Logistic regression	L_1-regularized logistic regression, logistic group lasso, L_1-regularized multinomial logistic regression, L_1-regularized nonlinear logistic regression
	LDA	Sparse LDA, DALASS, nearest shrunken centroids
	SVM	L_1-regularized SVM, L_1-regularized multiclass SVM, L_1-regularized multiclass SVM on ANOVA kernels
Unsupervised classification	Model-based Other methods	L_1-penalized model-based clustering, CAP penalties L_1-regularized K-means, L_1-regularized hierarchical clustering, L_1-regularized subspace clustering
Graphical models	Continuous Markov networks	Neighborhood selection method, graphical lasso, CLIME, L_1-regularized Cholesky decomposition
	Discrete Markov networks	L_1-regularized log-likelihood maximization, neighborhood selection by L_1-regularized logistic regression, L_1-regularized pseudo-likelihood maximization
	Gaussian Bayesian networks	L_1-regularized DAG estimation, hybrid approaches
	Discrete Bayesian networks	Hybrid approaches
Feature extraction	PCA	Sparse PCA, robust sparse PCA, SCoTLASS
	ICA	Sparse ICA
Time series	Wavelets	SURE shrinkage, regularized one-step estimator, basis pursuit
	Autoregressive models	L_1-regularized MAR models, group lasso for MAR models, L_1-regularized REGAR models
	Other regression models	Fused lasso, smoothed lasso, LAPS
	Change point analysis	L_1-regularized change point analysis

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x}_1' \\ \vdots \\ \boldsymbol{x}_i' \\ \vdots \\ \boldsymbol{x}_N' \end{pmatrix} = \begin{pmatrix} x_{11} & \dots & x_{1j} & \dots & x_{1p} \\ \vdots & & \vdots & & \vdots \\ x_{i1} & \dots & x_{ij} & \dots & x_{ip} \\ \vdots & & \vdots & & \vdots \\ x_{N1} & \dots & x_{Nj} & \dots & x_{Np} \end{pmatrix}.$$

The objective is to linearly predict the response for any new instance.

1.1.1 *The lasso*

Assuming centered data, the lasso (Tibshirani, 1996) estimate minimizes the residual sum of squares subject to an L_1-constraint:

$$\hat{\boldsymbol{\beta}} = \text{argmin}_{\boldsymbol{\beta}} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 \qquad \text{s.t.} \quad ||\boldsymbol{\beta}||_1 \leq s, \tag{1.1}$$

where $\boldsymbol{\beta} \in \mathbb{R}^p$, $s \geq 0$ and $||.||_q$ is the q-norm. Equivalently, the lasso can be defined in the Lagrangian form:

$$\hat{\boldsymbol{\beta}} = \text{argmin}_{\boldsymbol{\beta}} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_1, \tag{1.2}$$

where the regularization parameter $\lambda \geq 0$ has a one-to-one correspondence with the parameter s of Equation (1.1). Thus, the lasso estimator substitutes the L_2-penalty of the ridge estimator (Hoerl and Kennard, 1970), defined as the sum of the squared parameters, for an L_1-penalty.

Typically, the data is standardized so that the penalty is invariant with regard to the scale of the variables. If the data is not centered, a non-penalized intercept should be included within vector $\boldsymbol{\beta}$.

The lasso performs variable selection by driving a number of regression coefficients to exactly zero. Therefore, by increasing the value of λ, we can control the number of variables included in the model. A fairly small value of λ leads to the least squares solution. As we increase the value of λ, one coefficient at a time is made different from zero, although some variables can sporadically exit the model in the presence of correlated inputs. In other words, the λ parameter controls the degrees of freedom of the estimation. Zou et al. (2007) show that the number of nonzero coefficients is an unbiased estimation of the degrees of freedom of the lasso estimator.

Unlike the ridge regression problem, the lasso solution cannot be generally given in a closed-form expression. Exceptionally, in the case of an orthonormal input matrix $\boldsymbol{X}$, the lasso solution is

$$\hat{\beta}_j = \text{sign}(\hat{\beta}_j^{ls})(\hat{\beta}_j^{ls} - \lambda)_+, \quad j \in \{1, ..., p\}, \tag{1.3}$$

where $\hat{\beta}_j^{ls}$ is the least squares estimate for the j-th variable and $(\cdot)_+$ indicates the positive part. This is called *soft-thresholding*. On the other hand, the ridge solution in the orthonormal case is

$$\hat{\beta}_j = \hat{\beta}_j^{ls}/(1 + \lambda), \quad j \in \{1, ..., p\}. \tag{1.4}$$

Figure 1.1 compares the ridge and lasso estimators for $p = 1$. The X-axis represents the unrestricted coefficient $\hat{\beta}_1^{ls}$ and the Y-axis represents the corresponding regularized coefficient $\hat{\beta}_1$. Thus, the dotted line corresponds to the unrestricted least squares estimation.

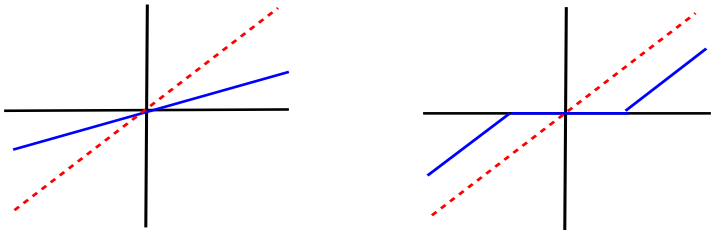


Figure 1.1: Ridge (left) and lasso (right) estimation of a regression coefficient in the orthonormal case. The dotted line corresponds to the unrestricted least squares estimation.

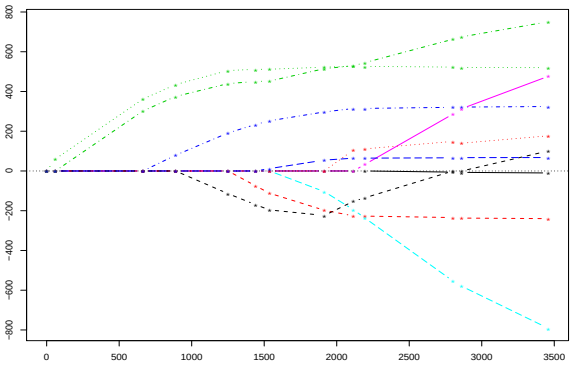


Figure 1.2: Regularization path for the *Diabetes* data set.

The emergence of the *Least Angle Regression* (LARS) algorithm (Efron et al., 2004) boosted the practical possibilities of the lasso enormously. With a slight modification of the basic algorithm, LARS is able to compute the entire lasso regularization path for the general case at the cost of a single ordinary least square fit. A robust version of LARS is proposed by Khan et al. (2007). Also, Fraley and Hesterberg (2009) devise a variation for dealing with large data sets.

The *regularization path* is the entire set of solutions for each λ value. For the regularization path to be efficiently computed, the LARS algorithm takes advantage of its linearity. More specifically, since the regularization path is piecewise linear, we only need to compute the solution for a finite number of λ values. These values represent a variable's removal from or addition to the model. Figure 1.2 shows the regularization path for the *Diabetes* data set, used by Efron et al. (2004). The Y-axis represents the magnitude for the regression coefficients and the X-axis represents the L_1-norm of the vector of coefficients $\boldsymbol{\beta}$. Each coefficient is represented by a different line. All the coefficients are zero at the start of the regularization path, where λ takes a high value. As λ decreases (rightward in the figure), the coefficients evolve towards the least squares solution.

Alternatively, Friedman et al. (2007a) devise a pathwise coordinate optimization algorithm to very efficiently compute the lasso solution for a grid of λ values.

The connexions of the lasso to boosting and forward stagewise regression fitting have been carefully studied in the literature (Efron et al., 2004; Rosset et al., 2004;

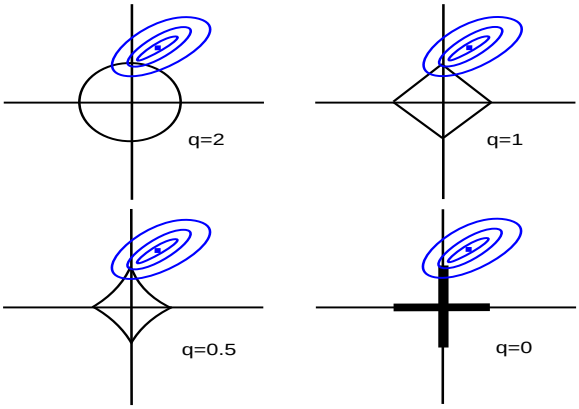


Figure 1.3: Contours of the least squares function (blue) and contours of the constraint regions (black) for some penalties.

Zhao and Yu, 2007; Hastie et al., 2007). This way, the lasso idea can be generalized to any convex loss function and computation is efficient.

Seeking for generality, Rosset and Zhu (2007) analyze the L_1-penalized problems with general loss functions and ascertain under which conditions a LARS procedure can be applied. Specifically, the loss function must be piecewise quadratic as a function of $\boldsymbol{\beta}$ along the regularization path. For example, the authors design a LARS algorithm for an L_1-penalized Huber's loss function. Besides, Wang and Leng (2007) transform different types of loss functions into quadratic approximations that can be computed by the LARS algorithm.

From a Bayesian perspective, least squares is the maximum a posteriori estimate with a non-informative prior on the coefficients. A shrinkage prior centered at zero for the parameters leads to more stable estimates. The L_1-penalty in Equation (1.2) corresponds to a Laplace prior on $\boldsymbol{\beta}$. Since the Bayesian approach considers uncertainty in the parameter estimates, the full posterior is only sparse in the limit of infinite data even though the prior distribution encourages sparse estimates. The Bayesian approach for the lasso is known as the Bayesian lasso (Park and Casella, 2008; Hans, 2010).

Although the L_1 and the L_2-penalties are the most popular, other L_q-penalties are possible. The bridge regression (Frank and Friedman, 1993) generalizes lasso and ridge to $q > 0$. In this case, the penalty in Equation (1.2) becomes $\lambda \, ||\boldsymbol{\beta}||_q^q$. Figure 1.3 illustrates the optimization problem for two variables for various penalties. Note that when $q = 0$, the penalty is just the number of free parameters.

The L_1-penalty's biggest advantage lies in the fact that variable selection only takes place when $q \leq 1$, whereas the related optimization problem is convex only when $q \geq 1$. The lasso is the intersection of both conditions so it achieves variable selection without surrendering the computational advantages of convexity.

Another advantage of the lasso over ridge is that predictions are less biased under some conditions. Ridge pushes all the coefficients towards zero with a force that depends on the regularization parameter and is proportional to the magnitude of the coefficient. In the orthonormal case, for example, the ridge solution is given by Equation (1.4). The lasso, on the other hand, further shrinks coefficients that are effectively discarded. This results in a weaker shrinkage of the coefficients of the

variables that remain within the model. These are supposed to be the most valuable predictors for the regression problem, i.e., they have a bigger influence on the response. There are, however, other situations where ridge outperforms the lasso. For example, ridge dominates the lasso when the predictors are highly correlated (Tibshirani, 1996; Zou and Hastie, 2005). In this case, ridge shrinks the coefficients of all redundant variables so that the total contribution is well-balanced, whereas the lasso tends to drop all but one of the redundant variables.

Greenshtein and Ritov (2004) theoretically study the prediction capability, considering the persistency property: under some conditions, and for an adequate λ choice (that depends on N), the expected squared prediction error approximates the irreducible error (the Bayes error). They show that this holds when the number of parameters grows, at most, at a polynomial rate with regard to N, and the true model is sparse. Also, Candès and Plan (2009) prove, under realistic assumptions, that the lasso achieves a squared error not far from what could be achieved if the true sparsity pattern were known.

Besides prediction performance, the interpretability of the model is often a primary goal. The focus is sometimes on the identification of a simple enough model rather than on the finest prediction accuracy. A sizeable amount of research has looked at the lasso's ability to recover the true sparsity pattern (i.e., to discard the irrelevant variables) and give a consistent estimation of the true coefficients. A consistent estimator produces estimates that converge to the true parameters for a fairly large sample.

For instance, Zhao and Yu (2006) introduce the *irrepresentable condition* concept, already outlined by Meinshausen and Bühlmann (2006). Briefly, in an asymptotic scenario, Zhao and Yu (2006) show that the true model can be recovered only if there is no high correlations between relevant predictors and irrelevant predictors. This condition can be formalized as

$$\max\left((\boldsymbol{X}'_{\vartheta}\boldsymbol{X}_{\vartheta})^{-1}\boldsymbol{X}'_{\vartheta}\boldsymbol{X}_{\vartheta^c}\right) < 1-\epsilon,$$

where $\boldsymbol{X}'_{\vartheta}$ includes the variables with non-zero coefficients in the true model, $\boldsymbol{X}_{\vartheta^c}$ includes the variables with zero coefficients in the true model and ϵ is some positive constant.

More recently, Meinshausen and Yu (2009) examine the lasso behavior if only a relaxed version of the irrepresentable condition is met. Specifically, although the true sparsity pattern cannot be exactly recovered, the estimation of the coefficients can still be consistent (in the L_2-sense) if both the number of relevant variables and the minimum eigenvalue of the design matrix (restricted to the relevant variables) are bounded.

Finally, Xu et al. (2010) show the connexion between the lasso formulation and the robust regression problem, which indicates that the lasso has valuable robustness properties.

1.1.2 *Variants and alternatives to the lasso*

Several modifications have been introduced to the lasso so as either to improve its statistical properties or to adapt a specific problem configuration. We focus on methods that still use an L_1-penalty, omitting methods, such as *Smoothly Clipped Absolute Deviation* (SCAD) (Fan and Li, 2001), that change the type of penalty.

Improving the lasso's properties

We start by describing some alternatives that are proposed to improve the lasso's properties. Any form of regularization introduces a bias in the estimation in exchange for a (hopefully) larger reduction in variance. In addition, when the number of true nonzero coefficients is small relative to p, the lasso, so as to discard the irrelevant variables, introduces considerable bias in the correct variable coefficients. To minimize the bias, the *relaxed lasso* (Meinshausen, 2007) introduces a two-stage estimation. First, the lasso discovers the sparsity pattern. Then, the lasso, with a lower regularization parameter, is again used on the selected variables only. Hence, we have two regularization parameters, one per stage, that need to be estimated for example by cross-validation. A similar idea, using ordinary least squares in the second phase, was already proposed by Efron et al. (2004). This is, however, only possible when $N \geq p$.

In the spirit of the relaxed lasso, the *Variable Inclusion and Shrinkage Algorithm* (VISA) (Radchenko and James, 2008) performs a two-stage estimation with two different regularization parameters. In the second stage, the VISA approach does not definitely discard the variables dropped in the first stage, although it gives a higher priority to the previously selected variables. In this paper, Radchenko and James (2008) provide a convincing theoretical justification of the method.

The *adaptive lasso* (Zou, 2006) penalizes each variable according to its importance. The adaptive lasso estimate is

$$\hat{\boldsymbol{\beta}} = \text{argmin}_{\boldsymbol{\beta}} \, ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 + \lambda \sum_{j=1}^{p} w_j |\beta_j|,$$

where, in the $N \geq p$ case, the weights w_j can be computed as $w_j = 1/|\hat{\beta}_j^{ls}|^\gamma$ ($\gamma > 0$). If $N < p$, the weights can be computed using estimates with minimal regularization instead of $\hat{\boldsymbol{\beta}}^{ls}$. Zou (2006) shows that the adaptive lasso properties are superior to those of the lasso. In particular, the adaptive lasso meets the so-called oracle properties (Fan and Li, 2001): (i) identify the true sparsity pattern; and (ii) have an optimal estimation rate of the coefficients. The adaptive lasso can be considered a convex approximation of the L_q-penalties, with $0 < q < 1$, which have been proven to have the oracle properties (Knight and Fu, 2000). Further theoretical analysis of the adaptive lasso is performed by Huang et al. (2008) and also by Pötscher and Schneider (2009), who study the distribution of the adaptive lasso estimator. The LARS algorithm can be used to compute the adaptive lasso regularization path. Figure 1.4 shows the adaptive lasso regularization path for the *Diabetes* data set; note the differences from Figure 1.2.

Alternatively to the lasso, the *Dantzig selector* (Candès and Tao, 2007) substitutes the sum of squared errors in Equation (1.2) by an L_∞ norm, i.e., the maximum absolute value of the components of the argument. Thus, the Dantzig selector yields

$$\hat{\boldsymbol{\beta}} = \text{argmin}_{\boldsymbol{\beta}} \, ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_\infty + \lambda ||\boldsymbol{\beta}||_1.$$

The Dantzig selector shares some statistical properties with the lasso, particularly as regards the recovery of the true sparsity pattern. However, the Dantzig selector estimator is less stable than the lasso. Bickel et al. (2009) examine the theoretical properties of the Dantzig selector compared to the lasso. James et al. (2009) propose an algorithm to find the entire regularization path for the Dantzig selector.

To achieve further robustness, Wang et al. (2007a) propose the *LAD-lasso*, whose loss function is with regard to the L_1-norm:

$$\hat{\boldsymbol{\beta}} = \text{argmin}_{\boldsymbol{\beta}} \, ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_1 + \lambda ||\boldsymbol{\beta}||_1.$$

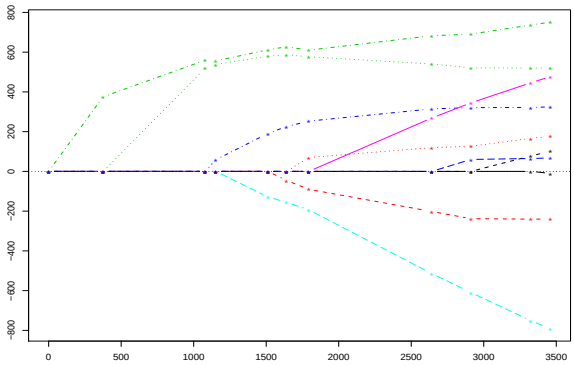


Figure 1.4: Regularization path of adaptive lasso for the *Diabetes* data set.

As mentioned above, when the data set contains strong correlations among the predictors, the ridge's prediction performance is better than the lasso's. Motivated by this, Zou and Hastie (2005) propose the *elastic net*, a popular method that mixes the lasso and the ridge penalties:

$$\hat{\boldsymbol{\beta}} = \text{argmin}_{\boldsymbol{\beta}} \, ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 + \lambda_1 ||\boldsymbol{\beta}||_1 + \lambda_2 ||\boldsymbol{\beta}||_2^2, \tag{1.5}$$

where $\lambda_1, \lambda_2 \geq 0$.

Besides performing better in the presence of correlated predictors, the elastic net has other interesting properties. For the $p > N$ case, in particular, whereas the lasso can select at most N predictors (then the solution saturates), the elastic net can include more than N predictors into the model. Furthermore, assuming that there is some group of relevant and redundant variables, the lasso tends to discard all but one from this group. We often want to keep all redundant (if relevant) variables for interpretation's sake. In microarray analysis, for example, one often wants to identify all the genes involved in a particular process, even when their expression levels are very alike. The elastic net selects the entire group of redundant variables, giving a balanced estimation of their coefficients. Bunea (2008) analyzes the variable selection capability of the elastic net for generalized linear models, compared to the lasso. Figure 1.5 shows the elastic net regularization path (computed with the LARS-EN algorithm, devised by Zou and Hastie (2005)) for the *Diabetes* data set; note the differences from Figure 1.2. Li and Li (2008) modify the elastic net penalty to accommodate prior biological knowledge. Lorbert et al. (2010) extend the elastic net to encourage similar variables to have similar coefficients.

Finally, some authors, like Bach (2008a) or Chatterjee and Lahiri (2011), instead of modifying the penalization criterion, have considered the bootstrap in order to improve the model selection accuracy of the Lasso.

Adapting the lasso to particular problems

Until now, we have described a number of methods that either improve the properties of the lasso or attempt to give an alternative. In the following, we describe several modifications of the lasso for tailoring to particular problem settings. Note that the elastic net could also be included in this subsection.

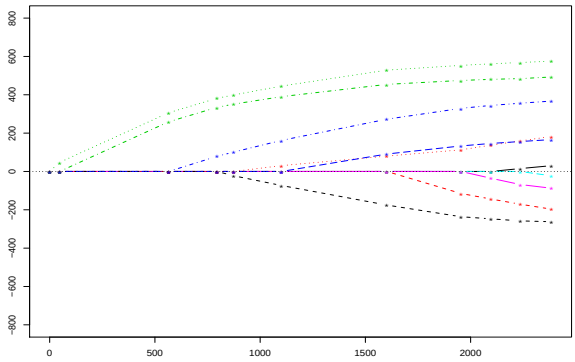


Figure 1.5: Regularization path of the elastic net for the *Diabetes* data set.

On occasions, the variables are grouped beforehand and we are interested in including only entire groups in the model. For this setting, Yuan and Lin (2006) propose the *group lasso*, defined as follows:

$$\hat{\boldsymbol{\beta}} = \text{argmin}_{\boldsymbol{\beta}} \, ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 + \lambda \sum_{j=1}^{J} ||\boldsymbol{\beta}_j||_{W_j}, \tag{1.6}$$

where the set of variables is partitioned into J groups and $\boldsymbol{\beta}_j$ are the parameters of the j-th group. The penalty is defined as $||\boldsymbol{\beta}_j||_{W_j} = (\boldsymbol{\beta}_j' \boldsymbol{W}_j \boldsymbol{\beta}_j)^{1/2}$, where $\boldsymbol{W}_j$ is typically chosen to be the identity matrix. This penalty can be considered a generalization of the L_2-penalty. A trivial application of the group lasso is the presence of categorical variables, each codified as a set of indicator dummy variables. Yuan and Lin (2006) also introduce an efficient LARS-type algorithm to approximate the group lasso solution, giving an exact solution when the design matrix $\boldsymbol{X}$ is orthogonal.

The consistency of the group lasso estimator, under some assumptions, was proved by Bach (2008b). Wang and Leng (2008) propose an adaptive version of the group lasso. Jacob et al. (2009) extend the group lasso for overlapping groups.

The *Composite Absolute Penalties* (CAP) approach (Zhao et al., 2009) generalizes the group lasso by using a specific L_{γ_j}-penalty for each group, plus some L_{γ_0}-penalty to combine the groups:

$$\hat{\boldsymbol{\beta}} = \text{argmin}_{\boldsymbol{\beta}} \, ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 + \lambda \sum_{j=1}^{J} \left(||\boldsymbol{\beta}_j||_{\gamma_j} \right)^{\gamma_0}. \tag{1.7}$$

Different objectives can be pursued by this flexible approach. For example, when L_∞-penalties are used for each group, the coefficients within each group are driven towards equality. Also, the CAP approach can account for hierarchical relationships between the predictors by defining groups with specific overlapping patterns. Zhao et al. (2009) develop an algorithm to compute the entire regularization path for a CAP problem with $\gamma_0 = 1$ and $\gamma_j = \infty$ for all j.

In other problems, variables are ordered significantly and (spatially) close variables should have similar coefficients. The *fused lasso* (Tibshirani et al., 2005) penalizes both the coefficients and the difference between adjacent coefficients:

$$\hat{\boldsymbol{\beta}} = \text{argmin}_{\boldsymbol{\beta}} \, ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=2}^{p} |\beta_j - \beta_{j-1}|.$$

with $\lambda_1, \lambda_2 \geq 0$. The fused lasso is motivated by the problem of analyzing protein mass spectroscopy data, where spatially closer variables (sites) are known to be jointly relevant or irrelevant. The solution of the fused lasso problem can be obtained, for example, by pathwise coordinate optimization (Friedman et al., 2007a).

Finally, the LARS/lasso approach can also be extended for multiresponse regression. Similä and Tikka (2006) propose an extension of the LARS algorithm by modifying the correlation criterion between the predictors and the current residual (which depends on multiple outputs). Unfortunately, the exact regularization path can only be recovered when $\boldsymbol{X}$ is orthonormal. Similä and Tikka (2007) suggest a handy approximation for the general case.

1.2 NONLINEAR REGRESSION

In the real world, the response is unlikely to follow a linear model on the inputs. To a greater or lesser extent, nonlinearity is expectable. In scenarios where data are scarce or not of good quality, a linear model is the best we can do without overfitting the data. There are other cases, however, where it is worth increasing the complexity of the model and moving beyond linearity.

Now, we deal with nonlinearity for predicting a continuous response. In a nutshell, two families of approaches applied to achieve nonlinearity. The first option is to look for a more complex model than the linear model by establishing a linear combination of some basis expansions of the original terms (Schumaker, 2007). Second, we can fit simple (linear) models for different areas of the data domain (Loader, 1999), where we do not have a unique linear model for the entire data set. In the following, we discuss the application of the L_1-penalty to each approach.

1.2.1 *Basis expansions*

Typically, the basis expansion family of methods works with a dictionary of functions (basis expansions), which contains a potentially very large set of elements. These functions are transformations of the original input terms, and can involve one or more input terms.

Regression splines (Schumaker, 2007) are a widely used basis expansion approach. In the univariate case, for instance, the input domain is divided into contiguous intervals, separated by a fixed set of knots. Whereas the placement of the knots can be data-driven, the number of knots is typically specified by the user. In each interval, an m-order polynomial function is fitted, so that the entire function is continuous, and has continuous derivatives up to order $m - 2$ to assure smoothness. These models are called m-order splines. Cubic splines, which use cubic polynomial functions ($m = 4$), are a usual choice. For example, six basis functions are required to represent a cubic spline with two knots, i.e., it has six free parameters; see (Schumaker, 2007) for details.

A more flexible spline-based model is *smoothing splines*. Here, a maximal set of knots is used, and complexity is controlled by penalizing (regularizing) the curvature of the fitted function. For example, assuming a univariate input x_i, the response function can be estimated as

$$\text{argmin}_h \sum_{i=1}^{N} \big(y_i - h(x_i)\big)^2 + \lambda \int h''(t)^2 dt, \tag{1.8}$$

where $h(\cdot)$ encloses the nonlinear expansion of the original terms and fulfills the requirements of being in a reproducing kernel Hilbert space. Note that, in an infinite-dimensional functional space, Equation (1.8) is connected to ridge regression.

Equation (1.8) can be generalized to multivariate fits, by considering a tensor product basis of the basis functions of each variable. Since the dimension of the tensor product basis grows exponentially on the number of the inputs, however, this approach can only be used in low-dimensional settings. The additive model and the smoothing spline analysis of variance (ANOVA) decomposition, described below, are two popular simplifications.

Whatever basis expansion approach we follow, when we use regularization, we typically handle the entire dictionary but we somehow restrict the coefficients of each basis function so as to control the complexity of the model. As with linear regression, L_1-regularization controls both the variance of the estimation and selects which functions from the dictionary are going to be part of the estimated model. The lasso/LARS approach can be applied to any basis expansion set of the original terms.

Additive models (Hastie and Tibshirani, 1990) consider general transformations on each multivariate input $\boldsymbol{x}_i$ separately:

$$\sum_{j=1}^{p} h_j(x_{ij}) + \epsilon, \tag{1.9}$$

where ϵ is the irreducible error term and $h_j(\cdot)$, $j = 1, ..., p$, are smooth functions.

Cubic smoothing splines can be used in an additive model (Wahba, 1990), so that the additive model estimate is given by

$$\begin{aligned} \text{argmin}_{h_1,...,h_p} \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{p} h_j(x_{ij})\right)^2 \\ + \sum_{j=1}^{p} \lambda_j \int h_j''(t)^2 dt, \end{aligned} \tag{1.10}$$

where $h_j(\cdot)$ are univariate cubic smoothing splines and $\lambda_1, ..., \lambda_p$ are p different regularization parameters.

For example, Avalos et al. (2007) consider an additive model with a cubic spline per additive component, decomposing expression (1.10) into a linear component and a nonlinear component. L_1-penalties are applied on the linear and nonlinear components separately. However, since no order restriction is imposed, some nonlinear terms could be retained, whereas their corresponding linear terms are discarded.

The *Sparse Additive Model* (SpAM), proposed by Ravikumar et al. (2009), can use arbitrary smooth functions. Function selection is made by imposing an L_1-penalty on the L_2-norms of the functions, yielding the optimization problem

$$\begin{aligned} \text{argmin}_{h_1,...,h_p} \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{p} h_j(x_{ij})\right)^2 \\ + \lambda \sum_{j=1}^{p} \sqrt{\int h_j^2(t) dt}. \end{aligned}$$

Hence, SpAM can be considered as a functional version of the group lasso (Yuan and Lin, 2006), where the L_2-norms of the functions play the role of the variable

groups. This idea, using kernel functions, was already introduced by Bach (2008b). Unlike the classic additive model, and thanks to regularization, SpAM can be used in high-dimensional settings, as it is more interpretable, less sensitive to overfitting and computationally efficient.

In the same spirit, Huang et al. (2010) apply an adaptive version (Zou, 2006) of the functional group lasso, related to SpAM, to the additive model with spline basis functions. Also, Cui et al. (2011) propose the so-called *lasso-type spline method*, which employs a group lasso-type penalty of the form

$$\lambda \sum_{j=1}^{p} \sqrt{||\boldsymbol{\beta}_j||_1},$$

where $\boldsymbol{\beta}_j$ are the coefficients associated with the (truncated) set of basis terms of the j-th spline basis function.

On the other hand, the *smoothing spline ANOVA* (Wahba, 1990), which generalizes the additive model, is defined as

$$\sum_{j_1} h_{j_1}(x_{ij_1}) + \sum_{j_1<j_2} h_{j_1 j_2}(x_{ij_1}, x_{ij_2}) + \sum_{j_1<j_2<j_3} h_{j_1 j_2 j_3}(x_{ij_1}, x_{ij_2}, x_{ij_3}) + ...,$$

where the expansion is truncated in some way. Functional terms $h_{j_1}(\cdot), h_{j_1 j_2}(\cdot)$, $h_{j_1 j_2 j_3}(\cdot), ...$ are smoothing spline functions over one or more variables. Hence, unlike sparse additive models, smoothing spline ANOVA decomposition considers input interactions and usually leads to more complex models.

The *Component Selection and Smoothing Operator* (COSSO), proposed by Lin and Zhang (2006), estimates a smoothing spline ANOVA model by imposing an L_1-penalty on the component norms instead of the L_2-penalty employed in the traditional smoothing spline ANOVA decomposition estimate. The *Variable selection using Adaptive Nonlinear Interaction Structures in High dimensions* (VANISH) (Radchenko and James, 2010) focus only on main terms and second order interactions, establishing an L_1-based penalty that gives preference to the main terms.

1.2.2 *Kernel smoothing regression*

Now, we deal with the estimation of a different model at each point of the input domain. These models are smooth, in a sense that refers to model continuity along the input domain. This is achieved by including only those data points in the model estimation that are closest (in the input space) to the point of interest. Distances are established by means of a kernel function $K_\tau(\cdot, \cdot)$, where parameter τ accounts for the bandwidth of the kernel.

A popular example of kernel smoothing methods is *locally weighted regression* (Cleveland, 1979; Loader, 1999), which, using the closest data, fits a (typically linear) regression model for each query point. The idea is that, for each point, there is a neighborhood where the regression surface is well approximated by a function from a parametric class. The locally weighted regression procedure is built on classical least squares regression, so that a weighted residual sum of squares, instead of the residual sum of squares, is minimized.

Assuming standardized data, the locally weighted regression solution for the query point $\boldsymbol{x}_k$ is given by

$$\hat{\boldsymbol{\beta}}_k = \text{argmin}_{\boldsymbol{\beta}} \sum_{i=1}^{N} K_\tau(\boldsymbol{x}_i, \boldsymbol{x}_k) \left(y_i - \boldsymbol{x}_i' \boldsymbol{\beta}_k\right)^2. \tag{1.11}$$

Locally weighted regression is usually less useful for large dimensions, in particular when the true underlying model is sparse. Whereas the kernel function is defined to use the entire set of variables, it is more adequate to use only the relevant ones. For the same reason, the direct use of the L_1-penalty on Equation (1.11) is troublesome, because the distances would be calculated prior to the regression, and before knowing what variables are relevant for prediction. Besides, it is convenient to restrict the kernel function to a few variables because of the curse of dimensionality.

The *lazy lasso* (Vidaurre et al., 2011d) combines locally weighted regression with L_1-regularization so as to offset the variance increment of the local fitting and achieve sparse models. To cope with this issue, it alternates variable selection and kernel evaluation. To evaluate the kernel, a vector of bandwidths is computed at each step as a function of the local regression coefficients. This use of sparse vectors of bandwidths avoids the curse of dimensionality. At each step, the kernel function is evaluated, and weights are assigned to data for the next lasso regression.

The *Regularization of Derivative Expectation Operator* (RODEO), proposed by Lafferty and Wasserman (2008), performs simultaneous bandwidth and variable selection by computing the infinitesimal change in the estimation as a function of the smoothing parameters, and then thresholding these derivatives to get a sparse estimate. Following a greedy strategy, RODEO updates the bandwidths at each step. This way, it shrinks the bandwidths of relevant variables more than the bandwidths of irrelevant variables. When soft-thresholding is used, this method is related to the lasso.

The *varying coefficient model* uses a subset of the input variables to define the locality pattern, i.e., the kernel function. In other words, for each value of such a subset (usually a univariate index) we estimate a simple (typically linear) model using the remainder of the variables. Wang and Xia (2009) propose the *L_1-regularized varying coefficient model*. Each data instance is associated with a univariate index variable, ranging between 0 and 1, so that data instances with similar indexes will also have similar regression coefficients. All vectors of regression coefficients, one per data instance, are jointly estimated. Sparsity is pursued with an L_1-penalty.

There are some other scenarios where the kernel function can be computed regardless of the set of input variables. In these settings, the application of L_1-regularization is simplified.

One example is spatial analysis, a different form of local analysis, where the influence of the covariates on the response follows different patterns according to some spatial location of the data (typically 2D coordinates). For instance, as in locally weighted regression, geographically weighted regression (Fotheringham et al., 2002) assigns weights to data, so that nearer data instances are given more importance than further data instances. Unlike locally weighted regression, distances are defined on separate geographical information attached to each instance. The *geographically weighted lasso* (Wheeler, 2009) introduces a lasso-wise penalization on the geographically weighted regression estimated coefficients.

Related to this, Meier and Bühlmann (2007) propose the *smoothed lasso*. Designed for temporal data, the smoothed lasso assigns greater weights to data points that are closer in time. The smoothed lasso is described in Section 1.7.

The applications of the L_1-penalty for supervised classification are an extensive field of research. They are many and variate, both from a frequentist and a Bayesian perspective. In this dissertation, we confine our discussion to three well-known and successful methods: logistic regression, discriminant analysis and support vector machines; see (Hastie et al., 2008) for a detailed overview. The L_1-penalty has also been applied, however, to other paradigms, like e.g., log-linear models (Shi et al., 2008). We also leave out the semi-supervised classification paradigm.

Again, we have variables $\{X_1, ..., X_p, Y\}$, where Y is the response variable. We also have a data set $\boldsymbol{D} = \{(x_{i1}, ..., x_{ip}, y_i), i = 1, ..., N\}$. However, this time $y_i \in \{1, ..., K\}$ is categorical. The objective is to assign any new instance $\boldsymbol{x} = (x_1, ..., x_p)$ to the correct class value in $\{1, ..., K\}$. Both linear and nonlinear L_1-penalized models can be used.

1.3.1 *Logistic regression*

Logistic regression (Hastie et al., 2008) belongs to the wider family of generalized linear models (GLMs). It aims to model the posterior probability of the response or class variable Y, $Pr(Y = k|\boldsymbol{x})$, as a transformation of a linear combination of the inputs. For a K-classes problem, $K - 1$ logit functions are defined as

$$\log \frac{Pr(Y = k|\boldsymbol{x})}{Pr(Y = K|\boldsymbol{x})} = \beta_0^{(k)} + \boldsymbol{x}'\boldsymbol{\beta}^{(k)}, \quad k \in \{1, ..., K - 1\}, \tag{1.12}$$

where $\{\beta_0^{(k)}, \boldsymbol{\beta}^{(k)}\}$ are the logistic linear regression parameters for the k-th class value. The denominator is set to be the K-th class value, but it could be any. This yields

$$Pr(Y = k|\boldsymbol{x}) = \frac{e^{(\beta_0^{(k)} + \boldsymbol{x}'\boldsymbol{\beta}^{(k)})}}{1 + \sum_{l=1}^{K-1} e^{(\beta_0^{(l)} + \boldsymbol{x}'\boldsymbol{\beta}^{(l)})}}, \quad k \in \{1, ..., K - 1\}, \tag{1.13}$$

and, hence, $Pr(Y = K|\boldsymbol{x}) = 1 - \sum_{k=1}^{K-1} Pr(Y = k|\boldsymbol{x})$. The model, including the intercepts, has $(p + 1)(K - 1)$ parameters overall and the maximum likelihood solution can be found by the iteratively reweighted least squares algorithm (IRLS), derived from Newton's method. Note that the inputs can be either the original set of predictors or some expansion thereof, so that nonlinear decision boundaries can be achieved. For example, Park and Hastie (2008) include two-level interactions and regularize with an L_2-penalty (no L_1-penalty is considered, though).

The L_1-penalty for logistic regression was first mentioned by Tibshirani (1996). He formulates the binary classification problem with continuous predictors as the minimization of the L_1-penalized negative log-likelihood function. This function is

$$-\sum_{i=1}^{N} \left(y_i \boldsymbol{x}_i'\boldsymbol{\beta} - \log(1 + e^{(\boldsymbol{x}_i'\boldsymbol{\beta})}) \right) + \lambda ||\boldsymbol{\beta}||_1, \tag{1.14}$$

where the input $\boldsymbol{x}_i$ includes the constant term 1 to integrate the intercept. This problem is solved by applying the original lasso algorithm at each step of the IRLS algorithm. However, the convergence of this method is not guaranteed, and it is not computationally efficient for large-dimensional problems.

For continuous predictors and a binary response, a number of contributions relate the lasso to logistic regression. For example, Roth (2004) adapts the algorithm proposed by Osborne et al. (2000), which solves the lasso, to Equation (1.14), showing the global convergence of the algorithm. Shevade and Keerthi (2003) devise a simple and easy way to implement the algorithm for the same task. Genkin et al. (2007) tackle the same problem in a Bayesian context. Also from a Bayesian perspective: Balakrishnan and Madigan (2008) propose online algorithms to fit an L_1-regularized logistic regression model, so that the entire data set does not have to be stored in memory; van Gerven et al. (2010) reformulate the Laplace prior on $\boldsymbol{\beta}$ as a scale mixture to force similarity between coefficients of nearby variables; Cawley and Talbot (2006) analytically integrate out the regularization parameter so that computations are accelerated. Park and Hastie (2007) develop an efficient regularization path-following algorithm for GLMs based on predictor-corrector methods of convex optimization. Alternatively, Friedman et al. (2010c) present an extremely efficient coordinate descent method for computing the GLM regression coefficients on a grid of λ parameter values for elastic net penalties. Shi et al. (2010) give a comprehensive list of state-of-the-art algorithms for the sparse logistic regression problem. In addition, they propose an algorithm comprising two stages: a fast iterative shrinkage phase and an accurate interior point phase.

Meier et al. (2008) adapt the group lasso (Yuan and Lin, 2006), defined in Equation (1.6), to the binary logistic regression model. The so-called *logistic group lasso* allows for categorical predictors by modeling each categorical predictor as a group of dummy variables. The logistic group lasso then aims to minimize the group L_1-penalized negative log-likelihood function

$$-\sum_{i=1}^{N}\left(y_i\boldsymbol{x}_i'\boldsymbol{\beta}-\log(1+e^{(\boldsymbol{x}_i'\boldsymbol{\beta})})\right)+\lambda\sum_{j=1}^{p}w_j||\boldsymbol{\beta}_j||_2.$$

If the j-th predictor is categorical, $\boldsymbol{\beta}_j$ are the parameters for the set of dummy variables. If the j-th predictor is continuous, $\boldsymbol{\beta}_j$ has only one component. The weights w_j scale the penalty with regard to the dimensionality of $\boldsymbol{\beta}_j$. Meier et al. (2008) devise an efficient algorithm based on pathwise coordinate optimization and prove that the resulting estimator is statistically consistent.

There are considerable additional work on multinomial logistic regression. From a Bayesian perspective, Krishnapuram et al. (2005) introduce a new method for sparse multinomial logistic regression, finding the maximum a posteriori for the formulation in Equation (1.13) with a Laplacian prior distribution on the parameters. Whereas a Gaussian prior (which is equivalent to an L_2-penalty) is easily accommodated into the IRLS algorithm, IRLS cannot handle the Laplacian prior. To estimate the $(p+1)(K-1)$ regression parameters, Krishnapuram et al. (2005) introduce a bound optimization approach with a computational cost equivalent to IRLS.

In situations where the log-likelihood function is not well-behaved and IRLS is not guaranteed to converge, Tian et al. (2008) propose a quadratic lower-bound algorithm to solve the binary L_1-regularized logistic regression, also applicable to the multinomial problem. Cawley et al. (2007) extend their previous results for a binary response (Cawley and Talbot, 2006) to the multinomial case.

Finally, note that the ideas of Section 1.2 can also be applied to logistic regression for achieving nonlinear classification. For example, the logistic additive model replaces the linear expression in Equation (1.12) by a nonlinear additive expression (Equation (1.9)). Ravikumar et al. (2009) target sparse additive logistic regression with

an L_1-penalty. Zhang et al. (2004) formulate logistic regression within the smoothing spline ANOVA decomposition framework, using L_1-regularization.

1.3.2 *Discriminant analysis*

Linear discriminant analysis (LDA) (Hastie et al., 2008) targets a linear separation among a set of classes assuming the inputs to be Gaussian. This is a major difference from logistic regression, which makes no assumption about the input distribution. In addition, LDA assumes a shared covariance matrix for the whole data set. Although this is rarely the case in practice, the simplicity of the resulting model is often worth it. From these assumptions, a set of K linear discriminant functions can be derived:

$$\psi_k(\boldsymbol{x}) = -\boldsymbol{x}'\hat{\boldsymbol{\Sigma}}^{-1}\hat{\boldsymbol{\mu}}_k + \frac{1}{2}\hat{\boldsymbol{\mu}}_k'\hat{\boldsymbol{\Sigma}}^{-1}\hat{\boldsymbol{\mu}}_k + \log\hat{\pi}_k, \quad k \in \{1, ..., K\}, \tag{1.15}$$

so that we assign to $\boldsymbol{x} \in \mathbb{R}^p$ the class k whose linear discriminant function $\psi_k(\boldsymbol{x})$ takes the highest value. Here, $\hat{\boldsymbol{\mu}}_k = \frac{1}{N_k}\sum_{i|y_i=k}\boldsymbol{x}_i$, N_k is the number of instances within the k-class, $\hat{\pi}_k = \frac{N_k}{N}$ is the estimation of the a priori probability of class k, and $\hat{\boldsymbol{\Sigma}}$ is the common covariance matrix of the entire set of vectors $\boldsymbol{x}_i$.

LDA is also useful for projecting the data in lower-dimensional spaces of at most $K-1$ dimensions. These spaces are optimal in the sense that they project the data in the most discriminative directions. The sequence of vectors that define the coordinates of the optimal subspaces are called the discriminant coordinates. This is closely connected to Fisher's reduced rank discriminant analysis.

When the underlying model is supposed to be sparse or the number of variables is high with regard to the number of instances, a variable filtering step usually precedes LDA. There are, however, a few attempts to combine L_1-regularization with discriminant analysis. Before discussing them, we briefly introduce the idea of *penalized discriminant analysis* (PDA) (Hastie et al., 1995).

PDA (Hastie et al., 1995) is based on the formulation of LDA as a regression problem via optimal scoring (Hastie et al., 1994), which turns the categorical response into a quantitative response. The optimal scoring problem is defined as

$$\text{argmin}_{\boldsymbol{\theta},\boldsymbol{\beta}}||\boldsymbol{Y\theta} - \boldsymbol{XB}||_2^2,$$

subject to

$$\frac{||\boldsymbol{Y\theta}||_2^2}{N} = 1, \tag{1.16}$$

where $\boldsymbol{Y}$ is a response indicator $N \times (K-1)$ matrix, $\boldsymbol{B} \in \mathbb{R}^{p\times(K-1)}$ has one vector of regression coefficients per class and the score matrix $\boldsymbol{\theta} \in \mathbb{R}^{(K-1)\times(K-1)}$ has $K-1$ scores per discriminant function. The vectors of scores are chosen to be orthonormal.

The idea is that the $K-1$ vectors of regression coefficients turn out to be, up to a constant, equal to the aforementioned discriminant coordinates. This way, many well-known techniques for regression into discriminant analysis can be plugged, including nonlinear extensions and regularization. In particular, PDA is formulated as follows

$$\text{argmin}_{\boldsymbol{\theta},\boldsymbol{\beta}}||\boldsymbol{Y\theta} - \boldsymbol{XB}||_2^2 + \lambda||\boldsymbol{\Omega}^{1/2}\boldsymbol{B}||_2^2,$$

subject to (1.16), where $\boldsymbol{\Omega} \in \mathbb{R}^{p \times p}$ is some symmetric and positive definite penalization matrix.

Sparse linear discriminant analysis (SDA) (Clemmensen et al., 2011) adds a handy L_1-penalty to PDA to obtain a sparse solution:

$$\text{argmin}_{\boldsymbol{\theta},\boldsymbol{\beta}} ||\boldsymbol{Y\theta} - \boldsymbol{XB}||_2^2 + \lambda_1 ||\boldsymbol{\Omega}^{1/2}\boldsymbol{B}||_2^2 + \lambda_2 ||\boldsymbol{B}||_1.$$

Note that this is closely related to the elastic net formulation in Equation (1.5).

An alternative approach, DALASS (Trendafilov and Jolliffe, 2007), is based on Fisher's formulation of the problem and estimates the discriminant coordinates under a lasso constraint. Both standard and orthogonal discriminant coordinates are considered. In this formulation, however, no L_2-penalty is considered.

The *nearest shrunken centroids* (NSC) method (Tibshirani et al., 2003) is useful for problems where the number of predictors greatly exceeds the number of instances. To simplify the estimation, the authors assume that the predictors are independent within each class, i.e., that the within-class covariance matrix is diagonal. Thus, the discriminant functions of Equation (1.15) simplify to

$$\psi_k(\boldsymbol{x}) = -\sum_{j=1}^{p} \frac{(x_j - \mu_{kj})^2}{W_{jj}} + 2\log \pi_k, \quad k \in \{1, ..., K-1\}, \tag{1.17}$$

where $\boldsymbol{W}$ is the within-class diagonal covariance matrix with diagonal elements W_{jj} and μ_{kj} is the j-th component of $\boldsymbol{\mu}_k$. In this setting, Tibshirani et al. (2003) define

$$d_{kj} = \frac{\mu_{kj} - N^{-1} \sum_{i=1}^{N} x_{ij}}{m_k (W_{jj} + \epsilon)},$$

where $m_k = \sqrt{1/N_k + 1/N}$ and ϵ is some small positive constant. Sparsity is promoted by applying soft-thresholding on d_{kj},

$$d_{kj}^* = \text{sign}(d_{kj})(d_{kj} - \lambda)_+,$$

where $(\cdot)_+$ indicates the positive part. Recall that soft-thresholding and the lasso are closely related (see Equation (1.3)). Now, a sparse version of the centroids $\boldsymbol{\mu}_k$, the shrunken centroids $\boldsymbol{\mu}_k^*$, can be computed as

$$\mu_{kj}^* = \mu_{kj} + d_{kj}^* m_k (W_{jj} + \epsilon), \quad j \in \{1, ..., p\},$$

so that μ_{kj}^* substitutes μ_{kj} in Equation (1.17).

The NSC method is applied, for example, to gene expression classification. For instance, Tai and Pan (2007) classify microarray data using a version of LDA similar to NSC. Instead of using a pure diagonal covariance matrix, Tai and Pan (2007) allow some off-diagonal elements of the covariance matrix to be nonzero (depending on prior knowledge about the problem).

1.3.3 *Support vector machines*

Support vector machines (SVMs) (Cortes and Vapnik, 1995) are a well-known paradigm of non-probabilistic binary classification methods. They can be used in multiclass classification, for example, by performing several pairwise binary classifications. A more sophisticated method adapts the classical SVM binary formulation to the multiclass case (Lee et al., 2004).

Roughly speaking, the standard SVM finds a separating hyperplane maximizing the margin between the training points of the different classes. Let us re-codify the classes so that $y_i \in \{-1, 1\}$. One of the possible formulations for the SVM is

$$\text{argmin}_{\beta_0,\boldsymbol{\beta}} \sum_{i=1}^{N} \Big[1 - y_i(h(\boldsymbol{x})'\boldsymbol{\beta} + \beta_0)\Big]_+ + \lambda||\boldsymbol{\beta}||_2^2, \tag{1.18}$$

where $h(\cdot)$ is some transformation of the inputs, β_0 and $\boldsymbol{\beta}$ are the separating hyperplane coefficients and λ can be considered a regularization parameter controlling the width of the margin. Therefore, the L_2-penalty in (1.18) controls the variance of the estimator.

Considering Equation (1.18) as a regularized function estimation problem, Bradley and Mangasarian (1998) replace the L_2-penalty for an L_1-penalty:

$$\text{argmin}_{\beta_0,\boldsymbol{\beta}} \sum_{i=1}^{N} \Big[1 - y_i(h(\boldsymbol{x})'\boldsymbol{\beta} + \beta_0)\Big]_+ + \lambda||\boldsymbol{\beta}||_1. \tag{1.19}$$

Thanks to the L_1-penalty, the binary SVM can perform feature selection (in the expanded space), leading some coefficients to zero. The whole set of solutions of Equations (1.18,1.19) can be computed by the SVM path algorithm (Hastie et al., 2004), inspired by the LARS algorithm. Further discussion about the L_1-regularized SVM can be found in (Zhu et al., 2003). Derived penalties can also be employed. For example, Wang et al. (2006) use a combination of L_1 and L_2-penalties, whereas Liu and Wu (2007) use a combination of L_0 and L_1-penalties.

Wang et al. (2007b) extend this idea for multiclass SVMs and present an algorithm for finding the regularization path. To do this, they generalize both the hinge loss function (the first term of Equation (1.19)) and the penalty term, formulating the following model:

$$\text{argmin}_{\beta_0,\boldsymbol{\beta}_1,\ldots,\boldsymbol{\beta}_K} \sum_{i=1}^{N} \sum_{k \neq y_i} \Big[1 + (h(\boldsymbol{x}_i)'\boldsymbol{\beta}_k + \beta_0)\Big]_+ + \lambda \sum_{k=1}^{K} ||\boldsymbol{\beta}_k||_1,$$

where $\boldsymbol{\beta}_k$ is the vector of coefficients for the k-class. The loss function was borrowed from (Lee et al., 2004).

Lee et al. (2006b) propose an alternative formulation for the L_1-regularized multiclass SVM based on ANOVA kernels, so as to provide a more interpretable view of the interactions between the predictors and class values.

1.4 UNSUPERVISED CLASSIFICATION

Unsupervised classification, or cluster analysis, deals with the identification of a set of classes, or clusters, within the data using only unlabeled instances. Although the application of L_1-regularization is less obvious in this field, there are still a few approaches that warrant attention.

1.4.1 *Probabilistic methods*

Probabilistic methods model the data instances as independent multivariate observations drawn from some mixture model with K components, each corresponding to a

cluster. A well-known approach is to employ a finite mixture of normal distributions with a shared diagonal covariance matrix. The mixture model is defined as

$$f(\boldsymbol{x}_i; \boldsymbol{\mu}, \boldsymbol{\sigma}) = \sum_{k=1}^{K} \pi_k f_k(\boldsymbol{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\sigma}),$$

where K is the number of clusters, $f_k(\cdot)$ are normal multivariate density functions, $\boldsymbol{\mu}_k \in \mathbb{R}^p$ is the vector of means for the k-th cluster, $\boldsymbol{\mu}$ is a $p \times K$ matrix whose k-th column is $\boldsymbol{\mu}_k$, $\boldsymbol{\sigma} \in \mathbb{R}^p$ is the vector of standard deviations that define the shared diagonal covariance matrix and π_k is the a priori probability of cluster k. In this setting, a variable j is dropped if the μ_{jk} values are equal for all k. To favor this condition, Xie et al. (2007) employ the EM algorithm (Dempster et al., 1977) to maximize the L_1-penalized log-likelihood for standardized data,

$$\sum_{i=1}^{N} \log f(\boldsymbol{x}_i; \boldsymbol{\mu}, \boldsymbol{\sigma}) - \lambda \sum_{k=1}^{K} \sum_{j=1}^{p} |\mu_{jk}|.$$

To achieve a sparser solution, Xie et al. (2008) simultaneously penalize, by a group penalty, all the mean parameters of each variable, so that the optimization criterion is

$$\sum_{i=1}^{N} \log f(\boldsymbol{x}_i; \boldsymbol{\mu}, \boldsymbol{\sigma}) - \lambda \sqrt{K} \sum_{j=1}^{p} ||\boldsymbol{\mu}_{j.}||_2,$$

where $\boldsymbol{\mu}_{j.} = (\mu_{j1}, ..., \mu_{jK})$. This is a group lasso-type penalty. Also, to include prior knowledge about grouped variables, Xie et al. (2008) propose the maximization of

$$\sum_{i=1}^{N} \log f(\boldsymbol{x}_i; \boldsymbol{\mu}, \boldsymbol{\sigma}) - \lambda \sum_{j'=1}^{J} \sqrt{Kp^{(j')}}\, ||\boldsymbol{\mu}_{j'}||_2,$$

where j' iterates throughout the J a priori defined groups, $p^{(j')}$ is the number of variables in group j and $\boldsymbol{\mu}_{j'}$ is the vector that results from concatenating vectors $\boldsymbol{\mu}_{j.}$ such that j belongs to group j'.

Alternatively, Wang and Zhu (2008) formulate the maximization criterion

$$\sum_{i=1}^{N} \log f(\boldsymbol{x}_i; \boldsymbol{\mu}, \boldsymbol{\sigma}) - \lambda \sum_{j'=1}^{J} \sqrt{Kp^{(j')}}\, ||\boldsymbol{\mu}_{j'}||_\infty,$$

where $||\cdot||_\infty$ is the L_∞-norm, i.e., the maximum absolute value of the components of the argument.

Note that these last two penalties are actually a generalization of the CAP penalty (Equation (1.7)).

1.4.2 *Other methods*

Witten and Tibshirani (2010) propose a general framework that, provided the optimizing criterion is additive on the features, establishes a weighted additive formulation, where weights are constrained by both L_1 and L_2-penalties:

$$\begin{aligned} &\text{argmax}_{\boldsymbol{w},\boldsymbol{\theta}} \sum_{j=1}^{p} w_j f_j(\boldsymbol{x}_{\cdot j}, \boldsymbol{\theta}_j) \\ &\text{s.t.} \quad ||\boldsymbol{w}||_2^2 \leq 1, \quad ||\boldsymbol{w}||_1 \leq s, \quad w_j \geq 0, \ \forall j, \end{aligned}$$

where $\boldsymbol{x}_{\cdot j}$ is the j-th columnn of $\boldsymbol{X}$, w_j is a weight assigned to variable j and $\boldsymbol{\theta}_j$ is some set of parameters for variable j.

Witten and Tibshirani (2010) apply this idea to derive sparse versions of K-means and hierarchical clustering, whose optimizing criterion is additive on the features. Let C_k be cluster k. For L_1-regularized K-means, they obtain $\{C_1, ..., C_K, \boldsymbol{w}\}$ by maximizing

$$\sum_{j=1}^{p} w_j \Big(\sum_{i=1}^{N} \sum_{i'=1}^{N} d_{ii'}^{(j)} - \sum_{k=1}^{K} \frac{1}{N_k} \sum_{i \in C_k} \sum_{i' \in C_k} d_{ii'}^{(j)} \Big)$$
$$\text{s.t.} \quad ||\boldsymbol{w}||_2^2 \leq 1, \quad ||\boldsymbol{w}||_1 \leq s, \quad w_j \geq 0, \ \forall j,$$

where $d_{ii'}^{(j)}$ is defined as $(x_{ij} - x_{i'j})^2$ and N_k is the number of instances in cluster k.

In the standard hierarchical clustering, the overall dissimilarity matrix (which is used as an input for the algorithm) is defined by the elements $d_{ii'} = \sum_{j=1}^{p} d_{ii'}^{(j)}$. For L_1-regularized hierarchical clustering, the objective is to find $\boldsymbol{w}$ and elements $d_{ii'}$ by maximizing

$$\sum_{j=1}^{p} w_j \sum_{i=1}^{N} \sum_{i'=1}^{N} d_{ii'}^{(j)} d_{ii'} \quad \text{s.t.}$$
$$\sum_{i=1}^{N} \sum_{i'=1}^{N} d_{ii'}^2 \leq 1, \ ||\boldsymbol{w}||_2^2 \leq 1, \ ||\boldsymbol{w}||_1 \leq s, \ w_j \geq 0, \ \forall j.$$

Then, usual hierarchical clustering is performed on this similarity matrix, leading to a sparse solution.

In a different paradigm, Elhamifar and Vidal (2009) combine spectral clustering (Ng et al., 2001) and L_1-regularization to carry out unsupervised classification on multiple affine subspaces embedded in a high-dimensional space. The idea is to cluster the data into separate subspaces, assuming that data instances are drawn from a union of subspaces. Elhamifar and Vidal (2009) use sparse representations, obtained by L_1-regularization, for this purpose. Then, spectral clustering is used on the similarity matrix induced from these subspaces.

1.5 GRAPHICAL MODELS

A graphical model (Whittaker, 1990; Lauritzen, 1996; Koller and Friedman, 2009), composed by a set of nodes and a set of edges and their corresponding parameters, codifies and provides a visual representation of the factorization of the probability distribution of the set of variables. Each variable is represented by a node, and the edges encode the conditional independences among different triplets of variables. Graphical models can be undirected or directed so that they are respectively called Markov networks and Bayesian networks (Pearl, 1988; Koller and Friedman, 2009). In Markov networks the edges are undirected, whereas in Bayesian networks the edges are appropriately called arcs. There are dependences that can be represented by a Markov network but not by a Bayesian network, and vice versa. Besides, the nodes can represent either continuous or discrete variables.

There is a vast amount of literature about graphical models, and their uses are manifold. In this dissertation, we focus on the structure induction problem, i.e., the

discovery of the set of conditional independences over a set of p nodes. The L_1-penalty has been used to promote sparse graphs, which lead to simpler probability distributions and more interpretable structures. There is plenty of research on sparse Markov networks by L_1-regularization, and less on sparse Bayesian networks. In either case, we intend to give an overview and some significant examples rather than providing a full coverage of the topic.

1.5.1 *Markov networks*

We deal with pairwise Markov networks, that permit only interactions between pairs of nodes and do not consider higher-order interactions. Thus, the absence of an arc between two nodes means that the respective pair of variables are conditionally independent. Unlike Bayesian networks, Markov networks can represent cyclic dependences.

