Analysis of High-Dimensional Data with Partial Least Squares and Boosting

vorgelegt von

Diplom-Mathematikerin Nicole Krämer

aus Berlin

von der Fakultät IV – Elektrotechnik und Informatik –
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
- Dr.rer.nat. -

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender: Prof. Dr. Manfred Opper
Berichter: Prof. Dr. Ulrich Kockelkorn
Berichter: Prof. Dr. Gerhard Tutz

Tag der wissenschaftlichen Aussprache: 15. Dezember 2006

Berlin 2007
D 83

Acknowledgement

First of all, I want to express my deep gratitude to Prof. Dr. Ulrich Kockelkorn for his thorough supervision. I would also like to thank Prof. Dr. Gerhard Tutz, who agreed to be the second examiner of this thesis and who supported my work with his advise and encouragement. I am obliged to Dr. Jörg Betzin for introducing me to a lot of exciting open questions regarding Partial Least Squares and for sharing his thoughts with me. Finally, I benefited a lot from the exchange of ideas with Dr. Roman Rosipal, Athanassios Kondylis, Dr. Anne-Laure Boulesteix and Dr. Mohamed Hanafi. Thank you very much for the support!

Überblick

In der statistischen Analyse von hochdimensionalen Daten geht es darum, Zusammenhänge zwischen einer großen Menge p an Variablen mit Hilfe einer begrenzten Anzahl n an Beobachtungen zu modellieren. Gemeinsam sind allen Analysemethoden die beiden folgenden Ziele. Zum einen ist es wichtig, (latente) Strukturen in den Daten zu erkennen, um so eine handhabbare, niedrigdimensionale Repräsentation zu gewinnen. Zum anderen ist es oft von großer Wichtigkeit, verständliche und leicht zu interpretierende Modelle zu entwickeln. Die hohe Dimensionalität der Daten führt oft zu großen Problemen, denn für $p \gg n$ versagen die traditionellen statistischen Verfahren. Zudem ist die Struktur der Daten oft komplexer. Die beobachteten Größen sind nicht – wie in der klassischen Statistik üblich – Vektoren in einem endlichdimensionalen Vektorraum, sondern zum Beispiel Funktionen. Beispiele für diese Art von Datenstrukturen sind Zeitreihen oder Messungen in der Nah-Infrarot-Spektroskopie. In dieser Arbeit soll die Analyse von hochdimensionalen und komplexen Daten mit Hilfe von zwei Verfahren untersucht werden: Partial Least Squares und Boosting in Funktionenräumen.

Partial Least Squares (PLS) modelliert den Zusammenhang zwischen verschiedenen Blöcken von Variablen mit Hilfe sogenannter latenter Variablen. Im Fall von mehr als zwei Blöcken werden die PLS-Verfahren auch als Pfadmodelle bezeichnet und können als eine Erweiterung der Kanonischen Korrelationsanalyse angesehen werden. Die mathematischen Eigenschaften von PLS-Pfadmodellen sind zum großen Teil noch unerforscht. Zum Beispiel ist weder klar, ob die Algorithmen zur Berechnung der latenten Variablen im Pfadmodell numerisch konvergieren, noch, ob sie – falls sie konvergieren – Lösungen von sinnvollen Optimierungsproblemen darstellen. In dieser Arbeit wird ein sauberes mathematisches Gerüst für die Beschreibung der Pfadmodelle aufgestellt. Es wird gezeigt, dass zu einem großen Teil der PLS-Algorithmen (derjenigen mit mindestens einem Block im Modus A) tatsächlich kein zweimal differenzierbares Optimierungsproblem existiert. Zudem wird anhand von simulierten Daten gezeigt, dass für die PLS-Algorithmen im Modus B die Verfahren nur zu einer lokalen Lösung eines Optimierungsproblems konvergieren können.

PLS kann auch in Regressionsproblemen eingesetzt werden, in dem man die erklärenden und die abhängigen Variablen als jeweils einen Block auffasst. In diesem Fall ermöglicht PLS zudem eine Dimensionsreduktion der Daten, die wiederum hoffentlich zu besseren Vorhersagen führt. In dieser Arbeit wird eine Erweiterung von PLS um einen Strafterm vorgestellt und auf die Schätzung von generalisierten additiven Modellen (GAM's) angewandt. Es zeigt sich, dass insbesondere für hochdimensionale Daten dieser Ansatz eine gute Alternative zu klassischen GAM-Verfahren ist. Ausgehend von der bereits bekannten Verbindung von PLS und dem Konjugierten-Gradienten-Verfahren (aus der numerischen linearen Algebra) wird gezeigt, dass PLS mit Strafterm äquivalent zu einem vorkonditionierten Konjugierten-Gradienten-Verfahren ist. Die Konditionierungsmatrix wird dabei durch den Strafterm bestimmt. Im Anschluss werden die Beziehungen zwischen der linearen Algebra und PLS ausgenutzt, um die sogenannten "Shrinkage"-Eigenschaften von PLS empirisch zu untersuchen. Darüber hinaus wird ein unverzerrter Schätzer für die Freiheitsgrade von PLS ermittelt.

Boosting ist ein Verfahren aus dem Bereich des Maschinellen Lernens. Die grundlegende Idee ist, verschiedene einfache Vorhersagemodelle so zu kombinieren, dass diese Kombination zu sehr viel besseren Vorhersagen führt. In dieser Arbeit werden Boostingverfahren für komplizierte Datenstrukturen entwickelt. Dabei interessiert uns vor allen Dingen der Fall, in dem die beobachteten Einflussgrößen Funktionen bzw. diskrete Messungen von Funktionen sind. Die gängigen Boosting-Methoden basieren implizit auf der Annahme, dass die Einflussvariablen Werte in einem endlichdimensionalen Vektorraum annehmen. Es wird gezeigt, dass die Erweiterung auf unendlichdimensionale Funktionenräume ohne Weiteres möglich ist. Zudem wird illustriert, wie man mit Hilfe von Boostingverfahren wichtige Charakteristika der Funktionen aufdeckt und wie man damit leicht interpretierbare und visualisierbare Modelle erzeugt. Dies geschieht durch eine Transformation der Ausgangsdaten mit Hilfe von Wavelet- bzw. Fouriertransformationen.

Outline

The crucial task in the statistical analysis of high-dimensional data is to model relationships between a large amount p of variables based on a small number n of observations. Quite generally, we pursue two goals. On the one hand, it is important to detect (latent) structures in the data in order to obtain a feasible, low-dimensional representation. On the other hand, we often need simple and comprehensible models that can be interpreted. The high-dimensionality of the data often forms an obstacle, as for $p \gg n$, the traditional statistical techniques fail to produce satisfactory results. Furthermore, the structure of the data can be complex. The observed variables are not – as usually assumed in classical statistics – elements of a finite-dimensional vector space, but , for instance, functions. Examples for this type of data are times series or experiments from the field of near-infra-red spectroscopy. In this work, we investigate high-dimensional and complex data with the help of two methods: Partial Least Squares and Boosting for functional data.

Partial Least Squares (PLS) models the relationship between different blocks of variables in terms of so-called latent variables. In the case of more than two blocks, the PLS-techniques are also called path models and can be seen as a generalization of Canonical Correlation Analysis. The mathematical properties of PLS are for the most parts not yet established. For example, it is neither known whether the PLS algorithms converge numerically, nor – in the case that they converge – if they produce solutions of a sensible optimization criterion. In this work, we establish a sound mathematical framework for the description of PLS path models. We show that for a large part of the PLS algorithms (those with at least one block in mode A), there is indeed no twice-differentiable optimization problem. Furthermore, we show on simulated data that the PLS algorithms in mode B can converge only to a local solution of an optimization problem.

PLS can also be used to solve regression problems. In this case, it leads to a substantial reduction of the dimension of the data, which hopefully leads to better prediction rules. In this work, we present an extension of PLS using penalization techniques. This method is then used to estimate generalized additive models (GAM's). This approach turns out to be a good alternative to traditional GAM-methods in the case of high-dimensional data. Based on the well-known relationship between PLS and the conjugate gradient technique (a method from the field of numerical linear algebra), we prove that penalized PLS is equal to a preconditioned conjugate gradient technique. Here, the preconditioner is determined by the penalty term. Subsequently, we exploit the connections between PLS and linear algebra to investigate empirically the so-called shrinkage properties of PLS. In addition, we derive an unbiased estimate of the degrees of freedom of PLS.

Boosting has its seed in the machine learning community. The basic idea is to combine several, simple models in such a way that their combination leads to better prediction rules. In this work, we develop Boosting algorithms for complex data structures. Our focus is on data that are (discrete) measurements of curves. The established Boosting methods implicitly assume that the observed variables lie in a finite-dimensional vector space. We show that an extension of Boosting to infinite-dimensional function spaces is straightforward. Furthermore, we illustrate how to detect relevant features of the investigated functions and how to produce simple and interpretable models. This is done by applying wavelet or Fourier transformations to the data and by then applying suitable Boosting algorithms.

Contents

Chapter 1

Preliminaries

This chapter serves as a reference for some basic concepts that are essential for the rest of this work.

1.1 Notation and Guideline

Matrices are denoted by bold upper case letters $\boldsymbol{X}, \boldsymbol{A}, \boldsymbol{U}, \ldots$ and vectors are denoted by bold lower case letters $\boldsymbol{v}, \boldsymbol{y}, \ldots$. This rule is also valid if we use Greek letters. The transpose of a matrix or a vector is indicated by a superscript t as in $\boldsymbol{A}^t$ and $\boldsymbol{v}^t$. The Moore-Penrose inverse of a matrix $\boldsymbol{A}$ is denoted by $\boldsymbol{A}^-$. Spaces are either in calligraphic style $(\mathcal{X}, \mathcal{Y}, \mathcal{F}, \ldots)$ or in blackboard bold style $(\mathbb{R}, \mathbb{N}, \mathbb{Z}, \mathbb{F}, \ldots)$. Functions are usually denoted by lower case letters as $f, g, h, \ldots$. The covariance and variance of random variables are denoted by Cov and Var. Their respective empirical counterparts are denoted by cov and var.

The table of contents hopefully reveals the rough structure of this work. Let us remark that Chapters 1, 2 and 8 serve as summaries of well-known concepts and do not contain any fundamentally new ideas. Before starting to read further, it might be beneficial to have a look at Chapter A in the appendix. There, we collect some basic mathematical principles that constantly emerge in this work.

All experiments and simulations are performed in `R` (R Development Core Team 2005).

1.2 Learning from Data

Let us introduce the general learning problem. We consider two random variables X and Y which define a random variable $Z = X \times Y$ on a product space $\mathcal{X} \times \mathcal{Y}$. We assume that there is a relationship between X and Y in the sense that given X, we can predict the outcome of Y with a high accuracy. We do not know the distribution of Z, but we observe a finite set

$$\mathcal{S} = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subset \mathcal{X} \times \mathcal{Y}$$

of observations. This set is called the sample. We assume that the observations are drawn independently from $Z = X \times Y$. The classical statistical approach is to identify the process Z that generates the sample. Assuming a class of models that is appropriate to describe Z, the model parameters of Z are estimated with the help of $\mathcal{S}$. From this, we can infer the conditional distribution of Y given X. In order to find an estimate $\widehat{f}$ of a function

$$F : \mathcal{X} \to \mathcal{Y} \tag{1.1}$$

that predicts an outcome y for a fixed value $x \in \mathcal{X}$ via $\widehat{f}(x)$, we have to evaluate the decision $\widehat{f}(x)$ if the correct outcome is y. This is done via a loss function

$$L : \mathcal{Y} \times \mathcal{Y} \;\; \to \;\; \mathbb{R}\,, \tag{1.2}$$

and the risk at a certain point $z = (x, y)$ is defined as $L(y, \widehat{f}(x))$. The optimal function is the one that minimizes the expected risk

$$R(f) \;\; = \;\; E_{X \times Y} \left[L(Y, f(X)) \right]. \tag{1.3}$$

over all functions f. If the distribution of Z is known, the optimal function can often be determined explicitly. For example, if we consider regression problems (that is $\mathcal{Y} = \mathbb{R}$) and L is the quadratic loss function

$$L(y, y') \;\; = \;\; (y - y')^2\,, \tag{1.4}$$

the optimal function is the conditional expectation $f(x) = E[Y|X = x]$.

In contrast, statistical learning theory focuses on mimicking the underlying process. The primary task is to find a function that minimizes (1.3) and not to identify the whole process Z. As the distribution of Z is not known, it is not possible to minimize (1.3). As a consequence, we have to estimate the optimal function based on the available sample $\mathcal{S}$. A popular approach is to fix a class of functions $\mathcal{F}$ and to minimize the empirical risk

$$\widehat{R}(f) \;=\; \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i)) \tag{1.5}$$

over all elements $f \in \mathcal{F}$. The quantity $\widehat{R}(f)$ is also called the training error. Sometimes, a regularization term $r(f)$ is added to (1.5). This strategy is called (regularized) empirical risk minimization. A combination of a loss function, a class of functions and a regularization term defines a strategy to estimate (1.1). Depending on the scientific community, a strategy is called a model, an algorithm, a fitting method or a learner. In this work, we use these terms more or less synonymously. A more precise specification of a learning strategy is given in Chapter 2. Some remarks on the term "model" are however necessary. In the literature, it is used to describe two different aspects of learning from data. On the one hand, a model is a description of how the data is generated. E.g. in the learning task (1.1), we can determine the structure of the function F (linear, polynomial) or assume that Z belongs to a certain class of distributions. On the other hand, a model is a strategy how to estimate the generation of the data. This refers to the description of a learning strategy described above. For every sample $\mathcal{S}$ and every learning strategy, we obtain an estimate of the function F, which we denote by $\widehat{f}$.

1.3 The Regression Model

In this work, we are mainly concerned with multivariate regression problems that have a one-dimensional response. That is we assume that $\mathcal{Y} = \mathbb{R}$ and $\mathcal{X} = \mathbb{R}^p$. In statistics, regression problems are usually modeled in the following way:

$$Y_i \;=\; F(X_i) + \varepsilon_i,\; i = 1, \ldots, n\,.$$

The predictor X is a multivariate, $p-$dimensional random variable. From now on, the predictors are assumed to be deterministic and only the response is assumed to be stochastic. In addition, we claim that the error terms are uncorrelated with zero mean and equal variance. In compact form, the regression model can determined via

$$Y_i \quad = \quad F(x_i) + \varepsilon_i,\ i = 1, \ldots, n\,, \tag{1.6}$$

with

$$E\left[\varepsilon_i\right] \quad = \quad 0$$

and

$$\mathrm{Cov}\left(Y_1, \ldots, Y_n\right) \quad = \quad \sigma^2 \boldsymbol{I}_n\,. \tag{1.7}$$

Here, $\boldsymbol{I}_n$ is the identity matrix of dimension n. It follows immediately that

$$E\left[Y_i\right] \quad = \quad F(x_i)\,. \tag{1.8}$$

Let us now consider multivariate linear regression problems. If the function F in (1.6) is assumed to be linear, the regression model can be represented by the multivariate linear regression model

$$Y_i \quad = \quad \boldsymbol{x}_i^t \boldsymbol{\beta} + \varepsilon_i\,. \tag{1.9}$$

Given data $\mathcal{S}$, the estimation of (1.6) is transformed into the estimation $\widehat{\boldsymbol{\beta}}$ of the regression vector $\boldsymbol{\beta}$. Recall that the number of variables is p and that the number of examples is n. We set

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x}_1^t \\ \ldots \\ \boldsymbol{x}_n^t \end{pmatrix} \in \mathbb{R}^{n \times p}, \quad \boldsymbol{y} = \begin{pmatrix} y_1 \\ \ldots \\ y_n \end{pmatrix} \in \mathbb{R}^n\,.$$

An intercept β_0 can be included into (1.9) by attaching an additional column that consist of 1's. Another possibility is to estimate (1.9) based on centered data. For this reason, we require that both $\boldsymbol{X}$ and $\boldsymbol{y}$ are centered.

The Ordinary Least Squares (OLS) estimator $\widehat{\boldsymbol{\beta}}_{OLS}$ is the solution of the optimization problem

$$\arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \left(y_i - \boldsymbol{x}_i^t \boldsymbol{\beta}\right)^2 = \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 .$$

Note that this equals the minimization of the empirical risk $\widehat{R}(\boldsymbol{\beta})$ defined (1.5) for the quadratic loss function (1.4) and $\mathcal{F}$ equal to the space of all linear functions. If we differentiate the empirical risk with respect to $\boldsymbol{\beta}$, we realize that the solution $\widehat{\boldsymbol{\beta}}_{OLS}$ must fulfill

$$\boldsymbol{X}^t\boldsymbol{X}\widehat{\boldsymbol{\beta}}_{OLS} = \boldsymbol{X}^t\boldsymbol{y} .$$

There is always a solution, as $\boldsymbol{X}^t\boldsymbol{y}$ lies in the space spanned by the columns of $\boldsymbol{X}^t\boldsymbol{X}$. However the solution is not unique if $\boldsymbol{X}^t\boldsymbol{X}$ does not have full rank. This is for example the case if there are more variables than observations. If there is no unique solution, we define the OLS estimator as the solution with minimal euclidean norm. It follows from proposition A.14 that

$$\widehat{\boldsymbol{\beta}}_{OLS} = \left(\boldsymbol{X}^t\boldsymbol{X}\right)^- \boldsymbol{X}^t\boldsymbol{y} .$$

We now use the singular value decomposition

$$\boldsymbol{X} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^t$$

of $\boldsymbol{X}$ that is defined in (A.8). Furthermore, $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^t\boldsymbol{\Sigma}$ is the matrix of eigenvalues of $\boldsymbol{X}^t\boldsymbol{X}$. Set

$$\boldsymbol{s} = \boldsymbol{\Sigma}\boldsymbol{V}^t\boldsymbol{y} . \tag{1.10}$$

In this work, we use one of the following representations of the OLS estimator:

$$\widehat{\boldsymbol{\beta}}_{OLS} = \left(\boldsymbol{X}^t\boldsymbol{X}\right)^- \boldsymbol{X}^t\boldsymbol{y} = \boldsymbol{U}\boldsymbol{\Lambda}^-\boldsymbol{s} = \sum_{i=1}^{\mathrm{rk}(\boldsymbol{X})} \frac{\boldsymbol{v}_i^t\boldsymbol{y}}{\sqrt{\lambda_i}}\boldsymbol{u}_i = \sum_{i=1}^{\mathrm{rk}(\boldsymbol{X})} \boldsymbol{z}_i , \tag{1.11}$$

with

$$\boldsymbol{z}_i = \frac{\boldsymbol{v}_i^t\boldsymbol{y}}{\sqrt{\lambda_i}}\boldsymbol{u}_i .$$

The OLS estimator usually performs poorly on new data if the number of examples is small compared to the number of observations or if $\boldsymbol{X}$ is highly collinear. Both phenomena lead to a covariance matrix $(1/n)\boldsymbol{X}^t\boldsymbol{X}$ that is (almost) singular, which affects the statistical properties of the estimator. This is discussed in great detail in Section 7.1.

1.4 Duality and Kernel Methods

In this section, we briefly recapitulate the concept of dual representations and the kernel trick. Let us consider the following example. In Section 1.3, we introduced the linear regression model (1.9). For any estimate $\widehat{\boldsymbol{\beta}}$ of $\boldsymbol{\beta}$, we can predict the value of y for a new observation $\boldsymbol{x}_{new}$ via the linear function

$$\widehat{y}_{new} \quad = \quad \boldsymbol{x}_{new}^t\widehat{\boldsymbol{\beta}} = \left\langle \boldsymbol{x}_{new}, \widehat{\boldsymbol{\beta}} \right\rangle . \tag{1.12}$$

Now suppose that we want to transform the original data $\boldsymbol{X}$ before applying OLS. One reason to do so is to model nonlinear relationships between predictor variables and response. If we have e.g. $p = 2$ predictor variables and we want to estimate a function (1.1) with F assumed to be a polynomial of degree ≤ 2. How can we use a method that is designed for linear regression problems (e.g. OLS) to solve this problem? We simply transform the data via a map

$$\Phi\left(x, x'\right) \quad = \quad \left(1, \sqrt{2}x, \sqrt{2}x', \sqrt{2}xx', x^2, x'^2\right) \tag{1.13}$$

and apply the linear algorithm to $\boldsymbol{y}$ and $\Phi\left(\boldsymbol{X}\right)$. A transformation is also necessary if the observed data are not yet embedded in an euclidean space. If for instance, the variables are on a nominal scale, we have to transform the variables into dummy variables and then plug the transformed data into any learning method designed for estimating linear relationships.

The transformation map

$$\Phi : \mathcal{X} \to \mathbb{F} \tag{1.14}$$

is called the feature map. The spaces $\mathcal{X}$ and $\mathbb{F}$ are called input space and feature space respectively. In order to apply a linear algorithm in $\mathbb{F}$, it is often necessary to assume that $\mathbb{F}$ is a Hilbert space. An important observation is the following. In a lot of cases, the vector $\widehat{\boldsymbol{\beta}}$ that defines the linear function in (1.12) is a linear combination of the data points,

$$\widehat{\boldsymbol{\beta}} = \sum_{i=1}^{n} \alpha_i \Phi\left(\boldsymbol{x}_i\right) . \tag{1.15}$$

The coefficients α_i are called dual variables. If we plug (1.15) into $\langle \Phi(\boldsymbol{x}_{new}), \widehat{\boldsymbol{\beta}} \rangle$, we realize that the linear function only depends on inner products between transformed data points,

$$f(\boldsymbol{x}) = \langle \Phi(\boldsymbol{x}), \widehat{\boldsymbol{\beta}} \rangle = \sum_{i=1}^{n} \alpha_i \left\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}_i) \right\rangle .$$

In this case, the estimation of the dual variables can be done by using the $n \times n$ Gram matrix

$$\boldsymbol{K} = \left(\langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j) \rangle \right)_{i,j=1,\dots,n} .$$

Note that condition (1.15) holds for the OLS estimate. This follows e.g. from (1.11), as the vectors $\boldsymbol{u}_i$ are a basis of the row space of $\boldsymbol{X}$. (Recall the singular value decomposition (A.8) of $\boldsymbol{X}$.) de Bie et al. (2005) describe various multivariate methods in terms of their primal and dual representation.

As we only need inner products in the dual representation, we do not have to map the data points explicitly to a feature space, it suffices to compute the function

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \tag{1.16}$$

with

$$k(\boldsymbol{x}, \boldsymbol{z}) = \langle \Phi(x), \Phi(z) \rangle . \tag{1.17}$$

The estimated function is

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i) .$$

The function k is called a kernel. Note that in example (1.13),

$$k(x, z) = \left(1 + \langle x, z \rangle\right)^2 .$$

The replacement of the usual inner product by the inner product in some feature space is called the kernel trick. Note that we do not even require the input space to be an inner product space at all. Literature on the kernel trick and its applications is abundant. A detailed treatise of the subject can be found in Schölkopf & Smola (2002).

So instead of defining a feature map, we define an admissible kernel function, that is, a function (1.16) which can be defined via a map (1.14) such that (1.17) holds. The choice of the optimal kernel is part of the model selection procedure that is illustrated in Chapter 2. What are the merits of this dual representation? We already mentioned the extension to nonlinear models. Furthermore, from a technical point of view, if $p \gg n$, the computation in the dual representation is usually faster than the computation in the primal representation. Finally, we can extend the whole multivariate machinery to spaces $\mathcal{X}$ of infinite dimension or with a complex structure by defining an appropriate inner product. An important example is the analysis of functional data, that is, $\mathcal{X}$ is a space of functions over some domain. This subject will be treated in more detail in Chapter 8.

Chapter 2

Model Selection

We now recapitulate the main tools to evaluate the performance of a learning method. The contents of this chapter are a summary of the corresponding chapter in Hastie et al. (2001). As described in Chapter 1, we estimate the relationship (1.1) by applying an appropriate fitting method. Normally, we we do not fit a single model but a group of models and have to choose the best model. This is usually called model selection. We therefore need a strategy how to select the best model out of a pool of models. After the best model is chosen, we have to evaluate its quality. This is called model validation. Recall that we evaluate a model in terms of its expected risk (1.3). As this quantity is usually unknown, we need a good estimate. In what follows, we focus on model selection and remark that the risk of the selected model should be estimated on a test set that was neither involved in the fitting process nor in the selection process.

In the rest of the chapter, we consider the general regression model (1.6). Given data, we fit a model and call the fitted function $\widehat{f}$. In order to evaluate the quality of the fitting method, we start by computing the expected risk of $\widehat{f}$ at a point x_i,

$$R\left(\widehat{f}(x_i)\right) = E_{Y^{new}}\left[L\left(Y^{new}, \widehat{f}(x_i)\right)\right]. \tag{2.1}$$

Here, Y^{new} is a new observation at point x_i. Note that the quantity $R\left(\widehat{f}(x_i)\right)$ depends on the sample $\mathcal{S}$ that is used to estimate $\widehat{f}$. If we use the quadratic loss function (1.4), the expected risk of $\widehat{f}$ at x_i equals

$$\begin{aligned} R\left(\widehat{f}(x_i)\right) &= E_{Y^{new}}\left[\left(Y^{new} - \widehat{f}(x_0)\right)^2\right] \\ &= E_{Y^{new}}\left[\left(Y^{new} - E[Y^{new}] + E[Y^{new}] - \widehat{f}(x_i)\right)^2\right] \\ &\overset{(1.8)}{=} E_{Y^{new}}\left[(Y^{new} - E[Y^{new}])^2 + \left(F(x_i) - \widehat{f}(x_i)\right)^2\right] \\ &= \sigma^2 + \left[\left(F(x_i) - \widehat{f}(x_i)\right)^2\right] . \end{aligned} \tag{2.2}$$

The term σ^2 is called the irreducible error. The second term depends on the data that we used to fit the model. If we are interested in the quality of our learning strategy that is used to obtain the estimate $\widehat{f}$, we compute the expected value of $R\left(\widehat{f}(x_i)\right)$ with respect to the data $Y^{(n)} = (Y_1, \ldots, Y_n)$. For the quadratic loss, we yield

$$E_{Y^{(n)}} E_{Y^{new}}\left[\left(Y^{new} - \widehat{f}(x_i)\right)^2\right] = \sigma^2 + \text{bias}^2\left(\widehat{f}(x_i)\right) + \text{Var}\left(\widehat{f}(x_i)\right) .$$

We expect the bias to decrease for more complex models and the variance to increase. If we choose a very simple model with a low variance, we might fail to capture the relevant structure of the data. If we fit a very complex model that is almost unbiased, we have a good explanation of the available sample but will probably fail to predict on new observations. The latter phenomenon is called overfitting. In Section 7.1, we study this bias-variance trade-off in more detail for linear shrinkage estimators.

Let us return to the essential question of this chapter. How do we select a model? Let us assume that we have a lot of data at hand. In this case, we can proceed in the following way. We split the data set into two parts: a training set and a validation set. We fit the models on the training set. We then compare their performance on the validation set. Note however that in most situations, the amount of available data is limited and we cannot afford to exclude a fraction of the data from model estimation. We therefore need different strategies to estimate the risk of a model. Roughly, we distinguish two different approaches. In the first approach, we repeat a random splitting into training and test set several times. This is called cross-validation and is discussed in Section 2.1. In the second approach, that is presented in Section 2.3, we use the fact that the risk of a model can be estimated in terms of its empirical risk and its complexity. More precisely, the complexity can be expressed in terms of degrees of freedom.

2.1 Cross Validation

The cross-validation technique (Stone 1974, Stone 1977) produces estimates of the risk $R(\widehat{f})$ of a model $\widehat{f}$. Recall that in order to select the best model, it is suggested to split the data into a training set and a validation set. The model is fitted on the training set and its performance is estimated on the validation set. As in a lot of cases, we do not have enough data at hand, a more refined strategy is pursued. We randomly split the data into K parts of roughly the same size. For $k = 1, \ldots, K$, we remove the kth block from the data and fit the model to the remaining $K-1$ parts. The kth block is used as a test set. That is, for each block k, we obtain an estimate of the risk of the model that was fitted on the other $K-1$ blocks. Finally, we average these K estimates. More formally, the function

$$\kappa : \{1, \ldots, n\} \quad \to \quad \{1, \ldots, K\}$$

assigns to the ith example its block membership. We denote by $\widehat{f}^{-k}$ the function that was fitted on all but the kth block.

Definition 2.1. The K-fold cross-validation error is

$$\mathrm{CV}\left(\widehat{f}\right) \quad = \quad \frac{1}{n}\sum_{i=1}^{n} L(y_i, \widehat{f}^{-\kappa(i)}(x_i))\,. \tag{2.3}$$

For $K = n$, this is called the leave-one-out error.

Let us note that the computational costs of K-fold cross-validation can become very high if K is large. In this case, we have to fit the models several times, which can be very time-consuming.

2.2 Degrees of Freedom

As already mentioned above, the quality of a model or a function $\widehat{f}$ is measured in terms of its expected risk (1.3). As this risk cannot be computed, we need a good estimate. The empirical risk (1.5) of $\widehat{f}$ is obviously not a good estimate of (1.3). We expect the empirical risk to be lower than the true risk, as we use the same data set to fit the model $\widehat{f}$ and to asses its performance. If we use the training data for model assessment, this leads to overoptimistic estimates of the risk. The gap between empirical and test error is usually particularly large for very complex models. In order to get a good estimate of the expected risk, we have to measure the gap between empirical error and the expected risk.

Recall the general regression model (1.6). Note that we defined the expected risk of $\widehat{f}$ at a data point x_i in (2.1). The estimated function $\widehat{f}$ depends on the sample $\mathcal{S}$, that is it depends on $Y^{(n)} = (Y_1, \ldots, Y_n)$. If we average over all points $x_1, \ldots, x_n$ and compute the expectation with respect to $Y^{(n)}$, we obtain the expected in-sample risk of our strategy,

$$R_{in} = R_{in}(x_1, \ldots, x_n) = E_{Y^{(n)}}\left[\frac{1}{n}\sum_{i=1}^{n} R\left(\widehat{f}(x_i)\right)\right].$$

The difference between R_{in} and the expected empirical risk is called the optimism:

$$\mathrm{op} = R_{in} - E_{Y^{(n)}}\left[\widehat{R}(\widehat{f})\right] = E_{Y^{(n)}}\left[\frac{1}{n}\sum_{i=1}^{n}\left\{R\left(\widehat{f}(x_i)\right) - \widehat{R}\left(\widehat{f}(x_i)\right)\right\}\right].$$

The key point is to find a good estimate $\widehat{\mathrm{op}}$ of op. We can then estimate the in-sample risk of a model in the following way:

$$\widehat{R_{in}} \;=\; \widehat{R} + \widehat{op}\,. \tag{2.4}$$

Proposition 2.2. *For the quadratic loss function (1.4), the optimism of a fitting method is*

$$op \;=\; \frac{2}{n}\sum_{i=1}^{n} Cov\left(\widehat{f}(x_i), Y_i\right). \tag{2.5}$$

Proof. It follows from (2.2) that

$$R_{in} \;=\; \sigma^2 + \frac{1}{n}\sum_{i=1}^{n} E_{Y^{(n)}}\left(F(x_i) - \widehat{f}(x_i)\right)^2.$$

Next, we have

$$\begin{aligned}
\widehat{R}(\widehat{f}) &= \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \widehat{f}(x_i)\right)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - F(x_i) + F(x_i) - \widehat{f}(x_i)\right)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}(Y_i - F(x_i))^2 + 2\frac{1}{n}\sum_{i=1}^{n}(Y_i - F(x_i))\left(F(x_i) - \widehat{f}(x_i)\right) \\
&\quad + \frac{1}{n}\sum_{i=1}^{n}\left(F(x_i) - \widehat{f}(x_i)\right)^2 .
\end{aligned}$$

It follows that

$$E_{Y^{(n)}}\widehat{R}(\widehat{f}) = \sigma^2 - \frac{2}{n}\sum_{i=1}^{n} Cov(Y_i, \widehat{f}(x_i)) + \frac{1}{n}\sum_{i=1}^{n} E_{Y^{(n)}}\left(F(x_i) - \widehat{f}(x_i)\right)^2 .$$

This concludes the proof. □

Before proceeding, it is beneficial to introduce a compact representation of a fitting method. If we denote by $\widehat{f}$ the fitted function that is obtained by using the sample $\mathcal{S}$, we define the following map

$$\begin{aligned}
H : \mathbb{R}^n &\rightarrow \mathbb{R}^n , \\
H(\boldsymbol{y}) &= \left(\widehat{f}(x_1), \ldots, \widehat{f}(x_n)\right)^t = \widehat{\boldsymbol{y}} .
\end{aligned} \tag{2.6}$$

Note that the function H depends on $x_1, \ldots, x_n$. If this function is linear in $\boldsymbol{y}$, we speak of a linear fitting method or a linear learner. In this case, H can be represented by a $n \times n$ matrix $\boldsymbol{H}$ that is called the hat-matrix.

Definition 2.3 (Degrees of Freedom)**.** The degrees of freedom of a fitting method that is represented by H is defined as

$$\mathrm{df}(H) = \frac{1}{\sigma^2}\sum_{i=1}^{n}\mathrm{Cov}\left(\widehat{f}(x_i), Y_i\right) .$$

In particular,

$$\mathrm{op} = \frac{2}{n}\sigma^2 \mathrm{df}(H) .$$

In order to find a better description of (2.5), it is necessary to assume that the error variables ε_i in (1.6) are normally distributed. The next useful lemma is due to Stein (1981).

Lemma 2.4 (Stein's Lemma). *Assume that $X \sim N(\mu, \sigma^2)$ is a univariate random variable with density function ϕ and that $g : \mathbb{R} \to \mathbb{R}$ is a differentiable function such that*

$$\lim_{x \to \pm\infty} g(x)\phi(x) \;=\; 0\,. \tag{2.7}$$

We have

$$Cov\,(g(X), X) \;=\; \sigma^2 E\left[g'(X)\right]\,.$$

We can easily extend Stein's lemma to multivariate random variables.