Continuous Markov networks

When the variables encoded by the Markov network are continuous, they are usually assumed to be Gaussian-distributed. Although other distributions are possible, the Gaussian distribution yields handy and easy to analyze models. A Gaussian distribution is defined by a vector of means $\boldsymbol{\mu}$ and a covariance matrix $\boldsymbol{\Sigma}$. The inverse of the covariance matrix (or precision matrix) represents the conditional independences between the variables. Therefore, if the precision matrix has a zero at position (j_1, j_2), there is no arc between nodes j_1 and j_2 in the corresponding Gaussian Markov network. The joint distribution on $\boldsymbol{x} \in \mathbb{R}^p$ is given by

$$Pr(\boldsymbol{x}) = \frac{1}{(2\pi)^{(p/2)}|\boldsymbol{\Sigma}|^{1/2}} \exp\Big(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})'\boldsymbol{\Omega}(\boldsymbol{x}-\boldsymbol{\mu})\Big),$$

where $\boldsymbol{\Omega} = \boldsymbol{\Sigma}^{-1}$ is the precision matrix.

The parameters of the precision matrix of a Gaussian Markov network are usually estimated by maximizing the log-likelihood, defined (up to a constant) as

$$\log\det(\boldsymbol{\Omega}) - \text{trace}(\boldsymbol{\Omega}\hat{\boldsymbol{\Sigma}}), \tag{1.20}$$

where $\hat{\boldsymbol{\Sigma}}$ is the empirical covariance matrix.

Besides, there is a connection between the Gaussian distribution parameters and multivariate linear regression. In particular, it can be shown that

$$\boldsymbol{\beta}_j = -\frac{\boldsymbol{\Omega}_{\cdot j}}{\Omega_{jj}}, \tag{1.21}$$

where $\boldsymbol{\beta}_j$ is the regression coefficient for the j-th variable on the remaining variables and $\boldsymbol{\Omega}_{\cdot j}$ is the j-th column of $\boldsymbol{\Omega}$.

In light of this, Meinshausen and Bühlmann (2006) propose to discover the sparsity pattern of a Gaussian Markov network by carrying out, individually, an L_1-regularized linear regression for every variable on the other variables. Hence, an edge between nodes j_1 and j_2 is created when the regression of variable j_1 on the rest selects variable j_2 (its coefficient is non-zero), *or* when the regression of variable j_2 on the rest selects variable j_1 (alternatively to this strategy, an *and* strategy can be followed). This approach is usually called the *neighborhood selection* method. With an appropriate selection of the penalization parameter, and under certain assumptions, the neighborhood selection method has been proven to be consistent for sparse high dimensional graphs. Specifically, the chosen regularization parameter of each L_1-regularized linear

regression is the one that outputs the best prediction of the variable given the others. However, the lasso estimate is based only on individual regressions and ignores the overall likelihood of the network. Besides, this method is asymmetric in the sense that the regression of j_1 could select j_2, whereas the regression of j_2 does not select j_1.

Peng et al. (2009) introduce a procedure that can be considered a symmetric version of the neighborhood selection method. This method estimates $\mathbf{\Omega}$ by explicitly forcing the estimator to be positive definite. Friedman et al. (2010a) also look for a symmetric version of the neighborhood selection method by adapting the group lasso (Yuan and Lin, 2006). The groups are defined by all the edges that connect to a given node, whereby some nodes are disconnected from the rest of the graph. In other words, the objective is sparsity at the node level instead of at the edge level.

To estimate a sparse precision matrix at one shot (and thus a sparse Gaussian Markov network), the *graphical lasso* (Friedman et al., 2007b) maximizes an L_1-penalized version of Equation (1.20),

$$\log \det(\mathbf{\Omega}) - \text{trace}(\hat{\mathbf{\Sigma}}\mathbf{\Omega}) - \lambda||\mathbf{\Omega}||_1, \tag{1.22}$$

using a blockwise coordinate descent strategy. The graphical lasso is based on the work of Banerjee et al. (2008), and can be considered an efficient version of their proposal. Also, Duchi et al. (2008) extend Banerjee et al.'s work to produce sparse blocks within the precision matrix.

Alternatively, Cai et al. (2011) propose the *constrained L_1-minimization for inverse matrix estimation* (CLIME):

$$\min||\mathbf{\Omega}||_1 \quad \text{s.t.} \quad |\hat{\mathbf{\Sigma}}\mathbf{\Omega}|_\infty \leq \lambda,$$

a convex optimization problem that can be decomposed into separate vector minimization problems and, thus, can be efficiently solved even for high dimensions. However, the resulting estimation is not symmetric, so a symmetrizing procedure must be performed afterwards.

A formulation similar to the graphical lasso, but excluding the diagonal elements from the penalty, is also taken by Yuan and Lin (2007) and Rothman et al. (2008).

Instead of directly penalizing the number of edges in the graphical model, Yuan (2010) exploits the relation between the precision matrix estimation and multivariate linear regression (Equation (1.21)) to give a preliminary estimation of $\mathbf{\Omega}$ in a similar fashion to Meinshausen and Bühlmann (2006) but using the Dantzig selector (Candès and Tao, 2007). Afterwards, the estimated precision matrix is simply performed to achieve symmetry, so that the result is optimal in some sense. The objective differs from Meinshausen and Bühlmann's. Whereas Meinshausen and Bühlmann (2006) focus on discovering the sparsity pattern, Yuan (2010) aims to find a proper $\mathbf{\Omega}$ estimation.

Also, Yuan (2010) realizes that a high dimensional precision matrix can be optimally estimated insofar as it can be approximated by a graphical model with a moderate degree. That is, not only does the true graphical structure has to be sparse according to the usual L_1-criterion (Equation (1.22)), but also the maximum number of edges per node needs to be under some bound.

Liu et al. (2009) generalizes the Gaussian assumption to a semiparametric distribution, much like the additive models (Hastie and Tibshirani, 1990) do for linear regression. They replace the vector of variables $(X_1, ..., X_p)$ by their functional counterparts $h(X_1, ..., X_p) = (h_1(X_1), ..., h_p(X_p))$, so that $h(X_1, ..., X_p)$ is Gaussian distributed. Sparsity is provided by an L_1-penalty.

When the variables are ordered, the precision matrix sparsity is usually introduced via the modified Cholesky decomposition. The precision matrix can be factorized (Cholesky decomposed) as $\boldsymbol{\Omega} = \boldsymbol{L}'\boldsymbol{D}^{-1}\boldsymbol{L}$, where $\boldsymbol{L}$ is a lower triangular matrix with ones in the diagonal and $\boldsymbol{D}$ is a diagonal matrix. The elements of $\boldsymbol{L}$ below the diagonal can be interpreted as the regression coefficients of each variable on its precessors. Note that, without an ordering, we cannot assume $\boldsymbol{L}$ to be triangular. Huang et al. (2006) use direct L_1-regularization of the coefficients of $\boldsymbol{L}$. Levina et al. (2008) also use the Cholesky decomposition of the precision matrix and apply a modification of the L_1-penalty that is shown to produce better results.

Discrete Markov networks

The Ising model is the most popular discrete Markov network and the one we discuss in this dissertation. All the nodes in the Ising model represent binary variables. Although hidden nodes are sometimes allowed, we deal with Ising models whose nodes are all "visible". The Ising model distribution is given by

$$\frac{1}{\psi(\boldsymbol{\theta})} \exp\Big(\sum_{(a,b)\in E} \theta_{ab} x_a x_b \Big),$$

where $\boldsymbol{x} \in \{0,1\}^p$, $\boldsymbol{\theta}$ is a $\binom{p}{2}$-dimensional vector of parameters (so that $\theta_{j_1 j_2}$ is zero if nodes j_1 and j_2 are not connected), E is the set of edges and the partition function $\psi(\boldsymbol{\theta})$ ensures that the distribution adds up to one. Whereas the normalization term can be computed in polynomial time in the Gaussian model, the evaluation of $\psi(\boldsymbol{\theta})$ is intractable for discrete models.

Lee et al. (2006a) maximize the L_1-regularized log-likelihood of the Ising model (penalizing $\boldsymbol{\theta}$) by a conjugate gradient procedure. They use an approximation of the partition function so as to reduce the computational burden. Although this approximation is exact for trees, it is somewhat troublesome in the presence of loops.

To avoid the evaluation of $\psi(\boldsymbol{\theta})$, Ravikumar et al. (2010) follow the line of Meinshausen and Bühlmann (2006) for the continuous case and estimate the graph structure by fitting one L_1-penalized logistic regression per node. As happens with the neighborhood selection method, the estimation is based only on individual regressions and ignores the overall likelihood of the network.

Another way around $\psi(\boldsymbol{\theta})$ evaluation is to use a pseudo-likelihood consisting of the product of univariate conditionals (Besag, 1975). The pseudo-likelihood maximization problem can be expressed as a set of logistic regression problems, which is susceptible of including an L_1-penalization. This is the approach taken by Schmidt et al. (2008) and Höfling and Tibshirani (2009).

1.5.2 *Bayesian networks*

Directed acyclic graphical models, also referred to as Bayesian networks, have directional arcs instead of undirected edges. This way, no directed cycles are formed. Although there are several Bayesian network models with restricted topologies, like trees or polytrees, we only discuss the structure induction of general Bayesian networks.

Bayesian networks are often estimated from some ordering on the variables. In what follows, however, we do not assume any given order. Typically, graph structure estimation is performed by either constrained-based algorithms (detecting conditional independences between triplets of variables) or score and search-based methods.

To represent a Bayesian network, we can choose between directed acyclic graphs (DAGs) and partial directed acyclic graphs (PDAGs).

Gaussian Bayesian networks

In Gaussian Bayesian networks, the density function of the joint probability distribution is Gaussian and can be expressed as the product of p univariate conditional normal densities.

Li and Yang (2005) estimate a DAG with lasso-based regressions, which iteratively alternates coefficient estimation and regularization parameter estimation. From a Bayesian perspective, they use a Wishart prior distribution for the precision matrix. The DAG prior is derived from this precision matrix prior, and this DAG prior turns out to be equivalent to a Laplace prior.

Vidaurre et al. (2010) take a hybrid approach where L_1-regularization is used as an edge filtering step, prior to a greedy search through the space of Markov-equivalent structures (using PDAGs). Assuming the lasso to be an ideal variable selector (specifically, if it does not miss true relations), the greedy search is limited to the Markov blanket of each variable. In this case, the optimality properties of the greedy search algorithm, proved by Nielsen et al. (2003), are preserved.

Discrete Bayesian networks

Discrete Bayesian networks (or Bayesian networks) are directed acyclic graphical models with discrete nodes. Unlike discrete Markov networks, Bayesian networks are decomposable and hence there is no need to compute the partition function. Schmidt et al. (2007) also use an L_1-regularization prefiltering step, followed by a greedy search procedure in the DAG-space, for the induction of Bayesian network structures.

1.6 FEATURE EXTRACTION

Dimensionality reduction amounts to the reduction of the number of variables in the data. The dimensionality reduction process is useful in many applications and constitutes a research field by itself. It can be categorized into feature selection and feature extraction. Feature selection intends to find a subset of the original variables that is useful for some subsequent task. The lasso or any of its variants can be used for feature selection. In this section, we focus on the feature extraction approach, which seeks for some transformation of the data into a lower dimensional space that somehow preserves the properties of the raw initial data.

1.6.1 *Principal component analysis*

Discriminant analysis, defined above, is in some sense a feature extraction method, because it allows to project the data onto an affine subspace of $K - 1$ dimensions, where K is the number of classes. Thus this dimensionality reduction is oriented to classification. Now, we deal with more general dimensionality reduction approaches, particularly those intended to capture maximal variance.

We assume an $N \times p$ data matrix $\boldsymbol{X}$ and we want to reduce the dimension p into a handier, lower dimensional q. We focus on the most popular feature extraction method, principal component analysis, along with some nonlinear extensions. The simplest version is the principal component analysis (PCA) method (Jolliffe, 2002), which obtains a sequence of linear approximations of the data. This sequence forms a rank-q ($q < p$) orthogonal matrix. We assume standardized data. PCA is based on the eigen decomposition of the sample covariance matrix, $\boldsymbol{X}'\boldsymbol{X}/N$, defined as

$$\boldsymbol{X}'\boldsymbol{X} = \boldsymbol{V}\boldsymbol{D}^2\boldsymbol{V}',$$

where $\boldsymbol{V}$ is a $p \times p$ orthogonal matrix spanning the row space of $\boldsymbol{X}$ (an orthogonal basis) and $\boldsymbol{D}$ is a $p \times p$ diagonal matrix, with diagonal entries $d_1 \geq d_2 \geq ... \geq d_p$. Such values are the singular values of $\boldsymbol{X}$. If any value d_j is zero, $\boldsymbol{X}$ is not full-rank. The columns of $\boldsymbol{V}$, $\{\boldsymbol{v}_1, ..., \boldsymbol{v}_p\}$, are known as the principal components directions of $\boldsymbol{X}$ and also called the loadings.

Therefore, the PCA transformation of $\boldsymbol{X}$ is built by the first q columns of $\boldsymbol{V}$, i.e., the ones corresponding to the highest singular values. The linear combination $\boldsymbol{z}_1 = \boldsymbol{X}\boldsymbol{v}_1$ has the highest variance among all the linear combinations of the columns of $\boldsymbol{X}$, and the linear combination $\boldsymbol{z}_2 = \boldsymbol{X}\boldsymbol{v}_2$ has the highest variance given that $\boldsymbol{v}_2$ is orthogonal to $\boldsymbol{v}_1$, and so on.

A direct application of the L_1-penalty properties to PCA is the so-called *sparse PCA*, proposed by Zou et al. (2006). Sparse PCA emerges from the observation that PCA is a linear combination of all the original variables, and hence the result is often not easy to interpret. Instead, Zou et al. (2006) aim to ascertain which variables are more relevant for the decomposition. This is achieved when the loadings $\{\boldsymbol{v}_1, ..., \boldsymbol{v}_p\}$ are sparse. To accomplish this goal, sparse PCA exploits the connection between PCA and regression to impose a lasso penalty (or any other variant). Assuming centered data, the proposed formulation is

$$(\hat{\boldsymbol{V}}_q, \hat{\boldsymbol{\theta}}) = \text{argmin}_{\boldsymbol{V}_q, \boldsymbol{\theta}} \sum_{i=1}^{N} ||\boldsymbol{x}_i - \boldsymbol{\theta}\boldsymbol{V}_q'\boldsymbol{x}_i||_2^2 + \lambda||\boldsymbol{v}||_1,$$
$$\text{s.t.} \quad \boldsymbol{\theta}'\boldsymbol{\theta} = \boldsymbol{I},$$

where $\boldsymbol{V}_q$ is the $p \times q$ loadings matrix ($\boldsymbol{V}$ restricted to the first q columns), $\boldsymbol{\theta} \in \mathbb{R}^{p \times q}$ and $\boldsymbol{I}$ is the $q \times q$ identity matrix. Although this criterion is not convex, when either $\boldsymbol{V}_q$ or $\boldsymbol{\theta}$ is fixed, the criterion becomes convex in the other parameter matrix. Thus, an alternating optimization problem is performed until convergence. The elastic net penalty (Zou et al., 2006) or the adaptive lasso penalty (Leng and Wang, 2009) have been also proposed.

d'Aspremont et al. (2007) devise a path-following algorithm for sparse PCA. This algorithm is based on a semidefinite relaxation of the problem. Farcomeni (2009) proposes to use an L_0-penalty instead of an L_1-penalty for explicitly controlling the degree of sparseness. Meng et al. (2011) replaces the L_2-norm in the loss function by an L_2-norm so as to achieve further robustness. From a Bayesian perspective, Gao (2008) introduces a sparse PCA formulation by replacing the Gaussian distribution of the noise used by the probabilistic PCA (Tipping and Bishop, 1999) by a Laplacian distribution. The resulting model is expected to be robust against outliers and promote sparsity.

Prior to Zou et al.'s research, Jolliffe et al. (2003) introduced a more direct approach to achieve sparsity, where the only parameters to be optimized are $\boldsymbol{V}_q$, i.e., without including $\hat{\boldsymbol{\theta}}$ in the formulation. This approach, called *Simplified Component Technique-Lasso* SCoTLASS, has a high computational cost, because it is not a convex optimization problem. However, a recently proposed modification of the singular value decomposition (Witten et al., 2009) is able to compute the SCoTLASS solution efficiently.

Kernel PCA (Schölkopf et al., 1999) extends PCA by expanding the original variables with some nonlinear transformation and then applying PCA on these transformed

data. As far as we know, no attempt has been made to apply the L_1-principle to the kernel PCA model.

1.6.2 *Independent component analysis*

Independent component analysis (ICA) (Hyvärinen et al., 2001) aims to separate the different sources from which some multivariate data are generated. ICA can be considered a variation of the latent variable model, described in (Bartholomew and Knott, 1999), which also aims to identify the latent sources of some data. The difference from the latent variable model is that ICA is built under the assumption of mutual statistical independence and non-Gaussianity of the sources, whereas the classical latent variable model assumes non-correlated, Gaussian distributed data. Note that correlation only implies second-order interactions. Independence, on the other hand, is a more general term that involves all the cross-moments. These assumptions mean that the ICA problem has a unique solution, whereas the traditional latent variable model suffers from lack of unicity.

Let us assume p components or sources. The ICA model is defined as

$$\boldsymbol{X} = \boldsymbol{S}\boldsymbol{A}' + \boldsymbol{\epsilon},$$

where $\boldsymbol{A}$ is an orthogonal $p \times p$ matrix of loadings and $\boldsymbol{S}$ is an $N \times p$ matrix that encodes the latent variables or factors, which represent common sources of variation for $\boldsymbol{X}$. Hence, the columns of $\boldsymbol{S}$ represent non-Gaussian, independent variables. It is assumed that $\boldsymbol{X}'\boldsymbol{X} = N\boldsymbol{I}$ and $\boldsymbol{S}'\boldsymbol{S} = N\boldsymbol{I}$. The objective is to find $\boldsymbol{A}$ such that $\boldsymbol{S}$ fulfills the mentioned conditions. $\boldsymbol{A}$ is typically estimated by information theory techniques, such as the minimization of the mutual information between the components of $\boldsymbol{X}\boldsymbol{A}$. This amounts to maximizing the departures from Gaussianity of the estimates, when the estimates are constrained to be uncorrelated.

Hyvärinen and Karthikesh (2002) propose the sparse ICA, to achieve a sparse estimation of $\boldsymbol{A}$ by imposing a convenient prior on the distribution of $\boldsymbol{A}$. This prior is related to the Laplacian prior used in the lasso.

We have included ICA in this section because it entails important applications for dimensionality reduction; see, for example, (Wang and Chang, 2006). However, it is also frequently used for signal analysis and also could be included in the next section.

Part II

REGULARIZATION IN NONPARAMETRIC REGRESSION

This part deals with the nonparametric estimation of a continuous response. Whatever approach we follow, we must somehow control the nonlinearity or complexity of the model. More complex models are less biased in exchange for increased variance. In general terms, we would choose simpler models for limited or ill-posed data and more complex models for well-behaved data. The objective is a model that optimizes the bias-variance trade-off, that is, that minimizes the expected prediction error. Automatic, data-dependent methods are preferred to control the model complexity. For example, techniques based on regularization are useful for adjusting the complexity of the model and restricting its variance by imposing some constraint on the model parameters.

There are two fundamental approaches for achieving nonlinearity. First, we can fit simple (linear) models for different areas of the data domain (Loader, 1999). Second, we can seek a more complex model than the linear model by establishing a linear combination of some basis expansions of the original terms (Schumaker, 2007). Within this part, chapters 2 (Vidaurre et al., 2011d) and 3 (Vidaurre et al., 2012) discuss two methodological contributions in the field of local linear regression, which belong to the first category of nonlinear models. Chapter 4 presents an analysis of different nonlinear approaches for modeling instantaneous firing rate prediction of single biological neurons, particularizing for neurons in the primary visual cortex (Vidaurre et al., 2011e). It is shown that regularization can play a key role in this setting. This type of models, which aim to understand the neural spiking activity as a function of a biological signal, are called encoding models.

2

THE LAZY LASSO

2.1 INTRODUCTION

Consider p independent covariates $X = \{X_1, ..., X_p\}$ and a response variable Y. Let $\boldsymbol{X}$ and $\boldsymbol{y} = (y_1, ..., y_N)$ be, respectively, an $N \times p$ data matrix and a continuous-valued vector, so that each row $\boldsymbol{x}_i$ is iid related to a continuous response y_i by means of some unknown (nonlinear) function $m(\cdot)$:

$$y_i = m(\boldsymbol{x}_i) + \varepsilon_i,$$

where $m(\cdot)$ is assumed to be sparse and ε_i is the irreducible error term, with $\mathrm{E}[\varepsilon_i|\boldsymbol{x}_i] = 0$. Therefore, $\mathrm{E}[y_i|\boldsymbol{x}_i] = m(\boldsymbol{x}_i)$. We denote the elements of $\boldsymbol{X}$ as x_{ij}.

The objective is to estimate the response at a point of interest $\boldsymbol{x} = (x_1, ..., x_p)$ using a sparsity assumption: only a subset of the covariates are indeed relevant for the estimation. We denote as $\boldsymbol{X}^*$ the data matrix $\boldsymbol{X}$ centered at $\boldsymbol{x}$ and augmented with a first column of ones, so that $x^*_{i0} = 1$ for all $i = 1, ..., N$. The homoscedasticity assumption is not strictly necessary, so that we can generically define the variance of ε_i as $\mathrm{Var}[\varepsilon_i|\boldsymbol{x}_i] = s^2(\boldsymbol{x}_i) = \sigma_i^2$.

Multivariate local regression (Loader, 1999) estimates a multivariate regression function valid for some neighborhood of $\boldsymbol{x}$. This function is often linear, corresponding to a first-order Taylor approximation of $m(\cdot)$ at $\boldsymbol{x}$, and can be defined on the original covariates or on some set of basis functions defined on the original covariates. We consider for simplicity the first case, although the generalization is straightforward. Local regression is appealing from both theoretical and practical sides. On the one hand, it is known to enjoy 100% minimax efficiency for some choice of bandwidth and kernel (Fan, 1993; Ruppert and Wand, 1994). On the other hand, it is computationally fast, easy to implement, flexible and robust to data design (Hastie and Loader, 1993).

The neighborhood is defined by a kernel function, which assigns weights $\boldsymbol{w} = (w_1, ..., w_N)$ to the data points in the data set on the grounds of their distance to $\boldsymbol{x}$. The kernel function has a bandwidth parameter, which strongly influences the estimation. In this chapter, we use a single value h for all regressors. The kernel function is defined as

$$w_i^2 = K_{\boldsymbol{h}}(\boldsymbol{x}_i - \boldsymbol{x}) = \prod_{j=1}^{p} \frac{1}{h} K\left(\frac{x_{ij} - x_j}{h}\right), \quad i = 1, ..., N, \tag{2.1}$$

where $K(\cdot)$ is a univariate, symmetric and nonnegative function with a compact support, such that $\int K(t)dt = 1$.

The estimated local linear regression function $g(\cdot)$ is defined by a vector of local coefficients $\boldsymbol{\beta}(\boldsymbol{x}) = (\beta_1(\boldsymbol{x}), ..., \beta_p(\boldsymbol{x}))^t$ and an intercept term $\hat{\beta}_0(\boldsymbol{x})$. For simplicity of notation, in the following we denote $\hat{\beta}_0(\boldsymbol{x})$ as $\hat{\beta}_0$ and $\hat{\boldsymbol{\beta}}(\boldsymbol{x})$ as $\hat{\boldsymbol{\beta}}$. Then, we have

$$\hat{y}_i = g(\boldsymbol{x}_i) = \hat{\beta}_0 + \boldsymbol{x}_i^t \hat{\boldsymbol{\beta}}.$$

Since the data is centered at $\boldsymbol{x}$, we have $\hat{y} = g(\boldsymbol{x}) = \hat{\beta}_0$. In the following, we denote $(\hat{\beta}_0, \hat{\boldsymbol{\beta}})$ as $\hat{\boldsymbol{\beta}}^*$, In the simplest case, we can estimate $\hat{\boldsymbol{\beta}}^*$ as

$$\hat{\boldsymbol{\beta}}^* = \left(\boldsymbol{X}^{*t} \boldsymbol{W} \boldsymbol{X}^*\right)^{-1} \boldsymbol{X}^{*t} \boldsymbol{W} \boldsymbol{y},$$

where $\boldsymbol{W} = \text{diag}(\boldsymbol{w}^2)$. We can interpret $\hat{\boldsymbol{\beta}}$ as an estimation of the gradient $\partial m(\boldsymbol{x})/\partial x_j$. For estimation of second derivatives, we would need at least a second-order fit.

There are two main motivations for introducing variable selection within the local regression methodology. First, since the best rate of convergence in nonparametric regression is $N^{-4/(4+p)}$ (Györfi et al., 2002), to exploit the sparse nature of $m(\cdot)$ is extremely convenient because, otherwise, the convergence is impractically slow if p is high. Second, ideally, only relevant covariates should be considered by the kernel function. If the solution is sparse, there will be several irrelevant covariates involved in the weights calculation, yielding an incorrect weighting scheme and a rather inaccurate prediction.

A possible naïve approximation would be to add an L_1-penalty to the locally weighted regression so as to reach a sparse solution (some regression coefficients equal to zero). This implies that the kernel function is evaluated before performing variable selection. Therefore, distances are calculated prior to the regression, and hence before we know what variables are relevant for prediction. We claim that this method is naïve and ineffective, and it is expected to lead to incorrect predictions and incorrect feature selections. This effect will be more pronounced for a large number of irrelevant variables. We will call this method the *naïve lazy lasso*.

The approach taken by Lafferty and Wasserman (2008), so-called *regularization of derivative expectation operator*, or *rodeo*, considers a diagonal bandwidth, and is of special interest to us because they consider sparsity in $m(\boldsymbol{x})$. Specifically, they use the estimated gradient of the regression function with respect to the bandwidth, $\partial m(\boldsymbol{x})/\partial \boldsymbol{h}$, to conduct a greedy search, considering that a high value of $\partial m(\boldsymbol{x})/\partial h_j$ is indicative of the relevance of variable X_j. In other words, this gradient tells how the regression function varies with infinitesimal changes of the bandwidth. If it varies little, then the variable is considered to be irrelevant and will be assigned a relatively large bandwidth. This approach, then, assumes that relevance of the variables depends just on how much they depart from the linear model. This method assumes a known value of σ^2. If σ^2 is unknown, it has to be separately estimated.

In this dissertation, we suggest an iterative algorithm that alternates variable selection and distance computation. At each step, distances are computed using the current variables in the model, and weights are assigned to data for the next L_1-regression, which is based in the LARS algorithm (Efron et al., 2004). We use a single overall bandwidth parameter, and the effective bandwidth of each variable is adaptively adjusted in the distance calculation stage. Besides, since, at each step, only a subset of the variables is involved, unlike rodeo, we mostly avoid large matrix inversions. Also, we devise a validation procedure that implies no additional cost. This work appear in the published paper (Vidaurre et al., 2011d).

In the framework of functional data analysis, the stepwise algorithm proposed by Ferraty et al. (2010) is related to ours. Whereas Ramsay and Silverman (2005) have popularized the functional data analysis field, the first nonparametric contributions are described by Ferraty and Vieu (2006) and more recently by Ferraty et al. (2010).

Now, we have an infinite (or very high) dimensional functional variable and a scalar response. The goal is to reduce the very high set of predictors to a set of highly predictive points, called design points. This method performs a greedy, forward addition of variables guided by the cross-validated error. At each step, the variable that, together with the variables currently in the model, most improves the accuracy is selected. A backward deletion process is subsequently applied.

There are, however, substantial differences between this method and ours. First, Ferraty et al's method is not a lazy approach, that is, it is not focused on a certain point of interest and considers the whole data set. Hence, the goals are different, although both algorithms could be adapted to pursue any objective. Second, the estimation procedure and the search strategy also differ. Ferraty et al. (2010) take the weighted least squares estimation at each step. Instead, we obtain a weighted L_1-regularized estimation, choosing the extent of regularization that most improves the model. Given that the whole L_1-regularization path can be obtained at the same cost that a least squares fit, and since their method performs one least squares fit per candidate variable at each step, our method is computationally more efficient when the number of variables is high. This is supported also for a cheap validation procedure. Furthermore, whereas their method adds one variable at a time (and deletes one variable at a time afterwards), our method can add and delete several variables at each step, possibly enlarging the number of different visited models. Finally, we compute the adaptive bandwidths directly as a function of the importance of each variable. Ferraty et al. (2010), on the other hand, perform a heuristic search for this purpose.

2.2 THE ALGORITHM

Cleveland and Devlin (1988) discussed the need to incorporate a variable selection procedure into the loess methodology if required, i.e. if we suspect the presence of irrelevant variables. Taking up this argument, we present an algorithm that combines L_1-regularization with the usual locally weighted regression paradigm.

We assume the hypothesis of local homoscedasticity, that is, σ_i^2 is supposed to be constant within a certain neighborhood. We are interested in the local regression coefficients $\hat{\boldsymbol{\beta}}^*$ minimizing

$$\sum_{i=1}^{N}\left(w_i y_i - w_i\beta_0 - \sum_{j=1}^{p} w_i x_{ij}\beta_j\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j|, \tag{2.2}$$

where the weights $\boldsymbol{w}$ are computed by the redefined kernel function

$$w_i^2 = K_{\boldsymbol{h}}(\boldsymbol{x}_i - \boldsymbol{x}) = \prod_{j\in\Omega}\frac{1}{h}K\left(\frac{x_{ij}-x_j}{h}\right), \quad i = 1,...,N, \tag{2.3}$$

where Ω is the set of relevant covariates. We set Ω to contain those variables that are relevant for the prediction of the response.

Since the problem of estimating simultaneously Ω and $\boldsymbol{\beta}$ cannot be solved analytically, we propose an iterative procedure, the *lazy lasso*, that calculates distances based on the current Ω at each step.

In the first iteration, we let $\Omega = \{1,...,p\}$ and we calculate $\boldsymbol{w}$ using Equation (2.3). As in loess, we weight the data set. Then, the LARS algorithm is run on this weighted data set to solve the minimization in (2.2). From the resulting LARS regularization path, we select the best vector of regression coefficients according to some criterion

Algorithm 1 lazy lasso

Initialization:
$\Omega = \{1, ..., p\}$
$overallBest := \infty$; $toStop := 0$
repeat
  Compute $\boldsymbol{w}$ with Equation (2.3)
  $\boldsymbol{W}^{(l)} := N \times N$ diagonal matrix, with $W_{ii} = w_i$
  $\boldsymbol{Z} := \boldsymbol{W}^{(l)}\boldsymbol{X}$
  $\boldsymbol{v} := \boldsymbol{W}^{(l)}\boldsymbol{y}$
  $\boldsymbol{path} := LARS(\boldsymbol{Z}, \boldsymbol{v})$
  $\tilde{\boldsymbol{\beta}}^* := best(\boldsymbol{path}; \boldsymbol{Z}, \boldsymbol{v})$
  Update Ω, including only those variables whose $\tilde{\beta}_j \neq 0$.
  $score := evaluate(\hat{\boldsymbol{\beta}}^*; \boldsymbol{Z}, \boldsymbol{v})$
  if $score \geq overallBest$ **then**
    $toStop := toStop + 1$
  else
    $toStop := 0$
    $overallBest := score$
    $\hat{\boldsymbol{\beta}}^* := \tilde{\boldsymbol{\beta}}^*$
  end if
until $toStop = \kappa$
$\hat{y} := \beta_0$

(see below for details). We update Ω according to this vector of regression coefficients (including only those covariates whose $\beta_j \neq 0$) Weights are again recalculated using the new Ω set, and the data set is weighted. Subsequently, LARS is run again over this weighted data set. The algorithm alternates LARS and weights calculation until some stopping criterion is met. Here, we stop the process when there is no improvement in the best score for a given number of iterations. This procedure can be complemented with an estimation of h by some plug-in procedure, using only those variables in Ω. This is computationally cheap for a single bandwidth. For details, see, for example, (Yang and Tschernig, 1999).

The pseudocode in Algorithm 1 roughly outlines the method. In the pseudocode, $\boldsymbol{path}$ is the LARS regularization path and $\boldsymbol{\beta}^*$ is the best set of regression coefficients at each iteration. The $evaluate(\cdot)$ and $best(\cdot)$ functions are based on the validation procedures that we detail in the next section. The algorithm terminates if there is no improvement in the best score for κ iterations.

To keep the local homoscedasticity assumption for the weight-ed data set, we have chosen the kernel function to be a k-nearest neighborhood function. This function assigns $w_i = 1$ for the τN data items closest to $\boldsymbol{x}^{(l)}$ and $w_i = 0$ otherwise. Hence, the weighted data set is just a subset of the original data set with constant σ_i^2 for because they are the closest points. As discussed below, local homoscedasticity is a requirement for the validation procedure. Although other functions, like the *Epanechnikov* function or the *tricube* function, are much more frequent in local regression, the weight function mainly affects the visual quality of the regression curve and does not significantly influence the prediction accuracy (Loader, 1999). However, both the *Epanechnikov* and the *tricube* functions alter the data set so that we cannot assume σ_i^2 to be homogeneous within a certain neighborhood.

Validation plays a crucial role in the lazy lasso. On the one hand, a specific point of the regularization path must be selected from each LARS run. On the other hand, a final solution should be selected from the final lazy lasso sequence. Hence, the number of solutions for evaluation can be considerably large. An efficient evaluation method is thus required.

We first deal with model selection along the LARS regularization path. Since we assume local homoscedasticity and we have used the k-nearest neighborhood function to weight the data set, we can now reasonably assume σ_i^2 to be constant for the weighted data set (that is, within this neighborhood of $\boldsymbol{x}$).

The Mallows' C_p statistic (Mallows, 1973), which needs $\sigma_i^2 = \sigma^2$ for all i, is defined as

$$C_p = \sum_{i=1}^{N} \frac{\left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j\right)}{\sigma^2} - n + 2\nu, \tag{2.4}$$

where ν is the effective degrees of freedom of the model. Since we are interested only in $\hat{\boldsymbol{\beta}}$, we use the usual C_p naturally adapted for local fitting at point $\boldsymbol{x}$ (Cleveland and Loader, 1996):

$$C_p(\boldsymbol{x}) = \sum_{i=1}^{N} \frac{w_i\left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j\right)}{\sigma^2(\boldsymbol{x})} - tr(\boldsymbol{W}) + 2\nu, \tag{2.5}$$

where $tr(\boldsymbol{W}) = \sum_{i=1}^{N} w_i^2$. A reasonable estimator for the (constant) local noise variance $\sigma^2(\boldsymbol{x})$ is

$$\hat{\sigma}^2(\boldsymbol{x}) = \sum_{i=1}^{N} \frac{w_i\left(y_i - \beta_0^{LS} \sum_{j=1}^{p} x_{ij}\beta_j^{LS}\right)}{tr(\boldsymbol{W}) - \nu}, \tag{2.6}$$

where β_j^{LS} corresponds to the $\lambda = 0$ (OLS) fit of the LARS regularization path. Finally, an unbiased estimation $\hat{\nu}$ for the lasso is the number of non-zero predictors in the model (Zou et al., 2007). Note that the k-nearest neighborhood weighting scheme also simplifies the estimation of ν. Therefore, the $C_p(\boldsymbol{x})$ assessment requires no additional computations, since $\hat{\sigma}^2(\boldsymbol{x})$, $\hat{\nu}$ and the residual sum of squares are LARS products.

From the above, we can evaluate the solutions that LARS outputs for a given weighted data set. This corresponds to the $best(\cdot)$ function in Algorithm 1. Unfortunately, this procedure does not work for comparing solutions from different LARS runs. This is because we do not have a universal $\sigma^2(\boldsymbol{x})$ estimation for different weighting schemes.

Leave-one-out cross-validation through a local version of the prediction sum of squares (PRESS) procedure (Allen, 1974) is a common and computationally efficient choice for validation in local learning; see (Cleveland and Loader, 1996; Loader, 1999) for details. It does not need a $\sigma^2(\boldsymbol{x})$ estimation. The PRESS statistic is defined as

$$PRESS(\boldsymbol{x}) = \frac{1}{tr(\boldsymbol{W})} \sum_{i=1}^{n} \left(\frac{w_i\left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j\right)}{1 - H_{ii}} \right)^2, \tag{2.7}$$

where $\boldsymbol{H}$ is the hat matrix, such that $\hat{\boldsymbol{v}} = \boldsymbol{H}\boldsymbol{v} = \boldsymbol{H}\boldsymbol{W}\boldsymbol{y}$. Diagonal elements H_{ii}, also called leverages, quantify the influence of the observed response on the fitted response for each data item. $\boldsymbol{H}$ has no direct closed form for the lasso but can be derived from a

linear approximation to the lasso fit (Tibshirani, 1996). Transforming the lasso penalty into a Lagrangian penalty $\sum_j \beta_j^2/|\beta_j|$, $\boldsymbol{H}$ becomes

$$\boldsymbol{H} = \boldsymbol{Z}(\boldsymbol{Z}^T\boldsymbol{Z} + \lambda\boldsymbol{B}^-)^{-1}\boldsymbol{Z}^T, \tag{2.8}$$

where $\boldsymbol{B}$ is a diagonal matrix such that $B_{jj} = |\hat{\beta}_j|$ and $\boldsymbol{B}^-$ is the $\boldsymbol{B}$ pseudoinverse. Note that, in the evaluation of (2.5), we could calculate ν as $tr(\boldsymbol{H})$. However, this is computationally expensive and, as noted by Efron et al. (2004), the accuracy gain is often negligible.

Note that Equation (2.7) includes a weighted residual sum of squares for all data items $\boldsymbol{x}_i$. In principle, it should involve calculating a vector of regression coefficient for each i. However, the PRESS statistic can make efficient use of $\hat{\boldsymbol{\beta}}$ instead of an ad-hoc estimation for each data item. In this chapter, moreover, we use the k-nearest neighborhood function, so weights are either 1 or 0. Therefore, the numerator in (2.7) is just the usual residual sum of squares within some neighborhood of $\boldsymbol{x}$. Equation (2.7) is the $evaluate(\cdot)$ function in Algorithm 1.

2.4 EXPERIMENTS

Now, we describe some experimental results on synthetic and real data sets that illustrate the behaviour of the lazy lasso and the naïve lazy lasso algorithms, compared to the lasso, loess, regression tree (RT) and rodeo.

We perform leave-one-out validation. For each data set (with N instances), we have built N models; for each model, the point of interest is one different data item, whose response is unknown, and the $N-1$ remaining data items make up the data set itself.

2.4.1 *Synthetic data sets*

The algorithms have been first tested on several data sets, generated from three different nonlinear controlled models: *m1*, *m2* and *m3*. Model *m1* represents the scenario of a single sparse nonlinear function. The sparse condition is expected to be detrimental for loess, which calculates the distances over all variables including the irrelevant ones. Models *m2* and *m3* are a more complex case, where the function generating the response and the sparsity pattern vary across the data set. Roughly speaking, the response may be obtained either from a single function for the entire data set (*m1*) or from different functions for different locations in the covariate space (*m2* and *m3*).

From each model, we have simulated 50 data sets. All generated data sets have $N = 2000$ samples.

The three models have $p = 100$ covariates, but only some covariates are relevant. Whereas for *m1* the subset of relevant covariates is constant for the whole data set, this subset varies across the data set in *m2* and *m3*. For *m1*, all covariates are sampled from a Gaussian distribution with mean $\mu = 0$ and standard deviations $\sigma^2 = 1$. For *m2* and *m3*, whereas irrelevant covariates are sampled from a Gaussian distribution with mean $\mu = 0$, relevant covariates have been sampled from a Gaussian distribution with non-zero mean (see below). Standard deviation is equal to 1 for all covariates in *m2* and *m3*.

The prediction function for model *m1* is

$$y_i = x_{i1}^2 + 5\sin x_{i2} + x_{i3}x_{i4} + \epsilon_i, \quad i = 1, \ldots, N,$$

where $\epsilon_i \sim \mathcal{N}(0, 0.25)$ for all data data items. Relevant covariates in *m1* are thus $X_j, j \in \{1, 2, 3, 4\}$, and irrelevant covariates are $X_j, j \notin \{1, 2, 3, 4\}$.

Models *m2* and *m3* have four different prediction functions, indexed, say, by $z = 1, 2, 3, 4$. These prediction functions have an equal probability of $1/4$ of generating responses. For model *m2* such prediction functions are linear:

$$y_i = 2x_{i\,3(z-1)+1} + 0.5x_{i\,3(z-1)+2} - x_{i\,3(z-1)+3} + \epsilon_i, \quad z = 1, 2, 3, 4,$$

whereas for model *m3*, they are nonlinear:

$$y_i = x^2_{i\,3(z-1)+1} + 5 \sin x_{i\,3(z-1)+2} + x_{i\,3(z-1)+3} + \epsilon_i, \quad z = 1, 2, 3, 4.$$

Hence, there are three relevant covariates for each data item, and covariates $X_j, j \in \{13, ..., 100\}$ are always irrelevant.

As mentioned earlier, for models *m2* and *m3*, relevant covariates are sampled from a Gaussian distribution with non-zero mean μ_j; specifically, when $z = 1$, $\mu_j = -3$ $(j = 1, 2, 3)$; when $z = 2$, $\mu_j = -1$ $(j = 4, 5, 6)$; when $z = 3$, $\mu_j = 1$ $(j = 7, 8, 9)$; and when $z = 4$, $\mu_j = 3$ $(j = 10, 11, 12)$. Regarding the noise term ϵ_i, we set $\sigma_i = 0.2$ for $z = 1, 3$ and $\sigma_i = 0.4$ for $z = 2, 4$.

Firstly, we ran a set of tests using constant bandwidths for all the data items in each data set. We experimented with values ranging from $2p/N$ $(= 0.1)$ to $8p/N$ $(= 0.4)$. Figure 2.1 summarizes the results over the 50 data sets. Rows correspond, respectively, to models *m1*, *m2* and *m3*. The charts in the left column illustrate the mean error against the bandwidth. The charts in the right column illustrate the mean number of selected variables against the bandwidth. We display the output of lasso and RT as a reference. Rodeo is not considered here because the bandwidth selection is always adaptive.

The proposed iterative algorithm outperforms the naïve approach and the other algorithms in most cases. Excepting *m1*, where RT error is lower than lazy lasso error for bandwidths over 0.28, lazy lasso accuracy is always the best. The improvement over loess is specially remarkable. The difference between lazy lasso and naïve lazy lasso accuracies is also significant. This is more marked for *m3*, which turns out to be the most difficult data set. On the other hand, the number of selected variables is similar for the lazy lasso and the naïve lazy lasso, and much lower than for the lasso. Interestingly, the number of selected covariates for the lazy lasso and the naïve lazy lasso approximates that of RT when the bandwidth moves up from the lowest values. Although not shown in the Figure 2 for space reasons, the number of correctly selected variables does not vary much for different bandwidths.

Figure 2.2 shows the boxplot of the error and number of selected variables for an adaptive bandwidth. Table 2.1 shows the mean number (and standard deviation) of correctly selected variables. Even though lasso and RTs do not need a bandwidth parameter, both have been included for comparison purposes.

Regarding prediction performance, the lazy lasso achieves by far the lowest mean error and the lowest standard deviation in all cases. This can be interpreted as a measure of robustness. Furthermore, the lazy lasso is shown to have a good variable selection ability. It selects a higher number of correct variables than the other methods, excepting rodeo, at the expense, however, of selecting more irrelevant variables than the naïve lazy lasso and RT. Although rodeo selects always all the relevant variables, it clearly selects, with lasso, the highest total number of variables. The differences are statistically significant.

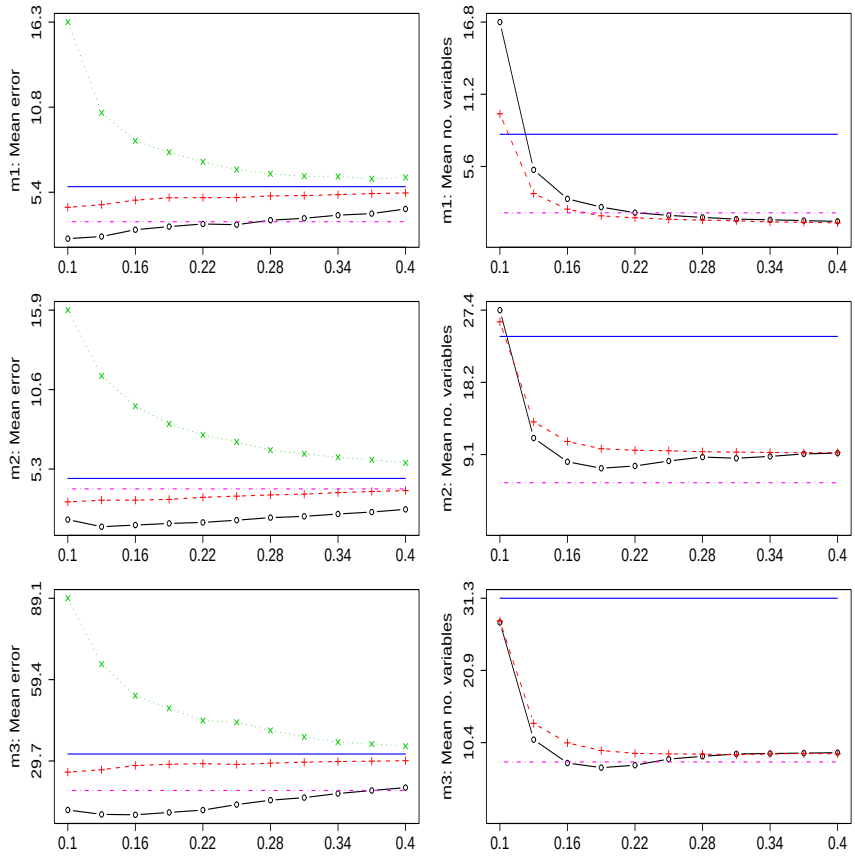


Figure 2.1: Evolution of the mean error (left) and the mean number of selected variables (right) for an increasing bandwidth, for *m1* (top), *m2* (middle) and *m3* (bottom). Solid-◯ lines represent the lazy lasso, dashed-+ lines represent the naïve lazy lasso, dotted-× lines represent loess, solid straight lines represent the lasso and dashed-dotted straight lines represent RT. Loess is not in the right-hand plots because it does not select variables.

From this synthetic setting, we conclude that the devised lazy lasso algorithm can outperform other nonlinear methods like loess, RT, rodeo or the naïve lazy lasso.Given that the lasso is a linear method, its performance is, as expected, worse for the presented scenarios.

2.4.2 *Pumadyn data set*

Next, we test the algorithm on the *Pumadyn* data set, a realistic simulation of the dynamics of a Puma 560 robot arm. It is available from the Delve Repository[2].

Pumadyn is a family of data sets rather than a single data set. The number of covariates may be eight or 32. The data may be either linear or non-linear. Finally, the amount of noise in the output can be set to moderate or high. All combinations of these three parameters are possible, but we confine our study to the non-linear option. However, we introduce a new parameter: the number of incorporated irrelevant variables, i.e., randomly generated variables not related to the response. Let p_0 be the number of variables of the original data set (eight or 32). We have generated new data

2 http://www.cs.toronto.edu/ delve/data/datasets.html

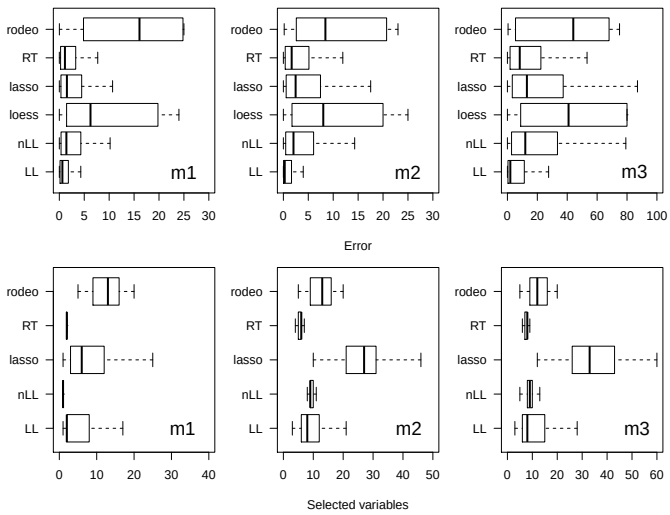


Figure 2.2: Boxplots of the error (top) and total number of selected variables (bottom), for models *m1*, *m2* and *m3*. Tested algorithms are the lazy lasso with adaptive bandwidth (LL), the naïve lazy lasso with adaptive bandwidth (nLL), the lasso, RT and rodeo.

sets by adding p_0, $2p_0$ and $3p_0$ irrelevant variables to each original data set. All data sets have $N = 8192$ data items.

Figure 2.3 shows the result of the experiments. The bandwidth is selected adaptively for the local algorithms. We ran a leave-one-out validation scheme. We do not show the number of correctly selected variables here because it is not clear which variables from the original set are really relevant. In general, all the algorithms have mostly discarded the added irrelevant variables.

As observed, the proposed method generally produces lower estimation errors than loess, the lasso and the naïve approximation. The results for RT are also very competitive, and rodeo, although selects more variables than the others, performs very well in the 32 variables data sets. Note that loess performances are better than the lasso for the moderate (m) noise data sets, whereas the lasso is better for the high (h) noise case. As expected, the more flexible the model is, the more likely it is to be affected by noise. Interestingly, the lazy lasso outperforms the lasso and loess in both cases.

Regarding the number of selected variables, there is not a dominant method. RT and lazy lasso select a reasonable amount of variables in most data sets. Rodeo often selects more variables than the other approaches. On average, the lasso appears to select more variables than the proposed local methods. Loess does not perform variable selection.

Table 2.1: Mean and standard deviations of the number of correctly selected variables (out of four) for models *m1*, *m2* and *m3*. The best result for each row is highlighted. The symbol $*$ is added when the difference to the second best method is statistically significant with a significance level of 0.05.

	LL	Naïve LL	Lasso	RT	Rodeo
m1	3.2($\pm$0.8)	1.2($\pm$0.4)	1.5($\pm$0.7)	1.9($\pm$0.2)	**4.0**($\pm$0.0)$*$
m2	**2.8**($\pm$0.4)$*$	2.0($\pm$0.3)	2.7($\pm$0.4)	1.2($\pm$0.5)	**2.8**($\pm$0.34)$*$
m3	**2.8**($\pm$0.2)$*$	2.2($\pm$0.6)	2.7($\pm$0.5)	1.6($\pm$0.6)	**2.8**($\pm$0.34)$*$

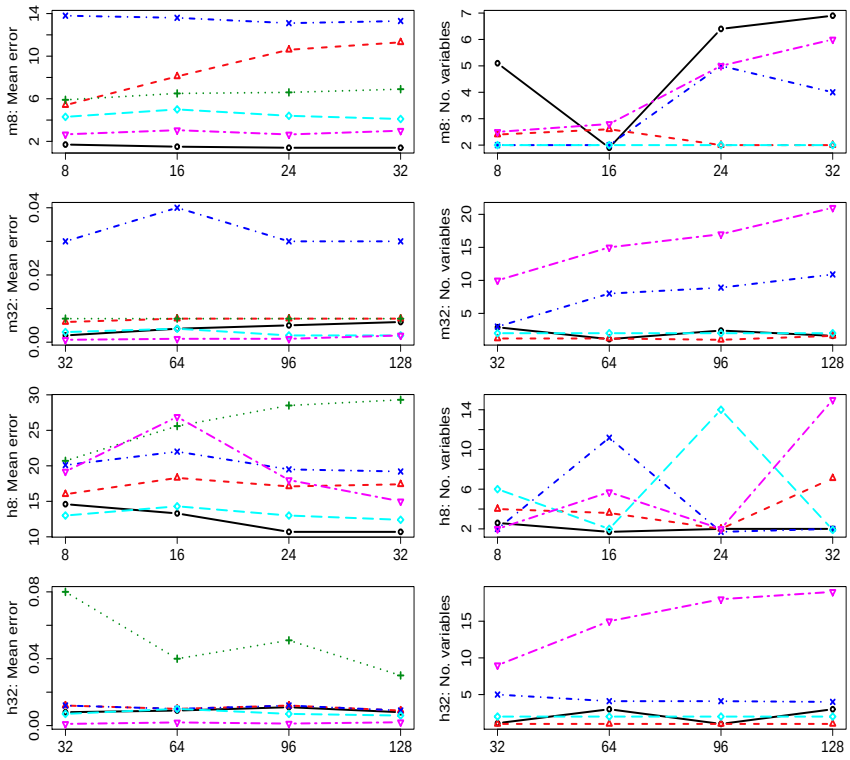


Figure 2.3: Evolution of the mean error (left) and the mean number of selected variables (right) for different amounts of artificially added variables in *Pumadyn* data sets. Each row corresponds to some amount of noise (moderate (m) or high (h)) and some number of variables in the original data set (8 or 32). The X-axis represents the total number of variables in the data set, including those artificially added. Solid-◯ lines represent the lazy lasso, dashed-△ lines represent the naïve lazy lasso, dotted-+ lines represent loess, dashed-dotted-× represent the lasso, dashed-◊ lines represent RT and dashed-dotted-▽ lines represent rodeo.

2.4.3 *Neuroscience fMRI data*

Finally, we report algorithm performance on the *StarPlus* data set[1], extracted from the neuroscience field. This is functional magnetic resonance imaging (fMRI) data collected at Carnegie Mellon University.

Experiments are conducted on six subjects and forty trials per subject. For each trial, the subject is shown a picture for four seconds and a sentence for four seconds. The objective is to discriminate between these two mental states: "picture" or "sentence" Each data item matches a unique 3-dimensional image. Images are captured every 0.5 seconds. Hence, each trial has 16 useful images. Briefly, there are six data sets, one per subject, and they all have $N = 40 \times 16 = 640$ data items. On the other hand, each image has a number of voxels, split into 25 localized regions of interest (ROIs). Here, instead of considering each individual voxel, we will use the mean activation of voxels at each ROI. Therefore, our data set has $p = 25$ covariates.

The brain's inherent complexity moves us to consider nonlinear models. This is possible thanks to the dimensionality reduction resulting from the use of ROIs instead

1 http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-81/www/

Table 2.2: Ratio of incorrectly classified images (top) and the total number of selected variables (bottom) for the *StarPlus* data set. Tested algorithms are the lazy lasso with adaptive bandwidth (LL), the naïve lazy lasso with adaptive bandwidth (Naïve LL), loess with adaptive bandwidth, the lasso, RT and rodeo. The best result for each row is highlighted in bold. The symbol $*$ is added when the difference to the second best method is statistically significant with a significance level of 0.05. Loess has been omitted from the variable selection report because it does not perform variable selection.

	Error					
Subject	LL	Naïve LL	Loess	Lasso	RT	Rodeo
04799	0.49	0.5	0.47	**0.43**	0.44	0.44
04820	0.45	0.5	0.45	0.44	0.46	**0.30**$*$
04847	0.33	0.45	0.39	0.4	0.46	**0.31**
05675	0.39	0.46	0.43	**0.38**	0.42	0.41
05680	**0.32**	0.39	0.37	0.35	0.34	0.35
05710	0.39	0.52	0.45	0.4	0.41	**0.35**

	Number of selected variables					
Subject	LL	Naïve LL	Loess	Lasso	RT	Rodeo
04799	3.9($\pm$5.6)	**1.4**($\pm$2.7)$*$	–	19.8($\pm$0.7)	18.9($\pm$1.3)	8.0($\pm$1.6)
04820	4.9($\pm$5.3)	**1.4**($\pm$2.3)$*$	–	9($\pm$0.3)	18.3($\pm$1)	17.5($\pm$0.5)
04847	7.2($\pm$7.5)	**2.5**($\pm$4)$*$	–	24.9($\pm$0.3)	17.5($\pm$1.3)	9.2($\pm$0.4)
05675	7.6($\pm$7.2)	**2.2**($\pm$3.9)$*$	–	11.4($\pm$0.8)	19.2($\pm$0.7)	10.1($\pm$1.5)
05680	5.2($\pm$4.4)	**2.9**($\pm$3.1)$*$	–	22.7($\pm$0.6)	15($\pm$1.7)	6.5($\pm$4.0)
05710	8.1($\pm$5.2)	**2.1**($\pm$3.7)$*$	–	15.4($\pm$1.5)	19.8($\pm$1.1)	10.1($\pm$6.7)

of individual voxels. In addition, a sparse model is helpful to identifying which brain regions are involved in this task. These factors make the lazy lasso adequate for modeling this kind of data.

Table 2.2 presents the results. For each data set, we performed N tests, each excluding a different image from the training set. We model the response as either -1 for the "picture" state or 1 for the "sentence" state. This way, classification is based on the sign of the response. For each data item, the error is 0 if it is correctly classified and 1 otherwise. The ratio of incorrectly classified images and the mean and standard deviation of the total number of selected ROIs is reported for each subject. We do not show the number of correctly selected variables because it is not clear which are relevant beforehand. We tested the statistical significance of the best prediction performance and best number of selected ROIs with regard to the second best method. The significance level was set to 0.05.

As observed, lazy lasso accuracy is not on average better than lasso accuracy. The lazy lasso achieves a better accuracy for three subjects, whereas the lasso is better for the other three subjects. Furthermore, these differences are not statistically significant. However, both algorithms are slightly better than loess and RT, while rodeo obtains the best accuracies. The naïve lazy lasso accuracy is definitely poorer. Although not shown in Table 2.2, the difference of accuracy between the lazy lasso and the naïve lazy lasso is significant in four out of six cases (*04847*, *05675*, *05680* and *05710*).

The big difference between the lazy lasso and loess, lasso, RT and rodeo is the number of selected ROIs. The lazy lasso selects much fewer ROIs than the lasso, RT and rodeo. Loess does not perform variable selection at all. Hence, at the cost of an insignificant loss of accuracy compared to the lasso, the lazy lasso exhibits a finer variable selection ability. The naïve lazy lasso selects the lowest number of ROIs, but its accuracy is poor.

Summing up, the lazy lasso has also been proven to work well in real environments. It often outperforms the naïve lazy lasso, the lasso, loess and RT, especially in complex scenarios.

2.5 DISCUSSION

In this chapter, we propose an iterative lazy variable selection and shrinkage method that relies on a traditional locally weighted regression paradigm and L_1-regularization. This algorithm was presented in the paper (Vidaurre et al., 2011d). We prove the usefulness of the procedure on three synthetic scenarios, several data sets derived from the *Pumadyn* real data set and the *StarPlus* data set. The lazy lasso is particularly appealing for sparse data sets.

On the regularization side, we provide a method for dealing with nonlinear data. Although LARS can be extended to tackle nonlinear functions, higher-order terms have to be defined in advance. On the local regression side, we provide a variable selection functionality. Local regression is known to be less useful in high-dimensional settings, due to the curse of dimensionality. Bias and variance cannot be kept at low and reasonable levels, respectively, when the number of data items in the local neighborhood is small compared to the number of variables. By reducing the dimension, our approach makes local regression more applicable in these cases.

Our approach is lazy in the sense that there is no overall model valid for all future data items. Hence, as happens with locally weighted regression, we need to run the whole algorithm each time a new data item is presented. Flexibility for dealing with nonlinearity and better prediction and variable selection performance are the advantages gained in exchange for a more expensive computation when compared with non-lazy techniques. Although this procedure is lazy, if computation time is a primary concern, the analyst can somehow extrapolate the incoming data items to the closest data items in the data set, whose regression coefficients have already been estimated. If these data items are close enough, they are likely to share the same set of relevant variables.