Lemma 2.5 (Multivariate Stein's Lemma). *Assume that*

$$X = (X_1, \ldots, X_n) \;\sim\; N\left(\boldsymbol{\mu}, \sigma^2 \boldsymbol{I}_n\right)$$

is a multivariate random variable with density function $\phi(x) = \prod_{i=1}^{p} \phi_i(x_i)$. Let

$$g = (g_1, \ldots, g_p) \;:\; \mathbb{R}^p \to \mathbb{R}^p$$

be a differentiable function which fulfills

$$\lim_{x \to \pm\infty} g_i(x)\phi_i(x) \;=\; 0\,. \tag{2.8}$$

We have

$$\sum_{i=1}^{n} Cov\,(g_i(X), X_i) \;=\; \sigma^2 E\left[trace\left(\frac{\partial}{\partial X} g(X)\right)\right]\,.$$

Proof. We fix $i \in \{1, \ldots, p\}$ and set

$$X_{-i} \;=\; (X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_p)\,.$$

We have

$$\begin{aligned} Cov\left(g_i(X), X_i\right) &= E_X\left[g_i(X)\left(X_i - E[X_i]\right)\right] \\ &= E_{X_{-i}} E_{X_i|X_{-i}}\left[g_i(X)\left(X_i - E[X_i]\right)\right] \\ &= E_{X_{-i}} E_{X_i}\left[g_i(X)\left(X_i - E[X_i]\right)\right] \end{aligned}$$

The last equality is valid as we assume that the variables X_i are independent. In the expression $E_{X_i}\left[g_i(X)\left(X_i - E[X_i]\right)\right]$ in the last line, $g_i(X)$ only varies in X_i and the other components are considered to be constant. We can now apply Stein's lemma 2.4 to $g_i(X_i)$ and X_i and obtain

$$Cov\left(g_i(X), X_i\right) = \sigma^2 E_{X_{-i}} E_{X_i}\left[\left(\frac{\partial}{\partial X_i} g_i(X)\right)\right] = \sigma^2 E_X\left[\left(\frac{\partial}{\partial X_i} g_i(X)\right)\right].$$

This proves the lemma. □

Corollary 2.6. *Assume that the function H defined in (2.6) is differentiable. If Y_i is normally distributed, $Y_i \sim N(F(x_i), \sigma^2)$ and H fulfills assumption (2.8), we have*

$$\sum_{i=1}^{n} Cov(\widehat{Y}_i, Y_i) = \sigma^2 E\left[trace\left(\frac{\partial}{\partial Y^{(n)}} H\left(Y^{(n)}\right)\right)\right].$$

In particular,

$$df(H) = E\left[\text{trace}\left(\frac{\partial H\left(Y^{(n)}\right)}{\partial Y^{(n)}}\right)\right].$$

In this case,

$$\widehat{df}(H) = \text{trace}\left(\frac{\partial H\left(Y^{(n)}\right)}{\partial Y^{(n)}}\right) \tag{2.9}$$

is an unbiased estimate for the degrees of freedom of H. If the learner is linear in $\boldsymbol{y}$, i.e. $\widehat{\boldsymbol{y}} = \boldsymbol{H}\boldsymbol{y}$ with $\boldsymbol{H} \in \mathbb{R}^{n \times n}$, we yield

$$\text{df}(H) = \text{trace}(\boldsymbol{H}).$$

As an illustration, let us consider the OLS estimator defined in (1.11). The function H corresponding to this estimator is

$$\widehat{\boldsymbol{y}} = \boldsymbol{X}\left(\boldsymbol{X}^t\boldsymbol{X}\right)^{-}\boldsymbol{X}^t\boldsymbol{y} = \mathcal{P}_{\boldsymbol{X}}\boldsymbol{y}\,.$$

The trace of the projection operator equals the dimension of the space spanned by the columns of $\boldsymbol{X}$. We obtain the well-known result

$$\mathrm{df}(OLS) \quad = \quad \mathrm{rank}\left(\boldsymbol{X}\right)\,.$$

2.3 Information Criteria

We now return to the estimation of the risk of the model. Information criteria are based on the idea that the quality of a model depends on its training error and on its complexity. Different approaches lead to different amounts of penalization of the complexity of a model. Information criteria differ in they way how much they penalize the complexity of the model. We already remarked in (2.4) that we can estimate the in-sample risk in terms of the empirical risk and its complexity. The Akaike Information Criterion (AIC) (Akaike 1973) is a generalization of (2.4). It is based on a general, asymptotic relationship between the Kullback-Leibler Information Criterion and maximum likelihood theory. We do not want to go too much into detail and refer e.g. to Burnham & Anderson (2004). In the case of normally-distributed error terms ε_i, the AIC-criterion is equivalent to (2.4),

$$\mathrm{AIC}(\widehat{f}) \quad = \quad \widehat{R}(\widehat{f}) + \frac{2}{n}df(H)\sigma^2\,.$$

The quantity σ can be estimated via

$$\widehat{\sigma}^2 \quad = \quad \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \widehat{f}(x_i)\right)^2\,.$$

We choose the model that minimizes the AIC information criterion

$$\widehat{f}_{AIC} \quad = \quad \arg\min_{\widehat{f}} \mathrm{AIC}(\widehat{f})\,.$$

As the general AIC criterion only holds asymptotically for large values of n, there is a corrected version of the AIC criterion for small sample sizes (Hurvich & Tsai 1989). Another criterion that is based on the principle of minimum description length (Hansen & Yu 2001) and that is used in Chapter 9 is

$$\begin{aligned} gMDL\left(\widehat{f}\right) &= \log\left(\frac{n}{n-df(H)}\widehat{R}(\widehat{f})\right) \\ &\quad + \frac{df(H)}{n}\left(\log\left(\sum_{i=1}^{n} y_i^2 - n\widehat{R}(\widehat{f})\right) - \log\left(df(H)\frac{n}{n-df(H)}\widehat{R}(\widehat{f})\right)\right) . \end{aligned}$$

The last two criteria penalize the complexity of a model more strongly than the AIC criterion.

Chapter 3

Partial Least Squares Path Models

Partial Least Squares (PLS) path models (Wold 1982, Wold 1985) are a framework for modeling linear relationships between several blocks of variables. In this sense, they can be seen as a generalization and an extension of Canonical Correlation Analysis (Hotelling 1936) to more than two blocks of variables. The relationship between different blocks are modeled in the following way. For each block of variables, we look for a latent variable – that is, a linear combination of these variables – such that latent variables that are assumed to be linked are highly correlated. These latent variables are estimated with algorithms that have a power method flavor.

The statistical and mathematical theory of PLS path model has not been fully established. In fact, PLS is defined via algorithms as in Wold (1982) and Lohmöller (1989) and not via a statistical model or an empirical optimization problem. Some fundamental questions have not been answered. For example, it is not guaranteed that the PLS algorithms converge numerically (although convergence is always observed in practice). More severely, for a wide class of algorithms (those with at least one block in mode A), it is not known if the latent variables computed by PLS are at least a stationary point of a sensible optimization problem. We show that this is not the case, if we require that the objective function of the optimization problem is at least twice differentiable. For a different class of algorithms (those with all blocks in mode B), it is known (Mathes 1993) that the solution of the PLS algorithms is a stationary point of a sensible optimization problem. It is however not known if we always obtain the optimal solution. We provide a negative answer to this problem.

3.1 The Partial Least Squares Framework

In the PLS path model framework, we model linear relationships between different blocks of variables. One variable is a vector of length n, as we observe n examples. We have K blocks of variables, and each block consists of p_k variables, which are subsumed in a matrix

$$\boldsymbol{X}_k \in \mathbb{R}^{n \times p_k} , \, k = 1, \ldots, K .$$

In total we have

$$p = \sum_{k=1}^{K} p_k$$

variables. The blocks of variables are called manifest variables in the PLS literature. The relationship between blocks is represented by a so-called inner model. Arrows between different blocks of variables $\boldsymbol{X}_k$ indicate in which way they are linked (see Figure 3.1). For each inner model, we can define an undirected link

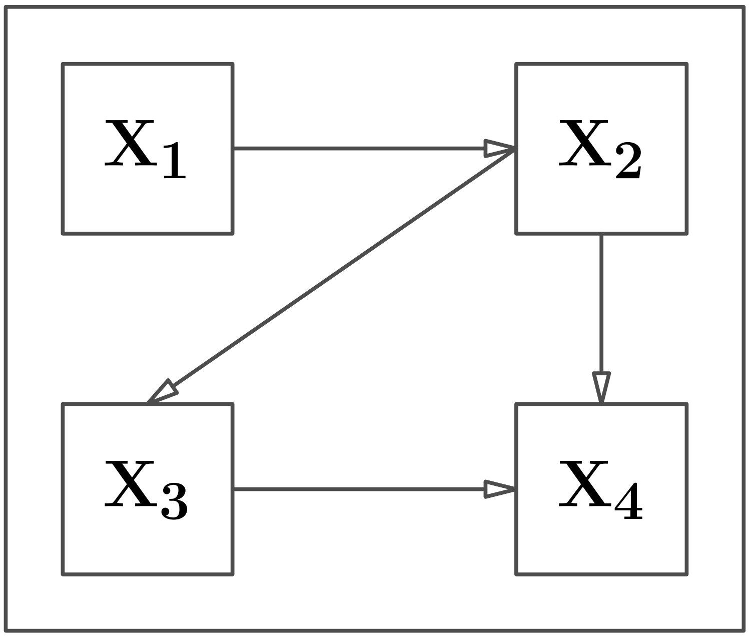

Figure 3.1: Illustration of a PLS path model with $K = 4$ blocks.

matrix $\boldsymbol{C} \in \{0,1\}^{K \times K}$ via

$$c_{kl} = \begin{cases} 1 & , \quad \boldsymbol{X}_l \to \boldsymbol{X}_k \text{ or } \boldsymbol{X}_l \to \boldsymbol{X}_k \\ 0 & , \quad \text{otherwise} \end{cases}$$

(and $c_{kk} = 0$). Furthermore, we assume that for each block $\boldsymbol{X}_k$, there is a single latent (or hidden) variable $\boldsymbol{z}_k \in \mathbb{R}^n$ that represents this block. This is called the outer model and is illustrated in Figure 3.2. We distinguish two types of

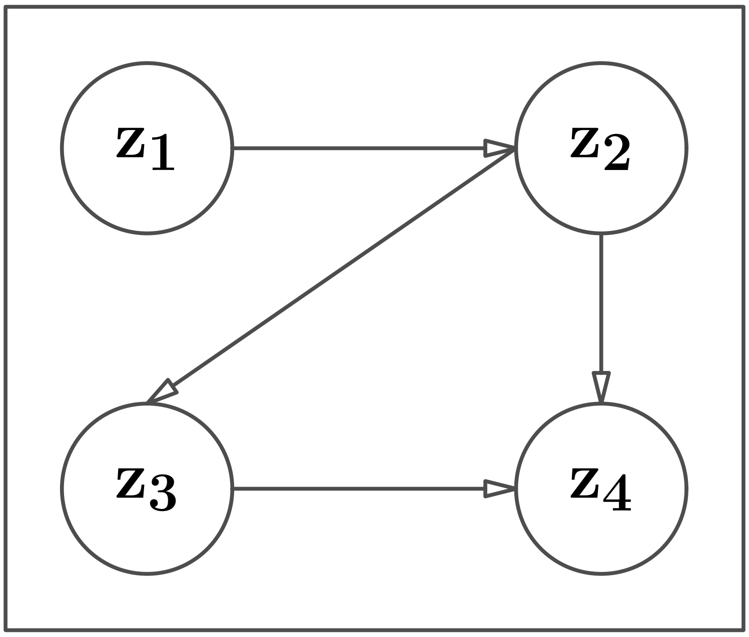

Figure 3.2: Illustration of the outer PLS model. Each block of manifest variables is replaced by one latent variable.

relationships between latent and manifest variables. The first one is the formative model, the second one is the reflective model (see Figure 3.3). In the formative model, we assume that the block $\boldsymbol{X}_k$ of manifest variables forms the latent variable $\boldsymbol{z}_k$. In terms of a regression model, this can be expressed as

$$\boldsymbol{z}_k = \boldsymbol{X}_k \boldsymbol{\beta} + \boldsymbol{\varepsilon} . \tag{3.1}$$

In the reflective model, we assume that the manifest variables are a reflection of the latent variable. The underlying regression model is

$$\boldsymbol{X}_k = \boldsymbol{z}_k \boldsymbol{\beta}^t + \boldsymbol{E} . \tag{3.2}$$

Given data, we want to estimate (1) the latent variables, (2) the relationship in the inner model, and (3) the relations in the outer model. In order to estimate $\boldsymbol{z}_k$, we need to define sensible optimality criteria. Ideally, these criteria have three features: Firstly, we want to find estimates $\boldsymbol{z}_k$ such that $\boldsymbol{z}_k$ and $\boldsymbol{z}_l$ are “close” if their corresponding blocks are linked. Secondly, we want to take into account

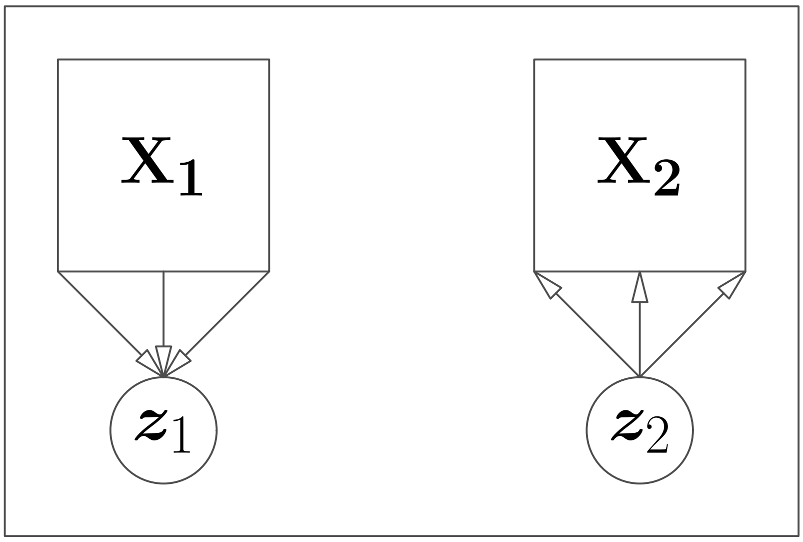

Figure 3.3: The difference between the two outer models. Left: The formative model. The manifest variables $\boldsymbol{X}_1$ form the latent variable $\boldsymbol{z}_1$. Right: The reflective model. The manifest variables $\boldsymbol{X}_2$ are a reflection of the latent variable $\boldsymbol{z}_2$.

the directions of the arrows in the inner model. Thirdly, we want to take into account the directions of the arrows in the outer model.

The tough part in the process is the estimation of the latent variables. Once they are estimated, the relationships that are indicated by an arrow can be derived by using a least-squares estimate.

In the literature on PLS, there is often a huge gap between the abstract model (in terms of the inner and the outer model) and that what is actually computed by the PLS path algorithms. Normally, the PLS algorithms are presented directly in connection with the PLS framework, insinuating that the algorithms produce optimal solutions of an obvious estimation problem attached to PLS. This estimation problem is however never defined. Furthermore, the directions of the arrows in the outer model are hoped to be taken care of by employing different "modes": mode A for reflective blocks and mode B for formative blocks. While the terms "reflective blocks" and "formative blocks" refer to the description of the outer model as illustrated in Figure 3.3, mode A and B correspond to algorithms. It is a-priori not clear how the abstract models and the PLS path models are connected. In order to understand the mathematical theory behind all the formulas, it is indispensable to set up a general optimization strategy before deriving algorithms that try to solve them. For this reason, in Section 3.2, we start the investigation by presenting different optimization criteria in order to define the latent variables $\boldsymbol{z}_k$. Afterwards, we present two algorithms in Section 3.3 that try to compute the optimal solution. Only in Section 3.4, we introduce the two PLS algorithms – Lohmöller in mode B and Wold in mode B– and show that they are equal to the algorithms in Section 3.3. We repeat that we always have to keep in mind the difference between what PLS **wants** to model and that what it effectively models.

We now try to give a general overview on different types of PLS path algorithms. All terms that are now given will be defined in subsequent sections. In the PLS literature, there are two generic algorithms, the Lohmöller procedure and the Wold procedure. Roughly, there are the following measures of closeness between latent variables: Horst, factorial and centroid. These measures are usually called schemes. They have in common that they do not (!) consider the directions of the arrows in the inner model. A variant of PLS that does fulfill this condition is the "path weighting scheme" (which is not considered in this work). We recall that the directions in the outer model are hoped to be taken care of by employing different "modes": mode A for reflective blocks and mode B for formative blocks.

Let us conclude this section with some additional definitions. In order to simplify notation, all variables are assumed to have zero mean. The empirical covariance matrix between blocks of variables is denoted by

$$\boldsymbol{S}_{kl} = \frac{1}{n}\boldsymbol{X}_k^t \boldsymbol{X}_l \in \mathbb{R}^{p_k \times p_l}\,. \tag{3.3}$$

We frequently work with vectors and matrices that have a block structure that is induced by $(p_1, \ldots, p_K)$. The covariance matrix $\boldsymbol{S} \in \mathbb{R}^{p\times p}$ of $\boldsymbol{X}$ can be partitioned into blocks of the form

$$\boldsymbol{S} = \begin{pmatrix} \boldsymbol{S}_{11} & \boldsymbol{S}_{12} & \ldots & \boldsymbol{S}_{1K} \\ \boldsymbol{S}_{21} & \boldsymbol{S}_{22} & \ldots & \boldsymbol{S}_{2K} \\ \vdots & & & \vdots \\ \boldsymbol{S}_{K1} & \ldots & \ldots & \boldsymbol{S}_{KK} \end{pmatrix} \in \mathbb{R}^{p\times p}$$

with $\boldsymbol{S}_{kl}$ defined in (3.3). We denote this block-wise structure by [ ]:

$$\boldsymbol{S} \quad = \quad [\boldsymbol{S}_{kl}] \in \mathbb{R}^{p\times p} \,.$$

Furthermore, we need matrices that only have entries in their diagonal blocks. We denote them by

$$\boldsymbol{S}_D = \operatorname{diag}\left[\boldsymbol{S}_{11}, \ldots, \boldsymbol{S}_{KK}\right] = \begin{pmatrix} \boldsymbol{S}_{11} & 0 & \ldots & 0 \\ 0 & \boldsymbol{S}_{22} & \ldots & 0 \\ \vdots & & & \vdots \\ 0 & \ldots & 0 & \boldsymbol{S}_{KK} \end{pmatrix} \in \mathbb{R}^{p\times p}$$

The subsript D indicates that we only take the diagonal blocks of the matrix $\boldsymbol{S}$. Any vector $\boldsymbol{w} \in \mathbb{R}^p$ can be partitioned into

$$\boldsymbol{w}^t = \left(\boldsymbol{w}_1^t, \ldots, \boldsymbol{w}_K^t\right)^t \quad , \quad \boldsymbol{w}_k \in \mathbb{R}^{p_k} \,. \tag{3.4}$$

3.2 Optimization Strategies

In this section, we abandon the PLS framework. Instead, we present different optimization criteria in order to define latent variables $\boldsymbol{z}_k$. The general idea is to extend and generalize Canonical Correlation Analysis (CCA) (Hotelling 1936) to more than two blocks of variables. We try to find latent vectors $\boldsymbol{z}_k = \boldsymbol{X}_k\boldsymbol{w}_k$ such that $\boldsymbol{z}_k$ and $\boldsymbol{z}_l$ are maximally correlated if the corresponding blocks are linked. In this sense, it is an extension of CCA. For two blocks of variables $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, CCA computes

$$\arg\max_{\boldsymbol{w}_1,\boldsymbol{w}_2} \quad \operatorname{cor}\left(\boldsymbol{X}_1\boldsymbol{w}_1, \boldsymbol{X}_2\boldsymbol{w}_2\right) \,.$$

We can scale the weights $\boldsymbol{w}_1, \boldsymbol{w}_2$ without changing the optimization problem, and we obtain the equivalent optimization problem

$$\begin{aligned} \arg\max_{\boldsymbol{w}_1,\boldsymbol{w}_2} \quad & \operatorname{cov}\left(\boldsymbol{X}_1\boldsymbol{w}_1, \boldsymbol{X}_2\boldsymbol{w}_2\right) \,, \\ \text{subject to} \quad & \frac{1}{n}\|\boldsymbol{X}_i\boldsymbol{w}_i\|^2 = 1 \,,\; i = 1,2 \,. \end{aligned}$$

This leads to the following general definition.

Optimization Problem 3.1. For K blocks of variables, we define the following, general optimization problem:

$$\begin{aligned} \arg\max_{\boldsymbol{w}} & \quad \sum_{k,l:c_{kl}\neq 0} g\left(\operatorname{cov}(\boldsymbol{X}_k\boldsymbol{w}_k, \boldsymbol{X}_l\boldsymbol{w}_l)\right) , \\ \text{subject to} & \quad \frac{1}{n}\|\boldsymbol{X}_k\boldsymbol{w}_k\|^2 = 1 . \end{aligned}$$

Here, g is one of three functions

$$g(x) = \begin{cases} x & , \text{Horst} \\ x^2 & , \text{factorial} \\ |x| & , \text{centroid} \end{cases} .$$

The terms “Horst”, “factorial” and “centroid” are called schemes in the PLS literature. We call the first scheme the Horst scheme as it is equivalent to a generalization of CCA to more than two blocks that is described in Horst (1961) and Horst (1965). The terms “factorial” and “centroid” stem from the PLS literature. We remark that the Horst scheme is not used in the PLS community, although it has been suggested as an alternative to the two other schemes by Hanafi & Qannari (2005).

Let us define the real-valued function

$$f_g(\boldsymbol{w}) = \sum_{k,l:c_{kl}\neq 0} g\left(\operatorname{cov}(\boldsymbol{X}_k\boldsymbol{w}_k, \boldsymbol{X}_l\boldsymbol{w}_l)\right) = \sum_{k,l=1}^{K} c_{kl}\, g\left(\boldsymbol{w}_k^t \boldsymbol{S}_{kl}\boldsymbol{w}_l\right) ,$$

with $\boldsymbol{w}$ defined in (3.4). The Lagrangian function associated to problem 3.1 is

$$L(\boldsymbol{w}, \boldsymbol{\lambda}) = f_g(\boldsymbol{w}) - \frac{1}{2}\sum_{k=1}^{K} \lambda_k \left(\boldsymbol{w}_k^t \boldsymbol{S}_{kk}\boldsymbol{w}_k - 1\right) \tag{3.5}$$

with $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_K) \in \mathbb{R}^K$ the Lagrangian multipliers. The factor $-1/2$ is added in order to avoid a rescaling of the multipliers λ_i. We set

$$\theta_{kl} = \theta_{kl}(\boldsymbol{w}) = \operatorname{cov}\left(\boldsymbol{X}_k\boldsymbol{w}_k, \boldsymbol{X}_l\boldsymbol{w}_l\right) = \boldsymbol{w}_k^t \boldsymbol{S}_{kl}\boldsymbol{w}_l . \tag{3.6}$$

Differentiating the Lagrangian function (3.5), we yield

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{w}_k} &= \sum_{l=1}^{K} c_{kl} g'\left(\theta_{kl}\right) \boldsymbol{S}_{kl} \boldsymbol{w}_l - \lambda_k \boldsymbol{S}_{kk} \boldsymbol{w}_k \,, \qquad (3.7) \\ \frac{\partial L}{\partial \lambda_k} &= \boldsymbol{w}_k^t \boldsymbol{S}_{kk} \boldsymbol{w}_k - 1 \,. \end{aligned}$$

We set

$$\boldsymbol{S}_g(\boldsymbol{w}) = \left[c_{kl} g'\left(\theta_{kl}\right) \boldsymbol{S}_{kl}\right] , \qquad (3.8)$$

which implies that

$$\frac{\partial f_g}{\partial \boldsymbol{w}}(\boldsymbol{w}) = \boldsymbol{S}_g(\boldsymbol{w}) \boldsymbol{w} \,.$$

These equations can be represented in a compact form.

Proposition 3.2. *The Lagrangian equations (A.6) and (A.7) of the optimization problem 3.1 are*

$$\begin{aligned} \boldsymbol{S}_g(\boldsymbol{w}) \boldsymbol{w} &= \boldsymbol{\Lambda} \boldsymbol{S}_D \boldsymbol{w} \,, \\ \boldsymbol{w}_k^t \boldsymbol{S}_{kk} \boldsymbol{w}_k &= 1 \,. \end{aligned}$$

The matrix $\boldsymbol{\Lambda}$ is a diagonal matrix that is of the form

$$\boldsymbol{\Lambda} = \operatorname{diag}\left[\lambda_1 \boldsymbol{I}_{p_1}, \ldots, \lambda_K \boldsymbol{I}_{p_K}\right] \in \mathbb{R}^{p \times p} \,.$$

We might think of $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_K)$ as a vector of multivariate eigenvalues. Note that in the Horst scheme, the matrix $\boldsymbol{S}_g(\boldsymbol{w})$ does not depend on $\boldsymbol{w}$ and in this case, the problem is called a multivariate eigenvalue problem (Chu & Watterson 1993).

Remark 3.3. Of course, in the centroid scheme, the function $f_g(\boldsymbol{w})$ is not differentiable on $\mathbb{R}^p$. We therefore have to restrict f_g onto the open subset

$$\mathcal{M} = \{\boldsymbol{w} \in \mathbb{R} | \boldsymbol{w}_k^t \boldsymbol{S}_{kl} \boldsymbol{w}_l \neq 0\} \,.$$

Note that we can decompose $\mathcal{M}$ into finitely many disjoint open subsets

$$\mathcal{M}_{\mathcal{I}} = \{\boldsymbol{w} \in \mathbb{R} | \operatorname{sign}\left(\boldsymbol{w}_k^t \boldsymbol{S}_{kl} \boldsymbol{w}_l\right) = \mathcal{I}_{kl}\} \,. \qquad (3.9)$$

Here $\mathcal{I} \in \{\pm 1\}^{K \times K}$ is a symmetric matrix with diagonal elements equal to 1. Whenever we speak of a derivative of f_g, we implicitly assume that f_g is restricted to one of these subsets. Note that on any of the subsets $\mathcal{M}_{\mathcal{I}}$, the matrix $\boldsymbol{S}_g(\boldsymbol{w})$ that is defined in (3.8) does not depend on $\boldsymbol{w}$.

Any solution of the equations in proposition 3.2 is – by definition – a stationary point of the optimization problem 3.1. In general, there might be more than one stationary point.

Lemma 3.4. *The stationary point* $\boldsymbol{w}$ *that is a solution of 3.1 is the one such that the sum of the corresponding multivariate eigenvalues is maximal.*

Proof. We first note that

$$g'(x)x = g(x) \begin{cases} 1, & \text{Horst, centroid} \\ 2, & \text{factorial} \end{cases} = \widetilde{c} g(x)\,.$$

It follows that for all $\boldsymbol{w}$ that are stationary points,

$$\begin{aligned} \sum_{k,l} c_{kl} g\left(\boldsymbol{w}_k \boldsymbol{S}_{kl} \boldsymbol{w}_l\right) &= \widetilde{c} \sum_{k,l} c_{kl} g'\left(\boldsymbol{w}_k \boldsymbol{S}_{kl} \boldsymbol{w}_l\right) \boldsymbol{w}_k \boldsymbol{S}_{kl} \boldsymbol{w}_l \\ &= \widetilde{c} \boldsymbol{w}^t \boldsymbol{S}_g(\boldsymbol{w}) \boldsymbol{w} \\ &= \widetilde{c} \sum_k \lambda_k \boldsymbol{w}_k^t \boldsymbol{S}_{kk} \boldsymbol{w}_k \\ &= \widetilde{c} \sum_k \lambda_k\,. \end{aligned}$$

□

If we want to maximize the covariance between latent variables instead of correlation, we have to change the constraints in 3.1. We obtain

$$\begin{aligned} \arg\max_{\boldsymbol{w}} \quad & \sum_{k,l:c_{kl}\neq 0} g\left(\operatorname{cov}(\boldsymbol{X}_k \boldsymbol{w}_k, \boldsymbol{X}_l \boldsymbol{w}_l)\,,\right) \\ \text{subject to} \quad & \frac{1}{n}\|\boldsymbol{w}_k\|^2 = 1\,, \end{aligned} \tag{3.10}$$

and the corresponding Lagrangian equations are

$$\begin{aligned} \boldsymbol{S}_g(\boldsymbol{w})\boldsymbol{w} &= \boldsymbol{\Lambda}\boldsymbol{w}\,, \\ \frac{1}{n}\boldsymbol{w}_k^t \boldsymbol{w}_k &= 1\,. \end{aligned}$$

We can always transform optimization problem 3.1 into (3.10): Denote by $\sqrt{\boldsymbol{S}_{kk}}$ the root of the positive-semidefinite matrix $\boldsymbol{S}_{kk}$. We set

$$\begin{aligned}
\widetilde{\boldsymbol{w}}_k &= \sqrt{\boldsymbol{S}_{kk}}\boldsymbol{w}_k\,, && (3.11)\\
\sqrt{\boldsymbol{S}_D} &= \operatorname{diag}\left[\sqrt{\boldsymbol{S}_{11}},\ldots,\sqrt{\boldsymbol{S}_{KK}}\right]\,, && (3.12)\\
\widetilde{\boldsymbol{S}} &= \left(\sqrt{\boldsymbol{S}_D}\right)^{-}\boldsymbol{S}\left(\sqrt{\boldsymbol{S}_D}\right)^{-}\,. && (3.13)
\end{aligned}$$

It follows readily from the singular value decomposition of $\boldsymbol{X}_k$ and $\boldsymbol{X}_l$ that

$$\boldsymbol{S}_{kl} = \sqrt{\boldsymbol{S}_{kk}}\left(\sqrt{\boldsymbol{S}_{kk}}\right)^{-}\boldsymbol{S}_{kl}\left(\sqrt{\boldsymbol{S}_{ll}}\right)^{-}\sqrt{\boldsymbol{S}_{ll}}\,.$$

We conclude that optimization problem 3.1 is equivalent to

$$\begin{aligned}
&\arg\max_{\widetilde{\boldsymbol{w}}} && \sum_{k,l:c_{kl}\neq 0} g\left(\widetilde{\boldsymbol{w}}_k^t\widetilde{\boldsymbol{S}}_{kl}\widetilde{\boldsymbol{w}}_l\right)\,,\\
&\text{subject to} && \frac{1}{n}\|\widetilde{\boldsymbol{w}}_k\| = 1\,.
\end{aligned}$$

3.3 Multivariate Power Methods

If we want to find the optimal solution of the optimization problem 3.1, we can proceed stepwise. First, we compute a solution of the associated Lagrangian equations. As this is a stationary point, we then have to check if this is the optimal one. One possibility to solve eigenproblems as in proposition 3.2 is to apply a multivariate version of the power method defined in algorithm A.9.

Algorithm 3.5 (Multivariate Power Method)**.** *After the initialization of weight vectors $\boldsymbol{w} = (\boldsymbol{w}_1,\ldots,\boldsymbol{w}_K)$ such that $\boldsymbol{w}_k^t\boldsymbol{S}_{kk}\boldsymbol{w}_k = 1$, we iteratively compute*

$$\begin{aligned}
\widetilde{\boldsymbol{w}}^{(i+1)} &= \boldsymbol{S}_D^{-}\boldsymbol{S}_g\left(\boldsymbol{w}^{(i)}\right)\boldsymbol{w}^{(i)} && \textit{iteration}\\
\boldsymbol{w}_k^{(i+1)} &= \widetilde{\boldsymbol{w}}_k^{(i+1)}/\sqrt{\widetilde{\boldsymbol{w}}_k^{(i+1)}\boldsymbol{S}_{kk}\widetilde{\boldsymbol{w}}_k^{(i+1)}}\,, k = 1\ldots,K && \textit{normalization}
\end{aligned}$$

If the multivariate power method converges to a vector $\boldsymbol{w}$, this is obviously a solution of the Lagrangian equations 3.2. Note that in algorithm 3.5, all weight vectors $\boldsymbol{w}_k$ are updated simultaneously. There is a variation of the power method, where in each round, only one weight vector is updated. We call this algorithm a multivariate Gauss-Seidel algorithm. In order to have a compact representation, let us define the quantities θ_{kl} in a more general way. For two vectors $\boldsymbol{w}$ and $\boldsymbol{v}$, we define

$$\overline{\theta}_{kl}(\boldsymbol{w}, \boldsymbol{v}) \quad = \quad \boldsymbol{w}_k^t \boldsymbol{S}_{kl} \boldsymbol{v}_l \,.$$

We set

$$\overline{\boldsymbol{S}}_g(\boldsymbol{w}, \boldsymbol{v}) \quad = \quad \left[c_{kl} g'\left(\overline{\theta}_{kl}(\boldsymbol{w}, \boldsymbol{v})\right) \boldsymbol{S}_{kl}\right] \,.$$

Note that

$$\theta_{kl}(\boldsymbol{w}) \quad = \quad \overline{\theta}_{kl}(\boldsymbol{w}, \boldsymbol{w}) \quad \text{and} \quad \boldsymbol{S}_g(\boldsymbol{w}) \quad = \quad \overline{\boldsymbol{S}}_g(\boldsymbol{w}, \boldsymbol{w}) \,.$$

Now, let us decompose

$$\overline{\boldsymbol{S}}_g(\boldsymbol{w}, \boldsymbol{v}) \quad = \quad \boldsymbol{U}_g^t(\boldsymbol{w}, \boldsymbol{v}) + \boldsymbol{U}_g(\boldsymbol{w}, \boldsymbol{v})$$

with $\boldsymbol{U}_g$ the strictly upper triangular part of $\overline{\boldsymbol{S}}_g$. (Recall that the block diagonal of $\overline{\boldsymbol{S}}_g(\boldsymbol{w}, \boldsymbol{v})$ is zero, as $c_{kk} = 0$.)

Algorithm 3.6 (Multivariate Gauss-Seidel Algorithm)**.** *After the initialization of weight vectors $\boldsymbol{w} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K)$ such that $\boldsymbol{w}_k^t \boldsymbol{S}_{kk} \boldsymbol{w}_k = 1$, we iteratively compute for $k = 1, \ldots, K$*

$$\begin{aligned} \widetilde{\boldsymbol{w}}^{(i+1)} &= \boldsymbol{S}_D^{-}\left(\boldsymbol{U}_g^t\left(\boldsymbol{w}^{(i)}, \boldsymbol{w}^{(i+1)}\right) \boldsymbol{w}^{(i+1)} + \boldsymbol{U}_g\left(\boldsymbol{w}^{(i)}, \boldsymbol{w}^{(i)}\right) \boldsymbol{w}^{(i)}\right) \\ \boldsymbol{w}_k(i+1) &= \widetilde{\boldsymbol{w}}_k^{(i)} / \sqrt{\widetilde{\boldsymbol{w}}_k^{(i)} \boldsymbol{S}_{kk} \widetilde{\boldsymbol{w}}_k^{(i)}} \end{aligned}$$

For the Horst scheme, the Gauss-Seidel algorithm is already defined in Chu & Watterson (1993).