The lazy lasso has potential applications in the context of functional data analysis, where predictors are points on the continuum (Ramsay and Silverman, 2005; Ferraty and Vieu, 2006; Ferraty et al., 2010). In this field, the objective is rather a global model that selects a set of highly predictive design points than a local estimation. As shown by Barrientos-Marin et al. (2010), however, the functional data estimation can be considerably benefited from some form of local analysis. To take advantage of this fact, for example, one could obtain a local model with the lazy lasso for each train data instance or for some selected subset. Then, those points (predictors) that have not been selected in any model, or have been selected in few models, could be discarded. Note that an adequate distance function had to be defined for dealing with functional data. To cope with the computational burden, we could apply some early stopping in the lazy lasso process and bound the number of selected variables at each step.

More future work will revolve around the adaptation of the algorithm to multi-response regression, the use of recent variations of the lasso and any improvements on the current algorithm. Robustness is also a major concern. There are robust versions that prevent the harmful effects of outliers for both loess (Cleveland, 1979) and the lasso (Khan et al., 2007). Methods that make the proposed algorithm more robust need to be investigated.

3

SPARSE BAYESIAN REGULARIZED LOCAL REGRESSION

3.1 INTRODUCTION

As in the previous chapter, we consider p independent covariates $\{X_1, ..., X_p\}$ and a response variable Y. Let $\boldsymbol{X}$ and $\boldsymbol{y} = (y_1, ..., y_N)$ be, respectively, an $N \times p$ data matrix and a continuous-valued vector, so that each row $\boldsymbol{x}_i$ is iid related to a continuous response y_i by means of some unknown (nonlinear) function $m(\cdot)$, which is assumed to be sparse and have continuous second order derivatives Again, the objective is to estimate the response at a point of interest $\boldsymbol{x} = (x_1, ..., x_p)$ for $m(\cdot)$ to be sparse. We denote as $\boldsymbol{X}^*$ the data matrix $\boldsymbol{X}$ centered at $\boldsymbol{x}$ and augmented with a first column of ones, so that $x^*_{i0} = 1$ for all $i = 1, ..., N$.

In this case, the homoscedasticity assumption is not strictly necessary, so that we can generically define the variance of ε_i as $\text{Var}[\varepsilon_i|\boldsymbol{x}_i] = s^2(\boldsymbol{x}_i) = \sigma_i^2$.

The neighborhood is defined by a kernel function, which computes weights $\boldsymbol{w} = (w_1, ..., w_N)$ on the grounds of the distance from $\boldsymbol{x}_i$ to $\boldsymbol{x}$. The kernel function has a bandwidth parameter, which strongly influences the estimation. High bandwidths increases the bias and decreases the variance of the estimation; low bandwidths do the opposite. The simplest approach is to use a bandwidth single value for all regressors, and the most general setting is to use a full matrix bandwidth, which allows flexible smoothing on all orientations. While the first usually leads to a severely biased estimation, the second can imply the estimation of a large number of parameters. A convenient compromise is a vector bandwidth or diagonal bandwidth, denoted as $\boldsymbol{h} = (h_1, ..., h_p)^t$, which permits adaptive smoothing at each coordinate direction. The kernel function is defined as

$$w_i^2 = K_{\boldsymbol{h}}(\boldsymbol{x}_i - \boldsymbol{x}) = \prod_{j=1}^{p} \frac{1}{h_j} K\left(\frac{x_{ij} - x_j}{h_j}\right), \quad i = 1, ..., N, \tag{3.1}$$

where $K(\cdot)$ is a univariate, symmetric and nonnegative function with a compact support, such that $\int K(t)dt = 1$. In this chapter, we use the well-known Gaussian kernel, defined as $K(t) = (2/\pi)^{1/2} \exp(-t^2/2)$.

Then, the estimated local linear regression function $g(\cdot)$ is defined by a vector of local regression coefficients $\hat{\boldsymbol{\beta}}(\boldsymbol{x}) = (\hat{\beta}_1(\boldsymbol{x}), ..., \hat{\beta}_p(\boldsymbol{x}))^t$ and an intercept term $\hat{\beta}_0(\boldsymbol{x})$. For simplicity of notation, in the following we denote, unless necessary, $\hat{\beta}_0(\boldsymbol{x})$ as $\hat{\beta}_0$ and $\hat{\boldsymbol{\beta}}(\boldsymbol{x})$ as $\hat{\boldsymbol{\beta}}$. Then, we have

Therefore, the cornerstone of (linear) local regression is the estimation of a suitable bandwidth. We focus on the multivariate case, where a direct estimation is not straightforward. This estimation implies unknown functionals which themselves depend on the bandwidth. Typically, the bandwidth is set by direct (plugin) computation (Wand and Jones, 1994; Yang and Tschernig, 1999), selected by cross-validation (Sain

et al., 1994; Hall et al., 2007) or found within some type of suboptimal search (Lafferty and Wasserman, 2008). It is known that a suitable plugin estimation of $\boldsymbol{h}$ can improve the cross-validated estimation. In this chapter, we work on the basis of plugin diagonal bandwidth estimations, which, as mentioned above, is a reasonable tradeoff between a scalar bandwidth and a full matrix bandwidth.

We state that the kernel estimation in the local regression framework should account for the importance of each variable. In other words, if a variable is absolutely irrelevant for the regression function, or noisy, it should not participate in the weights calculation (Vidaurre et al., 2011d). Since the best rate of convergence in nonparametric regression is $N^{-4/(4+p)}$ (Györfi et al., 2002), to exploit the sparse nature of $m(\cdot)$ is extremely convenient.

The method that we present in this dissertation and in the paper (Vidaurre et al., 2012) is related to the *rodeo* approximation (Lafferty and Wasserman, 2008) because both approaches consider sparsity in $m(\cdot)$. Lafferty and Wasserman (2008) use the estimated gradient of the regression function with respect to the bandwidth, $\partial m(\boldsymbol{x})/\partial \boldsymbol{h}$, to conduct a greedy search, considering that a high value of $\partial m(\boldsymbol{x})/\partial h_j$ is indicative of the relevance of variable X_j. Favouring computational speed and applicability in high-dimensional settings, *rodeo* does not generalize for arbitrary nonlinearities. This method assumes a known value of σ^2. If σ^2 is unknown, it has to be separately estimated.

In this chapter, we take a adaptive regularized multivariate local regression approach by defining appropriate distributions over the parameters, combining it with an efficient bandwidth estimation method. We call it *sparse regularized local regression*, or sRLR. The method includes the estimation of σ^2 and considers sparsity by analyzing the estimated bandwidths at each step. Several elements of the method are analogous to other approaches (which do not consider sparsity explicitly), as for example the work by Yang and Tschernig (1999). We expect that a suitable application of adaptive ridge regularization will further improve the bias-variance tradeoff of the estimation. In order to enjoy the advantages of the Bayesian paradigm, we also propose a Bayesian estimation of the regression coefficients, where, using sampling methods, we obtain an estimate of the posterior distribution of the response.

The rest of the chapter is organized as follows. First, we introduce the Bayesian hierarchy. Then, we describe the bandwidth selection procedure. After, we detail the algorithm for producing a maximum a posteriori (MAP) estimation of the response and the parameters. Then, we set out a more Bayesian approach. The next two subsections provide, respectively, theoretical and empirical justificacion of the methods. Finally, we draw some conclusions.

3.2 HIERARCHICAL MODEL

We first define the distribution of the response,

$$y|\boldsymbol{\beta}^*, \sigma^2 \sim \mathcal{N}(\beta_0, \sigma^2), \tag{3.2}$$

with, in principle, no distributional assumption on the form of the noise. We show below, nevertheless, how the homoscedasticity assumption makes the algorithm applicable in high-dimensions.

Next, we define a non-isotropic Gaussian prior distribution over $\boldsymbol{\beta}$:

$$\boldsymbol{\beta}|\boldsymbol{\alpha}^2 \sim \mathcal{N}(0, \boldsymbol{\Sigma}), \tag{3.3}$$

with parameter $\boldsymbol{\Sigma} = \text{diag}(\boldsymbol{\alpha}^2)$ and $\alpha_j^2 > 0$, $j = 1, ..., p$. We choose a Gaussian prior for being the conjugate of the Gaussian density, so that the problem is analytically tractable. From the frequentist perspective, this is equivalent to impose an adaptive L_2-penalty on the regression coefficients, the regularization parameters playing the role of the diagonal matrix $\sigma^2\boldsymbol{\Sigma}^{-1}$. Then, $\boldsymbol{\alpha}^2$ adaptively tunes the regularization for each parameter β_j.

Considering noninformative improper prior densities over σ^2 and α_j^2, given respectively by $1/\sigma^2$ and $1/\alpha_j^2$, along with a noninformative prior over β_0, we complete the hierarchical representation of the model.

3.3 BANDWIDTH SELECTION

In this section, we assume that σ^2 and $\boldsymbol{\alpha}^2$ are known quantities. The optimal estimation of $\boldsymbol{h}$ is given by the minimization of the integrated squared error (MISE) statistic

$$MISE(\boldsymbol{h}) = E\left[\int (m(\mathbf{x}) - g(\mathbf{x}))^2 f_X(\mathbf{x}) z(\mathbf{x}) d\mathbf{x}\right], \tag{3.4}$$

where $z(\cdot)$ is some weighting function and $f_X(\cdot)$ is the design density function. The weighting function is provided to allow the design density $f_X(\cdot)$ to be not compactly supported. In practice, $z(\cdot)$ can be for example an indicator of the support of $m(\cdot)$ or an indicator of some neighborhood of the point of interest $\mathbf{x}$).

Let us define the Hessian matrix $\boldsymbol{\Lambda}(\mathbf{x})$ with elements

$$\lambda_{jj'}(\mathbf{x}) = \frac{\partial^2 m(\mathbf{x})}{\partial X_j \, \partial X_{j'}}. \tag{3.5}$$

An asymptotic approximation of the MISE-optimal bandwidth $\hat{\boldsymbol{h}}$ is defined for example in (Yang and Tschernig, 1999). Particularizing for the Gaussian kernel, we have

$$\hat{\boldsymbol{h}} = \left(\frac{||K||_2^{2p}\psi(s)}{N}\right) \exp\left(\frac{\phi(\boldsymbol{C}(m))}{2}\right), \tag{3.6}$$

where $||K||_2 = \int K^2(t)dt$ and $\varphi(K) = \int t^2 K(t)dt$. If the support is infinite, then $||K||_2 = 1/\sqrt{\pi}$ and $\varphi(K) = 1$.

Under the homoscedasticity assumption, the functional $\psi(s)$ is defined as

$$\psi(s) = \int s^2(\mathbf{x}) z(\mathbf{x}) d\mathbf{x} = \sigma^2 \int z(\mathbf{x}) d\mathbf{x} \tag{3.7}$$

The elements of the non-negative definite matrix $\boldsymbol{C}(m) \in \mathbb{R}^{p \times p}$ are defined as

$$c_{jj'}(m) = \int \lambda_{jj}(\mathbf{x}) \lambda_{j'j'}(\mathbf{x}) \, f_X(\mathbf{x}) z(\mathbf{x}) d\mathbf{x} \simeq \frac{1}{N} \sum_{i=1}^{N} \lambda_{jj}(\boldsymbol{x}_i) \lambda_{j'j'}(\boldsymbol{x}_i).$$

where the approximation assumes that $z(\cdot)$ is an indicator of the support of $m(\cdot)$. Note that, since our bandwidth estimate is diagonal, we only need the diagonal elements of $\boldsymbol{\Lambda}(\mathbf{x})$. We define $\phi(\boldsymbol{M})$ as the solution of the unconstrained optimization problem

$$\phi(\boldsymbol{M}) = \text{argmin}_v \quad \frac{1}{4}\exp(\boldsymbol{v})^t \boldsymbol{M} \exp(\boldsymbol{v}) + \exp\left(\sum_{j=1}^{p} \frac{-v_j}{2}\right),$$

which can be proved to be convex because both terms are positive definite, and, thus, the Hessian with respect to v is positive definite. Then, this problem can be easily solved for example by a standard application of the Newton-Raphson algorithm.

Finally, a kernel density estimation of $f_X(\boldsymbol{x}_i)$ can be obtained as

$$f_X(\boldsymbol{x}_i) = \frac{1}{N}\sum_{i'=1}^{N} K_{\boldsymbol{h}}(\boldsymbol{x}_{i'} - \boldsymbol{x}_i).$$

Yang and Tschernig (1999) proposed to find an approximation of $\boldsymbol{\Lambda}(\boldsymbol{x}_i)$ by performing, for each variable, a local cubic estimation with several cross-terms left out. For $p = 10$ variables, for example, this estimation needs 40 parameters. This approximation requires itself a bandwidth, which is taken to be scalar and also depends on unknown functionals that need to be estimated as well.

In this paper, we take a simpler approach. Like Yang and Tschernig (1999), we use a local cubic estimation leaving many cross-terms out. We construct the matrix

$$\begin{aligned}\tilde{\boldsymbol{X}}_{(i,j)} = \Big[&\mathbf{1}, \{(\boldsymbol{X}_{\cdot j'} - x_{ij'})^2\}_{j'=1}^{p}, \{(\boldsymbol{X}_{\cdot j} - x_{ij})(\boldsymbol{X}_{\cdot j'} - x_{ij'})\}_{j'\neq j}, (\boldsymbol{X}_{\cdot j} - x_{ij}),\\ &\{(\boldsymbol{X}_{\cdot j} - x_{ij})(\boldsymbol{X}_{\cdot j'} - x_{ij'})^2\}_{j'\neq j}\{(\boldsymbol{X}_{\cdot j} - x_{ij})^2(\boldsymbol{X}_{\cdot j'} - x_{ij'})\}_{j'\neq j}(\boldsymbol{X}_{\cdot j} - x_{ij})^3\Big],\end{aligned}$$

where $\mathbf{1}$ is a vector of length p with all elements equal to 1. Then, we compute the regression

$$\gamma_{(i,j)} = 2\big(\tilde{\boldsymbol{X}}'_{(i,j)}\boldsymbol{W}\tilde{\boldsymbol{X}}_{(i,j)}\big)^{-1}\tilde{\boldsymbol{X}}'_{(i,j)}\boldsymbol{W}\boldsymbol{y},$$

where the diagonal matrix $\boldsymbol{W}$ is computed with the current estimated bandwidth $\hat{\boldsymbol{h}}$ for $\mathbf{x}$. The estimate of $\lambda_{jj}(\boldsymbol{x}_i)$ is then given by the $(1+j)$-th element of $\gamma_{(i,j)}$.

Note that this Hessian estimate is suboptimal because we apply the current estimated bandwidth for $\mathbf{x}$ to all data points involved in Equation (3.5), and the curvature of $m(\cdot)$ at these points can differ from that at $\mathbf{x}$. However, since we seek a local estimate of the bandwidth, we can take $z(\boldsymbol{x}_i) = 1$ if $\boldsymbol{x}_i$ is within some neighborhood of $\mathbf{x}$ and $z(\boldsymbol{x}_i) = 0$ otherwise. Then, we only need to estimate the Hessian in this neighborhood and, assuming that $m(\cdot)$ is not very wiggly, the current estimate $\hat{\boldsymbol{h}}$ is reasonable for this purpose.

Sparsity is considered here in the sense of (Lafferty and Wasserman, 2008): to discard a variable X_j amounts for using a sufficiently high bandwidth h_j. Then, its contribution for the calculation of $\boldsymbol{w^2}$ becomes negligible. For X_j to be dropped, we can use the criterion

$$\frac{K(0) - K(\kappa_j/h_j)}{K(0)} < \epsilon, \tag{3.8}$$

where $\kappa_j = \max_{i=1}^{N} |\mathrm{x_j} - \mathrm{x_{ij}}|$ and $\epsilon > 0$ is some small constant. From Equation (3.8), we can derive the following threshold

$$\tau_j = \sqrt{-\frac{\kappa_j}{2\log(1-\epsilon)}},$$

so that, if $\hat{h}_j \geq \tau_j$, then X_j can be discarded. We denote as Ω the set of variables under this threshold.

Figure 3.1, left, illustrates the value $K(\kappa_j/h_j)/h_j$ for several $h_j = 0.1, ..., 4.0$ and $|\mathrm{x_j} - \mathrm{x_{ij}}|$ within the unit support, so that each line represents a different h_j. The value

$K(\kappa_j/h_j)$ has been normalized so that $\int_0^1 K(t)dt = 1$. The thick red line corresponds to $h_j = 4.0$. Figure 3.1, right, shows the left side of Equation (3.8) for $h_j = 0.1, ..., 4.0$. $\epsilon = 0.1$ is represented by the dashed line. For $\epsilon = 0.1$, we have $\tau_j = 2.178$.

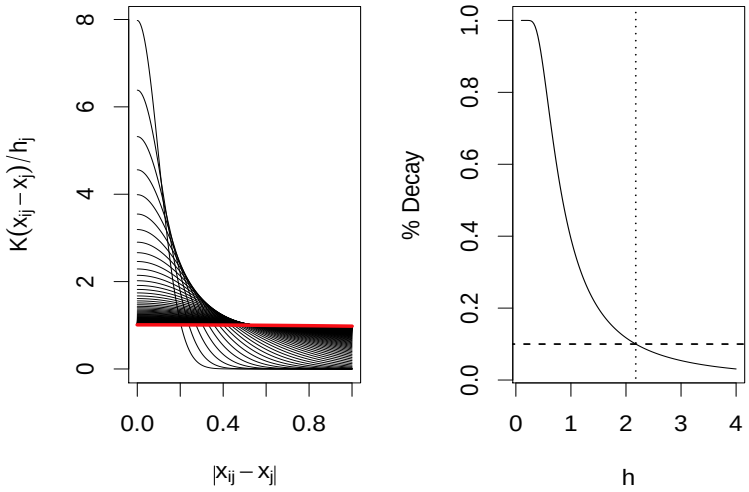


Figure 3.1: Left, values $K(\kappa_j/h_j)/h_j$ for $h_j = 0.1, ..., 4.0$ and $\kappa_j = 1.0$. Right, left side of Equation (3.8) for $h_j = 0.1, ..., 4.0$; $\epsilon = 0.1$ is represented by the dashed horizontal line.

3.4 MAP ESTIMATION

Let us assume by now that we have an estimation of the optimal diagonal bandwidth $\hat{\boldsymbol{h}}$, and, hence, we have weight values w_i, $i = 1, ..., N$. Then, we can obtain the parameters $\boldsymbol{\beta}^*$, σ^2 and $\boldsymbol{\alpha}^2$ that maximize the expected MAP function by means of the expectation-maximization (EM) algorithm (Dempster et al., 1977). We consider $\boldsymbol{\beta}^*$ to be the latent variable.

From distributions (3.2) and (3.3), the expectation of the complete-data log likelihood function is

$$\begin{aligned} \mathrm{E}[\log p(\boldsymbol{\beta}^*, \boldsymbol{y}|\boldsymbol{\alpha}^2, \sigma^2)] = \mathrm{E}[\log p(\boldsymbol{y}|\boldsymbol{\beta}^*, \sigma^2)] + \mathrm{E}[\log p(\boldsymbol{\beta}|\boldsymbol{\alpha}^2)] = \\ -\frac{N_w}{2}\log(2\pi\sigma^2) - \frac{\mathrm{E}[(y-\hat{y})^2]}{2\sigma^2} \\ -\frac{p}{2}\log(2\pi) - \frac{1}{2}\log\det(\boldsymbol{\Sigma}) - \frac{\mathrm{E}[\boldsymbol{\beta}^t\boldsymbol{\Sigma}^{-1}\boldsymbol{\beta}]}{2}, \end{aligned} \tag{3.9}$$

where $N_w = \sum_{i=1}^N w_i^2$, and we approximate

$$\mathrm{E}[(y-\hat{y})^2] \simeq \sum_{i=1}^N (w_i y_i - w_i\beta_0 - w_i \boldsymbol{x}_i^t\boldsymbol{\beta})^2.$$

In the E-step, we obtain the values of $\boldsymbol{\beta}^*$ that maximize the expectation (3.9). Setting the derivatives with respect to $\boldsymbol{\beta}^*$, for some σ^2 and $\boldsymbol{\alpha}^2$, we obtain the parameter estimation at this step as

$$\hat{\boldsymbol{\beta}}^* = \left(\sigma^2\boldsymbol{\Sigma}^{-1} + \boldsymbol{X}^{*^t}\boldsymbol{W}\boldsymbol{X}^*\right)^{-1}\boldsymbol{X}^{*^t}\boldsymbol{W}\boldsymbol{y}, \tag{3.10}$$

Algorithm 2 Iterative algorithm for regularized local MAP estimation

Initialize $\boldsymbol{h} = h_0$, $\sigma^2 = \sigma_0^2$, $\boldsymbol{\alpha}^2 = \boldsymbol{\infty}$.
Repeat until convergence:
  Compute $\hat{\boldsymbol{h}}$ as described in Section 3.3.
  Compute $\boldsymbol{w}$ with Equation (3.1).
  Repeat until convergence:
    Update $\hat{\boldsymbol{\beta}}^*$ with Equation (3.10).
    Update σ^2 and $\boldsymbol{\alpha}^2$ with, respectively, Equations (3.11) and (3.15).

with $\boldsymbol{W} = \text{diag}(\boldsymbol{w}^2)$.

In the M-step, we estimate the values of σ^2 and $\boldsymbol{\alpha}^2$ that maximize Equation (3.9) for the current value of $\boldsymbol{\beta}^*$. We set the derivatives of Equation (3.9) with respect to σ^2 to zero to obtain

$$\hat{\sigma}^2 = \frac{\text{E}[\,(y - \hat{y})^2]}{N_{\boldsymbol{w}}} \simeq \frac{\sum_{i=1}^{N}(w_i y_i - w_i \beta_0 - w_i \boldsymbol{x_i}^t \hat{\boldsymbol{\beta}})^2}{N_{\boldsymbol{w}}}. \tag{3.11}$$

To estimate $\boldsymbol{\alpha}^2$, we first have

$$\frac{\partial \log\det(\boldsymbol{\Sigma})}{\partial \alpha_j^2} = \frac{\partial \log(\prod_{j'=1}^{p} \alpha_{j'}^2)}{\partial \alpha_j^2} = \frac{1}{\alpha_j^2}. \tag{3.12}$$

On the other hand, using the definition of variance, we have

$$\frac{\partial \,\text{E}[\boldsymbol{\beta}^t \boldsymbol{\Sigma}^{-1} \boldsymbol{\beta}]}{\partial \alpha_j^2} = \frac{\partial \,\text{E}[\Sigma_{j'=1}^{p} \beta_{j'}^2 / \alpha_{j'}^2]}{\partial \alpha_j^2} = \frac{\partial \sum_{j'=1}^{p} \text{E}[\beta_{j'}^2]/\alpha_{j'}^2}{\partial \alpha_j^2} = -\frac{\hat{\beta}_j^2 + \hat{s}_{j+1,j+1}}{\alpha_j^{2^2}}, \tag{3.13}$$

where $\hat{\boldsymbol{S}}$ is the estimated covariance matrix of the posterior distribution of $\boldsymbol{\beta}$

$$\hat{\boldsymbol{S}} = \left(\text{diag}^{-1}(0, \boldsymbol{\alpha}^2) + \frac{1}{\hat{\sigma}^2} \boldsymbol{X}^{*^t} \boldsymbol{W} \boldsymbol{X}^*\right)^{-1}, \tag{3.14}$$

$\hat{s}_{j+1,j+1}$ is the $(j+1)$-th element of the diagonal of $\hat{\boldsymbol{S}}$ and $\hat{\boldsymbol{\beta}}$ is computed with Equation (3.10). Equation (3.14) is a standard result that arises from the convolution of two Gaussian distributions, given by the prior, that corresponds to the first term within the parenthesis in Equation (3.14), and the likelihood function, that corresponds to the second term.

Putting results (3.12) and (3.13) together and reorganizing terms, we can estimate α_j^2 as

$$\hat{\alpha}_j^2 = \hat{\beta}_j^2 + \hat{s}_{j+1\,j+1}. \tag{3.15}$$

Note that the estimation of $\boldsymbol{\beta}$ depends on σ^2, whose estimation depends itself on $\boldsymbol{\beta}$. The same happens with $\boldsymbol{\alpha}^2$, whose estimation, besides, depends recursively on itself through $\boldsymbol{S}$. Iterating the estimation of $\hat{\boldsymbol{\beta}}$ with Equation (3.10) and the estimation of σ^2 and $\boldsymbol{\alpha}^2$ with Equations (3.11) and (3.15), and repeating until convergence, the EM algorithm is able to find a MAP solution for a fixed $\boldsymbol{w}$ in a finite number of steps. We summarize the MAP estimation in Algorithm 2.

We can make use of sparsity as defined above by restricting the above computations to those variables that are currently included in Ω. Note also that we are handling

sparsity at two levels. First, sparsity of the function $m(\cdot)$ is determined by the magnitude of the bandwidths. Second, sparsity of $\hat{\boldsymbol{\beta}}^*$ is handled by automatic relevance determination. The second type of sparsity is just a particular case of the first, and, then, we cannot use it to simplify the estimation of $\boldsymbol{h}$.

3.5 ESTIMATION BY MONTE CARLO SAMPLING

In this section, we evolve from the MAP approach to a sampling method for determining the posterior distribution of $\boldsymbol{\beta}^*$, σ^2 and $\boldsymbol{\alpha}^2$, and then the predictive distribution.

3.5.1 *Parameter distribution*

First, we estimate the posterior parameter distribution of $\boldsymbol{\beta}^*$, defined by the sufficient statistics $\mathrm{E}[\boldsymbol{\beta}^*] = \hat{\boldsymbol{\beta}}^*$ and $\mathrm{Var}[\boldsymbol{\beta}^*] = \hat{\mathbf{S}}$. Instead of sampling from the joint posterior of $\boldsymbol{h}$, $\boldsymbol{\beta}^*$, σ^2 and $\boldsymbol{\alpha}^2$, we sample from the complete-data parameter posterior of σ^2 and $\boldsymbol{\alpha}^2$ given the current estimates of $\boldsymbol{\beta}^*$ and $\boldsymbol{h}$. We alternate between two steps.

first step, given the current posterior density estimate, the objective is to obtain L samples $\Theta_l \equiv \{{\sigma^2}^{(l)}, {\boldsymbol{\alpha}^2}^{(l)}\}$.

First, from the current estimate of $\boldsymbol{h}$, we compute $\boldsymbol{w}$ by Equation (3.1) and $N_{\boldsymbol{w}} = \sum_{i=1}^{N} w_i^2$. We shall now be able to sample from $p(\sigma^2, \boldsymbol{\alpha}^2 | \boldsymbol{y})$ by following the hierarchy defined above. For each l, we can sample $1/{\sigma^2}^{(l)}$ from a Gamma distribution (which maintains the conjugacy of the hierarchy) with parameters

$$a_{\sigma^2} = \frac{N_{\boldsymbol{w}}}{2} + 1 \quad \text{and} \quad b_{\sigma^2} = \frac{1}{2}\sum_{i=1}^{N}(w_i y_i - w_i \hat{\beta}_0 - w_i \boldsymbol{x_i}^t \hat{\boldsymbol{\beta}})^2, \tag{3.16}$$

where $\hat{\boldsymbol{\beta}}^*$ is the current MAP estimate of $\boldsymbol{\beta}^*$ obtained from Equation (3.10). Since the mode of the Gamma distribution is given by $(a_{\sigma^2} - 1)/b_{\sigma^2}$, this is consistent with the result in Equation (3.11). Also, we sample $1/{\alpha_j^2}^{(l)}$, for $j = 1, ..., p$, from a Gamma distribution with parameters

$$a_{\alpha_j^2} = \frac{3}{2} \quad \text{and} \quad b_{\alpha_j^2} = \frac{1}{2}(\hat{\beta}_j^2 + \hat{s}_{j+1\,j+1}), \tag{3.17}$$

where $\hat{s}_{j+1,j+1}$ is the $(j+1)$-th element of the diagonal of $\hat{\mathbf{S}}$.

Equations (3.16) and (3.17) were obtained by "completing the square" on the product of the logarithm of the noninformative prior density (i.e., a Gamma distribution with $a = b = 0$) and the log likelihood given by Equation (3.9).

Finally, for each sample Θ_l, we estimate the optimal bandwidth, $\hat{\boldsymbol{h}}^{(l)}$, as described in Section 3.3 and compute $\boldsymbol{w}^{(l)}$ by Equation (3.1).

In the second step, we update the posterior distribution of the parameters using the $\boldsymbol{w}^{(l)}$ samples. For each vector $\boldsymbol{w}^{(l)}$, we iteratively estimate ${\hat{\boldsymbol{\beta}}^*}^{(l)}$ and $\hat{\mathbf{S}}^{(l)}$ with Equations (3.10) and (3.14), and update ${\hat{\sigma}^2}^{(l)}$ and ${\hat{\boldsymbol{\alpha}}^2}^{(l)}$ with Equations (3.11) and (3.15) until convergence.

Once we have the posterior estimate of the parameters for each sample Θ_l, we round the P step by performing the following update

$$\hat{\boldsymbol{\beta}}^* = \frac{1}{L}\sum_{l=1}^{L} {\hat{\boldsymbol{\beta}}^*}^{(l)} \quad \text{and} \quad \hat{\mathbf{S}}^{-1} = \frac{1}{L}\sum_{l=1}^{L} {\hat{\mathbf{S}}^{(l)}}^{-1}. \tag{3.18}$$

Algorithm 3 Sampling algorithm for finding the posterior distribution of $\boldsymbol{\beta}^*$, σ^2 and $\boldsymbol{\alpha}^2$

Initialize the posterior distribution of $\boldsymbol{\beta}^*$ and $\hat{\boldsymbol{h}}$. Obtain $\boldsymbol{w}$.

Step (i):
Draw L samples Θ_l given that:
σ^2 is Gamma distributed with parameters given by Equation (3.16);
$\boldsymbol{\alpha}^2$ are Gamma distributed with parameters given by Equation (3.17).
For each Θ_l, estimate $\hat{\boldsymbol{h}}^{(l)}$ as described in Section 3.3.
For each $\hat{\boldsymbol{h}}^{(l)}$, compute $\boldsymbol{w}^{(l)}$ with Equation (3.1).

Step (ii):
For each l, using $\boldsymbol{w}^{(l)}$, iterate until convergence:
Compute $\hat{\boldsymbol{\beta}}^{(l)}$ and $\hat{\mathbf{S}}^{(l)}$ with Equations (3.10) and (3.14);
Update $\hat{\sigma}^{2^{(l)}}$ and $\hat{\boldsymbol{\alpha}}^{2^{(l)}}$ with (3.11) and (3.15).
Update the posterior distribution of $\hat{\boldsymbol{\beta}}^{*^{(l)}}$ and $\hat{\mathbf{S}}^{(l)}$ with Equation (3.18).
Update the posterior distribution of σ^2 and $\boldsymbol{\alpha}^2$ with (3.16) and (3.17).
Compute $\hat{\boldsymbol{h}}$ as described in Section 3.3.
Compute $\boldsymbol{w}$ with Equation (3.1).

Repeat steps (i) and (ii) until the posterior distribution of $\boldsymbol{\beta}$ stabilizes.

Since it is not analytically simple to characterize the sum of Gamma densities with different scales as a Gamma density itself, it is not straightforward to formulate an estimate of the posterior distribution of σ^2 and $\boldsymbol{\alpha}^2$ as a function of the samples $\sigma^{2^{(l)}}$ and $\boldsymbol{\alpha}^{2^{(l)}}$. However, we can still approximate these distributions, via $\hat{\boldsymbol{\beta}}^*$ and $\hat{\mathbf{S}}^{-1}$, with Equations (3.16) and (3.17). Then, using the expectation of $\sigma^{2^{(l)}}$, we can compute the optimal bandwidth $\hat{\boldsymbol{h}}$ as described in Section 3.3.

We summarize the method in Algorithm 3.

3.5.2 *Predictive distribution*

Next, we formulate the predictive distribution

$$p(y|\boldsymbol{y}) = \int p(y|\boldsymbol{\beta}^*, \sigma^2, \boldsymbol{y})\, d\boldsymbol{\beta}^* d\sigma^2,$$

which can be shown to be Student,

$$y|\boldsymbol{y} \sim \mathrm{St}\big(\mu = \hat{\beta}^0, \iota = \hat{\sigma}^2 + (1, \boldsymbol{x})^t \hat{\mathbf{S}}(1, \boldsymbol{x}), \nu = N - p\big),$$

where μ is the mean, ι is the precision and ν is the degrees of freedom of the distribution.

3.6 ASYMPTOTIC PROPERTIES

It is well-known that local regression have high asymptotic efficiency, that can be maximal for an adequate choice of bandwidth and kernel function. The existence of a unique MISE-optimal vector bandwidth, estimated by Equation (3.6), was shown by Yang and Tschernig (1999) under the following assumptions:

- The design density $f_X(\cdot)$ is continuously differentiable up to the second order and is bounded away from zero in the support of $m(\cdot)$.
- The regression function $m(\cdot)$ is continuously differentiable up to order 4.
- The optimal bandwidth is positive and depends on N, such that, for the relevant variables, $h_j \to 0$ and $Nh_j^{p+4} \to \infty$.
- The matrix $\boldsymbol{C}(m)$ is non-negative definite, and is positive definite for all non-negative vectors.

Then, assuming that the estimation of the Hessian is asymptotically optimal, and assuming homoscedasticity, if the estimations of σ^2 and $\boldsymbol{\alpha}^2$ are optimal, the estimated bandwidth is asymptotically MISE-optimal according to the results of Yang and Tschernig (1999).

3.7 EXPERIMENTS

In order to show the performance of the proposed approach for finite samples, we have run Algorithm 2 on 100 data sets generated from eight different models. In all cases, there are $N = 500$ samples and $p = 10$ covariates, sampled from the uniform distribution on $[0, 1]$. We have set $\sigma^2 = 0.1$. The test point is $\boldsymbol{x} = (0.5, ..., 0.5)$. Table 3.1 shows the regression function $m(\cdot)$ for each model.

Table 3.1: Regression function for each model.

Model 1	$m(\mathbf{x}) = \mathrm{x}_1^2 + \mathrm{x}_2^2$
Model 2	$m(\mathbf{x}) = \sin(3\mathrm{x}_1 + 3\mathrm{x}_2)$
Model 3	$m(\mathbf{x}) = \sin(3\mathrm{x}_1)\sin(6\mathrm{x}_2)$
Model 4	$m(\mathbf{x}) = 0.5\sin(3\mathrm{x}_1) + 0.5\sin(12\mathrm{x}_2)$
Model 5	$m(\mathbf{x}) = \mathrm{x}_1^2\mathrm{x}_2^2$
Model 6	$m(\mathbf{x}) = 2(\mathrm{x}_1 + 1)^3 + 2\sin(10\mathrm{x}_2)$

The four former regression functions were taken from the experimental section of (Yang and Tschernig, 1999) and the other two were taken from the work of Lafferty and Wasserman (2008).

In addition to the sRLR methodology, we have tested *rodeo* and the estimated optimal bandwidth selector from (Yang and Tschernig, 1999) (OBS for short). By including OBS in the experiments set, we intend to demonstrate the need to account for sparsity if $m(\cdot)$ is indeed sparse. We have used $\gamma_N = 0.05$ and $\kappa = 5$.

Table 3.2 reports the mean absolute error (and standard deviation) along the 100 data sets for sRLR (MAP estimation) and the other methods. Statistical significance is checked by means of the t-test. The performance of the proposed approach is better (with statistical significance) than OBS in four out of six experiments, and always significantly better than *rodeo*. Besides, unlike the others, it does not exhibit the worst accuracy in any of the experiments.

We conjecture that this accuracy advantage over the other methods is because, for kernel design purposes, to directly discard the irrelevant covariates is more effective than to assign high bandwidths to them. Table 3.3 shows the number of times that

Table 3.2: Mean (and standard deviation) of the absolute error for each model and method. The best result for each row is highlighted in bold. The symbol $*$ is added when the difference to the second best method is statistically significant with a significance level of 0.005.

Model	sRLR	*rodeo*	OBS
Model 1	**0.0200**($\pm$0.0127)$*$	0.0700($\pm$0.0525)	0.1342($\pm$0.0318)
Model 2	0.0185($\pm$0.0121)	0.0406($\pm$0.0486)	**0.0178**($\pm$0.0071)
Model 3	0.0311($\pm$0.0228)	0.0698($\pm$0.0578)	**0.0180**($\pm$0.0133)$*$
Model 4	**0.0428**($\pm$0.0353)$*$	0.1382($\pm$0.0972)	0.1228($\pm$0.0437)
Model 5	**0.0081**($\pm$0.0056)$*$	0.0384($\pm$0.0298)	0.0379($\pm$0.0084)
Model 6	**0.2361**($\pm$0.0875)$*$	1.0634($\pm$0.2996)	1.8556($\pm$0.3296)

Table 3.3: Number of times that each variable has been selected by sRLR across the 100 experiment replications.

Model	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
Model 1	100	100	0	1	0	0	0	1	0	0
Model 2	100	100	0	0	0	1	0	0	0	0
Model 3	54	100	13	10	10	11	6	8	13	7
Model 4	94	57	3	9	9	7	9	10	10	7
Model 5	100	100	0	0	0	0	0	0	0	0
Model 6	100	100	0	4	0	0	0	0	0	0

each variable has been selected by sRLR across the 100 experiment replications. Note that for Models 1, 2, 5 and 6, it selects the correct covariates, and only them. The superior performance of OBS for Model 3 is probably due to the fact that one of the relevant covariates (X_1) has been sometimes missed. For Model 4, even when the selection of covariates is not perfect, sRLR exhibits lower errors than OBS.

Figure 3.2 shows boxplots of the bandwidth vector for each model and method. We can observe that both *rodeo* and OBS tend to assign higher bandwidths to the irrelevant covariates. This is more marked for OBS. In Models 3 and 4, where sRLR occasionally selects some irrelevant covariates, the bandwidths computed by our approach are also higher than those of the relevant covariates. Note that, in general, the bandwidths estimated by OBS are higher than the ones computed by sRLR and *rodeo*. This is probably because OBS handles the whole set of covariates instead of a subset of relevant covariates.

To gain more insight into the full Bayesian approach, we have run Algorithm 3 on data generated from the models described above, using different values for the variance of ε_i: $\sigma^2 \in \{0.1, 0.2, 0.4\}$. Also, to introduce more uncertainty, we add some Gaussian noise to the relevant covariates (after computing the response $\boldsymbol{y}$ with the regression functions in Table 3.1), and then we redo the experiments. Specifically, we make $x_{ij} := x_{ij} + \varrho_{ij}$, where j corresponds to a relevant variable, $i = 1, ..., N$, and $\varrho_{ij} \sim \mathcal{N}(0, 0.2)$.

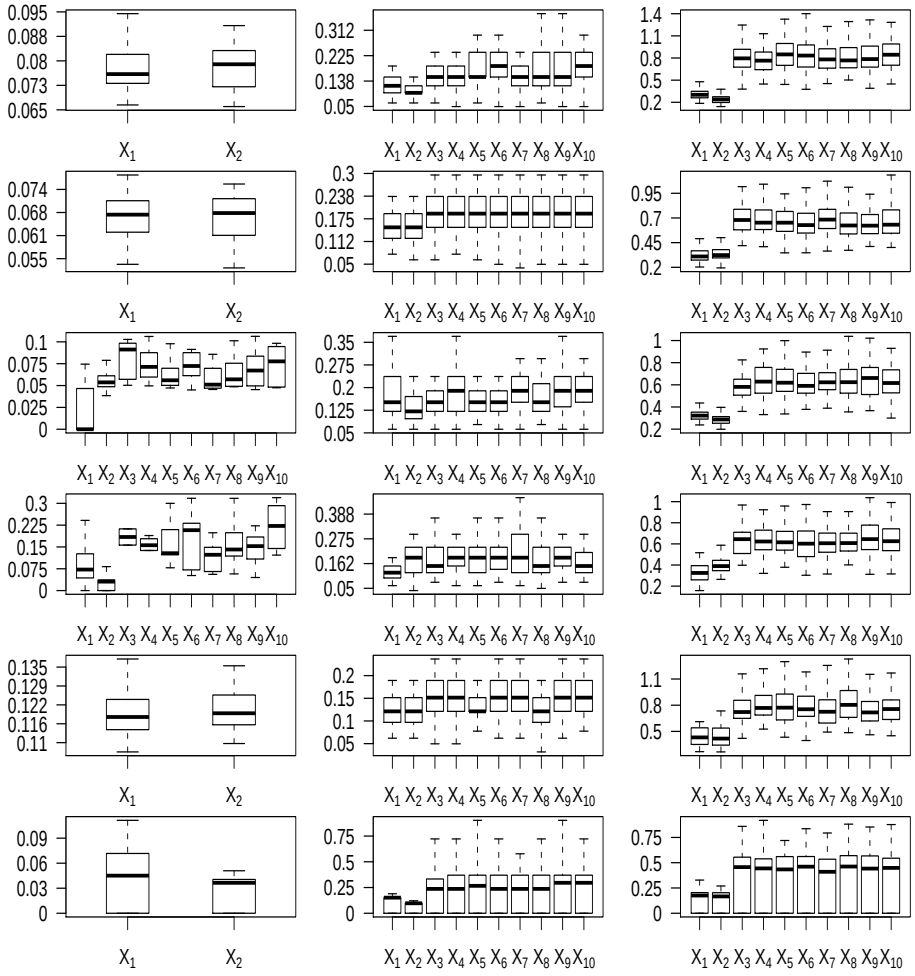


Figure 3.2: Boxplots of the bandwidth vector for each model and method. From top to bottom, Models 1-6 are illustrated. Left graphs correspond to sRLR, middle graphs correspond to *rodeo* and right graphs correspond to OBS. For sRLR, the bandwidths of all covariates are shown only for Models 3 and 4.

To give an example, Figure 3.3 shows the predictive distribution for some execution and Model 1. As expected, the more uncertainty the model conveys, the higher the variance of the response distribution is. This applies both for σ^2 and noise in the covariates.

3.8 DISCUSSION

In this chapter, we have shown how to give a Bayesian perspective to local regularized regression, describing both a MAP estimation procedure and a full Bayesian treatment. Besides, we have devised a bandwidth selection approach adequate for high-dimensional, sparse regression functions, which is integrated with the estimation of the regression parameters. Such sparse bandwidth selection procedure is based on optimal bandwidth selection methodology. We have empirically demonstrated the performance of the method and described its asymptotic behavior.

It is worth to point out that sRLR relies on some assumptions that can be removed in exchange of computational cost. For example, we assume homoscedasticity. This is to avoid the need of estimating σ_i^2 at each point. Besides, we are implicitly assuming

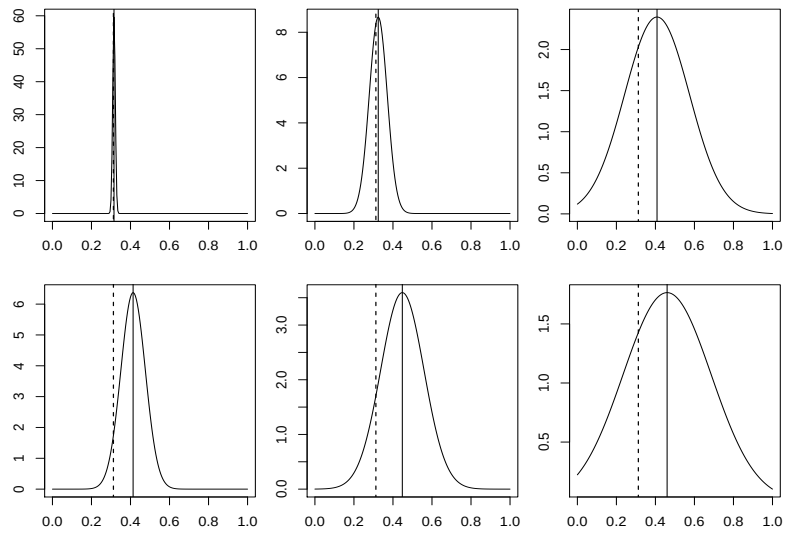


Figure 3.3: Estimated distribution of the responses for Model 1, and $\sigma^2 = 0.1$ (left), $\sigma^2 = 0.2$ (middle) and $\sigma^2 = 0.4$ (right). Top graphs were generated from data with non-noisy covariates and bottom graphs were generated from data with noisy covariates. The vertical solid line indicates the mode of the distribution and the vertical dotted line indicates the true response.

that the estimated optimal regularization parameters $\hat{\boldsymbol{\alpha}}^2$ are adequate for all points of the data set, whereas, in fact, they are locally estimated. Note that the estimated optimal regularization parameters $\hat{\boldsymbol{\alpha}}^2$ are globally used for estimating the Hessian for each data point within the optimal bandwidth estimation procedure. One possibility would be to perform the described EM algorithm for each data point, i.e., to alternate Equation (3.10) and Equations (3.11) and (3.15) for all $\boldsymbol{x}_i$. We have empirically observed, however, that, with regard to $\hat{\boldsymbol{\alpha}}^2$, the results of the algorithm change little with this modification.

A further step would be to consider a fully adaptive ridge regression procedure, where the regularization penalty is a symmetric positive definite matrix (no longer diagonal). The minimax efficiency of adaptive ridge regression for quadratic losses was studied, e.g., by Strawderman (1978). As future work, we plan to examine the statistical efficiency of this method for both vector and full-matrix penalties.

4

NONLINEARITY IN NEURAL ENCODING MODELS

4.1 INTRODUCTION

In this chapter, we describe a successful application of well-known ideas on regularized regression to the problem of estimating the instantaneous firing rate of neurons in the primary visual cortex. This work appears in the published paper (Vidaurre et al., 2011e).

In neuroscience, encoding is the task of studying the spike firing rate of a neuron or ensemble of neurons in response to some stimuli. The prediction of neural firing patterns from external, dynamic stimuli is an important task for understanding neuronal behavior (Brown et al., 2004; Kass et al., 2005). It is well known that pure linear models often fail to infer spiking responses (Machens et al., 2004). In general, this may depend on the type of stimulus, the specific physiology of the observed neurons and the species under study. However, some amount of nonlinearity (often high) is always to be expected.

We consider models where the response is the spike firing rate of a single neuron at each time point, and the input variables or predictors are the previous stimuli within a certain time window. Typically, nonlinearity can be introduced after (over) a linear temporal filter on the time-varying stimulus, like the linear-nonlinear Poisson (LNP) model (Brenner et al., 2000; Paninski, 2004; Paninski et al., 2004; Truccolo et al., 2005; Pillow and Simoncelli, 2006; Plourde et al., 2011), or before the linear temporal filter (Ahrens et al., 2008). In the regression framework, this linear temporal filter is equivalent to the vector of regression coefficients. By establishing a static nonlinear transformation on some linear combination of the inputs, the LNP model can, for example, capture the particular nonlinearity of the actual spike generation. This is done within the generalized linear model (GLM) framework, which provides much popular statistical machinery. The LNP model is usually employed for single trial analysis, where the continuous point process is discretized into bins that are small enough to contain one spike at most. Spiking history can be readily included in the LNP model; see, e.g., (Truccolo et al., 2005). We focus, instead, on the second approach, which builds nonlinearity separately on each input before applying the linear temporal filter, thus dealing with nonlinear dendritic behavior and other preliminary nonlinear neuron processes. Both approaches often consider only an additive (linear) contribution of the inputs, i.e., they do not include interactions among different inputs.

Our starting points are the models introduced by Ahrens et al. (2008): the bilinear model and the fullrank model. Both models make use of a basis expansion of the predictors (Schumaker, 2007) to achieve nonlinearity, ignoring any possible interactions among different stimuli. Therefore, these models address the preliminary neuron processes of the second approach mentioned above. Unlike the LNP model, these and the models we propose attempt to model trial-averaged data, working on the basis of the

peristimulus time histogram (PSTH) (Gerstein and Perkel, 1969) in order to focus on the stimulus effect and ignore trial-to-trial variability.

We intend to reach more complex nonlinear relations, which will presumably help to deal with the complex within-neuron processes. The techniques used in this chapter, whose building blocks are well known to the statistics community, generalize the linear regression setting to accommodate nonlinearity by expanding the original set of variables, much like the bilinear and fullrank models. Note that the LNP model can also deal with input-level nonlinearity by considering interactions between the stimuli and other higher-order terms, on which linear regression is performed; see, e.g., (Plourde et al., 2011). The user is usually required to define these higher-order terms, although some can be subsequently discarded using a sparse regularization technique (Tibshirani, 1996). Contrariwise, the models introduced in this work behave purely non-parametrically, and it is the data that automatically determine the functional form of each predictor. Functionals defined on predictor subsets can also be considered, leading to extremely flexible interaction modeling. The ascertain of these interactions and their shape can provide useful biological insight into neuron behavior.

On the other hand, many of the methods in the literature ignore the internal variation of the subject. Once a model (parameterized by a set of parameters) is obtained, it will remain unchanged for all future inferences. However, some studies reveal that several neural systems can vary the stimulus effect and the spiking pattern; see for example (Bezdudnaya et al., 2006; Haider et al., 2007). Here, we also study how to generate ad-hoc, adaptive model parameters for each time point and what impact it has on the model performance. We make use of local analysis based on local regression (Loader, 1999), a technique supported by a solid theoretical background. We also study how to incrementally combine several models in a local manner to prevent (stimulus) overfitting. Our objective here is to be able to deal with a spiking response to stimuli that changes over time.

The proposed local models are to some extent related to models that include spike-history terms, which also consider the internal variation of the subject by making the current response to be dependent on previous responses. Local models include internal variation in a more general manner, without expressing variation just by means of previous responses.

Escola et al. (2011) recently propose to model multistate neurons by hidden Markov models, where each state corresponds to a different GLM. Although this is a successful idea, local models are easier to obtain and, since subject variation is in this work defined on a continuum, we do not need to beforehand fix the number of states. On the other hand, Czanner et al. (2008) propose the state-space GLM, which combines a point-process representation of the spikes series with a hidden Markov model that defines a (continuous) state of the model at each time point. They discuss the PSTH and the GLM as specific cases. These and other models that take into account the model state are often based on Monte Carlo random sampling and, hence, are computationally expensive and less simple than the proposed local approaches.

4.2 SETTING AND PRELIMINARY METHODS

We consider that, at each time point $t \in \{1, ..., T\}$, we have a d-dimensional stimulus $\boldsymbol{s}(t)_{t=1}^{T}$ and a single-neuron response $r(t)_{t=1}^{T}$ to these stimuli. The objective is to predict each response $r(t)$ from previous consecutive stimuli $\boldsymbol{s}(t-i)_{i=1}^{p}$ Let $r(t)$ represent the

number of spikes in a time slice centered at t, i.e., the firing rate at t. This is the well-known PSTH. Here, $r(t)$ is measured as the average spike count at time point t over B trials,

$$r(t) = B^{-1} \sum_{b=1}^{B} r_b(t),$$

where $r_b(t)$ is the measured spike count at time t for trial b. This way, the model omits membrane-memory effects and erratic bursting. Hence, we can model $r(t)$ as a function only of previous stimuli plus some noise:

$$r(t) = g(\boldsymbol{s}(t-i)_{i=1}^{p}) + \epsilon(t),$$

where $g(\cdot) : \mathbb{R}^p \to \mathbb{R}$ is some nonlinear function, p is a parameter indicating the number of past stimuli influencing the current response and $\epsilon(t)$ is the noise term. We model the noise as

$$\epsilon(t) \sim N(0, \sigma^2(t)). \tag{4.1}$$

Noise stands for variability in the response that is not explained by the stimuli, and may either be given by internal processes or be purely random. Provided that $r(t)_{t=1}^{T}$ is trial-averaged, the Gaussian noise assumption is reasonable (Averbeck et al., 2006; Ahrens et al., 2008).

The basis for our comparisons is two basic nonlinear models devised by Ahrens et al. (2008) that lean on the linear regression framework. These are the bilinear model and the fullrank model. The *bilinear* model computes the estimated response $\hat{r}(t)$ as

$$\hat{r}(t) = \hat{\mu} + \sum_{i=1}^{p} \sum_{j=1}^{d} \beta_{ij} f(s_j(t-i)), \tag{4.2}$$

where $s_j(t-i)$ is the j-th component of $\boldsymbol{s}(t-i)$ and $\hat{\mu}$ is the baseline firing rate, which we can estimate by the mean of the response, $\sum_{t=1}^{T} r(t)/T$. We denote as $\boldsymbol{\beta}$ the vector of coefficients $(\beta_{11}, ..., \beta_{1d}, ..., \beta_{p1}, ..., \beta_{pd})$.

The nonlinear function $f(\cdot) : \mathbb{R} \to \mathbb{R}$ is defined as a linear combination of a fixed set of basis functions $f_k(\cdot) : \mathbb{R} \to \mathbb{R}$, $k \in \{1, ..., q\}$. Each basis function $f_k(\cdot)$ has a single input. Such functions are defined as piecewise linear functions determined by a predefined set of equidistant nodes $\{\delta_1, ..., \delta_q\}$ that covers the entire range of the stimulus. Figure 4.1(a) shows a representation of these functions for $q = 10$.

Therefore, given a second vector of coefficients $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_q)$ and some univariate stimulus s_j, $f(\cdot)$ is defined as

$$f(s_j) = \sum_{k=1}^{q} \alpha_k f_k(s_j), \tag{4.3}$$

where

$$f_k(s_j) = \begin{cases} (s_j - \delta_{k-1})/(\delta_k - \delta_{k-1}) & \text{if } k > 1 \text{ and } \delta_{k-1} \leq s_j < \delta_k \\ (\delta_{k+1} - s_j)/(\delta_{k+1} - \delta_k) & \text{if } k < q \text{ and } \delta_k \leq s_j < \delta_{k+1} \\ 0 & \text{otherwise.} \end{cases}$$

Thus, vectors $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ need to be estimated, resulting in a total of $pd + q$ parameters. This is done by updating $\boldsymbol{\beta}$ by ordinary least squares for a fixed $\boldsymbol{\alpha}$ and then updating $\boldsymbol{\alpha}$ by ordinary least squares for a fixed $\boldsymbol{\beta}$. Starting from an arbitrary initial value for

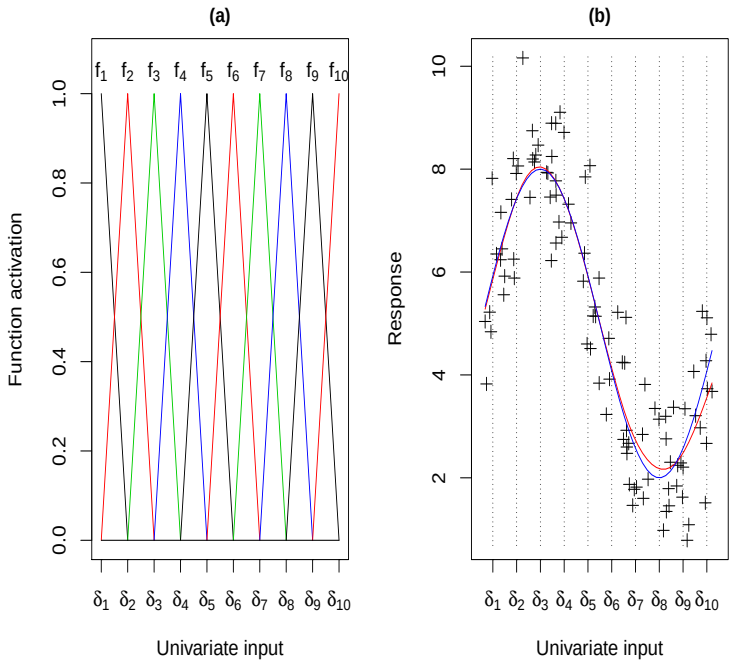


Figure 4.1: (a) Piecewise linear functions used by the bilinear and fullrank models ($q = 10$). (b) A cubic smoothing spline fit to a univariate input, with knots $\{\delta_1, ..., \delta_{10}\}$. The red line is the fitting curve and the blue line is the true function.

either $\boldsymbol{\beta}$ or $\boldsymbol{\alpha}$, this step is repeated until convergence. This is known as alternating least squares (Young et al., 1976).

As noted by Ahrens et al. (2008), the bilinear model is related to separable receptive fields (DeAngelis et al., 1995). When the receptive field is separable, the response cannot be expressed as a product of a function that only depends on time and a function that only depends on space (here, on the basis expansion instead).

On the other hand, the *fullrank* model is determined by qpd instead of $q + pd$ parameters. The fullrank model is defined as:

$$\hat{r}(t) = \hat{\mu} + \sum_{i=1}^{p} \sum_{j=1}^{d} \sum_{k=1}^{q} \gamma_{ijk} f_k(s_j(t-i)),$$

where functions $f_k(\cdot) : \mathbb{R} \to \mathbb{R}$ are defined as before.

Parameters γ_{ijk} are estimated by least squares. In both the bilinear and the fullrank schemes, the parameters are estimated from a fixed train data set with N response values and p stimuli values per response. These parameters will be used to estimate future responses.

Both the bilinear model and the fullrank model have three shortcomings. First, the basis functions are data-independently defined. Second, they do not consider interactions between input variables, so the nonlinear power of the models is limited. We discuss the basic details of nonlinear regression below. Third, the learnt parameters are not time-varying, so that the response function is always the same. A fixed noise

standard deviation $\sigma(t) = \sigma$ is also assumed. In the following section, we propose the use of some nonlinear techniques to overcome these limitations.

4.3 BASIS EXPANSIONS AND LOCAL METHODS

In this section, we survey and discuss how to apply some nonlinear approaches to spike train data to obtain better results. Whatever approach we follow, we must somehow control the nonlinearity or complexity of the model. More complex models are less biased in exchange for increased variance. In general terms, we would choose simpler models for limited or ill-posed data and more complex models for well-behaved data. The objective is a model that optimizes the bias-variance trade-off, that is, that minimizes the expected prediction error. Automatic, data-dependent methods are preferred to control the model complexity. For example, techniques based on regularization (Bickel and Li, 2006), are useful for adjusting the complexity of the model and restricting its variance by imposing some constraint on the model parameters.

There are two fundamental approaches for achieving nonlinearity. First, we can seek a more complex model than the linear model by establishing a linear combination of some basis expansions of the original terms (Schumaker, 2007). This is the approach taken by the bilinear and fullrank models. Second, we can fit simple (linear) models for different areas of the data domain (Loader, 1999), accounting for time-varying subject states. This is the case of local regression, discussed in previous chapters. In the following, we present these techniques in detail. Also, we present the incremental local learning approaches. Whereas pure local models are useful for monitoring the changing behavior of the neuron (see the experimental subsection for examples), the proposed incremental local learning scheme, in addition, is robust to stimulus overfitting.

Regression splines (Schumaker, 2007) is a family of popular nonlinear models that makes use of basis expansions of the original terms. In the univariate case, the input domain is divided into contiguous intervals, separated by a fixed set of knots. Whereas the placement of the knots can be data-driven, the number of knots is often specified by the user. In each interval, a polynomial function of order M is fitted, in such a way that the entire function is continuous, and has continuous derivatives up to order $M - 2$ to assure continuity. Figure 1(b) shows a univariate cubic regression spline fit ($M = 4$) with $q = 10$ knots.