Proposition 3.7. *If the multivariate Gauss-Seidel algorithm converges to a vector $\boldsymbol{w}$, this is a stationary point of 3.1.*

Proof. If $\boldsymbol{w}$ is the solution of the multivariate Gauss-Seidel algorithm, we conclude that $\widetilde{\boldsymbol{w}} = \boldsymbol{\Lambda w}$ with $\widetilde{\boldsymbol{w}}$ defined in algorithm 3.6. We plug this into the formula for $\widetilde{\boldsymbol{w}}$ and obtain

$$\boldsymbol{S}_D \boldsymbol{\Lambda} \widetilde{\boldsymbol{w}} = \boldsymbol{U}_g^t\left(\boldsymbol{w}, \boldsymbol{w}\right) \boldsymbol{w} + \boldsymbol{U}_g\left(\boldsymbol{w}, \boldsymbol{w}\right) \boldsymbol{w} = \boldsymbol{S}_g\left(\boldsymbol{w}\right) \boldsymbol{w} \,.$$

These are the the Lagrangian equations 3.2. □

3.4 The Partial Least Squares Path Algorithms

Now, we return to the PLS framework introduced in Section 3.1. There are two important algorithms that try to compute the latent variables in the path model - Lohmöller (Lohmöller 1989) and Wold (Wold 1985). The algorithms presented here are in mode B. Mode B refers to a path model where all blocks are supposed to be formative. PLS in mode A (which corresponds to reflective blocks) is discussed in Section 3.5. The common point of view is that these algorithms are alternating algorithms in the sense that we iteratively estimate first the inner model and then the outer model. We show that Lohmöller corresponds to the multivariate power method 3.5 and that Wold corresponds to the multivariate Gauss-Seidel algorithm 3.6. As a consequence we can conclude that - if all blocks are formative - the PLS solutions are indeed stationary points of the optimization criterion 3.1.

Let us start with Lohmöller's algorithm.

Algorithm 3.8 (Lohmöller's algorithm)**.** *After the initialization of weight vector* $\boldsymbol{w}^{(0)}$ *such that* $\boldsymbol{z}_k^{(0)} = \boldsymbol{X}_k \boldsymbol{w}_k^{(0)}$ *has length* $\sqrt{n}$ *we iteratively compute for all* k *simultaneously*

$$\begin{array}{rcll}
\widetilde{\boldsymbol{z}}_k^{(i+1)} & = & \sum_{l=1}^{K} c_{kl} g'\left(\theta_{kl}(\boldsymbol{w}^{(i)})\right) \boldsymbol{z}_l^{(i)} & \textit{inner model (environmental variable)} \\
\widetilde{\boldsymbol{w}}_k^{(i+1)} & = & \left(\boldsymbol{X}_k^t \boldsymbol{X}_k\right)^{-} \boldsymbol{X}_k^t \widetilde{\boldsymbol{z}}_k^{(i+1)} & \textit{outer model in mode B} \\
\boldsymbol{w}_k^{(i+1)} & = & \sqrt{n}\widetilde{\boldsymbol{w}}_k^{(i+1)} / \left\|\boldsymbol{X}_k \widetilde{\boldsymbol{w}}_k^{(i+1)}\right\| & \textit{normalization} \\
\boldsymbol{z}_k^{(i+1)} & = & \boldsymbol{X}_k \boldsymbol{w}_k^{(i+1)} & \textit{update}
\end{array}$$

We remark that the term "environmental variable" is part of the PLS nomenclatura.

Proposition 3.9. *The Lohmöller algorithm is equal to the multivariate power method.*

Proof. The proof is straightforward. Note that the formula for the environmental variable equals

$$\widetilde{\boldsymbol{z}}_k^{(i)} = \sum_{l=1}^{K} c_{kl} g'\left(\theta_{kl}(\boldsymbol{w}^{(i)})\right) \boldsymbol{z}_l^{(i)} = \sum_{l=1}^{K} c_{kl} g'\left(\theta_{kl}(\boldsymbol{w}^{(i)})\right) \boldsymbol{X}_l \boldsymbol{w}_l^{(i)}.$$

It follows that

$$\begin{aligned}
\widetilde{\boldsymbol{w}}_k^{(i+1)} &= \left(\boldsymbol{X}_k^t\boldsymbol{X}_k\right)^- \boldsymbol{X}_k^t\widetilde{z}_k^{(i)} \\
&= \frac{1}{n}\left(\boldsymbol{S}_{kk}\right)^{-1}\left(\sum_{l=1}^{K} c_{kl}g'\left(\theta_{kl}(\boldsymbol{w}^{(i)})\right)\boldsymbol{X}_k^t\boldsymbol{X}_l\boldsymbol{w}_l^{(i)}\right) \\
&= \frac{1}{n}\left(\boldsymbol{S}_{kk}\right)^{-1}\left(\sum_{l=1}^{K} c_{kl}g'\left(\theta_{kl}(\boldsymbol{w}^{(i)})\right)\boldsymbol{S}_{kl}\boldsymbol{w}_l^{(i)}\right),
\end{aligned}$$

which equals the formula in algorithm 3.5 – up to a scaling factor. □

Now, we consider Wold's algorithm. In this algorithm , only one block is updated in each inner loop.

Algorithm 3.10 (Wold's algorithm). *After the initialization of weight vector* $\boldsymbol{w}^{(0)}$ *such that* $\boldsymbol{z}_k^{(0)} = \boldsymbol{X}_k\boldsymbol{w}_k^{(0)}$ *has length* $\sqrt{n}$ *we iteratively compute for* $k = 1, \ldots, K$

$$\begin{array}{rcll}
\widetilde{\boldsymbol{z}}_k^{(i+1)} &=& \sum_{l=1}^{k-1} c_{kl}g'\left(\boldsymbol{w}_k^{(i)}\boldsymbol{S}_{kl}\boldsymbol{w}_l^{(i+1)})\right)\boldsymbol{z}_l^{(i+1)} & \textit{inner model (environmental} \\
&& +\sum_{l=k+1}^{K} c_{kl}g'\left(\boldsymbol{w}_k^{(i)}\boldsymbol{S}_{kl}\boldsymbol{w}_l^{(i)}\right)\boldsymbol{z}_l^{(i)} & \textit{variable)} \\
\widetilde{\boldsymbol{w}}_k^{(i+1)} &=& \left(\boldsymbol{X}_k^t\boldsymbol{X}_k\right)^- \boldsymbol{X}_k^t\widetilde{z}_k^{(i+1)} & \textit{outer model in mode B} \\
\boldsymbol{w}_k^{(i+1)} &=& \sqrt{n}\widetilde{\boldsymbol{w}}_k^{(i+1)}/\left\|\boldsymbol{X}_k\widetilde{\boldsymbol{w}}_k^{(i+1)}\right\| & \textit{normalization} \\
\boldsymbol{z}_k^{(i+1)} &=& \boldsymbol{X}_k\boldsymbol{w}_k^{(i+1)} & \textit{update}
\end{array}$$

It follows readily that the Wold algorithm is the same as the multivariate Gauss-Seidel algorithm.

Proposition 3.11. *The Wold algorithm is equal to the multivariate Gauss-Seidel algorithm.*

Proof. The proof is analogous to the proof of proposition 3.9. It follows from the definition of the environmental variable that

$$\begin{aligned}
n\left(\boldsymbol{S}_{kk}\right)\widetilde{\boldsymbol{w}}_k^{(i+1)} &= \boldsymbol{X}_k^t\widetilde{z}_k^{(i)} \\
&= \sum_{l=1}^{k-1} c_{kl}g'\left(\overline{\theta}_{kl}(\boldsymbol{w}^{(i)}, \boldsymbol{w}^{(i+1)})\right)\boldsymbol{S}_{kl}\boldsymbol{w}_l^{(i+1)} \\
&\quad + \sum_{l=k+1}^{K} c_{kl}g'\left(\overline{\theta}_{kl}(\boldsymbol{w}^{(i)}, \boldsymbol{w}^{(i+1)})\right)\boldsymbol{S}_{kl}\boldsymbol{w}_l^{(i)},
\end{aligned}$$

and this equals (**??**). □

This leads to the following conclusion.

Proposition 3.12. *Suppose that we use mode B and one of the schemes "Horst", "factorial" or "centroid". If Wold's or Lohmöller's algorithm converge, their respective solutions are stationary points of optimization problem 3.1.*

It has already been shown in Mathes (1993) that the solution of the PLS path algorithms are solutions of the Lagrangian equations 3.2.

We end this section with a geometric interpretation of the PLS path model algorithms. In order to keep the example simple, we only consider the Horst scheme and assume that all blocks are linked. The Lagrangian equations expressed in terms of the latent variables $\boldsymbol{z}_k$ are

$$\lambda_k \boldsymbol{z}_k = \mathcal{P}_{\boldsymbol{X}_k}\left(\sum_{l\neq k} \boldsymbol{z}_l\right) .$$

Here, λ_k is the normalization term. In other words, for every stationary point of the optimization problem, the latent variable $\boldsymbol{z}_k$ equals – up to its length – the projection of the sum of the other latent variables onto the space spanned by the columns of $\boldsymbol{X}_k$. This is the generalization of the geometric interpretation of CCA described in Kockelkorn (2000).

Finally, let us briefly mention that we can apply the kernel trick to the PLS path model framework. To do so, let us start with the remark that we can represent any vector $\boldsymbol{w}_k \in \mathbb{R}^{p_k}$ by

$$\boldsymbol{w}_k = \boldsymbol{X}_k^t \boldsymbol{\alpha}_k + \widetilde{\boldsymbol{w}}_k \quad , \quad \widetilde{\boldsymbol{w}}_k \perp \operatorname{span}(\boldsymbol{V}_k) .$$

Here,

$$\boldsymbol{X}_k = \boldsymbol{U}_k \Sigma_k \boldsymbol{V}_k^t$$

is the singular value decomposition of block $\boldsymbol{X}_k$. It follows that

$$\boldsymbol{z}_k = \boldsymbol{X}_k \boldsymbol{w}_k = \boldsymbol{X}_k \left(\boldsymbol{X}_k^t \boldsymbol{\alpha}_k + \widetilde{\boldsymbol{w}}_k\right) = \boldsymbol{X}_k \boldsymbol{X}_k^t \boldsymbol{\alpha}_k + \boldsymbol{X}_k \widetilde{\boldsymbol{w}}_k = \boldsymbol{X}_k \boldsymbol{X}_k^t \boldsymbol{\alpha}_k .$$

As the optimization problem 3.1 only depends on $\boldsymbol{w}_k$ via $\boldsymbol{z}_k$, we conclude that we can assume that

$$\boldsymbol{w}_k = \boldsymbol{X}_k^t \boldsymbol{\alpha}_k .$$

Plugging this into the PLS path algorithms, we obtain a dual representation in terms of the dual variables $\boldsymbol{\alpha}_k$. A combination of generalized CCA with the kernel trick is studied in Yamanishi et al. (2003). In this work, the Lagrangian equations 3.2 for the Horst scheme are expressed in terms of the Kernel matrices $\boldsymbol{X}_k \boldsymbol{X}_k^t$ – with the (erroneous?) additional constraint that all multivariate eigenvalues λ_i are equal.

3.5 No Smooth Optimality Criterion for Mode A

We now describe why mode B is supposedly related to formative blocks of variables and present a heuristic how to adapt the algorithms for reflective blocks. In the estimation of the outer model in the PLS algorithms, the unstandardized weight vectors

$$\widetilde{\boldsymbol{w}}_k^{(i+1)} = \left(\boldsymbol{X}_k^t \boldsymbol{X}_k\right)^- \boldsymbol{X}_k^t \widetilde{\boldsymbol{z}}_k^{(i+1)}$$

can be interpreted as the OLS regression coefficients of the linear regression model (3.1) (with the latent variable replaced by the environmental variable). For this reason, it is argued that the algorithms in mode B (3.8 and 3.10 respectively) refer to formative blocks. The heuristic is now as follows. For any reflective block of variables, we first estimate the coefficients $\boldsymbol{\beta}$ of the linear regression model (3.2) that refers to reflective blocks. If we use OLS, we obtain

$$\widehat{\boldsymbol{\beta}} = \frac{1}{\widetilde{\boldsymbol{z}}_k^t \widetilde{\boldsymbol{z}}_k} \boldsymbol{X}_k^t \widetilde{\boldsymbol{z}}_k .$$

This implies that

$$\widehat{\boldsymbol{X}_k} = \widetilde{\boldsymbol{z}}_k \widehat{\boldsymbol{\beta}}^t . \tag{3.14}$$

Given the manifest variables $\boldsymbol{X}_k$, we now have to estimate the latent variable $\boldsymbol{z}_k$ based on (3.14). That is, we have to find the solution $\boldsymbol{z}_k$ of the over-determined equations

$$\boldsymbol{X}_k = \boldsymbol{z}_k \widehat{\boldsymbol{\beta}}^t .$$

We obtain the estimate

$$\boldsymbol{z}_k = \frac{1}{\widehat{\boldsymbol{\beta}}^t \widehat{\boldsymbol{\beta}}} \boldsymbol{X}_k \widehat{\boldsymbol{\beta}} .$$

Note that the scale of $\boldsymbol{z}_k$ is not important, as all latent variables are normalized. To summarize, for all blocks k that are reflective, we replace the estimation of $\widetilde{\boldsymbol{w}}_k$ in the outer model by

$$\widetilde{\boldsymbol{w}}_k^{(i+1)} = \boldsymbol{X}_k^t \widetilde{\boldsymbol{z}}_k^{(i+1)} . \tag{3.15}$$

This estimation mode is called mode A. We do not discuss the validity of this heuristics.

Let us define

$$\widetilde{\boldsymbol{S}}_D = \operatorname{diag}\left[\widetilde{\boldsymbol{D}}_i\right] \quad , \quad \widetilde{\boldsymbol{D}}_i = \begin{cases} \boldsymbol{I}_{p_k} & , k \text{ of mode } A \\ \boldsymbol{S}_{kk} & , k \text{ of mode } B \end{cases} .$$

Proposition 3.13. *If we replace the estimation of the weights in the outer model in algorithms 3.8 and 3.10 by (3.15) for all reflective blocks, any solution of the Lohmöller and the Wold algorithm fulfills*

$$\boldsymbol{S}_g(\boldsymbol{w})\boldsymbol{w} = \boldsymbol{\Lambda}\widetilde{\boldsymbol{S}}_D \boldsymbol{w} , \tag{3.16}$$

$$\boldsymbol{w}_k^t \boldsymbol{S}_{kk} \boldsymbol{w}_k = 1 . \tag{3.17}$$

Proof. The proof follows immediately from the definition of the algorithms. We only prove the statement for the Lohmöller algorithm. The result for the Wold algorithm follows in the same way. Analogously to the proof of proposition 3.9, we have

$$\widetilde{\boldsymbol{z}}_k^{(i)} = \sum_{l=1}^{K} c_{kl} g' \left(\theta_{kl}(\boldsymbol{w}^{(i)})\right) \boldsymbol{X}_l \boldsymbol{w}_l^{(i)} .$$

It follows that

$$\begin{aligned}
\widetilde{\boldsymbol{w}}_k^{(i+1)} &= \left(\widetilde{\boldsymbol{D}}_k\right)^- \boldsymbol{X}_k^t \widetilde{z}_k^{(i)} \\
&= \left(\widetilde{\boldsymbol{D}}_k\right)^- \left(\sum_{l=1}^K c_{kl} g'\left(\theta_{kl}(\boldsymbol{w}^{(i)})\right) \boldsymbol{X}_k^t \boldsymbol{X}_l \boldsymbol{w}_l^{(i)}\right) \\
&= \left(\widetilde{\boldsymbol{D}}_k\right)^- \left(\sum_{l=1}^K c_{kl} g'\left(\theta_{kl}(\boldsymbol{w}^{(i)})\right) \; \boldsymbol{S}_{kl} \boldsymbol{w}_l^{(i)}\right) .
\end{aligned}$$

Replacing $\boldsymbol{w}_k^{(i)}$ and $\widetilde{\boldsymbol{w}}_k^{(i)}$ by their respective limits $\boldsymbol{w}_k$ and $\widetilde{\boldsymbol{w}}_k$, the proof is complete. □

Equations (3.16) and (3.17) do not yet have the form of Lagrangian equations (A.6) and (A.7), as the constraints (3.17) do not fit to the right hand side of (3.16). In order to obtain a valid form, we have to multiply each side of (3.16) with

$$\overline{\boldsymbol{S}}_D = \operatorname{diag}\left[\overline{\boldsymbol{D}}_i\right] \quad , \quad \overline{\boldsymbol{D}}_i = \begin{cases} \boldsymbol{S}_{kk} & , k \text{ of mode } A \\ \boldsymbol{I}_{p_k} & , k \text{ of mode } B \end{cases} .$$

This yields

$$\overline{\boldsymbol{S}}_D \left(\boldsymbol{S}_g(\boldsymbol{w})\right) \boldsymbol{w} = \Lambda \boldsymbol{S}_D \boldsymbol{w} \, , \tag{3.18}$$

$$\boldsymbol{w}_k^t \boldsymbol{S}_{kk} \boldsymbol{w}_k = 1 \, . \tag{3.19}$$

These cannot be the Lagrangian equations of any twice differentiable optimization problem.

Theorem 3.14. *Suppose that at least one block is of mode A. For almost all data sets $\boldsymbol{X}$, the equations (3.18) and (3.19) are not the Lagrangian equations of an optimization problem (A.3) and (A.4), where the objective function f in (A.3) is twice-differentiable.*

The term "for almost all data sets" refers to the fact that the set of matrices $\boldsymbol{X}$ for which the above statement does not hold, has measure zero.

Proof. Suppose that (3.18) and (3.19) are the Lagrangian equations of an optimization problem as in (A.3) and (A.4). This implies that $\overline{\boldsymbol{S}}_D \left(\boldsymbol{S}_g(\boldsymbol{w})\right) \boldsymbol{w}$ is the first derivative of the objective function (A.3). It follows that the Hessian matrix of this objective function is

$$\frac{\partial}{\partial \boldsymbol{w}} \overline{\boldsymbol{S}}_D \left(\boldsymbol{S}_g(\boldsymbol{w})\right) \boldsymbol{w} \;=\; \overline{\boldsymbol{S}}_D \frac{\partial}{\partial \boldsymbol{w}} \left(\boldsymbol{S}_g(\boldsymbol{w})\boldsymbol{w}\right) = \overline{\boldsymbol{S}}_D \boldsymbol{H}_g(\boldsymbol{w}) \tag{3.20}$$

The matrix $\boldsymbol{H}_g(\boldsymbol{w})$ is the Hessian matrix of f_g. It follows from the Theorem of Schwartz A.4 that (3.20) is a symmetric matrix. We have

$$\left(\overline{\boldsymbol{S}}_D \boldsymbol{H}_g(\boldsymbol{w})\right)^t = \left(\boldsymbol{H}_g(\boldsymbol{w})\right)^t \left(\overline{\boldsymbol{S}}_D\right)^t = \boldsymbol{H}_g(\boldsymbol{w}) \overline{\boldsymbol{S}}_D$$

and consequently, the matrix $\boldsymbol{H}_g(\boldsymbol{w}) \overline{\boldsymbol{S}}_D$ is symmetric if and only if the two matrices commute,

$$\boldsymbol{H}_g(\boldsymbol{w}) \overline{\boldsymbol{S}}_D = \overline{\boldsymbol{S}}_D \boldsymbol{H}_g(\boldsymbol{w}) \,. \tag{3.21}$$

This is in however in general not the case. To see this, we have to compute the Hessian matrix of f_g for the respective schemes. If we consider the Horst scheme, we have

$$\frac{\partial^2 f_g}{\partial^2 \boldsymbol{w}} = \frac{\partial (\boldsymbol{S}_g(\boldsymbol{w})\boldsymbol{w})}{\partial \boldsymbol{w}} = \boldsymbol{S}_g(\boldsymbol{w}) \,,$$

as the matrix $\boldsymbol{S}_g(\boldsymbol{w})$ does not depend on $\boldsymbol{w}$. For the centroid scheme, we restrict the function on one of the open subsets $\mathcal{M}_{\mathcal{I}}$ defined in (3.9). As the matrix $\boldsymbol{S}_g(\boldsymbol{w})$ does not depend on $\boldsymbol{w}$ on this subset, we have

$$\frac{\partial^2 f_{g_{|\mathcal{M}_{\mathcal{I}}}}}{\partial^2 \boldsymbol{w}} = \left[c_{kl} \mathcal{I}_{kl} \boldsymbol{S}_{kl}\right] \,.$$

It follows that for the Horst scheme or the centroid scheme, condition (3.21) is equivalent to the following equations. If the blocks k and l are linked, we have

$$\begin{aligned} \boldsymbol{S}_{kl}\boldsymbol{S}_{ll} &= \boldsymbol{S}_{kk}\boldsymbol{S}_{kl} \quad , \quad \text{k and l of mode A}\,, \\ \boldsymbol{S}_{kl}\boldsymbol{S}_{ll} &= \boldsymbol{S}_{kl} \qquad\quad , \quad \text{k of mode A, l of mode B}\,. \end{aligned}$$

These conditions are in general not fulfilled. For the factorial scheme, the equations become more complicated, but is still possible to show that condition (3.21) is not fulfilled. Recall (3.8) from which follows that the first derivative of f_g is a block vector with the kth entry equal to

$$\sum_{l=1}^{K} c_{kl} \left(\boldsymbol{w}_k^t \boldsymbol{S}_{kl} \boldsymbol{w}_l\right) \boldsymbol{w}_l \,.$$

If we differentiate this expression with respect to $\boldsymbol{w}_l$, we obtain

$$\boldsymbol{H}_g(\boldsymbol{w}) \quad = \quad \boldsymbol{S}_g(\boldsymbol{w}) + \left[c_{kl}\boldsymbol{S}_{kl}\boldsymbol{w}_l\boldsymbol{w}_k^t\boldsymbol{S}_{kl}\right] \,.$$

We conclude that if the blocks k and l are linked, condition (3.21) is equivalent to

$$\begin{aligned}
\left(\boldsymbol{S}_{kl} + \boldsymbol{S}_{kl}\boldsymbol{w}_l\boldsymbol{w}_k^t\boldsymbol{S}_{kl}\right)\boldsymbol{S}_{ll} &= \boldsymbol{S}_{kk}\left(\boldsymbol{S}_{kl} + \boldsymbol{S}_{kl}\boldsymbol{w}_l\boldsymbol{w}_k^t\boldsymbol{S}_{kl}\right) \quad , \quad \text{k and l of mode A}\,,\\
\left(\boldsymbol{S}_{kl} + \boldsymbol{S}_{kl}\boldsymbol{w}_l\boldsymbol{w}_k^t\boldsymbol{S}_{kl}\right)\boldsymbol{S}_{ll} &= \left(\boldsymbol{S}_{kl} + \boldsymbol{S}_{kl}\boldsymbol{w}_l\boldsymbol{w}_k^t\boldsymbol{S}_{kl}\right) \qquad , \quad \text{k of mode A, l of mode B}\,.
\end{aligned}$$

These equations are in general not fulfilled. □

As a consequence, we advocate to be cautious to use mode A. Suppose that the algorithm is applied to different start vectors and that the resulting weight vectors are different. There is no way to decide which one of them is better, as we do not know of any optimality criterion attached to mode A. Note that the above described scenario is not hypothetical. We show in Section 3.6 that the algorithms in mode B do not necessarily converge to the solution of 3.1.

Note furthermore that we can easily modify the PLS algorithms in mode A such that their solutions are stationary points of sensible optimization problems. Let us assume that all blocks are of mode A. We replace the normalization step of the weight vectors by

$$\boldsymbol{w}_k^{(i+1)} \quad = \quad \frac{1}{n\|\widetilde{\boldsymbol{w}}_k^{(i+1)}\|}\widetilde{\boldsymbol{w}}_k^{(i+1)} \,.$$

It is straightforward to show that any solution of the PLS path algorithms fulfills the Lagrangian equations associated to (3.10). In other words, by modifying the normalization step in mode A, we obtain a stationary point of the optimization problem attached to maximizing covariances instead of correlations.

3.6 No Convergence to the Optimum for Mode B

The numerical convergence of the algorithms is only proven for the Horst scheme (Chu & Watterson 1993). Recently, Hanafi (2006) showed that the Wold algorithm in mode B converges monotonically. That is, for the sequence of weight vectors $\boldsymbol{w}^{(i)}$ computed by this algorithm, the real-valued sequence $f_g\left(\boldsymbol{w}^{(i)}\right)$ is monotonically increasing and bounded. Even if the algorithms converge, it is not guaranteed that the obtained vector $\boldsymbol{w}$ is the solution of the optimization problem 3.1. Chu & Watterson (1993) present a counter example for the Horst scheme if we apply the multivariate power algorithm. It is shown that the solution of the multivariate power algorithm depends on the starting value and that it is likely that the algorithm converges to a local solution. Hanafi (2006) present a counter example for the centroid scheme. In this section, we present a counter example for both the factorial and the centroid scheme and for the both PLS algorithms – Lohmöller and Wold. For the obvious reason, we only consider mode B.

We present two examples. Let us start with the remark that the second counter example is not chosen because it reflects any real world situation, but in order to make the results reproducible. In fact, counter examples can be found easily, and the convergence to local optima seems to be the "generic" case (at least for the centroid scheme) if the manifest variables are not highly correlated. This is shown in the first example. In both examples, we use $K = 3$ blocks of variables. Each block consists of $p_k = 4$ variables. This implies that the matrix $\boldsymbol{X} = (\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3)$ consists of $4 \times 3 = 12$ columns.We assume that all blocks are connected.

We consider both PLS algorithms – Wold and Lohmöller –in the factorial scheme and the centroid scheme respectively. This yields four different variants. We run these four different algorithms 500 times. In each iteration, the standardized starting vectors $\boldsymbol{w}_k^{(0)}$ are drawn randomly.

Fairly realistic, but not exactly reproducible example

The number of examples is $n = 50$. Each row of $\boldsymbol{X}$ is a sample of a multivariate normal distribution with zero mean and the covariance matrix equal to the identity matrix. In order to save computational time, we first transform the data as described in (3.11),(3.12) and (3.13). In this example, the two algorithms in the factorial scheme always converge to the set of multivariate eigenvalues

$$\lambda_1 = 0.19735 \quad \lambda_2 = 0.0741 \quad \lambda_3 = 0.27133 \quad \sum \lambda_i = 0.54278\,.$$

As this is the only solution out of the 500 experiments, this indicates that it is the global optimum.

Scheme	Algorithm	Global Optimum	Local Solution
factorial	Wold	100 %	0 %
	Lohmöller	100 %	0%
centroid	Wold	46.8 %	53.2%
	Lohmöller	54 %	46 %

Table 3.1: Results for the first example

For the centroid scheme, we first remark that – given the same start vector – the Lohmöller and the Wold algorithm can produce different results. In this example, different results are observed in 16.4 % of the cases. For the centroid scheme, the algorithms converged to one of the two sets of multivariate eigenvalues

$$\begin{array}{lllll} \text{Solution 1} & \lambda_1 = 0.59238 & \lambda_2 = 0.50684 & \lambda_3 = 0.60448 & \sum \lambda_i = 1.70370\,. \\ \text{Solution 2} & \lambda_1 = 0.53533 & \lambda_2 = 0.42023 & \lambda_3 = 0.66051 & \sum \lambda_i = 1.61607\,. \end{array}$$

The second one is only a local solution. As we only observe these two solutions, we conjecture that the first one is the global optimum.

For each algorithm and each scheme, we count the number of experiments in which the algorithms converged to the respective solutions. Table 3.1 illustrates that both algorithms have a substantial chance to converge to the local optimum.

Unrealistic, but reproducible example

The number of examples is $n = 12$. We define the 12×12 matrix $\boldsymbol{X} = (\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3)$ in the following way:

$$\boldsymbol{X}_{i,j} = \begin{cases} 1 & , i = j, j = i + 1 \\ 0 & , \text{otherwise} \end{cases} .$$

We center the columns of the matrix $\boldsymbol{X}$. In this example, we observe a convergence to local solutions in both schemes.

Exactly as in the first example, the Lohmöller and the Wold algorithms sometimes produce different results. This happens in 17% of the cases for the factorial scheme and in 23% of the cases for the centroid scheme. Secondly, the result depends on

Scheme	Algorithm	Global Optimum	Local Optimum
factorial	Wold	87.4 %	12.6 %
	Lohmöller	80.4 %	19.6 %
centroid	Wold	42.8 %	57.2 %
	Lohmöller	57.4 %	42.6 %

Table 3.2: Results for the second example

the starting vector. For the factorial scheme, the algorithm converges to one of the two sets of multivariate eigenvalues

Solution 1	$\lambda_1 = 0.74718$	$\lambda_2 = 0.99703$	$\lambda_3 = 0.39944$	$\sum \lambda_i = 2.14365\,,$
Solution 2	$\lambda_1 = 0.47979$	$\lambda_2 = 0.48329$	$\lambda_3 = 0.53733$	$\sum \lambda_i = 1.50041\,.$

The latter solution is only a local solution and we conjecture that the first solution is the global optimum. The result is similar for the centroid scheme. The algorithms converged to one of the two sets of multivariate eigenvalues.

Solution 1	$\lambda_1 = 1.13148$	$\lambda_2 = 1.31360$	$\lambda_3 = 0.97503$	$\sum \lambda_i = 3.42011\,,$
Solution 2	$\lambda_1 = 1.00000$	$\lambda_2 = 1.00000$	$\lambda_3 = 1.00000$	$\sum \lambda_i = 3.00000\,.$

Again, for each algorithm and each scheme, we count the number of experiments in which the algorithms converge to the respective solutions. Table 3.2 illustrates that both algorithms have a substantial chance to converge to the local optimum.

3.7 Conclusion

In the PLS literature, there has been a lot of obscurity regarding the mathematical background of the path modeling algorithms. In this chapter, we have hopefully shed some light into this subject. To summarize, we showed two results. Firstly, we proved that the PLS path algorithms in mode A produce algebraic equations that are not linked to any sufficiently smooth optimization problem. This marks a severe setback in the search of a justification of mode A in terms of optimality criteria. What conclusions can be drawn from this result? Note that in principle, it might still be possible to derive optimization problems attached to mode A that are not twice differentiable. But we strongly advise a different modus operandi. Instead of first defining algebraic equations and then searching for associated optimization problems, we should rather first set up a sensible optimization problem and then search for algorithms that solve it. We pointed out that if we maximize covariances instead to correlations, we yield equations quite similar to the ones of mode A. Secondly, for those PLS algorithms that do produce critical points of optimization problems, we showed that the algorithms do not necessarily converge to the maximum. As a consequence, every algorithm should be run several times in order to detect possible other solutions.

Chapter 4

Partial Least Squares for Regression

If there are more variables than examples, the usual linear regression tools such as ordinary least squares (OLS) regression cannot be applied since the $p \times p$ matrix $\boldsymbol{X}^t\boldsymbol{X}$ is singular. From a technical point of view, we can solve this problem by replacing the inverse of $\boldsymbol{X}^t\boldsymbol{X}$ by a generalized inverse as described in Section 1.3. However, if $p > n$, OLS fits the training data perfectly and we cannot expect this method to perform well on a new data set. Partial Least Squares Regression (PLSR) (Wold 1975, Wold et al. 1984) is an alternative regression tool which is especially appropriate in the case of highly correlated predictors and high-dimensional data. PLSR is a standard tool for analyzing chemical data (Martens & Naes 1989), and in recent years, the success of PLSR has lead to applications in other scientific fields such as physiology (Rosipal et al. 2003) or bioinformatics (Boulesteix & Strimmer 2006), to name but a few.

PLSR can handle multivariate responses. We now give a general introduction and then focus on univariate responses. Quite generally, PLSR tries to model linear relationships between two blocks of variables $\boldsymbol{X}$ and $\boldsymbol{Y}$ in terms of latent variables. In this sense, it fits into the PLS path model framework that is investigated in Chapter 3. Note however that in contrast to the two PLS path model algorithms (Lohmöller and Wold), the relationship between $\boldsymbol{X}$ and $\boldsymbol{Y}$ is not symmetric and that we usually consider more than one latent variable. Although extensions of PLS path models to more than one latent variable per block are possible, it is rather confusing to describe PLSR as a special case of PLS path models. We therefore give a self-contained introduction.

4.1 NIPALS and SIMPLS

There are quite a few versions of PLSR. Mainly, they differ in the way in which the latent variables are extracted from the data. It is not our aim to explain all variants, and we will focus on two versions. For a general overview on different forms of PLSR see Rosipal & Krämer (2006).

The main idea is to build a few orthogonal components $\boldsymbol{t}_1, \ldots, \boldsymbol{t}_m$ from the original predictors $\boldsymbol{X}$. A component is a linear combination of the original predictors that hopefully reflects the relevant structure of the data. We use these latent components as regressors in a least squares regression in place of $\boldsymbol{X}$. PLSR is similar to Principal Components Regression (PCR). The difference is that PCR extracts components that explain the variance in the predictor variables whereas PLSR extracts components that have a large covariance with $\boldsymbol{Y}$.

We now formalize this idea. A latent component $\boldsymbol{t}$ is a linear combination $\boldsymbol{t} = \boldsymbol{X}\boldsymbol{w}$ of the predictor variables. The vector $\boldsymbol{w}$ is usually called the weight vector. We want to find a component with maximal covariance to $\boldsymbol{Y}$, that is we want to maximize

$$\|\mathrm{cov}\,(\boldsymbol{X}\boldsymbol{w}, \boldsymbol{Y})\|^2 \;=\; \boldsymbol{w}^t\boldsymbol{X}^t\boldsymbol{Y}\boldsymbol{Y}^t\boldsymbol{X}\boldsymbol{w}\,.$$

We have to constrain $\boldsymbol{w}$ in order to obtain identifiability, choosing

$$\arg\max \quad \boldsymbol{w}^t\boldsymbol{X}^t\boldsymbol{Y}\boldsymbol{Y}^t\boldsymbol{X}\boldsymbol{w}\,, \tag{4.1}$$

$$\text{subject to} \quad \|\boldsymbol{w}\| = 1\,. \tag{4.2}$$

Let us remark that (4.1) and (4.2) are equivalent to

$$\max \quad \frac{\boldsymbol{w}^t\boldsymbol{X}^t\boldsymbol{Y}\boldsymbol{Y}^t\boldsymbol{X}\boldsymbol{w}}{\boldsymbol{w}^t\boldsymbol{w}}\,. \tag{4.3}$$

The solution of (4.3) is only unique up to a scalar. The normalization of the weight vectors $\boldsymbol{w}$ to length 1 is not essential for the PLSR algorithm (expect for computational considerations as e.g. numerical stability) and PLSR algorithms differ in the way they scale the weight vectors and components. In this paper, we do not scale the vectors, in order to keep the notation as simple as possible.

We conclude that the solution $\boldsymbol{w}_1$ is the eigenvector of the matrix

$$\boldsymbol{B} = \boldsymbol{X}^t\boldsymbol{Y}\boldsymbol{Y}^t\boldsymbol{X} \tag{4.4}$$

that corresponds to the largest eigenvalue of $\boldsymbol{B}$. This eigenvector is usually computed in an iterative way, e.g. by using the power algorithm that is defined in A.9.