A more flexible, likewise spline-based, model is known as smoothing splines. Here, a maximal set of knots is used, and complexity is controlled by penalizing (regularizing) the curvature of the fitted function. These models can be extended to the multivariate case, giving rise to very flexible models. Here, we make use of two different spline methods, with a different degree of flexibility (complexity).

The first is known as *multivariate adaptive regression splines*, or MARS, devised by Friedman (1991), where the basis functions are linear splines. For simplicity, we assume a unidimensional stimulus ($d = 1$). For each input variable, indexed by $i \in \{1, ..., p\}$, and each instance in the training data set, indexed by n, two piecewise linear functions are defined, $f^1_{ni}(\cdot) : \mathbb{R}^p \to \mathbb{R}$ and $f^2_{ni}(\cdot) : \mathbb{R}^p \to \mathbb{R}$, with one knot at $s(n - i)$. For a given vector $\boldsymbol{x} \in \mathcal{R}^p$, these are defined as

$$f^1_{ni}(\boldsymbol{x}) = \begin{cases} x_i - s(n-i) & \text{if } x_i > s(n-i) \\ 0 & \text{otherwise} \end{cases}$$

$$f_{ni}^2(\boldsymbol{x}) = \begin{cases} s(n-i) - x_i & \text{if } x_i < s(n-i) \\ 0 & \text{otherwise} \end{cases}$$

The MARS algorithm performs a forward greedy search, sequentially adding terms $h_{i^*}(\cdot)$, which are a single piecewise linear function or a product of two (or more) piecewise linear functions. To prevent overfitting, a backward deletion procedure is applied afterwards. Generalized cross-validation (Craven and Wahba, 1979) is used to decide how much the model should be pruned. Thus, MARS considers interactions between the stimuli. The resulting model has the form

$$\hat{r}(t) = \hat{\mu} + \sum_{i^*=1}^{p^*} \beta_{i^*} h_{i^*}(s(t-i)_{i=1}^{p}), \tag{4.4}$$

where p^* is the number of terms included into the model, $s(t-i)_{i=1}^{p}$ represents a vector with the values of the stimulus from $t-1$ to $t-p$ and β_{i^*} are the parameters for each term, which are estimated by least squares. The maximum number of functional terms in such products can be considered as a parameter of the algorithm. MARS produces continuous models with continuous derivatives.

On the other hand, we consider the *sparse additive model* (SpAM), devised by Ravikumar et al. (2009). SpAM was briefly described in chapter 1. SpAM employs regularization instead of greedy searching to control the flexibility of the model. Like the additive model, proposed by Hastie and Tibshirani (1999), SpAM considers an additive linear combination of univariate functions and ignores interactions among the input variables:

$$\hat{r}(t) = \hat{\mu} + \sum_{i=1}^{p} f_i(s(t-i)), \tag{4.5}$$

where each $f_i(\cdot) : \mathbb{R} \to \mathbb{R}$ is, e.g., a cubic regression spline (see Figure 4.1(b)).

To achieve sparseness, SpAM employs an L_1-penalty imposed on the component L_2-norms of the functions, which is given by $(\int_0^T f_i^2(t)dt)^{1/2}$, so that the magnitude of the functional predictors is penalized. This leads to a number of terms (depending on the value of some regularization parameter) being effectively discarded. Unlike the classic additive model, regularization allows SpAM to be used in high-dimensional settings, as it is more interpretable, less sensitive to overfitting and also computationally efficient. The estimator is obtained by formulating a convex optimization problem, which is solved with a backfitting algorithm, described in (Hastie and Tibshirani, 1999).

Note that, whereas we need to manually setup the basis expansion configuration for the bilinear and fullrank models beforehand, both MARS and SpAM can automatically adjust model complexity during the training process. In addition, MARS can account for interactions among the predictors.

On the other hand, kernel smoothing models or local regression models, fit a simple model at each query point. For example, linear local regression (Loader, 1999) fits a linear regression model for each query point (for each spike count to be predicted), using some weighted neighborhood composed of the closest data to the query item. Here we apply this idea to the nonlinear methods described above.

Data (within the neighborhood) are typically weighted according to some kernel function $K_\tau()$ and the distances to the target item. The distances are measured either on the input variable space or can be supplied by the domain itself. In our case, the time dimension perfectly suits for our purposes. Let $n \in \{t-N, ..., t-1\}$, so that the distances are computed as the number of time slices between the target item and each

previous data item, so that data items closer in time will be given more importance than further data items. We compute the weights w_n with the well-known tricube kernel function:

$$w_n = K_\tau(t-n+1) = \begin{cases} (1-(t-n+1)^3)^3 & \text{if } (t-n+1) \leq \tau \\ 0 & \text{otherwise,} \end{cases} \tag{4.6}$$

where τ is a smoothing parameter or bandwidth that indicates the width of the neighborhood. In previous chapters, we refer the bandwidth as h. In this chapter, we denote it as τ for notational requirements. We need to estimate τ, which can be unique for all predictions or adaptively selected. For simplicity, we consider a unique smoothing parameter, $\tau = N$, so that all data items in the built-in data set are used. Remember that the neighborhood only includes past data items.

Note that this weighting scheme can be applied to any of the algorithms presented above (bilinear, fullrank, MARS and SpAM), just by computing the weights and weighting the data set accordingly. This would yield their corresponding local versions. For an efficient model assessment, generalized cross-validation (Hastie et al., 2008) is a very efficient approximation to leave-one-out cross-validation for Gaussian data.

Note that overfitting is a sensitive issue when using very flexible models like the above nonlinear procedures. In this case, there are two types of overfitting: overfitting to the firing rate and overfitting to the stimulus. Model selection based on cross-validation, for example, can prevent overfitting to the firing rate. In this way, provided that data is trial-averaged, we prevent the selected model from describing the trials within training data too accurately and future trials poorly. On the other hand, overfitting to the stimulus is the non-generalization to stimulus values that differ the training data values. Note that the nature of local analysis, whose goal is precisely to give an accurate estimation for nearby (and thus similar) inputs, leads to overfitting to the stimulus. Therefore, unlike stationary models, models based on local analysis are suitable for evaluating and describing the changing behavior of the neuron, but not for extrapolating to general stimulus values.

However, there are still some approaches to exploit the flexibility of local analysis for stimuli extrapolation. Here, we use an incremental local learning approach, where models corresponding to previous instants are combined in a mixture defined as

$$\hat{r}(t) = \sum_{l=t-L}^{t-1} \omega_l \, \hat{r}_{G(l)}(t).$$

Here $G(l)$ is the model obtained by some stationary method at time point l, $\hat{r}_{G(l)}(t)$ is the response that this model produces at time point t, L is the number of models within the mixture and ω_l are model weights. We can compute the unnormalized model weights with a Gaussian kernel as

$$\omega_l^* = \exp\Big(-\frac{1}{2\varsigma^2} \big(\boldsymbol{s}(l)_{i=1}^p - \boldsymbol{s}(t)_{i=1}^p\big)' \boldsymbol{D} \, \big(\boldsymbol{s}(l)_{i=1}^p - \boldsymbol{s}(t)_{i=1}^p\big) \Big),$$

where $||\cdot||_2$ is the L_2-norm operator, ς is a radius parameter and $\boldsymbol{D}$ is an $nd \times nd$ positive definite matrix that, if the stimulus is standardized, not auto-correlated and its components are independent, we can assume to be the identity. The actual model weights are obtained as

$$\omega_l = \frac{\omega_l^*}{\sum_{l'=t-L}^{t-1} \omega_{l'}^*}.$$

Hence, at each time point, a new model is included into the mixture and the oldest model is discarded. Since we are only considering the previous L time points to yield the current prediction, this is a form of local analysis. Therefore, L defines the extent of locality. Since the final prediction is made by combining the strengths of a collection of basic predictors, this approach can be interpreted as an ensemble learning method (Hastie et al., 2008).

This form of incremental local learning is, to some extent, robust to input (stimulus) overfitting while keeping interesting properties of pure local analysis (Schaal and Atkeson, 1998). In this particular approach, the trade-off between stationarity and locality, parametrized by L, comes from using a local mixture of stationary models.

Note that the computational burden is the same than that of the pure local approach. We need, however, a bigger amount of memory to store the L models.

In summary, this chapter investigates the effect of different types of nonlinearity to improve spike firing rate prediction for averaged trials. Firstly, the use of regression splines methods (MARS and SpAM) aims to deal the high complexity of neurological processing. Besides, we consider combinations of different stimuli instead of pure additive models by using MARS. Secondly, local methods can incorporate the subject evolution into the model and suppress the assumption of a stationary model by using only the recent subject states to build the model. Also, it avoids the fixed noise variance assumption. The bilinear and fullrank models, as suggested by Ahrens et al. (2008), use the same model (obtained from a separate training data set) to predict all future data items, thus ignoring the internal evolution of the subject. Finally, incremental local learning is a type of local approach between stationary and pure local analysis.

4.4 EXPERIMENTS ON SYNTHETIC DATA

We first study the different nonlinear approaches on five different synthetic scenarios or spike generating processes, following a similar experiment design than Ahrens et al.'s (2008). The first three scenarios are the same as the three data models used by Ahrens et al. (2008). The remaining two are proposed here so as to reflect dynamic changes in the subject stimulus-response function. Results are evaluated by their predictive power (Sahani and Linden, 2003).

In all experiments we generate a one-dimensional stimuli vector, $s(t)_{t=1}^{T}$, from a normal distribution $\mathcal{N}(0,1)$, with $T = 1000$ time points. From this stimuli vector we obtain a firing probability $P(t)_{t=1}^{T}$ for each scenario according to one of the five generating processes described below (plus some uniform noise, $\mathcal{U}(-0.1, 0.1)$). All generating processes take into account the last $\eta = 10$ time points (the previous η stimuli). The first η values of $P(t)_{t=1}^{T}$ are set to zero by convention. Each resulting vector $P(t)_{t=1}^{T}$ is scaled to lie in $[0,1]$.

The five generating processes are:

- **I. One stationary filter**. $P(t) = \sum_{i=1}^{\eta} \xi_i^{(1)} (s(t-i))^2$, where $\xi_i^{(1)} = sin(v_i)/2$ and $\boldsymbol{v}$ is a vector containing η equidistant increasing values (in radians) between $\pi/2$ and π.

- **II. Two stationary filters**. $P(t) = \sum_{i=1}^{\eta} \xi_i^{(1)} s(t-i) + \sum_{i=1}^{\eta} \xi_i^{(2)} (s(t-i))^2$, where elements $\xi_i^{(1)}$ are defined as in process I, and $\xi_i^{(2)} = (sin(v_i) + 1)/4$. Here, $\boldsymbol{v}$ is a

vector containing η equidistant increasing values (in radians) between $\pi/4$ and $3\pi/2$. Figure 4.2(a) shows filter vectors $\boldsymbol{\xi}^{(1)}$ and $\boldsymbol{\xi}^{(2)}$.

- **III. Nonlinear feature selective process**. $P(t) = \sum_{i=1}^{\eta} \mathcal{I}(|s(t-i) - v_i| < 0.2(max(s(t)_{t=1}^{T}) - min(s(t)_{t=1}^{T}))$, where $\mathcal{I}(\cdot)$ equals 1 when its argument is true and 0 otherwise, and $\boldsymbol{v}$ is a vector containing η equidistant increasing values between $min(s(t)_{t=1}^{T})$ and $max(s(t)_{t=1}^{T})$.

- **IV. One non-stationary filter**. $P(t) = \sum_{i=1}^{\eta} \xi_i^{(1)}(t)(s(t-i))^2$, where $\xi_i^{(1)}(t) = (T-t+1)sin(v_i)/2T$, and $\boldsymbol{v}$ is a vector containing η equidistant increasing values (in radians) between $\pi/2$ and π. Figure 4.2(b) shows the non-stationary filter vector $\boldsymbol{\xi}^{(1)}(t)$ for several time points.

- **V. Two non-stationary filters**. $P(t) = \sum_{i=1}^{\eta} \xi_i^{(1)}(t)s(t-i) + \sum_{i=1}^{\eta} \xi_i^{(2)}(t)(s(t-i))^2$, where elements $\xi_i^{(1)}(t)$ are defined as in process IV, and $\xi_i^{(2)}(t) = (sin(v_i) + 1)/u_t$. Here, $\boldsymbol{v}$ is a vector containing η equidistant increasing values (in radians) between $\pi/4$ and $3\pi/2$ and $\boldsymbol{u}$ is a vector containing T equidistant increasing values in the interval $[4, 10]$. Figure 4.2(c) shows the non-stationary filter vector $\boldsymbol{\xi}^{(2)}(t)$ for several time points.

At each time bin, the generating processes can produce up to twelve spikes. Five trials are considered. Hence, for each generating process, we obtain five spike trials of length T from a binomial distribution $Binom(12, P(t))$, $t = 1, ..., T$. By averaging such spike trains over the five trials, we compute the observed firing rate $r(t)_{t=1}^{T}$.

We have tested the bilinear and fullrank models, MARS and SpAM, their local counterparts (L.bilinear, L.fullrank, L.MARS and L.SpAM) and the corresponding incremental local approaches (IL.bilinear, IL.fullrank, IL.MARS and IL.SpAM) on these five experimental settings. For each generating process, we sampled ten pairs $(P(t)_{t=1}^{T}, r(t)_{t=1}^{T})$, corresponding to ten simulated neurons, from the same stimuli vector $s(t)_{t=1}^{T}$. All models have been trained taking $p = 20$ previous stimuli into account. For the non-local methods, we built the models on the first half of the data, using $N = T/2 - p$

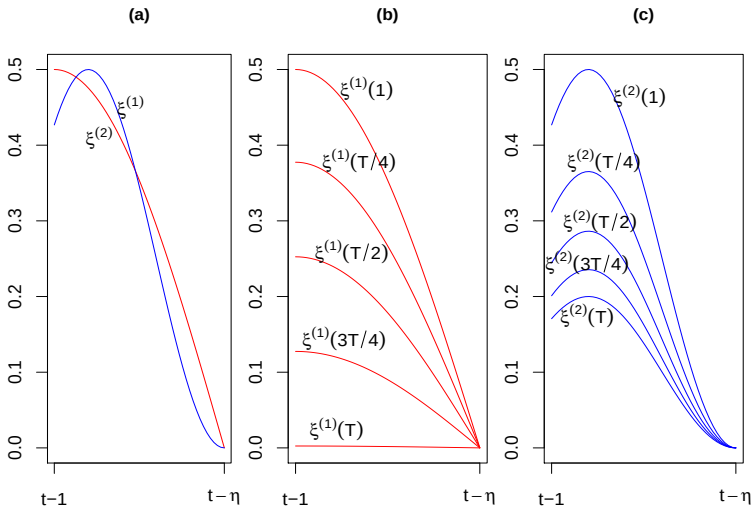


Figure 4.2: (a) Stationary filter vectors $\boldsymbol{\xi}^{(1)}$ and $\boldsymbol{\xi}^{(2)}$. (b) Non-stationary filter vector $\boldsymbol{\xi}^{(1)}(t)$ at various time points. (c) Non-stationary filter vector $\boldsymbol{\xi}^{(2)}(t)$ at various time points.

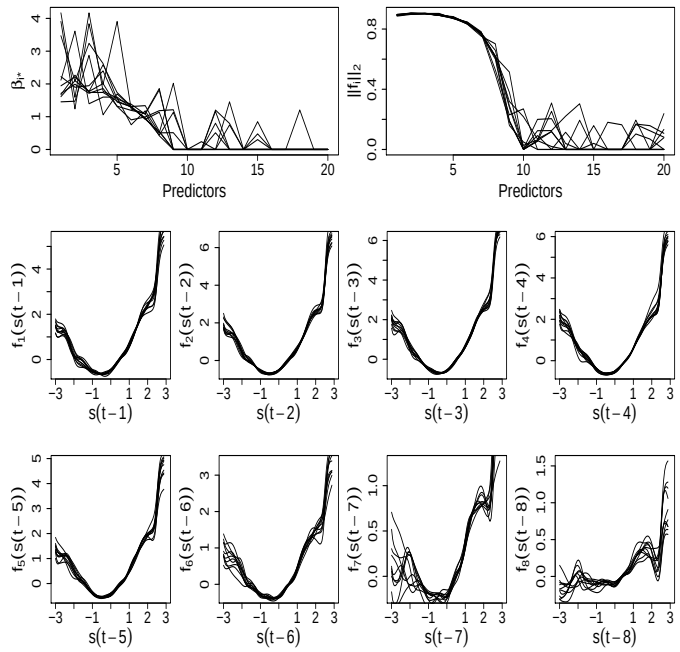


Figure 4.3: Predictor importance for the stationary MARS (top left) and stationary SpAM (top right) models, and the eight most relevant functional predictors in the stationary SpAM models, for generating process II.

data items, and we tested them on the second half. For the local methods, a different model is built for each data item in the second half of the data, where the training data set includes the $N = 300$ last responses. The bilinear and the fullrank models were trained using $q = 5$ piecewise functions. L is set to 100 for the local incremental methods.

An advantage of the bilinear and fullrank methods is that these models can be graphically expressed, providing a compact description of the neuronal function. For instance, Ahrens et al. (2008) show several plots. The MARS and SpAM models, since they are non-parametric, are more complex to be graphically described. However, one can still use simple graphs to ascertain the importance of the predictors within the model. Figure 4.3 (top graphs) shows measures of the importance of each predictor for MARS (left) and SpAM (right), for data obtained from generating process II. Each line represents one neuron. In the MARS case, this is the highest absolute coefficient β_{i^*} (Equation (4.4)) that involves each predictor. In the SpAM case, this is the L_2-norm of functions $f_i(\cdot)$ (Equation (4.5)) of each predictor. The eight bottom graphs of Figure 4.3 depict the univariate functions for the eight most relevant predictors in the SPaM models, giving an idea of the influence of these predictors on the response. Again, each line represents one neuron.

Figure 4.4 and Figure 4.5 illustrate the model fits for the local bilinear model and local SpAM, respectively, at various time points ($t = 501, 667, 834, 1000$) for generating process I. Within each figure, each pair of graphs is related to a time point. Each line represents a different neuron. Local bilinear models are represented by coefficients β_{i1} of Equation (4.2) (left graphs) and coefficients α_k of Equation (4.3) (right graphs). Local SpAM models are represented by the L_2-norm of functions $f_i(\cdot)$. Figure 4.6 and Figure 4.7 show the same for generating process IV.

Note that the models that correspond to generating process I (stationary) vary less across the different time points ($t = 501, 667, 834, 1000$) than those of generating pro-

cess IV (non-stationary). This indicates to the practitioner that, for generating process I, a global model is preferable, whereas, for generating process IV, a local model is more adequate. Valuable biological insight about the underlying biological process can be extracted from this fact.

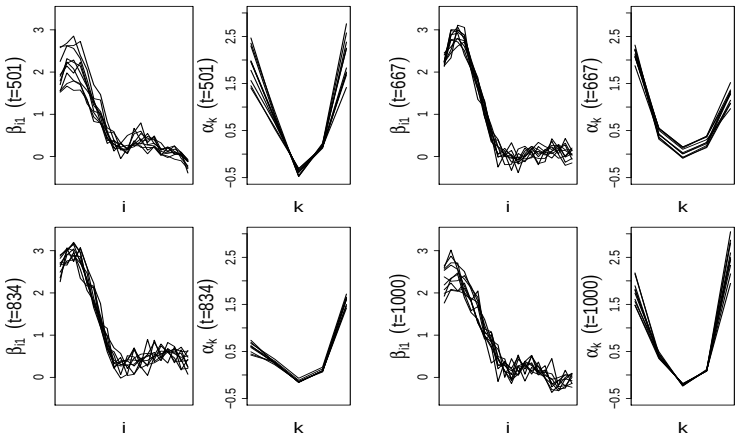


Figure 4.4: Local bilinear models for generating process I at various time points.

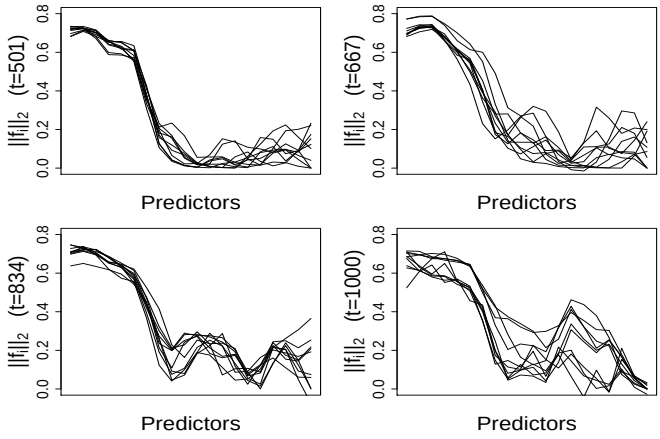


Figure 4.5: Predictor norms of the local SpAM fits for generating process I at various time points.

Figures 4.8-4.12 depict the true firing rate (blue) superimposed on the estimated firing rate (red), for some simulated neuron (the same for all figures) and $t = 901, ..., 1000$. Focusing on the stationary models (left graphs), simple visual inspection appears to indicate that the fullrank and bilinear models predict the firing rate approximately as well as the more complex MARS and SpAM models for all generating processes, although the performance of the bilinear model is slightly worse for generating process III, whose nonlinearity is more difficult to capture. In the local models (right graphs), however, L.SpAM seems to do the best job overall. L.SpAM is more accurate than L.MARS because L.MARS considers interactions between the inputs, whereas L.SpAM focuses the nonlinear strength separately at each input. Since the local models are built with fewer training data items (lower N) than the stationary models and there is no input interaction in any of the generating processes, MARS may slightly

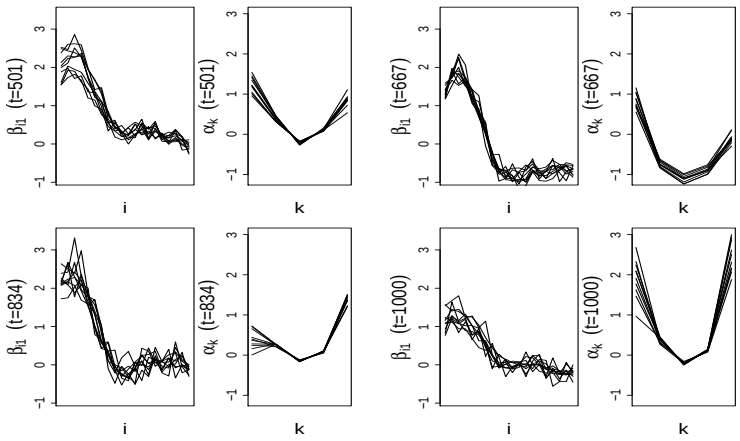


Figure 4.6: Local bilinear models for generating process IV at various time points.

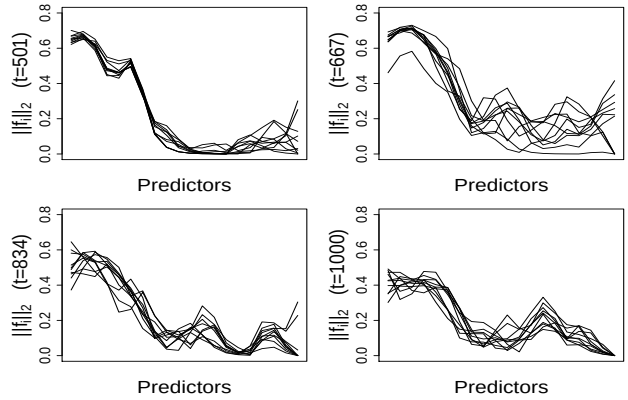


Figure 4.7: Predictor norms of the local SpAM fits for generating process IV at various time points.

overfit the data. As expected, all local models behave much better for generating processes IV and V because they are non-stationary.

Figure 4.13 illustrates a quantitative comparison. It reports the predictive power (Sahani and Linden, 2003) of the methods, comparing each non-local method with its local counterpart. Each point represents a neuron simulated by each of the five generating processes. Generating processes are distinguished by different kinds of symbols. In the left-hand graphs, if one point lies on the left side of the dashed (diagonal) line, then the predictive power of the local method is greater than that of the non-local method, and the opposite applies if the point lies on the right side. The right-hand graphs (and horizontal histograms) show the predictive power difference between each local and non-local method as a function of the noise power (Sahani and Linden, 2003). It is clear that the local methods excel for generating processes IV and V, whereas the non-local methods are better for generating processes I, II and III. The biggest difference is for generating process V, where the local methods are much better than the non-local methods.

Figure 4.14 illustrates a comparison of L.bilinear and L.fullrank against L.MARS and L.SpAM. Interestingly, it now is clear that L.SpAM outperforms both L.bilinear and L.fullrank for all generating processes. L.MARS, however, only behaves much

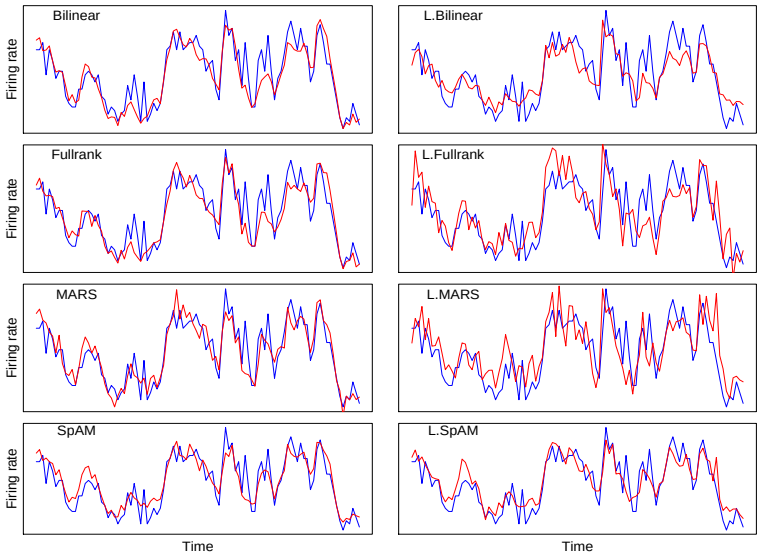


Figure 4.8: Predicted signal (red) and true firing rate (blue) for generating process I in some time range for one neuron.

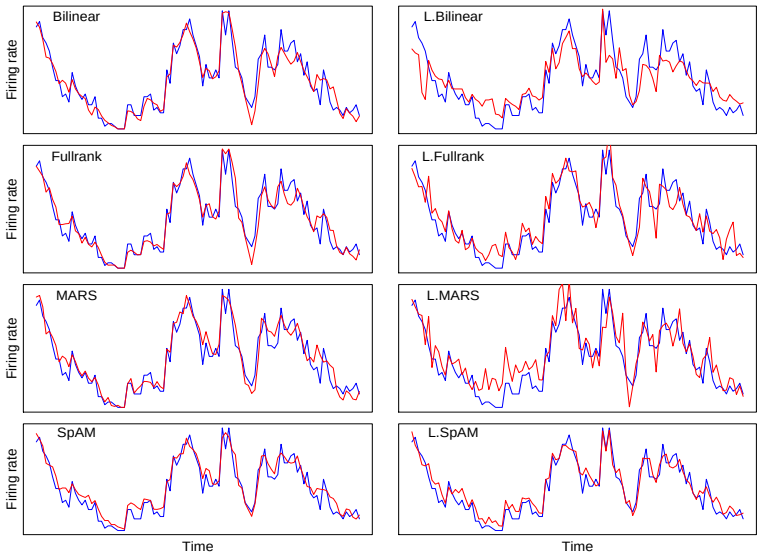


Figure 4.9: Predicted signal (red) and true firing rate (blue) for generating process II in some time range for one neuron.

better for the generating process III, which is a nonlinear process that the bilinear and fullrank methods cannot entirely capture.

Finally, we have checked the robustness to overfitting of the incremental local learning approaches. In this case, we built the models on data until time point t, and we tested the accuracy of the obtained models at time point $t + \Delta t$, where Δt is the timescale on which the firing rate becomes non-correlated (i.e., the first time the autocorrelation hits zero). Δt is, on average, around 10. This is done for all time points. Figures 4.15 and 4.16 show, for generating processes IV and V, predicted and real firing rate for some neuron, giving a graphical example of the overfitting phenomenon in the pure local models. Note that the incremental local approaches appear to be robust to stimulus overfitting. Table 4.1 reports mean absolute error (and standard deviation)

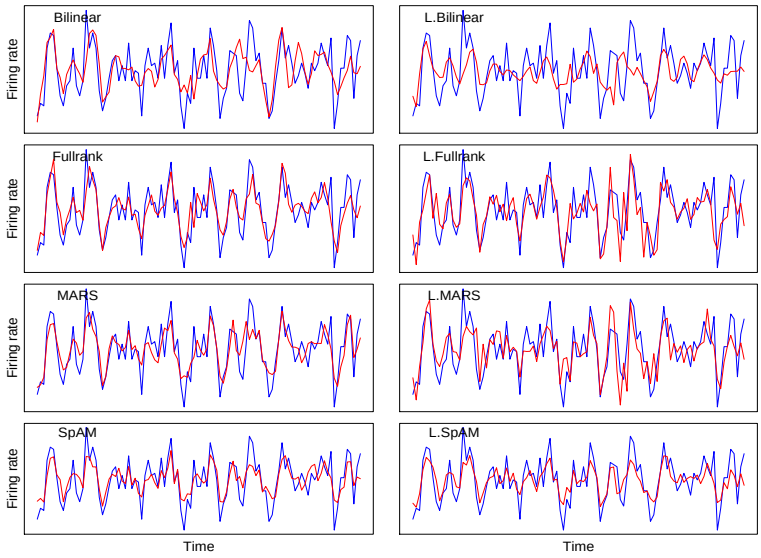


Figure 4.10: Predicted signal (red) and true firing rate (blue) for generating process III in some time range for one neuron.

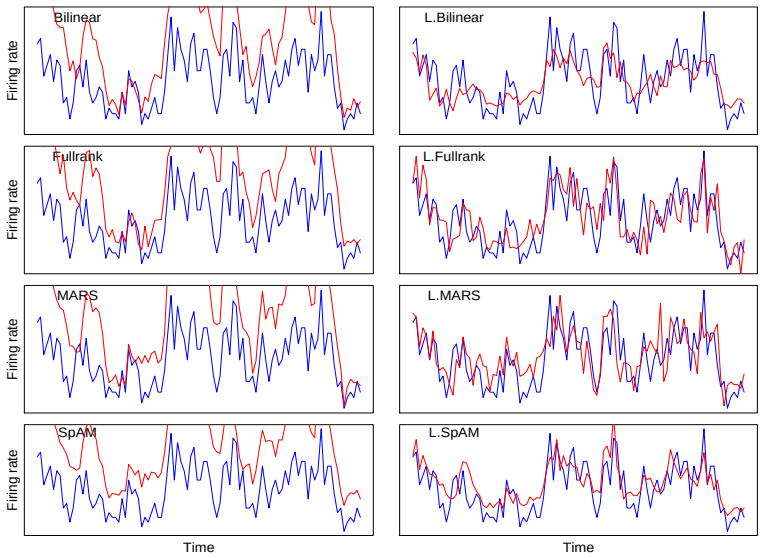


Figure 4.11: Predicted signal (red) and true firing rate (blue) for generating process IV in some time range for one neuron.

for each process and each stationary and incremental local learning approach. Note that the incremental local approaches behave better than the stationary counterparts for generating process IV and V, without showing any sign of stimulus overfitting in any case.

From this experiment, we can conclude that, if there is no interaction between the stimuli, SpAM is a handy algorithm for a broad spectrum of neural spike estimation scenarios, since regularization can automatically adjust the complexity of the model. Interactions between the stimuli are studied in the next section. Also, locality should be taken into account when the encoding function is suspected to vary over time, possible by means of an incremental local approach. We believe that the use of stationary models to characterize neuron responses can sometimes lead to inaccurate predictions

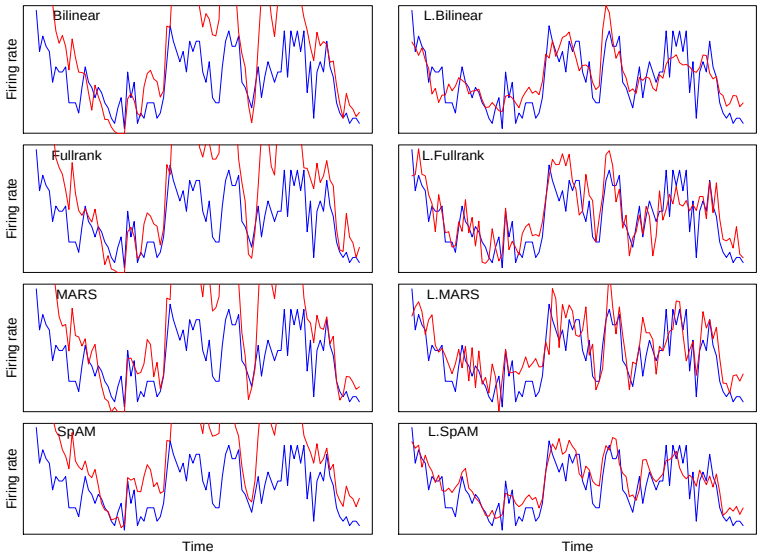


Figure 4.12: Predicted signal (red) and true firing rate (blue) for generating process V in some time range for one neuron.

and is based on an often unrealistic assumption. In the next section, we check these hypotheses on a real scenario.

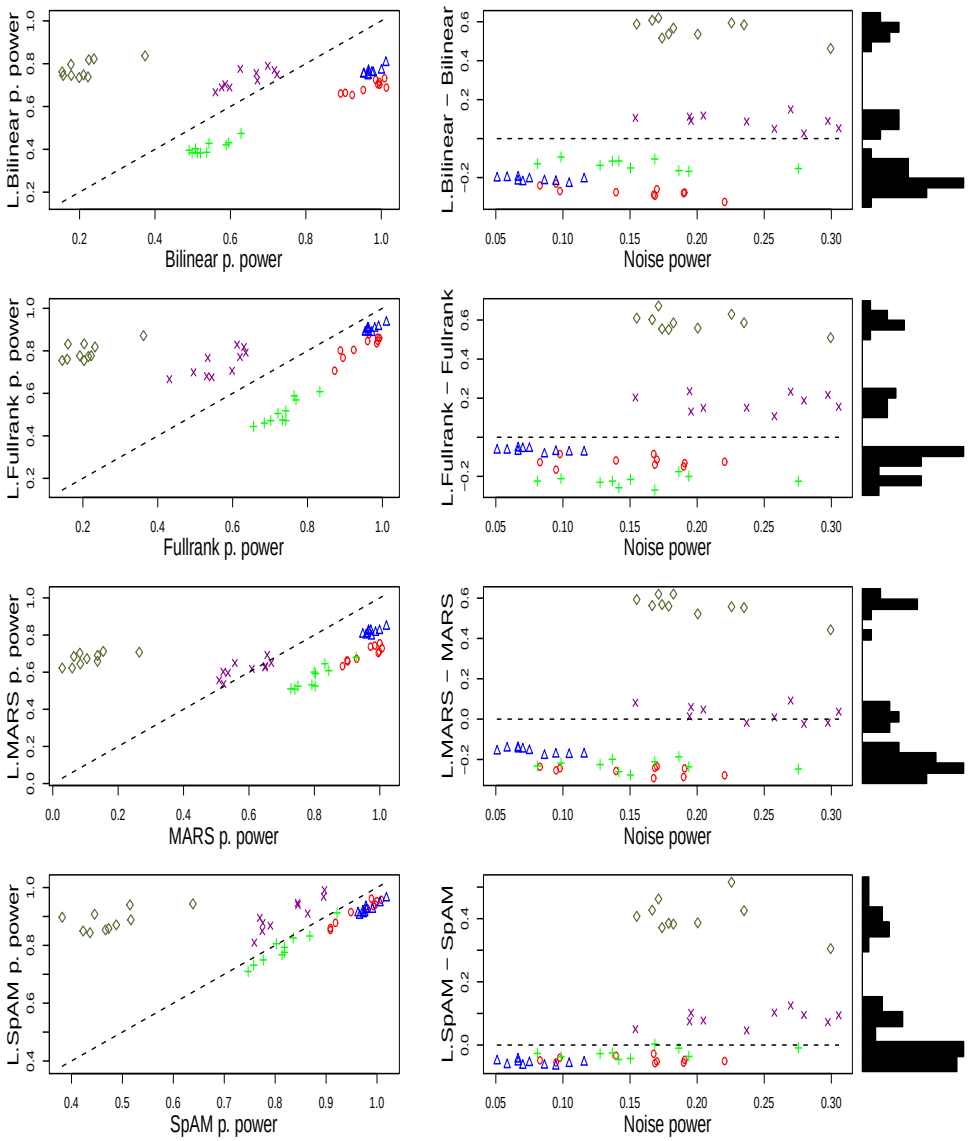


Figure 4.13: Comparison of the non-local methods against their local counterparts in terms of predictive power (p.power). Red ◯-dots correspond to generating process I, blue △-dots correspond to generating process II, green +-dots correspond to generating process III, magenta ×-dots correspond to generating process IV and grey ♢-dots correspond to generating process V.

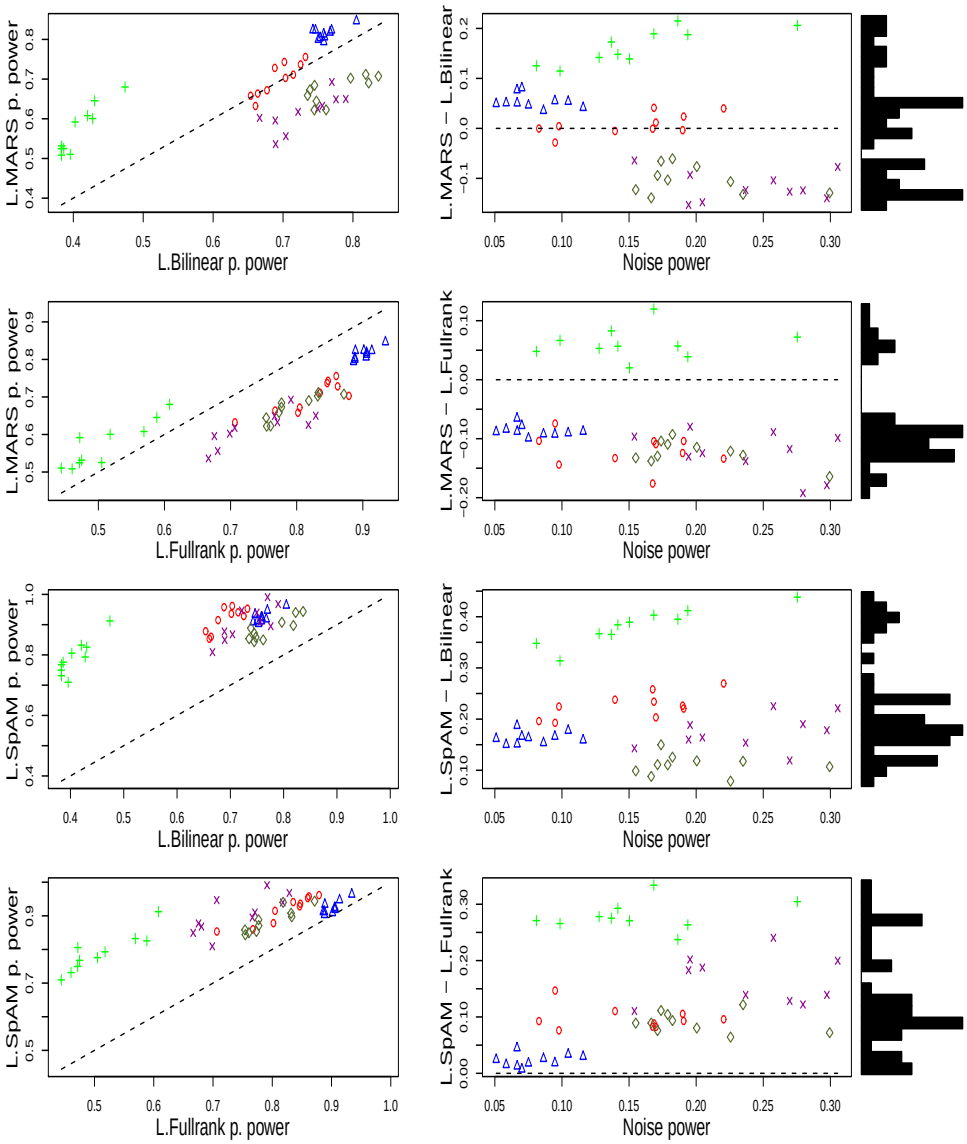


Figure 4.14: Comparison of the local bilinear and fullrank models against the local MARS and local SpAM methods in terms of predictive power (p.power). Red ◯-dots correspond to generating process I, blue △-dots correspond to generating process II, green +-dots correspond to generating process III, magenta ×-dots correspond to generating process IV and grey ♢-dots correspond to generating process V.

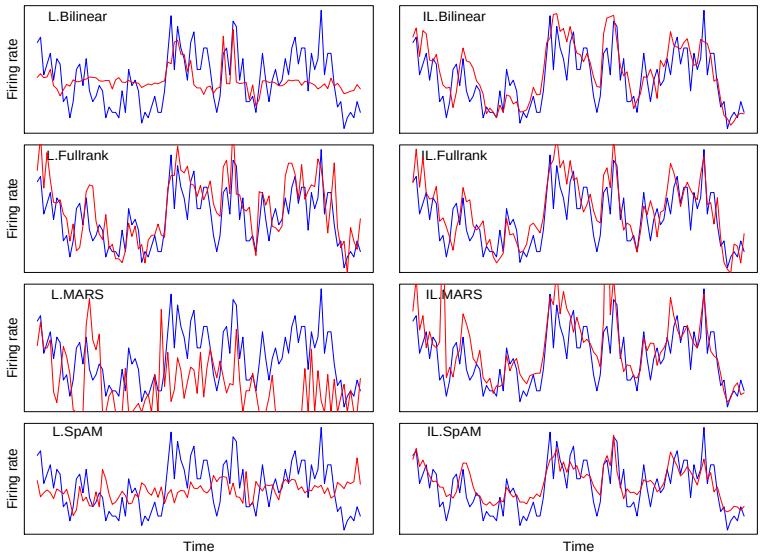


Figure 4.15: Predicted signal (red) and true firing rate (blue) for generating process IV in some time range for one neuron. For each t, models were trained using data until time point t and were tested at time point $t + \Delta t$.

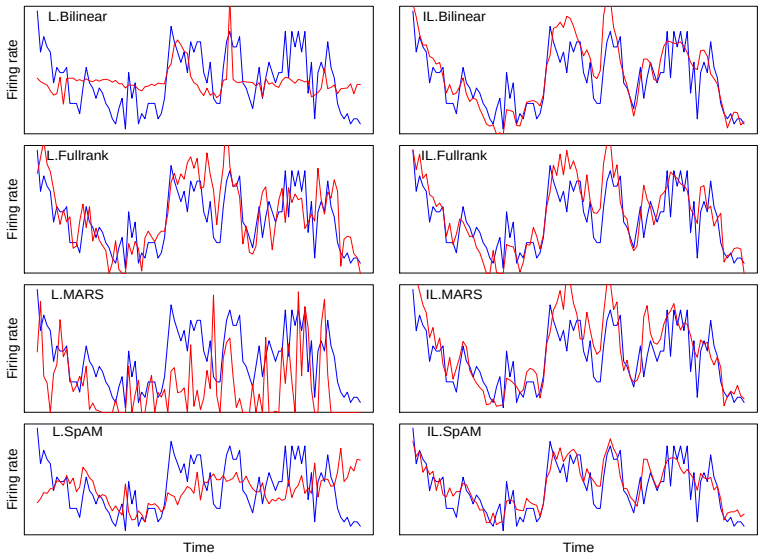


Figure 4.16: Predicted signal (red) and true firing rate (blue) for generating process V in some time range for one neuron. For each t, models were trained using data until time point t and were tested at time point $t + \Delta t$.

4.5 EXPERIMENTS ON REAL DATA

We have also investigated the impact of nonlinearity on real data, in particular large-scale neuronal recordings in cat primary visual cortex (area 17). The data were collected by Tim Blanche at the laboratory of Nicholas Swindale, University of British Columbia, and can be downloaded from the NSF-funded CRCNS Data Sharing website[1].

1 http://crcns.org

Table 4.1: Mean absolute error (and standard deviation) for each process and each stationary and incremental local learning approach. The best result for each row is highlighted in bold.

	Generating process				
Method	I	II	III	IV	V
Bilinear	1.19($\pm$0.9)	1.39($\pm$1.1)	3.49($\pm$1.9)	1.25($\pm$0.9)	1.38($\pm$1.1)
Fullrank	1.20($\pm$0.9)	1.40($\pm$1.1)	3.57($\pm$2.0)	1.32($\pm$0.9)	1.42($\pm$1.1)
MARS	1.28($\pm$0.9)	1.46($\pm$1.2)	3.56($\pm$2.0)	1.36($\pm$1.0)	1.47($\pm$1.2)
SpAM	1.39($\pm$0.8)	1.52($\pm$1.1)	3.49($\pm$1.8)	1.47($\pm$0.8)	1.49($\pm$1.0)
IL.Bilinear	1.19($\pm$0.9)	1.38($\pm$1.1)	3.45($\pm$1.8)	**0.75**($\pm$0.6)	0.69($\pm$0.6)
IL.Fullrank	1.22($\pm$0.9)	1.40($\pm$1.1)	3.50($\pm$2.0)	0.78($\pm$0.6)	0.72($\pm$0.6)
IL.MARS	1.31($\pm$0.9)	1.46($\pm$1.2)	3.53($\pm$2.0)	0.83($\pm$0.7)	0.76($\pm$0.7)
IL.SpAM	**1.16**($\pm$0.7)	**1.19**($\pm$0.9)	**3.41**($\pm$1.8)	0.78($\pm$0.6)	**0.62**($\pm$0.5)

Data corresponds to extracellular neural activity under several types of visual stimuli. We work with the simplest kind of stimulus, consisting of an oriented drifting bar moving on a screen. The drifting bar moves in 18 different directions.

The data set contains eight trials of spiking data for ten (simultaneously recorded) neurons. At each trial, the 18 stimulus values are presented in random order for approximately 4 seconds each. We have partitioned the time range in bins of 100ms, counting the number of spikes at each bin. Therefore, there are 40 bins per stimulus value and $T = 720$ time bins in total.

Note that encoding the stimulus as the number of degrees or as a categorical variable is an incoherent representation. For example, if we represent the bar orientation as the number of degrees, we are implying that the 0° orientation lies far away from the 340° orientation, whereas they are actually only 20° apart. Instead, we use two variables to represent each orientation, $s_1(t) = 0.5\,sin(radians(t))$ and $s_2(t) = 0.5\,cos(radians(t))$. These pairs are Cartesian coordinates on the circumference of diameter 1.0. In this way, we have the maximum Euclidean distance (1.0) between "opposite" stimuli.

Note also that we cannot directly average the spike counts across the trials, because the stimulus values are presented in a different order at each trial. Noise cannot be assumed to be Gaussian (Equation (4.1)) if trials are not averaged and, hence, the aforementioned methods cannot be applied. Instead, we reorder each trial's spike counts so that the 40 bins of the 0° orientation are followed by the 40 bins of the 20° orientation, and so on. We now average the spike counts across the reordered trials. Figure 4.17 illustrates the (averaged) firing rate for two neurons. The graphs on the left are not ordered, so each point in the series is the mean of the spike counts at the same time point. The graphs on the right are ordered, so each point in the series is the mean of spike counts at different time points within the trials (with the same stimulus value, however).

It is obvious that we cannot use previous stimuli for firing rate prediction if they correspond to different stimulus values. However, we hypothetize that the response does not depend here on previous stimulus values, but on the current stimulus value and the amount of time it has been held.

In Figure 4.17 (left graphs), the "non-reordered" firing rate does not appear to follow a clear pattern. Firstly, the smoothed firing rate is less informative for the non-

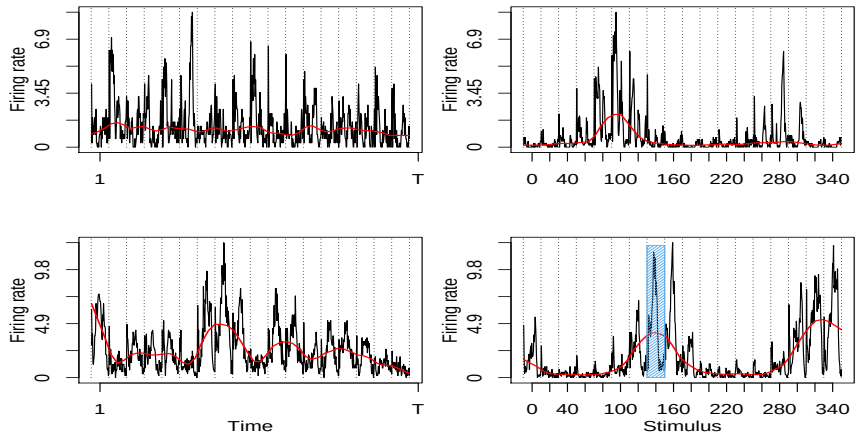


Figure 4.17: Mean spike counts for one neuron (top) and another neuron (bottom) before ordering (left) and after ordering the bins (right). Vertical dotted lines mark the change of stimulus. The red curves are a smoothed version of the spike counts. A common pattern is highlighted in blue.

reordered version. For example, the red curve is very flat for the neuron on top, indicating that there is no substantial change across the entire time range. Interestingly, the right graphs are more informative. The neurons are clearly more active for certain ranges of the stimulus. For instance, the neuron on top fires mostly for stimulus values around 80°-140° and stimulus values around 260°-320°, which are opposite orientations and same direction. Secondly, if we observe the high-frequency scale, it is patent that, in the reordered spike counts, the spiking pattern within each segment of 40 bins is roughly repeated in neighboring segments (segments concerning close stimulus values). For instance, the pattern highlighted in blue in Figure 4.17 (right, bottom graph) is almost identical, up to some scaling factor, to that in the neighboring segments. Although not shown, the same applies for the other eight neurons. This encourages us to consider each 40-bin segment separately from preceding segments.

The firing rate $r(t)$ can thus be modeled as

$$\begin{aligned} r(t) &= g(\boldsymbol{z}(t-1)) + \epsilon(t) \\ \boldsymbol{z}(t-1) &= (s_1(t-1), s_2(t-1), \theta_s(t-1)), \end{aligned} \tag{4.7}$$

where $\theta_s(t-1)$ is the number of time points (up to $t-1$) holding the same stimulus value $(s_1(t-1), s_2(t-1))$. We scale $\theta_s(t-1)$ so that it lies into the interval $(-1, 1)$ (like $s_1(t-1)$ and $s_2(t-1)$).

Locality is now applied on the input space rather than on the time dimension. The weights w_n, defined in Equation (4.6), are now defined as

$$w_n = K_\tau(||\boldsymbol{z}(t) - \boldsymbol{z}(n)||_2) = \begin{cases} (1 - ||\boldsymbol{z}(t) - \boldsymbol{z}(n)||_2^3)^3 & \text{if } ||\boldsymbol{z}(t) - \boldsymbol{z}(n)||_2 \leq \tau \\ 0 & \text{otherwise,} \end{cases}$$

where τ is the bandwidth parameter.

The bilinear and fullrank models, MARS, SpAM, and their local versions can be applied to a data set built according to Equation (4.7). However, stationary models are difficult to use here. If we train a model on a fixed part of the averaged spike counts

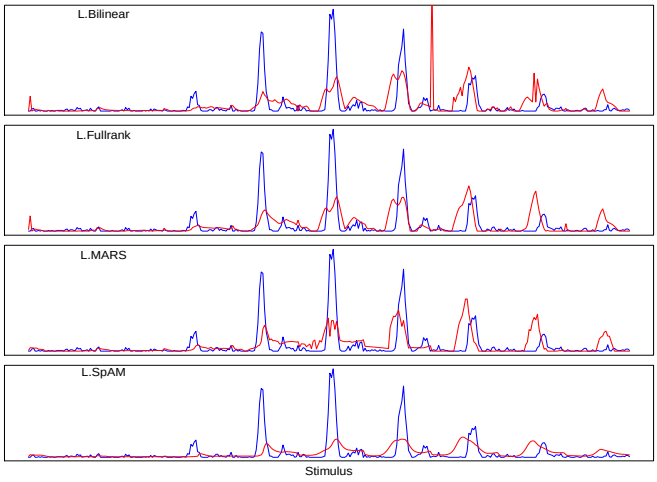


Figure 4.18: Predicted signal (red) and true signal (blue) for neuron t18 from primary visual cortex (area 17) data.

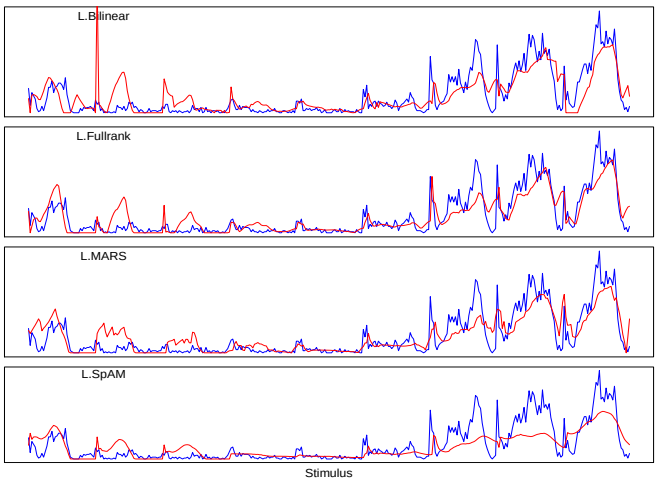


Figure 4.19: Predicted signal (red) and true signal (blue) for neuron t27 from primary visual cortex (area 17) data.

data, only a subset of the possible stimulus values is used to build the model. Since none of the stimulus values are the same in the testing part, extrapolation is unlikely to work, and the prediction will be highly unstable. Therefore, we only consider here non-stationary (local) models. We have used the $N = 120$ closest responses (in the stimulus space) to build the models. Spikes have been estimated for $t = N + 1, ..., T$.

Figure 4.18 and Figure 4.19 illustrate the estimated spike counts against the true estimated spike counts for two neurons, t18 and t27, respectively. It appears that L.MARS and L.fullrank models offer the best fit, whereas L.bilinear is slightly worse and L.SpAM (run with minimal regularization) is smoother than the others. L.MARS probably performs better than L.SpAM because it takes into account interactions among the inputs. The mean and standard deviation of the predictive power for the L.bilinear and L.fullrank models, L.MARS and L.SpAM are, respectively, $0.08(\pm 0.41)$, $0.11(\pm 0.26)$, $0.16(\pm 0.43)$ and $0.06(\pm 0.22)$.

To evaluate the models, we use the Kolmogorov-Smirnov test based on the time-rescaling theorem (Brown et al., 2001). We also evaluate the models obtained by the LNP approach, which includes spike history terms (Truccolo et al., 2005). In short, we compute rescaled times

$$v_a = 1 - exp\left[-\int_{u_a}^{u_{a+1}} \frac{\hat{r}(t)}{\Delta} dt\right],$$

where $u_1 < ... < u_a < ... < u_A$ denote the set of individual spike times and Δ is the bin length. Note that $\hat{r}(t)/\Delta$ is just the estimated conditional intensity of the point process at time t, which is directly estimated by the LNP model. It can be shown that the v_a values are independent uniformly distributed random variables if and only if the estimated response $\hat{r}(t)$ corresponds to the true conditional distribution of the process. Hence, to perform a usual Kolmogorov-Smirnov test, we just need to order the v_a values from the smallest one to the largest one and check if they are uniformly distributed. We denote the ordered values as v_{a^*}.

An intuitive way to evaluate the Kolmogorov-Smirnov test's result is the K-S plot, which plots the quantiles of the cumulative distribution function of the uniform distribution in the unit interval, given by $CDF_{a^*} = (a^* - 0.5)/A$ against the ordered values v_{a^*}, where A is the number of spike events. The resulting points should lie around a straight 45° line if the model correctly describes the data.

Figure 4.20 shows K-S plots for neurons t18, t27, to2 and t26. Each colored line represents a trial. The proposed models correctly describe most trials for neurons t18 and t27, whereas the LNP approach fails to describe neuron t18. On the other hand, there are other neurons, like to2 and t26, that appear to be harder to model using the introduced approaches. A few trials for neuron to2 are, however, described a bit better by the LNP approach, probably because of a strong dependence between spiking activity and spike history. Neither model is able to model t26.

We have also tested how robust are the incremental local learning approaches in this setting. Note that the validation procedure that we use in Section 4 is not directly applicable here, because, as commented above, we have conveniently reordered the trials. Here, we built the models on data corresponding to stimuli different from those we use for testing. In other words, we excluded from training data those data items that have the same stimulus than the test data item that is to be predicted. L was set to 80 and locality was time-based. Table 4.2 reports mean absolute error (and standard deviation) for each neuron and each incremental local learning approach. Pure local approaches overfit the stimulus and produce worse results.

4.6 DISCUSSION

In this chapter, we have studied several nonlinear models on a number of different cases of spike firing rate prediction, showing that regularization can play a key role. Although all spike generating processes are intrinsically nonlinear in the synthetic scenarios, the source of their nonlinearity is certainly different. Whereas the first two of the above generating processes can be approximated by a simple extension of the linear model, more complex models are required to describe the third generating process. The fourth and fifth generating processes intend to simulate a response whose underlying model varies across the time. This could account for the habituation of the subject to the stimulus or any other internal change in the subject. In this case, local and incremental local models that take into account this variation, even if they are relatively simple, definitely outperform other more complex stationary models.

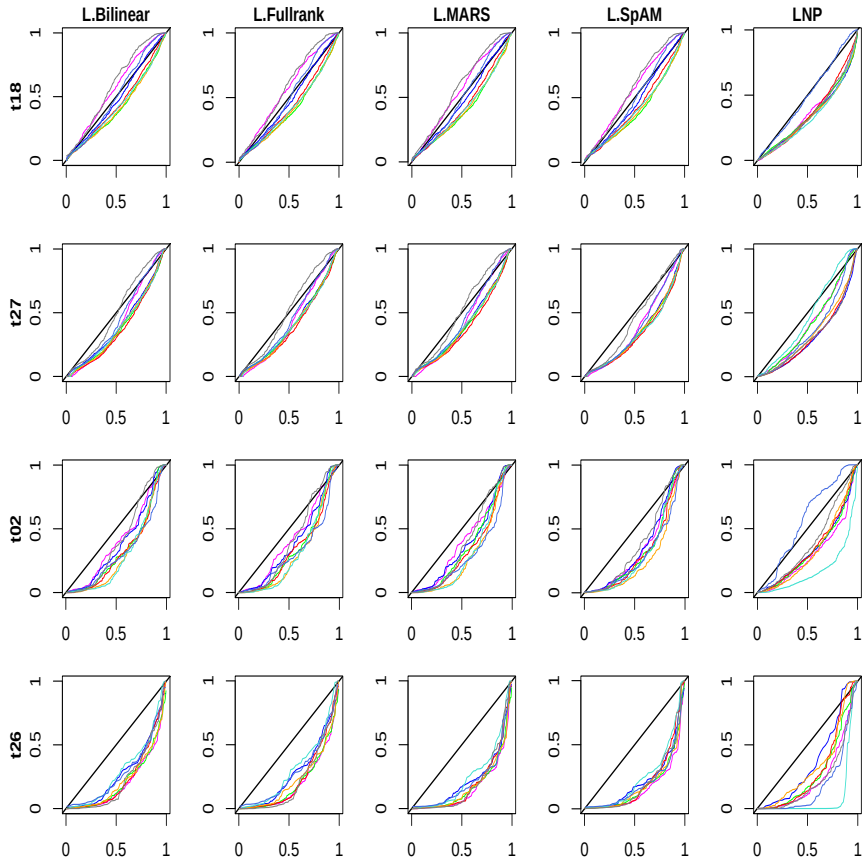


Figure 4.20: K-S plots for the L.bilinear and L.fullrank model, L.MARS, L.SpAM and LNP. Each trial is represented by a colored line. The black straight 45° line represents a perfect fit.

Due to the huge variability of neural processes, it is impossible to choose a level of complexity, a kind of nonlinear approach or a family of models that universally fit well for the neural firing rate prediction problem. Some preliminary analysis is needed to ascertain the best model for a specific problem. For example, we studied the response of some neurons in the cat primary visual cortex (area 17) to simple stimuli. The models presented above had to be refined somewhat to tackle this problem. We finally found that the response of these neurons is to some extent independent from previous stimuli, the current stimulus value and the exposure time being the key inputs. In addition, the subject follows different patterns of response for different stimulus values. These patterns, however, are alike for close stimulus values. For this reason, locality plays a fundamental role in the prediction.

Of the studied models, bilinear, fullrank and SpAM are additive, i.e., they do not consider interactions among different stimuli. Therefore, they appear to deal with nonlinearity only at the earliest (dendritic) stages of neuron processing. Other models that apply nonlinearity on the output of a linear model intend to capture the processes at the latest stage of the neural process (spike generation). Models in the literature typically follow one of these approaches. However, to fully understand the encoding properties of a neuron, it may be necessary to consider interactions among different stimuli. We believe that they could be the basis of the intermediate processing stages

Table 4.2: Mean absolute error (and standard deviation) for each neuron and each local and incremental local learning approach. The best result for each row is highlighted in bold.

	Neuron				
Method	t00	t02	t04	t08	t10
L.Bilinear	0.56(±0.2)	1.61(±0.9)	0.98(±1.4)	0.61(±0.9)	0.84(±1.1)
L.Fullrank	0.72(±0.9)	1.97(±1.1)	1.51(±1.6)	0.73(±1.0)	1.12(±1.1)
L.MARS	0.58(±0.9)	0.71(±1.0)	1.06(±1.4)	0.73(±1.0)	0.85(±1.1)
L.SpAM	0.27(±0.2)	0.59(±0.7)	1.03(±1.4)	0.46(±0.6)	0.73(±1.0)
IL.Bilinear	0.50(±0.5)	1.01(±1.1)	0.90(±1.4)	**0.35**(±0.6)	1.15(±1.3)
IL.Fullrank	0.29(±0.3)	1.01(±1.2)	1.07(±1.3)	0.47(±0.6)	0.81(±1.0)
IL.MARS	0.32(±0.3)	**0.56**(±0.9)	**0.89**(±1.3)	0.58(±0.8)	**0.62**(±1.0)
IL.SpAM	**0.26**(±0.2)	**0.56**(±0.8)	1.02(±1.4)	0.45(±0.6)	0.76(±1.0)

Method	t18	t23	t25	t26	t27
L.Bilinear	1.22(±2.4)	1.93(±0.8)	0.19(±0.2)	0.43(±0.7)	2.85(±2.0)
L.Fullrank	1.53(±2.5)	1.35(±0.9)	0.29(±0.6)	0.95(±1.1)	2.12(±1.8)
L.MARS	1.26(±2.3)	1.09(±0.8)	0.25(±0.7)	0.52(±0.9)	2.74(±2.0)
L.SpAM	1.27(±2.4)	1.87(±0.7)	0.09(±0.1)	0.40(±0.6)	2.28(±2.1)
IL.Bilinear	1.29(±2.5)	1.30(±0.9)	0.22(±0.4)	0.65(±1.1)	1.76(±2.1)
IL.Fullrank	1.38(±2.4)	1.03(±0.8)	0.10(±0.1)	0.42(±0.7)	**1.67**(±1.7)
IL.MARS	1.27(±2.2)	1.03(±0.8)	**0.07**(±0.1)	**0.39**(±0.7)	2.03(±1.9)
IL.SpAM	**1.17**(±2.4)	**0.91**(±0.7)	0.09(±0.1)	0.40(±0.7)	2.09(±2.1)

of the neuron. MARS does consider interactions among stimuli and has output the best results in the real data experiments.

There exist other models in the statistics field to deal with interactions among the inputs. For example, the Component Selection and Smoothing Operator (COSSO) (Lin and Zhang, 2006) is a method based on L_1-regularization for simultaneous function selection and smoothing. COSSO is defined in the context of smoothing spline ANOVA (Wahba, 1990), where the estimated function potentially includes interactions of any order among the inputs. However, we have found that COSSO does not perform well for the neural spike count prediction task. Unlike MARS and SpAM, COSSO is not designed for high-dimensional problems. Although smoothing spline ANOVA can capture more complex relations, it tends, in this particular case, to overfit in spite of regularization, and the resulting models are poorer than for MARS and SpAM.

Finally, note that local bilinear and local fullrank models can be adapted to single trials, following the guidelines introduced by Ahrens et al. (2008). The extension of MARS and SpAM to this setting is a more complex issue.

Part III

REGULARIZATION FOR SUPERVISED CLASSIFICATION

This part deals with the use of regularization for supervised classification. As in the regression framework, the main objective of regularization is to improve the bias-variance trade-off of the estimation and to permit the estimation in an ill-posed situation. Also, regularization can yield variable selection and other problem-specific objectives.

Chapter 5 describes a novel methodology for introducing non-penalized regularization in the estimation of a naïve Bayes model (Vidaurre et al., 2011b). Chapter 6 introduces a regularized model based on the naïve Bayes principles that discards redundant predictors, either continuous or discrete (Vidaurre et al., 2011c). Finally, Chapter 7 presents a signal classification framework that can be used for brain-computer interface design (Vidaurre et al., 2011a).

5

FORWARD STAGEWISE NAÏVE BAYES

5.1 INTRODUCTION

Bayesian network classifiers (Friedman et al., 1997) are a popular supervised classification paradigm. A well-known Bayesian network classifier is the *naïve Bayes* (Minsky, 1961), a simple Bayesian network classifier that assumes that the predictors or variables are independent given each class value. Despite its simplicity and strong assumptions, the naïve Bayes classifier has been proven to work satisfactorily in many domains (Domingos and Pazzani, 1997; Hand and Yu, 2001). Typically, the parameters of the naïve Bayes model are found by maximizing the joint likelihood of the model.

The naïve Bayes model's accuracy, however, declines in the presence of noisy predictors. A noisy predictor can be a predictor that either carries no useful information for the classification (irrelevant) or is strongly dependent on another predictor (redundant). Redundancy is particularly harmful, because the predictor information has double the influence than it should.

For variable selection purposes, it is common to use filtering approaches, which perform variable selection disregarding the classifier, or (greedy) wrapper algorithms, which simultaneously introduce variables into the model and iteratively estimate the parameters. We focus on the wrapper paradigm. The (stepwise) selective naïve Bayes (Langley and Sage, 1994) is a popular example of greedy wrapper algorithm.

Regularization techniques introduce additional information, usually to solve an ill-posed problem or to avoid overfitting. Also, by imposing certain restrictions, regularization trades off a little bias against a larger reduction in variance. An example of regularization within the naïve Bayes model is the L_1/L_2-regularized naïve Bayes, taken by van Gerven and Heskes (2008), which applies optimization techniques to minimize the negative log-likelihood function of the data given the model plus an L_1/L_2-group penalty on the model complexity. This penalty encourages some predictors to be discarded. Whereas they apply this idea only to the continuous predictor case, we extend it to deal with discrete predictors. Also, we introduce an adaptive penalty (Zou, 2006) that further improves the method's performance.

The main contribution discussed in this chapter, however, is a stagewise version of the selective naïve Bayes that is particularly useful when there are predictors that are relevant but, to some extent, redundant. At each iteration, instead of adding an "entire" predictor to the model, the parameters of the selected predictor are updated just a little. This method is inspired by the forward stagewise selection method for linear regression, which is also related to boosting and can be considered a form of regularization. We call this method *forward stagewise naïve Bayes*. This chapter is based on the published paper (Vidaurre et al., 2011b).

Let $\{X_1, ..., X_p\}$ be the set of p predictors and Y the class variable. The labeled data set, containing N instances, is denoted as $\boldsymbol{D} = \{(x_{n1}, \ldots, x_{np}, y_n), n = 1, ..., N\}$. $\boldsymbol{X}$ denotes the $N \times p$ predictor data matrix, with elements x_{ni}, $n \in \{1, ..., N\}$, $i \in \{1, ..., p\}$, and $\boldsymbol{y} = (y_1, ..., y_N)^T$ denotes the vector of responses. We assume that the class variable, Y, may take values $j \in \{1, ..., J\}$. The objective is to learn a classifier from $\boldsymbol{D}$ so as to predict the class value for incoming data points just given by predictor values.

We assume that predictors are either discrete or continuous, although generalizations for combining the two are extremely straightforward.

When the inputs are discrete, we assume that each predictor X_i has M_i possible states. Assuming that the predictors are conditionally independent given the class variable, we denote their *conditional probability table* (CPT) as an $M_i \times J$ matrix $\boldsymbol{\Theta}_i$. Each element θ_{ikj} of $\boldsymbol{\Theta}_i$, $j \in \{1, ..., J\}$, $k \in \{1, ..., M_i\}$, is the probability of the predictor X_i taking its k-th state given the j-th class variable state, i.e. $\theta_{ikj} = P(X_i = k|Y = j; \boldsymbol{\Theta}_i)$.

We assume that, when the inputs are continuous, predictors follow a Gaussian distribution within each class value. We denote as $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i$ the vectors whose elements are, for each state of Y, the expectation and standard deviation of X_i, respectively, i.e., $X_i|Y = j \sim \mathcal{N}(\mu_{ij}, \sigma_{ij})$, $j \in \{1, ..., J\}$. We denote the conditional density function for predictor X_i, given that $Y = j$, as $f(x_i|j; \mu_{ij}, \sigma_{ij})$.

Let $\boldsymbol{\Theta} = \{\boldsymbol{\Theta}_1, ..., \boldsymbol{\Theta}_p\}$, $\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_p\}$ and $\boldsymbol{\sigma} = \{\boldsymbol{\sigma}_1, ..., \boldsymbol{\sigma}_p\}$. Likewise, we denote the whole set of predictor parameters as $\boldsymbol{\Omega} = \{\boldsymbol{\Omega}_1, ..., \boldsymbol{\Omega}_p\}$, where $\boldsymbol{\Omega}_i$ generically denotes either $\boldsymbol{\Theta}_i$ or $\{\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i\}$. Also, we denote class prior probabilities as $\boldsymbol{\pi} = (\pi_1, ..., \pi_J)$. Considering the predictors to be conditionally independent given the class, the full likelihood function for the *naïve Bayes* (NB) (Minsky, 1961) model is defined as

$$L(\boldsymbol{D}; \boldsymbol{\Omega}) = \prod_{n=1}^{N} \Big[\pi_{y_n} \prod_{i=1}^{p} \psi(X_i = x_{ni}|Y = y_n, \boldsymbol{\Omega}_i) \Big], \tag{5.1}$$

where function $\psi(\cdot)$ computes the contribution of each predictor to the full likelihood. The likelihood is thus decomposable and can be computed separately for each predictor. We now define the contribution of each predictor to the full likelihood.

Let $\boldsymbol{W}(i)$ be an $N \times M_i$ indicator matrix for discrete predictor X_i. For the n-th instance, the elements of the indicator matrix are defined as $w(i)_{nk} = 1$ if $x_{ni} = k$ and $w(i)_{nk} = 0$ if $x_{ni} \neq k$. Similarly, $\boldsymbol{S}$ is defined as the $N \times J$ indicator matrix for class variable Y. Hence, the contribution of a discrete predictor X_i and instance n to the full likelihood is

$$\psi(X_i = x_{ni}|Y = y_n, \boldsymbol{\Omega}_i) = P(X_i = x_{ni}|Y = y_n, \boldsymbol{\Theta}_i) = \boldsymbol{w}(i)_{n\cdot} \boldsymbol{\Theta}_i \boldsymbol{s}_{n\cdot}^T, \tag{5.2}$$

where $\boldsymbol{w}(i)_{n\cdot}$ is the n-th row vector of $\boldsymbol{W}(i)$ and $\boldsymbol{s}_{n\cdot}$ is the n-th row vector of $\boldsymbol{S}$. Hence, $\boldsymbol{w}(i)_{n\cdot}$ and $\boldsymbol{s}_{n\cdot}$ are selecting the appropriate conditional probability for the n-th instance from $\boldsymbol{\Theta}_i$.

On the other hand, the contribution of a continuous predictor X_i and instance n to the full likelihood is defined as

$$\psi(X_i = x_{ni}|Y = y_n, \boldsymbol{\Omega}_i) = f(x_{ni}|y_n; \mu_{iy_n}, \sigma_{iy_n}) = \frac{1}{\sqrt{2\pi}\sigma_{iy_n}} \exp - \frac{(x_{ni} - \mu_{iy_n})^2}{2\sigma_{iy_n}^2}. \tag{5.3}$$

Let $\mathbf{\Omega}_i^{(0)}$ be the parameters of predictor X_i such that they are exactly equal for all class values, that is, either $\theta_{ikj}^{(0)}$ in the discrete case or $\{\mu_{ij}^{(0)}, \sigma_{ij}^{(0)}\}$ in the continuous case are equal for all $j \in \{1, ..., J\}$. This is equivalent to removing predictor X_i from the model.

To estimate a NB model, we compute the maximum likelihood estimation (MLE) of the parameters, denoted as $\hat{\mathbf{\Theta}}_i^{(1)}$, $\hat{\boldsymbol{\mu}}_i^{(1)}$, $\hat{\sigma}_i^{(1)}$ and $\hat{\boldsymbol{\pi}}$, as

$$\begin{aligned} \hat{\theta}_{ikj}^{(1)} &= \frac{N_{ijk}}{N_j}, \\ \hat{\mu}_{ij}^{(1)} &= \frac{\sum_{n;y_n=j} x_{ni}}{N_j}, \\ \hat{\sigma}_{ij}^{(1)} &= \sqrt{\frac{\sum_{n;y_n=j}(x_{ni} - \hat{\mu}_{ij})^2}{N_j}}, \\ \hat{\pi}_j &= \frac{N_j}{N}, \end{aligned} \tag{5.4}$$

where N_{ijk} is the number of instances in the training data set, where predictor X_i takes the value k and Y takes the value j, and N_j is the number of instances where Y takes the value j.

The NB formulation for the probability of the class given the (continuous or discrete) predictors is

$$\begin{aligned} P(Y = j | X_1 = k_1, ..., X_p = k_p, \hat{\mathbf{\Omega}}^{(1)}, \hat{\boldsymbol{\pi}}) \propto \\ \hat{\pi}_j \prod_{i=1}^{p} \psi(X_i = k_i | Y = j, \hat{\mathbf{\Omega}}_i^{(1)}) = \phi_j. \end{aligned} \tag{5.5}$$

Thus, given vector $\boldsymbol{\phi} = (\phi_1, ..., \phi_J)$, whose components are computed with Equation (5.5), the actual classification is performed by

$$\hat{j} = \text{maxpos}(\boldsymbol{\phi}), \tag{5.6}$$

where $\text{maxpos}(\cdot)$ returns the position of the maximum element of the vector argument. Ties can be broken at random. Note that, although $\boldsymbol{\phi}$ depends on the input data configuration, it is omitted from the notation for simplicity's sake.

5.3 METHODS RELATED TO NAÏVE BAYES

In this section, we introduce some existing methods related to NB: the selective naïve Bayes (Langley and Sage, 1994), the weighted naïve Bayes (Ferreira et al., 2001) and the L_1/L_2-regularized naïve Bayes (van Gerven and Heskes, 2008). Also, we generalize the L_1/L_2-regularized naïve Bayes to handle both discrete and continuous predictors and propose a simple improvement on this method.

5.3.1 *Existing methods*

The *selective naïve Bayes* (SNB) model (Langley and Sage, 1994) is a popular greedy, wrapper, stepwise algorithm for obtaining a NB model and performing variable selection. The SNB approach obeys Equation (5.5) and, hence, makes use of the MLE.

However, it is applied over only a subset of predictors. A forward greedy search finds this subset of predictors, where predictors are included in the model as long as the prediction accuracy (over training data) keeps increasing. Langley and Sage (1994) also introduce a backwards search strategy, but they conclude that forward search is often more advantageous. On this ground, we use forward search.

The *weighted naïve Bayes* (WNB) model (Ferreira et al., 2001) includes all the predictors, which it weights according to their relevance for the classification. It is conceived only for discrete predictors. Weights are computed as

$$w_i = \sqrt{\sum_{j=1}^{J}\sum_{k=1}^{M_i}\Big[P(Y=j|X_i=k) - P(Y=j)\Big]^2},$$

so that the resulting model is

$$P(Y=j|X_1=k_1,...,X_p=k_p,\boldsymbol{\Omega}) \propto \hat{\pi}_j \prod_{i=1}^{p} \psi(X_i=k_i|Y=j,\boldsymbol{\Omega}_i)^{w_i} = \phi_j.$$

The classification rule is the same as for NB (Equation (5.6)).

Using regularization techniques, the L_1/L_2-*regularized naïve Bayes* approach (L_1/L_2-NB) (van Gerven and Heskes, 2008), designed for continuous predictors, is formulated as the optimization problem

$$\begin{aligned} \text{argmin}_{\boldsymbol{\mu},\boldsymbol{\sigma}} \ & -\log L(\boldsymbol{D};\boldsymbol{\mu},\boldsymbol{\sigma}) + \lambda \sum_{i=1}^{p} \sqrt{\sum_{j=1}^{J}(\mu_{ij}-\hat{\mu}_{ij}^{(0)})^2 + \sum_{j=1}^{J}(\sigma_{ij}-\hat{\sigma}_{ij}^{(0)})^2}, \\ \text{s.t.} \quad & -\sigma_{ij} < 0 \qquad \forall i,j, \end{aligned} \tag{5.7}$$

where $L(\boldsymbol{D};\boldsymbol{\mu},\boldsymbol{\sigma})$ is defined in Equations (5.1) and (5.3) and λ is some regularization parameter. This optimization problem has Jp inequality constraints.

This way, the set of parameters of each single predictor (inside the square root) forms a group. This penalty is hence a group lasso-type penalty or L_1/L_2-penalty (Yuan and Lin, 2006), which is able to discard entire groups. Therefore, all the parameters $\{\mu_{ij},\sigma_{ij}\}$ of some predictors will be prompted to be equal to $\{\hat{\mu}_{ij}^{(0)},\hat{\sigma}_{ij}^{(0)}\}$, so that such predictors will be effectively excluded.

Note that this optimization problem is convex. First, it is well known that the Gaussian likelihood defined in Equation (5.3) is log-concave and hence the negative log-likelihood is convex (Boyd and Vandenberghe, 2004). This can be easily proved by taking the Hessian, which is positive semidefinite and thus proves convexity. Second, the L_1/L_2-penalty defined in (5.7) is just a sum of L_2-penalties. Since the L_2-norm function is convex, it is the sum of L_2-norms. The sum of two convex functions is convex. Finally, the inequality constraint functions are just nonnegativity constraints. Therefore, problem (5.7) is convex and is, in fact, denoted in standard form. Although the entire objective function is non-smooth (non-differentiable), it is composed of a smooth loss function and a block-separable penalty and, hence, the problem can be solved by unconstrained (block) coordinate gradient descent optimization (Tseng, 2001). The constraint can be subsumed into the penalty term by setting it to ∞ when $\sigma_{ij} < 0$ for any pair (i,j).

5.3.2 *Generalized L_1 / L_2-regularized naïve Bayes*

Now, we extend the L_1/L_2-NB formulation to deal with the discrete predictor case and propose an adaptive formulation of the problem for achieving better predictions.

We formulate the optimization problem for discrete predictors as

$$\begin{aligned} \operatorname{argmin}_{\boldsymbol{\Theta}} \quad & -\log L(\boldsymbol{D};\boldsymbol{\Theta}) + \lambda \sum_{i=1}^{p} \sqrt{\sum_{j=1}^{J}\sum_{k=1}^{M_i}(\theta_{ikj} - \hat{\theta}_{ijk}^{(0)})^2}, \\ \text{s.t.} \quad & \mathbf{1}_i^t \boldsymbol{\theta}_{ij} - 1 = 0 \qquad \forall i, j, \\ & -\theta_{ikj} < 0 \qquad \forall i, j, k, \\ & \theta_{ikj} - 1 < 0 \qquad \forall i, j, k, \end{aligned} \tag{5.8}$$

where the loss function $L(\boldsymbol{D};\boldsymbol{\Theta})$ is defined in Equations (5.1) and (5.2), $\boldsymbol{\theta}_{ij}$ is the j-th column of $\boldsymbol{\Theta}_i$ and $\mathbf{1}_i$ is a column vector with M_i ones. Therefore, there are Jp equality and $\sum_{i=1}^{p} JM_i$ inequality constraints (each pair of inequality constraints can be subsumed in one open box constraint $\theta_{ikj} \in (0,1)$).

This problem is also convex and is denoted in standard form. Since the expression in Equation (5.2) is linear on $\boldsymbol{\Theta}$, it is clear that the negative log-likelihood is convex and differentiable. The penalty in the loss function is also convex (but non-differentiable), and thus the entire loss function is convex. Both the equality and inequality constraint functions are affine. Even if we mixed both continuous and discrete predictors, the problem would still be convex. However, with the equality constraints, we cannot follow the gradient descent direction, so that (block) coordinate gradient descent optimization is not directly applicable. Instead, we take a simple approximation: starting with initial values $\Theta_i^{(0)}$, $i = 1, ..., p$, we update Θ_i towards $\Theta_i^{(1)}$ at each iteration, while the others predictors are held fixed, until the objective function in Equation 5.8 reaches a minimum. This is a just a line-search.

A possible improvement on this approach is to use an adaptive penalty, which will hopefully improve the accuracy of the estimator. In L_1-penalized linear regression (Tibshirani, 1996), for example, such penalties reduce the bias and lead to a consistent estimation (Zou, 2006). The innovation is to penalize each predictor variable according to its importance. Each variable penalty is thus scaled by $1/|\beta_i^{(1)}|$, where $\beta_i^{(1)}$ is (in the $N > p$ case) the ordinary least squares regression coefficient or MLE. Note that $|\beta_i^{(1)}|$ is just the absolute upper bound of this coefficient in the regularized problem, i.e., the upper bound of the penalty for this variable.

We can apply an analogous idea to the L_1/L_2-NB formulation by computing weights $\boldsymbol{w} = (w_1, ..., w_p)$, for the discrete and continuous predictor cases, respectively, as

$$w_i = \sqrt{\sum_{j=1}^{J}\sum_{k=1}^{M_i}(\hat{\theta}_{ijk}^{(1)} - \hat{\theta}_{ijk}^{(0)})^2},$$

$$w_i = \sqrt{\sum_{j=1}^{J}(\hat{\mu_{ij}}^{(1)} - \hat{\mu_{ij}}^{(0)})^2 + \sum_{j=1}^{J}(\hat{\sigma_{ij}}^{(1)} - \hat{\sigma_{ij}}^{(0)})^2},$$

so that loss functions in (5.8) and (5.7) become, respectively,

$$-\log L(\boldsymbol{D};\boldsymbol{\mu},\boldsymbol{\sigma}) + \lambda \sum_{i=1}^{p} w_i \sqrt{\sum_{j=1}^{J}\sum_{k=1}^{M_i}(\theta_{ikj} - \hat{\theta}_{ijk}^{(0)})^2},$$

$$-\log L(\boldsymbol{D};\boldsymbol{\Theta}) + \lambda \sum_{i=1}^{p} w_i \sqrt{\sum_{j=1}^{J}(\mu_{ij} - \hat{\mu}_{ij}^{(0)})^2 + \sum_{j=1}^{J}(\sigma_{ij} - \hat{\sigma}_{ij}^{(0)})^2}.$$

Note that each w_i is a tight upper bound of the penalty for predictor X_i, and here we have the parallelism with the adaptive penalty for linear regression. We call this approach *adaptive L_1-regularized naïve Bayes* (aL_1/L_2-NB).

5.4 NOISY PREDICTORS

In this section, we define irrelevance and redundancy and remark on some ideas that motivate the approach introduced below. We show that it is sometimes beneficial to use a point of compromise between $\hat{\boldsymbol{\Omega}}^{(0)}$ and $\hat{\boldsymbol{\Omega}}^{(1)}$ instead of the MLE like SNB does. Also, we discuss why the L_1/L_2-NB approach (including the adaptive version) can discard only irrelevant predictors and not redundant predictors.

First, we define the redundancy and irrelevance concepts and briefly discuss their effect on the NB model. We define a predictor as *noisy* if it is irrelevant for the class variable or is redundant to another predictor. Similar definitions of irrelevance and redundancy can be found, for example, in (Kohavi and John, 1996; Langley and Sage, 1994).

A discrete predictor X_i is *irrelevant* for Y if

$$P(Y = j|X_i = k) = P(Y = j), \quad \forall k \in \{1, ..., M_i\}, \forall j \in \{1, ..., J\},$$

so that the value of X_i does not give any information about the value of Y. Equivalently, we can say that the within-class parameters of predictor X_i are equal for all class values. The definition for a continuous predictor is analogous.

On the other hand, two predictors X_{i_1} and X_{i_2} are *redundant* when there is a dependency between them. Let $H(\cdot)$ represent the entropy function. Two predictors X_{i_1} and X_{i_2} are fully redundant when

$$H(X_{i_1}|X_{i_2}) = H(X_{i_2}|X_{i_1}) = 0. \tag{5.9}$$

On the other hand, they are completely independent when

$$H(X_{i_1}|X_{i_2}) = H(X_{i_1}), \quad H(X_{i_2}|X_{i_1}) = H(X_{i_2}).$$

Note that these conditions are just extremes of a continuum. In real-world data, predictors are rarely fully redundant or completely independent. Instead, they typically are somewhere between these two extreme conditions.

When $N \to \infty$ and p is finite (i.e., the complete information case), irrelevant variables do not increment the expected error of a NB classifier because $\hat{\boldsymbol{\Omega}}^{(1)} = \hat{\boldsymbol{\Omega}}^{(0)}$ holds exactly. In the realistic case, when N is finite, we only have $\hat{\boldsymbol{\Omega}}^{(1)} \simeq \hat{\boldsymbol{\Omega}}^{(0)}$. In the presence of many irrelevant predictors, these small differences accumulate and can finally bias the actual decision and degrade the classification accuracy.

It is well-known, however, that, for NB, redundant predictors have a more harmful effect than irrelevant predictors (Drugan and Wiering, 2010). Figure 5.1 shows testing

errors of NB models obtained from three different types of discrete synthetic data sets. The first type has three non-noisy predictors, $\{X_1, X_2, X_3\}$, that are generated from the following probabilities

$$\boldsymbol{\Theta}_1 = \begin{pmatrix} 0.80 & 0.33 & 0.33 \\ 0.10 & 0.33 & 0.33 \\ 0.10 & 0.33 & 0.33 \end{pmatrix}, \quad \boldsymbol{\Theta}_2 = \boldsymbol{\Theta}_3 = \begin{pmatrix} 0.33 & 0.30 & 0.30 \\ 0.33 & 0.10 & 0.60 \\ 0.33 & 0.60 & 0.10 \end{pmatrix}, \tag{5.10}$$

so that predictor X_1 discriminates between the first and the other two class values, and predictors X_2 and X_3 mainly discriminate between the second and third class values; $\boldsymbol{\pi}$ is defined as being equal for all three class values. The other two types have, in addition, 50 irrelevant discrete predictors and 50 (fully) redundant discrete predictors, respectively. The class can take three values, each with the same frequency. We have conducted 100 experiments, generating training data sets with $N = 1000$ instances and test data sets with $N_{te} = 3000$ instances. Notice that both kinds of noisy predictors, but especially the redundant ones, decrease accuracy.

Using the same data, Figure 5.2 illustrates, for one experiment, the evolution of the testing error for an increasing number predictors. The X-axis represents the number of predictors in the model. The first three added predictors (leftmost part of the graphs) are relevant, and the others, up to 50, are irrelevant (lefthand graph) or redundant (righthand graph). Predictors are redundant with regard to the first non-noisy predictor. The solid line represents the error computed on the complete testing data set, whereas the other lines represent the error for each of the three class values. The black line represents the mean of the other three lines. We find that the class value that is best discriminated by the first predictor (short-dashed line) decreases the error in the presence of redundant predictors, but the other class values are no longer distinguishable. Irrelevant predictors, on the other hand, produce a more uniform and moderate increment of the error.

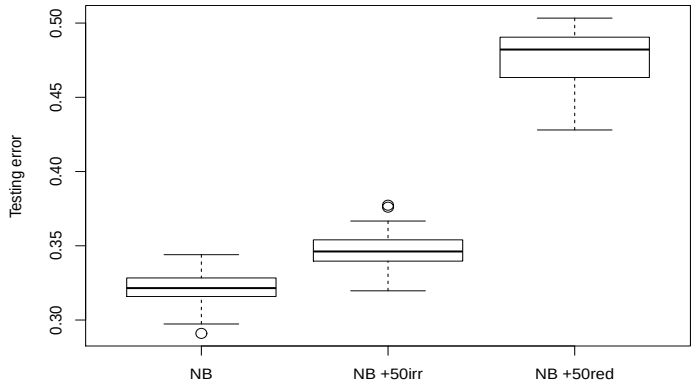


Figure 5.1: Boxplots for the testing errors of NB without noisy variables (left), NB with 50 irrelevant predictors and NB with 50 redundant predictors.

Ideally, SNB only adds predictors that reduce the classification error to the model. Hence, it will discard both redundant and irrelevant predictors, and retain those variables that are relevant but not redundant. However, as mentioned above, relevance and redundancy are not absolute concepts. What will SNB do with a set of relevant but non-fully redundant predictors? Let us suppose that there are two predictors, X_{i_1} and X_{i_2}, that are (non-fully) redundant, and each carries valuable information. Here,

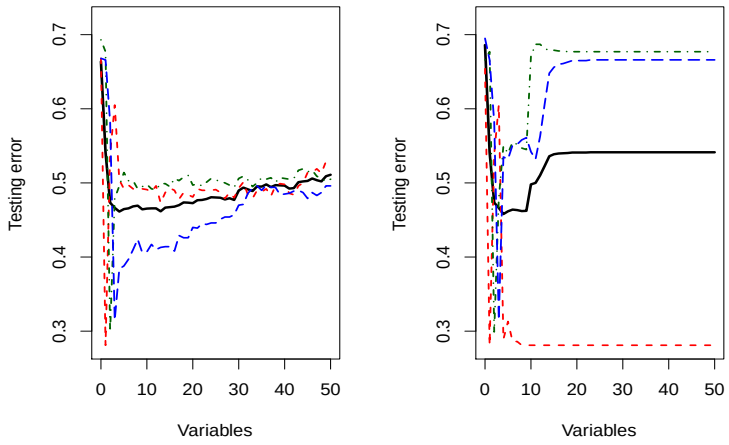


Figure 5.2: Evolution of the testing error for an increasing number of irrelevant (left) and redundant (right) predictors. The first three predictors are non-noisy.

we claim that a NB model that balances the contribution of these predictors may be better than a classic NB model that either excludes or fully includes them, like SNB does.

We use an example to illustrate this point. Let us first define

$$\begin{aligned}\boldsymbol{\Theta}_i^{(\alpha_i)} &= \alpha_i \boldsymbol{\Theta}_i^{(1)} + (1-\alpha_i)\boldsymbol{\Theta}_i^{(0)},\\ \boldsymbol{\mu}_i^{(\alpha_i)} &= \alpha_i \boldsymbol{\mu}_i^{(1)} + (1-\alpha_i)\boldsymbol{\mu}_i^{(0)},\\ \boldsymbol{\sigma}_i^{(\alpha_i)} &= \alpha_i \boldsymbol{\sigma}_i^{(1)} + (1-\alpha_i)\boldsymbol{\sigma}_i^{(0)},\end{aligned} \tag{5.11}$$

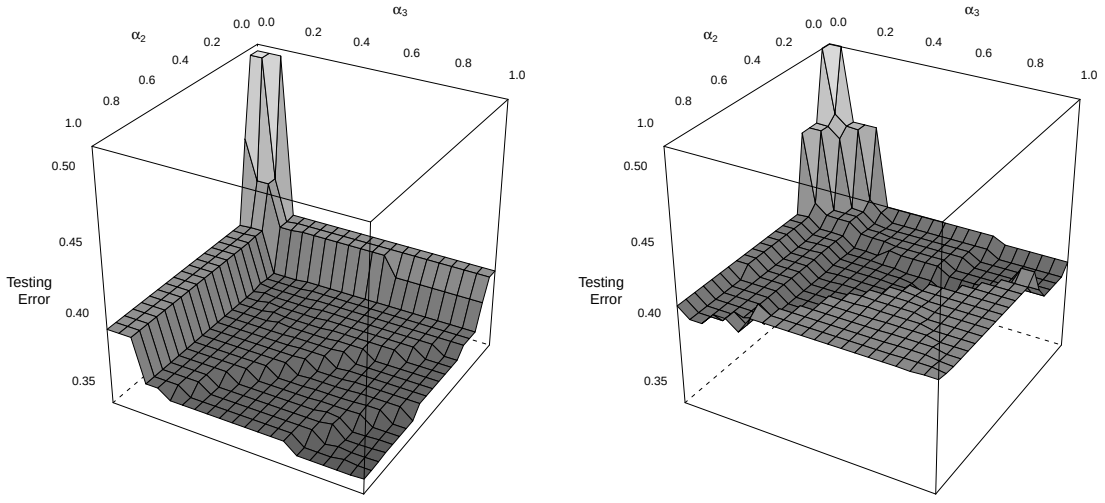
where $\alpha_i \in [0,1]$. Hence, $\hat{\boldsymbol{\Omega}}_i^{(\alpha_i)}$ is a linear combination of $\hat{\boldsymbol{\Omega}}_i^{(0)}$ and $\hat{\boldsymbol{\Omega}}_i^{(1)}$, where α_i refers to predictor X_i. Within this notation, we can say that SNB only considers values $\alpha_i \in \{0,1\}$ (exclusion or inclusion, respectively, of predictor X_i).

Now, we consider a training data set with $N = 1000$ instances and a testing data set with $N_{te} = 3000$ instances, with three predictors whose CPTs are given in (5.10). Now, we consider making X_2 and X_3 redundant by setting $x_{n2} = x_{n3}$ for some proportion of the data instances.

Let us consider NB models with parameters $\hat{\boldsymbol{\Theta}}_1^{(1)}$, $\hat{\boldsymbol{\Theta}}_2^{(\alpha_2)}$ and $\hat{\boldsymbol{\Theta}}_3^{(\alpha_3)}$. For a grid of values $\alpha_2, \alpha_3 \in [0,1]$, Figure 5.3(a) shows testing errors when X_2 and X_3 are not made redundant, that is, if we have not set $x_{n2} = x_{n3}$ at any time. Figure 5.3(b) shows testing errors when X_2 and X_3 are somewhat redundant, that is, after setting $x_{n2} = x_{n3}$ for some proportion of the data instances.

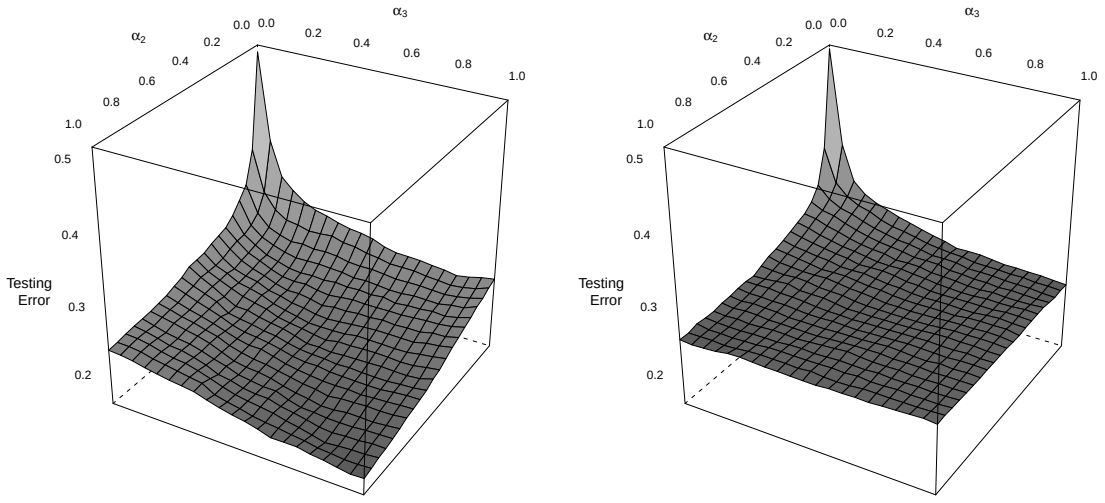
We find that, when X_2 and X_3 are independent, the minimum error is achieved when α_2, α_3 are equal to 1, i.e., when $\hat{\boldsymbol{\Theta}}_2 = \hat{\boldsymbol{\Theta}}_2^{(1)}$ and $\hat{\boldsymbol{\Theta}}_3 = \hat{\boldsymbol{\Theta}}_3^{(1)}$. On the other hand, when there is some dependence between X_2 and X_3, and X_1 is already part of the model, the best model is somewhere in $0 < \alpha_2, \alpha_3 < 1$.

Figure 5.4 illustrates the same scenario for continuous predictors. Figure 5.4(a) shows testing errors when X_2 and X_3 are independent, and Figure 5.4(b) shows testing errors when X_2 and X_3 are somewhat redundant. Although the effect is less obvious than in the discrete case, the conclusion is analogous.



(a) (b)

Figure 5.3: (a) Testing error when discrete predictors X_2 and X_3 are not made redundant, (b) testing error when predictors are somewhat redundant.



(a) (b)

Figure 5.4: (a) Testing error when continuous predictors X_2 and X_3 are not made redundant, (b) testing error when predictors are somewhat redundant.

These examples illustrate that it may be worthwhile finding a tradeoff between the MLE ($\hat{\mathbf{\Omega}}_i^{(1)}$) and the parameters that remove the predictor from the model ($\hat{\mathbf{\Omega}}_i^{(0)}$). This is the main motivation for proposing the forward stagewise naïve Bayes approach.

Finally, we note that, although L_1/L_2-NB is a natural choice for applying regularization to the NB model, it discards only irrelevant and not redundant predictors. It discards irrelevant predictors because, since $\hat{\mathbf{\Omega}}_i^{(0)}$ is not very different from $\hat{\mathbf{\Omega}}_i^{(1)}$ in this case, they make only a small contribution to the loss function in optimization problems (5.7) and (5.8). Note, however, that setting $\hat{\mathbf{\Omega}}_i = \hat{\mathbf{\Omega}}_i^{(0)}$ amounts to removing this predictor from the NB model, but it does not lead to the exclusion of the predictor from the loss function calculation in the optimization problem. In other words, accord-

ing to this formulation, all predictors participate in the loss function (Equation (5.1)), even when they can be simplified from the classification rule (Equations (5.5) and (5.6)). Therefore, if, for example, two predictors are fully redundant but separately relevant, the L_1/L_2-NB (or aL_1/L_2-NB) approach will add them both to the model, because, according to the log-likelihood formulation, both have a relevant impact on the loss function, no matter what the state of the other is. In other words, the inclusion of one predictor does not change the effect of the other on the loss function. In general terms, any algorithm that solves optimization problems (5.7) or (5.8) will select either both predictors or neither.

5.5 FORWARD STAGEWISE NAÏVE BAYES

We now introduce a more cautious version of the SNB approach, the *forward stagewise naïve Bayes* (fsNB). Like SNB, fsNB is a greedy algorithm but, instead of moving a set of parameters from $\hat{\mathbf{\Omega}}_i^{(0)}$ to $\hat{\mathbf{\Omega}}_i^{(1)}$ at each iteration, it takes small steps from $\hat{\mathbf{\Omega}}_i^{(\alpha_i)}$ to $\hat{\mathbf{\Omega}}_i^{(\alpha_i+\epsilon)}$, where $\epsilon > 0$ is some small constant and α_i determines the current parameters of predictor X_i (see Equation (5.11)). We can informally say that fsNB is to SNB what stagewise regression is to stepwise regression.

The rationale of this approach is to deal with the situation discussed above, i.e., when there are partially redundant variables that each carry separate information. By giving a balanced estimation of their parameters, we expect to retain the valuable information while minimizing the harmful effect of redundancy.

Concerning the greedy strategy, there is one important matter to address. At each iteration, we need to evaluate each predictor so as to decide which is going to be adjusted. There are two simple strategies for finding which predictor is most worth updating. Let us suppose that the parameters of predictor X_i are $\hat{\mathbf{\Omega}}_i^{(\alpha_i)}$. The first strategy is to evaluate predictor X_i by checking $\hat{\mathbf{\Omega}}_i^{(\alpha_i+\epsilon)}$. The second strategy is to check $\hat{\mathbf{\Omega}}_i^{(1)}$. Whatever we do, the predictor that leads to the greatest error decrement will be updated by ϵ (and the others are unchanged). Neither approach is problem free. In the first case, it is often not possible to decide how important a predictor is by just looking at some small increment ϵ. In the second case, if we look at the complete update of the parameters of the predictor, the contribution of other predictors with low $\alpha_{i'}$ ($i' \neq i$) could become negligible. Even when predictor X_i is important, the model accuracy may decrease considerably if the contribution of other important variables (almost) disappears.

Figure 5.5 illustrates this situation for two predictors, one relevant (left) and one irrelevant (right). It shows, at some early step of the algorithm, the evolution of the training and testing errors when we increase α_i for each predictor. Note that, in order to select the relevant rather than the irrelevant predictor, we have to look at a point between $\hat{\mathbf{\Omega}}^{(\alpha_i+\epsilon)}$ and $\hat{\mathbf{\Omega}}^{(1)}$, where the training (and testing error) is most decreased.

To do this, we consider some further steps ν at each iteration, i.e., we check the error for $\hat{\mathbf{\Omega}}_i^{\alpha_i+\epsilon}, \hat{\mathbf{\Omega}}_i^{\alpha_i+2\epsilon}, ..., \hat{\mathbf{\Omega}}_i^{\alpha_i+t\epsilon}, ..., \hat{\mathbf{\Omega}}_i^{\alpha_i+\nu\epsilon}$ for each predictor. This way, at each iteration, we select the optimal values $\{i, t\}$, and update the parameters accordingly. Parameters ϵ and ν define how detailed is the search at each step and may have an impact in the computational efficiency of the algorithm. Reasonable variations of them, however, does not greatly change the algorithm accuracy.

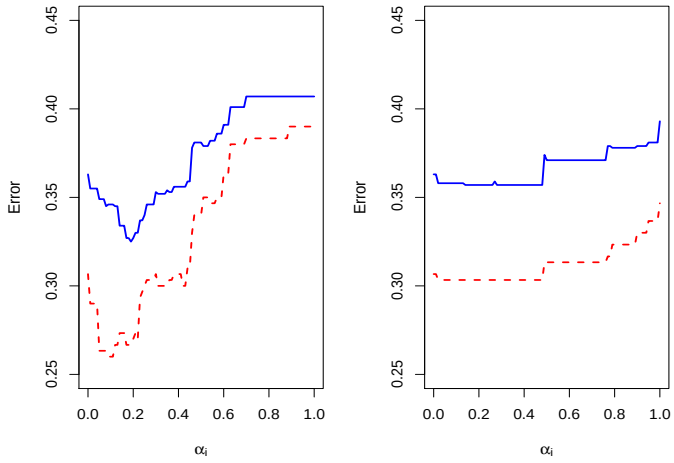


Figure 5.5: Training error (dashed line) and testing error (solid line) across the evolution of two variables, one relevant (left) and one irrelevant (right).

Algorithm 4 Forward stagewise naïve Bayes (fsNB)

Initialize $\alpha_i = 0$, $\forall i \in \{1, ..., p\}$, so that $\hat{\mathbf{\Omega}}_i^{(\alpha_i)} = \hat{\mathbf{\Omega}}_i^{(0)}$
while $\alpha_i \neq 1$, $\forall i \in \{1, ..., p\}$, **do**
  $error^* = \infty$
  for $i \in \{1, ..., p\}$ such that $\alpha_i \neq 1$ **do**
    for $t \in \{1, ...\nu\}$ **do**
      Compute $\hat{\mathbf{\Omega}}_i^+ = \hat{\mathbf{\Omega}}_i^{(\alpha_i + t\epsilon)}$
      $\phi_j^{(n)} = \pi_j \prod_{i'=1}^{p} \psi(X_{i'} = x_{ni'} | Y = j, \hat{\mathbf{\Omega}}_{i'}^+)$
      $error = 1/N \sum_{n=1}^{N} I\big(\text{maxpos}(\boldsymbol{\phi}^{(n)}), y_n\big)$
      if $error \leq error^*$ **then**
        $error^* = error$
        $i^* = i$
        $t^* = t$
      end if
    end for
  end for
  $\alpha_{i^*} = \alpha_{i^*} + t^*\epsilon$
end while

Algorithm 4 details the fsNB method in pseudocode format. The main part consists of two nested loops that look for the best pair $\{i, t\}$ at each iteration. Like SNB, the fitting criterion is the training error. The function $I(\cdot, \cdot)$ is an indicator function that outputs 1 if its arguments are equal and 0 otherwise.

To minimize the computational cost, we can stop the procedure early if the training error has not improved during a certain number of iterations. We have observed that the minimum testing error is very rarely found after the training error comes to a standstill, which makes this strategy promising.

Both the L_1/L_2-NB (using a grid of λ values) and the fsNB approaches generate a potentially large set of models, from which a final model needs to be selected. We can use a validation subset of the data set (if data are abundant), K-fold cross-validation, or some penalized criterion, which is typically the training loss plus some estimation of the optimism of the training loss rate. Here, we use the AIC statistic (Akaike, 1974):

$$AIC = Q\left(\boldsymbol{D}, \hat{\boldsymbol{\Omega}}\right) + \frac{2}{N} d,$$

where the loss function is the mean cross-entropy or deviance

$$Q\left(\boldsymbol{D}, \hat{\boldsymbol{\Omega}}\right) = \frac{1}{N} \sum_{n=1}^{N} -2 \log P(Y = y_n | X_1 = x_{n1}, ..., X_p = x_{np}, \hat{\boldsymbol{\Omega}}, \hat{\boldsymbol{\pi}}),$$

and d represents the degrees of freedom of the model, which we compute as

$$d = \sum_{i=1}^{p} I(\alpha_i > 0), \tag{5.12}$$

where $I(\cdot)$ outputs 1 if the argument is true and 0 otherwise. Since a NB model is linear and $0 < d \leq p$, this is a reasonable estimation.

For L_1/L_2-NB, a possible natural choice, instead of Equation (5.12), for computing d, in the discrete and continuous case, respectively, would be

$$d = \sum_{i=1}^{p} \frac{1}{JM_i} \sum_{j=1}^{J} \sum_{k=1}^{M_i} \frac{\theta_{ikj} - \hat{\theta}_{ijk}^{(0)}}{\hat{\theta}_{ijk}^{(1)} - \hat{\theta}_{ijk}^{(0)}},$$

$$d = \sum_{i=1}^{p} \frac{1}{J} \sum_{j=1}^{J} \left(\frac{\mu_{ij} - \hat{\mu}_{ij}^{(0)}}{\hat{\mu}_{ij}^{(1)} - \hat{\mu}_{ij}^{(0)}} + \frac{\sigma_{ij} - \hat{\sigma}_{ij}^{(0)}}{\hat{\sigma}_{ij}^{(1)} - \hat{\sigma}_{ij}^{(0)}} \right).$$

For fsNB, this would simplify to

$$d = \sum_{i=1}^{p} \alpha_i.$$

We have found, however, that the results are better using Equation (5.12). Therefore, in this work, we compute d using Equation (5.12).

Figure 5.6 shows, for some generated data set with three relevant variables and twelve irrelevant variables, training and testing errors (left) and the AIC statistic, loss function and AIC penalty term (right) for a sequence of NB models generated by fsNB. Note that, in this example, the best model is nearly the same for the test data as for AIC.

5.7 EXPERIMENTS

So far, we have presented some examples to illustrate the claims. Now, we perform a more systematic evaluation of the methods. We test the methods first on some synthetic data sets and then on some data sets derived from the *Diabetes* data set, taken from the UCI repository[1].

1 http://archive.ics.uci.edu/ml

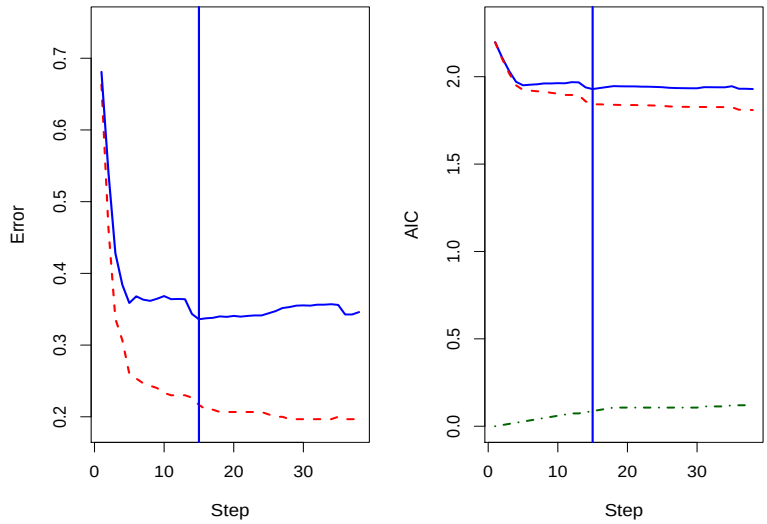


Figure 5.6: Left: Training error (dashed line) and testing error (solid line). Minimum testing error step is highlighted with a vertical line. Right: AIC statistic (solid line), loss function (dashed line) and AIC penalty term (dashed-dotted line). The step with the lowest AIC statistic value is highlighted with a vertical line.

5.7.1 *Synthetic data sets*

We now run the algorithms on a number of synthetic training/test data sets, generated from several scenarios. Each data set has $p = 20$ predictors, which can be discrete ($M_i = 3$) or continuous. Training data sets have $N = 300$ instances and test data sets have $N_{te} = 3000$ instances. There are $J = 3$ class values.

Within each data set, there are $p_1 = 3$ non-noisy predictors, $p_2 = 7$ non-fully redundant predictors, which nevertheless carry some information, and $p_3 = 10$ totally noisy predictors, which may be irrelevant or redundant to any of the p_1 non-noisy predictors. We call these three groups, respectively, V_1, V_2 and V_3. Hence, $p = p_1 + p_2 + p_3$.

For each experiment, we randomly generate the "true" parameters that produce the data as follows. In the discrete case, for each predictor in V_1, we sample

$$\boldsymbol{\theta}_{ij} \sim \text{Dir}(\boldsymbol{c}),$$

where $\boldsymbol{\theta}_{ij}$ is the j-th column of $\boldsymbol{\Theta}_i$ and $\text{Dir}(\boldsymbol{c})$ is a Dirichlet distribution with the vector of shape parameters $\boldsymbol{c}$, whose components are all equal except one, which is different for each $j \in \{1, ..., J\}$.

Within each data set, all predictors in V_2 have the same CPT, which is similarly generated from a Dirichlet distribution. Each predictor in V_2 is slightly redundant to the preceding and following predictor, i.e., $H(X_i|X_{i-1}) < H(X_i)$ and $H(X_i|X_{i+1}) < H(X_i)$ for $i \in \{p_1 + 2, p_1 + p_2 - 1\}$ (assuming that predictors in V_2 are preceded by predictors in V_1 and followed by predictors in V_3). This redundancy is achieved by setting the value of the predictor X_i to be equal to either X_{i-1} or X_{i+1} with a probability equal to 0.5.

If predictors in V_3 are irrelevant, they have the parameters of a multinomial distribution, and they are generated from a Dirichlet distribution with equal hyperparameters. In other words, the CPT columns of each predictor in V_3 are all equal. If predictors

in V_3 are redundant, the parameters are generated as for irrelevant predictors. In this case, however, each predictor has a very low conditional entropy given some randomly selected predictor from V_1. This is achieved by setting the value of the predictor in V_3 to be equal to the predictor in V_1 with a probability equal to 0.9. Note that, once the corresponding predictor in V_1 has been added, this predictor does not carry any additional useful information at all.

In the continuous case, predictors are generated from Gaussian distributions. For each predictor in V_1, we sample

$$\mu_{ij} \sim \text{Unif}(-2, 2), \quad \sigma_{ij} = 0.75,$$

where $\text{Unif}(-2, 2)$ is the uniform distribution between -2 and 2.

As in the discrete case, all predictors in V_2 have the same parameters. Again, let X_i be equal to either X_{i-1} or X_{i+1} with a probability equal to 0.5.

If predictors in V_3 are irrelevant, we have $\mu_{ij} = m_i$, for all $j \in \{1, ..., C\}$. The value m_i is generated from a uniform distribution in the interval $(-2, 2)$. If predictors in V_3 are redundant, parameters are generated similarly, but, for each data instance, each predictor in V_3 is bound, with a probability equal to 0.9, to have the same value as some predictor in V_1, plus some small noise.

Finally, we set $\boldsymbol{\pi} = (1/3, 1/3, 1/3)$ in all cases. Hence, we have four different scenarios, which are the four possible combinations of discrete/continuous predictors and irrelevant/redundant predictors within V_3.

We generate 100 different data sets from each scenario using the Bayes rule (taking into account the mentioned redundancies). Table 5.1 shows, for each data set type, the means and standard deviations of the testing misclassification error, number of selected variables and number of (fully) noisy selected variables, for NB, SNB, WNB, aL_1/L_2-NB and fsNB. We have run fsNB with parameters $\epsilon = 0.025, \nu = 20$, which we have empirically observed to be a good choice in general. Also, we use early stopping. For comparison's sake, we have also run NB on a subset of predictors, selected by (prefiltering) correlation-based feature selection (Hall, 2000). We denote this approach as CFS+NB.

We find that there are two clearly different scenarios. First, when the noisy predictors are irrelevant, the methods that do not select variables (NB and WNB) perform best. This is certainly expectable, because, as discussed above, NB is relatively robust to irrelevant predictors, and there are not enough to significantly reduce accuracy. Note, however, that, in the discrete case at least, fsNB is closer to NB and WNB than the other wrapper selective methods and also than CFS+NB. Second, when the noisy predictors are redundant, fsNB beats the others. CFS+NB also works fine and turns out to be the most accurate method in the continuous case. The differences between fsNB and SNB are probably due to the fsNB's balanced estimation of parameters of the predictors in V_2. The number of selected predictors is not very different for fsNB and SNB in this case. CFS+NB clearly selects more predictors than the wrapper approaches. Finally, note that, excepting for the continuous with irrelevant noise variables data set, aL_1/L_2-NB does not excel. Although L_1/L_2-NB is not shown in Table 5.1, aL_1/L_2-NB is slightly better than its non-adaptive counterpart. Regarding computational cost, SNB and fsNB take, respectively, 125.10 and 6732.25 evaluations on average. The computational cost is similar for all data sets.

Table 5.1: Mean testing misclassification error (top), mean number of selected variables (middle) and mean number of (fully) noisy selected variables (bottom) for each synthetic data set type and each method. Data set types are discrete with irrelevant noise variables (DI), continuous with irrelevant noise variables (CI), discrete with redundant noise variables (DR) and continuous with redundant noise variables (CR). The best result for each row is highlighted in bold. NB and WNB have been omitted from the variable selection report because they do not perform variable selection.

	Misclassification error					
Data set type	NB	SNB	WNB	aL_1/L_2-NB	fsNB	CFS+NB
DI	0.076(±0.03)	0.080(±0.03)	**0.075**(±0.03)	0.079(±0.06)	0.077(±0.02)	0.166(±0.18)
CI	**0.082**(±0.04)	0.083(±0.05)	**0.082**(±0.04)	0.083(±0.05)	0.083(±0.05)	0.087(±0.02)
DR	0.152(±0.07)	0.076(±0.02)	0.131(±0.08)	0.171(±0.10)	**0.070**(±0.03)	0.082(±0.06)
CR	0.132(±0.05)	0.097(±0.03)	0.132(±0.05)	0.158(±0.10)	0.090(±0.03)	**0.083**(±0.08)

	Number of selected variables					
Data set type	NB	SNB	WNB	aL_1/L_2-NB	fsNB	CFS+NB
DI	–	**6.2**(±1.4)	–	6.5(±2.1)	6.3(±1.7)	10.8(±0.8)
CI	–	6.0(±1.5)	–	6.0(±1.2)	**5.3**(±1.8)	10.0(±0.1)
DR	–	5.7(±1.5)	–	10.2(±2.1)	**5.6**(±1.7)	10.4(±0.6)
CR	–	**5.5***(±1.3)	–	11.8(±1.9)	6.6(±2.3)	10.8(±1.0)

	Number of noisy selected variables					
Data set type	NB	SNB	WNB	aL_1/L_2-NB	fsNB	CFS+NB
DI	–	0.6(±0.8)	–	**0.2**(±0.7)	0.3(±0.9)	0.8(±0.9)
CI	–	0.8(±0.8)	–	**0.1**(±0.3)	**0.1**(±0.2)	0.4(±0.5)
DR	–	0.4(±0.7)	–	5.1(±0.8)	**0.2**(±0.5)	0.6(±0.6)
CR	–	**0.8**(±0.4)	–	5.9(±0.8)	1.1(±0.7)	1.0(±1.0)

5.7.2 *Diabetes data sets*

Next, we carry out some experiments with real data. We use the *Diabetes* data set, which has $N = 442$ instances and $p = 10$ continuous predictors. Although the response is continuous, we generate data sets for binary classification by means of the rule

$$y_n = \begin{cases} 0 & \text{if } \tilde{y}_n < \tau, \\ 1 & \text{if } \tilde{y}_n \geq \tau. \end{cases}$$

where $\tilde{y}_n$ is the continuous response and τ is some real constant. We generate three different data sets by setting τ to be equal to the first three quartiles. Therefore, for each data set, we have, respectively, $\boldsymbol{\pi} = (1/4, 3/4)$, $\boldsymbol{\pi} = (1/2, 1/2)$ and $\boldsymbol{\pi} = (3/4, 1/4)$.

Table 5.2 illustrates the results obtained from 10-fold cross-validation, which include testing misclassification error and number of selected variables. As before, the tested methods are NB, SNB, WNB, aL_1/L_2-NB, fsNB and CFS+NB. We have run fsNB with parameters $\epsilon = 0.025, \nu = 20$, using early stopping.