Subsequent components $\boldsymbol{t}_2, \boldsymbol{t}_3, \ldots$ are chosen such that they maximize the squared covariance to $\boldsymbol{Y}$ and that all components are mutually orthogonal. In PLSR, there are different techniques to extract subsequent components, and we now present two of them. On the one hand, for the ith component, we compute

$$\begin{aligned} \max \quad & \boldsymbol{w}^t\boldsymbol{X}^t\boldsymbol{Y}\boldsymbol{Y}^t\boldsymbol{X}\boldsymbol{w}\,, & (4.5)\\ \text{subject to} \quad & \|\boldsymbol{w}\| = 1\,, & (4.6)\\ & \boldsymbol{X}\boldsymbol{w} \perp \boldsymbol{t}_j, j < i\,. & (4.7) \end{aligned}$$

This task is called SIMPLS (de Jong 1993). On the other hand, we can deflate the original predictor variables $\boldsymbol{X}$. That is, we only consider the part of $\boldsymbol{X}$ that is orthogonal on all components $\boldsymbol{t}_j, j < i$:

$$\boldsymbol{X}_i = \boldsymbol{X} - \mathcal{P}_{\boldsymbol{t}_1,\ldots,\boldsymbol{t}_{i-1}}\boldsymbol{X}\,.$$

We then replace $\boldsymbol{X}$ by $\boldsymbol{X}_i$ in (4.1). This task is called the NIPALS method and is described in Wold (1975). The two methods are equivalent if $\boldsymbol{Y}$ is univariate in the sense that we end up with the same components $\boldsymbol{t}_i$. We present a proof in corollary 6.4.

To summarize, the PLSR algorithm is of the following form.

Algorithm 4.1 (NIPALS). *After setting $\boldsymbol{X}_1 = \boldsymbol{X}$, the weight vectors $\boldsymbol{w}_i$ and the components $\boldsymbol{t}_i$ of PLSR are determined by iteratively computing*

$$\begin{aligned} \boldsymbol{w}_i &= \textit{dominant eigenvector of } \boldsymbol{X}_i^t\boldsymbol{Y}\boldsymbol{Y}^t\boldsymbol{X}_i^t && \textit{weight vector}\\ \boldsymbol{t}_i &= \boldsymbol{X}_i\boldsymbol{w}_i && \textit{component}\\ \boldsymbol{X}_{i+1} &= \boldsymbol{X}_i - \mathcal{P}_{\boldsymbol{t}_i}\boldsymbol{X}_i && \textit{deflation} \end{aligned}$$

PLSR used to be overlooked by statisticians and was considered an algorithm rather than a sound statistical model. This attitude is to some extent understandable, as in the early literature on the subject, PLSR was explained solely in terms of formulas as in algorithm 4.1. Due to its success in applications, the interest in the statistical properties of PLSR has risen. It can be related to other biased regression techniques such as Principal Components Regression and Ridge Regression and these methods can be cast under a unifying framework (Stone & Brooks 1990). The shrinkage properties of PLSR have been studied extensively (Frank & Friedman 1993, de Jong 1995, Goutis 1996, Butler & Denham 2000) and are also discussed in Chapter 7. Furthermore, it can be shown that PLSR is closely connected to Krylov subspaces and the conjugate gradient method (Helland 1988, Phatak & de Hoog 2002). In Chapter 6, we establish a similar connection between penalized PLSR (introduced in Chapter 5) and a preconditioned conjugate gradient method.

Finally, let us remark that in many articles on PLSR, both $\boldsymbol{X}$ and $\boldsymbol{Y}$ are deflated with respect to $\boldsymbol{t}_i$:

$$\boldsymbol{X}_{i+1} = \boldsymbol{X} - \mathcal{P}_{\boldsymbol{t}_1,\ldots,t_i}\boldsymbol{X} \quad , \quad \boldsymbol{Y}_{i+1} = \boldsymbol{Y} - \mathcal{P}_{\boldsymbol{t}_1,\ldots,t_i}\boldsymbol{Y} \, .$$

If we are only interested in the latent components $\boldsymbol{t}_i$, this is however not necessary. To see this, we use the fact (see proposition A.10) that

$$(\boldsymbol{I}_n - \mathcal{P}_{\boldsymbol{t}_1,\ldots,\boldsymbol{t}_i}) = (\boldsymbol{I}_n - \mathcal{P}_{\boldsymbol{t}_1,\ldots,\boldsymbol{t}_i})(\boldsymbol{I}_n - \mathcal{P}_{\boldsymbol{t}_1,\ldots,\boldsymbol{t}_i}) \, .$$

We conclude that

$$\boldsymbol{X}_{i+1}^t\boldsymbol{Y}_{i+1} = \boldsymbol{X}^t(\boldsymbol{I}_n - \mathcal{P}_{\boldsymbol{t}_1,\ldots,\boldsymbol{t}_i})(\boldsymbol{I}_n - \mathcal{P}_{\boldsymbol{t}_1,\ldots,\boldsymbol{t}_i})\boldsymbol{Y} = \boldsymbol{X}^t(\boldsymbol{I}_n - \mathcal{P}_{\boldsymbol{t}_1,\ldots,\boldsymbol{t}_i})\boldsymbol{Y} = \boldsymbol{X}_{i+1}^t\boldsymbol{Y} \, .$$

If the response $\boldsymbol{Y} = \boldsymbol{y}$ is univariate, PLSR is sometimes refered to as PLS1. In this case, the computation of the weight vectors is very easy. Let us define the p-dimensional vector

$$\boldsymbol{b} = \boldsymbol{X}^t\boldsymbol{y} \, .$$

It follows that the first eigenvector of the matrix $\boldsymbol{B} = \boldsymbol{b}\boldsymbol{b}^t$ (defined in (4.4)) equals $\boldsymbol{b}$. To summarize, the univariate PLS algorithm is of the following form.

Algorithm 4.2 (Univariate NIPALS)**.** *After setting $\boldsymbol{X}_1 = \boldsymbol{X}$, the weight vectors $\boldsymbol{w}_i$ and the components $\boldsymbol{t}_i$ of univariate PLSR are determined by iteratively* *computing*

$$\begin{aligned}\boldsymbol{w}_i &= \boldsymbol{X}_i^t \boldsymbol{y} && \textit{weight vector}\\ \boldsymbol{t}_i &= \boldsymbol{X}_i \boldsymbol{w}_i && \textit{component}\\ \boldsymbol{X}_{i+1} &= \boldsymbol{X}_i - \mathcal{P}_{\boldsymbol{t}_i}\boldsymbol{X}_i && \textit{deflation}\end{aligned}$$

In the next section, we list some properties of PLSR and show how to derive an estimate $\widehat{\boldsymbol{\beta}}$ from the latent components.

4.2 Basic Properties of Partial Least Squares Regression

We set

$$\boldsymbol{T} = (\boldsymbol{t}_1, \ldots, \boldsymbol{t}_m) .$$

As already mentioned, the original predictors $\boldsymbol{X}$ are replaced by $\boldsymbol{T}$ and the response is then regressed onto the columns of $\boldsymbol{T}$. The fitted response is

$$\widehat{\boldsymbol{y}} = \mathcal{P}_{\boldsymbol{T}}\boldsymbol{y} = \boldsymbol{T}(\boldsymbol{T}^t\boldsymbol{T})^{-1}\boldsymbol{T}^t\boldsymbol{y} . \qquad (4.8)$$

In order to predict the response for new observations, we have to determine the vector of regression coefficients,

$$\widehat{\boldsymbol{y}} = \boldsymbol{X}\widehat{\boldsymbol{\beta}} .$$

Therefore, a representation of the components $\boldsymbol{t}_i = \boldsymbol{X}_i\boldsymbol{w}_i$ as a linear combination of the original predictors $\boldsymbol{X}$ is needed. In other words, we have to derive weight vectors $\widetilde{\boldsymbol{w}}_i$ with

$$\boldsymbol{X}\widetilde{\boldsymbol{w}}_i = \boldsymbol{X}_i\boldsymbol{w}_i .$$

They are in general different from the "pseudo" weight vectors $\boldsymbol{w}_i$ that are computed by the NIPALS algorithm. Before stating this result, it is beneficial to cast the PLSR method in a broader framework.

PLSR is an iterative process. In each step, we compute weight vectors $\boldsymbol{w}_i$ and then deflate $\boldsymbol{X}$ with respect to the latent components $\boldsymbol{t}_i = \boldsymbol{X}_i\boldsymbol{w}_i$. For any algorithm that produces a weight vector $\boldsymbol{w}$ (probably depending on the data $\boldsymbol{X}$ and $\boldsymbol{y}$) , we can define a generic latent component regression algorithm in the following way.

Algorithm 4.3 (Generic Latent Component Regression). *After setting $\boldsymbol{X}_1 = \boldsymbol{X}$, the weight vectors $\boldsymbol{w}_i$ and the components $\boldsymbol{t}_i$ are determined by iteratively computing*

$$\begin{array}{rcll} \boldsymbol{w}_i & & & \textit{weight vector} \\ \boldsymbol{t}_i & = & \boldsymbol{X}_i \boldsymbol{w}_i & \textit{component} \\ \boldsymbol{X}_{i+1} & = & \boldsymbol{X}_i - \mathcal{P}_{\boldsymbol{t}_i} \boldsymbol{X}_i & \textit{deflation} \end{array}$$

The response $\boldsymbol{y}$ is then regressed onto the latent components $\boldsymbol{t}_1, \ldots, \boldsymbol{t}_m$ as in (4.8).

What are the merits of this generic approach? Let us emphasize that we do **not** want to introduce yet another unifying framework that comprises all kinds of regression methods. The reason is that in fact, a lot of properties of PLSR become much clearer if we consider the general framework. For example, most of the properties of PLSR do not depend on the particular method for computing the weight vectors, but on the fact that the latent components are mutually orthogonal. Papers on PLSR tend to be rather technical and proofs are sometimes hard to follow. We now present some alternative proofs that only rely on algorithm 4.3 and exploit some basic properties of projections. As a nice side-effect, we can apply these results (with no extra effort) to other multivariate regression tools as Continuum Regression (Stone & Brooks 1990) or the penalized PLSR approach introduced in Chapter 5.

Set

$$\boldsymbol{T} = \boldsymbol{T}^{(m)} = (\boldsymbol{t}_1, \ldots, \boldsymbol{t}_m) \quad \text{and} \quad \boldsymbol{W} = \boldsymbol{W}^{(m)} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_m) \,.$$

These are the matrices of components and weight vectors respectively that are defined in algorithm 4.3. We fix m and omit the superscript for the sake of readability. We recall that for $k < i$

$$\boldsymbol{X}_i = \prod_{j=k}^{i-1} \left(\boldsymbol{I}_n - \mathcal{P}_{\boldsymbol{t}_j}\right) \boldsymbol{X}_k = \left(\boldsymbol{I}_n - \mathcal{P}_{\boldsymbol{t}_k, \ldots, \boldsymbol{t}_{i-1}}\right) \boldsymbol{X}_k \,. \tag{4.9}$$

The last equality follows from the fact that the components $\boldsymbol{t}_i$ are mutually orthogonal. In particular

$$\boldsymbol{X}_i = \left(\boldsymbol{I}_n - \mathcal{P}_{\boldsymbol{t}_1, \ldots, \boldsymbol{t}_{i-1}}\right) \boldsymbol{X} \,. \tag{4.10}$$

If not stated otherwise, the following results hold for the Generic Latent Components (GLC) Regression approach introduced in algorithm 4.3.

Lemma 4.4. *We have*

$$\boldsymbol{X}_i \boldsymbol{w}_j \;=\; \boldsymbol{0}$$

for $j < i$.

Proof. The condition $j < i$ implies (recall (4.9))

$$\boldsymbol{X}_i \;=\; \boldsymbol{X}_j - \mathcal{P}_{\boldsymbol{t}_j,\ldots,\boldsymbol{t}_{i-1}} \boldsymbol{X}_j \,,$$

and consequently

$$\begin{aligned} \boldsymbol{X}_i \boldsymbol{w}_j &= \boldsymbol{X}_j \boldsymbol{w}_j - \mathcal{P}_{\boldsymbol{t}_1,\ldots,\boldsymbol{t}_{i-1}} \boldsymbol{X}_j \boldsymbol{w}_j \\ &= \boldsymbol{t}_j - \mathcal{P}_{\boldsymbol{t}_1,\ldots,\boldsymbol{t}_{i-1}} \boldsymbol{t}_j \\ &\overset{j \leq i-1}{=} \boldsymbol{t}_j - \boldsymbol{t}_j = \boldsymbol{0}\,. \end{aligned}$$

□

Corollary 4.5. *The weight vectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_m$ defined by univariate NIPALS are mutually orthogonal.*

Proof. It follows from the definition of the PLSR weight vectors that for $i > j$

$$\langle \boldsymbol{w}_i, \boldsymbol{w}_j \rangle = \left\langle \boldsymbol{X}_i^t \boldsymbol{y}, \boldsymbol{w}_j \right\rangle = \boldsymbol{y}^t \boldsymbol{X}_i \boldsymbol{w}_j = \boldsymbol{y}^t \boldsymbol{0} = 0\,.$$

□

We now return to the generic latent components approach and set

$$\boldsymbol{R} \;=\; \boldsymbol{T}^t \boldsymbol{X}\ \boldsymbol{W} \in \mathbb{R}^{m \times m}\,.$$

Proposition 4.6. *The matrix $\boldsymbol{R}$ is upper triangular, that is*

$$r_{ij} \;=\; \boldsymbol{t}_i^t \boldsymbol{X} \boldsymbol{w}_j = 0\,, \tag{4.11}$$

if $i < j$. The matrix $\boldsymbol{R}$ is invertible. Furthermore, we have

$$\boldsymbol{X}\boldsymbol{W} \;=\; \boldsymbol{T}\,diag\left(\frac{1}{\boldsymbol{t}_1^t \boldsymbol{t}_1}, \ldots, \frac{1}{\boldsymbol{t}_m^t \boldsymbol{t}_m}\right) \boldsymbol{R}\,. \tag{4.12}$$

In particualar, the columns of $\boldsymbol{T}$ and the columns of $\boldsymbol{XW}$ span the same space.

Proof. First note that (4.10) is equivalent to

$$\boldsymbol{X} = \boldsymbol{X}_j + \mathcal{P}_{\boldsymbol{t}_1,\ldots,\boldsymbol{t}_{j-1}}\boldsymbol{X}\,.$$

It follows that

$$\begin{aligned}\boldsymbol{X}\boldsymbol{w}_j &= \left(\boldsymbol{X}_j + \mathcal{P}_{\boldsymbol{t}_1,\ldots,\boldsymbol{t}_{j-1}}\boldsymbol{X}\right)\boldsymbol{w}_j \\ &= \boldsymbol{X}_j\boldsymbol{w}_j + \mathcal{P}_{\boldsymbol{t}_j,\ldots,\boldsymbol{t}_{j-1}}\boldsymbol{X}\ \boldsymbol{w}_j \\ &= \boldsymbol{t}_j + \sum_{i=1}^{j-1}\frac{\boldsymbol{t}_i^t\boldsymbol{X}\boldsymbol{w}_j}{\boldsymbol{t}_i^t\boldsymbol{t}_i}t_i\,. \end{aligned} \tag{4.13}$$

As all components $\boldsymbol{t}_i$ are mutually orthogonal, we conclude that

$$\boldsymbol{t}_i^t\boldsymbol{X}\boldsymbol{w}_j = \begin{cases}\boldsymbol{t}_i^t\boldsymbol{t}_i \neq 0 & ,i=j \\ 0 & ,i>j \\ \boldsymbol{t}_i^t\boldsymbol{X}\boldsymbol{w}_j & \text{,i<j}\end{cases}.$$

We conclude that $\boldsymbol{R}$ is an upper triangular matrix with all diagonal elements $\neq 0$. Next, note that (4.13) is equivalent to

$$\boldsymbol{X}\boldsymbol{w}_j = \frac{r_{jj}}{\boldsymbol{t}_j^t\boldsymbol{t}_j}\boldsymbol{t}_j + \sum_{i=1}^{j-1}\frac{r_{ij}}{\boldsymbol{t}_i^t\boldsymbol{t}_i}\boldsymbol{t}_i\,.$$

This equals (4.12). □

We can now determine the regression coefficients for the generic latent regression approach.

Proposition 4.7. *The regression vector $\widehat{\boldsymbol{\beta}}$ of the generic latent components (GLC) approach defined in algorithm 4.3 is*

$$\widehat{\boldsymbol{\beta}} = \boldsymbol{W}\left(\boldsymbol{W}^t\boldsymbol{X}^t\boldsymbol{X}\boldsymbol{W}\right)^{-}\boldsymbol{W}^t\boldsymbol{X}^t\boldsymbol{y}\,. \tag{4.14}$$

In particular, the generic latent component (GLC) estimator is the solution of *the constrained minimization problem*

$$\begin{aligned} \min_{\boldsymbol{\beta}} \quad & \|\boldsymbol{y} - \boldsymbol{X\beta}\| \\ \textit{subject to} \quad & \boldsymbol{\beta} \in \textit{span}\left\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_m\right\}\,. \end{aligned} \tag{4.15}$$

Proof. We deduce from (4.12) that the columns of $\boldsymbol{XW}$ span the same space as the columns of $\boldsymbol{T}$. As GLC is simply ordinary least squares regression with predictors $\boldsymbol{t}_1, \ldots, \boldsymbol{t}_m$, we have

$$\widehat{\boldsymbol{y}} = \mathcal{P}_{\boldsymbol{T}}\boldsymbol{y} = \mathcal{P}_{\boldsymbol{XW}}\boldsymbol{y} = \boldsymbol{XW}\left(\boldsymbol{W}^t\boldsymbol{X}^t\boldsymbol{XW}\right)^{-1}\boldsymbol{W}^t\boldsymbol{X}^t\boldsymbol{y}\,.$$

The second statement can be proven by noting that the constrained minimization problem is equivalent to an unconstrained minimization problem for $\boldsymbol{\beta} = \boldsymbol{W\alpha}$ with $\boldsymbol{\alpha} \in \mathbb{R}^m$. If we plug this into the formula for the OLS estimator, we obtain (4.14). □

The formulas in proposition 4.7 are beneficial for theoretical purposes, but they are computationally inefficient. For PLSR, the calculation can be done in a recursive and faster way. The key point is to find "primal" weight vectors $\widetilde{\boldsymbol{w}}_i$ such that for every i

$$\boldsymbol{t}_i = \boldsymbol{X}_i\boldsymbol{w}_i = \boldsymbol{X}\widetilde{\boldsymbol{w}}_i\,. \tag{4.16}$$

This can be done by exploiting relationship (4.12) and the fact that $\boldsymbol{R}$ is bidiagonal for PLSR.

Proposition 4.8. *For PLSR, the matrix $\boldsymbol{R} = \boldsymbol{T}^T\boldsymbol{XW}$ is upper bidiagonal, that is*

$$r_{ij} = \boldsymbol{t}_i^t\boldsymbol{X}\boldsymbol{w}_j = 0\,,$$

if $i < j$ or $i + 1 > j$.

This result is already shown in Manne (1987). In order to avoid redundancy, we defer the proof until Chapter 5.

Proposition 4.9. *The primal weight vectors $\widetilde{\boldsymbol{w}}_i$ and the regression vector of the univariate NIPALS algorithm are determined by setting $\widetilde{\boldsymbol{w}}_0 = \boldsymbol{0}$ and $\widehat{\boldsymbol{\beta}}^{(0)} = \boldsymbol{0}$ and* *by computing iteratively*

$$\begin{aligned}\widetilde{\boldsymbol{w}}_i &= \boldsymbol{w}_i - \frac{\widetilde{\boldsymbol{w}}_{i-1}^t \boldsymbol{X}^t \boldsymbol{X} \boldsymbol{w}_i}{\widetilde{\boldsymbol{w}}_{i-1}^t \boldsymbol{X}^t \boldsymbol{X} \widetilde{\boldsymbol{w}}_{i-1}} \widetilde{\boldsymbol{w}}_{i-1}\,, \\ \widehat{\boldsymbol{\beta}}^{(i)} &= \widehat{\boldsymbol{\beta}}^{(i-1)} + \frac{\widetilde{\boldsymbol{w}}_i^t \boldsymbol{X}^t \boldsymbol{y}}{\widetilde{\boldsymbol{w}}_i^t \boldsymbol{X}^t \boldsymbol{X} \widetilde{\boldsymbol{w}}_i} \widetilde{\boldsymbol{w}}_i\,.\end{aligned}$$

Proof. We proof the statements via induction. For $i = 1$, we have $\widetilde{\boldsymbol{w}}_1 = \boldsymbol{w}_1$ as $\boldsymbol{X}_1 = \boldsymbol{X}$ and

$$\widehat{\boldsymbol{y}} = \mathcal{P}_{\boldsymbol{t}_1} \boldsymbol{y} = \boldsymbol{X} \boldsymbol{w}_1 \left(\boldsymbol{X} \boldsymbol{w}_1 \boldsymbol{w}_1^t \boldsymbol{X}^t \right)^{-1} \boldsymbol{w}_1^t \boldsymbol{X}^t \boldsymbol{y} = \boldsymbol{X} \frac{\widetilde{\boldsymbol{w}}_1^t \boldsymbol{X}^t \boldsymbol{y}}{\widetilde{\boldsymbol{w}}_1^t \boldsymbol{X}^t \boldsymbol{X} \widetilde{\boldsymbol{w}}_1} \widetilde{\boldsymbol{w}}_1\,.$$

For a general i, we have

$$\boldsymbol{t}_{i+1} = \boldsymbol{X}_{i+1} \boldsymbol{w}_{i+1} = \left(\boldsymbol{X} - \mathcal{P}_{\boldsymbol{t}_1,\ldots,\boldsymbol{t}_i} \boldsymbol{X} \right) \boldsymbol{w}_{i+1} = \boldsymbol{X} \boldsymbol{w}_{i+1} - \mathcal{P}_{\boldsymbol{t}_i} \boldsymbol{X} \boldsymbol{w}_{i+1}\,.$$

The last equality holds as $\boldsymbol{R} = \boldsymbol{T}^t \boldsymbol{X} \boldsymbol{W}$ is bidiagonal. Using formula (A.9) for the projection operator and the induction hypothesis (4.16), it follows that

$$\boldsymbol{t}_{i+1} = \boldsymbol{X} \boldsymbol{w}_{i+1} - \boldsymbol{X} \widetilde{\boldsymbol{w}}_i \left(\widetilde{\boldsymbol{w}}_i^t \boldsymbol{X}^t \boldsymbol{X} \widetilde{\boldsymbol{w}}_i \right)^{-1} \boldsymbol{w}_i^t \boldsymbol{X}^t \boldsymbol{X} \boldsymbol{w}_{i+1}\,.$$

We conclude that

$$\widetilde{\boldsymbol{w}}_{i+1} = \boldsymbol{w}_{i+1} - \frac{\widetilde{\boldsymbol{w}}_i^t \boldsymbol{X}^t \boldsymbol{X} \boldsymbol{w}_{i+1}}{\widetilde{\boldsymbol{w}}_i^t \boldsymbol{X}^t \boldsymbol{X} \widetilde{\boldsymbol{w}}_i} \widetilde{\boldsymbol{w}}_i\,.$$

The regression estimate after i steps is

$$\begin{aligned}\boldsymbol{X} \widehat{\boldsymbol{\beta}}^{(i)} &= \mathcal{P}_{\boldsymbol{t}_1,\ldots,\boldsymbol{t}_i} \boldsymbol{y} \\ &= \mathcal{P}_{\boldsymbol{t}_1,\ldots,\boldsymbol{t}_{i-1}} \boldsymbol{y} + \mathcal{P}_{\boldsymbol{t}_i} \boldsymbol{y} \\ &= \boldsymbol{X} \widehat{\boldsymbol{\beta}}^{(i-1)} + \mathcal{P}_{\boldsymbol{t}_i} \boldsymbol{y} \\ &= \boldsymbol{X} \widehat{\boldsymbol{\beta}}^{(i-1)} + \boldsymbol{X} \frac{\widetilde{\boldsymbol{w}}_i^t \boldsymbol{X}^t \boldsymbol{y}}{\widetilde{\boldsymbol{w}}_i^t \boldsymbol{X}^t \boldsymbol{X} \widetilde{\boldsymbol{w}}_i} \widetilde{\boldsymbol{w}}_i\,.\end{aligned}$$

This concludes the proof. □

4.3 The Degrees of Freedom of Partial Least Squares

The number m of PLSR components is an additional model parameter that has to be determined. In most applications, this is done by computing the cross-validated error. In Chapter 2, we discussed a different strategy for model selection that involves the computation of the degrees of freedom. Note that the PLSR estimate (4.8) is not a linear function of $\boldsymbol{y}$. Hence, we can only estimate the degrees of freedom using equation (2.9). As a consequence, we have to compute the first derivative of the PLSR estimator. This has been done before. Phatak et al. (2002) compute the first derivative of $\widehat{\boldsymbol{\beta}}^{(m)}_{PLS}$ in order to obtain asymptotic results on the variance of the estimator. In that work, some general rules on matrix differential calculus are applied to the formula in proposition 4.7. It turns out that the calculation of the first derivative of PLSR is not only numerically instable, it is also time-consuming. The reason is that the formula for the first derivative of $\widehat{\boldsymbol{\beta}}$ involves matrices of order $(mn)\times n$. We therefore choose a different approach. Serneels et al. (2004) use a recursive formula for $\widehat{\boldsymbol{\beta}}^{(m)}_{PLS}$ that is equivalent to algorithm proposition 4.9 and derive a fast algorithm for the first derivative PLSR. We now show how to compute the derivative of PLSR estimate $\widehat{\boldsymbol{y}}$ in a recursive way. We start with the remark that by definition,

$$\boldsymbol{X}_i \;=\; \boldsymbol{X}_{i-1} - \mathcal{P}_{\boldsymbol{t}_{i-1}}\boldsymbol{X}\,.$$

Using the definition of the weight vectors of PLSR, we conclude that

$$\boldsymbol{w}_i = \boldsymbol{X}_i^t\boldsymbol{y} = \boldsymbol{w}_{i-1} - \boldsymbol{X}^t\mathcal{P}_{\boldsymbol{t}_{i-1}}\boldsymbol{y}\,.$$

Let us define the modified latent components

$$\overline{\boldsymbol{t}}_i \;=\; \boldsymbol{X}\boldsymbol{w}_i\,.$$

We conclude that

$$\overline{\boldsymbol{t}}_i = \overline{\boldsymbol{t}}_{i-1} - \boldsymbol{X}\boldsymbol{X}^t\mathcal{P}_{\boldsymbol{t}_{i-1}}\boldsymbol{y}\,.$$

Furthermore, we conclude from proposition 4.9 that

$$\boldsymbol{t}_i = \boldsymbol{X}\widetilde{\boldsymbol{w}}_i = \boldsymbol{X}\left(\boldsymbol{w}_i - \frac{\widetilde{\boldsymbol{w}}_{i-1}^t\boldsymbol{X}^t\boldsymbol{X}\boldsymbol{w}_i}{\widetilde{\boldsymbol{w}}_{i-1}^t\boldsymbol{X}^t\boldsymbol{X}\widetilde{\boldsymbol{w}}_{i-1}}\widetilde{\boldsymbol{w}}_{i-1}\right) = \overline{\boldsymbol{t}}_i - \frac{\boldsymbol{t}_{i-1}^t\overline{\boldsymbol{t}}_i}{\boldsymbol{t}_{i-1}^t\boldsymbol{t}_{i-1}}\boldsymbol{t}_{i-1} = \overline{\boldsymbol{t}}_i - \mathcal{P}_{\boldsymbol{t}_{i-1}}\overline{\boldsymbol{t}}_i\,.$$

Finally,

$$\widehat{\boldsymbol{y}}^{(i)} = \widehat{\boldsymbol{y}}^{(i-1)} + \mathcal{P}_{\boldsymbol{t}_i}\boldsymbol{y}\,.$$

This leads to the following recursive algorithm for the computation of $\widehat{\boldsymbol{y}}$.

Algorithm 4.10. *We define* $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{X}^t$. *After setting*

$$\boldsymbol{t}_0 = \overline{\boldsymbol{t}}_0 = \widehat{\boldsymbol{y}}^{(0)} = \boldsymbol{0}\,,$$

we iteratively compute the PLSR estimates $\widehat{\boldsymbol{y}}$ *via*

$$\begin{array}{rcll}
\overline{\boldsymbol{t}}_m & = & \begin{cases} \boldsymbol{K}\boldsymbol{y} & m = 1 \\ \overline{\boldsymbol{t}}_{m-1} - \boldsymbol{K}\mathcal{P}_{\boldsymbol{t}_{m-1}}\boldsymbol{y} & m > 1 \end{cases} & \textit{modified latent components} \\
\boldsymbol{t}_m & = & \overline{\boldsymbol{t}}_m - \mathcal{P}_{\boldsymbol{t}_{m-1}}\overline{\boldsymbol{t}}_m & \textit{latent components} \\
\widehat{\boldsymbol{y}}^{(m)} & = & \widehat{\boldsymbol{y}}^{(m-1)} + \mathcal{P}_{\boldsymbol{t}_m}\boldsymbol{y} & \textit{prediction}
\end{array}$$

In order to to compute the first derivative of $\widehat{\boldsymbol{y}}$, we have to calculate the first derivative of the projection operator. The formula can be found in proposition A.11 in the appendix.

Algorithm 4.11 (First derivative of PLSR)**.** *After setting*

$$\boldsymbol{t}_0 = \overline{\boldsymbol{t}}_0 = \widehat{\boldsymbol{y}}^{(0)} = \boldsymbol{0} \quad \textit{and} \quad d\boldsymbol{t}_1 = d\overline{\boldsymbol{t}}_1 = \boldsymbol{K}\,,$$

the first derivative of the PLSR estimator can be obtained by iteratively computing

$$\begin{array}{rcll}
\overline{\boldsymbol{t}}_m & = & \begin{cases} \boldsymbol{K}\boldsymbol{y} & m = 1 \\ \overline{\boldsymbol{t}}_{m-1} - \boldsymbol{K}\mathcal{P}_{\boldsymbol{t}_{m-1}}\boldsymbol{y} & m > 1 \end{cases} & \textit{modified latent components} \\
\frac{\partial \overline{\boldsymbol{t}}_m}{\partial \boldsymbol{y}} & = & \frac{\partial \overline{\boldsymbol{t}}_{m-1}}{\partial \boldsymbol{y}} - \boldsymbol{K}\left(\frac{\partial \mathcal{P}_{\boldsymbol{t}_{m-1}}\boldsymbol{y}}{\partial \boldsymbol{y}}\right) & \textit{derivative of } \overline{\boldsymbol{t}}_m \\
\boldsymbol{t}_m & = & \overline{\boldsymbol{t}}_m - \mathcal{P}_{\boldsymbol{t}_{m-1}}\overline{\boldsymbol{t}}_m & \textit{latent components} \\
\frac{\partial \boldsymbol{t}_m}{\partial \boldsymbol{y}} & = & \frac{\partial \overline{\boldsymbol{t}}_{m-1}}{\partial \boldsymbol{y}} - \frac{\partial \mathcal{P}_{\boldsymbol{t}_{m-1}}\overline{\boldsymbol{t}}_m}{\partial \boldsymbol{y}} & \textit{derivative of } \boldsymbol{t}_m \\
\widehat{\boldsymbol{y}}^{(m)} & = & \widehat{\boldsymbol{y}}^{(m-1)} + \mathcal{P}_{\boldsymbol{t}_m}\boldsymbol{y} & \textit{prediction} \\
\frac{\partial \widehat{\boldsymbol{y}}^{(m)}}{\partial \boldsymbol{y}} & = & \frac{\partial \widehat{\boldsymbol{y}}^{(m-1)}}{\partial \boldsymbol{y}} + \frac{\partial \mathcal{P}_{\boldsymbol{t}_m}\boldsymbol{y}}{\partial \boldsymbol{y}} & \textit{derivative of } \widehat{\boldsymbol{y}}
\end{array}$$

We can now define the estimated degrees of freedom of PLSR with m components via

$$\widehat{\mathrm{df}}(\mathrm{PLSR}, m) \;=\; \mathrm{trace}\left(\frac{\partial \widehat{\boldsymbol{y}}^{(m)}}{\partial \boldsymbol{y}}\right)\,.$$

This is – by definition – an unbiased estimate of the degrees of freedom of PLSR in the case of normally distributed error terms. Although its computation is considerably faster than the one proposed in Phatak et al. (2002), it still suffers from numerical instability. This leads to peculiar and sometimes apparently wrong

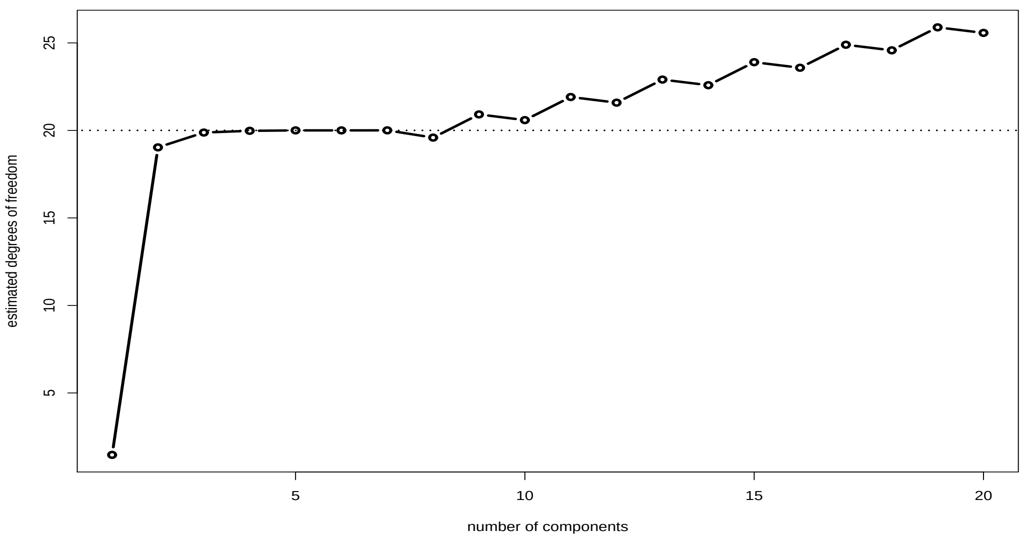

Figure 4.1: The estimated degrees of freedom of PLSR as a function of the number of components.

results. This is illustrated with the following example. We consider the linear regression model

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

with $p = 20$ predictor variables and $n = 500$ examples. First, we choose the predictor matrix $\boldsymbol{X}$ from a multivariate normal distribution with zero mean and covariance matrix $\boldsymbol{S}$ equal to

$$s_{ij} = \begin{cases} 1 & , i = j \\ 0.7 & , i \neq j \end{cases} .$$

This leads to highly collinear data $\boldsymbol{X}$. The regression vector $\boldsymbol{\beta}$ is a randomly chosen vector $\boldsymbol{\beta} \in \{0,1\}^{20}$. We assume that the error terms are normally distributed with $\sigma = 8$. We compute the degrees of freedom for all 20 components. Figure 4.1 shows the peculiar behavior of the estimated degrees of freedom. We expect the degrees of freedom to be upper-bounded by $p = 20$ – the number of predictor variables. This is indicated by the dashed line. Note however that after a few components ($m \geq 9$), the estimated degrees of freedom exceed this value. The function displayed in Figure 4.1 still increases and has the form of a jagged line. This phenomenon persistently occurs for other data sets. We conjecture that algorithm 4.11 runs into serious numerical problems. Therefore, we recommend to be cautious to implement the algorithm in its current form.

Chapter 5

Penalized Partial Least Squares

Nonlinear regression effects may be modeled via additive regression models of the form

$$Y = \beta_0 + f_1(X_1) + \cdots + f_p(X_p) + \varepsilon\,, \tag{5.1}$$

where the functions $f_1, \ldots, f_p$ have unspecified functional form. An approach which allows a flexible representation of the functions $f_1, \ldots, f_p$ is the expansion in B-Splines basis functions (Hastie & Tibshirani 1990). To prevent overfitting, there are two general approaches. In the first approach, each function f_j is the sum of only a small set of basis functions,

$$f_j(x) = \sum_{k=1}^{K_j} \beta_{kj} B_{kj}(x)\,. \tag{5.2}$$

The basis functions B_{kj} are chosen adaptively by a selection procedure. The second approach circumvents the problem of basis function selection. Instead, we allow a generous amount $K_j \gg 1$ of basis functions in the expansion (5.2). As this usually leads to high-dimensional and highly correlated data, we penalize the coefficients β_{jk} in the estimation process (Eilers & Marx 1996). However, if the number p of predictors is large compared to the number n of observations in the available sample, these methods are impracticable.

Quite generally, a different approach to deal with high dimensionality is to use dimension reduction techniques such as Partial Least Squares Regression (PLSR) which is presented in Chapter 4. In this chapter, we suggest an adaptation of the principle of penalization to PLSR. More precisely, we present a penalized version of the optimization problem (4.3) attached to PLSR. Although the motivation stems from its use for B-splines transformed data, the proposed approach is very general and can be adapted to other penalty terms or to other dimension reduction techniques such as Principal Components Analysis. It turns out that the new method shares a lot of properties of PLSR and that that its computation requires virtually no extra costs. Furthermore, we show that this new penalization technique is closely related to the kernel trick that is illustrated in Section 1.4. We show that penalized PLSR is equivalent to ordinary PLSR using a generalized inner product that is defined by the penalty term. In the case of high-dimensional data, the new method is shown to be an attractive competitor to other techniques for estimating generalized additive models. In Chapter 6, we highlighten the close connection between penalized PLSR and preconditioned linear systems.

This chapter is joint work with Anne-Laure Boulesteix and Gerhard Tutz.

5.1 Penalized Regression Splines

The fitting of generalized additive models by use of penalized regression splines (Eilers & Marx 1996) has become a widely used tool in statistics. The basic idea is to expand the additive component of each variable X_j in basis functions as in (5.2) and to estimate the coefficients by penalization techniques. As suggested in Eilers & Marx (1996), B-splines are used as basis functions. Splines are one-dimensional piecewise polynomial functions. The points at which the pieces are connected are called knots or breakpoints. We say that a spline is of order d if all polynomials are of degree $\leq d$ and if the spline is $(d-1)$ times continuously differentiable at the breakpoints. A particular efficient set of basis functions are B-splines (de Boor 1978). An example of B-splines is given in Figure 5.1.

The number of basis functions depends on the order of the splines and the number of breakpoints. For a given variable X_j, we consider a set of corresponding B-splines basis functions $B_{1j}, \ldots, B_K$. These basis functions define a nonlinear map

$$\Phi_j(x) \quad = \quad (B_{1j}(x), \ldots, B_K(x))^t \; .$$

By performing such a transformation on each of the variables $X_1, \ldots, X_p$, the

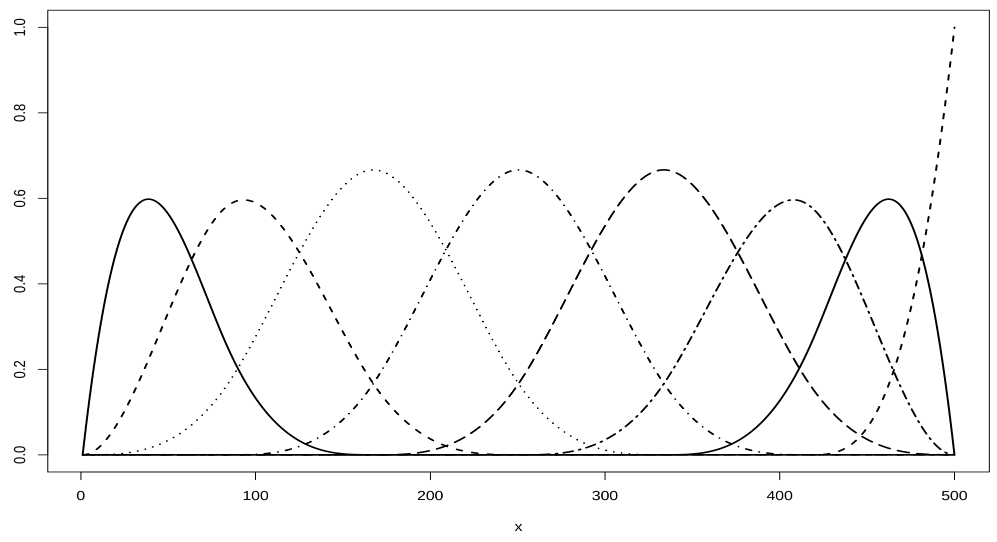

Figure 5.1: Illustration of a basis of B-splines of order 3.

observation vector $\boldsymbol{x}_i$ turns into a vector

$$\begin{aligned} \boldsymbol{z}_i &= (B_{11}(x_{i1}),\dots,B_{K1}(x_{i1}),\dots,B_{1p}(x_{ip}),\dots,B_{Kp}(x_{ip}))^t \qquad (5.3) \\ &= \Phi(\boldsymbol{x}_i)\,. \end{aligned}$$

of length pK. Here

$$\begin{aligned} \Phi:\mathbb{R}^p &\rightarrow \mathbb{R}^{pK}\,, \\ \Phi(\boldsymbol{x}) &= (\Phi_1(x_1),\dots,\Phi_p(x_p))\ , \end{aligned}$$

is the function defined by the B-splines. The resulting data matrix obtained by the transformation of $\boldsymbol{X}$ has dimensions $n\times pK$ and will be denoted by $\boldsymbol{Z}$ in the rest of the chapter. In the examples in Sections 5.5 and **??**, we consider the most widely used cubic B-splines, i.e. we choose $d=3$.

The estimation of (5.1) is transformed into the estimation of the pK-dimensional vector $\boldsymbol{\beta}$ that consists of the coefficients β_{jk}:

$$\boldsymbol{\beta}^t = (\beta_{11},\dots,\beta_{K1},\dots\beta_{12},\dots,\beta_{Kp}) = \left(\boldsymbol{\beta}^t_{(1)},\dots,\boldsymbol{\beta}^t_{(p)}\right)\ .$$

As explained above, the vector $\boldsymbol{\beta}$ determines a nonlinear, additive function

$$f(\boldsymbol{x}) = \beta_0 + \sum_{j=1}^{p} f_j(x_j) = \beta_0 + \sum_{j=1}^{p}\sum_{k=1}^{K} \beta_{kj} B_{kj}(x_j) = \beta_0 + \Phi(\boldsymbol{x})^t \boldsymbol{\beta}\,.$$

As $\boldsymbol{Z}$ is usually high-dimensional, the estimation of $\boldsymbol{\beta}$ by minimizing the squared empirical risk

$$\widehat{R}(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n} (y_i - f(\boldsymbol{x}_i))^2 = \frac{1}{n}\|\boldsymbol{y} - \beta_0 - \boldsymbol{Z}\boldsymbol{\beta}\|^2$$

usually leads to overfitting. Following Eilers & Marx (1996), we use for each variable many basis functions, say $K \approx 20$, and estimate by penalization. The idea is to penalize the second derivative of the function f,

$$\int (f''(x))^2 \, dx\,.$$

Eilers & Marx (1996) show that the following difference penalty term is a good approximation of the penalty on the second derivative of the functions f_j,

$$P(\boldsymbol{\beta}) \;=\; \sum_{j=1}^{p}\sum_{k=3}^{m} \lambda_j (\Delta^2 \beta_{kj})^2\,.$$

Here $\lambda_j \geq 0$ are smoothing parameters that control the amount of penalization. These are also called the second-order differences of adjacent parameters. The difference operator $\Delta^2\beta_{kj}$ has the form

$$\begin{aligned}\Delta^2\beta_{kj} &= (\beta_{kj} - \beta_{k-1,j}) - (\beta_{k-1,j} - \beta_{k-2,j}) \\ &= \beta_{kj} - 2\beta_{k-1,j} + \beta_{k-2,j}.\end{aligned}$$

This penalty term can be expressed in terms of a penalty matrix $\boldsymbol{P}$. We denote by $\boldsymbol{D}_K$ the $(K-1)\times K$ matrix

$$\boldsymbol{D}_K \;=\; \begin{pmatrix} 1 & -1 & . & . & . \\ . & 1 & -1 & . & . \\ . & . & . & . & . \\ . & . & . & 1 & -1 \end{pmatrix}$$

that defines the first order difference operator. Setting

$$\boldsymbol{K}_2 = (\boldsymbol{D}_{K-1}\boldsymbol{D}_K)^t \boldsymbol{D}_{K-1}\boldsymbol{D}_K \,,$$

we conclude that the penalty term equals

$$P(\boldsymbol{\beta}) = \sum_{j=1}^{p} \lambda_j \boldsymbol{\beta}_{(j)}^t \boldsymbol{K}_2 \boldsymbol{\beta}_{(j)} = \boldsymbol{\beta}^t (\boldsymbol{\Delta}_{\boldsymbol{\lambda}} \otimes \boldsymbol{K}_2)\boldsymbol{\beta}\,.$$

Here $\boldsymbol{\Delta}_{\boldsymbol{\lambda}}$ is the $p \times p$ diagonal matrix containing $\lambda_1, \ldots, \lambda_p$ on its diagonal and $\otimes$ is the Kronecker product. The generalization of this method to higher-order differences of the coefficients of adjacent B-splines is straightforward. We simply replace $\boldsymbol{K}_2$ by

$$\boldsymbol{K}_q = (\boldsymbol{D}_{K-q+1} \ldots \boldsymbol{D}_K)^t (\boldsymbol{D}_{K-q+1} \ldots \boldsymbol{D}_K)\,.$$

To summarize, the penalized least squares criterion has the form

$$\widehat{R}_P(\boldsymbol{\beta}) \;=\; \frac{1}{n}\|\boldsymbol{y} - \beta_0 \mathbf{1} - \boldsymbol{Z}\boldsymbol{\beta}\|^2 + \boldsymbol{\beta}^t \boldsymbol{P} \boldsymbol{\beta} \qquad (5.4)$$

with $\boldsymbol{Z}$ the transformed data that is defined in (5.3) and the penalty matrix $\boldsymbol{P}$ defined as

$$\boldsymbol{P} \;=\; \boldsymbol{\Delta}_{\boldsymbol{\lambda}} \otimes \boldsymbol{K}_q\,. \qquad (5.5)$$

This is a symmetric matrix that is positive semidefinite.

5.2 Dimension Reduction for B-Splines Transformations

As a linear approach, PLSR might fail to yield high prediction accuracy in the case of nonlinear relationships between predictors and responses. The idea to transform the original predictors using splines functions preliminarily to PLSR in order to incorporate such nonlinear structures has been proposed by Durand & Sabatier (1997) and Durand (1993) in different variants. The method proposed by Durand & Sabatier (1997) is based on a variant of PLSR that may be computed via an iterative algorithm. They suggest an approach that incorporates splines transformations of the predictors within each iteration of the iterative algorithm. In contrast, the method proposed by Durand (2001) is global. The predictors are first transformed using splines basis functions as a preliminary step, then PLSR is performed on the transformed data matrix. The choice of the degree d of the polynomial pieces and of the number of knots is performed by an either ascending or descending search procedure that is not automatic.

For large numbers of variables, this search procedure is computationally intensive. We suggest an alternative approach based on the penalty strategy of Eilers & Marx (1996). In the next section, we show how the penalty scheme of Eilers & Marx (1996) can be mapped into the PLSR dimension reduction framework.

5.3 Partial Least Squares and Penalization

We now present a general framework to combine PLSR with penalization terms. We remark that this is not limited to spline transformed variables or the the special shape of the penalty matrix $\boldsymbol{P}$ that is defined in (5.5). For this reason, we present the new method in terms of the original data matrix $\boldsymbol{X}$ and we only demand that $\boldsymbol{P}$ is a symmetric matrix such that $\boldsymbol{I}_p + \boldsymbol{P}$ is positive definite.

We start with a general response $\boldsymbol{Y}$ and then focus on univariate responses. We modify the optimization criterion (4.3) of PLS in the following way. The first component $\boldsymbol{t}_1 = \boldsymbol{X}\boldsymbol{w}_1$ is the solution of the problem

$$\arg\max_{\boldsymbol{w}} \frac{\boldsymbol{w}^t \boldsymbol{X}^t \boldsymbol{Y} \boldsymbol{Y}^t \boldsymbol{X} \boldsymbol{w}}{\boldsymbol{w}^t \boldsymbol{w} + \boldsymbol{w}^t \boldsymbol{P} \boldsymbol{w}} . \tag{5.6}$$

Using Lagrangian multipliers and recalling the definition

$$\boldsymbol{B} = \boldsymbol{X}^t \boldsymbol{Y} \boldsymbol{Y}^t \boldsymbol{X} ,$$

we deduce that the solution must fulfill

$$\boldsymbol{B}\boldsymbol{w}_1 = \nu \left(\boldsymbol{I}_p + \boldsymbol{P}\right) \boldsymbol{w}_1 , \nu \in \mathbb{R} .$$

For a general $\boldsymbol{Y}$, this is called a generalized eigenvalue problem. If $\boldsymbol{Y} = \boldsymbol{y}$ is univariate, we have $\boldsymbol{B} = \boldsymbol{b}\boldsymbol{b}^t$ and the solution is

$$\boldsymbol{w}_1 = \left(\boldsymbol{I}_p + \boldsymbol{P}\right)^{-1} \boldsymbol{b} .$$

We set

$$\boldsymbol{M} = (\boldsymbol{I}_p + \boldsymbol{P})^{-1} . \tag{5.7}$$

From now on, we only consider univariate responses. Subsequent weight vectors and components are computed by deflating $\boldsymbol{X}$ as described in Section 4.1 and then maximizing (5.6) with $\boldsymbol{X}$ replaced by $\boldsymbol{X}_i$. In particular, we can compute the weight vectors and components of penalized PLSR by simply replacing $\boldsymbol{w}_i = \boldsymbol{X}_i^t \boldsymbol{y}$ by

$$\boldsymbol{w}_i = \boldsymbol{M}\boldsymbol{X}_i^t \boldsymbol{y}$$

in algorithm 4.2. The following generalization of proposition 4.8 holds.

Proposition 5.1. *For penalized PLS, the matrix $\boldsymbol{R} = \boldsymbol{T}^t\boldsymbol{X}\boldsymbol{W}$ is upper bidiagonal.*

The proof can be found in chapter 6. This proposition can be used for an efficient computation of the regression vector. This is a generalization of proposition 4.9.

Algorithm 5.2 (Penalized PLSR)**.** *For a penalty matrix $\boldsymbol{P}$, we define $\boldsymbol{M} = (\boldsymbol{I}_p + \boldsymbol{P})^{-1}$. After setting $\boldsymbol{X}_1 = \boldsymbol{X}$, $\widetilde{\boldsymbol{w}}_0 = \boldsymbol{0}$ and $\widehat{\boldsymbol{\beta}}^{(0)} = \boldsymbol{0}$, the weight vectors $\boldsymbol{w}_i, \widetilde{\boldsymbol{w}}_i$, the components $\boldsymbol{t}_i$ and the regression vectors $\widehat{\boldsymbol{\beta}}^{(i)}$ of penalized PLSR are determined by iteratively computing*

$$\begin{array}{rcll}
\boldsymbol{w}_i &=& \boldsymbol{M}\boldsymbol{X}_i^t\boldsymbol{y} & \textit{weight vector} \\
\widetilde{\boldsymbol{w}}_i &=& \boldsymbol{w}_i - \frac{\widetilde{\boldsymbol{w}}_{i-1}^t \boldsymbol{X}^t\boldsymbol{X}\boldsymbol{w}_i}{\widetilde{\boldsymbol{w}}_{i-1}^t\boldsymbol{X}^t\boldsymbol{X}\widetilde{\boldsymbol{w}}_{i-1}}\widetilde{\boldsymbol{w}}_{i-1} & \textit{primal weight vector} \\
\widehat{\boldsymbol{\beta}}^{(i)} &=& \widehat{\boldsymbol{\beta}}^{(i-1)} + \frac{\widetilde{\boldsymbol{w}}_i^t\boldsymbol{X}^t\boldsymbol{y}}{\widetilde{\boldsymbol{w}}_i^t\boldsymbol{X}^t\boldsymbol{X}\widetilde{\boldsymbol{w}}_i}\widetilde{\boldsymbol{w}}_i & \textit{regression vector} \\
\boldsymbol{t}_i &=& \boldsymbol{X}_i\boldsymbol{w}_i & \textit{component} \\
\boldsymbol{X}_{i+1} &=& \boldsymbol{X}_i - \mathcal{P}_{\boldsymbol{t}_i}\boldsymbol{X}_i & \textit{deflation}
\end{array}$$

Note that we defined penalized PLSR only in terms of the NIPALS algorithm. It is however straightforward to adapt the SIMPLS algorithm to the penalization approach. For a univariate response, we show their equivalence in corollary 6.4. As the two methods differ for multivariate responses in the case of PLSR, we expect them to be different for the penalized version as well.

In the next section, we derive a representation of penalized PLSR in terms of a Kernel matrix and illustrate the geometric intuition behind the penalty term.

5.4 Kernel Penalized Partial Least Squares

The computation of the penalized PLSR estimator as presented in algorithm 5.2 involves matrices and vectors of dimension $p \times p$ and p respectively. If the number of predictors p is very large, this leads to high computational costs. In this section, we show that we can represent the penalized PLSR algorithm in terms of matrices and vectors of dimension $n \times n$ and n respectively.

Let us define the $n \times n$ matrix $\boldsymbol{K}$ via

$$\boldsymbol{K} = \boldsymbol{K}_{\boldsymbol{M}} = (\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle_{\boldsymbol{M}}) = \boldsymbol{X}\boldsymbol{M}\boldsymbol{X}^t \,.$$

This matrix is the Gram matrix or the kernel matrix of $\boldsymbol{X}$ if we use the inner product $\langle \cdot, \cdot \rangle_{\boldsymbol{M}}$ defined by $\boldsymbol{M}$. In order to apply the kernel trick described in Section 1.4, we have to show that the penalized PLSR estimator can be represented in terms of dual variables,

$$\begin{aligned} \widehat{\boldsymbol{\beta}}^{(m)} &= \boldsymbol{M}\boldsymbol{X}^t \boldsymbol{\alpha}^{(m)} \,, \\ \boldsymbol{\alpha}^{(m)} &\in \mathbb{R}^n \,. \end{aligned}$$

To show this, we first recall equation (4.10) from which follows that

$$\boldsymbol{X}_i^t \boldsymbol{y} = \boldsymbol{X}^t \left(\boldsymbol{I}_n - \mathcal{P}_{\boldsymbol{t}_1, \ldots, \boldsymbol{t}_{i-1}} \right) \boldsymbol{y} = \boldsymbol{X}^t \left(\boldsymbol{y} - \widehat{\boldsymbol{y}}^{(i-1)} \right) \,.$$

We conclude that the weight vectors $\boldsymbol{w}_i$ of penalized PLSR are simply

$$\boldsymbol{w}_i = \boldsymbol{M}\boldsymbol{X}^t \boldsymbol{y}_{res}^{(i)} \quad , \quad \boldsymbol{y}_{res}^{(i)} = \boldsymbol{y} - \widehat{\boldsymbol{y}}^{(i-1)} \,.$$

Plugging this into proposition 4.9, we also obtain a dual representation of the modified weight vectors

$$\widetilde{\boldsymbol{w}}_i = \boldsymbol{M}\boldsymbol{X}^t \widetilde{\boldsymbol{\alpha}}_i$$

by iteratively computing

$$\begin{aligned}
\widetilde{\boldsymbol{w}}_i &= \boldsymbol{w}_i - \frac{\widetilde{\boldsymbol{w}}_{i-1}^t \boldsymbol{X}^t \boldsymbol{X} \boldsymbol{w}_i}{\widetilde{\boldsymbol{w}}_{i-1}^t \boldsymbol{X}^t \boldsymbol{X} \widetilde{\boldsymbol{w}}_{i-1}} \widetilde{\boldsymbol{w}}_{i-1} \\
&= \boldsymbol{M} \boldsymbol{X}^t \boldsymbol{y}_{res}^{(i)} - \frac{\widetilde{\boldsymbol{\alpha}}_{i-1}^t \boldsymbol{X} \boldsymbol{M} \boldsymbol{X}^t \boldsymbol{X} \boldsymbol{M} \boldsymbol{X}^t \boldsymbol{y}_{res}^{(i)}}{\widetilde{\boldsymbol{\alpha}}_{i-1}^t \boldsymbol{X} \boldsymbol{M} \boldsymbol{X}^t \boldsymbol{X} \boldsymbol{M} \boldsymbol{X}^t \widetilde{\boldsymbol{\alpha}}_{i-1}} \boldsymbol{M} \boldsymbol{X}^t \widetilde{\boldsymbol{\alpha}}_{i-1} \\
&= \boldsymbol{M} \boldsymbol{X}^t \left(\boldsymbol{y}^{res} - \frac{\widetilde{\boldsymbol{\alpha}}_{i-1}^t \boldsymbol{K}_{\boldsymbol{M}}^2 \boldsymbol{y}_{res}^{(i)}}{\widetilde{\boldsymbol{\alpha}}_{i-1}^t \boldsymbol{K}_{\boldsymbol{M}}^2 \widetilde{\boldsymbol{\alpha}}_{i-1}} \widetilde{\boldsymbol{\alpha}}_{i-1} \right).
\end{aligned}$$

This leads in turn to a dual representation of the regression vector $\widehat{\boldsymbol{\beta}}^{(i)}$. We can now state algorithm 5.2 in terms of the Kernel matrix $\boldsymbol{K}_{\boldsymbol{M}}$ and the response vector $\boldsymbol{y}$.

Algorithm 5.3. *For a penalty term $\boldsymbol{P}$, we define $\boldsymbol{M} = (\boldsymbol{I}_p + \boldsymbol{P})^{-1}$ and $\boldsymbol{K}_{\boldsymbol{M}} = \boldsymbol{X}\boldsymbol{M}\boldsymbol{X}^t$. After setting*

$$\boldsymbol{\alpha}^{(i)} = \widetilde{\boldsymbol{\alpha}}_i = \boldsymbol{0}\,,$$

the dual representation of the penalized PLSR estimator can be computed iteratively via

$$\begin{aligned}
\boldsymbol{y}_{res}^{(i)} &= \boldsymbol{y} - \widehat{\boldsymbol{y}}^{(i-1)} && \textit{residuals} \\
\widetilde{\boldsymbol{\alpha}}_i &= \boldsymbol{y}_{res}^{(i)} - \frac{\widetilde{\boldsymbol{\alpha}}_{i-1}^t \boldsymbol{K}_M^2 \boldsymbol{y}_{res}^{(i)}}{\widetilde{\boldsymbol{\alpha}}_{i-1}^t \boldsymbol{K}_M^2 \widetilde{\boldsymbol{\alpha}}_{i-1}} \widetilde{\boldsymbol{\alpha}}_{i-1} && \textit{primal weight vector} \\
\boldsymbol{\alpha}^{(i)} &= \boldsymbol{\alpha}^{(i-1)} + \frac{\widetilde{\boldsymbol{\alpha}}_i^t \boldsymbol{K}_M \boldsymbol{y}}{\widetilde{\boldsymbol{\alpha}}_i^t K^2 \widetilde{\boldsymbol{\alpha}}_i} \widetilde{\boldsymbol{\alpha}}_i && \textit{regression vector} \\
\boldsymbol{t}_i &= \boldsymbol{K}_M \widetilde{\boldsymbol{\alpha}}_i && \textit{component} \\
\widehat{\boldsymbol{y}}^{(i+1)} &= \widehat{\boldsymbol{y}}^{(i)} + \mathcal{P}_{\boldsymbol{t}_i} \boldsymbol{y} && \textit{estimation of } \boldsymbol{y}
\end{aligned}$$

A Kernel version of PLSR has already been defined in Rännar et al. (1994) in order to speed up the computation of PLSR. The importance of this "dual" representation becomes apparent if we want to extend PLSR to nonlinear problems by using the kernel trick. We already discussed this aspect in Section 1.4. A nonlinear version of PLSR using the kernel trick is presented in Rosipal & Trejo (2001). Note that ordinary Kernel PLSR applied to the transformed data (5.3) is in fact Kernel PLSR with the feature map Φ defined by the B-splines.

If we represent penalized PLSR in terms of the kernel matrix $\boldsymbol{K}_{\boldsymbol{M}}$, we realize that penalized PLSR is closely connected to the kernel trick in other respects. It follows immediately from algorithm 5.3 that penalized PLSR is equivalent to

PLSR with the usual inner product replaced by the inner product

$$\langle \boldsymbol{x}, \boldsymbol{z} \rangle_{\boldsymbol{M}} = \boldsymbol{x}^t \boldsymbol{M} \boldsymbol{z} \,.$$

Why is this a sensible inner product? Let us consider the eigen decomposition of the penalty matrix $\boldsymbol{P}$,

$$\boldsymbol{P} = \boldsymbol{S} \boldsymbol{\Theta} \boldsymbol{S}^t \,.$$

We prefer direction $\boldsymbol{s}$ such that $\boldsymbol{s}^t \boldsymbol{P} \boldsymbol{s}$ is small, that is we prefer directions that are defined by eigenvectors $\boldsymbol{s}_i$ of $\boldsymbol{P}$ with a small corresponding eigenvalue θ_i. If we represent the vectors $\boldsymbol{x}$ and $\boldsymbol{z}$ in terms of the eigenvectors of $\boldsymbol{P}$,

$$\widetilde{\boldsymbol{x}} = \boldsymbol{S}^t \boldsymbol{x} \quad , \quad \widetilde{\boldsymbol{z}} = \boldsymbol{S}^t \boldsymbol{z} \,,$$

we conclude that

$$\langle \boldsymbol{x}, \boldsymbol{z} \rangle_{\boldsymbol{M}} = \widetilde{\boldsymbol{x}}^t \left(\boldsymbol{I}_p + \boldsymbol{\Theta} \right)^{-1} \widetilde{\boldsymbol{z}} = \sum_{i=1}^{p} \frac{1}{1 + \theta_i} \widetilde{\boldsymbol{x}}_i \widetilde{\boldsymbol{z}}_i \,.$$

This implies that directions $\boldsymbol{s}_i$ with a small eigenvalue θ_i receive a higher weighting than directions with a large eigenvalue. This allows an intuitive geometric interpretation of the penalty term.

5.5 Example: Birth Data Set

In this section, we analyze a real data set describing pregnancy and delivery for 42 infants who were sent to a neonatal intensive care unit after birth. The data are taken from the `R` software package `exactmaxsel` and are introduced in Boulesteix (2006). Our goal is to predict the number of days spent in the neonatal intensive care unit (y) based on the following predictors: birth weight (in g), birth height (in cm), head circumference (in cm), term (in week), age of the mother (in year), weight of the mother before pregnancy (in kg), weight of the mother before delivery (in kg), height of the mother (in cm), time (in month). Some of the predictors are expected to be strongly associated with the response (e.g., birth weight, term), in contrast to poor predictors like time or height of the mother.

The parameter settings are as follows. We make the simplifying assumption that $\lambda = \lambda_1 = \ldots = \lambda_p$, which reduces the problem of selecting the optimal smoothing parameter to a one-dimensional problem. As already mentioned above, we use cubic splines. Furthermore, the order of difference of adjacent weights is set to 2.

The shape of the fitted functions f_j depends on the two model parameters λ and m. We first illustrate that the number m of penalized PLSR components controls the smoothness of the estimated functions. For this purpose, we only consider the predictor variable "weight". Figure 5.2 displays the fitted functions obtained by penalized PLSR for $\lambda = 2000$ and 4 different numbers of components $m = 1, 5, 9, 13$. For small values of m, the obtained functions are smooth. For

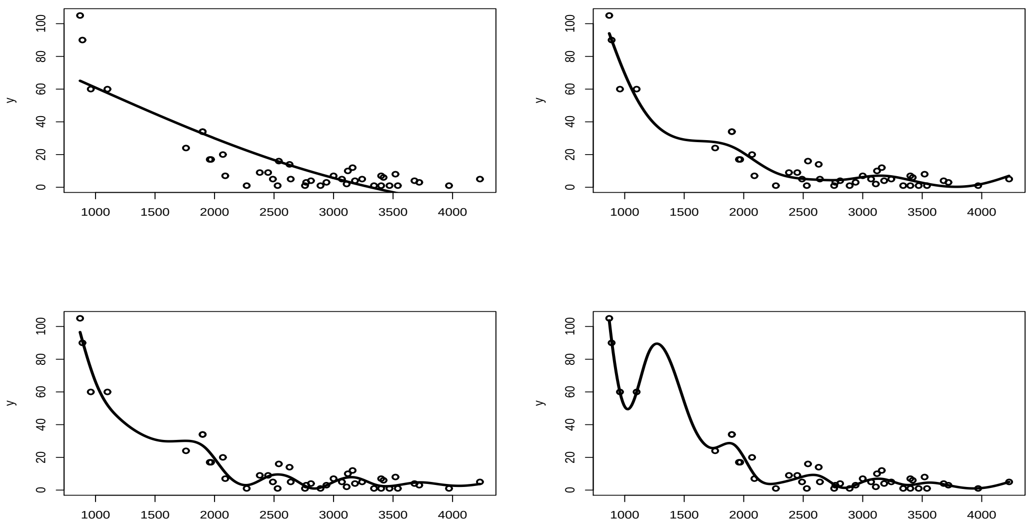

Figure 5.2: Fitted function for the predictor variable "weight" using penalized PLSR. The value of λ is 2000 and the numbers of components are $1, 5$ (top) and $9, 13$ (bottom).

higher values of m, the functions adapt themselves more and more to the data which leads to overfitting for high values of m.

We compare our novel method to PLSR without penalization as described in Durand (2001) and the `gam()` package in `R`. This is the implementation of an adaptive selection procedure for the basis functions in (5.2). More details can be found in Wood (2000) and Wood (2006). This is the standard tool for estimating generalized additive models. In order to assess the performance of the three methods, we randomly split the data into a training set of size 32 and a test set of size 10. The optimal parameter values are chosen by minimizing the leave-one-out error on the training set. The optimal model is then evaluated at the test set.

We remark that the split into training and test set is done before transforming the original predictors using B-splines. This random splitting is repeated 50 times.

In order to have comparable results, we normalize the response. i.e. $\mathrm{var}(\boldsymbol{y}) = 1$. A boxplot of the test error for the three methods is displayed in Figure 5.3.

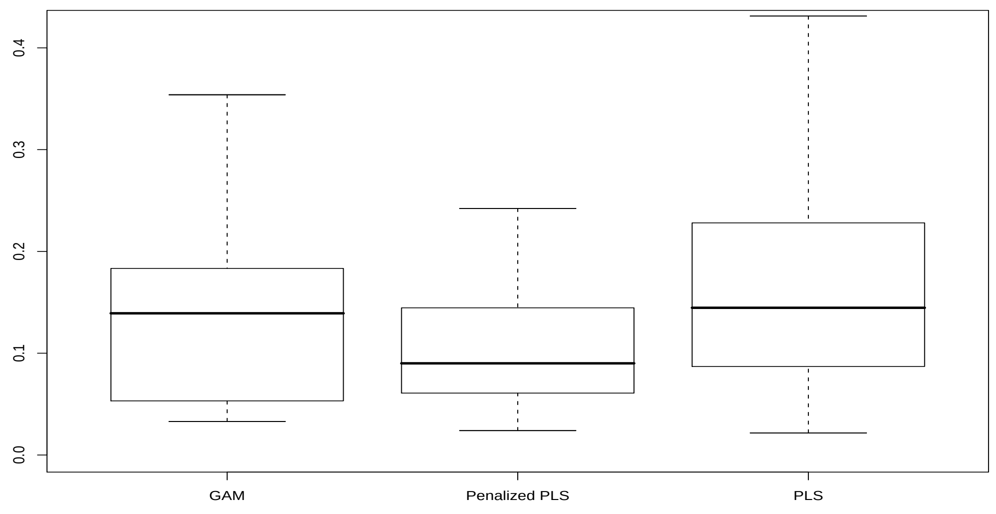

Figure 5.3: Boxplot of the 50 test errors for the three methods GAM, penalized PLSR and PLSR without penalization.

The median of the test errors and the optimal parameter values (estimated on the complete data set via leave-one-out) can be found in Table 5.1. Furthermore, we conduct a Wilcoxon rank sum test to test the alternative hypothesis that the test error of penalized PLSR is lower than the test error of GAM and PLSR respectively. The p-values can also be found in Table 5.1. Penalized PLSR is the

	median test error	m_{opt}	λ_{opt}	p-values
GAM	0.139	–	–	0.089
penalized PLSR	0.090	2	330	–
PLSR	0.145	8	–	0.004

Table 5.1: Comparison of GAM, penalized PLS and PLS. The first three columns display the median test error and optimal model parameters for the birth data set and normalized response. The last column displays the p-value of the Wilcoxon rank sum test.

best out of the three method. In particular, it receives a considerably lower error than PLSR without penalization.

5.6 Conclusion

In this chapter, we proposed an extension of Partial Least Squares Regression using penalization techniques. Apart from its computational efficiency (it is virtually as fast as PLSR), it also shares a lot of mathematical properties of PLSR. This will be stressed further in the next chapter. There, we prove that penalized PLSR is equal to a preconditioned conjugate gradient descent. Our new method also obtains good results in applications. In the example that is discussed in Section 5.5, PLSR clearly outperforms PLSR without penalization. Furthermore, the results indicate that it is a competitor of `gam()` in the case of very high-dimensional data.