Note that fsNB is the most accurate, followed by SNB and CFS+NB. Note that aL_1/L_2-NB is always worse than fsNB and SNB, which is a possible sign of certain redundancy among the predictors (that aL_1/L_2-NB is not purging). In these data sets, WNB obtains very similar results to NB. None of the methods, however, is very accurate when $\boldsymbol{\pi} = (1/4, 3/4)$. In this case, the simple "most frequent class" rule obtains an accuracy similar to NB (0.28), which is not greatly improved by any method. On the other hand, the number of selected predictors is reasonable for SNB, fsNB and CFS+NB, and higher for aL_1/L_2-NB. The L_1/L_2-NB approach (not shown) achieves similar results to aL_1/L_2-NB, for both accuracy and selected variables. The mean number of evaluations for SNB is 30.7, whereas fsNB needs 905.4 evaluations on average.

Table 5.2: Mean 10-CV cross-validated misclassification error (top) and number of selected variables (bottom) for each data set derived from the *Diabetes* data set and each method. The best result for each row is highlighted in bold. NB and WNB have been omitted from the variable selection report because they do not perform variable selection.

	Misclassification error					
Data set	NB	SNB	WNB	aL_1/L_2-NB	fsNB	CFS+NB
$\pi = (1/4, 3/4)$	0.28(±0.06)	0.23(±0.05)	0.28(±0.06)	0.25(±0.07)	**0.21**(±0.06)	0.26(±0.10)
$\pi = (1/2, 1/2)$	0.28(±0.07)	0.27(±0.06)	0.28(±0.07)	0.27(±0.08)	**0.26**(±0.07)	0.27(±0.15)
$\pi = (3/4, 1/4)$	0.20(±0.04)	0.18(±0.05)	0.20(±0.04)	0.19(±0.05)	**0.16**(±0.06)	0.18(±0.15)

	Number of selected predictors					
Data set	NB	SNB	WNB	aL_1/L_2-NB	fsNB	CFS+NB
$\pi = (1/4, 3/4)$	–	**1.7**(±0.48)	–	4.4(±3.80)	2.2(±0.42)	3.0(±0.42)
$\pi = (1/2, 1/2)$	–	**3.3**(±0.48)	–	8.3(±0.48)	4.4(±0.95)	4.2(±0.51)
$\pi = (3/4, 1/4)$	–	3.3(±0.67)	–	8.4(0.84±)	**2.4**(±0.51)	4.9(±0.78)

5.7.3 *Neuroscience fMRI data*

Finally, we report results on functional magnetic resonance imaging (fMRI) data, the *StarPlus* data set[1], collected at Carnegie Mellon University.

Experiments are conducted on six subjects and forty trials per subject. For each trial, the subject is shown a picture for four seconds and a sentence for four seconds. The objective is to discriminate between these two mental states: "picture" or "sentence". Each data item matches a unique 3-dimensional image. Images are captured every 0.5 seconds. Hence, each trial has 16 useful images. Briefly, there are six data sets, one per subject, and they all have $N = 40 \times 16 = 640$ data items. On the other hand, each image has a number of voxels, split into 25 localized regions of interest (ROIs). Here, instead of considering each individual voxel, we will use the mean activation of voxels at each ROI. Therefore, our data set has $p = 25$ predictors.

Table 5.3 shows the results obtained from 10-fold cross-validation. Algorithms and parameter configuration are the same than in previous experiments.

In this example, fsNB beats the other wrapper algorithms in four out of six subjects, whereas SNB is the best wrapper method for the other two subjects. CFS+NB performs better than fsNB and SNB in one of the subjects. On the other hand, fsNB selects fewer predictors than SNB in all cases. The number of selected predictors is not very different from CFS+NB. The performance of aL_1/L_2-NB is poor, and the model selection procedure often prefers the model with no predictors. Note that, in general, none of the approaches behave particularly well. We conjecture that this is because the data have a very nonlinear nature.

The mean number of evaluations for SNB is 126.2, whereas fsNB needs 9311.1 evaluations on average.

5.7.4 *Comparison across data sets*

To conclude the experimental discussion, we perform an overall analysis of the methods that includes the results obtained from all the data sets described above. To do so, we follow the guidelines outlined by García and Herrera (2008), performing all pairwise comparisons among the classifiers to detect (statistically) significant differences between each pair. In particular, we use the Bergmann and Hommel (1988) dynamic

1 http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-81/www/

Table 5.3: Mean 10-CV cross-validated misclassification error (top) and number of selected variables (bottom) for each subject in the *Starplus* data set and each method. The best result for each row is highlighted in bold. NB and WNB have been omitted from the variable selection report because they do not perform variable selection.

	Misclassification error					
Subject	NB	SNB	WNB	aL_1/L_2-NB	fsNB	CFS+NB
04799	0.47(±0.08)	0.45(±0.04)	0.47(±0.08)	0.52(±0.07)	**0.41**(±0.26)	0.50(±0.23)
05675	0.44(±0.07)	**0.43**(±0.06)	0.44(±0.07)	0.51(±0.06)	0.50(±0.11)	0.46(±0.19)
04820	0.44(±0.07)	0.43(±0.06)	0.44(±0.07)	0.55(±0.03)	0.37(±0.34)	**0.34**(±0.21)
05680	0.45(±0.05)	0.44(±0.06)	0.45(±0.05)	0.57(±0.04)	**0.35**(±0.26)	0.48(±0.16)
04847	0.36(±0.06)	**0.33**(±0.05)	0.36(±0.06)	0.57(±0.06)	0.35(±0.06)	0.44(±0.18)
05710	0.40(±0.07)	0.45(±0.06)	0.40(±0.07)	0.55(±0.02)	**0.36**(±0.26)	0.48(±0.32)
	Number of selected predictors					
Subject	NB	SNB	WNB	aL_1/L_2-NB	fsNB	CFS+NB
04799	–	3.8(±1.51)	–	**0.2**(±0.02)	1.0(±0.77)	1.1(±0.81)
05675	–	3.5(±1.32)	–	**0.3**(±0.01)	0.9(±0.30)	1.0(±0.66)
04820	–	4.9(±1.31)	–	**0.1**(±0.01)	1.3(±1.04)	0.9(±1.01)
05680	–	5.5(±2.50)	–	**0.3**(±0.02)	1.2(±0.79)	1.2(±0.36)
04847	–	4.2(±1.51)	–	**0.1**(±0.02)	1.8(±0.03)	2.1(±0.77)
05710	–	5.9(±1.82)	–	**0.2**(±0.03)	1.5(±0.91)	2.0(±0.52)

procedure to adjust the raw p-values. Table 5.4 shows, for each pair, these adjusted p-values. We can observe that fsNB is significantly better than all the other procedures, with a significance level of 0.05.

Table 5.4: Adjusted p-values, via the Bergmann-Hommel's dynamic procedure, for each pair of methods.

Pair	Adjusted p-value	Pair	Adjusted p-value
fsNB vs NB	1.57E-6	SNB vs NB	0.783
fsNB vs WNB	3.35E-6	SNB vs WNB	0.783
fsNB vs aL_1/L_2-NB	3.35E-6	SNB vs aL_1/L_2-NB	0.783
fsNB vs SNB	9.86E-4	CFS+NB vs SNB	2.210
fsNB vs CFS+NB	0.030	aL_1/L_2-NB vs NB	2.424
CFS+NB vs NB	0.352	WNB vs NB	2.424
CFS+NB vs WNB	0.352	aL_1/L_2-NB vs NB vs	2.424
CFS+NB vs aL_1/L_2-NB	0.352		

5.8 DISCUSSION

In this chapter, we have proposed a forward stagewise version of the forward stepwise selective naïve Bayes approach. This work has been published in the paper (Vidaurre et al., 2011b). This approach has some advantages over the usual selective naïve Bayes, and often beats other naïve Bayes-based algorithms, like the weighted naïve Bayes. We have illustrated this point empirically on both synthetic and real data sets. The forward stagewise approach is computationally more expensive than selective naïve Bayes. Computational complexity, however, can be modulated via the ν parameter, which, with ϵ, defines the extent of the search at each step.

We have also extended the L_1/L_2-regularized naïve Bayes approach taken by van Gerven and Heskes (2008) to accommodate discrete predictors. In addition, we have introduced a handy modification of this method based on adaptive penalties (Zou, 2006). Unlike the forward stagewise naïve Bayes, however, the L_1/L_2-regularized naïve Bayes approach does not discard redundant predictors and, hence, performs

poorly when the data set contains large sets of these noisy predictors. This phenomenon has been discussed and observed in a comprehensive synthetic experimental setting. L_1/L_2-regularized naïve Bayes fares relatively well, though, when noisy predictors are irrelevant. Nonetheless, irrelevant predictors are considerably less harmful to the classification than redundant predictors.

In addition, note that, whereas it is straightforward for the forward stagewise naïve Bayes approach to deal with data sets with both discrete and continuous predictors, it is not so simple for the L_1/L_2-regularized naïve Bayes method. This is because the continuous and discrete penalties scale differently. Besides discretizing the continuous predictors, we have two choices to address this issue. First, we can use two separate regularization parameters for each type of penalty, which is an expensive solution if they have to be estimated. Second, we can somehow scale the continuous predictors to make the penalties scale similarly. This is an approximate and rather tricky solution, and we do not expect the results to be good.

Also, the weighted naïve Bayes approach cannot be used with continuous predictors unless they are discretized beforehand. In summary, flexibility is another advantage of the proposed forward stagewise naïve Bayes approach.

Future work could focus on the possibility of converting the forward stagewise naïve Bayes approach into a boosting method, where all intermediate models collaborate to output a final prediction. Plugging more complex Bayesian classifiers into this framework is also on the agenda. Of course, the algorithm structure accepts other distributions than multinomial and Gaussian.

6

AN L_1-REGULARIZED NAÏVE BAYES-INSPIRED CLASSIFIER

6.1 INTRODUCTION

This chapter is, like Chapter 5, very related to the naïve Bayes idea. Some training schemes have been proposed on top of the naïve Bayes idea. For example, the *weighted naïve Bayes* (Ferreira et al., 2001), discussed in Chapter 5, assigns a weight to each predictor so that some predictors have more influence than others. Unfortunately, the naïve Bayes model (including weighted naïve Bayes) always includes all predictors in the model and behaves poorly in the presence of redundant predictors. Langley and Sage (Langley and Sage, 1994) discussed this issue and proposed the *selective naïve Bayes* classifier. This classifier greedily includes predictors in a search-based algorithm. However, this is a heuristic method and is not guaranteed to find an optimal model. Without a prefiltering step, as in (Djebbari and Labbe, 2009; Blanco et al., 2005) for example, the selective naïve Bayes is seldom applicable for high-dimensional settings on computational grounds. On the other hand, the so-called *semi-naïve Bayes* (Pazzani, 1996) performs a heuristic greedy search to select predictors and find dependences between them, fusing these predictors to a single predictor. The same computational issue applies here.

Regularization techniques have occasionally been used to improve naïve Bayes. For example, Boullé (2007) applied regularization on a selective naïve Bayes procedure. The criterion used to fit data to a model is the data likelihood plus a penalization term. This is derived from a Bayesian approach with a prior distribution that assigns higher probabilities to networks with fewer predictors. This is embedded in a greedy search heuristic that iteratively selects predictors for inclusion in (or exclusion from) the model.

In this chapter, we introduce a supervised classification method that is inspired on naïve Bayes and based on linear regression. As we discuss below, it enjoys advantages from both frameworks. On the one hand, this formulation allows to apply regularization techniques from linear regression that permit to discard both redundant and irrelevant predictors. Redundant predictors are known to be harmful for naïve Bayes and variants, and also for our model. On the other hand, like naïve Bayes, it can directly deal with both continuous and discrete predictors and can be directly used in multi-class problems. Thus, our method is applicable to a wide range of data sets. This chapter is based on the submitted paper (Vidaurre et al., 2011c).

The proposed method establishes an L_1-regularized linear combination of the likelihood contributions of each predictor, choosing the coefficients of the linear combination so that the resulting value is as close to 1.0 as possible for each instance. The L_1-penalty yields a sparse vector of coefficients, dropping the likelihood contribution of some predictors and, thus, enhancing the interpretability of the model. As we

will show, this method can discard both redundant and irrelevant predictors (their respective likelihood contributions).

The devised loss function also meets the requirements for applying a LARS type algorithm (Rosset and Zhu, 2007). This algorithm would efficiently compute the entire regularization path at one shot. This is beneficial in high dimensional settings, where gradient-based methods are difficult to apply on computational grounds.

6.2 THE METHOD

The notation and main concepts used all along this chapter were introduced in Chapter 5. In this section, therefore, we directly introduce the method. First, we separately focus on each predictor to build a penalized linear expression whose minimization will yield a classifier that discards irrelevant and redundant predictors.

We first obtain the ML parameters $\hat{\boldsymbol{\Theta}}_i^{(1)}$, for X_i being discrete, and $\hat{\boldsymbol{\mu}}_i^{(1)}$ and $\hat{\boldsymbol{\sigma}}_i^{(1)}$, for X_i being discrete, from Equations (5.4), defined in the previous chapter. Let $\hat{\boldsymbol{\Omega}}_i^{(1)}$ be either $\hat{\boldsymbol{\Theta}}_i^{(1)}$ or $\{\hat{\boldsymbol{\mu}}_i^{(1)}, \hat{\boldsymbol{\sigma}}_i^{(1)}\}$. Now, we establish the linear expression:

$$\sum_{n=1}^{N}\sum_{i=1}^{p}\beta_i P(Y=y_n|X_i=x_{ni},\hat{\boldsymbol{\Omega}}_i^{(1)}),\ \ s.t.\ \ \sum_{i=1}^{p}\beta_i=1,\quad 0\leq\beta_i\leq 1, \tag{6.1}$$

where, following the Bayes' rule, we have

$$P(Y=y_n|X_i=x_{ni},\hat{\boldsymbol{\Omega}}_i^{(1)})=\frac{\psi(X=x_{ni}|Y=y_n,\hat{\boldsymbol{\Omega}}_i^{(1)})P(Y=y_n)}{\sum_{j=1}^{c}\psi(X=x_{ni}|Y=j,\hat{\boldsymbol{\Omega}}_i^{(1)})P(Y=j)} \tag{6.2}$$

where the function $\psi(\cdot)$ is defined in Equations (5.2) and (5.3).

Vector $\hat{\boldsymbol{\beta}}=(\hat{\beta}_1,...,\hat{\beta}_p)$ would be chosen to maximize (6.1), hence giving more weight to predictors that are more relevant for the classification. The rationale of this approach is that relevant predictors will have values $P(Y=y_n|X_i=x_{ni},\hat{\boldsymbol{\Omega}}_i^{(1)})$ closer to one than irrelevant predictors. Hence, when maximizing (6.1) across the data set, the coefficients β_i of the relevant predictors are promoted to be higher. Note also that, as long as $\sum_{i=1}^{p}\beta_i=1$, expression

$$\sum_{i=1}^{p}\beta_i P(Y=y_n|X_i=x_{ni},\hat{\boldsymbol{\Omega}}_i^{(1)})$$

ranges from 0 to 1, like a probability. We can use this as a basis for classifying future instances. Specifically, given $\hat{\boldsymbol{\beta}}$ and $\hat{\boldsymbol{\Omega}}^{(1)}$, we would select, for a new instance given by x_i, the class value $j\in\{1,...,c\}$ that maximizes

$$\sum_{i=1}^{p}\beta_i P(Y=j|X_i=x_i,\hat{\boldsymbol{\Omega}}_i^{(1)}). \tag{6.3}$$

Note that $\beta_i=0$ implies that predictor X_i is not selected. Likewise, higher values of β_i would attach more importance to predictor X_i. Predictors that are considered to be relevant (i.e., with a high β_i) are expected to have a higher probability $P(Y=j|X_i=x_i,\hat{\boldsymbol{\Omega}}_i^{(1)})$ for the true class, as it was in the training data set.

To obtain $\boldsymbol{\beta}$, we could devise a linear optimization problem that maximizes (6.1) for the data set. However, it will typically not drive any β_i to exactly zero, and, hence, will

not perform variable selection. Since the condition $\sum_{i=1}^{p} \beta_i = 1$ leads to a numerically difficult problem, we alternatively propose an L_1-constrained problem to estimate $\boldsymbol{\beta}$:

$$\min_{\boldsymbol{\beta}} \sum_{n=1}^{N} \left[1 - \sum_{i=1}^{p} \beta_i P(Y = y_n | X_i = x_{ni}, \hat{\boldsymbol{\Omega}}_i^{(1)})\right]_+^2 + \lambda \sum_{i=1}^{p} |\beta_i| \quad (6.4)$$
$$s.t. \quad 0 \leq \beta_i \leq 1, \forall i.$$

where $[\cdot]_+$ is the positive part of the argument. Penalty term $\lambda \sum_{i=1}^{p} |\beta_i|$ in (6.4) is equivalent to imposing a restriction

$$\sum_{i=1}^{p} |\beta_i| \leq s,$$

and there is a one-to-one correspondence between λ and s (Tibshirani, 1996). Hence, for some $\lambda = \lambda_1$ such that $\sum_{i=1}^{p} \beta_i \leq 1$ is imposed, we have, as before,

$$0 \leq \sum_{i=1}^{p} \beta_i P(Y = y_n | X_i = x_{ni}, \hat{\boldsymbol{\Omega}}_i^{(1)}) \leq 1$$

and therefore

$$\max_{\boldsymbol{\beta}} \sum_{n=1}^{N} \sum_{i=1}^{p} \beta_i P(Y = y_n | X_i = x_{ni}, \hat{\boldsymbol{\Omega}}_i^{(1)}) = \quad (6.5)$$
$$\min_{\boldsymbol{\beta}} \sum_{n=1}^{N} \left[1 - \sum_{i=1}^{p} \beta_i P(Y = y_n | X_i = x_{ni}, \hat{\boldsymbol{\Omega}}_i^{(1)})\right]_+$$

For mathematical convenience, the expression in brackets in the right term of Equation (6.5) is squared in (6.4). Thus, a vector $\hat{\boldsymbol{\beta}}$ solving (6.4) for $\lambda = \lambda_1$ will be an estimator of the maximizer of (6.1). Because of the variable selection effect of the lasso penalty, $\hat{\boldsymbol{\beta}}$ is expected to be sparse.

In this chapter, instead of fixing λ to λ_1, we let λ to traverse the whole regularization path, choosing it either to maximize the classification accuracy on a validation data set or to minimize some penalization criterion like AIC (Akaike, 1974).

Equation (6.4) fulfills the requirements listed in (Rosset and Zhu, 2007) and can be solved by an efficient LARS procedure. Specifically, as sufficient conditions, the loss function is a quadratic loss function and the penalty function is a lasso penalty. We detail in the next subsection a LARS-type algorithm with a couple of modifications to include the restriction $0 \leq \beta_i \leq 1$. As we discuss below, this formulation allows us to discard both redundant and irrelevant predictors. The method, which we will call L_1-NB, is summarized in Algorithm 5. In the pseudocode, AIC is used for model selection.

Although our approach is definitely different from naïve Bayes, we rely, to some extent, on the same two principles. First, since the loss function in Equation (6.4) is linear on $P(Y = y_n | X_i = x_{ni}, \hat{\boldsymbol{\Omega}}_i^{(1)})$, we are assuming that the values of the class are linearly separable given the predictors. Naïve Bayes establishes the same assumption. Second, both do not model any explicit relation between the predictors. Nonetheless, unlike naïve Bayes, we are implicitly avoiding redundancy.

We have discussed in Chapter 5 why L_1/L_2-regularization of differences applied to naïve Bayes (van Gerven and Heskes, 2008) does not discard redundant predictors. Unfortunately, redundant predictors are known to have a more harmful effect on the classification accuracy than irrelevant predictors in the naïve Bayes model (Langley and Sage, 1994; Drugan and Wiering, 2010).

Let us consider two extreme examples. For simplicity's sake we will consider discrete predictors only. Let X_i be an irrelevant predictor and let X_{i_1} and X_{i_2} be two redundant predictors, for example, a predictor that appears twice. In the first case, irrelevance, $P(Y = j|X_i = k)$ takes the same value for all $k \in \{1, ..., m_i\}$ and, hence, X_i has no influence on the decision. In the second case, redundancy, however, the duplicated predictor is twice as important as it should be:

$$\begin{aligned} & P(Y = j|X_1 = k_1, ..., X_q = k_q, \boldsymbol{\Theta}_1, ..., \boldsymbol{\Theta}_q) \\ \propto \quad & P(Y = j) \cdots P(X_{i_1} = k_{i_1}|Y = j, \boldsymbol{\Theta}_{i_1}) \cdots P(X_q = k_q|Y = j, \boldsymbol{\Theta}_q) \\ = \quad & P(Y = j) \cdots P(X_{i_1} = k_{i_1}|Y = j, \boldsymbol{\Theta}_{i_1})^2 \cdots P(X_q = k_q|Y = j, \boldsymbol{\Theta}_q). \end{aligned}$$

It is clear that the weighted naïve Bayes by Ferreira et al. (2001) does not penalize redundant predictors. Since the weights assessment relies on the very concept of relevance, this method assigns high weights to all (separately considered) relevant predictors, no matter whether they are redundant. Other weighted naïve Bayes algorithms whose weights estimation relies on prediction accuracy (Zhang and Sheng, 2004) instead of on the individual informative power of the predictor would in fact be able to discard redundant predictors. As mentioned above, this is because redundant predictors diminish the accuracy. These algorithms, however, are usually based on heuristic procedures like hill climbing or Markov chain Monte Carlo simulation and are computationally demanding when p is high.

It is well-known that, in regression, the original lasso only draws one out of a set of redundant predictors (Zou and Hastie, 2005). This can be easily shown, for example, on the LARS algorithm. Once LARS has selected one predictor, the correlation with the residuals of the other redundant predictors decreases dramatically. That is, the angle between the vector of residuals and the predictor evolves towards the right angle at the same pace for all redundant predictors. Since error correlation is the criterion for including new predictors, predictors that are redundant to those already included in the model are unlikely to be selected.

Interestingly, the proposed classifier can discard redundant predictors by solving (6.4). First, if X_{i_1} and X_{i_2} are discrete and Equation (5.9) is satisfied, the value of X_{i_2} can be determined if X_{i_1} is known and vice versa. Hence, there is a bijection between the i_1-th and the i_2-th columns of matrix $\boldsymbol{X}$. Obviously, this means that $P(X_{i_1} = x_{ni_1}|Y = y_n, \hat{\boldsymbol{\Theta}}_{i_1})^{(1)}$ and $P(X_{i_2} = x_{ni_2}|Y = y_n, \hat{\boldsymbol{\Theta}}_{i_2}^{(1)})$ are equal, $n = 1, ..., N$. Therefore, it follows

Algorithm 5 L_1-NB

Obtain ML parameters $\hat{\boldsymbol{\Omega}}_i^{(1)}$
Obtain matrix $\boldsymbol{B}$, $B_{ni} = P(Y = y_n|X_i = x_{ni}, \hat{\boldsymbol{\Omega}}_i^{(1)})$
Obtain solutions $\hat{\boldsymbol{\beta}}^{(l)}$, $l = 1, ..., L$, with LARS from $\boldsymbol{B}$
$\hat{\boldsymbol{\beta}} := \text{argmin}_{\boldsymbol{\beta}^{(l)}} AIC(\boldsymbol{\beta}^{(l)})$, $l = 1, ..., L$
Classify unlabeled classes using $\hat{\boldsymbol{\beta}}$ and Equation (6.3)

from Equation (6.2) that $P(Y = y_n|X_{i_1} = x_{ni_1}, \hat{\boldsymbol{\Theta}}_{i_1}^{(1)})$ and $P(Y = y_n|X_{i_2} = x_{ni_2}, \hat{\boldsymbol{\Theta}}_{i_2}^{(1)})$, $n = 1, ..., N$, are equal too.

Hence, if two predictors, X_{i_1} and X_{i_2}, are highly correlated then vector $P(Y = y_n|X_{i_1} = x_{ni_1}, \hat{\boldsymbol{\Theta}}_{i_1}^{(1)})$, $n = 1, ..., N$, and vector $P(Y = y_n|X_{i_2} = x_{ni_2}, \hat{\boldsymbol{\Theta}}_{i_2}^{(1)})$, $n = 1, ..., N$, will also be highly correlated. Therefore, Equation (6.4), which can be solved by LARS, would drop either X_{i_1} or X_{i_2} for the same reason that lasso does.

Second, if X_{i_1} and X_{i_2} are continuous and redundant, either X_{i_1} or X_{i_2} would also be discarded.

Proposition 1. *If X_{i_1} and X_{i_2} are continuous and redundant, vector $P(Y = y_n|X_{i_1} = x_{ni_1}, \hat{\boldsymbol{\Theta}}_{i_1}^{(1)})$, $n = 1, ..., N$, and vector $P(Y = y_n|X_{i_2} = x_{ni_2}, \hat{\boldsymbol{\Theta}}_{i_2}^{(1)})$, $n = 1, ..., N$, are also equal.*

Proof. If X_{i_1} and X_{i_2} are continuous and Equation (5.9) is satisfied, then $X_{i_1} = g(X_{i_2})$, $g()$ being some deterministic linear function $X_{i_1} = g(X_{i_2}) = b_0 + b_1 X_{i_2}$.

In this case, we have that $\boldsymbol{\mu}_{i_1} = b_1 \boldsymbol{\mu}_{i_2} + b_0$, $\boldsymbol{\sigma}_{i_1} = |b_1| \boldsymbol{\sigma}_{i_2}$, and, trivially, $f(x_{ni_1}|y_n; \mu_{i_1 y_n}, \sigma_{i_1 y_n}) = |b_1^{-1}| f(x_{ni_2}|y_r; \mu_{i_2 y_n}, \sigma_{i_2 y_n})$. By plugging this into Equation (6.2) we obtain

$$\begin{aligned} P(Y = y_n|X_{i_1} = x_{ni_1}, \hat{\boldsymbol{\mu}}_{i_1}^{(1)}, \hat{\boldsymbol{\sigma}}_{i_1}^{(1)}) &= \frac{|b_1^{-1}| f(x_{ni_2}|y_n; \hat{\mu}_{iy_n}^{(1)}, \hat{\sigma}_{iy_n}^{(1)}) P(Y = y_n)}{\sum_{j=1}^{c} |b_1^{-1}| f(x_{ni}|j; \hat{\mu}_{ij}^{(1)}, \hat{\sigma}_{ij}^{(1)}) P(Y = j)} \\ &= \frac{f(x_{ni_2}|y_n; \hat{\mu}_{iy_n}^{(1)}, \hat{\sigma}_{iy_n}^{(1)}) P(Y = y_n)}{\sum_{j=1}^{c} f(x_{ni}|j; \hat{\mu}_{ij}^{(1)}, \hat{\sigma}_{ij}^{(1)}) P(Y = j)} \\ &= P(Y = y_n|X_{i_2} = x_{ni_2}, \hat{\boldsymbol{\mu}}_{i_2}^{(1)}, \hat{\boldsymbol{\sigma}}_{i_2}^{(1)}). \end{aligned}$$

□

On the other hand, for all irrelevant predictors X_i, we have that

$$P(Y = y_n|X_i = x_{ni}, \hat{\boldsymbol{\Theta}}_i^{(1)}) = P(Y = y_n), \quad n = 1, ..., N.$$

Thus, irrelevant predictors give rise to equal vectors $P(Y = y_n|X_i = x_{ni}, \hat{\boldsymbol{\Theta}}_i^{(1)})$, $n = 1, ..., N$, and will be also discarded.

6.4 AN EFFICIENT LARS-TYPE ALGORITHM

In this section, we present a LARS (Efron et al., 2004) variant to accommodate the restriction $0 \leq \beta_i \leq 1$. This restriction can be considered as two separate conditions: $\beta_i \geq 0$ and $\beta_i \leq 1$.

The LARS algorithm is an iterative procedure that adds a predictor to the model at each step. LARS starts with no predictors. Firstly, it includes the predictor that is most correlated with the response into the active set of predictors $\mathcal{A}$. The response is regressed on this predictor, so that the coefficient of this predictor is moved towards the least squares solution until a new predictor reaches the same absolute correlation with the vector of residuals as that of $\mathcal{A}$. This new predictor is included in the active set $\mathcal{A}$. Now, the vector of residuals is regressed on the predictors in $\mathcal{A}$, moving their coefficients towards the joint least squares solution until a new predictor not in $\mathcal{A}$ reaches the same absolute correlation with such vector of residuals as that of $\mathcal{A}$. When

$n \geq p$, this procedure is repeated until all predictors are into the model. Otherwise, after $n-1$ steps, the residuals are zero and the algorithm terminates.

We denote the LARS input matrix as $\boldsymbol{B}$, so that $B_{ni} = P(Y = y_n | X_i = x_{ni}, \hat{\boldsymbol{\Theta}}_i)$. Let $\boldsymbol{B}_{\mathcal{A}}$ be the columns of $\boldsymbol{B}$ indexed by $\mathcal{A}$, $\hat{\boldsymbol{\beta}}_{\mathcal{A}}^{(l)}$ be the regression coefficients of the predictors in $\mathcal{A}$ at step l, $\mathbf{1}$ be a column vector with N elements equal to one, and $\boldsymbol{c} = (c_1, ..., c_p) = \boldsymbol{B}(\mathbf{1} - \boldsymbol{B}_{\mathcal{A}}\hat{\boldsymbol{\beta}}_{\mathcal{A}}^{(l)})$ be the correlation with the residuals.

Hence, at each step, the coefficients in $\mathcal{A}$ are updated as

$$\hat{\boldsymbol{\beta}}_{\mathcal{A}}^{(l+1)} = \hat{\boldsymbol{\beta}}_{\mathcal{A}}^{(l)} + \gamma \boldsymbol{w}_{\mathcal{A}}, \tag{6.6}$$

where $\boldsymbol{w}_{\mathcal{A}}$ is the joint least squares direction for the predictors in $\mathcal{A}$, and γ is "how much" $\hat{\boldsymbol{\beta}}_{\mathcal{A}}^{(l)}$ must be updated at step l. Then, γ is computed as the minimum value such that some predictor $i \notin \mathcal{A}$ reaches the same absolute correlation with such vector of residuals as that of $\mathcal{A}$. Algebraic details about the exact computation of γ and $\boldsymbol{w}_{\mathcal{A}}$ are described in (Efron et al., 2004).

The LARS modification for computing the exact regularization path of the lasso problem is based on detecting when a non-zero coefficient hits zero. Then, this predictor is dropped from $\mathcal{A}$ and the new least squares direction is computed. Working out γ in (6.6), for each predictor $i \in \mathcal{A}$, this happens when γ reaches

$$\gamma_i = -\frac{\hat{\beta}_i^{(l)}}{w_i}. \tag{6.7}$$

It will happen first at

$$\tilde{\gamma} = min_{\gamma_i > 0}\{\gamma_i\}. \tag{6.8}$$

Hence, if $\tilde{\gamma} < \gamma$, γ is corrected to be $\tilde{\gamma}$, and the new coefficients are computed by (6.6). The corresponding predictor is dropped from $\mathcal{A}$ for the next iteration.

Now, to accomplish the first condition $\beta_i \geq 0$, we compute γ as the minimum value such that some predictor $i \notin \mathcal{A}$ reaches the same positive correlation with the vector of residuals as that of $\mathcal{A}$. Thus, the difference is that the negative correlations with the residuals of predictors $i \notin \mathcal{A}$ are ignored for computing γ and deciding which predictor $i \notin \mathcal{A}$ enters the model. This modification is presented in (Efron et al., 2004).

Condition $\beta_i \leq 1$ does not appear in (Efron et al., 2004) and is slightly more complex. In this case, we need to detect when a regression coefficient hits 1. Let $\mathcal{M}$ be the set containing all predictors that have already reached 1. Again, for each predictor $i \in \mathcal{A}$, we work out γ in (6.6):

$$\gamma_i = \frac{1 - \hat{\beta}_i^{(l)}}{w_i}, \tag{6.9}$$

so that

$$\tilde{\gamma} = min_{\gamma_i > 0}\{\gamma_i\}. \tag{6.10}$$

If $\tilde{\gamma} < \gamma$, then we would set $\gamma = \tilde{\gamma}$, compute the new coefficients by (6.6) and move this predictor from $\mathcal{A}$ to $\mathcal{M}$. The new direction $\boldsymbol{w}_{\mathcal{A}}$ is computed on the current residual as usual.

However, it is well known that, when there is some dependence between the predictors, some predictors can decrease their regression coefficients at some step of the algorithm. We need to verify when it happens for predictors in $\mathcal{M}$, because they

would detach from 1 and should be included in $\mathcal{A}$ again. Since the regularization path is piecewise linear, it can only occur when a new predictor is included into or dropped from the model.

Let us include the predictors in $\mathcal{M}$ into the calculation of the joint least squares direction at step l. Let $\boldsymbol{w}_{\mathcal{A}\cup\mathcal{M}}$ denote this direction, assuming that $\mathcal{M}$ is not empty. Predictors from $\mathcal{M}$ that have a positive direction $w_i \geq 0$ are definitely discarded at this step. Let $\mathcal{M}^-$ contain all predictors in $\mathcal{M}$ excepting predictors with direction $w_i \geq 0$. Now, we calculate a new joint least squares direction $\boldsymbol{w}_{\mathcal{A}\cup\mathcal{M}^-}$. Again, we check if there are predictors in $\mathcal{M}^-$ whose direction w_i is positive. If this occurs, we delete them from $\mathcal{M}^-$ and update $\boldsymbol{w}_{\mathcal{A}\cup\mathcal{M}^-}$. In summary, at each step l, this procedure must be repeated until $\mathcal{M}^-$ is empty or all its predictors have a negative direction. These predictors must be moved from $\mathcal{M}$ to $\mathcal{A}$ for the next step.

Notice that, unlike $\beta_i \geq 0$, the $\beta_i \leq 1$ restriction implies additional computations. Specifically, at each step, additional least squares directions must be computed if $\mathcal{M}$ is not empty. If p is high and efficiency is a main concern, a possibility is, once regression coefficients reach 1, to attach these predictors to $\mathcal{M}$ for the rest of the algorithm. Hence, at each step, the joint least squares direction is only computed in $\mathcal{A}$ and we do not need to check whether any predictor in $\mathcal{M}$ has to be be moved to $\mathcal{A}$. Note that this can potentially produce a different regularization path. This is the approach followed in this chapter because the exact calculation of the regularization path is not crucial.

Summing up, the three described modifications are trivially combined by choosing γ as the value that first triggers any of the following events:

- A non-zero coefficient hits zero (Equations (6.7,6.8)).
- Some predictor $i \notin \mathcal{A}$ reaches the same positive correlation with the vector of residuals as that of $\mathcal{A}$.
- A non-zero coefficient hits 1 (Equations (6.9,6.10)).

Note that the computational cost of the LARS algorithm is dominated by the inversion of $\boldsymbol{B}'_{\mathcal{A}}\boldsymbol{B}_{\mathcal{A}}$ for computing the joint least squares direction at each step. The entire LARS solution path for $p < n$ variables, however, can be computed at the same cost than a least squares fit, i.e., $O(p^3 + Np^2)$. This is achieved by updating the Cholesky factorization of $\boldsymbol{B}'_{\mathcal{A}}\boldsymbol{B}_{\mathcal{A}}$ found at the previous step. At the final step, we have computed the Cholesky factorization of $\boldsymbol{B}'\boldsymbol{B}$. Nevertheless, the introduced modifications can induce more than p steps, and, hence, the computational cost can be slightly increased.

Second, we test the proposed method on a high dimensional data set.

6.5 EXPERIMENTS

We present some illustrative results on two different scenarios. First, we evaluate the effect of redundant and irrelevant predictors. Second, we test the proposed method on a high dimensional data set.

6.5.1 *Irrelevance and redundancy*

For the first objective, we test the behavior of our method on one of the *Soybean* data sets (Michalski, 1980), where all the predictors are discrete with four or five

categories. We focus on the version with no missing values, called *Soybean Small* in the UCI repository[1]. This data set has $N = 47$ instances and $p = 21$ predictors. We have chosen *Soybean* because it is a moderate-sized and well-behaved data set, suitable for testing how sensitive the algorithm is to the above issues.

Based on the original data set, we built several new data sets by adding different numbers of irrelevant and redundant predictors. We added 0, p, $2p$, $3p$, $4p$ and $5p$ irrelevant (randomly generated) predictors, and the same numbers of redundant predictors. We tested all combinations of redundancy and irrelevance. Redundant predictors are randomly generated values that are highly correlated (0.8) with an existing predictor, which is itself highly correlated to the class. We have tested a total of $6 \times 6 = 36$ data sets.

We compared the proposed method to ordinary naïve Bayes, naïve Bayes with prefiltering feature selection, weighted naïve Bayes (Ferreira et al., 2001) with prefiltering feature selection and selective naïve Bayes (Langley and Sage, 1994). Prefiltering is based on mutual information to the class. We introduced three random predictors sampled from a multinomial distribution with five categories and equal probabilities for each category. After, we discarded those predictors whose mutual information is lower than one of the three random predictors. An analogous prefiltering approach was taken for example in (Bi et al., 2003).

For each data set we performed 5-fold cross-validation, so that 80% of the data is used for training at each fold. Model evaluation was based on the AIC statistic. Note that this is needed by both our approach (for selecting λ) and selective naïve Bayes.

Graphs in Figures 6.1 and 6.2 show, respectively, the accuracy and the number of selected predictors. For a given number of irrelevant predictors, each graph displays the results for increasing numbers of redundant predictors. Figure 6.1 indicates with a horizontal thick line that the difference of the L_1-NB accuracy to the second best method is statistically significant with a significance level of 0.05. We do not show the number of correctly selected predictors because it is not clear which variables from the original set should really be selected. The total number of predictors in the data set is marked by the ordinary naïve Bayes line.

As expected, irrelevant predictors do not affect the performance of the evaluated classifiers much, except for selective naïve Bayes. Their accuracies do not greatly decrease as the number of irrelevant predictors grows. On the other hand, excepting our approach and selective naïve Bayes, there is an increment of selected predictors for data sets containing more irrelevant predictors.

The effect of redundant predictors is stronger. As a general rule, selective naïve Bayes exhibits lower accuracy in the presence of redundant predictors. The L_1-NB accuracy is the least affected by this issue, and, generally, it shows the best classification performance. Note that accuracy is very similar for ordinary naïve Bayes and weighted naïve Bayes. More impressive are the graphs considering the number of selected predictors. As expected, prefiltering does not satisfactorily handle redundancy. The more redundant predictors there are, the greater the number of selected predictors. On the other hand, the number of selected variables for L_1-NB and selective naïve Bayes barely fluctuates at around 3 predictors for all data sets, always selected from the original set of variables.

With regard to L_1/L_2-regularization of differences (van Gerven and Heskes, 2008), even for a moderate number of predictors, the gradient-based method that solves this

1 http://archive.ics.uci.edu/ml

problem becomes computationally intractable. For this reason, we left this method out of the complete test set. However, we ran it on some data sets to check that it is unable to discard redundant predictors. For example, the mean accuracy for the original *Soybean* data set is 0.98 (standard deviation equals 0.1). When adding p redundant predictors, accuracy falls to 0.79 (standard deviation equals 0.19). Even worse, up to 32 predictors (out of 42) are selected on average (standard deviation equals 11.8). Hence, redundant predictors are definitely not excluded.

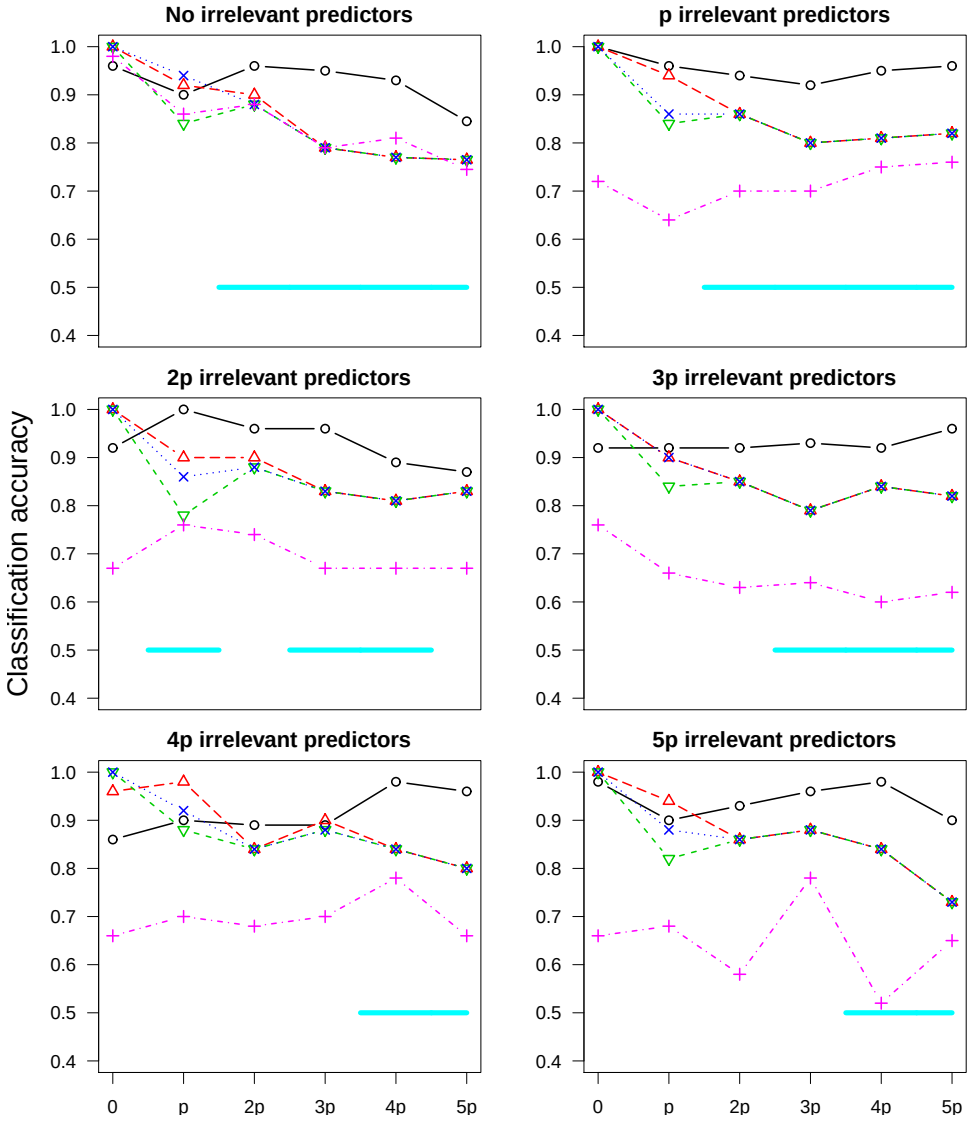


Figure 6.1: Classification accuracy (Y-axis) for increasing redundant predictors (X-axis). The solid-◯ line plots L_1-NB, the long-dashed-△ line plots ordinary naïve Bayes, the short-dashed-▽ line plots naïve Bayes with prefiltering feature selection, the dotted-× line plots weighted naïve Bayes with prefiltering feature selection and he dashed-dotted-+ line plots selective naïve Bayes.

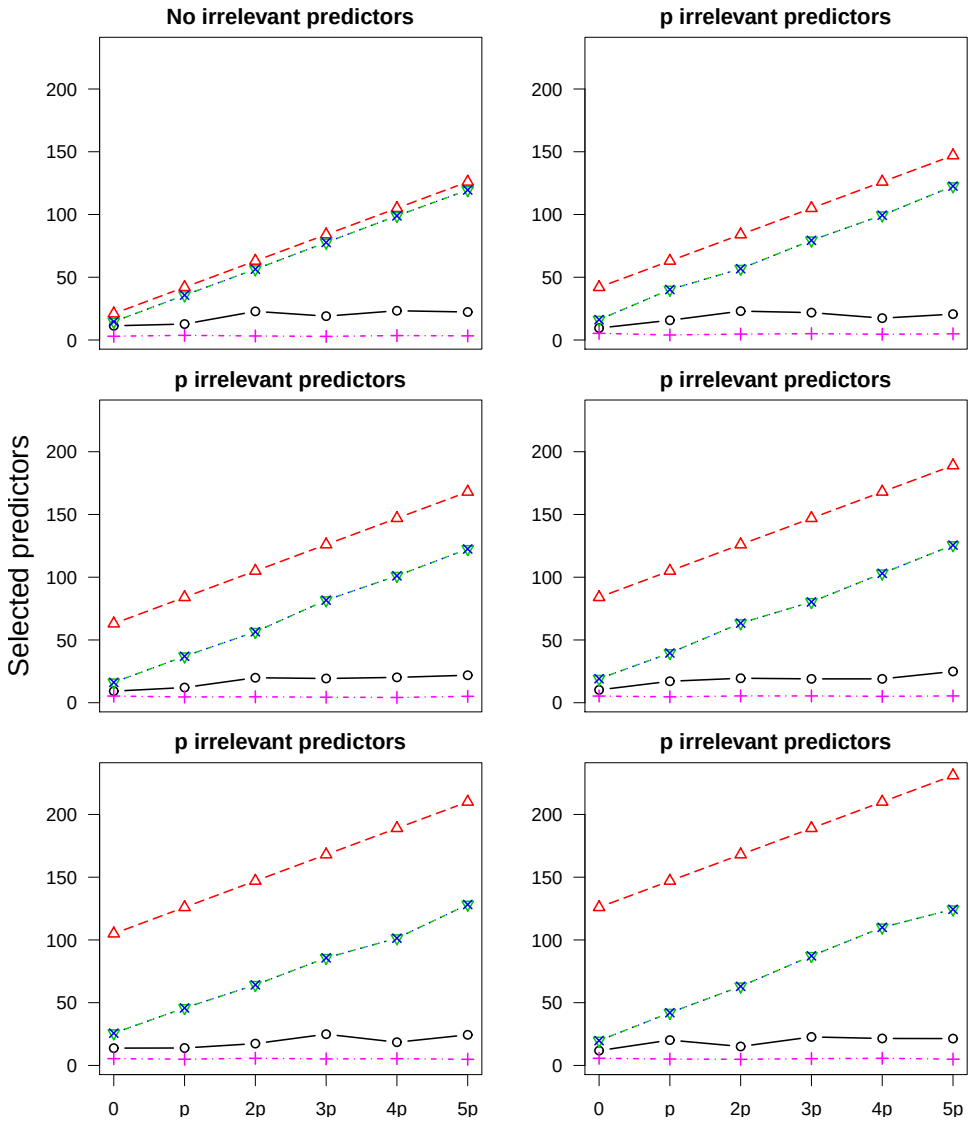


Figure 6.2: Number of selected predictors (Y axis) against increasing redundant predictors (X axis). The solid-◯ line plots L_1-NB, the long-dashed-△ line plots ordinary naïve Bayes, the short-dashed-▽ line plots naïve Bayes with prefiltering feature selection, the dotted-× line plots weighted naïve Bayes with prefiltering feature selection and the dashed-dotted-+ line plots selective naïve Bayes.

6.5.2 *High-dimensional data: brain imaging*

Next, we test the method in a high-dimensional setting The discrimination of mental states from neural activity is a hot topic in cognitive neuroscience. Data are usually high-dimensional, and efficiency is crucial in a pattern recognition analysis. Functional magnetic resonance imaging (fMRI) is of particular interest. Such data often contains thousands or even millions of predictors mapping 3D voxels.

Table 6.1: Mean accuracy (and standard deviation) and mean number of selected predictors (and standard deviation) for L_1-NB, discretized L_1-NB (dL_1-NB), naïve Bayes with pre-filtering feature selection (NB+FSS), discretized naïve Bayes with pre-filtering feature selection (dNB+FSS), weighted naïve Bayes with pre-filtering feature selection (WNB+FSS), discretized weighted naïve Bayes with pre-filtering feature selection (dWNB+FSS), selective naïve Bayes with pre-filtering feature selection (SNB+FSS) and discretized selective naïve Bayes with pre-filtering feature selection (dSNB+FSS). Best results are highlighted.

Method	Accuracy	#predictors
L_1-NB	$0.29(\pm 0.09)$	$511(\pm 130)$
dL_1-NB	$\mathbf{0.55}(\pm 0.13)$	$509(\pm 93)$
NB+FSS	$0.23(\pm 0.01)$	$914.1(\pm 770.1)$
dNB+FSS	$0.39(\pm 0.1)$	$3811.9(\pm 419.7)$
Method	Accuracy	#predictors
WNB+FSS	$0.23(\pm 0.01)$	$914.1(\pm 770.1)$
dWNB+FSS	$0.43(\pm 0.1)$	$3811.9(\pm 419.7)$
SNB+FSS	$0.19(\pm 0.08)$	$69.3(\pm 15)$
dSNB+FSS	$0.31(\pm 0.01)$	$\mathbf{61.4}(\pm 6.5)$

In this chapter we deal with a data set that considers visual stimuli (Haxby et al., 2002) as provided within the MVPA MatLab Toolbox[1]. A single subject is analyzed over 12 trials or runs. At each trial, the subject is shown pictures illustrating each of eight types of content (classes) for a length of time. A brain image is taken every few seconds. Each image is thus an instance, also referred to as repetition time (TR). In summary, at each trial, we have 9 TRs for each content type. Also, there are some TRs that does not match any content. We ignore these no-content TRs, so that $N = 12 \times 8 \times 9 = 864$ instances are available. There are $p = 39912$ voxels.

In addition to the proposed method, we have tested naïve Bayes, weighted naïve Bayes and selective naïve Bayes. We also trained the classifiers over a discretized version of the data set. Discretization conformed to the MDL-based scheme from (Fayyad and Irani, 1993).

All except L_1-NB were preceded by feature selection. For selective naïve Bayes, this is necessary on computational grounds. Again, we introduced a number of random predictors and discarded those predictors whose univariate relation with the class is weaker than one of the random predictors. For the original data set (with continuous predictors), we use the ANOVA p-value. For the discretized data set, we use mutual information to the class. The number of random predictors was chosen to maximize the accuracy on some validation data.

Taking advantage of the trial structure of the data set, we performed 12-fold cross-validation for all learning procedures, leaving out one trial at each iteration for testing. One run was also reserved for validation purposes (model selection and number of random predictors in the prefiltering step). Table 6.1 presents the results.

1 http://code.google.com/p/princeton-mvpa-toolbox

All methods behave much better on the discretized data set. The discretized L_1-NB method shows the best overall accuracy (0.55 ± 0.13). The difference between the best and the second-best accuracy is statistically significant according to a t-test. Of the Gaussian classifiers, L_1-NB also exhibits the best accuracy (0.29 ± 0.09). The difference between the best and the second-best Gaussian method is again statistically significant. On the other hand, L_1-NB selects a higher number of predictors than selective naïve Bayes. The number of selected predictors for L_1-NB is not, however, out of proportion, and interpretability is good. Note that, unlike L_1-NB, selective naïve Bayes is preceded by prefiltering. The number of selected predictors of the other naïve Bayes approaches are higher, specially when discretization is involved. This is probably due to the nature of the discretization process.

In the neuroscience domain, it is known that sparse brain areas are simultaneously activated under certain stimuli. The distributed nature of the brain is very closely related to redundancy from a pattern analysis perspective. This explains why L_1-NB is far better than the other approaches based on naïve Bayes.

It appears that the assumptions of building a Gaussian naïve Bayes model for fMRI data are too strong. This could be the cause of the poor performance of the classifiers for continuous data. The normality assumption for the predictors given the class is not always met. For example, if we take the voxel that is most correlated to the class and perform a Shapiro-Wilk hypothesis test to check normality within each class, we obtain 0.376, 0.276, 0.3608, 0.4819, 0.001, 0.565, 0.021 and 0.0579 p-values. For a p-value threshold of 0.05, the predictor does not follow a normal distribution within classes 5 and 7. Only 1045 out of 39912 voxels (a proportion of 0.025) fulfill the normality assumption within the eight classes. On average, voxels fulfill the normality assumption only within 2.2 classes. The frequently superior performance of the naïve Bayes classifiers when they are applied on discretized data was reported, e.g., in (Dougherty et al., 1995).

Summing up, we have shown that the proposed method is a flexible and accurate classifier, in both synthetic and real data sets. In particular, it deals with both numeric and continuous predictors. It has been demonstrated that naïve Bayes methods are not competitive against L_1-NB in data sets with a strong correlation between predictors.

6.6 DISCUSSION

So far, we have discussed the issue of irrelevant predictors and redundant predictors for the naïve Bayes model. We have proposed a model that, initially inspired by the naïve Bayes scheme, deals successfully with these spurious predictors.

This has been proved empirically on several data sets, where different numbers of irrelevant and redundant predictors have been added. As shown, our method works on both discrete and continuous data sets. Moreover, a high-dimensional setting, extracted from the neuroscience domain, has been tested. We found that the proposed method works much better on a discretized version of this data set. This is probably due to the particular features of fMRI data.

Like the naïve Bayes model, we have not explicitly considered dependence between predictors in this chapter. However, since the L_1-penalty deals with redundancy (as seen above, with regard to the loss function), we can discard redundant predictors.

7

CLASSIFICATION OF NEURAL SIGNALS FROM SPARSE AUTOREGRESSIVE FEATURES

7.1 INTRODUCTION

Brain-computer interfaces (BCI) establish a direct communication between the brain and some external device, where brain activity is decoded to control the device. This is a very promising application of machine learning techniques that, for example, can help to greatly improve the quality of life of disabled people.

Of BCI technologies, non-invasive BCIs, which use neuroimaging inputs collected without entering the body, are of particular interest. Some of the possible data sources are electroencephalography (EEG) data, magnetoencephalography (MEG) data and functional magnetic resonance imaging (fMRI) data. In this chapter, we focus on binary classification with EEG inputs. Therefore, data usually consist of a set of signals, each collected from a specific location on the scalp (called a channel). For a general review of EEG-based classification, see, for example, (Lotte et al., 2007).

Usually, some feature extraction procedure is used to assemble the classifying predictors from the raw signals. To number a few possibilities, we have time-frequency features (Wang et al., 2004), power spectral density values (Chiappa and Bengio, 2004) or the fitted autoregressive (AR) coefficients (Penny et al., 2000). We focus on AR coefficients.

We assume that each instance of the data set is constituted by a signal or a simultaneously recorded set of signals (each corresponding to a different channel). Typically, a vector of AR linear coefficients is estimated separately for each single signal by, for example, least squares or Burg's algorithm (Burg, 1967). These coefficients would be the inputs for the subsequent classifier. When more than one channel is available, the AR coefficients for all channels are concatenated to build a single instance. Hence, if a p-order AR model is fitted for each signal, we have p predictors per channel. Alternatively, the signal can be divided into various segments so that a p-order AR model is fitted for each segment. In this case, we would have a number of predictors equal to p multiplied by the number of segments, multiplied by the number of channels. This is done separately for each instance.

Huan and Palaniappan (2004) compare these two methods, estimating the AR coefficients using both the least squares method and Burg's algorithm. All their estimates are of sixth-order AR coefficients, which have been reported in the literature to empirically produce good results. They use either linear discriminant analysis or a multilayer perceptron for the binary classification step, concluding that the best feature extraction approach is the simplest least squares method of fitting AR models for entire signals. No variable selection is performed. Unfortunately, this type of methods entails two major drawbacks. First, the order of the AR models is fixed beforehand, without considering the data. Also, the same order is used for all channels. Second,

the AR coefficients are estimated by exclusively minimizing the AR prediction error, regardless of the classification performance.

We aim to overcome these pitfalls by choosing an initial arbitrarily high AR order and then looking for a sparse AR solution by means of L_1-regularization. Hsu et al. (2008) apply the lasso to AR models. The novelty of our approach is that, instead of making a separate AR estimate for each signal, we estimate the AR coefficients for all signals altogether. The objective is to discard (or select) the same variables for all signals. The selected variables can be different for each channel, and entire channels can even be discarded. We use the group lasso (Yuan and Lin, 2006), which can discard or select entire groups by means of a block L_1-penalization, to generate a sparse solution. This work appears in the submitted paper (Vidaurre et al., 2011a).

There are efficient algorithms to solve the group lasso, either finding the whole regularization path (Yuan and Lin, 2006) or computing the solution for a grid of different regularization parameter values (Meier et al., 2008). In this chapter, we devise an efficient LARS-type algorithm (Yuan and Lin, 2006; Efron et al., 2004) based on multiresponse linear regression that provides computational advantages for this particular problem. Whatever algorithm we use, we select the group lasso solution that maximizes some classifier-related measure. Since the classifier somehow guides the AR coefficient estimation, it is with the second aforementioned issue that we are concerned here.

Our method can be deemed a wrapper method. Wrapper methods are often computationally expensive. In this case, though, the number of predictors (selected AR coefficients) is moderate, so the computational cost is affordable.

7.2 BASIC METHODOLOGY

We consider N signals $\boldsymbol{z}_i = (z_{i1}, ..., z_{iT})^t$, $i \in \{1, ..., N\}$, each labeled as $c_i \in \{0, 1\}$. Let us denote the class vector as $\boldsymbol{c} \in \{0, 1\}^N$. We want to obtain a classifier that assigns any future signal $\boldsymbol{z}_i$, $i > N$, to a class in $\{0, 1\}$.

The autoregressive p-order model presumes, for each signal $\boldsymbol{z}_i$, that

$$z_{ij} = \beta_{i0} + \sum_{k=1}^{p} \beta_{ik} z_{i(j-k)} + \epsilon_j, \quad j \in \{p+1, ..., T\},$$

where ϵ_j is Gaussian white noise. Given some estimator $\hat{\boldsymbol{\beta}}_i = (\hat{\beta}_{i0}, \hat{\beta}_{i1}, ..., \hat{\beta}_{ip})^t$, the squared sum of autoregressive errors is defined for $\boldsymbol{z}_i$ as

$$SSE(\hat{\boldsymbol{\beta}}_i, \boldsymbol{z}_i) = \sum_{j=p+1}^{T} \left(z_{ij} - \hat{\beta}_{i0} - \sum_{k=1}^{p} \hat{\beta}_{ik} z_{i(j-k)} \right)^2.$$

Now, let us consider a classifier ψ and a function $f_\psi(\cdot, \cdot)$, which returns some classifier-related fitness measure, and whose arguments are, respectively, a set of inputs and a set of responses. Given a high enough order p, we jointly estimate the N sparse vectors of autoregressive coefficients, stacked in an $N \times p$ matrix $\boldsymbol{B} = [\hat{\boldsymbol{\beta}}_1^t, ..., \hat{\boldsymbol{\beta}}_N^t]$, as

$$\hat{\boldsymbol{B}} = \text{argmax}_{\boldsymbol{B}} f_\psi(\boldsymbol{B}, \boldsymbol{c}) \quad \text{s.t.} \quad \sum_{i=1}^{N} SSE(\hat{\boldsymbol{\beta}}_i, \boldsymbol{z}_i) < s,$$

where s is some positive constant. So, $\hat{\boldsymbol{B}}$ is the input of the classifier and is chosen to optimize the classification. Hence, we want to discard those coefficients that are

useless for the classification. Also, the discarded coefficients should be the same for all vectors $\hat{\boldsymbol{\beta}}_i$, $i \in \{1, ..., N\}$.

To estimate $\boldsymbol{B}$, we define the following elements:

$$\boldsymbol{y} = (z_{1(p+1)}, ..., z_{1T}, ..., z_{N(p+1)}, ..., z_{NT})^t,$$

$$\boldsymbol{X} = \begin{pmatrix}
1 & z_{11} & \dots & z_{1p} & 0 & 0 & \dots & 0 & \dots & 0 & 0 & \dots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
1 & z_{1(T-p)} & \dots & z_{1(T-1)} & 0 & 0 & \dots & 0 & \dots & 0 & 0 & \dots & 0 \\
0 & 0 & \dots & 0 & 1 & z_{21} & \dots & z_{2p} & \dots & 0 & 0 & \dots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \dots & 0 & 1 & z_{2(T-p)} & \dots & z_{2(T-1)} & \dots & 0 & 0 & \dots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \dots & 0 & 0 & \dots & 0 & \dots & \dots & 1 & z_{N1} & \dots & z_{Np} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \dots & 0 & 0 & \dots & 0 & \dots & \dots & 1 & z_{N(T-p)} & \dots & z_{N(T-1)}
\end{pmatrix},$$

$$\boldsymbol{\beta}^* = (\beta_{10}, \beta_{11}, ..., \beta_{1p}, ..., \beta_{N0}, \beta_{N1}, ..., \beta_{Np})^t,$$

where vector $\boldsymbol{\beta}^*$ is the concatenation of the elements of $\boldsymbol{B}$. Therefore, if we include the intercept as an additional predictor for each signal, we have $\boldsymbol{y} \in \mathbb{R}^{Nq}$, $\boldsymbol{X} \in \mathbb{R}^{Nq \times N(p+1)}$ and $\boldsymbol{\beta}^* \in \mathbb{R}^{N(p+1)}$, where $q = T - p$. Otherwise, we have $\boldsymbol{y} \in \mathbb{R}^{Nq}$, $\boldsymbol{X} \in \mathbb{R}^{Nq \times Np}$ and $\boldsymbol{\beta}^* \in \mathbb{R}^{Np}$. Here, we choose to include the intercept.

Assuming that the N signals are independent, we can establish the linear relation:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta}^* + \boldsymbol{\epsilon}, \tag{7.1}$$

where $\boldsymbol{\epsilon} = (\epsilon_{1(p+1)}, ..., \epsilon_{1T}, ..., \epsilon_{N(p+1)}, ..., \epsilon_{NT})^t$ is Gaussian white noise.

We can impose an L_1-penalty that would drive some AR coefficients to zero (depending on some regularization parameter λ). However, since we are interested in discarding the same coefficients for each signal, we use a group lasso penalty instead. The proposed formulation allows us to define p groups $\boldsymbol{\theta}_k = (\beta^*_{1k}, ..., \beta^*_{Nk})^t$, $k \in \{1, ..., p\}$, to estimate $\hat{\boldsymbol{\beta}}^*$ as the minimizer of

$$||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^*||_2^2 + \lambda \sum_{k=1}^{p} ||\boldsymbol{\theta}_k||_2, \tag{7.2}$$

where $||.||_2$ is the 2-norm. The intercepts $(\beta_{10}, ..., \beta_{N0})$ are not penalized.

This is a usual group lasso problem (Yuan and Lin, 2006), and the complete regularization path can be obtained by group LARS (Yuan and Lin, 2006; Efron et al., 2004) if $\boldsymbol{X}$ is orthogonal. Otherwise, the group LARS offers an approximation of the regularization path, which serves our purposes. However, even though group LARS is a very efficient method, $\boldsymbol{X}$ can become a huge matrix and computation can be expensive unless we exploit its sparse structure. Similar arguments have been followed in the multiresponse regression literature. Below, we introduce a forward selection approach that considers this structure. This algorithm is based on group LARS and the *multiresponse sparse regression* algorithm (Similä and Tikka, 2006).

Usually, we would choose, from the set of solutions, the solution minimizing the mean squared error (the first term in Equation (7.2)) on some testing data set, the solution minimizing some penalized criterion on the training data set or the solution minimizing the K-fold cross-validated mean square error. However, the rationale of our approach is to select the best solution according to the performance of the subsequent classifier. Thus, we propose to select the solution $\hat{\boldsymbol{\beta}}^*$ that maximizes $f_\psi(\hat{\boldsymbol{B}}, \boldsymbol{c})$, where $\hat{\boldsymbol{B}}$ is the vector $\hat{\boldsymbol{\beta}}^*$ in matrix form as defined above.

Linear discriminant analysis (LDA) logistic regression (LR) or support vector machines (SVM) are all possible options for the classifier ψ; see (Hastie et al., 2008) for a general review. LDA and LR are linear in the most basic version, whereas SVM, which constructs a linear boundary in a transformed version of the feature space, is nonlinear. We give now a brief description of these methods; a complete overview can be found in Chapter 1.

For LDA, $f_\psi(\hat{\boldsymbol{B}}, \boldsymbol{c})$ is naturally defined as the log-likelihood function:

$$f_\psi(\hat{\boldsymbol{B}}, \boldsymbol{c}) = \sum_{i=1}^{N} \Big(-\hat{\boldsymbol{\beta}}_i^t \hat{\boldsymbol{\Sigma}}^{-1} \hat{\boldsymbol{\mu}}_{c_i} + \frac{1}{2} \hat{\boldsymbol{\mu}}_{c_i}^t \hat{\boldsymbol{\Sigma}}^{-1} \hat{\boldsymbol{\mu}}_{c_i} + log\, \hat{\pi}_{c_i} \Big), \tag{7.3}$$

where $\hat{\boldsymbol{\mu}}_{c_i}$ is the mean of those vectors $\boldsymbol{\beta}_i$ whose class is c_i, $\hat{\boldsymbol{\Sigma}}$ is the common covariance matrix of $\hat{\boldsymbol{B}}$ and $\hat{\pi}_{c_i}$ is the estimated a priori probability of class c_i. We can estimate π_{c_i} as $\frac{N_{c_i}}{N}$, where N_{c_i} is the number of instances whose class is c_i. To compute (7.3), we remove the coefficients that correspond to dropped groups (since $\hat{\boldsymbol{\beta}}_i^*$ is sparse by groups). Note that this is necessary to compute $\hat{\boldsymbol{\Sigma}}$. Otherwise, matrix $\hat{\boldsymbol{B}}$ has columns with all elements equal to zero and is not full rank.

For LR, $f_\psi(\hat{\boldsymbol{B}}, \boldsymbol{c})$ can also be the log-likelihood function, defined as

$$f_\psi(\hat{\boldsymbol{B}}, \boldsymbol{c}) = \sum_{i=1}^{N} \Big(c_i \hat{\boldsymbol{w}}^t \hat{\boldsymbol{\beta}}_i - log(1 + e^{\hat{\boldsymbol{w}}^t \hat{\boldsymbol{\beta}}_i}) \Big), \tag{7.4}$$

where $\hat{\boldsymbol{w}} \in \mathcal{R}^p$ is the estimated vector of logistic regression coefficients, computed by the iteratively reweighted least squares algorithm (IRLS).

In order to avoid overfitting, we select the solution of Equation (7.2) that minimizes a penalized version of $f_\psi(\hat{\boldsymbol{B}}, \boldsymbol{c})$. In particular, since Equations (7.3) and (7.4) are log-likelihood functions, we can employ the *Akaike information criterion* (Akaike, 1974):

$$AIC = -\frac{2}{N} f_\psi(\hat{\boldsymbol{B}}, \boldsymbol{c}) + 2\frac{df}{N}, \tag{7.5}$$

where df is the number of columns of $\hat{\boldsymbol{B}}$ with non-zero coefficients.

Finally, for SVM, $f_\psi(\hat{\boldsymbol{B}}, \boldsymbol{c})$ can be defined as some margin maximizing loss function. For convenience, we redefine the class to be in $\{-1, 1\}$. One possible formulation of the SVM estimates the separating hyperplane parameters $\boldsymbol{w} \in \mathbb{R}^{p^+}$ as the minimizer of

$$\sum_{i=1}^{N} \Big(1 - c_i h(\boldsymbol{\beta}_i)^t \boldsymbol{w} \Big)_+ + \alpha\, ||\boldsymbol{w}||^2, \tag{7.6}$$

where $(\cdot)_+$ indicates the positive part, $h(\cdot)$ is some mapping function, p^+ is the dimension of the expanded feature space and $\alpha > 0$ is the regularization cost parameter. Hence, function $h(\cdot)$ gives the nonlinear power to the SVM. Typically, a kernel function that computes the distance between any two points in the expanded feature space defined by $h(\cdot)$ is all we need for an efficient computation. In this chapter, we use a radial basis function kernel.

The entire regularization path for Equation (7.6) can be computed with a small multiple of the computational cost of fitting an SVM model for a single α parameter (Hastie et al., 2004). From this regularization path, we select the model that minimizes the K-fold cross-validated error. Here, $f_\psi(\hat{\boldsymbol{B}}, \boldsymbol{c})$ can be the value of the left term of Equation (7.6) that corresponds to the selected model. Since we use a cross-validated estimation of the loss function, we do not need to use a penalization such as AIC in Equation (7.5).

Whereas the LDA formulation can be used for multiclass classification, the LR and SVM expressions are for binary classification but can be easily generalized. See (Lee et al., 2004), for example, for details about the multiclass SVM.

7.3 MULTIPLE CHANNEL CLASSIFICATION

An instance is usually defined as a group of signals instead of just one signal. For instance, an EEG instance is usually a set of signals recorded from different points on the scalp, that is, from different channels. Here, it makes sense to consider that the same AR coefficients are perhaps not appropriate for all channels.

Let us modify the above notation to accommodate this problem. We consider N sets of signals, each with M channels, and labeled as $\boldsymbol{c} \in \{0,1\}^N$ as before. Now, each signal is denoted as $\boldsymbol{z}_{il} = (z_{il1}, ..., z_{ilT})$, $i \in \{1, ..., N\}$, $l \in \{1, ..., M\}$. We define

$$\boldsymbol{y} = (z_{11(p+1)}, ..., z_{11T}, ..., z_{1M(p+1)}, ..., z_{1MT}, ..., z_{NM(p+1)}, ..., z_{NMT})^t,$$

$$\boldsymbol{X}_{il} = \begin{pmatrix} 1 & z_{il1} & \dots & z_{ilp} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & z_{il(T-p)} & \dots & z_{il(T-1)} \end{pmatrix}, \quad \boldsymbol{X} = \begin{pmatrix} \boldsymbol{X}_{11} & \dots & \boldsymbol{0} & \dots & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{0} & \dots & \boldsymbol{X}_{1M} & \dots & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{0} & \dots & \boldsymbol{0} & \dots & \boldsymbol{X}_{N1} & \dots & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{0} & \dots & \boldsymbol{0} & \dots & \boldsymbol{0} & \dots & \boldsymbol{X}_{NM} \end{pmatrix},$$

$$\boldsymbol{\beta}^* = (\beta_{111}, ..., \beta_{11p}, ..., \beta_{1M1}, ..., \beta_{1Mp}, ..., \beta_{NM1}, ..., \beta_{NMp})^t,$$

where $\boldsymbol{y} \in \mathbb{R}^{NMq}$, $\boldsymbol{X}_{il} \in \mathbb{R}^{q \times p+1}$, $\boldsymbol{X} \in \mathbb{R}^{NMq \times NM(p+1)}$ and $\boldsymbol{\beta}^* \in \mathbb{R}^{MN(p+1)}$. As before, $q = T - p$. The same linear relation as Equation (7.1) can be applied here.

Now, we have two choices to define the groups. We can either define p groups as

$$\boldsymbol{\theta}_k = (\beta_{11k}, ..., \beta_{1Mk}, ..., \beta_{N1k}, ..., \beta_{NMk})^t, \quad k \in \{1, ..., p\}, \tag{7.7}$$

or Mp groups as

$$\boldsymbol{\theta}_{lk} = (\beta_{1lk}, ..., \beta_{Nlk})^t, \quad l \in \{1, ..., M\}, \quad k \in \{1, ..., p\}. \tag{7.8}$$

Equation (7.7) defines the same AR coefficients for all the channels, whereas Equation (7.8) defines adaptive AR coefficients for each channel. The group lasso formulation in Equation (7.2) is unchanged in the first case. In the second case, it becomes

$$||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^*||_2^2 + \lambda \sum_{k=1}^{p} \sum_{l=1}^{M} ||\boldsymbol{\theta}_{lk}||_2. \tag{7.9}$$

The solutions can be used as inputs for the classifier ψ as before, and, specifying for this classifier, the solution $\hat{\boldsymbol{\beta}}^*$ that maximizes $f_\psi(\hat{\boldsymbol{B}}, \boldsymbol{c})$ is selected. Here, $\hat{\boldsymbol{B}} = [\hat{\boldsymbol{\beta}}_1^T, ..., \hat{\boldsymbol{\beta}}_N^T]$, where $\hat{\boldsymbol{\beta}}_i = (\beta_{i10}, \beta_{i11}, ..., \beta_{i1p}, ..., \beta_{iM0}, \beta_{iM1}, ..., \beta_{iMp})^t$.

Since there is no reason to assume that all the channels contribute equally, we prefer the second grouping approach. This way, we can exclude entire channels that are not of interest, and possibly gather biological insight from this fact.

Besides, this approach would keep just one of a redundant set of channels (with similar signals), if any, provided it is relevant to the classification. This is a well-known lasso property. In the methods followed by Huan and Palaniappan (2004), among others, redundant channels would affect the classification performance and could lead to an ill-posed problem.

In this section, we briefly describe the group LARS (Efron et al., 2004) and the multiresponse sparse regression algorithm (Similä and Tikka, 2006). We then introduce an efficient algorithm to find an approximate regularization path for Equations (7.2) and (7.9). Although we define the method for Equation (7.9), it can be straightforwardly adapted to Equation (7.2). We assume centered data (no intercepts) for notation simplicity.

The group LARS algorithm is an iterative procedure that adds a set of predictors, or group, to the model at each step. The group LARS starts with no groups. Firstly, it adds the group that is most correlated with the response to the active set of groups $\mathcal{A}$. The response is regressed on this group, moving the coefficients of this group towards the least squares solution until a new group reaches the same correlation with the vector of residuals as the active set. This new group is added to the active set $\mathcal{A}$. Now, the vector of residuals is regressed on the groups in $\mathcal{A}$, moving their coefficients towards the joint least squares solution until a new group reaches the same correlation with that vector of residuals as the active set. When the number of instances is greater than the number of predictors, this procedure is repeated until all predictors are in the model. With a small modification, the group LARS can find the entire regularization path of a group lasso problem if the design matrix is orthogonal. Otherwise, it provides an approximate solution.

The multiresponse sparse regression algorithm extends the (group) LARS algorithm to multiresponse linear regression by modifying the correlation criterion between the predictors and the current residual, which depends on multiple outputs.

Here, we derive an algorithm based on group LARS and multiresponse sparse regression for efficiently computing an approximate set of solutions of Equations (7.2) and (7.9). In the sequel, we will use the notation $\boldsymbol{X}_{il}$, $i \in \{1, ..., N\}$, $l \in \{1, ..., M\}$. Groups are defined in Equation (7.8). The active set $\mathcal{A}$ is thus defined by a set of pairs (l, k), $l \in \{1, ..., M\}$, $k \in \{1, ..., p\}$. We also define $\boldsymbol{y}_{il} = (\boldsymbol{z}_{il(p+1)}, ..., \boldsymbol{z}_{ilT})^t$ and $\boldsymbol{\beta}_{il} = (\boldsymbol{\beta}_{il1}, ..., \boldsymbol{\beta}_{ilp})^t$.

The algorithm follows the group LARS steps described by Yuan and Lin (2006), with some modifications. First, we devise a new correlation measure. We define the correlation of the lk-th group with the current residual as

$$\rho_{lk} = \sum_{i=1}^{N} \left(\boldsymbol{r}_{il}^t \boldsymbol{X}_{il}^{(k)} \right)^2,$$

where $\boldsymbol{X}_{il}^{(k)}$ is the k-th column of $\boldsymbol{X}_{il}$ and $\boldsymbol{r}_{il} \in \mathbb{R}^q$ is the current residual for the i-th signal and the l-th channel, defined as

$$\boldsymbol{r}_{il} = \boldsymbol{y}_{il} - \boldsymbol{X}_{il} \hat{\boldsymbol{\beta}}_{il}.$$

Second, the joint least squares solution $\boldsymbol{d}_{il} \in \mathbb{R}^p$ is computed separately for the i-th signal and the l-th channel. Setting to zero the elements of $\boldsymbol{d}_{il}$ that are not in $\mathcal{A}$, we compute the remainder as

$$\boldsymbol{d}_{il}^{\mathcal{A}} = (\boldsymbol{X}_{il}^{\mathcal{A}^t} \boldsymbol{X}_{il}^{\mathcal{A}})^{-1} \boldsymbol{X}_{il}^{\mathcal{A}^t} \boldsymbol{r}_{il},$$

where $\boldsymbol{X}_{il}^{\mathcal{A}}$ denotes the columns of $\boldsymbol{X}_{il}$ indexed by the active set $\mathcal{A}$. Hence, the regression coefficients for the i-th signal and the l-th channel are updated at each step as

$$\boldsymbol{\beta}_{il} = \boldsymbol{\beta}_{il} + \gamma \boldsymbol{d}_{il},$$

where the $\gamma \in [0, 1]$ constant is computed as

$$\gamma = \min_{\gamma_{lk}} \quad \text{s.t.} \tag{7.10}$$
$$\sum_{i=1}^{N} \left[\boldsymbol{X}_{il}^{(k)^t} (\boldsymbol{r}_{il} - \gamma_{lk} \boldsymbol{X}_{il} \boldsymbol{d}_{il}) \right]^2 = \sum_{i=1}^{N} \left[\boldsymbol{X}_{il'}^{(k')^t} (\boldsymbol{r}_{il'} - \gamma_{lk} \boldsymbol{X}_{il'} \boldsymbol{d}_{il'}) \right]^2,$$

Here, (l', k') are a pair of indexes arbitrarily chosen from $\mathcal{A}$. Basic algebraic manipulations lead to

$$\gamma_{il} = \frac{-v + v' \pm \sqrt{v^2 + v'^2 - 2vv' - 4ub + 4u'b'}}{2(u - u')},$$

where we define

$$u = \sum_{i=1}^{N} (\boldsymbol{X}_{il}^{(k)^t} \boldsymbol{X}_{il} \boldsymbol{d}_{il})^2, \qquad u' = \sum_{i=1}^{N} (\boldsymbol{X}_{il'}^{(k')^t} \boldsymbol{X}_{il'} \boldsymbol{d}_{il'})^2,$$
$$v = 2 \sum_{i=1}^{N} (\boldsymbol{X}_{il}^{(k)^t} \boldsymbol{r}_{il})(\boldsymbol{X}_{il}^{(k)^t} \boldsymbol{X}_{il} \boldsymbol{d}_{il}), \qquad v' = 2 \sum_{i=1}^{N} (\boldsymbol{X}_{il'}^{(k')^t} \boldsymbol{r}_{il'})(\boldsymbol{X}_{il'}^{(k')^t} \boldsymbol{X}_{il'} \boldsymbol{d}_{il'}),$$
$$b = \sum_{i=1}^{N} (\boldsymbol{X}_{il}^{(k)^t} \boldsymbol{r}_{il})^2, \qquad b' = \sum_{i=1}^{N} (\boldsymbol{X}_{il'}^{(k')^t} \boldsymbol{r}_{il'})^2.$$

Thus, the indexes $(l, k) \notin \mathcal{A}$ that minimize Equation (7.10) correspond to the group that is added to the active set in the next iteration. As with group LARS, we now update the residual

$$\boldsymbol{r}_{il} = \boldsymbol{y}_{il} - \gamma \boldsymbol{X}_{il} \boldsymbol{\beta}_{il} \quad \forall il.$$

This procedure is repeated until $\gamma = 1$.

It turns out that the group LARS solution is the same yielded for the multiresponse sparse regression algorithm. The same connection holds for the algorithm that we propose. Unlike group LARS, however, we do not need to store an $NMq \times NM(p+1)$ matrix in memory (in the multiple channel case), speeding up the computations. Since $\boldsymbol{X}$ is not an orthogonal matrix, this solution is only approximated. At the cost of storing the entire matrix $\boldsymbol{X}$ in memory, an exact group lasso regularization path can be computed following the algorithm described in recent work by Friedman et al. (2010b). In this chapter, however, we do not follow this approach.

7.5 FURTHER COMPUTATIONAL ISSUES

We deal with the multiple channel classification problem, because it is both more frequent and more computationally demanding. We can split our method in two steps: the preprocessing group lasso step and the classification step.

Whereas we use quick algorithms on data matrices of at most $N \times Mp$ elements at the classification step, the preprocessing step deals with $NMq \times NMp$ design matrices. Even considering that the group LARS algorithm is fast (and so are other algorithms

solving the group lasso problem, e.g. (Meier et al., 2008)), the dimension of the design matrix can be huge. Hence, the computation effort, both from time and memory perspectives, is clearly greater at the first step.

To overcome this limitation, some approaches can be followed.

First, if the number of channels is high, a possible heuristic approximation is to run the algorithm for each channel, separately, following the single-channel approach described above. Using some statistical test, we would discard the channels whose classification accuracy is low enough. Keeping only the most accurate channels, we would let them be the entrance for the multiple channel algorithm. In the machine learning jargon, this is a prefiltering step previous to the wrapper algorithm.

Second, if the signals are long, we can perform a convolution process to extract shorter signals, keeping the low frequences and removing the high frequences, which are probably due to noise, from the original signal. For instance, we can make use of the discrete wavelet transform to divide the signal length by any power of two.

Third, although big, the design matrix is significantly sparse. More precisely, only $NMpq$ out of N^2M^2pq elements are different from zero. Such circumstance can be utilized to accelerate computations and save space in memory. For example, the R library `Matrix` implements some classes and methods for sparse matrices.

Sometimes a fast algorithm is needed to run in a real-time (online) environment with very limited computational resources available. Some BCI applications are of this type. Such an approach can be hard to apply in the high-dimensional setting. However, offline (previously stored) data can be used to simplify the online task. Offline data are usually available in the BCI field, where the device has to be trained for each subject prior to its real use. For instance, we can run the algorithm on the offline data set so as to select which channels and AR coefficient indexes are relevant. Afterwards, in the online phase, we can use least squares to estimate the AR coefficients that correspond to the previously selected channels and AR coefficient indexes.

This two-step procedure follows the spirit of the relaxed lasso (Meinshausen, 2007). The relaxed lasso firstly discovers the sparsity pattern by lasso. Then, either least squares or the lasso, with a small penalty (i.e., with no variable selection), is used just on the selected variables. Among other nice theoretical properties, the relaxed lasso is less biased than the original lasso.

7.6 EXPERIMENTS

To test the proposed framework, we use the EEG data recorded by Zak Keirn at Purdue University (Keirn and Aunon, 1990). The data is a collection of experiments on seven different subjects. Each subject is told to perform five different mental activities, say: stay relaxed, solve a mathematics problem, write a letter, mentally rotate a geometric figure and count a series of made-up numbers. Each subject repeats each task for a number of trials. Specifically, subjects 1, 3, 4 and 6 perform ten trials, subjects 2 and 7 perform five trials and subject 5 performs fifteen trials. Each trial lasts ten seconds with a sampling frequency of 250 Hz, that is, it has a total of 2500 sample points per channel. EEG signals were recorded from seven channels. The positions are defined by the 10-20 system of electrode placement. Keirn and Aunon (1990) describe the collection procedure in more detail.

We use these data to test our signal classifier on the ten possible pairwise activity combinations, that is, on ten binary classification problems. The ten pairwise combinations are listed in the first column of Table 7.1. Binary classification was performed

on these data for example by Keirn and Aunon (1990) and Huan and Palaniappan (2004). In both papers, the trials are divided into subperiods that make up the instances for classification. Keirn and Aunon (1990) use two-second subperiods and train a unique model for all subjects. Huan and Palaniappan (2004), instead, use half-second subperiods, and have 20 available instances for each subject, trial and activity, with $2500/20 = 125$ sample points per channel. Huan and Palaniappan (2004) estimate individual models for each subject, using ten trials. Subjects performing only five trials are ignored, as are the last five trials of the subject who performs fifteen trials.

In this work, like Huan and Palaniappan (2004), we train individual models for each subject. To be able to use all available data in equilibrated conditions, we consider each group of five trials as a different data set. Thus, there are thirteen five-trials sets: two for subjects 1, 3, 4 and 6, one for subjects 2 and 7 and three for subject 5. We omitted one set (from subject 4) due to missing data. Therefore, for each pair of activities (each binary classification problem), we obtain twelve different, individual models.

We divide each trial into ten one-second subperiods of 250 sample points. This way, we have ten instances per trial. Since we have five trials for each of two activities, we have 100 instances for training and testing for each experiment. We use a maximum AR order of 15. Then, we have $T = 250$, $N = 100$, $M = 7$, $p = 15$ and $q = T - p = 235$. In this setting, we test the basic methodology explained above, using Equation (7.9), and the algorithm devised in the last subsection. We compare our approach (using also this setting) with the best feature extraction method reported by Huan and Palaniappan (2004), that is, the sixth-order AR coefficients computed by least squares. For the classification methods, we use LDA, LR and SVM.

On the one hand, Table 7.1 shows the mean classification accuracy for each method and each combination of activities, averaging across all thirteen five-trials sets. These results give an idea of the global performance of each method for each binary classification problem. On the other hand, Table 7.2 gives the best accuracy for each five-trials set, reporting which pairwise combination of activities and classification algorithm produced this result. The left columns show the best of our methods, and the right columns show the best of Huan and Palaniappan's methods. In a practical scenario, the pairwise combination of activities and the algorithm that best discriminate for a given subject would be chosen to implement the customized BCI device for this subject. All results are obtained by 5-fold cross-validation.

In both tables, the methods related to our approach are referred to as sparse LDA (sLDA), sparse logistic regression (sLR) and sparse SVM (sSVM). The methods related to the AR coefficients feature extraction approach are referred to as LDA, LR and SVM. Best results are highlighted. Statistical significance is checked by means of the t-test, so that the symbol $*$ is added when the difference between the best and the second best method is statistically significant with a significance level of 0.05. In Table 7.2, each five-trials set is identified by a number indicating the subject (1,...,7) and a letter (a,b) indicating the five-trials set within this subject.

As observed, the devised method outperforms the best method reported by Huan and Palaniappan (2004) in most experiments. The biggest differences can be observed in Table 7.2, where the best pair of activities and method is selected for each five-trials set. Interestingly, the SVM classifier offers the best results, possibly indicating that, in this scenario, the classification can be enhanced by appropriate nonlinear modeling.

Table 7.1: Summary of the classification accuracy for each combination of activities and methods.

	Accuracy ± std. deviation					
Activities	sLDA	sLR	sSVM	LDA	LR	SVM
Relax,maths	0.70 ± 0.07	$\mathbf{0.73} \pm 0.06$	$\mathbf{0.73} \pm 0.10$	0.70 ± 0.06	0.69 ± 0.05	$\mathbf{0.73} \pm 0.08$
Relax,letter	0.62 ± 0.08	0.63 ± 0.09	$\mathbf{0.68} \pm 0.10$	0.63 ± 0.07	0.59 ± 0.07	$\mathbf{0.68} \pm 0.08$
Relax,rotate	0.71 ± 0.10	0.74 ± 0.06	$\mathbf{0.78} \pm 0.10*$	0.70 ± 0.09	0.67 ± 0.07	0.73 ± 0.10
Relax,count	0.64 ± 0.12	0.65 ± 0.05	$\mathbf{0.71} \pm 0.10$	0.66 ± 0.09	0.65 ± 0.07	0.70 ± 0.08
Maths,letter	0.65 ± 0.10	0.66 ± 0.07	$\mathbf{0.73} \pm 0.10$	0.71 ± 0.06	0.67 ± 0.05	$\mathbf{0.73} \pm 0.07$
Maths,rotate	0.69 ± 0.07	0.69 ± 0.09	$\mathbf{0.73} \pm 0.08$	0.70 ± 0.06	0.65 ± 0.04	$\mathbf{0.73} \pm 0.10$
Maths,count	0.67 ± 0.08	0.68 ± 0.09	$\mathbf{0.70} \pm 0.10$	0.66 ± 0.09	0.64 ± 0.10	0.69 ± 0.08
Letter,rotate	0.75 ± 0.12	0.74 ± 0.10	$\mathbf{0.83} \pm 0.12*$	0.74 ± 0.10	0.70 ± 0.10	0.77 ± 0.14
Letter,count	0.66 ± 0.08	0.70 ± 0.10	$\mathbf{0.73} \pm 0.12$	0.65 ± 0.07	0.61 ± 0.07	0.71 ± 0.08
Rotate,count	0.65 ± 0.10	0.65 ± 0.10	0.71 ± 0.10	0.67 ± 0.11	0.64 ± 0.08	$\mathbf{0.72} \pm 0.10$

Table 7.2: Accuracy of the best combination of activities for each five-trials set.

Subject	Accuracy	Method	Activities	Accuracy	Method	Activities
1a	$\mathbf{0.93} \pm 0.05$	sSVM	Maths,rotate	0.85 ± 0.06	LDA	Relax,count
1b	0.85 ± 0.06	sSVM	Relax,rotate	$\mathbf{0.89} \pm 0.05$	LR	Relax,maths
2a	$\mathbf{0.92} \pm 0.08$	sSVM	Relax,rotate	$\mathbf{0.92} \pm 0.06$	SVM	Letter,rotate
3a	$\mathbf{0.86} \pm 0.05*$	sSVM	Letter,rotate	0.71 ± 0.11	LR	Relax,maths
3b	$\mathbf{0.93} \pm 0.12*$	sSVM	Letter,rotate	0.82 ± 0.08	SVM	Letter,rotate
4a	$\mathbf{0.94} \pm 0.20*$	sLR	Relax,maths	0.87 ± 0.09	SVM	Relax,rotate
5a	$\mathbf{0.99} \pm 0.02*$	sSVM	Letter,rotate	0.86 ± 0.10	SVM	Letter,rotate
5b	$\mathbf{0.81} \pm 0.06*$	sLDA	Letter,rotate	0.73 ± 0.09	LDA	Maths,letter
5c	0.75 ± 0.60	sLR	Maths,count	$\mathbf{0.77} \pm 0.08$	SVM	Relax,rotate
6a	0.88 ± 0.10	sSVM	Relax,rotate	$\mathbf{0.95} \pm 0.04*$	SVM	Letter,rotate
6b	$\mathbf{0.96} \pm 0.20$	sLDA	Relax,rotate	0.9 ± 0.04	SVM	Relax,rotate
7a	$\mathbf{0.92} \pm 0.02$	sSVM	Relax,rotate	0.9 ± 0.08	SVM	Letter,rotate

7.7 DISCUSSION

In this chapter, we have proposed a new feature extraction method based on sparse autoregressive features for multiple signal classification. We have applied the method, together with different classification algorithms, to an EEG signal classification problem, and compared its performance to a state-of-the-art AR feature extraction approach.

Our method excels when the underlying model is sparse, for example when some channels are irrelevant, redundant or very noisy. Besides, the performance benefits from the fact that feature extraction is guided by some classification-related measure. Moreover, we do not need to previously estimate the order of the AR models, because, starting from a high enough order, model selection is completely data driven. Finally, thanks to regularization, our approach is applicable to data sets with few instances, whereas other methods might suffer from overfitting in the same scenario.

The drawback of our method is that, since we altogether estimate the AR coefficients of all signals, it can be computationally demanding. However, we have provided several shortcuts to overcome this problem. As future work, we plan to develop more direct ways to reduce the complexity. In particular, we want to find a proper factorization of the sparse matrix $\boldsymbol{X}$, defined above, that turns out to be useful for the group lasso estimation.

Note that the proposed approach can be classed in the semisupervised classification paradigm. In semisupervised classification, there is typically a considerable amount of data, but only a portion is labeled. To make the most of the data, it is beneficial to also use the information provided by the unlabeled data. In our particular case, if we have a collection of unlabeled EEG signals, they can be included in the group lasso estimation of the AR coefficients (Equations (7.2) and (7.9)), even though the posterior model selection relies on a fully supervised classification algorithm, that is, only on the labeled data.

Part IV

REGULARIZATION FOR GRAPHICAL MODELS INDUCTION

8

A REGULARIZED ESTIMATION OF GAUSSIAN BAYESIAN NETWORKS

This section deals with the estimation of directed graphical models on continuous data, where we assume the variables to be Gaussian-distributed.

It is based on the published paper by Vidaurre et al. (2010). A previous version of this methodology was presented in the workshop "Sparse Optimization and Variable Selection" of the 2008 edition of the International Conference of Machine Learning (ICML). For this method to be optimal, we must assume the lasso to be an optimal variable selector. This theoretical question was discussed in Chapter 1.

8.1 LEARNING AN l_1-REGULARIZED GAUSSIAN BAYESIAN NETWORK IN THE EQUIVALENCE CLASS SPACE

8.2 INTRODUCTION

A Gaussian Bayesian network (GBN) (Geiger and Heckerman, 1994) is a probabilistic graphical model that encodes a joint Gaussian density, $f(\boldsymbol{X})$, on a p-dimensional random variable $\boldsymbol{X} = (X_1, \ldots, X_p)$:

$$f(\boldsymbol{x}) \equiv \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\Big(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\Big), \tag{8.1}$$

where $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_p)^T$ is the vector of means and $\boldsymbol{\Sigma}$ is the $p \times p$ covariance matrix. In a GBN, the density function of the joint distribution can be expressed as the product of p univariate normal densities defined as

$$f_i(x_i|\boldsymbol{pa}(x_i)) \equiv N(m_i + \sum_{x_j \in \boldsymbol{pa}(x_i)} \beta_{ji}(x_j - m_j), v_i). \tag{8.2}$$

The variance is v_i and the mean is composed of subparameters m_i and $\boldsymbol{\beta_i} = (\beta_{1i}, \ldots, \beta_{li})^T$, where l is the number of parents of variable X_i, denoted by $\boldsymbol{Pa}(X_i)$. β_{ji} is the linear regression coefficient of X_j in the regression of X_i on $\boldsymbol{Pa}(X_i)$. It represents how strong the relationship between X_i and X_j is; if $\beta_{ji} = 0$, then X_j will not be a parent of X_i. Multivariate normal density and the product of normal densities given in equation (8.2) are alternative and interchangeable representations.

There are two basic approaches to GBN structure learning from data: algorithms based on constrained methods and *score+ search* methods. Constraint-based approaches build networks that fulfill the conditional independences estimated from data (Smith and Whittaker, 1998). The conditional independences between variables are checked by means of statistical tests. A recent method of this kind was developed by Margaritis (2005).

On the other hand, score+search algorithms are founded on a scoring function for network evaluation in an attempt to find the model that best fits the data. The methods suggested in (Bach and Jordan, 2003; Davies, 2002) are two examples of score+search algorithms. A representation, a scoring function and a search strategy have to be defined.

First, to represent the solutions and move in the search space, we typically choose between directed acyclic graphs (DAGs), partial directed acyclic graphs (PDAGs) or variable orderings. Variable orderings are an intermediate representation that must be mapped to a graph to be meaningful. An equivalence class, modelled by a PDAG, is the set of graphs with the same conditional independences, encoding a unique probability density. Equivalence classes are often the preferred representation. They are the only representation capable of meeting the inclusion boundary requirement (Kočka et al., 2001; Chickering, 2002b) described below. In the PDAG, compelled arcs (arcs with the same orientation for all the members in the class) are modelled by directed arcs; the others are represented by undirected edges.

Second, assuming that we have a data set $\mathscr{D}$ of size N, to measure how well the model fits the data, the likelihood of the model given the data is defined as

$$L(\mathscr{D}; \boldsymbol{v}, \boldsymbol{m}) = \prod_{i=1}^{p}\prod_{n=1}^{N} \frac{1}{\sqrt{2\pi v_i}} \exp\Big(-\frac{1}{2v_i}(x_{ni} - m_i - \sum_{x_j \in \boldsymbol{pa}(x_i)} \beta_{ji}(x_{nj} - m_j))^2\Big).$$

A penalized scoring criterion, made up of the likelihood function and a penalization term that favors simple models, is usually employed. An example of such a criterion is minimum description length (MDL). A scoring criterion is score equivalent if it returns the same value for all the members inside the class. Sometimes, a unique orientation for all arcs is needed to provide an exact causal semantic to the network. When the density function itself, without causality implications, is of interest, to be score equivalent is a positive property.

Finally, the third element is the search strategy. Whether it is better to search in equivalence class spaces or alternatively in DAG spaces is still an open question. Some researchers think that it is not always worthwhile to work with equivalence classes because it is more complex. Gillispie and Perlman (2001) analyzed the expected number of DAGs inside an equivalence class. They found that on average this number is not big. This could be an argument in favour of DAG spaces. However, the variance is high, and, potentially, there could be a huge number of DAGs inside some equivalence classes, even when we are trying to achieve sparse graphs. For example, a tree of p vertices gives rise to a class of size p, whereas the equivalence class contains $p!$ DAGs in the case of the complete DAG (worst case) (Gillispie, 2006). For this reason, it is often a good idea to work with equivalence classes because they are a more efficient and robust representation: this representation is more able to deal with situations involving equivalence classes of high cardinality.

Three main problems arise when working in the DAG space rather than with equivalence classes. First, some operators defined to move between DAGs may operate between graphs in the same equivalence class. This is a waste of time unless it is checked by the algorithm. Therefore, if we work with a score equivalent criterion, these moves do not change the overall network fitness. Second, the move from the current equivalence class might not be the best if we ignore the equivalence class space and analyze only the neighborhood of just one of its DAGs. Given that all members of a class score the same value (assuming a score equivalent criterion), there is no reason other than randomness to prefer a specific DAG member. The third problem is related to how likely the final DAG is to belong to a specific equivalence class. If all DAGs in the same class are interpreted as different models, it is reasonable to expect the final output to be an equivalence class covering many rather than just a few DAGs. This makes it unpredictable.

A model is defined as *inclusion optimal* with regard to a density $f(\boldsymbol{X})$ if it is able to represent the density with a minimum number of arcs. The *inclusion boundary* (IB) concept (Kočka et al., 2001) is defined as a neighborhood of a model where the search strategy selects new models. Intuitively, the IB can be defined as the union set of equivalence classes that can be reached from each DAG inside the current class by single arc addition or deletion (see Figure 8.1). The *Greedy Equivalence Search* algorithm (GES) for Bayesian network learning (Meek, 1997) uses the IB. It has appealing theoretical properties for finding inclusion optimal models (Chickering, 2002b; Chickering and Meek, 2002). The KES algorithm –*k-Greedy Equivalence Search*– by Nielsen et al. (2003) generalizes GES and respects the IB too. Hence it retains the same theoretical properties. KES features a stochastic factor. This way multiple runs can be made to extract common patterns in the solutions, whereby the final model includes just the arcs that showed up in most of the runs.

In this chapter, we extend the KES algorithm to continuous (Gaussian) densities and learn sparser models based on regularization techniques. By definition, a GBN defines

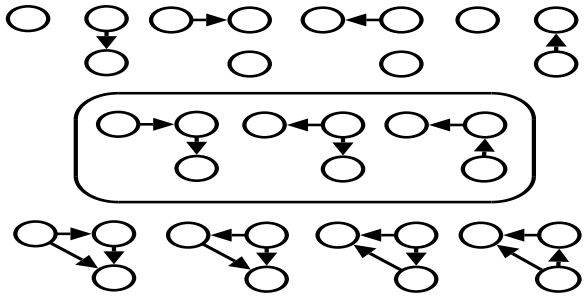


Figure 8.1: An equivalence class for 3 nodes. Around it, its inclusion boundary, dropping an arc (top) and adding an arc (bottom).

a regression for each variable X_i over its parents (see expression (8.2)). Supposing standardized data (so that $m_i = 0$ and $v_i = 1$), it is defined as

$$x_i = \sum_{x_j \in \boldsymbol{pa}(x_i)} \beta_{ji} x_j + w_i,$$

where w_i is the Gaussian noise term. Thus, it makes sense to apply a regularization technique to these regressions, linking the variable selection task to the neighborhood selection process in network structure learning. Here, we use the lasso (Tibshirani, 1996).

The key idea is to use lasso regression in the equivalence class space (prior to the learning stage) for each variable X_i on the remaining variables, discarding variables whose coefficients have been moved to zero as possible parents of X_i in the GBN. This produces simple models that properly fit available data. We keep the set of parents that yields the best MDL score. As an additional contribution, to reduce the computational burden, we also take advantage of the "approximate" convexity of the MDL score for each separate variable against the remainder. This makes it possible to stop the lasso algorithm before it ends.

The lasso has already been employed in the literature for some sorts of probabilistic graphical model learning, taking advantage of its ability for variable selection (neighborhood selection).

Li and Yang (2005) performed neighborhood selection by using the lasso to estimate a DAG from a given ordering, and then transformed the DAG into an undirected graph, which is the final aim of their algorithm. Under a Bayesian perspective, they used a Wishart prior distribution for the precision matrix. The DAG prior is derived from this precision matrix prior, and such DAG prior turns out to be equivalent to a Laplace prior. Since the objective is learning the undirected graph, they used an arbitrary ordering. Hence this method cannot be used to estimate a final directed graph.

Meinshausen and Bühlmann (2006) carried out neighborhood selection also in an undirected Gaussian graphical model setting. They used lasso regressions individually with every variable against the rest, in such a way that an edge is created when the regression coefficient is not zero. With an appropriate selection of the penalization parameter, the method is proved to be consistent for sparse high dimensional graphs under certain assumptions. However, the lasso estimate is based only on individual regressions and ignores the overall likelihood of the network. This may entail some problems. For example, the regression coefficient of a given variable X_i, when X_j is

the response, may (and probably will) be different from the regression coefficient of X_j, when X_i is the response; it means that it is possible that only one coefficient is shrunk to zero. In such a case, it is not clear whether the undirected edge must be present or not in the learnt graph.

Recently, Banerjee et al. (2008) presented two new efficient algorithms to estimate the (sparse) covariance matrix that exactly maximizes the L1-penalized maximum likelihood. Friedman et al. (2007b) developed an even faster algorithm for this task, called graphical lasso. Both papers state that the method of Meinshausen and Bühlmann (2006) is an approximation, as in general it does not reach the maximum likelihood solution.

The use of lasso directly to learn a graphical model entails an important drawback: it fails to recover the true sparsity pattern when variables are highly correlated, in particular when there exists high correlation between relevant and irrelevant variables (Zhao and Yu, 2006). This could be the case in some real world scenarios. Thus, as proposed in this chapter, it is reasonable to use additional heuristics instead of employing the lasso directly for network induction.

There has been less work focusing on (directed) GBNs, but we claim that the orientation is semantically useful in some problems like gene networks analysis. Undirected graphs also imply a high complexity (NP-hard) in the estimation of the parameters when the distribution is not Gaussian. Schmidt et al. (2007) worked alternatively on the DAG space and the variable ordering space using the lasso to restrict both search spaces. The drawbacks of their method are those commented above for greedy searching in the DAG space. In this chapter, instead of directly creating an arc when the corresponding regression coefficient is not zero as in (Meinshausen and Bühlmann, 2006), we employ the lasso as a previous neighborhood selector working in the equivalence class space. The regression is also carried out for every variable against the rest, as in (Schmidt et al., 2007). Therefore, the lasso is viewed as a variable filtering first step and not as a direct model selection method. Afterwards, we employ the greedy algorithm KES in the GBN training scenario.

8.3 KES ALGORITHM

The concept of equivalence for Bayesian networks (Chickering, 2002b) has been widely discussed in the literature: two DAGs are *Markov equivalent* (just equivalent from now on) if they represent the same set of conditional independences.

A DAG G' is said to *include* a graph G, $G \subset G'$, if for all the models $M(G,\theta)$ parameterized by θ whose structure is represented by G, there exists a second model $M'(G',\theta')$ parameterized by θ' with a structure G' that represents the same density function, i.e.

$$G \subset G' \text{ iff } \forall\theta\ \exists\theta' \quad | \quad f(x_{M(G,\theta)}) \equiv f(x_{M'(G',\theta')}).$$

Two DAGs G and G' are said to be *equivalent* if

$$G \subset G' \wedge G' \subset G.$$

For example, the DAG $A \rightarrow B \rightarrow C$ is equivalent to the DAG $A \leftarrow B \leftarrow C$, because all the joint density functions (or probability functions) that can be encoded by the first DAG can also be encoded by the second, and vice versa.

The equivalence relationship is reflexive, symmetric and transitive, and hence gives rise to the concept of *equivalence class*. The equivalence class definition is the same for Gaussian Bayesian networks. A *v-structure*, also called immorality, is induced when two disconnected vertices are parents of a third vertex. G and G' are equivalent if they have the same skeletons and the same v-structures (Verma and Pearl, 1990). A *covered arc* is an arc that is not part of any v-structure. Thus, it is possible to move across all the individuals inside an equivalence class just by covered arc reversing.

A formal definition of *inclusion boundary* can be presented on the basis of the DAG inclusion concept. Let G and G' be two DAGs, we denote $G \prec G'$ to mean that $G \subset G'$, and there is no DAG G'' such that $G \subset G'' \subset G'$. The IB of G is then defined as

$$IB(G) = \{G' | G' \prec G\} \quad \cup \quad \{G'' | G \prec G''\}.$$

The set of DAGs defined by the first term of the union is called the lower IB; the set of DAGs defined by the second term is the upper IB. A set of operators defined by a neighborhood is said to satisfy the IB condition if, for a DAG G, the induced neighborhood includes $IB(G)$ (Kočka et al., 2001). Two neighborhoods satisfying this condition are:

- ENR (Equivalence class-based No arcs Reversals), that considers all simple edge additions and deletions from all the DAGs belonging to the equivalence class.
- ENCR (Equivalence class-based Non-Covered arcs Reversals), that considers all simple edge additions, deletions and non-covered arc reversals from all the DAGs belonging to the equivalence class.

The ENR neighborhood exactly matches and the ENCR neighborhood includes the inclusion boundary, but they are computationally complex to calculate. As we will note below, these neighborhoods may be somehow approximated.

The so-called Meek Conjecture (Meek, 1997) essentially claims that, if a DAG G includes another DAG G', G is reachable from G' through a finite sequence of edge additions and covered arc reversals. An important conclusion of this premise is drawn: in the limit of large data sets, if the probability distribution (density function) has a perfect map in a DAG, a greedy search algorithm is suitable for finding the optimal solution after a finite set of edge additions and covered arc reversals. Chickering (2002b) presented a proof of the Meek Conjecture.

Basing on this, the GES algorithm (Meek, 1997) starts with the empty graph and greedily explores the equivalence class space, moving at each iteration to the state where score improvement is the greatest, and stopping the search when a locally optimal model is reached. It does so in two phases, considering different neighborhoods in each one. In the first phase the neighborhood is based on edge addition, whereas in the second phase the neighborhood contains the networks obtained by deleting a single edge. Chickering (2002b) presented a version of GES where the entire IB is examined at each step, removing the separation into two phases. In (Chickering, 2002a), six operators were introduced to enrich the search space: insert undirected arc, delete undirected arc, insert directed arc, delete directed arc, reverse directed arc and create v-structure.

Because of the high computational cost of working with neighborhoods (specifically, ENR and ENCR) that satisfy the IB condition, mainly to enumerate all the DAGs of the class, Castelo and Kočka (2003) defined an approximate approach. The strategy

is to somehow simulate the class by performing a sufficient number of covered arc reversals. At each individual or set of covered arc reversals, we get a DAG member of the class, and we explore some neighborhood of this member in a cheaper DAG space. It is shown that the number of covered arc reversals does not need to be large, since the average size of a class is bounded by a constant (Gillispie and Perlman, 2001). The number of covered arc reversals should depend on the size of the true equivalence class, if we have any idea of its cardinality (Muruzábal and Cotta, 2007).

The *Stochastic Equivalence Search* algorithm (SES) introduces a modification in the GES search strategy: it does not select the best member of the IB at each step, but randomly picks any of the models that improve the score.

In search of a trade-off, GES and SES are generalized in the KES algorithm (Nielsen et al., 2003), controlling the degree of randomness versus greediness by a parameter $0 \leq k \leq 1$. In a nutshell, KES is an iterative algorithm that extracts an uninformed k proportion (at least one model) of the IB at each step, selecting the best model in this set. GES corresponds to the $k = 1$ case; SES is the $k = 0$ case. Nielsen et al. (2003) presented a theoretical analysis of KES, supported on known results about GES. We will work here with the KES algorithm, performing the IB neighborhood approximation proposed by Nielsen et al. (2003) and already suggested by Castelo and Kočka (2003).

8.4 KES COMBINED WITH THE LASSO

We propose a lasso-based previous step to preselect a set of potential parents for each variable on which the KES algorithm will work greedily. Arcs corresponding to zero coefficients in the penalized regression are discarded. The Sparse Candidate algorithm (Friedman et al., 1999) also performs a preselection phase. This algorithm needs the user to set the maximum number of parents per node in advance. The Sparse Candidate algorithm restricts the search space so that there is only one set of possible parents for each variable in the subsequent maximization step (typically a greedy algorithm). These sets are constructed by including the variables that are most closely associated with the target variable; this is usually quantified with a pairwise measure like *mutual information*.

Alternatively, we employ the lasso with each variable on the rest, discarding as parents the variables whose regression coefficients become zero. Afterwards we launch the KES algorithm on the equivalence class search space. Thus, unlike the Sparse Candidate algorithm, we do not need to establish a maximum number of parents in advance.

The pseudocode of our proposal is shown in Algorithm 6, illustrating the two phases of the algorithm. Parents restriction appears in the *for* loop, calculating for each variable the set of coefficients that yields the best MDL score in the regularization path (represented by a matrix with p columns and a variable number of rows, one for each point in the regularization path where a coefficient vanishes or reappears into the model). The score+search KES algorithm, restricted by the previous step, is enclosed in the *repeat* loop. Notice that MDL is used in both phases.

Note that not all the variables have to share the same value of s when carrying out their regression against the rest of variables, as some nodes in the true structure may have stronger connectivity than others. A stronger regularizer (smaller s) would be required for these nodes, so that more regression coefficients $\hat{\beta}_i^s$ will be driven to zero. A common strategy for selecting s is to choose it by cross-validation from a grid of

values. However, it would have to be applied individually for each variable and hence could introduce a heavy computational load in high-dimensional problems.

Instead, Schmidt et al. (2007) took advantage of the piecewise constant nature of the number of non-zero coefficients against parameter s and suggested to take for each variable the best set of "parent candidates" from such regularization path according to some criterion, that has a finite and reduced number of solutions. That is the approach we use in this chapter. Hence we do not estimate an explicit value of s. We employ the (score equivalent) MDL criterion, which is defined as

$$MDL(X_i) = m \ \log(N)/2 + NLL(X_i|\boldsymbol{Pa}(X_i)), \tag{8.3}$$

where m is the number of parameters different from zero and $NLL(X_i|\boldsymbol{Pa}(X_i))$ is the negative log-likelihood of the network made up of this node and its parents.

In short, we evaluate all the points (all the sets of p coefficients) where a new one coefficient vanishes (or appears) in each variable's regularization path and choose the set of coefficients that minimizes the MDL score. The chosen set of penalized regression coefficients is not used again beyond the parents preselection stage, and the GBN parameters will be learnt later, irrespective of these coefficients.

In the second phase, we must choose between two strategies: the first is that a variable X_j is included as a possible parent of X_i if X_j is selected by the lasso when the response variable is X_i, or if X_i is selected by the lasso when the response variable is X_j; the second, if both conditions are fulfilled. We have tested the two strategies, which we will call OR-Lasso and AND-Lasso, respectively.

Even restricted to the above subset of potential parents, the KES algorithm does not lose the theoretical properties that we described above if we assume L1-regularization to be an ideal variable selector (specifically, if it does not miss true relations). As noted above, this applies under certain conditions. In this case, it will find a set of

Algorithm 6 Lasso embedded in KES

for $i = 1$ **to** p **do**
  $Path(i)$:= matrix with the L1-regularization path of X_i
  $Beta(i)$:= MDL-best row of p coefficients in $Path(i)$
  $PotentialParents(i)$:= set of variables s.t. $Beta(i)(j) \neq 0$,
    $1 \leq j \leq p$
end for
Initialize G := Empty or randomly generated model
$minimum := false$
repeat
  $K := \max(k \cdot size(IB(G, PotentialParents)), 1)$
    where $IB(G, PotentialParents)$ is the $IB(G)$ constrained
    by the lasso
  S := set of K models drawn from $IB(G, PotentialParents)$
  G' := the model from S with the best MDL score
  if $MDL(G') < MDL(G)$ **then**
    $G := G'$
  else
    $minimum := true$
  end if
until $minimum$ is $true$

Table 8.1: Mean and standard deviation (in seconds) of 10 runs measuring the run time for some network sizes of *Factor* structure. All data sets have 1000 instances.

No. variables	KES	KES + Lasso	Lasso
20	3.08($\pm$0.22)	1.75($\pm$0.06)	0.25($\pm$0.06)
50	84.68($\pm$13.28)	27.57($\pm$0.82)	1.77($\pm$0.05)
100	1145.3($\pm$28.11)	259.0($\pm$9.90)	16.3($\pm$0.84)

variables with all the parents, children and co-parents: the Markov blanket (Schmidt et al., 2007). Hence, ideally, the lasso only discards false parents.

We use a DAG to represent the current equivalence class in our implementation. Therefore, we need a method to approximate the IB from a DAG representation. We do not calculate the complete IB neighborhood at each step; rather we approximate its size from the total number of possible parents (taking the the lasso restriction into account). Then we derive a k-proportional number of models of this estimated size and keep the best one. Thus, we are not actually drawing a k-proportion from the set of models that are better than the current one, but an approximate k-proportion from the total (approximate because the size of the IB has been approximated), keeping the best model from this set.

The worst-case computational cost of the algorithm is equivalent to KES. This would be the unlikely situation where the best lasso preselection yields that all variables can be parents of all variables. However, the lasso is proved to hold a parsimonious property (Knight and Fu, 2000). Thus, the variable selector nature of the lasso usually restricts the search space quite a lot and makes the mean computational cost of the proposed approach definitely lower than KES algorithm alone. Some run times are shown in Table 8.1 to illustrate this issue.

Moreover, this chapter introduces a simple but useful heuristic on the LARS algorithm when used in the parents restriction phase. As noted before, we must search for the MDL-optimal $\hat{\boldsymbol{\beta}}^s$ for each and every variable in the regularization paths. The MDL score along the regularization path of each lasso with each variable as the response (starting with all $\hat{\beta}_i^s = 0$ and ending with all $\hat{\beta}_i^s \neq 0$) often follows a convex curve with only one minimum (see Figure 8.2).

These irregularities are the source of our heuristic, as more than one local minimum is sometimes present (Figures 8.2b-8.2d). For this reason the whole curve should be inspected to be sure of getting the global minimum. However, the global minimum is usually reached at an early phase (an exception is shown in Figure 8.2c). Even more importantly, when the MDL curve has grown for long enough, it normally never decays significantly again. This leads us to stop LARS if the MDL curve has grown for a number of steps, say equal to 20% of the full model variables, where we can be reasonably sure of having obtained the global minimum. In this way we are saving some computational cost.

In some cases, some local minima are reached later on (Figure 8.2c,8.2d, and especially in Figure 8.2d), but they are far from the global minimum and should be ignored. To make our stopping method robust to these minor slumps and preserve its efficiency, instead of checking punctual MDL values, we calculate the mean in a certain window of MDL values. This way we only keep searching if the decay is relevant, that is, we stop LARS when the MDL-mean of this window has been growing for enough iterations. If the number of values to be included in this mean is too small

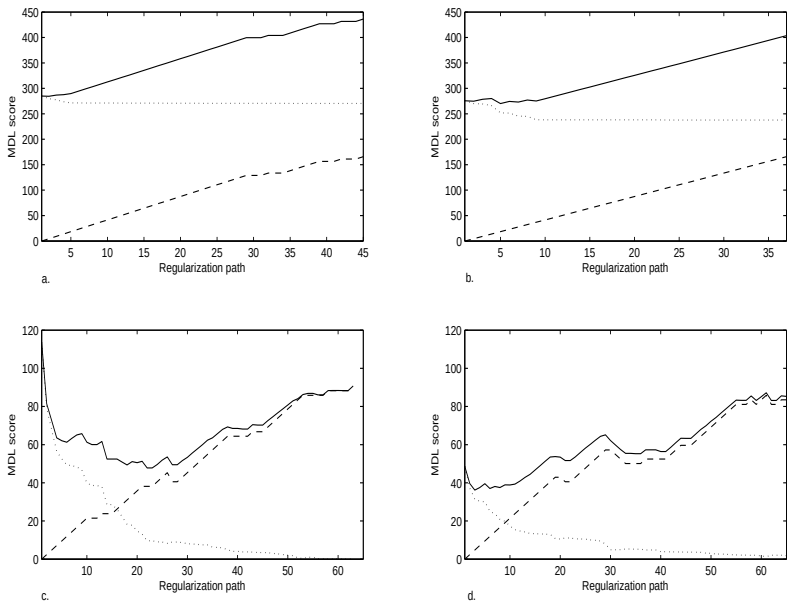


Figure 8.2: At the top, MDL curve along the regularization path for two variables of the *Alarm* network; at the bottom, two variables of isoprenoid biosyntesis for *Arabidopsis thaliana*. The solid line represents MDL, the dotted line is the negative log-likelihood, and the dashed line is the penalization term in MDL (first term in equation (8.3)). Therefore, the solid line is the sum of the other two lines.

(or is just one), we are being conservative and will probably advance along the regularization path further than necessary. On the other hand, if we take many values, we risk stopping early and missing the real global minimum. Empirically, we have found that taking the last five values leads to good results, although this point should be further researched.

In Table 8.2 we show the iteration number where the algorithm stops using different window sizes for each variable against the rest in the *Arabidopsis thaliana* isoprenoid biosynthesis data set. The optimal iteration, where the global minimum is reached, is depicted in the second column; if the stopping iteration is lower than the optimal one, we are trapped in a local minimum. Note that the window size does not have a big impact on most variables. For example, for the first variable, the algorithm always stops around iteration 18, that is, beyond the global minimum in iteration 10, regardless of the window size. More meaningful is the effect of the number of iterations that the MDL-mean of the window needs to keep increasing to stop. In Table 8.3 we show the ratio of this parameter against the number of variables. To simplify, we call such ratio "stopping ratio". Note that the difference between 0.05 and 0.3 is significant. Again, the iteration corresponding to the global minimum is depicted in the second column. Note that in both experiments we always advanced further than the optimal iteration and thus covered the global minimum. For all variables, the lowest size of the regularization path is over 60, and the mean size is 82. However, the global minimum is usually in the first third of the regularization path.

Table 8.2: LARS stopping iteration for different window sizes, for the *Arabidopsis thaliana* data set. The second column represents the iteration when the global minimum is reached.