We might think of other penalty terms. Kondylis & Whittaker (2006) consider a preconditioned version of PLSR by giving weights to the predictor variables. Higher weights are given to those predictor variables that are highly correlated to the response. These weights can be expressed in terms of a penalty matrix. Goutis & Fearn (1996) combine PLS with an additive penalty term to data derived from near infra red spectroscopy. The penalty term controls the smoothness of the regression vector.

The introduction of a penalty term can easily be adapted to other dimension reduction techniques. For example, for Principal Components Analysis, the penalized optimization criterion is

$$\max_{\boldsymbol{w}} \frac{\operatorname{var}(\boldsymbol{X}\boldsymbol{w})}{\boldsymbol{w}^t\boldsymbol{w} + \boldsymbol{w}^t\boldsymbol{P}\boldsymbol{w}} \, .$$

The novel penalized PLSR approach has however one drawback. The amount of smoothness used for any of the additive components f_j is the same. Using different values λ_j for each component leads to a model selection problem that involves a high-dimensional model parameter $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_p)$. This is often infeasible. In Section 5.5, we illustrated that the amount of smoothness can also be controlled in terms of the number m of components. In order to obtain more flexibility, it might be possible to assign different numbers of components to each predictor variable. An elegant way to do so is the Boosting framework that is introduced in Chapter 9. We discuss a possible combination of penalized PLSR and Boosting in Chapter 10.

Chapter 6

From Linear Regression to Linear Algebra

As already mentioned in Section 1.3, the OLS estimator $\widehat{\boldsymbol{\beta}}_{OLS}$ is the solution of

$$\arg\min_{\boldsymbol{\beta}} \quad \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\| \,. \tag{6.1}$$

This problem is equivalent to computing the solution of the normal equations

$$\boldsymbol{A}\boldsymbol{\beta} \;=\; \boldsymbol{b}\,, \tag{6.2}$$

with

$$\boldsymbol{A} \;=\; \boldsymbol{X}^t\boldsymbol{X} \quad \text{and} \quad \boldsymbol{b} \;=\; \boldsymbol{X}^t\boldsymbol{y}\,.$$

Using the Moore-Penrose inverse of $\boldsymbol{A}$, it follows that – as displayed in (1.11) –

$$\widehat{\boldsymbol{\beta}}_{OLS} = \boldsymbol{A}^{-}\boldsymbol{b} = \sum_{i=1}^{rk(\boldsymbol{X})} \boldsymbol{z}_i\,.$$

We already mentioned in Section 1.3 that in the case of high-dimensional data, the matrix $\boldsymbol{A}$ is (almost) singular and that the OLS estimator performs poorly on new data sets. A popular strategy is to regularize the least squares criterion (6.1) in the hope of improving the performance of the estimator. This often corresponds to finding approximate solutions of the normal equations (6.2). For example, Ridge Regression

$$\arg\min_{\boldsymbol{\beta}} \left\{\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|^2\right\}\,,\ \lambda > 0\,.$$

corresponds to the solution of the modified normal equations

$$(\boldsymbol{A} + \lambda \boldsymbol{I}_p)\,\boldsymbol{\beta} \;=\; \boldsymbol{b}\,.$$

Here $\lambda > 0$ is the Ridge parameter. Principal Components Regression uses the eigen decomposition of $\boldsymbol{A}$,

$$\boldsymbol{A} \;=\; \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^t = \sum_{i=1}^{p} \lambda_i \boldsymbol{u}_i \boldsymbol{u}_i^t\,,$$

and approximates $\boldsymbol{A}$ and $\boldsymbol{b}$ via the first m eigenvectors

$$\boldsymbol{A} \;\approx\; \textstyle\sum_{i=1}^{m} \lambda_i \boldsymbol{u}_i \boldsymbol{u}_i^t \quad , \quad \boldsymbol{b} \;\approx\; \textstyle\sum_{i=1}^{m} (\boldsymbol{u}_i^t \boldsymbol{b})\,\boldsymbol{u}_i\,.$$

The Principal Component Regression (PCR) estimator is then defined as

$$\widehat{\boldsymbol{\beta}}_{PCR} \;=\; \sum_{i=1}^{m} \boldsymbol{z}_i\,.$$

Here $\boldsymbol{z}_i$ is the component of $\widehat{\boldsymbol{\beta}}_{OLS}$ along the ith principal component. It can be shown that the PLSR algorithm 4.2 for a univariate response $\boldsymbol{y}$ is equivalent to the conjugate gradient method (Hestenes & Stiefel 1952). This is a procedure that iteratively computes approximate solutions of (6.2) by minimizing the quadratic function

$$\phi(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{\beta}^t \boldsymbol{A}\boldsymbol{\beta} - \boldsymbol{\beta}^t \boldsymbol{b} = \frac{1}{2}\langle \boldsymbol{\beta}, \boldsymbol{A}\boldsymbol{\beta}\rangle - \langle \boldsymbol{\beta}, \boldsymbol{b}\rangle \tag{6.3}$$

along directions that are **A**-orthogonal. (A precise definition of the algorithm is given below.) Recall that two vectors $\boldsymbol{x}$ and $\boldsymbol{x}'$ are $\boldsymbol{A}$-orthogonal if

$$\langle \boldsymbol{x}, \boldsymbol{x}'\rangle_{\boldsymbol{A}} = \boldsymbol{x}^t \boldsymbol{A}\boldsymbol{x}' = 0\,.$$

The approximate solution of the conjugate gradient method obtained after m steps is equal to the PLSR estimator obtained after m iterations. The conjugate gradient algorithm is in turn closely related to Krylov subspaces and the Lanczos algorithm (Lanczos 1950). The latter is a method for approximating eigenvalues. The connection between PLSR and these methods is well-elaborated in Phatak & de Hoog (2002). We now establish a similar connection between penalized PLSR and the above mentioned methods. Set

$$\boldsymbol{A_M} \;=\; \boldsymbol{MA} \quad \text{and} \quad \boldsymbol{b_M} \;=\; \boldsymbol{Mb}\,.$$

Here, $\boldsymbol{M}$ is the matrix (5.7) that is defined by the penalty term $\boldsymbol{P}$ of penalized PLSR. We now illustrate that penalized PLSR finds approximate solutions of the preconditioned normal equation

$$\boldsymbol{A_M\beta} \;=\; \boldsymbol{b_M}\,. \tag{6.4}$$

We remark that the following results are also valid for PLSR by setting $\boldsymbol{M} = \boldsymbol{I}_p$.

Definition 6.1. For every vector $\boldsymbol{c} \in \mathbb{R}^d$ and every matrix $\boldsymbol{C} \in \mathbb{R}^{d\times d}$, we call the set of vectors

$$\boldsymbol{c}, \boldsymbol{Cc}, \ldots, \boldsymbol{C}^{m-1}\boldsymbol{c}$$

the Krylov sequence of length m. The space spanned by this Krylov sequence is called the Krylov space of $\boldsymbol{C}$ and $\boldsymbol{c}$ and is denoted by $\mathcal{K}^{(m)}(\boldsymbol{C}, \boldsymbol{c})$.

Let us start with the following observation.

Lemma 6.2. *The space spanned by the weight vectors* $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_m$ *of penalized PLSR equals the Krylov space* $\mathcal{K}^{(m)}(\boldsymbol{A_M}, \boldsymbol{b_M})$. *The space spanned by the penalized PLSR components* $\boldsymbol{t}_1, \ldots, \boldsymbol{t}_m$ *equals the Krylov space* $\mathcal{K}^{(m)}(\boldsymbol{K_M}, \boldsymbol{K_M y})$.

This is the generalization of a result for PLSR. Recall that $\boldsymbol{K_M} = \boldsymbol{XMX}^t$.

Proof. This can be shown via induction. For $m = 1$, we know that $\boldsymbol{w}_1 = \boldsymbol{b_M}$ and $\boldsymbol{t}_1 = \boldsymbol{X b}_M = \boldsymbol{XMX}^t\boldsymbol{y} = \boldsymbol{K_M y}$. For a fixed $m > 1$, we conclude from the induction hypothesis and (4.12) that every vector $\boldsymbol{s}$ that lies in the span of $\boldsymbol{t}_1, \ldots, \boldsymbol{t}_m$ is of the form

$$\boldsymbol{s} \;=\; \boldsymbol{Xv} \quad , \quad \boldsymbol{v} \;\in\; \operatorname{span}\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_m\} = \mathcal{K}^{(m)}\left(\boldsymbol{A_M}, \boldsymbol{b_M}\right)\,.$$

We conclude that

$$\boldsymbol{w}_{m+1} = \boldsymbol{MX}_{m+1}^t\boldsymbol{y} = \boldsymbol{MX}^t\boldsymbol{y} - \boldsymbol{MX}^t\mathcal{P}_{\boldsymbol{t}_1,\ldots,\boldsymbol{t}_m}\boldsymbol{y} = \boldsymbol{Mb} - \boldsymbol{MX}^t\boldsymbol{Xs}\,.$$

This implies that

$$\boldsymbol{w}_{m+1} = \boldsymbol{b_M} - \boldsymbol{A_M s} \in \mathcal{K}^{(m+1)}\left(\boldsymbol{A_M}, \boldsymbol{b_M}\right)\,.$$

Note that

$$\mathcal{K}^{(m)}\left(\boldsymbol{K_M}, \boldsymbol{K_M y}\right) = \boldsymbol{X}\mathcal{K}^{(m)}\left(\boldsymbol{A_M}, \boldsymbol{b_M}\right) .$$

It follows that

$$\boldsymbol{t}_{m+1} = \boldsymbol{X}_{m+1}\boldsymbol{w}_{m+1} = \boldsymbol{X}\boldsymbol{w}_{m+1} - \underbrace{\mathcal{P}_{\boldsymbol{t}_1,\ldots \boldsymbol{t}_m}\boldsymbol{X}\boldsymbol{w}_{m+1}}_{\in\mathcal{K}^{(m)}(\boldsymbol{K_M},\boldsymbol{K_M y})} \in \mathcal{K}^{(m+1)}\left(\boldsymbol{K_M}, \boldsymbol{K_M y}\right) .$$

This concludes the proof. □

Corollary 6.3. *The penalized PLSR estimator obtained after m steps is equal to the solution of the constrained minimization problem*

$$\begin{array}{ll} arg\min\limits_{\boldsymbol{\beta}} & \|\boldsymbol{y} - \boldsymbol{X\beta}\|^2 \\ \textit{such that} & \boldsymbol{\beta} \in \mathcal{K}^{(m)}\left(\boldsymbol{A_M}, \boldsymbol{b_M}\right) \end{array}$$

Proof. This follows immediately from proposition 4.7. □

Corollary 6.4. *For univariate penalized PLSR, the components derived by the NIPALS algorithm are (up to the sign) equal to the components derived by SIMPLS defined in equations (4.5), (4.6) and (4.7).*

The equivalence of both methods for PLSR is shown in de Jong (1993).

Proof. We show via induction that the components $\boldsymbol{t}_1, \ldots, \boldsymbol{t}_m$ derived from SIMPLS span the space $\mathcal{K}^{(m)}\left(\boldsymbol{K_M}, \boldsymbol{K_M y}\right)$. For $m = 1$, this is obviously true, as $\boldsymbol{t}_1 = \boldsymbol{K_M y}$. The Lagrangian function associated to the optimization problem of SIMPLS is

$$L(\boldsymbol{w}) = \boldsymbol{w}^t\boldsymbol{b} - \lambda\left(\boldsymbol{w}^t\boldsymbol{w} + \boldsymbol{w}^t\boldsymbol{P}\boldsymbol{w} - 1\right) - \sum_{i=1}^{m}\mu_i\boldsymbol{w}^t\boldsymbol{X}^t\boldsymbol{t}_i .$$

Computing the first derivative, we obtain the equation

$$\boldsymbol{b} - 2\lambda\left(\boldsymbol{I}_p + \boldsymbol{P}\right)\boldsymbol{w}_{m+1} - \sum_{i=1}^{m}\mu_i\boldsymbol{X}^t\boldsymbol{t}_i = 0 .$$

This implies that

$$\boldsymbol{w}_{m+1} \sim \left(\boldsymbol{I}_p + \boldsymbol{P}\right)^{-1}\left(\boldsymbol{b} - \sum_{i=1}^{m}\mu_i\boldsymbol{X}^t\boldsymbol{t}_i\right) = \boldsymbol{b_M} - \sum_{i=1}^{m}\mu_i\boldsymbol{M}\boldsymbol{X}^t\boldsymbol{t}_i .$$

Using the induction hypothesis, we conclude that

$$\boldsymbol{t}_{m+1} \sim \boldsymbol{X}\boldsymbol{b}_{\boldsymbol{M}} - \sum_{i=1}^{m} \mu_i \boldsymbol{X}\boldsymbol{M}\boldsymbol{X}^t \boldsymbol{t}_i = \boldsymbol{K}_{\boldsymbol{M}}\boldsymbol{y} - \sum_{i=1}^{m} \mu_i \boldsymbol{K}_{\boldsymbol{M}} \boldsymbol{t}_i \in \mathcal{K}^{(m+1)}\left(\boldsymbol{K}_{\boldsymbol{M}}, \boldsymbol{K}_{\boldsymbol{M}}\boldsymbol{y}\right) .$$

□

Finally, we show the following result.

Proposition 6.5. *The weight vectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_m$ of penalized PLS are mutually $\boldsymbol{M}^{-1}$-orthogonal. The matrix $\boldsymbol{R} = \boldsymbol{T}^t \boldsymbol{X}\boldsymbol{W}$ is upper bidiagonal.*

Proof. It follows from the definition of the penalized PLS weight vectors and lemma 4.4 that for $i > j$

$$\langle \boldsymbol{w}_i, \boldsymbol{w}_j \rangle_{\boldsymbol{M}^{-1}} = \left\langle \boldsymbol{M}\boldsymbol{X}_i^t \boldsymbol{y}, \boldsymbol{w}_j \right\rangle_{\boldsymbol{M}^{-1}} = \boldsymbol{y}^t \boldsymbol{X}_i \boldsymbol{M}\boldsymbol{M}^{-1} \boldsymbol{w}_j = \boldsymbol{y}^t \boldsymbol{X}_i \boldsymbol{w}_j = \boldsymbol{y}^t \boldsymbol{0} = 0 .$$

Furthermore,

$$\boldsymbol{t}_i \in \mathcal{K}^{(m)}(\boldsymbol{K}_{\boldsymbol{M}}, \boldsymbol{K}_{\boldsymbol{M}}\boldsymbol{y}) = \boldsymbol{X}\mathcal{K}^{(m)}(\boldsymbol{A}_{\boldsymbol{M}}, \boldsymbol{b}_{\boldsymbol{M}}) = \boldsymbol{X}\text{span}\left(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_i\right) .$$

We can conclude that

$$\begin{aligned} \boldsymbol{M}\boldsymbol{X}^t \boldsymbol{t}_i &\in \boldsymbol{M}\boldsymbol{X}^t \boldsymbol{X}\mathcal{K}^{(i)}(\boldsymbol{A}_{\boldsymbol{M}}, \boldsymbol{b}_{\boldsymbol{M}}) \\ &= \boldsymbol{A}_{\boldsymbol{M}}\mathcal{K}^{(i)}(\boldsymbol{A}_{\boldsymbol{M}}, \boldsymbol{b}_{\boldsymbol{M}}) \\ &\subset \mathcal{K}^{(i+1)}(\boldsymbol{A}_{\boldsymbol{M}}, \boldsymbol{b}_{\boldsymbol{M}}) \\ &= \text{span}\left(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{i+1}\right) . \end{aligned}$$

In particular,

$$\boldsymbol{M}\boldsymbol{X}^t \boldsymbol{t}_i = \sum_{k=1}^{i+1} \alpha_k \boldsymbol{w}_k . \tag{6.5}$$

Now recall that the weight vectors are $\boldsymbol{M}^{-1}$-orthogonal. We conclude that for

$j > i+1$

$$\begin{aligned}
\boldsymbol{t}_i^t \boldsymbol{X} \boldsymbol{w}_j &= \langle \boldsymbol{M}\boldsymbol{X}^t \boldsymbol{t}_i, \boldsymbol{w}_j \rangle_{\boldsymbol{M}^{-1}} \\
&\overset{(6.5)}{=} \langle \sum_{k=1}^{i+1} \alpha_k \boldsymbol{w}_k, \boldsymbol{w}_j \rangle_{\boldsymbol{M}^{-1}} \\
&= \sum_{k=1}^{i+1} \alpha_k \langle \boldsymbol{w}_k, \boldsymbol{w}_j \rangle_{\boldsymbol{M}^{-1}} \\
&= 0
\end{aligned}$$

□

6.1 Preconditioned Conjugate Gradient Methods

We now present the conjugate gradient method for the preconditioned normal equation (6.4). The conjugate gradient method is normally applied if the involved matrix is symmetric. Note that in general, the matrix $\boldsymbol{A_M}$ is not symmetric with respect to the canonical inner product, but with respect to the inner product

$$\langle \boldsymbol{x}, \boldsymbol{x}' \rangle_{\boldsymbol{M}^{-1}} = \boldsymbol{x}^t \boldsymbol{M}^{-1} \boldsymbol{x}'$$

defined by $\boldsymbol{M}^{-1}$, as

$$\langle \boldsymbol{x}, \boldsymbol{A_M} x' \rangle_{\boldsymbol{M}^{-1}} = \boldsymbol{x}^t \boldsymbol{M}^{-1} \boldsymbol{M} \boldsymbol{A} \boldsymbol{x}' = \boldsymbol{x}^t \boldsymbol{A} \boldsymbol{x}' = \boldsymbol{x}^t \boldsymbol{A}^t \boldsymbol{M}^t \boldsymbol{M}^{-1} \boldsymbol{x}' = \langle \boldsymbol{A_M} \boldsymbol{x}, \boldsymbol{x}' \rangle_{\boldsymbol{M}^{-1}} .$$

We can rewrite the quadratic function ϕ defined in (6.3) as

$$\phi(\boldsymbol{\beta}) = \frac{1}{2} \langle \boldsymbol{\beta}, \boldsymbol{A_M} \boldsymbol{\beta} \rangle_{\boldsymbol{M}^{-1}} - \langle \boldsymbol{\beta}, \boldsymbol{b_M} \rangle_{\boldsymbol{M}^{-1}} .$$

We replace the canonical inner product by the inner product defined by $\boldsymbol{M}^{-1}$ and minimize this function iteratively along directions that are $\boldsymbol{A_M}$-orthogonal.

Definition 6.6. We say that two vectors $\boldsymbol{x}$ and $\boldsymbol{x}'$ are $\boldsymbol{A_M}$ orthogonal with respect to the inner product defined by $\boldsymbol{M}^{-1}$ if

$$\langle \boldsymbol{x}, \boldsymbol{A_M} x' \rangle_{\boldsymbol{M}^{-1}} = \boldsymbol{x}^t \boldsymbol{M}^{-1} \boldsymbol{A_M} \boldsymbol{x}' = \boldsymbol{x}^t \boldsymbol{A} \boldsymbol{x}' = 0 .$$

We start with an initial guess $\boldsymbol{\beta}_0 = \mathbf{0}$ and define

$$\boldsymbol{d}_0 = \boldsymbol{r}_0 = \boldsymbol{b}_{\boldsymbol{M}} - \boldsymbol{A}_{\boldsymbol{M}}\boldsymbol{\beta}_0 = \boldsymbol{b}_{\boldsymbol{M}}\,.$$

The quantity $\boldsymbol{d}_m$ is the search direction and $\boldsymbol{r}_m = \boldsymbol{b}_{\boldsymbol{M}} - \boldsymbol{A}_{\boldsymbol{M}}$ is the residual. For a given direction $\boldsymbol{d}_m$, we have to determine the optimal step size, that is we have to find

$$a_m = \arg\min_a \phi\left(\boldsymbol{\beta}_m + a\boldsymbol{d}_m\right)\,.$$

It is straightforward to check that

$$a_m = \frac{\langle \boldsymbol{d}_m, \boldsymbol{r}_m\rangle_{\boldsymbol{M}^{-1}}}{\langle \boldsymbol{d}_m, \boldsymbol{A}_{\boldsymbol{M}}\boldsymbol{d}_m\rangle_{\boldsymbol{M}^{-1}}}\,.$$

The new approximate solution is then

$$\boldsymbol{\beta}_{m+1} = \boldsymbol{\beta}_m + a_m\boldsymbol{d}_m\,.$$

After updating the residuals via

$$\boldsymbol{r}_{m+1} = \boldsymbol{b}_{\boldsymbol{M}} - \boldsymbol{A}_{\boldsymbol{M}}\boldsymbol{\beta}_{m+1},$$

we define a new search direction $\boldsymbol{d}_{m+1}$ that is $\boldsymbol{A}_M$-orthogonal to the previous search directions. This is ensured by projecting the residual $\boldsymbol{r}_m$ onto the space that is $\boldsymbol{A}_{\boldsymbol{M}}$-orthogonal to $\boldsymbol{d}_0, \ldots, \boldsymbol{d}_m$. We obtain

$$\boldsymbol{d}_{m+1} = \boldsymbol{r}_{m+1} - \sum_{i=0}^{m} \frac{\langle \boldsymbol{r}_{m+1}, \boldsymbol{A}_{\boldsymbol{M}}\boldsymbol{d}_i\rangle_{\boldsymbol{M}^{-1}}}{\langle \boldsymbol{d}_i, \boldsymbol{A}_{\boldsymbol{M}}\boldsymbol{d}_i\rangle_{\boldsymbol{M}^{-1}}}\boldsymbol{d}_i\,.$$

Algorithm 6.7 (Preconditioned conjugate gradient method)**.** *After setting* $\boldsymbol{\beta}_0 = \mathbf{0}$ *and* $\boldsymbol{d}_0 = \boldsymbol{r}_0 = \boldsymbol{b}_{\boldsymbol{M}}$ *, the approximate solutions* $\boldsymbol{\beta}_m$ *of the preconditioned linear system (6.4) are determined by iteratively computing*

$$\begin{array}{rcll}
\boldsymbol{\beta}_{m+1} & = & \boldsymbol{\beta}_m + \frac{\langle \boldsymbol{d}_m, \boldsymbol{r}_m\rangle_{\boldsymbol{M}^{-1}}}{\langle \boldsymbol{d}_m, \boldsymbol{A}_{\boldsymbol{M}}\boldsymbol{d}_m\rangle_{\boldsymbol{M}^{-1}}}\boldsymbol{d}_m & \textit{step size}\\
\boldsymbol{r}_{m+1} & = & \boldsymbol{b}_{\boldsymbol{M}} - \boldsymbol{A}_{\boldsymbol{M}}\boldsymbol{\beta}_{m+1} & \textit{residuals}\\
\boldsymbol{d}_{m+1} & = & \boldsymbol{r}_{m+1} - \frac{\langle \boldsymbol{r}_{m+1}, \boldsymbol{A}_{\boldsymbol{M}}\boldsymbol{d}_m\rangle_{\boldsymbol{M}^{-1}}}{\langle \boldsymbol{d}_m, \boldsymbol{A}_{\boldsymbol{M}}\boldsymbol{d}_m\rangle_{\boldsymbol{M}^{-1}}}\boldsymbol{d}_m & \textit{search direction}
\end{array}$$

In the rest of this section, we prove the following result.

Theorem 6.8. *The penalized PLSR algorithm is equivalent to the preconditioned conjugate gradient algorithm 6.7 for the preconditioned system (6.4).*

The following lemma follows almost immediately from the definition of the algorithms and can be proven via induction.

Lemma 6.9. *We have*

$$\mathit{span}\,\{\boldsymbol{d}_0,\ldots,\boldsymbol{d}_{m-1}\} = \mathit{span}\,\{\boldsymbol{r}_0,\ldots,\boldsymbol{r}_{m-1}\} = \mathit{span}\,\{\boldsymbol{\beta}_1,\ldots,\boldsymbol{\beta}_m\} = \mathcal{K}^{(m)}(\boldsymbol{A_M},\boldsymbol{b_M})\,.$$

Lemma 6.10. *We have*

$$\boldsymbol{\beta}_m = \sum_{i=0}^{m-1} \frac{\langle \boldsymbol{d}_i, \boldsymbol{b}_M\rangle_{\boldsymbol{M}^{-1}}}{\langle \boldsymbol{d}_i, \boldsymbol{A_M}\boldsymbol{d}_i\rangle_{\boldsymbol{M}^{-1}}}\boldsymbol{d}_i\,.$$

Proof. This corresponds to the iterative definition of $\boldsymbol{\beta}_m$. We only have to show that

$$\langle \boldsymbol{d}_i, \boldsymbol{r}_i\rangle_{\boldsymbol{M}^{-1}} \quad = \quad \langle \boldsymbol{d}_i, \boldsymbol{b}_M\rangle_{\boldsymbol{M}^{-1}}\,.$$

Note that

$$\boldsymbol{r}_i = \boldsymbol{b_M} - \sum_{j=0}^{i-1} a_j \boldsymbol{A}_M \boldsymbol{d}_j\,.$$

As – by definition – $\boldsymbol{d}_i$ is $\boldsymbol{A_M}$-orthogonal with respect to $\boldsymbol{M}^{-1}$ onto all directions $\boldsymbol{d}_j$, $j < i$, the proof is complete. □

Now we are able to proof the equivalence of penalized PLSR and the preconditioned conjugate gradient method.

Proof of theorem 6.8. As the search directions $\boldsymbol{d}_i$ span the Krylov space $\mathcal{K}^{(m)}(\boldsymbol{A_M},\boldsymbol{b_M})$ (see lemma 6.9), we can replace the matrix $\boldsymbol{W}$ in formula (4.14) of the penalized PLSR estimator by the matrix $\boldsymbol{D} = (\boldsymbol{d}_0,\ldots,\boldsymbol{d}_{m-1})$. As the search directions are $\boldsymbol{A}_M$-orthogonal, we have

$$\begin{aligned}
\widehat{\boldsymbol{\beta}}^{(m)} &= \boldsymbol{D}\left(\boldsymbol{D}^t\boldsymbol{A}\boldsymbol{D}\right)^{-1}\boldsymbol{D}^t\boldsymbol{b} \\
&= \boldsymbol{D}\left(\boldsymbol{D}^t\boldsymbol{M}^{-1}\boldsymbol{A}_M\boldsymbol{D}\right)^{-1}\boldsymbol{D}^t\boldsymbol{M}^{-1}\boldsymbol{b}_M \\
&= \sum_{i=0}^{m-1} \frac{\langle \boldsymbol{d}_i, \boldsymbol{b_M}\rangle_{\boldsymbol{M}^{-1}}}{\langle \boldsymbol{d}_i, \boldsymbol{A_M}\boldsymbol{d}_i\rangle_{\boldsymbol{M}^{-1}}}\boldsymbol{d}_i\,.
\end{aligned}$$

and this equals the formula in lemma 6.10. □

Corollary 6.11. *The length of the regression vector* $\widehat{\boldsymbol{\beta}}$ *of penalized PLSR is monotonically increasing, if we use the norm defined by* $\boldsymbol{M}^{-1}$

$$\|\widehat{\boldsymbol{\beta}}^{(1)}\|_{\boldsymbol{M}^{-1}} \leq \|\widehat{\boldsymbol{\beta}}^{(2)}\|_{\boldsymbol{M}^{-1}} \leq \ldots \leq \|\widehat{\boldsymbol{\beta}}^{(p)}\|_{\boldsymbol{M}^{-1}} .$$

Proof. In the special case of PLSR (that is $\boldsymbol{M} = \boldsymbol{I}_p$), this result is already known (de Jong 1995) and can be proven by using the equivalence of PLSR and the conjugate gradient method. Replacing the usual inner product by the inner product defined by $\boldsymbol{M}^{-1}$ in the proof in Phatak & de Hoog (2002), we obtain the result for penalized PLSR. As this general result is not needed in the rest of the work, we omit the details for the sake of briefness. □

6.2 Why Krylov Subspaces?

Krylov spaces are closely connected to the Lanczos algorithm (Lanczos 1950), a method for approximating eigenvalues or the generalized inverse of a symmetric matrix $\boldsymbol{A}$. We approximate the eigenvalues or the inverse of $\boldsymbol{A}$ by restricting the map that is defined by $\boldsymbol{A}$ onto a Krylov subspace $\mathcal{K}^{(m)}(\boldsymbol{A}, \boldsymbol{b})$ defined by $\boldsymbol{A}$ and a right-hand side $\boldsymbol{b}$. A priori, this does not make sense, as for a vector $\boldsymbol{v}$ that lies in $\mathcal{K}^{(m)}$, the vector $\boldsymbol{A}\boldsymbol{v}$ does not necessarily lie in $\mathcal{K}^{(m)}$. We therefore define the map $\boldsymbol{A}$ restricted to $\mathcal{K}^{(m)}$ in the following way. Let us assume that the columns of the matrix $\boldsymbol{W}$ form an orthonormal basis of $\mathcal{K}^{(m)}$. In particular, we can represent $\boldsymbol{v}$ as $\boldsymbol{v} = \boldsymbol{W}\boldsymbol{u}$. After applying the linear map $\boldsymbol{A}$ to $\boldsymbol{v}$, we project $\boldsymbol{A}\boldsymbol{v}$ onto $\mathcal{K}^{(m)}$. The projection onto $\mathcal{K}^{(m)}$ is $\boldsymbol{W}\boldsymbol{W}^t$, hence the image of $\boldsymbol{A}\boldsymbol{W}\boldsymbol{u}$ is

$$\boldsymbol{W}\boldsymbol{W}^t\boldsymbol{A}\boldsymbol{W}\boldsymbol{u} .$$

It follows that (in terms of the basis $\boldsymbol{W}$), the map $\boldsymbol{A}_{|\mathcal{K}^{(m)}}$ that is defined as

$$\mathcal{K}^{(m)}(\boldsymbol{A}, \boldsymbol{b}) \xrightarrow{\boldsymbol{A}} \mathcal{K}^{(m+1)}(\boldsymbol{A}, \boldsymbol{b}) \xrightarrow{\mathcal{P}_{\mathcal{K}^{(m)}}} \mathcal{K}^{(m)}(\boldsymbol{A}, \boldsymbol{b})$$

is

$$\boldsymbol{D}^{(m)} = \boldsymbol{W}^t\boldsymbol{A}\boldsymbol{W} \in \mathbb{R}^{m \times m} . \tag{6.6}$$

The eigenvectors and eigenvalues of $\boldsymbol{D}^{(m)}$ are called ritz pairs.

The inverse of $\boldsymbol{A}$ is then approximated by $\boldsymbol{W}\left(\boldsymbol{W}^t\boldsymbol{A}\boldsymbol{W}\right)^{-1}\boldsymbol{W}^t$. An approximate solution of the normal equations (6.2) is

$$\widehat{\boldsymbol{\beta}}^{(m)} = \boldsymbol{W}\left(\boldsymbol{W}^t\boldsymbol{A}\boldsymbol{W}\right)^{-1}\boldsymbol{W}^t\boldsymbol{b}\,.$$

Note that this equals the formula for the PLSR estimator presented in (4.14), as we showed in lemma 6.2 and proposition 4.5 that the PLS weight vectors $\boldsymbol{W}$ – if they are standardized to length 1 – form an orthonormal basis of the Krylov space $\mathcal{K}^{(m)}$.

In general, Krylov methods find approximate solutions of (6.2) by searching for solutions in Krylov spaces and we showed that the PLSR estimator is an approximate OLS solution in a Krylov subspace. Why is this a sensible idea? The following lemma shows that for a certain class of matrices $\boldsymbol{A}$, the vector $\boldsymbol{A}^-\boldsymbol{b}$ is an element of a Krylov space defined by $\boldsymbol{A}$ and $\boldsymbol{b}$. Hence there is a natural representation of $\boldsymbol{A}^-\boldsymbol{b}$ in terms of the Krylov sequence and we hope that we can find an approximate solution in a Krylov subspace of low dimension.

Proposition 6.12. *If $\boldsymbol{A} \in \mathbb{R}^{p\times p}$ is symmetric or if $\boldsymbol{A}$ is regular, there is a polynomial $\pi \in \mathbb{R}[X]$ of degree $\leq p-1$ such that*

$$\boldsymbol{A}^- = \pi(\boldsymbol{A})\,.$$

In particualar

$$\boldsymbol{A}^-\boldsymbol{b} \in \mathcal{K}^{(p)}(\boldsymbol{A},\boldsymbol{b})$$

for any vector $\boldsymbol{b}$.

For a general matrix $\boldsymbol{A}$, it can be shown (Ibsen & Meyer 1998) that the same statement is true if we replace the Moore-Penrose inverse by the Drazin inverse (Drazin 1968).

Proof. For a symmetric matrix $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^t$, let us define the polynomial π via the at most p equations

$$\pi(\lambda_i) = \begin{cases} \frac{1}{\lambda_i} & \lambda_i \neq 0 \\ 0 & \lambda_i = 0 \end{cases}.$$

In matrix notation, this equals $\pi(\boldsymbol{\Lambda}) = \boldsymbol{\Lambda}^-$. This polynomial corresponds to the definition of the Moore-Penrose inverse of a symmetric matrix that is presented in proposition A.13. It follows that

$$\pi(\boldsymbol{A}) = \boldsymbol{A}^- .$$

If $\boldsymbol{A}$ is regular, we can use the Theorem of Caley-Hamilton which states that

$$\chi_{\boldsymbol{A}}(\boldsymbol{A}) = \boldsymbol{0} .$$

Here,

$$\chi_{\boldsymbol{A}}(\lambda) = \sum_{i=0}^{p} c_i \lambda^i$$

is the characteristical polynomial of $\boldsymbol{A}$. As $\boldsymbol{A}$ is regular, we have $c_0 \neq 0$. From this we can conclude that

$$\boldsymbol{I}_p = \boldsymbol{A} \underbrace{\left(-\frac{1}{c_0} \sum_{i=0}^{p-1} c_{i+1} \boldsymbol{A}^i \right)}_{=\pi(\boldsymbol{A})} .$$

As the degree of π is $p-1$, the proof is complete. □

If we transfer this into the context of linear regression estimators, we obtain the following two corollaries.

Corollary 6.13. *After at most p iterations, the PLSR estimator equals the OLS estimator.*

This result is well-known and is usually proven using a geometric argument.

Proof. This follows immediately from corollary 6.3 which states that the PLSR estimator after m steps is equivalent to the OLS estimator under the additional constraint that is an element of the Krylov space $\mathcal{K}^{(m)}(\boldsymbol{A}, \boldsymbol{b})$. □

Corollary 6.14. *Suppose that $\boldsymbol{A} = \boldsymbol{X}^t\boldsymbol{X}$ is regular. After at most p iterations, the penalized PLSR estimator equals the OLS estimator.*

Proof. If $\boldsymbol{A}$ is invertible, it follows from proposition 6.12 that

$$\boldsymbol{A}_{\boldsymbol{M}}^{-1}\boldsymbol{b}_{\boldsymbol{M}} = \pi\left(\boldsymbol{A}_{\boldsymbol{M}}\right)\boldsymbol{b}_{\boldsymbol{M}} .$$

Furthermore,

$$\boldsymbol{A}_M^{-1}\boldsymbol{b}_M = (\boldsymbol{M}\boldsymbol{A})^{-1}\boldsymbol{M}\boldsymbol{b} = \boldsymbol{A}^{-1}\boldsymbol{M}^{-1}\boldsymbol{M}\boldsymbol{b} = \boldsymbol{A}^{-1}\boldsymbol{b} = \widehat{\boldsymbol{\beta}}_{OLS}\,.$$

We can conclude that the OLS estimator is an element of a Krylov space that is defined by $\boldsymbol{A}_M$ and $\boldsymbol{b}_M$. Now the statements follows readily from corollary 6.3. □

From now on, we only consider PLSR and abbreviate

$$\mathcal{K}^{(m)} \quad = \quad \mathcal{K}^{(m)}(\boldsymbol{A},\boldsymbol{b}),\,.$$

As already mentioned above, PLSR tries to find an approximation of $\widehat{\boldsymbol{\beta}}_{OLS}$ in a Krylov space of low dimension. The properties of Krylov spaces determine some of the statistical properties of the corresponding estimator. This is discussed in more detail in Chapter 7. There, we need a formula for the smallest dimension m^* such that the OLS estimator lies in the Krylov space $\mathcal{K}^{(m^*)}$. Recall the eigen decomposition

$$\boldsymbol{A} \quad = \quad \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^t$$

of $\boldsymbol{A}$ and set $\boldsymbol{s} = \boldsymbol{U}\boldsymbol{b}$. We define

$$\mathcal{M} \quad = \quad \{\lambda_i | s_i \neq 0\} \quad , \quad m^* \quad = \quad |\mathcal{M}|\,.$$

The quantity m^* is called the grade of $\boldsymbol{A}$ with respect to $\boldsymbol{b}$.

Proposition 6.15. *We have*

$$\dim \mathcal{K}^{(m)} \quad = \quad \begin{cases} m & m \leq m^* \\ m^* & m > m^* \end{cases}\,.$$

In particular

$$\dim \mathcal{K}^{(m^*)} = \dim \mathcal{K}^{(m*+1)} = \ldots = \dim \mathcal{K}^{(p)} = m^*\,. \tag{6.7}$$

Proof. This is a well-know result. We first show that $\dim \mathcal{K}^{(m^*)} = m^*$. Suppose that

$$\sum_{j=0}^{m^*-1} \gamma_j \boldsymbol{A}^j \boldsymbol{b} \;=\; \boldsymbol{0}$$

for some $\gamma_0, \ldots, \gamma_{m^*-1} \in \mathbb{R}$. Setting $f(\lambda) = \sum \gamma_i \lambda^i$ and using the eigen decomposition of $\boldsymbol{A}$, this equation is equivalent to

$$f\left(\boldsymbol{\Lambda}\right) \boldsymbol{s} \;=\; \boldsymbol{0}$$

It follows that $f(\lambda_i) s_i = 0$ for $i = 1, \ldots, p$. Hence, each element $\lambda_i \in \mathcal{M}$ is a zero of the polynomial $f(\lambda)$. This is a polynomial of degree $\leq m^* - 1$. As it has $m^* = |\mathcal{M}|$ different zeroes, it must be trivial, i.e. $\gamma_j = 0$.

Next, we show that if $m > m^*$, we have $\dim \mathcal{K}^{(m)} = m^*$. It is clear that $\dim \mathcal{K}^{(m)} \geq m^*$ as $\mathcal{K}^{(m^*)} \subset \mathcal{K}^{(m)}$. Let $\mathcal{S}$ be any set of $m^* + 1$ vectors in the Krylov sequence. Set

$$\mathcal{I} \;=\; \{i \in \{1, \ldots m\} | \boldsymbol{A}^{i-1}\boldsymbol{b} \in \mathcal{S}\}\,.$$

Hence $|\mathcal{I}| = m^* + 1$. The condition that $\mathcal{S}$ is linear dependent is equivalent to the following. There is a nontrivial polynomial

$$g(\lambda) \;=\; \sum_{i \in I} \gamma_i \lambda^i$$

such that $g\left(\lambda_i\right) = 0$ for $\lambda_i \in \mathcal{M}$. As the polynomial g is of degree $|\mathcal{I}| = m^* + 1$ and $|\mathcal{M}| = m^*$, there is always a nontrivial polynomial g that fulfills $g\left(\lambda_i\right) = 0$ for $\lambda_i \in \mathcal{M}$. $\square$

If $\boldsymbol{A}$ and $\boldsymbol{b}$ represent normal equations, we have $\boldsymbol{s} = \boldsymbol{\Sigma} \boldsymbol{V}^t \boldsymbol{y}$ and

$$\mathcal{M} \;=\; \{\lambda_i \neq 0 | \boldsymbol{v}_i^t \boldsymbol{y} \neq 0\}\,.$$

It follows from its definition that

$$m^* \;\leq\; \operatorname{rk}(\boldsymbol{A})\,.$$

The inequality is strict if $\boldsymbol{A}$ has non-zero eigenvalues of multiplicity > 1 or if there is a principal component $\boldsymbol{v}_i$ that is not correlated to $\boldsymbol{y}$, i.e. $\boldsymbol{v}_i^t \boldsymbol{y} = 0$.

6.3 A Polynomial Representation

In the last section, we showed that the PLSR estimator obtained after m steps lies in a Krylov subspace of dimension m. Hence there is a polynomial $\pi^{(m)}$ of degree $\leq m-1$ such that

$$\widehat{\boldsymbol{\beta}}^{(m)}_{PLS} = \pi^{(m)}(\boldsymbol{A})\,\boldsymbol{b}\,.$$

This polynomial depends on the matrix $\boldsymbol{D}^{(m)}$ defined in (6.6) and determines the shrinkage properties of PLSR that are illustrated in Chapter 7. In order to show that the degree of the polynomial is exactly $m-1$, we need to collect some well-known properties of the matrix $\boldsymbol{D}^{(m)}$.

Proposition 6.16. *The matrix $\boldsymbol{D}^{(m)}$ is symmetric and positive semidefinite. Furthermore, $\boldsymbol{D}^{(m)}$ is tridiagonal, that is $d_{ij} = 0$ for $|i-j| \geq 2$.*

Proof. The first two statements are obvious. Let $i \leq j-2$. As $\boldsymbol{w}_i \in \mathcal{K}^{(i)}$, the vector $\boldsymbol{A}\boldsymbol{w}_i$ lies in the subspace $\mathcal{K}^{(i+1)}$. As $j > i+1$, the vector $\boldsymbol{w}_j$ is orthogonal on $\mathcal{K}^{(i+1)}$, in other words $\langle \boldsymbol{w}_j, \boldsymbol{A}\boldsymbol{w}_i\rangle = 0$. As $\boldsymbol{D}^{(m)}$ is symmetric, we also have $\langle \boldsymbol{w}_i, \boldsymbol{A}\boldsymbol{w}_j\rangle = 0$ which proves the assertion. □

Definition 6.17. We say that a symmetric tridiagonal matrix $\boldsymbol{D}$ is unreduced if all subdiagonal entries are non-zero, i.e. $d_{i,i+1} \neq 0$ for all i.

Proposition 6.18. *If $\dim \mathcal{K}^{(m)} = m$, the matrix $\boldsymbol{D}^{(m)}$ is unreduced. More precisely, all subdiagonal elements are > 0.*

Proof. Set $\boldsymbol{p}_i = \boldsymbol{A}^{i-1}\boldsymbol{b}$ and denote by $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_m$ the basis of PLS weight vectors. Its existence is guaranteed as we assume that $\dim \mathcal{K}^{(m)} = m$. We have to show that the subdiagonal elements $\langle \boldsymbol{w}_i, \boldsymbol{A}\boldsymbol{w}_{i-1}\rangle$ are >0. As the length of $\boldsymbol{w}_i$ does not change the sign of this expression, we can assume that the vectors $\boldsymbol{w}_i$ are not normalized to have length 1. As the weight vectors lie in the Krylov space $\mathcal{K}^{(m)}$ and are mutually orthogonal (recall proposition 4.5), we conclude that the weight vectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_m$ are equal to the Gram-Schmidt basis obtained from $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m$. This implies

$$\boldsymbol{w}_i = \boldsymbol{p}_i - \sum_{k=1}^{i-1} \frac{\langle \boldsymbol{p}_i, \boldsymbol{w}_k\rangle}{\langle \boldsymbol{w}_k, \boldsymbol{w}_k\rangle}\boldsymbol{w}_k\,. \tag{6.8}$$

As the vectors $\boldsymbol{w}_i$ are pairwise orthogonal, it follows that

$$\langle \boldsymbol{w}_i, \boldsymbol{p}_i\rangle = \langle \boldsymbol{p}_i, \boldsymbol{p}_i\rangle > 0\,. \tag{6.9}$$

We conclude that

$$\begin{aligned}
\langle \boldsymbol{w}_i, \boldsymbol{A}\boldsymbol{w}_{i-1}\rangle &\overset{(6.8)}{=} \left\langle \boldsymbol{w}_i, \boldsymbol{A}\left(\boldsymbol{p}_{i-1} - \sum_{k-1}^{i-2} \frac{\langle \boldsymbol{p}_{i-1}, \boldsymbol{w}_k\rangle}{\langle \boldsymbol{w}_k, \boldsymbol{w}_k\rangle}\boldsymbol{w}_k\right)\right\rangle \\
&\overset{\boldsymbol{A}\boldsymbol{p}_{i-1}=\boldsymbol{p}_i}{=} \langle \boldsymbol{w}_i, \boldsymbol{p}_i\rangle - \sum_{k=1}^{i-2} \frac{\langle \boldsymbol{p}_{i-1}, \boldsymbol{w}_k\rangle}{\langle \boldsymbol{w}_k, \boldsymbol{w}_k\rangle}\langle \boldsymbol{w}_i, \boldsymbol{A}\boldsymbol{w}_k\rangle \\
&\overset{\text{prop.6.16}}{=} \langle \boldsymbol{w}_i, \boldsymbol{p}_i\rangle \\
&\overset{(6.9)}{=} \langle \boldsymbol{p}_i, \boldsymbol{p}_i\rangle > 0\,.
\end{aligned}$$

□

Proposition 6.19. *The eigenvalues of an unreduced matrix are distinct.*

The proof van be found in **?**.

Remark 6.20. All eigenvalues of $\boldsymbol{D}^{(m^*)}$ are eigenvalues of $\boldsymbol{A}$.

Proof. Recall that $\boldsymbol{D}^{(m^*)}$ represents the map

$$\boldsymbol{A}_{|\mathcal{K}^{(m^*)}} \quad : \quad \mathcal{K}^{(m^*)} \xrightarrow{\boldsymbol{A}} \mathcal{K}^{(m^*+1)} = \mathcal{K}^{(m^*)}\,.$$

with respect to the basis $\boldsymbol{W}^{(m^*)}$. As any eigenvalue of $\boldsymbol{A}_{|\mathcal{K}^{(m^*)}}$ is obviously an eigenvalue of $\boldsymbol{A}$, the proof is complete. □

Proposition 6.21. *If* $\dim \mathcal{K}^{(m)} = m$*, we have* $\det\left(\boldsymbol{D}^{(m)}\right) \neq 0$.

Proof. Let us start with the remark that the characteristical polynomials $\chi^{(m)}$ of $\boldsymbol{D}^{(m)}$ are related in the following way,

$$\chi^{(m)}(\lambda) \quad = \quad (d_{m,m} - \lambda)\,\chi^{(m-1)}(\lambda) - d_{m-1,m}^2\chi^{(m-2)}(\lambda)\,. \tag{6.10}$$

Here, $d_{i,j}$ is the (i,j) entry of the matrix $\boldsymbol{D}^{(m)}$. Furthermore, both $\chi^{(m-1)}$ and $\chi^{(m-2)}$ are polynomials with non-negative zero's. The two polynomials must have a different sign on $\{x \leq 0\}$. Suppose that m is the smallest number such that $\boldsymbol{D}^{(m)}$ is not regular. This implies that 0 is an eigenvalue of $\boldsymbol{D}^{(m)}$. Plugging $\lambda = 0$ into (6.10), we obtain

$$d_{m,m}\chi^{(m-1)}(0) \quad = \quad d_{m-1,m}^2\chi^{(m-2)}(0)\,.$$

As the signs of $\chi^{(m-1)}(0)$ and $\chi^{(m-2)}(0)$ cannot be equal and $\chi^{(m-1)}(0) \neq 0$, this is only possible if $d_{m-1,m} = 0$. Now recall proposition 6.18 which implies that $d_{m-1,m} = 0$ is only possible if $m > m^*$. □

Denote the m different eigenvalues of $\boldsymbol{D}^{(m)}$ by

$$\mu_1^{(m)} > \ldots > \mu_m^{(m)} > 0\,. \tag{6.11}$$

These eigenvalues are called ritz values. Set

$$f^{(m)}(\lambda) \;=\; 1 - \prod_{i=1}^{m} \left(1 - \frac{\lambda}{\mu_i^{(m)}}\right)\,. \tag{6.12}$$

This is a polynomial of degree m. As $f^{(m)} = 0$, there is a polynomial $\pi^{(m)}$ of degree $m-1$ such that

$$f^{(m)}(\lambda) \;=\; \lambda \pi^{(m)}(\lambda)\,. \tag{6.13}$$

Proposition 6.22 ((Phatak & de Hoog 2002))**.** *Suppose that $m \leq m^*$. We have*

$$\widehat{\boldsymbol{\beta}}_{PLS}^{(m)} \;=\; \pi^{(m)}(\boldsymbol{A})\boldsymbol{b}\,,$$

with the polynomial $\pi^{(m)}$ defined in (6.13).

Proof. The polynomial $\pi^{(m)}$ is simply the polynomial representation of the inverse of $\boldsymbol{D}^{(m)}$

$$\left(\boldsymbol{D}^{(m)}\right)^{-1} \;=\; \pi^{(m)}\left(\boldsymbol{D}^{(m)}\right)\,.$$

This follows from (6.13). We plug this into the formula of proposition 4.7 and obtain

$$\widehat{\boldsymbol{\beta}}_{PLS}^{(m)} \;=\; \boldsymbol{W}^{(m)} \pi^{(m)} \left(\left(\boldsymbol{W}^{(m)}\right)^t \boldsymbol{A} \boldsymbol{W}^{(m)}\right) \left(\boldsymbol{W}^{(m)}\right)^t \boldsymbol{b}\,.$$

Recall that the columns of $\boldsymbol{W}^{(m)}$ form an orthonormal basis of $\mathcal{K}^{(m)}$. It follows that $\boldsymbol{W}^{(m)}\left(\boldsymbol{W}^{(m)}\right)^t$ is the operator that projects on the space $\mathcal{K}^{(m)}$. In particular

$$\boldsymbol{W}^{(m)}\left(\boldsymbol{W}^{(m)}\right)^t \boldsymbol{A}^j \boldsymbol{b} = \boldsymbol{A}^j \boldsymbol{b}$$

for $j = 1, \ldots, m-1$. This implies that

$$\widehat{\boldsymbol{\beta}}_{PLS}^{(m)} \;=\; \pi^{(m)}(\boldsymbol{A})\boldsymbol{b}\,.$$

□

Chapter 7

Shrinkage Properties of Partial Least Squares

In this chapter, we study the shrinkage properties of PLS regression. It is well known (Frank & Friedman 1993) that we can express the PLSR estimator obtained after m steps in the following way:

$$\widehat{\boldsymbol{\beta}}^{(m)}_{PLS} = \sum_{i=1}^{\mathrm{rk}(\boldsymbol{X})} f^{(m)}(\lambda_i)\boldsymbol{z}_i \,,$$

where $\boldsymbol{z}_i$ is the component of the Ordinary Least Squares (OLS) estimator along the ith principal component of the covariance matrix $\boldsymbol{X}^t\boldsymbol{X}$ and λ_i is the corresponding eigenvalue. The quantities $f^{(m)}(\lambda_i)$ are called shrinkage factors. We show that these factors are determined by the tridiagonal matrix $D^{(m)}$ defined in (6.6). Combining the results of Butler & Denham (2000) and Phatak & de Hoog (2002), we give a simpler and clearer proof of the shape of the shrinkage factors of PLSR and derive some of their properties. In particular, we reproduce the fact that some of the values $f^{(m)}(\lambda_i)$ are greater than 1. This was first proved by Butler & Denham (2000).

We argue that these "peculiar shrinkage properties" (Butler & Denham 2000) do not necessarily imply that the Mean Squared Error (MSE) of the PLSR estimator is worse compared to the MSE of the OLS estimator. In the case of deterministic shrinkage factors, i.e. factors that do not depend on the response $\boldsymbol{y}$, any value $\left|f^{(m)}(\lambda_i)\right| > 1$ is of course undesirable. But in the case of PLSR, the shrinkage factors are stochastic – they also depend on $\boldsymbol{y}$. In particular, bounding the absolute value of the shrinkage factor by 1 might not automatically yield a lower MSE, in disagreement to what is conjectured e.g. in Frank & Friedman (1993).

Having issued this warning, we explore whether bounding the shrinkage factors leads to a lower MSE or not. It is very difficult to derive theoretical results, as the quantities of interest - $\widehat{\boldsymbol{\beta}}^{(m)}_{PLS}$ and $f^{(m)}(\lambda_i)$ respectively - depend on $\boldsymbol{y}$ in a complicated, nonlinear way. As a substitute, we study the problem on several artificial data sets and one real world example. It turns out that in most cases, the MSE of the bounded version of PLSR is indeed smaller than the one of PLSR.

During the rest of the chapter we occasionally make the assumption that

$$\dim \mathcal{K}^{(m)} = m\,. \tag{7.1}$$

The maximal number for which this holds is m^* (see proposition 6.15). Note however that

$$\mathcal{K}^{(m^*-1)} \subset \mathcal{K}^{(m^*)} = \mathcal{K}^{(m^*+1)} = \ldots = \mathcal{K}^{(p)}$$

(see (6.7)) and the PLSR solutions do not change anymore.

7.1 What is Shrinkage?

We have presented two estimators for the regression parameter $\boldsymbol{\beta}$ – OLS and (penalized) PLSR – which also define estimators for $\boldsymbol{X\beta}$ via

$$\widehat{\boldsymbol{y}} = \boldsymbol{X}\widehat{\boldsymbol{\beta}}\,.$$

One possibility to evaluate the quality of an estimator is to determine its Mean Squared Error (MSE). In general, the MSE of an estimator $\widehat{\boldsymbol{\theta}}$ for a vector-valued parameter $\boldsymbol{\theta}$ is defined as

$$\begin{aligned}
\text{MSE}\left(\widehat{\boldsymbol{\theta}}\right) &= E\left[\text{trace}\left(\widehat{\boldsymbol{\theta}}-\boldsymbol{\theta}\right)\left(\widehat{\boldsymbol{\theta}}-\boldsymbol{\theta}\right)^t\right] \\
&= E\left[\left(\widehat{\boldsymbol{\theta}}-\boldsymbol{\theta}\right)^t\left(\widehat{\boldsymbol{\theta}}-\boldsymbol{\theta}\right)\right] \\
&= \left(E\left[\widehat{\boldsymbol{\theta}}\right]-\boldsymbol{\theta}\right)^t\left(E\left[\widehat{\boldsymbol{\theta}}\right]-\boldsymbol{\theta}\right) + E\left[\left(\widehat{\boldsymbol{\theta}}^t - E\left[\widehat{\boldsymbol{\theta}}\right]\right)^t\left(\widehat{\boldsymbol{\theta}}^t - E\left[\widehat{\boldsymbol{\theta}}\right]\right)\right].
\end{aligned}$$

This is the well-known bias-variance decomposition of the MSE. The first part is the squared bias and the second part is the variance term.

We start by investigating the class of linear estimators, i.e. estimators that are of the form $\widehat{\boldsymbol{\theta}} = \boldsymbol{H}\boldsymbol{y}$ for a matrix $\boldsymbol{H}$ that does not depend on $\boldsymbol{y}$. It follows immediately from the regression model (1.9) and (1.7) that for a linear estimator,

$$E\left[\widehat{\boldsymbol{\theta}}\right] = \boldsymbol{H}\boldsymbol{X}\boldsymbol{\beta}, \quad \mathrm{Var}\left[\widehat{\boldsymbol{\theta}}\right] = \sigma^2 \mathrm{trace}\left(\boldsymbol{H}\boldsymbol{H}^t\right) .$$

The OLS estimators are linear, as

$$\widehat{\boldsymbol{\beta}}_{OLS} = \left(\boldsymbol{X}^t\boldsymbol{X}\right)^- \boldsymbol{X}^t\boldsymbol{y} \quad , \quad \widehat{\boldsymbol{y}}_{OLS} = \boldsymbol{X}\left(\boldsymbol{X}^t\boldsymbol{X}\right)^- \boldsymbol{X}^t\boldsymbol{y} .$$

Note that the estimator of $\widehat{\boldsymbol{y}}_{OLS}$ is simply the projection $\mathcal{P}_{\boldsymbol{X}}$ onto the space that is spanned by the columns of $\boldsymbol{X}$. The estimator $\widehat{\boldsymbol{y}}_{OLS}$ is unbiased as

$$E\left[\widehat{\boldsymbol{y}}_{OLS}\right] = \mathcal{P}_{\boldsymbol{X}}\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{X}\boldsymbol{\beta} .$$

The estimator $\widehat{\boldsymbol{\beta}}_{OLS}$ is only unbiased if $\boldsymbol{\beta} \in \mathrm{range}\left(\boldsymbol{X}^t\boldsymbol{X}\right)^-$.

$$E\left[\widehat{\boldsymbol{\beta}}_{OLS}\right] = E\left[\left(\boldsymbol{X}^t\boldsymbol{X}\right)^- \boldsymbol{X}^t\boldsymbol{y}\right] = \left(\boldsymbol{X}^t\boldsymbol{X}\right)^- \boldsymbol{X}^t E\left[\boldsymbol{y}\right] = \left(\boldsymbol{X}^t\boldsymbol{X}\right)^- \boldsymbol{X}^t\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{\beta} .$$

Let us now have a closer look at the variance term. It follows directly from $\mathrm{trace}(\mathcal{P}_{\boldsymbol{X}}\mathcal{P}_{\boldsymbol{X}}^t) = \mathrm{rk}(\boldsymbol{X})$ that

$$\mathrm{Var}\left(\widehat{\boldsymbol{y}}_{OLS}\right) = \sigma^2 \mathrm{rk}(\boldsymbol{X}) .$$

For $\widehat{\boldsymbol{\beta}}_{OLS}$ we have

$$\left(\boldsymbol{X}^t\boldsymbol{X}\right)^- \boldsymbol{X}^t \left(\left(\boldsymbol{X}^t\boldsymbol{X}\right)^- \boldsymbol{X}^t\right)^t = \left(\boldsymbol{X}^t\boldsymbol{X}\right)^- \boldsymbol{X}^t\boldsymbol{X}\left(\boldsymbol{X}^t\boldsymbol{X}\right)^- = \left(\boldsymbol{X}^t\boldsymbol{X}\right)^- = \boldsymbol{U}\boldsymbol{\Lambda}^-\boldsymbol{U}^t ,$$

hence

$$\mathrm{Var}\left(\widehat{\boldsymbol{\beta}}_{OLS}\right) = \sigma^2 \sum_{i=1}^{\mathrm{rk}(\boldsymbol{X})} \frac{1}{\lambda_i} . \tag{7.2}$$

We conclude that the MSE of the estimator $\widehat{\boldsymbol{\beta}}_{OLS}$ depends on the non-zero eigenvalues of $\boldsymbol{A} = \boldsymbol{X}^t\boldsymbol{X}$. Small eigenvalues of $\boldsymbol{A}$ correspond to directions in $\boldsymbol{X}$ that have a low variance. Equation (7.2) shows that if some eigenvalues are small, the variance of $\widehat{\boldsymbol{\beta}}_{OLS}$ is very high, which leads to a high MSE.

One possibility to (hopefully) decrease the MSE is to modify the OLS estimator by shrinking the directions of the OLS estimator that are responsible for a high variance. This of course introduces bias. We shrink the OLS estimator in the hope that the increase in bias is small compared to the decrease in variance.

In general, a shrinkage estimator for $\boldsymbol{\beta}$ is of the form

$$\widehat{\boldsymbol{\beta}}_{shr} = \sum_{i=1}^{\mathrm{rk}(\boldsymbol{X})} f(\lambda_i)\boldsymbol{z}_i \,,$$

where f is some real-valued function. The values $f(\lambda_i)$ are called shrinkage factors. Examples are Principal Component Regression

$$f(\lambda_i) = \begin{cases} 1 & \text{ith principal component included} \\ 0 & \text{otherwise} \end{cases}$$

and Ridge Regression

$$f(\lambda_i) = \frac{\lambda_i}{\lambda_i + \lambda} \,,$$

with $\lambda > 0$ the Ridge parameter. We illustrate in Section 7.2 that PLSR is a shrinkage estimator as well. It turns out that the shrinkage behavior of PLSR regression is rather complicated.

Let us investigate in which way the MSE of the estimator is influenced by the shrinkage factors. If the shrinkage estimators are linear, i.e. the shrinkage factors do not depend on $\boldsymbol{y}$, this is an easy task. Let us first write the shrinkage estimator in matrix notation. We have

$$\widehat{\boldsymbol{\beta}}_{shr} = \boldsymbol{H}_{shr}\boldsymbol{y} = \boldsymbol{U}\boldsymbol{\Sigma}^{-}\boldsymbol{D}_{shr}\boldsymbol{V}^t\boldsymbol{y} \,.$$

The diagonal matrix $\boldsymbol{D}_{shr}$ has entries $f(\lambda_i)$. The shrinkage estimator for $\boldsymbol{y}$ is

$$\widehat{\boldsymbol{y}}_{shr} = \boldsymbol{X}\boldsymbol{H}_{shr}\boldsymbol{y} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-}\boldsymbol{D}_{shr}\boldsymbol{V}^t\boldsymbol{y} \,.$$

We calculate the variance of these estimators.

$$\begin{aligned}
\operatorname{trace}\left(\boldsymbol{H}_{shr}\boldsymbol{H}_{shr}^t\right) &= \operatorname{trace}\left(\boldsymbol{U}\boldsymbol{\Sigma}^-\boldsymbol{D}_{shr}\boldsymbol{\Sigma}^-\boldsymbol{D}_{shr}\boldsymbol{U}^t\right)\\
&= \operatorname{trace}\left(\boldsymbol{\Sigma}^-\boldsymbol{D}_{shr}\boldsymbol{\Sigma}^-\boldsymbol{D}_{shr}\right)\\
&= \sum_{i=1}^{\mathrm{rk}(\boldsymbol{X})}\frac{\left(f\left(\lambda_i\right)\right)^2}{\lambda_i}
\end{aligned}$$

and

$$\begin{aligned}
\operatorname{trace}\left(\boldsymbol{X}\boldsymbol{H}_{shr}\boldsymbol{H}_{shr}^t\boldsymbol{X}^t\right) &= \operatorname{trace}\left(\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^-\boldsymbol{D}_{shr}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^-\boldsymbol{D}_{shr}\boldsymbol{V}^t\right)\\
&= \operatorname{trace}\left(\boldsymbol{\Sigma}\boldsymbol{\Sigma}^-\boldsymbol{D}_{shr}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^-\boldsymbol{D}_{shr}\right)\\
&= \sum_{i=1}^{\mathrm{rk}(\boldsymbol{X})}\left(f\left(\lambda_i\right)\right)^2 .
\end{aligned}$$

Next, we calculate the bias of the two shrinkage estimators. We have

$$E\left[\boldsymbol{H}_{shr}\boldsymbol{y}\right] = \boldsymbol{H}_{shr}\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{D}_{shr}\boldsymbol{\Sigma}^-\boldsymbol{U}^t\boldsymbol{\beta}\,.$$

It follows that

$$\begin{aligned}
\operatorname{bias}^2\left(\widehat{\boldsymbol{\beta}}_{shr}\right) &= \left(E\left[\boldsymbol{H}_{shr}\boldsymbol{y}\right]-\boldsymbol{\beta}\right)^t\left(E\left[\boldsymbol{H}_{shr}\boldsymbol{y}\right]-\boldsymbol{\beta}\right)\\
&= \left(\boldsymbol{U}^t\boldsymbol{\beta}\right)^t\left(\boldsymbol{\Sigma}\boldsymbol{D}_{shr}\boldsymbol{\Sigma}^- - \boldsymbol{I}_p\right)^t\left(\boldsymbol{\Sigma}D_{shr}\boldsymbol{\Sigma}^- - \boldsymbol{I}_p\right)\left(\boldsymbol{U}^t\boldsymbol{\beta}\right)\\
&= \sum_{i=1}^{\mathrm{rk}(\boldsymbol{X})}\left(f(\lambda_i)-1\right)^2\left(\boldsymbol{u}_i^t\boldsymbol{\beta}\right)^2 .
\end{aligned}$$

Replacing $\boldsymbol{H}_{shr}$ by $\boldsymbol{X}\boldsymbol{H}_{shr}$ it is easy to show that

$$\operatorname{bias}^2\left(\widehat{\boldsymbol{y}}_{shr}\right) = \sum_{i=1}^{p}\lambda_i\left(f(\lambda_i)-1\right)^2\left(\boldsymbol{u}_i^t\boldsymbol{\beta}\right)^2 .$$

Proposition 7.1. *For the shrinkage estimator* $\widehat{\boldsymbol{\beta}}_{shr}$ *and* $\widehat{\boldsymbol{y}}_{shr}$ *defined above we have*

$$\begin{aligned}
MSE\left(\widehat{\boldsymbol{\beta}}_{shr}\right) &= \sum_{i=1}^{rk(\boldsymbol{X})}\left(f(\lambda_i)-1\right)^2\left(\boldsymbol{u}_i^t\boldsymbol{\beta}\right)^2+\sigma^2\sum_{i=1}^{rk(\boldsymbol{X})}\frac{\left(f\left(\lambda_i\right)\right)^2}{\lambda_i},\\
MSE\left(\widehat{\boldsymbol{y}}_{shr}\right) &= \sum_{i=1}^{rk(\boldsymbol{X})}\lambda_i\left(f(\lambda_i)-1\right)^2\left(\boldsymbol{u}_i^t\boldsymbol{\beta}\right)^2+\sigma^2\sum_{i=1}^{rk(\boldsymbol{X})}\left(f\left(\lambda_i\right)\right)^2 .
\end{aligned}$$

If the shrinkage factors are deterministic, i.e. they do not depend on $\boldsymbol{y}$, any value $f(\lambda_i) \neq 1$ increases the bias. Values $|f(\lambda_i)| < 1$ decrease the variance, whereas values $|f(\lambda_i)| > 1$ increase the variance. Hence an absolute value > 1 is always undesirable. The situation might be different for stochastic shrinkage factors. We discuss this in the following section.

Note that there is a different notion of shrinkage, namely that the L_2- norm of an estimator is smaller than the L_2-norm of the OLS estimator. Why is this a desirable property? Let us again consider the case of linear estimators. Set $\widehat{\boldsymbol{\theta}}_i = \boldsymbol{H}_i \boldsymbol{y}$ for $i = 1, 2$. We have

$$\left\|\widehat{\boldsymbol{\theta}}_i\right\|_2^2 = \boldsymbol{y}^t \boldsymbol{H}_i^t \boldsymbol{H}_i \boldsymbol{y}\,.$$

The property that for all $\boldsymbol{y} \in \mathbb{R}^n$

$$\left\|\widehat{\boldsymbol{\theta}}_1\right\|_2 \leq \left\|\widehat{\boldsymbol{\theta}}_2\right\|_2$$

is equivalent to the condition that $\boldsymbol{H}_1^t \boldsymbol{H}_1 - \boldsymbol{H}_2^t \boldsymbol{H}_2$ is negative semidefinite. The trace of negative semidefinite matrices is ≤ 0. Furthermore trace$\,(\boldsymbol{H}_i^t \boldsymbol{H}_i) =$ trace$\,(\boldsymbol{H}_i \boldsymbol{H}_i^t)$, so we conclude that

$$\operatorname{Var}\left(\widehat{\boldsymbol{\theta}}_1\right) \leq \operatorname{Var}\left(\widehat{\boldsymbol{\theta}}_2\right)\,.$$

We already remarked in Chapter 6 that

$$\|\widehat{\boldsymbol{\beta}}_{PLS}^{(1)}\|_2 \leq \|\widehat{\boldsymbol{\beta}}_{PLS}^{(2)}\|_2 \leq \ldots \leq \|\widehat{\boldsymbol{\beta}}_{PLS}^{(m^*)}\|_2 = \|\widehat{\boldsymbol{\beta}}_{OLS}\|_2\,.$$

7.2 The Shrinkage Factors of Partial Least Squares

In this section, we give a simpler and clearer proof of the shape of the shrinkage factors of PLSR. Basically, we combine the results of Butler & Denham (2000) and Phatak & de Hoog (2002). In proposition 6.22, we derived a formula for the PLSR estimator in terms of the ritz values of $\boldsymbol{A}$. From this, we can immediately conclude the following corollary.

Corollary 7.2 ((Phatak & de Hoog 2002)). *Suppose that* $\dim \mathcal{K}^{(m)} = m$. *If we* *denote by*

$$\boldsymbol{z}_i = \frac{\boldsymbol{v}_i^t \boldsymbol{y}}{\sqrt{\lambda_i}} \boldsymbol{u}_i$$

the component of $\widehat{\boldsymbol{\beta}}_{OLS}$ *along the ith eigenvector of* $\boldsymbol{A}$*, then*

$$\widehat{\boldsymbol{\beta}}_{PLS}^{(m)} = \sum_{i=1}^{rk(\boldsymbol{X})} f^{(m)}(\lambda_i) \boldsymbol{z}_i \,,$$

with $f^{(m)}(\lambda)$ *is defined in (6.13).*

Proof. This follows immediately from proposition 6.22. We have

$$\begin{aligned}
\widehat{\boldsymbol{\beta}}_{PLS}^{(m)} &= \pi^{(m)}(\boldsymbol{A})\boldsymbol{b} \\
&= \boldsymbol{U}\pi^{(m)}(\boldsymbol{\Lambda})\boldsymbol{\Sigma}\boldsymbol{V}^t\boldsymbol{y} \\
&= \sum_{i=1}^{\mathrm{rk}(\boldsymbol{X})} \pi^{(m)}(\lambda_i)\sqrt{\lambda_i}\boldsymbol{v}_i^t\boldsymbol{y}\boldsymbol{u}_i \\
&= \sum_{i=1}^{\mathrm{rk}(\boldsymbol{X})} \pi^{(m)}(\lambda_i)\lambda_i \frac{1}{\sqrt{\lambda_i}}\boldsymbol{v}_i^t\boldsymbol{y}\boldsymbol{u}_i \\
&\overset{(6.13)}{=} \sum_{i=1}^{\mathrm{rk}(\boldsymbol{X})} f^{(m)}(\lambda_i)\boldsymbol{z}_i \,.
\end{aligned}$$

□

The following theorem is a special form of the Cauchy Interlace Theorem. In this version, we use a general result from Parlett (1998) and exploit the tridiagonal structure of $\boldsymbol{D}^{(m)}$.