Variable	Optimal iteration	Window size						
		1	2	3	4	5	6	7
AACT1	10	18	18	18	18	17	18	19
AACT2	3	14	14	14	14	13	14	14
CMK	2	9	10	11	12	12	13	14
DPPS1	3	14	14	14	14	12	14	15
DPPS2	2	9	10	11	12	12	13	14
DPPS3	1	9	10	11	12	12	13	14
DXPS1	22	33	33	33	33	30	33	34
DXPS2	2	9	10	11	12	12	13	14
DXPS3	6	15	15	15	15	14	15	14
DXR	4	14	14	14	14	13	14	14
FPPS1	5	14	14	14	14	13	14	15
FPPS2	3	9	10	11	12	12	13	14
GGPPS1	3	17	17	17	17	16	17	15
GGPPS2	2	9	10	11	12	12	13	14
GGPPS3	1	9	10	11	12	12	13	14
GGPPS4	1	9	10	11	12	12	13	14
GGPPS5	1	9	10	11	12	12	13	14
GGPPS6	2	9	10	11	12	12	13	14
GGPPS8	2	9	10	11	12	12	13	14
GGPPS9	2	9	10	11	12	12	13	14
GGPPS10	2	9	10	11	12	12	13	14
GGPPS11	4	14	14	14	14	13	14	14
GGPPS12	3	14	14	14	14	13	14	14
GPPS	7	17	17	17	17	16	17	17
HDR	3	9	10	11	12	13	13	14
HDS	4	9	10	11	12	13	13	14
HMGR1	4	9	10	11	12	13	13	14
HMGR2	6	15	15	15	15	14	15	16
HMGS	4	14	14	14	14	13	14	15
IPPI1	6	9	10	11	12	21	22	21
IPPI2	4	9	10	11	12	21	22	21
MCT	3	9	10	11	12	21	22	21
MECPS	3	9	10	11	12	21	22	21
MK	3	9	10	11	12	21	22	21
MPDC1	3	9	10	11	12	21	22	21
MPDC2	4	9	10	11	12	21	22	21
PPDS1	2	9	10	11	12	21	22	21
PPDS2	2	9	10	11	12	21	22	21
UPPS1	3	14	14	14	14	13	14	15

8.5.1 *Synthetic networks*

The *Alarm* network Beinlich et al. (1989) contains 37 nodes and 46 arcs. The *Insurance* network Binder et al. (1997) has 27 variables and 52 arcs. Both are commonly used

Table 8.3: LARS stopping iteration for different stopping ratios (see text), for the *Arabidopsis thaliana* data set. The second column represents the iteration when the global minimum is reached.

Variable	Optimal iteration	Stopping ratio				
		0.05	0.1	0.15	0.2	0.3
AACT1	10	14	16	18	20	24
AACT2	3	10	12	14	16	20
CMK	2	8	10	12	14	18
DPPS1	3	10	12	14	16	20
DPPS2	2	8	10	12	14	18
DPPS3	1	8	10	12	14	18
DXPS1	22	25	31	33	35	39
DXPS2	2	8	10	12	14	18
DXPS3	6	11	13	15	17	21
DXR	4	10	12	14	16	20
FPPS1	5	10	12	14	16	20
FPPS2	3	8	10	12	14	18
GGPPS1	3	10	15	17	19	23
GGPPS2	2	8	10	12	14	18
GGPPS3	1	8	10	12	14	18
GGPPS4	1	8	10	12	14	18
GGPPS5	1	8	10	12	14	18
GGPPS6	2	8	10	12	14	18
GGPPS8	2	8	10	12	14	18
GGPPS9	2	8	10	12	14	18
GGPPS10	2	8	10	12	14	18
GGPPS11	4	10	12	14	16	20
GGPPS12	3	10	12	14	16	20
GPPS	7	13	15	17	19	23
HDR	3	8	10	12	14	18
HDS	4	8	10	12	14	18
HMGR1	4	8	10	12	14	18
HMGR2	6	11	13	15	17	21
HMGS	4	10	12	14	16	20
IPPI1	6	8	10	12	24	28
IPPI2	4	8	10	12	24	28
MCT	3	8	10	12	24	28
MECPS	3	8	10	12	24	28
MK	3	8	10	12	24	28
MPDC1	3	8	10	12	24	28
MPDC2	4	8	10	12	24	28
PPDS1	2	8	10	12	24	28
PPDS2	2	8	10	12	24	28
UPPS1	3	10	12	14	16	20

to test Bayesian network learning algorithms. Finally, we have generated a synthetic network called *Factor*, with 100 variables and 382 arcs. This holds an arc from variable X_i to X_j if i is a divisor of j. Since we need a GBN, we will use the dependences of each network to simulate continuous Gaussian data sets of 100 samples for *Alarm*, 1000 for *Insurance* and 50 for *Factor* (testing the $p > N$ case). Parameter β_i in equation (8.2) is generated at random from a standard normal distribution. The rest of the parameters of the density functions are fixed ($m_i = 0$ and $v_i = 1$). We run the KES algorithm for 10 different values of $k : 0.1, 0.2, \ldots, 0.9, 1.0$, with and without a previous lasso step. Here we illustrate the results using the OR-Lasso strategy, as the AND-Lasso strategy turned out to be very restrictive in this case, producing networks with few arcs. For each k, we performed 10 runs and calculated the mean and the standard deviation. We show the results for the Alarm network in Figure 8.3. We also tried the approach proposed in Schmidt et al. (2007), that works in the DAG space, although we do not show the results in the figures for clarity. Instead, we present means and standard deviations in Table 8.4.

Note that our approach (working in the equivalence class space) on the whole outperforms the networks obtained working in the DAG space. This is specially notable in the *Alarm* network. For the *Factor* network, the number of correct arcs is equivalent in both spaces, whereas in the equivalence class space the number of false positives is slightly lower. On the other hand, for the *Insurance* network, where $n > p$, the results between the DAG space and the equivalence class space are quite similar. Note also that the random generation of the network parameters may affect the final output, and could be the cause of some differences between the learnt structures and the original network.

Regarding the comparisons, within the equivalence class space, between networks learnt with lasso preselection and networks learnt without it, the main conclusion we can draw is that the lasso information makes the output network sparser and tends to include fewer false arcs, and even, encouragingly, more correct arcs. Again, this is more obvious for the *Alarm* network. For the *Insurance* network the number of correct arcs is not very different, but the lasso preselection yields less false positives (and therefore, fewer total arcs). The same happens for the *Factor* network, where the number of false positives and the number of correct arcs are surprisingly alike for the KES algorithm.

Without lasso, we also observe that intermediate values of k (moderate randomness) result in slightly better networks. The *Factor* network, where k has not any influence at all, is an exception. When using the lasso, the quality of the networks is roughly equivalent in all cases of k. Moreover, the variance of the results is lower. Consequently, we could say that, with the preselection phase, the algorithm becomes quite robust to k, that is, there is not so much variation on parameter k, and this is no longer a cause

Table 8.4: Mean and standard deviation of some measures for networks obtained by the approach proposed in Schmidt et al. (2007).

Network	No. of arcs	False positives	Matches
Alarm	33.50($\pm$1.58)	13.70($\pm$1.70)	19.80($\pm$0.42)
Insurance	63.80($\pm$4.10)	22.60($\pm$3.03)	41.20($\pm$2.44)
Factor	168.10($\pm$2.02)	70.50($\pm$2.76)	97.60($\pm$1.17)

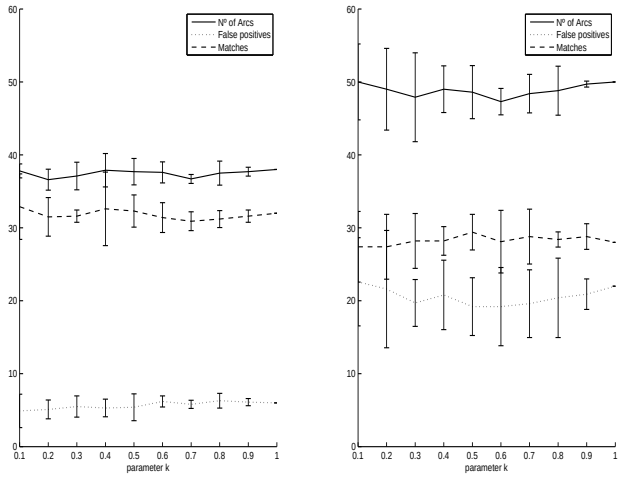


Figure 8.3: Behaviour of the KES algorithm for the *Alarm* network: Left, with lasso preselection. Right, without lasso parents preselection. Total arcs, true arcs and false arcs are shown.

for concern. Furthermore, as mentioned before, the computational cost is significantly lower when the lasso is employed.

8.5.2 *Pathways of the Arabidopsis thaliana plant*

Probabilistic graphical models, and specifically GBNs, may be used to model genetic networks. In the GBN case, each variable represents a continuous gene expression level; see (Friedman, 2004) for a review. Narrowing the field down to static GBNs (that do not model gene co-regulation against time), Wu et al. (2003) defined the GBN by previously determining the conditional independence relationships, Imoto et al. (2004) combined microarray data and known biological information to train the model.

Next we will test our method with a real-world data set taken from a biological environment: a list of 118 gene-expression patterns measured under different conditions for 40 genes that are found to be relevant in isoprenoid biosynthesis for *Arabidopsis thaliana* (Wille et al., 2004). *Arabidopsis thaliana* is the first plant whose complete genoma has been sequenced. Isoprenoids are the largest family of biological substances in nature and the oldest known biomolecules; over 30000 known compounds help in a great variety of biochemical processes. Understanding the nature of their synthesis is a task with many practical pharmaceutical and food applications. It is known that such synthesis follows two different gene routes in high-order plants, like *Arabidopsis thaliana*: the MVA (mevalonate) pathway and the MEP (methylerythritol 4-phosphate) pathway. Of the 40 measured genes, 16 come from the MVA pathway, 19 from the MEP pathway and the remaining 5 are encoding proteins located in the mitochondrion. Figure 8.4a shows the true pathways: the MEP pathway is the set of proteins on the left and the MVA pathway is the set on the right; genes related to mithocondrial proteins are UPPS1, DPPS2, GGPPS1, GGPPS5 and GGPPS9.

Our aim here is to check if our method is capable of neatly separating the two pathways. We train 10 networks for each k value ($k \in \{0.1, 0.2, \ldots, 0.9, 1.0\}$), and we

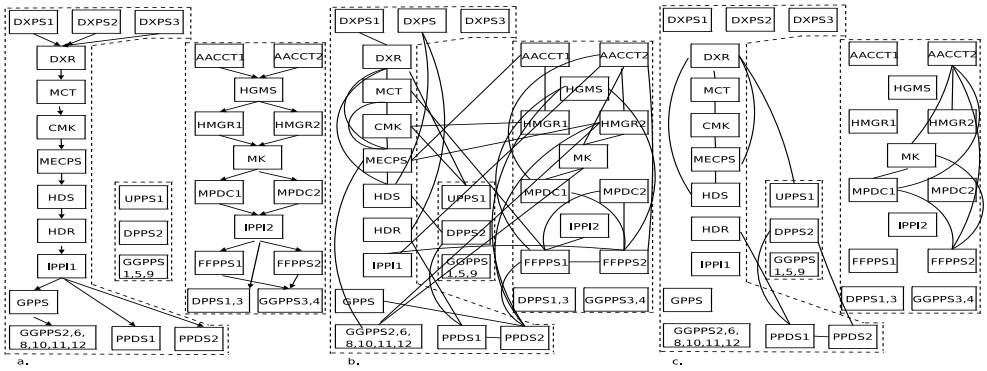


Figure 8.4: a. Real isoprenoid biosynthesis network; b. network shown in Wille et al. (2004); c. network shown in Li and Gui (2006).

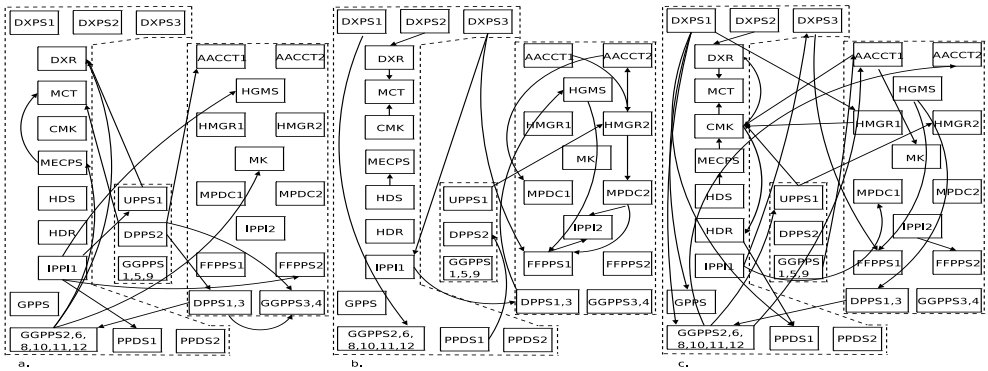


Figure 8.5: a. Network trained without lasso preselection ($fr = 0.8$, $k = 0.2$); b. network trained with AND-Lasso preselection ($fr = 0.7$, $k = 0.7$); c. network trained with OR-Lasso preselection ($fr = 0.8$, $k = 0.8$).

only keep those arcs that have been repeated with a frequency of at least fr. In order to experiment with different degrees of sparseness, we examine some values for fr: 0.6, 0.7, 0.8, 0.9 and 1.0. The networks obtained by Wille et al. (2004) and Li and Gui (2006) are shown in Figure 8.4b and Figure 8.4c, respectively.

It is also interesting to investigate the cross relationship between both pathways. One might expect a limited connectivity between the two pathways (Li and Gui, 2006), since they are both developed in different parts of the cell (see Figure 8.4a). However, Wille et al. (2004) cited some reports about these interactions, showing that cross-link connections do exist under certain circumstances.

Both the method suggested by Wille et al. (2004) and the threshold gradient method (TGD) proposed by Li and Gui (2006) identify a separation of both pathways. Note that undirected networks are used in both papers, whereas we train directed networks. The simpler method shown by Wille et al. (2004) trained a dense network. Although this network in essence owns many true arcs and distinguishes the two pathways, it connects many genes that are independent in the true network, with relatively dense cross-link connections. On the other hand, TGD, using bootstrap and keeping only the arcs that appear in at least a 50% of the obtained networks, reached a sparse network and no cross-link between pathways was drawn. However there are some

connections between the MEP pathway and mithocondrial proteins (located at the center of the network) that do not exist in the true pathways diagram. Furthermore, this network misses many true edges. For example, DXPS1, DXPS2 and DXPS3 appear to be completely independent, as do all GPPSS genes.

We have built different networks depending on the parent preselection strategy used (no preselection, AND-Lasso, and OR-Lasso) and on the fr and k values. Except for the networks trained without lasso preselection, parameter k has little influence. If we do not employ the lasso (Figure 8.5a), the KES algorithm outputs very dense networks where we can barely distinguish the two pathways. Only when fr is set to a high value sparser networks are reached, but it is not yet possible to appreciate the two pathways. For example, there are no arcs connecting genes inside the MVA pathway in Figure 8.5a. Note that this network has an equivalent number or even fewer arcs than the others; this is because of the higher variability of the networks trained without constraints: there are many arcs but they differ from one run to another.

AND-Lasso networks (Figure 8.5b) turn out to be the most interesting because they identify two interconnected modules in the network with relatively few arcs. The MVA pathway is specially well connected as compared with the real one in Figure 8.4a, and the DXPS, DXR, MCT, CMK, MECPS and HDS set is also connected in the MEP pathway except for the missing CMK - MECPS arc. Moreover, most AND-Lasso networks discover a relation between IPPI1 and some MVA pathway genes, which, as noted by Wille et al. (2004), could have an interesting biological interpretation.

When using an OR-Lasso (Figure 8.5c), there are many arcs that appear in at least 80% of the trained networks. Although we find some cross-link connections, we also find a higher arc density in each pathway. For instance, in the MEP pathway, DXPS, DXR, MCT, CMK, MECPS, HDS and HDR are closely connected. Again, we find an interaction of IPPI1 with elements of the MVA pathway.

Table 8.5 shows a quantitative comparison among the networks in Figures 8.4b, 8.4c, 8.5a, 8.5b and 8.5c. We score each pathway using a simplified version of the *Structural Hamming Distance* (SHD) Tsamardinos et al. (2006). This measure is defined by the number of operators (add or delete an undirected edge, and add, remove or reverse a directed arc) that are different in the PDAG to be scored and in the real PDAG. Since we want to evaluate also the undirected networks shown in Figures 8.4b and 8.4c, we will ignore the arc orientation and will not count reversal operators.

We also measure the rate between the number of arcs within the pathways and the number of arcs crossing between pathways and linking with mithocondrial genes. This value might be considered as a measure of the ability to separate the pathways, and will be denoted as *separability*.

Roughly speaking, networks of Figures 8.4c and 8.5b (corresponding to TGD and AND-Lasso methods) appear to be the best. Whereas TGD presents the best separability, AND-Lasso gives good SHD results and could produce the most refined pathways at some extent.

In summary, we have built networks that can compete with state-of-the-art methods, showing a very good computational performance. This has been shown with some synthetic data sets and a real biological network.

8.6 DISCUSSION

We have presented a procedure for learning the structure of Gaussian Bayesian networks based on a well-known regularization method for parent filtering and on the

Table 8.5: SHD and separability measures for the networks shown in Figures 8.4b, 8.4c and 8.5a, 8.5b, 8.5c. SHD assesses independently each pathway.

Network	SHD in pathway		Total SHD	Separability
	MEP	MVA		
Wille et al. (2004)	16	22	38	1.52
TGD Li and Gui (2006)	14	21	35	5.66
KES	15	16	31	0.50
AND-Lasso	11	20	31	3.75
OR-Lasso	15	17	32	1.60

KES algorithm, a greedy algorithm working on the equivalence class space. As discussed above, there are theoretical properties that support both methods. To the best of our knowledge, this is the first time that regularization has been employed in equivalence classes searching.

One advantage of our method is its computational efficiency. It is known that LARS solves the lasso with a reduced computational burden. Also, with parent restriction, the KES algorithm is significantly faster than ordinary KES, while offering better results. This means that several executions or bootstraping, trying to extract common patterns, can be run in situations where this would normally be infeasible. With the aim of improving efficiency, we have developed a simple but useful MDL-based method for stopping LARS in advance when it is used for parent preselection.

Note, however, that, since the optimality proof of KES is based on the IB neighborhood, our method is still heuristic. Although that it is unclear whether the asymptotic optimality is maintained in the considered restricted search, it is empirically proved to work well in practice. More recently, Gámez et al. (2011) explored optimality for restricted search algorithms. The authors propose a constrained search followed by an unconstrained search, achieving optimality under certain assumptions.

We have applied our approach with good results to three synthetic databases and a real biological data set, the isoprenoid biosynthesis pathways of *Arabidopsis thaliana*. Our results are successfully backed by previous domain knowledge, even though the available data offer just gene co-expression levels, not always directly related to regulatory patterns, that is, it does not always exactly reflect regulatory dependences. Previous work on building graphical models for this data set used undirected Gaussian networks. We think that our link orientation could supply useful information on the underlying biological processes in some situations. In fact, the original biological network that we are trying to model is also directed.

Part V

CONCLUSIONS

9

CONCLUSIONS

So far, we have introduced a number of contributions that range from innovations on machine learning methodology to applications to specific problems in neuroscience, showing that this methodology and its interesting properties can be of paramount importance in a wide variety of problems. In particular, regularization usually enriches the models with variable selection and a reasonable bias-variance trade-off.

From our contributions from kernel design in nonparametric regression, we can conclude the importance of considering sparsity and the noxious consequences of including irrelevant covariates in the weighting scheme calculation. These ideas can be applied also in the classification context when a kernel function is involved. For example, we could consider how the efficiency of a support vector machine can be improved by considering a nearly optimal (and sparse) bandwidth.

From the study of regularization for learning naïve Bayes classifiers, we can also extract some important ideas. Most importantly, regularization can help to overcome the problems derived from the presence of irrelevant and redundant predictors. We have discussed from a theoretical perspective that irrelevance and redundancy are neither absolute nor exclusive concepts. A predictor can be somewhat relevant for the prediction and partially redundant with another predictor. An automatic induction of the contributions of each predictors turns out to be valuable. In the future, some of these ideas can be applied to obtain Bayesian classifiers with a more complex structure than naïve Bayes.

Computational neuroscience is an emerging science that requires advances from statistical analysis and machine learning. Large efforts are being made for better understanding the brain, both from an operational and structural level. Future work will step up in this direction. In particular, we are interested in the application of some of the discussed ideas about regression, supervised classification and induction of graphical models' structures as a support for making hypotheses about the organization of the cortical column in the neocortex.

BIBLIOGRAPHY

Ahrens, M. B., Paninski, L., and Sahani, M. (2008). Inferring input nonlinearities in neural encoding models. *Network: Computation in Neural Systems*, 19:35–67.

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19:716–723.

Allen, D. M. (1974). The relationship between variable selection and data augmentation and a method for prediction. *Technometrics*, 16:125–127.

Avalos, M., Grandvalet, Y., and Ambroise, C. (2007). Parsimonious additive models. *Computational Statistics and Data Analysis*, 51:2851–2870.

Averbeck, B. B., Sohn, J. W., and Lee, D. (2006). Activity in prefrontal cortex during dynamic selection of action sequences. *Nature Neuroscience*, 9:276–282.

Bach, F. and Jordan, M. I. (2003). Learning graphical models with Mercer kernels. In *Advances in Neural Information Processing Systems 15*, pages 1009–1016. MIT Press, Cambridge, MA.

Bach, F. R. (2008a). Bolasso: model consistent lasso estimation through the bootstrap. In *15th International Conference on Machine Learning*, pages 33–40.

Bach, F. R. (2008b). Consistency of the group lasso and multiple kernel learning. *Journal of Machine Learning Research*, 9:1179–1225.

Balakrishnan, S. and Madigan, D. (2008). Algorithms for sparse linear classifiers in the massive data setting. *Journal of Machine Learning Research*, 9:313–337.

Banerjee, O., Ghaoui, L. E., and d'Aspremont, A. (2008). Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *Journal of Machine Learning Research*, 9:485–516.

Barrientos-Marin, J., Ferraty, F., and Vieu, P. (2010). Locally modelled regression and functional data. *Journal of Nonparametric Statistics*, 22:617–632.

Bartholomew, D. J. and Knott, M. (1999). *Latent Variable Models and Factor Analysis*. Hodder Arnold, second edition.

Beinlich, I., Suermondt, G., Chávez, R., and Cooper, G. F. (1989). The ALARM monitoring system. In *Proceedings of the Second European Conference on Artificial Intelligence and Medicine*, pages 247–256. Springer-Verlag.

Bergmann, G. and Hommel, G. (1988). Improvements of general multiple test procedures for redundant systems of hypotheses. In *Multiple Hypotheses Testing*, pages 100–115. Springer, Berlin.

Besag, J. (1975). Statistical analysis of non-lattice data. In *Fourth Conference on Artificial Intelligence*, pages 179–195.

Bezdudnaya, T., Cano, M., Bereshpolova, Y., Stoelzel, C. R., Alonso, J. M., and Swadlow, H. A. (2006). Thalamic burst mode and inattention in the awake LGND. *Neuron*, 49:421–432.

Bi, J., Bennett, K., Embrechts, M., Breneman, C., and Song, M. (2003). Dimensionality reduction via sparse support vector machines. *Journal of Machine Learning Research*, 3:1229–1243.

Bickel, P. J. and Li, B. (2006). Regularization in statistics. *TEST*, 15:271–344.

Bickel, P. J., Ritov, Y., and Tsybakov, A. B. (2009). Simultaneous analysis of lasso and Dantzig selector. *The Annals of Statistics*, 37:1705–1732.

Binder, J., Koller, D., Russell, S., and Kanazawa, K. (1997). Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29:213–244.

Blanco, R., Inza, I., Merino, M., Quiroga, J., and Larrañaga, P. (2005). Feature selection in Bayesian classifiers for the prognosis of survival of cirrhotic patients treated with TIPS. *Biomedical Informatics*, 38:376–388.

Boullé, M. (2007). Compression-based averaging of selective naïve Bayes classifiers. *Journal of Machine Learning Research*, 8:1659–1685.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Bradley, P. S. and Mangasarian, O. L. (1998). Feature selection via concave minimization and support vector machines. In *15th International Conference on Machine Learning*, pages 82–90.

Brenner, N., Bialek, W., and de Ruyter van Steveninck, R. (2000). Adaptive rescaling maximizes information transmission. *Neuron*, 26:695–702.

Brown, E. N., Barbieri, R., Ventura, V., Kass, R. E., and Frank, L. M. (2001). The time-rescaling theorem and its application to neural spike train data analysis. *Neural Computation*, 14:325–346.

Brown, E. N., Kass, R., and Mitra, P. P. (2004). Multiple neural spike train data analysis: State-of-the-art and future challenges. *Nature Neuroscience*, 7:456–461.

Bunea, F. (2008). Honest variable selection in linear and logistic regression models via l_1 and $l_1 + l_2$ penalization. *Electronic Journal of Statistics*, 2:1935–7524.

Burg, J. P. (1967). Maximum entropy spectral analysis. In *37th Meeting of the Society of Exploration Geophysics*.

Cai, T., Liu, W., and Luo, X. (2011). A constrained l_1 minimization approach to sparse precision matrix estimation. *Journal of the American Statistical Association*, 106:594–607.

Candès, E. J. and Plan, Y. (2009). Near-ideal model selection via l_1 minimization. *The Annals of Statistics*, 37:2145–2177.

Candès, E. J. and Tao, T. (2007). The Dantzig selector: Statistical estimation when p is much larger than n. *Journal of the Royal Statistical Society: Series B*, 35:2313–2351.

Candès, E. J. and Tao, T. (2010). The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56:2053–2080.

Castelo, R. and Kočka, T. (2003). On inclusion-driven learning of Bayesian networks. *Journal of Machine Learning Research*, 4:527–574.

Cawley, G. C. and Talbot, N. L. C. (2006). Gene selection in cancer classification using sparse logistic regression with Bayesian regularization. *Bioinformatics*, 22:2348–2355.

Cawley, G. C., Talbot, N. L. C., and Girolami, M. (2007). Sparse multinomial logistic regression via Bayesian l_1 regularisation. In *Neural Information Processing Systems*, pages 209–216.

Chatterjee, A. and Lahiri, S. N. (2011). Bootstrapping lasso estimators. *Journal of the American Statistical Association*, 106:608–625.

Chiappa, S. and Bengio, S. (2004). HMM and IOHMM modeling of EEG rhythms for asynchronous BCI systems. In *European Symposium on Artificial Neural Networks*, pages 199–204.

Chickering, D. M. (2002a). Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research*, 2:445–498.

Chickering, D. M. (2002b). Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554.

Chickering, D. M. and Meek, C. (2002). Finding optimal Bayesian networks. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, volume 3, pages 94–102.

Clemmensen, L., Hastie, T., , Witten, D., and Ersboll, B. (2011). Sparse discriminant analysis. *Technometrics*, In press.

Cleveland, W. S. (1979). Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74:829–836.

Cleveland, W. S. and Devlin, S. J. (1988). Locally weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83:596–610.

Cleveland, W. S. and Loader, C. (1996). Smoothing by local regression: Principles and methods. In *Statistical Theory and Computational Aspects of Smoothing*, pages 10–49. Heidelberg: Physica-Verlag.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20:273–297.

Craven, P. and Wahba, G. (1979). Smoothing noise data with spline functions: Estimating the correct degree of smoothing by the method of generalized cross-validation. *Numerische Mathematik*, 31:377–403.

Cui, X., Peng, H., Wen, S., and Zhu, L. (2011). Component selection in the additive regression model. *Submitted*.

Czanner, G., Eden, U. T., Wirth, S., Yanike, M., Suzuki, W. A., and Brown, E. N. (2008). Analysis of between-trial and within-trial neural spiking dynamics. *Journal of Neurophysiology*, 99:2672–2693.

d'Aspremont, A., Bach, F. R., and Ghaoui, L. E. (2007). Full regularization path for sparse principal component analysis. In *24th International Conference on Machine Learning*, pages 177–184.

Davies, S. (2002). *Fast Factored Density Estimation and Compression with Bayesian Networks*. PhD thesis, Carnegie Mellon University.

DeAngelis, G. C., Ohzawa, I., and Freeman, R. D. (1995). Receptive-field dynamics in the central visual pathways. *Trends of Neuroscience*, 18:451–458.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B*, 39:1–38.

Djebbari, A. and Labbe, A. (2009). Refining gene signatures: A Bayesian approach. *BMC Bioinformatics*, 10:410–420.

Domingos, P. and Pazzani, M. (1997). Beyond independence: Conditions for the optimality of the simple Bayesian classifier. *Machine Learning*, 29:103–130.

Donoho, D. L. (2006). Compressed sensing. *IEEE Transactions on Information Theory*, 52:1289–1306.

Dougherty, J., Kohavi, R., and Sahami, M. (1995). Searching for dependencies in Bayesian classifiers. In *International Conference on Machine Learning*, pages 194–202.

Drugan, M. M. and Wiering, M. A. (2010). Feature selection for Bayesian network classifiers using the MDL-FS score. *International Journal of Approximate Reasoning*, 51:695–717.

Duchi, J., Gould, S., and Koller, D. (2008). Projected subgradient methods for learning sparse Gaussians. In *24th Conference on Uncertainty on Artificial Intelligence*, pages 145–152.

Efron, B., Johnstone, I., Hastie, T., and Tibshirani, R. (2004). Least angle regression. *The Annals of Statistics*, 32:407–499.

Elhamifar, E. and Vidal, R. (2009). Sparse subspace clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2797.

Escola, S., Fontanini, A., Katz, D., and Paninski, L. (2011). Hidden Markov models for the stimulus-response relationships of multistate neural systems. *Neural Computation*, 23:1071–1132.

Fan, J. (1993). Local linear regression smoothers and their minimax efficiencies. *Annals of Statistics*, 21:196–216.

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96:1348–1360.

Farcomeni, A. (2009). An exact approach to sparse principal component analysis. *Computational Statistics*, 24:583–604.

Fayyad, U. M. and Irani, K. B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 1022–1027.

Ferraty, F., Hall, P., and Vieu, P. (2010). Most-predictive design points for functional data predictors. *Biometrika*, 97:807–824.

Ferraty, F. and Vieu, P. (2006). *Nonparametric Functional Data Analysis: Theory and Practice*. Springer.

Ferreira, J. T. A. S., Denison, D. G. T., and Hand, D. J. (2001). Data mining with products of trees. In *Advances in Intelligent Data Analysis*, volume 2189 of *Lecture Notes in Computer Science*, pages 167–176. Springer, Berlin.

Fotheringham, A. S., Brunsdon, C., and Charlton, M. (2002). *Geographically Weighted Regression: The Analysis of Spatially Varying Relationships*. Wiley.

Fraley, C. and Hesterberg, T. (2009). Least angle regression and LASSO for large datasets. *Statistical Analysis and Data Mining*, 1:251–259.

Frank, I. E. and Friedman, J. (1993). A statistical view of some chemometrics regression tools. *Technometrics*, 35:109–148.

Friedman, J. (1991). Multivariate adaptive regression splines. *Annals of Statistics*, 19:1–141.

Friedman, J., Hastie, T., Höfling, H., and Tibshirani, R. (2007a). Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1:302–332.

Friedman, J., Hastie, T., and Tibshirani, R. (2007b). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9:432–441.

Friedman, J., Hastie, T., and Tibshirani, R. (2010a). Applications of the lasso and grouped lasso to the estimation of sparse graphical models. Technical report, Stanford University.

Friedman, J., Hastie, T., and Tibshirani, R. (2010b). A note on the group lasso and a sparse group lasso. Technical report, Stanford University.

Friedman, J., Hastie, T., and Tibshirani, R. (2010c). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33:1–22.

Friedman, N. (2004). Inferring cellular networks using probabilistic graphical models. *Science*, 303:799–805.

Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, 29:131–163.

Friedman, N., Nachman, I., and Pe'er, D. (1999). Learning Bayesian network structure from massive datasets: The sparse candidate algorithm. In *Proceedings of the International Conference on Artificial Intelligence*, pages 206–215. Morgan Kaufmann.

Gámez, J. A., Mateo, J. L., and Puerta, J. A. (2011). Learning Bayesian networks by hill climbing: Efficient methods based on progressive restriction of the neighborhood. *Data Mining and Knowledge Discovery*, 22:106–148.

Gao, J. (2008). Robust l_1 principal component analysis and its Bayesian variational inference. *Neural Computation*, 20:555–572.

García, S. and Herrera, F. (2008). An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694.

Geiger, D. and Heckerman, D. (1994). Learning Gaussian networks. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, pages 235–243. Morgan Kaufmann.

Genkin, A., Lewis, D. D., and Madigan, D. (2007). Large-scale Bayesian logistic regression for text categorization. *Technometrics*, 49:291–304.

Gerstein, G. L. and Perkel, D. H. (1969). Simultaneously recorded trains of action potentials: Analysis and functional interpretation. *Science*, 164:828–830.

Gillispie, S. B. (2006). Formulas for counting acyclic digraph Markov equivalence classes. *Statistical Planning and Inference*, 136:1410–1432.

Gillispie, S. B. and Perlman, M. D. (2001). Enumerating Markov equivalence classes of acyclic digraph models. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, pages 171–177. Morgan Kaufmann.

Greenshtein, E. and Ritov, Y. (2004). Persistence in high dimensional linear predictor-selection and the virtue of over-parametrization. *Bernoulli*, 10:971–988.

Györfi, L., Kohler, M., Krzyzak, A., and Walk, H. (2002). *A Distribution-Free Theory of Nonparametric Regression*. Springer.

Haider, B., Duque, A., Hasenstaub, A. R., Yu, Y., and McCormick, D. A. (2007). Enhancement of visual responsiveness by spontaneous local network activity in vivo. *Journal of Neurophysiology*, 97:4186–4202.

Hall, M. (2000). Induction of selective Bayesian classifiers. In *Proceedings of the 17th International Conference on Machine Learning*, pages 359–366.

Hall, P., Li, Q., and Racine, J. S. (2007). Nonparametric estimation of regression functions in the presence of irrelevant regressors. *The Review of Economics and Statistics*, 89:784–789.

Hand, D. J. and Yu, K. (2001). Idiot's Bayes - not so stupid after all? *International Statistical Review*, 69:385–398.

Hans, C. (2010). Model uncertainty and variable selection in Bayesian lasso regression. *Statistics and Computing*, 20:221–229.

Hardoon, D. R. and Shawe-Taylor, J. (2011). Sparse canonical correlation analysis. *Machine Learning*, 83:331–353.

Hastie, T., Buja, A., and Tibshirani, R. (1995). Penalized discriminant analysis. *The Annals of Statistics*, 23:73–102.

Hastie, T. and Loader, C. (1993). Local regression: Automatic kernel carpentry. *Statistical Science*, 8:120–143.

Hastie, T., Rosset, S., Tibshirani, R., and Zhu, J. (2004). The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5:1391–1415.

Hastie, T., Taylor, J., Tibshirani, R., and Walther, G. (2007). Forward stagewise regression and the monotone lasso. *Electronic Journal of Statistics*, 1:1935–7524.

Hastie, T. and Tibshirani, R. (1990). *Generalized Additive Models*. Chapman and Hall.

Hastie, T. and Tibshirani, R. (1999). *Generalized Additive Models*. Chapman and Hall.

Hastie, T., Tibshirani, R., and Buja, A. (1994). Flexible discriminant analysis by optimal scoring. *Journal of the American Statistical Association*, 89:1255–1270.

Hastie, T., Tibshirani, R., and Friedman, J. (2008). *The Elements of Statistical Learning: Data Mining, Inference, and Predictions*. Springer, second edition.

Haxby, J. V., Gobbini, M., Furey, M. L., Ishal, A., Schouten, J. L., and Pietrini, P. (2002). Distributed and overlapping representation of faces and objects in ventral temporal cortex. *Science*, 293:2425–2430.

Hoerl, A. and Kennard, R. (1970). Ridge regression: Biased estimates for nonorthogonal problems. *Technometrics*, 12:55–67.

Höfling, H. and Tibshirani, R. (2009). Estimation of sparse binary pairwise Markov networks using pseudo-likelihoods. *Journal of Machine Learning Research*, 10:883–906.

Hsu, N. J., Hung, H. L., and Chang, Y. M. (2008). Subset selection for vector autoregressive processes using Lasso. *Computational Statistics and Data Analysis*, 52:3645–3657.

Huan, N. J. and Palaniappan, R. (2004). Neural network classification of autoregressive features from electroencephalogram signals for brain-computer interface design. *Journal of Neural Engineering*, 1:142–150.

Huang, J., Horowitz, J. L., and Wei, F. (2010). Variable selection in nonparametric additive models. *The Annals of Statistics*, 38:2282–2313.

Huang, J., Liu, N., Pourahmadi, M., and Liu, L. (2006). Covariance matrix selection and estimation via penalised normal likelihood. *Biometrika*, 93:85–98.

Huang, J., Ma, S., and Zhang, C. H. (2008). Adaptive lasso for sparse high-dimensional regression models. *Statistica Sinica*, 18:1603–1618.

Hyvärinen, A., Karhunen, J., and Oja, E. (2001). *Independent Component Analysis*. Wiley.

Hyvärinen, A. and Karthikesh, R. (2002). Imposing sparsity on the mixing matrix in independent component analysis. *Neurocomputing*, 49:151–162.

Imoto, S., Higuchi, T., Goto, T., Tashiro, K., Kuhara, S., and Miyano, S. (2004). Combining microarrays and biological knowledge for estimating gene networks via Bayesian networks. *Journal of Bioinformatics and Computational Biology*, 2:77–98.

Jacob, L., Obozinski, G., and Vert, J. P. (2009). Group lasso with overlap. In *26th International Conference on Machine Learning*, pages 433–440.

James, G. M., Radchenko, P., and Lv, J. (2009). DASSO: Connections between the Dantzig selector and lasso. *Journal of the Royal Statistical Society: Series B*, 27:127–142.

Jolliffe, I. T. (2002). *Principal Component Analysis*. Springer, second edition.

Jolliffe, T., Trendafilov, N. T., and Uddin, M. (2003). A modified principal component technique based on the LASSO. *Journal of Computational and Graphical Statistics*, 12:531–547.

Kass, R. E., Ventura, V., and Brown, E. N. (2005). Statistical issues in the analysis of neuronal data. *Journal of Neurophysiology*, 94:8–25.

Keirn, Z. A. and Aunon, J. I. (1990). A new mode of communication between man and his surroundings. *IEEE Transactions on Biomedical engineering*, 37:1209–1214.

Khan, J. A., van Aelst, S., and Zamar, R. H. (2007). Robust linear model selection based on least angle regression. *Journal of the American Statistical Association*, 102:1289–1299.

Knight, K. and Fu, W. (2000). Asymptotics for lasso-type estimators. *The Annals of Statistics*, 28:1356–1378.

Kohavi, R. and John, G. H. (1996). Wrappers for feature subset selection. *Artificial Intelligence*, 29:273–324.

Koller, D. and Friedman, N. (2009). *Structured Probabilistic Models: Principles and Techniques*. MIT Press.

Kočka, T., Bouckaert, R. R., and Studený, M. (2001). On characterizing inclusion of Bayesian networks. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, pages 261–268. Morgan Kaufmann.

Krishnapuram, B., Carin, L., and Figueiredo, M. (2005). Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:957–968.

Lafferty, J. and Wasserman, L. (2008). Rodeo: Sparse, greedy nonparametric regression. *The Annals of Statistics*, 36:28–63.

Langley, P. and Sage, S. (1994). Induction of selective Bayesian classifiers. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, pages 399–406.

Lauritzen, S. (1996). *Graphical Models*. Oxford University Press.

Lee, H., Battle, A., Raina, R., and Ng, A. Y. (2007). Efficient sparse coding algorithms. In *Neural Information Processing Systems*, pages 801–808.

Lee, S. I., Ganapathi, V., and Koller, D. (2006a). Efficient structure learning of Markov networks using l_1-regularization. In *Neural Information Processing Systems*, pages 817–824.

Lee, Y., Kim, Y., Lee, S., and Koo, J. Y. (2006b). Structured multicategory support vector machines with analysis of variance decomposition. *Biometrika*, 93:555–571.

Lee, Y., Lin, Y., and Wahba, G. (2004). Multicategory support vector machines, theory and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association*, 99:67–81.

Leng, C. and Wang, H. (2009). On general adaptive sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 18:201–215.

Levina, E., Rothman, A., and Zhu, J. (2008). Sparse estimation of large covariance matrices via a nested lasso penalty. *Annals of Applied Statistics*, 2:245–263.

Li, C. and Li, H. (2008). Network-constrained regularization and variable selection for analysis of genomic data. *Bioinformatics*, 24:1175–1182.

Li, F. and Yang, Y. (2005). Using modified lasso regression to learn large undirected graphs in a probabilistic framework. In *20th Conference on Artificial Intelligence*, pages 801–806.

Li, H. and Gui, J. (2006). Gradient directed regularization for sparse Gaussian concentration graphs, with applications to inference of genetic networks. *Biostatistics*, 7:302–317.

Li, Y. and Zhu, J. (2008). l_1-norm quantile regression. *Journal of Computational and Graphical Statistics*, 17:163–185.

Lin, Y. and Zhang, H. H. (2006). Component selection and smoothing in multivariate nonparametric regression. *The Annals of Statistics*, 34:2272–2297.

Liu, H., Lafferty, J., and Wasserman, L. (2009). The nonparanormal: Semiparametric estimation of high dimensional undirected graphs. *Journal of Machine Learning Research*, 10:2295–2328.

Liu, Y. and Wu, Y. (2007). Variable selection via a combination of the l_0 and l_1 penalties. *Journal of Computation and Graphical Statistics*, 16:782–798.

Loader, C. (1999). *Local Regression and Likelihood*. Springer.

Lorbert, A., Eis, D., Kostina, V., Blei, D. M., and Ramadge, P. J. (2010). Exploiting covariate similarity in sparse regression via the pairwise elastic net. In *14th International Conference on Artificial Intelligence and Statistics*, pages 477–484.

Lotte, F., Congedo, M., Lécuyer, A., Lamarche, F., and Arnaldi, B. (2007). A review of classification algorithms for EEG-based brain-computer interfaces. *Journal of Neural Engineering*, 4:1–13.

Machens, C. K., Wehr, M. S., and Zador, A. M. (2004). Linearity of cortical receptive fields measured with natural sounds. *Journal of Neuroscience*, 24:1089–1100.

Mallows, C. L. (1973). Some comments on C_p. *Technometrics*, 15:661–675.

Margaritis, D. (2005). Distribution-free learning of Bayesian network structure in continuous domains. In *Proceedings of The Twentieth National Conference on Artificial Intelligence*, pages 825–830.

Meek, C. (1997). *Graphical Models: Selecting Causal and Statistical Models*. PhD thesis, Carnegie Mellon University.

Meier, L. and Bühlmann, P. (2007). Smoothing l_1-penalized estimators for high-dimensional time-course data. *Electronic Journal of Statistics*, 1:597–615.

Meier, L., van de Geer, S., and Bühlmann, P. (2008). The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B*, 70:53–71.

Meinshausen, N. (2007). Lasso with relaxation. *Computational Statistics and Data Analysis*, 52:374–393.

Meinshausen, N. and Bühlmann, P. (2006). High dimensional graphs and variable selection with the lasso. *The Annals of Statistics*, 34:1436–1462.

Meinshausen, N. and Yu, B. (2009). Lasso-type recovery of sparse representations for high-dimensional data. *The Annals of Statistics*, 37:246–270.

Meng, D., Zhao, Q., and Xu, Z. (2011). Improve robustness of sparse pca by l_1-norm maximization. *Pattern Recognition*, 45:487–497.

Michalski, R. S. (1980). Learning by being told and learning from examples: An experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis. *Policy Analysis and Information Systems*, 4:125–161.

Minsky, M. (1961). Steps toward artificial intelligence. In *Computers and Thought*, pages 406–450. McGraw-Hill.

Muruzábal, J. and Cotta, C. (2007). A study on the evolution of Bayesian network graph structure. In *Advances in Probabilistic Graphical Models*, volume 214, pages 193–213. Springer-Verlag.

Ng, A. Y., Jordan, M. I., and Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. In *Neural Information Processing Systems*, pages 849–856.

Nielsen, J., Kočka, T., and Peña, J. (2003). On local optima in learning Bayesian networks. In *19th Conference on Uncertainty in Artificial Intelligence*, pages 435–442.

Osborne, M., Presnell, B., and Turlach, B. (2000). On the LASSO and its dual. *Journal of Computational and Graphical Statistics*, 9:319–337.

Paninski, L. (2004). Maximum likelihood estimation of cascade point-process neural encoding models. *Network: Computation in Neural Systems*, 15:243–262.

Paninski, L., Pillow, J. W., and Simoncelli, E. P. (2004). Maximum likelihood estimation of a stochastic integrate-and-fire neural encoding model. *Neural Computation*, 16:2533–2561.

Park, M. Y. and Hastie, T. (2007). l_1-regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society: Series B*, 69:659–677.

Park, M. Y. and Hastie, T. (2008). Penalized logistic regression for detecting gene interactions. *Biostatistics*, 9:30–50.

Park, T. and Casella, G. (2008). The Bayesian Lasso. *Journal of the American Statistical Association*, 103:681–686.

Pazzani, M. J. (1996). Searching for dependencies in Bayesian classifiers. In *Learning from Data: Artificial Intelligence and Statistics V*, pages 239–248.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligence Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Peng, J., Wang, P., Zhou, N., and Zhu, J. (2009). Partial correlation estimation by joint sparse regression models. *Journal of the American Statistical Association*, 104:735–746.

Penny, W. D., Roberts, S. J., Curran, E. A., and Stokes, M. J. (2000). EEG-based communication: A pattern recognition approach. *IEEE Transactions on Rehabilitation Engineering*, 8:214–215.

Pillow, J. W. and Simoncelli, E. P. (2006). Dimensionality reduction in neural models: An information-theoretic generalization of spike-triggered average and covariance analysis. *Journal of Vision*, 6:414–428.

Plourde, E., Delgutte, B., and Brown, E. N. (2011). A point process model for auditory neurons considering both their intrinsic dynamics and the spectrotemporal properties of an extrinsic signal. *IEEE Transactions on Biomedical Engineering*, 58:1507–1510.

Porzelius, C., Schumacher, M., and Binder, H. (2010). Sparse regression techniques in low-dimensional survival data settings. *Statistics and Computing*, 20:151–163.

Pötscher, B. M. and Schneider, U. (2009). On the distribution of the adaptive LASSO estimator. *Journal of Statistical Planning and Inference*, 139:2775–2790.

Radchenko, P. and James, G. M. (2008). Variable inclusion and shrinkage algorithms. *Journal of the American Statistical Association*, 103:1304–1315.

Radchenko, P. and James, G. M. (2010). Variable selection using adaptive nonlinear interaction structures in high dimensions. *Journal of the American Statistical Association*, 105:1541–1553.

Ramsay, J. and Silverman, B. W. (2005). *Functional Data Analysis*. Springer.

Ravikumar, P., Lafferty, J., Liu, H., and Wasserman, L. (2009). Sparse additive models. *Journal of the Royal Statistical Society: Series B*, 71:1009–1030.

Ravikumar, P., Wainwright, M. J., and Lafferty, J. D. (2010). High-dimensional graphical model selection using l_1-regularized logistic regression. *The Annals of Statistics*, 38:1287–1319.

Rosset, S. and Zhu, J. (2007). Piecewise linear regularized solution paths. *The Annals of Statistics*, 35:1012–1030.

Rosset, S., Zhu, J., and Hastie, T. (2004). Boosting as a regularized path to a maximum margin classifier. *Journal of Machine Learning Research*, 5:941–973.

Roth, V. (2004). The generalized LASSO. *IEEE Transactions in Neural Networks*, 15:16–28.

Rothman, A. J., Bickel, P. J., Levina, E., and Zhu, J. (2008). Sparse permutation invariant covariance estimation. *Electronic Journal of Statistics*, 2:1935–7524.

Ruppert, D. and Wand, M. (1994). Multivariate locally weighted least squares regression. *Annals of Statistics*, 22:1346–1370.

Sahani, M. and Linden, J. F. (2003). How linear are auditory cortical responses? In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems*, 15, pages 125–132.

Sain, S. R., Baggerly, K. A., and Scott, D. W. (1994). Cross-validation of multivariate densities. *Journal of the American Statistical Association*, 89:807–817.

Schaal, S. and Atkeson, C. G. (1998). Constructive incremental learning from only local information. *Neural Computation*, 10:2047–2084.

Schmidt, M., Murphy, K., Fung, G., and Rosales, R. (2008). Structure learning in random fields for heart motion abnormality detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

Schmidt, M. W., Niculescu-Mizil, A., and Murphy, K. P. (2007). Learning graphical model structure using l_1-regularization paths. In *22nd Conference on Artificial Intelligence*, pages 1278–1283.

Schölkopf, B., Smola, B., and Müller, K. R. (1999). Kernel principal component analysis. In *Advances in Kernel Methods*, pages 327–352. MIT Press.

Schumaker, L. L. (2007). *Spline Functions: Basic Theory*. Cambridge Mathematical Library, third edition.

Shevade, S. and Keerthi, S. (2003). A simple and efficient algorithm for gene selection using sparse logistic regression. *Bioinformatics*, 19:2246–2253.

Shi, J., Yin, W., Osher, S., and Sajda, P. (2010). A fast hybrid algorithm for large-scale l_1-regularized logistic regression. *Journal of Machine Learning Research*, 11:713–741.

Shi, W., Wahba, G., Wright, S., Lee, K., and an B. Klein, R. K. (2008). LASSO-patternsearch algorithm with application to ophthalmology and genomic data. *Statistics and its Interface*, 1:137–153.

Similä, T. and Tikka, J. (2006). Common subset selection of inputs in multiresponse regression. In *International Joint Conference on Neural Networks*, pages 1908–1915.

Similä, T. and Tikka, J. (2007). Input selection and shrinkage in multiresponse linear regression. *Computational Statistics and Data Analysis*, 52:406–422.

Smith, P. W. and Whittaker, J. (1998). Edge exclusion tests for graphical Gaussian models. In Jordan, M., editor, *Learning in Graphical Models*, pages 555–574. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Sprechmann, P., Ramírez, I., Sapiro, G., and Eldar, Y. (2010). Collaborative hierarchical sparse modeling. In *44th Annual Conference on Information Sciences and Systems*, pages 1–6.

Strawderman, W. E. (1978). Minimax adaptive generalized ridge regression estimators. *Journal of the Royal Statistical Society. Series B*, 73:623–627.

Tai, F. and Pan, W. (2007). Incorporating prior knowledge of gene functional groups into regularized discriminant analysis of microarray data. *Bioinformatics*, 23:3170–3177.

Tian, G. L., Tang, M. L., Fang, H. B., and Tan, M. (2008). Efficient methods for estimating constrained parameters with applications to lasso logistic regression. *Computational Statistics and Data Analysis*, 52:3528–3542.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B*, 58:267–288.

Tibshirani, R., Hastie, T., Narasimhan, B., and Chu, G. (2003). Class prediction by nearest shrunken centroids with applications to DNA microarrays. *Statistical Science*, 18:104–117.

Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., and Knight, K. (2005). Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B*, 67:91–108.

Tikhonov, A. N. (1943). On the stability of inverse problems (in russian). *Doklady Akademii Nauk SSSR*, 39:176–179.

Tipping, M. and Bishop, C. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B*, 6:611–622.

Trendafilov, N. T. and Jolliffe, I. T. (2007). DALASS: Variable selection in discriminant analysis via the LASSO. *Computational Statistics and Data Analysis*, 51:3718–3736.

Truccolo, W., Eden, U. T., Fellows, M. R., Donoghue, J. P., and Brown, E. N. (2005). A point process framework for relating neural spiking activity to spiking history, neural ensemble and extrinsic covariate effects. *Journal of Neurophysiology*, 93:1074–1089.

Tsamardinos, I., Brown, L. E., and Aliferis, C. F. (2006). The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65:31–78.

Tseng, P. (2001). Convergence of block coordinate descent method for nondifferentiable minimation. *Journal of Optimization Theory and Applications*, 109:475–494.

van Gerven, M. and Heskes, T. (2008). L_1/L_p regularization of differences. Technical Report ICIS-R08009, Radboud University Nijmegen.

van Gerven, M. A. J., Cseke, B., de Lange, F. P., and Heskes, T. (2010). Efficient Bayesian multivariate fMRI analysis using a sparsifying spatio-temporal prior. *Neuroimage*, 50:150–161.

Verma, T. and Pearl, J. (1990). Equivalence and synthesis of causal models. In *Proceedings of the 6th Conference on Uncertainty in Artificial Intelligence*, pages 1287–1330. Morgan Kaufmann.

Vidaurre, D., Bielza, C., and Larrañaga, P. (2010). Learning an l_1-regularized Gaussian Bayesian network in the equivalence class space. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 40(5):1231–1242.

Vidaurre, D., Bielza, C., and Larrañaga, P. (2011a). Classification of neural signals from sparse autoregressive features. *Neurocomputing*, Submitted.

Vidaurre, D., Bielza, C., and Larrañaga, P. (2011b). Forward stagewise naïve Bayes. *Progress in Artificial Intelligence*, 1:57–69.

Vidaurre, D., Bielza, C., and Larrañaga, P. (2011c). An l_1-regularized naïve Bayes-inspired classifier for discarding redundant predictors. *International Journal of Pattern Recognition and Artificial Intelligence*, Submitted.

Vidaurre, D., Bielza, C., and Larrañaga, P. (2011d). Lazy lasso for local regression. *Computational Statistics*, In press.

Vidaurre, D., Bielza, C., and Larrañaga, P. (2011e). On nonlinearity in neural encoding models applied to the primary visual cortex. *Network: Computation in Neural Systems*, 22:97–125.

Vidaurre, D., Bielza, C., and Larrañaga, P. (2011f). A survey on l_1-regularization for statistics and machine learning. *International Statistical Review*, Submitted.

Vidaurre, D., Bielza, C., and Larrañaga, P. (2012). Sparse Bayesian regularized local regression. *Computational Statistics and Data Analysis*, Submitted.

Vidaurre, D., Rodríguez, E. E., Rudomin, P., Bielza, C., and Larrañaga, P. (2011g). A new feature extraction method for signal classification applied to cat spinal cord signals. *Journal of Neural Engineering*, Submitted.

Wahba, G. (1990). *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics.

Wand, M. P. and Jones, M. C. (1994). Multivariate plug-in bandwidth selection. *Computational Statistics*, 9:97–117.

Wang, H. and Leng, C. (2007). Unified LASSO estimation via least squares approximation. *Journal of the American Statistical Association*, 102:1039–1048.

Wang, H. and Leng, C. (2008). A note on adaptive group lasso. *Computational Statistics and Data Analysis*, 52:5277–5286.

Wang, H., Li, G., and Jiang, G. (2007a). Robust regression shrinkage and consistent variable selection through the LAD-Lasso. *Journal of Business Economic Statistics*, 25:347–355.

Wang, H. and Xia, Y. (2009). Shrinkage estimation of the varying coefficient model. *Journal of the American Statistical Association*, 104:747–757.

Wang, J. and Chang, C. I. (2006). Independent component analysis-based dimensionality reduction with applications in hyperspectral image analysis. *IEEE Transactions on Geoscience and Remote Sensing*, 44:1586–1600.

Wang, L., Shen, X., and Zheng, Y. (2007b). On l_1-norm multiclass support vector machines. *Journal of the American Statistical Association*, 102:583–594.

Wang, L., Zhu, J., and Zou, H. (2006). The doubly regularized support vector machine. *Statistica Sinica*, 16:589–615.

Wang, S. and Zhu, J. (2008). Variable selection for model-based high-dimensional clustering and its application to microarray data. *Biometrics*, 64:440–448.

Wang, T., Deng, J., and He, B. (2004). Classifying EEG-based motor imagery tasks by means of time-frequency synthesized spatial patterns. *Clinical Neurophysiology*, 115:2744–2753.

Wheeler, D. C. (2009). Simultaneous coefficient penalization and model selection in geographically weighted regression: The geographically weighted lasso. *Environment and Planning A*, 41:722–742.

Whittaker, J. (1990). *Graphical Models in Applied Multivariate Statistics*. Wiley.

Wille, A., Zimmermann, P., Vranová, E., Fürholz, A., Laule, O., Bleuler, S., Hennig, L., Prelić, A., von Rohr, P., Thiele, L., Zitzler, E., Gruissem, W., and Bühlmann, P. (2004). Sparse graphical Gaussian modeling of the isoprenoid gene network in Arabidopsis thaliana. *Genome Biology*, 5, II: R92.1 - R92.13.

Witten, D. M. and Tibshirani, R. (2010). A framework for feature selection in clustering. *Journal of the American Statistics Association*, 105:713–726.

Witten, D. M., Tibshirani, R., and Hastie, T. (2009). A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, 10:515–534.

Wu, X., Ye, Y., and Subramanian, K. R. (2003). Interative analysis of gene interactions using graphical Gaussian models. In *BIOKDD03: 3rd ACM SIGKDD Workshop on Data Mining in Bioinformatics*, pages 63–69.

Xie, B., Pan, W., and Shen, X. (2007). Penalized model-based clustering with application to variable selection. *Journal of Machine Learning Research*, 8:1145–1164.

Xie, B., Pan, W., and Shen, X. (2008). Variable selection in penalized model-based clustering via regularization on grouped parameters. *Biometrics*, 64:921–930.

Xu, H., Caramanis, C., and Mannor, S. (2010). Robust regression and lasso. *IEEE Transactions on Information Theory*, 56:3561–3574.

Yang, L. and Tschernig, R. (1999). Multivariate bandwidth selection for local linear regression. *Journal of the Royal Statistical Society. Series B*, 61:793–815.

Young, F. W., de Leeuw, J., and Takane, Y. (1976). Regression with qualitative and quantitative variables: An alternating least squares method with optimal scaling features. *Psychometrika*, 41:505–529.

Yuan, M. (2010). High dimensional inverse covariance matrix estimation via linear programming. *Journal of Machine Learning Research*, 11:2261–2286.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B*, 70:53–71.

Yuan, M. and Lin, Y. (2007). Model selection and estimation in the Gaussian graphical model. *Biometrika*, 94:1–17.

Zhang, H. and Sheng, S. (2004). Learning weighted naïve Bayes with accurate ranking. In *Proceedings of the Fourth IEEE International Conference on Data Mining*, pages 567–570.

Zhang, H. H., Wahba, G., Lin, Y., Voelker, M., Ferris, M., Klein, R., and Klein, B. (2004). Variable selection and model building via likelihood basis pursuit. *Journal of the American Statistical Association*, 99:659–672.

Zhao, P., Rocha, G., and Yu, B. (2009). The composite absolute penalties family for grouped and hierarchical variable selection. *The Annals of Statistics*, 37:3468–3497.

Zhao, P. and Yu, B. (2006). On model selection consistency of Lasso. *Journal of Machine Learning Research*, 7:2541–2567.

Zhao, P. and Yu, B. (2007). Stagewise Lasso. *Journal of Machine Learning Research*, 8:2701–2726.

Zhu, J., Rosset, S., Hastie, T., and Tibshirani, R. (2003). 1-norm support vector machines. In *Neural Information Processing Systems*, pages 49–56.

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101:1418–1429.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B,*, 67:301–320.

Zou, H., Hastie, T., and Tibshirani, R. (2006). Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15:265–286.

Zou, H., Hastie, T., and Tibshirani, R. (2007). On the "degrees" of freedom of the lasso. *The Annals of Statistics*, 35:2173–2192.