Theorem 7.3. *Each interval*

$$\left[\mu_{m-j}^{(m)}, \mu_{m-(j+1)}^{(m)}\right]$$

$(j = 0, \ldots, m-2)$ *contains an eigenvalue of* $\boldsymbol{D}^{(m+k)}$) $(k \geq 1)$*. In addition, there is an eigenvalue of* $\boldsymbol{D}^{(m+k)}$ *outside the open interval* $(\mu_m^{(m)}, \mu_1^{(m)})$.

This theorems ensures in particular that there is an eigenvalue of $\boldsymbol{A}$ in the interval $\left[\mu_k^{(m)}, \mu_{k-1}^{(m)}\right]$. Theorem 7.3 holds independently of assumption (7.1).

Proof. By definition, for $k \geq 1$

$$\boldsymbol{D}^{(m+k)} = \begin{pmatrix} \boldsymbol{D}^{(m-1)} & \bullet^t & 0 \\ \bullet & * & * \\ 0 & * & * \end{pmatrix} .$$

Here $\bullet = (*, \ldots, 0, 0)$, so

$$\boldsymbol{D}^{(m)} = \begin{pmatrix} \boldsymbol{D}^{(m-1)} & \bullet^t \\ \bullet & a_m \end{pmatrix} .$$

An application of theorem 10.4.1 in Parlett (1998) gives the desired result. $\square$

We now show that some of the shrinkage factors of PLSR are $\neq 1$.

Theorem 7.4 ((Butler & Denham 2000)). *For each $m \leq m^*$, we can decompose the interval $[\lambda_p, \lambda_i]$ into $m+1$ disjoint intervals*[1]

$$I_1 \leq I_2 \leq \ldots \leq I_{m+1}$$

such that

$$f^{(m)}(\lambda_i) \begin{cases} \leq 1 & \lambda_i \in I_j \text{ and } j \text{ odd} \\ \geq 1 & \lambda_i \in I_j \text{ and } j \text{ even} \end{cases} .$$

Proof. Set

$$g^{(m)}(\lambda) = 1 - f^{(m)}(\lambda) .$$

It follows from (6.12) that the zero's of $g^{(m)}(\lambda)$ are $\mu_m^{(m)}, \ldots, \mu_1^{(m)}$. As $\boldsymbol{D}^{(m)}$ is unreduced, all eigenvalues are distinct. Set $\mu_0^{(m)} = \lambda_1$ and $\mu_{m+1}^{(m)} = \lambda_p$. Define

$$I_j =]\mu_i^{(m)}, \mu_{i+1}^{(m)}[\text{ for } j = 0, \ldots, m .$$

By definition, $g^{(m)}(0) = 1$. Hence $g^{(m)}(\lambda)$ is non-negative on the intervals I_j if j is odd and $g^{(m)}$ is non-positive on the intervals I_j if j is even. It follows from theorem 7.3 that all intervals I_j contain at least one eigenvalue λ_i of $\boldsymbol{A}$. $\square$

In general, we cannot conclude that $f^{(m)}(\lambda_i) \neq 1$ for all λ_i and $m = 1, \ldots, m^*$. However, in practical applications, the shrinkage factors seem to be $\neq 1$ all of

[1] We say that $I_j \leq I_k$ if $\sup I_j \leq \inf I_k$.

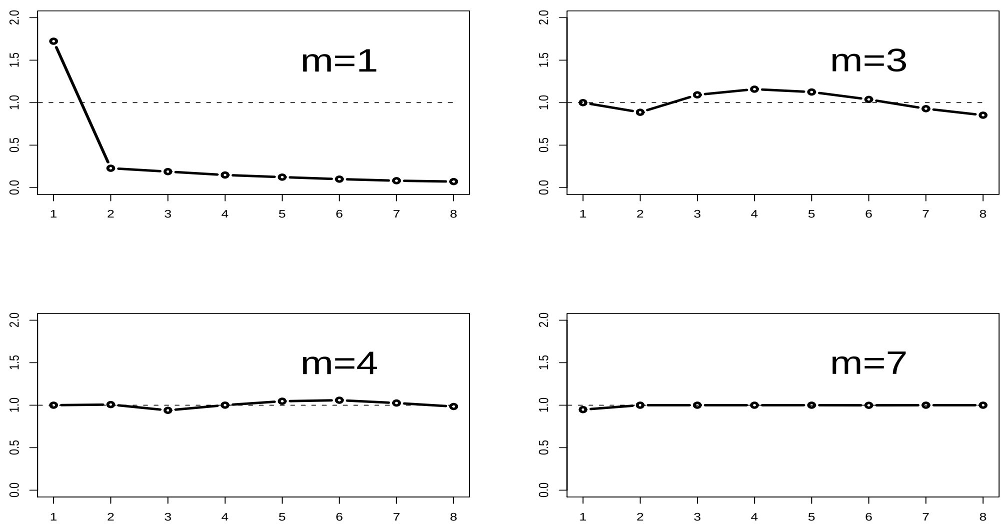

Figure 7.1: An illustration of the shrinkage behavior of PLSR. The number of variables is $p = 8$. The eigenvalues of $\boldsymbol{X}^t\boldsymbol{X}$ are enumerated in decreasing order, $\lambda_1 \geq \lambda_2 \geq \ldots$. The shrinkage factors $f^{(m)}(\lambda_i)$ are plotted as a function of i for different values of m. The amount of absolute shrinkage $\left|1 - f^{(m)}(\lambda_i)\right|$ is particularly prominent if m is small.

the time. Figure 7.1 illustrates the shrinkage behavior of PLSR. This example is taken from Butler & Denham (2000). Using some of the results of Section 6.3 and the fact that the eigenvalues of $\boldsymbol{D}^{(m^*-1)}$ and $\boldsymbol{D}^{(m^*)}$ are distinct (Parlett 1998), we can deduce that some factors are indeed $\neq 1$. Details can be found in Butler & Denham (2000) and Krämer (2006). Furthermore, using theorem 7.3, more precisely $\lambda_p \leq \mu_i^{(m)} \leq \lambda_i$, it is possible to bound the terms

$$1 - \frac{\lambda_i}{\mu_i^{(m)}}\,.$$

Based on these bounds, it is possible to derive bounds on the shrinkage factors. We will not pursue this further, readers who are interested in the bounds should consult Lingjaerde & Christopherson (2000). Instead, we have a closer look at the MSE of PLSR.

In Section 7.1, we showed that a value $|f^{(m)}(\lambda_i)| > 1$ is not desirable, as both the bias and the variance of the estimator increases. Note however that in the case of PLSR, the factors $f^{(m)}(\lambda_i)$ are stochastic; they depend on $\boldsymbol{y}$ – in a nonlinear way. The variance of the PLS estimator for the ith principal component is

$$\text{Var}\left(f^{(m)}(\lambda_i)\frac{(\boldsymbol{v}_i)^t\,\boldsymbol{y}}{\sqrt{\lambda_i}}\right),$$

with both $f^{(m)}(\lambda_i)$ and $\frac{\boldsymbol{v}_i^t\boldsymbol{y}}{\sqrt{\lambda_i}}$ depending on $\boldsymbol{y}$.

Among others, Frank & Friedman (1993) propose to truncate the shrinkage factors of the PLSR estimator in the following way. Set

$$\widetilde{f}^{(m)}(\lambda_i) \;=\; \begin{cases} +1 & f^{(m)}(\lambda_i) > +1 \\ -1 & f^{(m)}(\lambda_i) < -1 \\ f^{(m)}(\lambda_i) & \text{otherwise} \end{cases},$$

and define a new estimator

$$\widehat{\boldsymbol{\beta}}^{(m)}_{TRN} \;:=\; \sum_{i=1}^{\text{rk}(\boldsymbol{X})} \widetilde{f}^{(m)}(\lambda_i)\boldsymbol{z}_i\,. \tag{7.3}$$

If the shrinkage factors are numbers, this surely improves the MSE (as shown in Section 7.1). But in the case of stochastic shrinkage factors, the situation might be different. Let us suppose for a moment that $f^{(m)}(\lambda_i) = \frac{\sqrt{\lambda_i}}{\boldsymbol{v}_i^t\boldsymbol{y}}$. It follows that

$$0 = \text{Var}\left(f^{(m)}(\lambda_i)\frac{\boldsymbol{v}_i^t\boldsymbol{y}}{\sqrt{\lambda_i}}\right) \leq \text{Var}\left(\widetilde{f}^{(m)}(\lambda_i)\frac{\boldsymbol{v}_i^t\boldsymbol{y}}{\sqrt{\lambda_i}}\right),$$

so it is not clear whether the truncated estimator TRN leads to a lower MSE, which is conjectured in e.g. Frank & Friedman (1993).

The assumption that $f^{(m)}(\lambda_i) = \frac{\sqrt{\lambda_i}}{\boldsymbol{v}_i^t y}$ is of course purely hypothetical. It is not clear whether the shrinkage factors behave this way. It is hard if not infeasible to derive statistical properties of the PLSR estimator or its shrinkage factors, as they depend on $\boldsymbol{y}$ in a complicated, nonlinear way. As an alternative, we compare the two different estimators on different data sets.

7.3 Simulations

In this section, we explore the difference between the methods PLSR and truncated PLSR (TRN). We investigate several artificial data sets, and in the next section, we consider a real world example.

We compare the MSE of the two methods - PLSR and truncated PLSR - on 27 different artificial data sets. We use a setting similar to the one in Frank & Friedman (1993). For each data set, the number of examples is $n = 50$. We consider three different number of predictor variables:

$$p = 5, 40, 100\,.$$

The input data $\boldsymbol{X}$ is chosen according to a multivariate normal distribution with zero mean and covariance matrix $\boldsymbol{C}$. We consider three different covariance matrices:

$$\begin{aligned} \boldsymbol{C}_1 &= \boldsymbol{I}_p\,, \\ (\boldsymbol{C}_2)_{ij} &= \frac{1}{|i-j|+1}\,, \\ (\boldsymbol{C}_3)_{ij} &= \begin{cases} 1 & , \quad i = j \\ 0.7 & , \quad i \neq j \end{cases}\,. \end{aligned}$$

The matrices $\boldsymbol{C}_1$, $\boldsymbol{C}_2$ and $\boldsymbol{C}_3$ correspond to no, moderate and high collinearity respectively. The regression vector $\boldsymbol{\beta}$ is a randomly chosen vector $\boldsymbol{\beta} \in \{0,1\}^p$. In addition, we consider three different signal-to-noise ratios:

$$\text{stnr} = \sqrt{\frac{\text{var}\,(\boldsymbol{X}\boldsymbol{\beta})}{\sigma^2}} = 1, 3, 7\,.$$

We yield $3 \times 3 \times 3 = 27$ different parameter settings. For each setting, we estimate the MSE of the two methods: For $k = 1, \ldots, K = 200$ we generate $\boldsymbol{y}$ according to (1.9) and (1.7). We determine for each method and each m the respective estimator $\widehat{\boldsymbol{\beta}}_k^{(m)}$ and define

$$\widehat{MSE}\left(\widehat{\boldsymbol{\beta}}^{(m)}\right) = \frac{1}{K}\sum_{k=1}^{K}\left(\widehat{\boldsymbol{\beta}}_k^{(m)} - \boldsymbol{\beta}\right)^t \left(\widehat{\boldsymbol{\beta}}_k^{(m)} - \boldsymbol{\beta}\right)\,.$$

If there are more predictor variables than examples, this approach is not sensible, as the true regression vector $\boldsymbol{\beta}$ is not identifiable. This implies that different regressions vectors $\boldsymbol{\beta}_1 \neq \boldsymbol{\beta}_2$ can lead to $\boldsymbol{X}\boldsymbol{\beta}_1 = \boldsymbol{X}\boldsymbol{\beta}_2$. Hence for $p = 100$, we only estimate the MSE of $\widehat{\boldsymbol{y}}$ for the two methods. We display the estimated MSE of

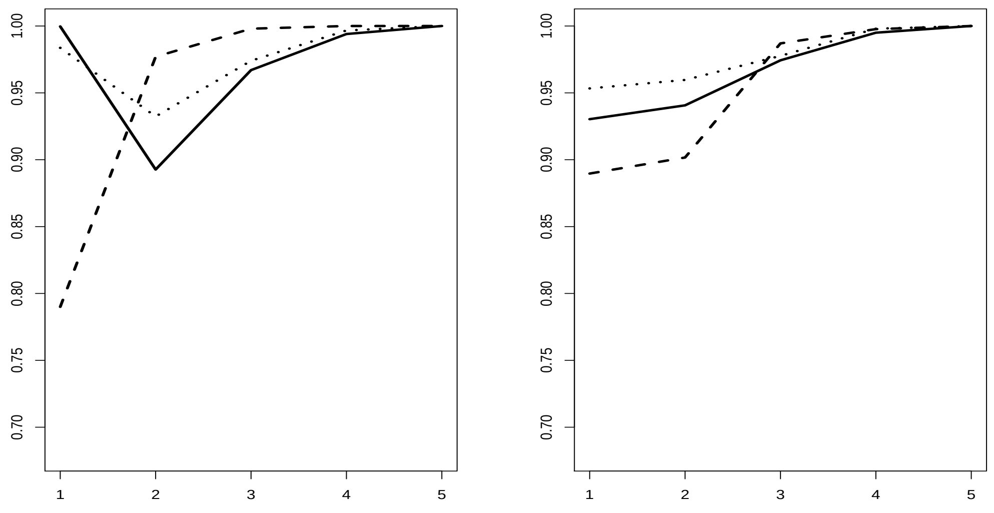

Figure 7.2: MSE-RATIO for $p = 5$. The figures show the averaged MSE-RATIO for different parameter settings. Left: Comparison for high (straight line), moderate (dotted line) and no (dashed line) collinearity. Right: Comparison for stnr 1 (straight line), 3 (dotted line) and 7 (dashed line).

the method TRN as a fraction of the estimated MSE of the method PLSR, i.e. for each m we display

$$MSE - RATIO \quad = \quad \frac{\widehat{MSE}\left(\widehat{\boldsymbol{\beta}}^{(m)}_{TRN}\right)}{\widehat{MSE}\left(\widehat{\boldsymbol{\beta}}^{(m)}_{PLS}\right)} .$$

As already mentioned, we display the MSE-RATIO for $\widehat{\boldsymbol{y}}$ in the case $p = 100$. The results are displayed in Figures 7.2, 7.3 and 7.4. In order to have a compact representation, we consider the averaged MSE-RATIOS for different parameter settings. For example, we fix a degree of collinearity (say high collinearity) and display the averaged MSE-RATIO over the three different signal-to-noise ratios. The results for all 27 data sets are shown in the tables in the appendix.

There are several observations. The MSE of truncated PLSR is lower almost all of the times. The decrease of the MSE is particularly large if the number of components m is small, but > 1 . For larger m, the difference decreases. This

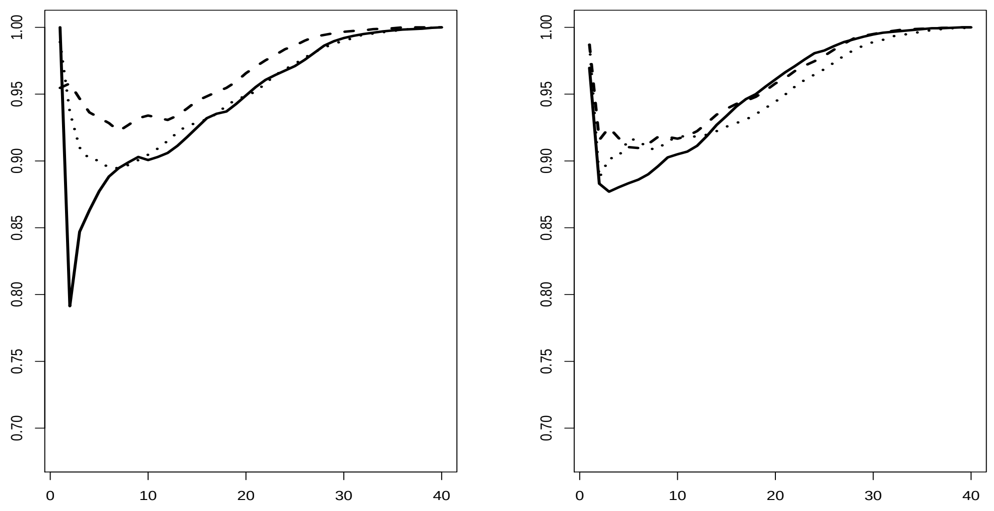

Figure 7.3: MSE-RATIO for $p = 40$.

is not surprising, as for large m, the difference between the PLSR estimator and the OLS estimator decreases. Hence we expect the difference between TRN and PLSR to become smaller. The reduction of the MSE is particularly prominent in complex situations, i.e. in situations with high collinearity in $\boldsymbol{X}$ or with a low signal-to-noise-ratio.

Another feature, which cannot be deduced from Figures 7.2, 7.3 and 7.4 but from the tables in the appendix, is the fact that the optimal number of components

$$\begin{aligned} m_{PLS}^{opt} &= \operatorname{argmin} \widehat{MSE}\left(\widehat{\boldsymbol{\beta}}_{PLS}^{(m)}\right) \\ m_{TRN}^{opt} &= \operatorname{argmin} \widehat{MSE}\left(\widehat{\boldsymbol{\beta}}_{TRN}^{(m)}\right) \end{aligned}$$

are equal almost all of the times. This is also true if we consider the MSE of $\widehat{\boldsymbol{y}}$. We can benefit from this if we want to select an optimal model for truncated PLSR. We return to this subject in Section 7.5.

7.4 Example: Tecator Data Set

In this example, we consider near infrared spectra (NIR) of $n = 171$ meat samples that are measured at $p = 100$ different wavelengths from 850 – 1050 nm. This data set is taken from the StatLib datasets archive and can be downloaded from

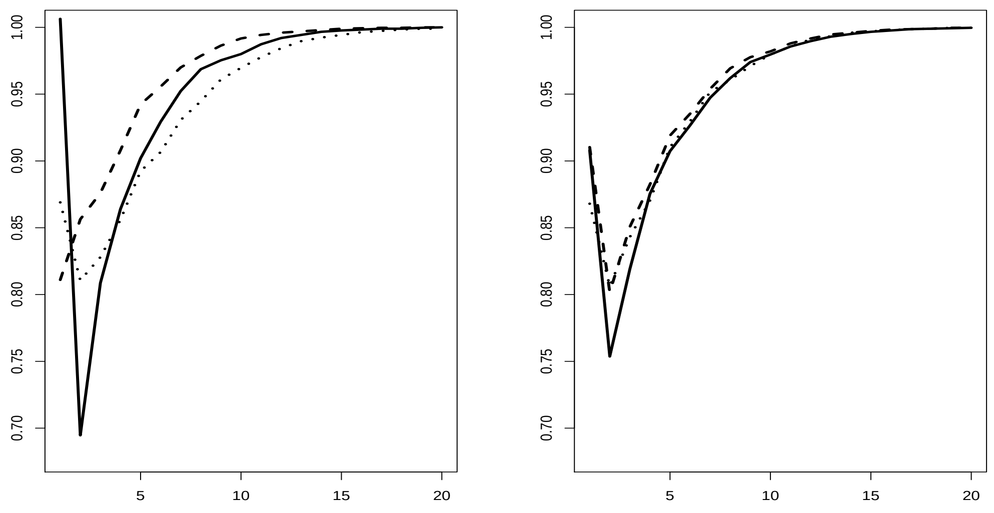

Figure 7.4: MSE-RATIO for $p = 100$. In this case, we display the MSE-RATIO for $\widehat{\boldsymbol{y}}$ instead of $\widehat{\boldsymbol{\beta}}$. Only the first 20 components are displayed.

`http://lib.stat.cmu.edu/datasets/tecator`. The task is to predict the fat content of a meat sample on the basis of its NIR spectrum. We choose this data set as PLSR is widely used in the chemometrics field. In this type of applications, we usually observe a lot of predictor variables which are highly correlated. We estimate the MSE of the two methods PLSR and truncated PLSR by computing the 10fold cross-validated error of the two estimators. The results are displayed in Figure 7.5.

Again, truncated PLSR is better almost all of the times, although the difference is small. Note furthermore that the optimal number of components are almost identical for the two methods: We have $m_{PLS}^{opt} = 15$ and $m_{TRN}^{opt} = 16$.

7.5 Conclusion

We illustrated in Section 7.3 that bounding the absolute value of the PLSR shrinkage factors by one seems to improve the MSE of the estimator. So should we now discard PLSR and always use truncated PLSR instead? There might be (at least) two objections. Firstly, it would be very lightheaded if we relied on results of a small-scale simulation study. Secondly, TRN is computationally more extensive than PLSR. We need the full singular value decomposition of $\boldsymbol{X}$. In each step,

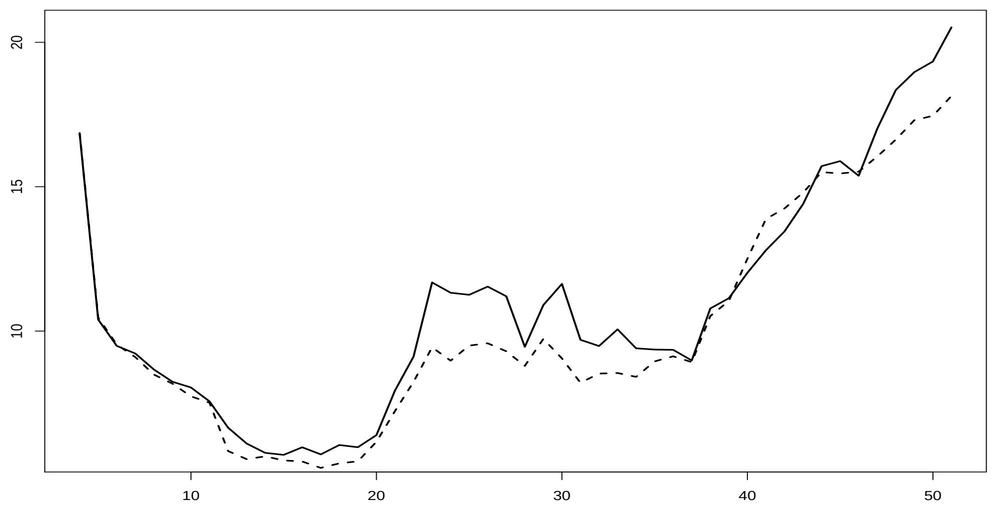

Figure 7.5: 10fold cross-validated test error for the Tecator data set. The straight line corresponds to PLSR, the dashed line corresponds to truncated PLSR.

we have to compute the PLSR estimator and adjust its shrinkage factors "by hand". However, the experiments suggest that it can be worthwhile to compare PLSR and truncated PLSR. We pointed out in Section 7.3 that the two methods do not seem to differ much in terms of optimal number of components. In order to reduce the computational costs of truncated PLSR, we therefore suggest the following strategy. We first compute the optimal PLSR model on a training set and choose the optimal model with the help of a model selection criterion. In a second step, we truncate the shrinkage factors of the optimal model. We then use a validation set in order to quantify the difference between PLSR and TRN and choose the method with the lower validation error.

Chapter 8

Functional Data Analysis

In the preceding chapters, we introduced Partial Least Squares for Regression and two of its variants – penalized PLSR and truncated PLSR. We investigated their mathematical and statistical properties. In particular, we measured their performance in terms of a low test error and a mean squared error respectively. One important feature of a fitting method – apart from its predictive power - is its interpretability. This aspect has been neglected so far and we illustrate that the understandability of PLSR can be limited. In the remaining two chapters, we propose a different approach that is based on Boosting methods and that exploits the special type of data that is common in a lot of PLSR applications: The data that we observe are functions.

8.1 Example: Biscuit Dough Data Set

The following example is described in detail in Osborne et al. (1994) and is also used in Brown et al. (2001). The data can be downloaded from `http://www.stat.tamu.edu/~mvannucci/webpages/codes.html`. The task is to predict with high accuracy the amount of fat in biscuit dough. As the direct measurement of fat is costly and time-consuming, we use NIR (near infra red) spectroscopy instead. For each of the $n = 39$ training examples of biscuit dough, the amount of fat and the reflectance of NIR light for different wavelengths is measured. In this example, $p = 700$ equidistant wavelengths in the range from 1100 to 2398 nanometers are used. For each example, we obtain a function of the reflectance, which is called a spectrum. The 39 spectra are depicted in Figure 8.1. The task is to predict the amount of fat of a new sample after observing its spectrum.

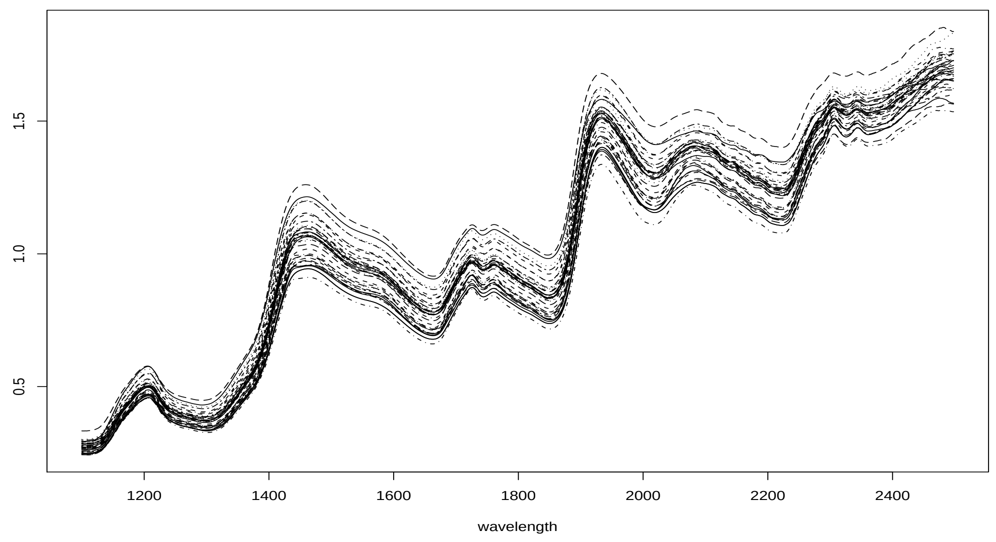

Figure 8.1: The $n = 39$ NIR spectra of the biscuit dough.

We already mentioned that a very popular method in the chemometrics field is PLSR. Let us investigate its performance on this particular data set. We estimate the optimal number of latent components using the leave-one-out error. The risk of this model is estimated on a test set that consists of 31 examples. The minimal leave-one-out error is obtained with $m_{opt} = 13$. A widely used diagnostic tool is the plot of the standardized regression coefficients as a function of the wavelength. That is, for each of the 700 variables X_i, we divide the estimated regression coefficient $\widehat{\beta}_i$ by the standard deviation of X_i and plot these values into a coordinate system. This is done in Figure 8.2. One desirable feature is the detection of regions of relevant wavelengths. In this particular example, the regression coefficients are however very hard to interpret. We cannot detect any type of pattern, as the regression coefficients look rather like white noise. Therefore, a chemometrician is willing to sacrifice some predictive power in order to have an interpretable model. But even the standardized regression coefficients of a model with $m = 3$ latent components do not disclose any valuable information.

Note that each example $\boldsymbol{x}_i$, which consists of $p = 700$ measurements, is in fact a discretized curve. Instead of regarding these data points $\boldsymbol{x}_i$ as vectors in a high-dimensional space $\mathbb{R}^p$, we can also view them as elements of a space of real-valued functions.

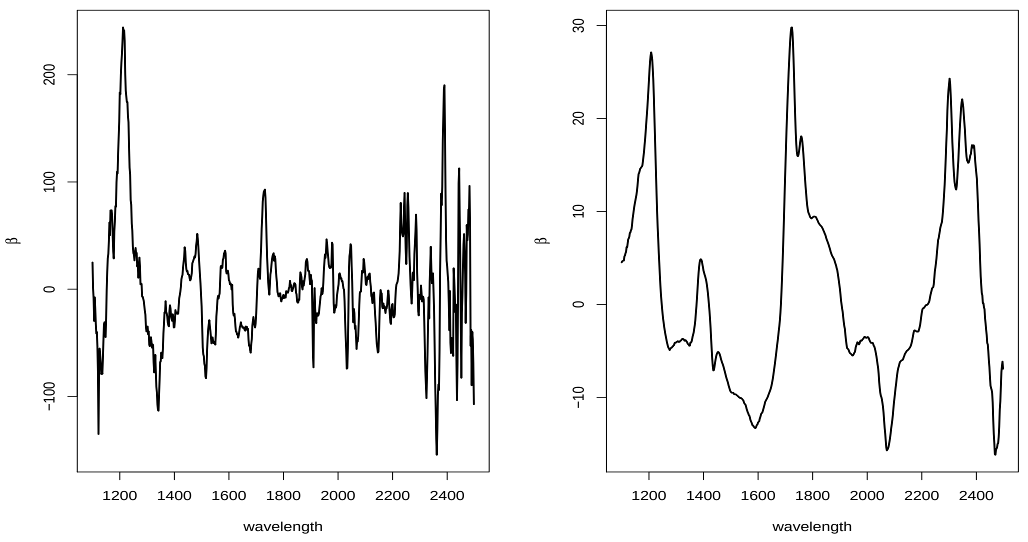

Figure 8.2: Standardized PLSR regression coefficients as a function of the wavelength. Left: PLSR model with 13 components. Right: PLSR model with 3 components.

8.2 Example: Speech Recognition Data Set

This example is taken from Biau et al. (2005). The data consists of 48 recordings of the word "Yes" and 52 recordings of the word "No". One recording is represented by a discretized time series of length 8192. The data can be downloaded from `http://www.math.univ-montp2.fr/~biau/bbwdata.tgz`. Two recordings are displayed in Figure 8.3. The task is to find a classification rule that assigns the correct word to each time series. At first glance, the classification problem seems complicated, due to the high amount of variables compared to the number of examples. If we regard one recording as one function rather than measurements of 8192 variables, we can however exploit the special structure of this problem. In this particular example, it is possible to describe the important information that is encoded in the functions in terms of their oscillations. The classification rule can then be learned using these extracted features.

Functional data analysis deals with learning from data that are curves. In the following two section, which are condensed from Ramsay & Silverman (2005), we present its main ideas.

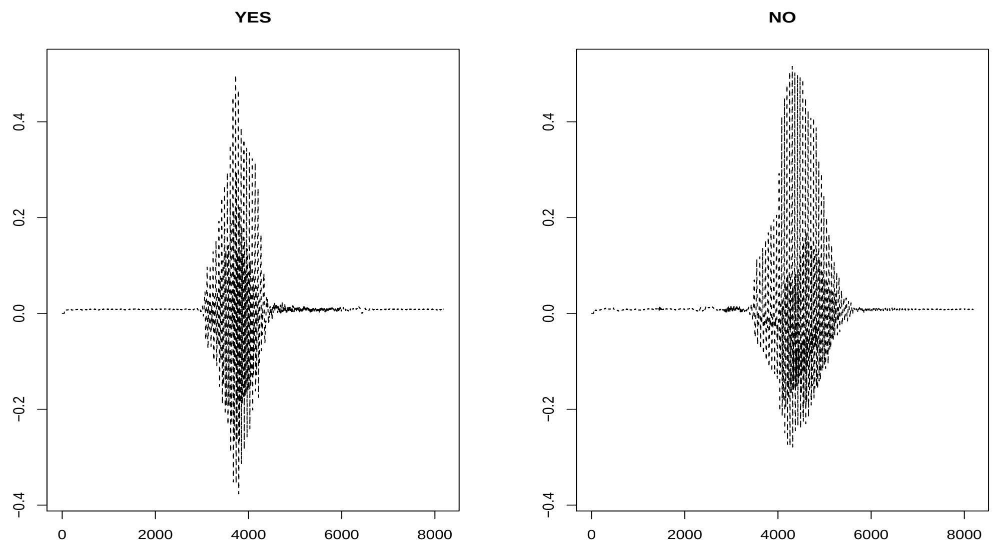

Figure 8.3: Two examples of the speech recognition data. Left: One recording of the word "Yes". Right: One recording of the word "No".

8.3 From Observations to Functions

We speak of functional data if the variables that we observe are curves. Let us first consider the case that only the predictor samples x_i are curves, that is

$$x_i \in \mathcal{X} = \{x : T \to \mathbb{R}\}.$$

We usually assume that the functions fulfill a regularity condition, and in the rest of the chapter, we consider the Hilbert space $\mathcal{X} = L^2(T)$ of all square-integrable functions $T \to \mathbb{R}$. In our examples, T is a subset of $\mathbb{R}$.

In most applications, we do not measure a curve $x_i(t)$, but discrete values

$$\boldsymbol{x}_i \quad = \quad (x_i(t_1), \ldots, x_i(t_p))^t$$

of a curve. An important step in the analysis of functional data is therefore the transformation of the discretized objects to smooth functions. The general approach is the following. We represent each example as a linear combination

$$x_i(t) \quad = \quad \sum_{l=1}^{K_x} c_{il}\phi_l(t) \tag{8.1}$$

of a set of basis functions $\phi_1, \ldots, \phi_{K_x}$. Each example $x_i(t)$ is represented by its vector

$$\boldsymbol{c}_i \quad = \quad (c_{i1}, \ldots, c_{iK_x}) \in \mathbb{R}^{Kx}$$

of coefficients. This coefficient vector is estimated by applying the (regularized) empirical risk minimization principle described in Section 1.2. If we use the quadratic loss function and represent the values of the base functions ϕ_l at measurement points t_j by

$$\boldsymbol{\Phi} \quad = \quad (\phi_l(t_j)) \in \mathbb{R}^{p \times K_x} \, ,$$

the penalized least squares criterion is

$$\arg\min_{\boldsymbol{c}_i} \quad \left\{ \|\boldsymbol{x}_i - \boldsymbol{\Phi}\boldsymbol{c}_i\|^2 + r(\boldsymbol{c}_i) \right\} \, .$$

Here, r is a regularization term that e.g. controls the smoothness of the function $x_i(t)$. Regularization might be necessary if the measurements of $x_i(t)$ are noisy or if the points t_j are not equidistant. There are some natural candidates for the set of basis functions. We introduced the widely-used B-splines in Chapter 5. If the data have a periodic structure, this can be reflected efficiently with an expansion in terms of Fourier functions. They are defined as

$$\begin{aligned} \phi_{2i-1}(t) &= \sin(i\omega t) \, , \\ \phi_{2i}(t) &= \cos(i\omega t) \, , \end{aligned}$$

and are illustrated in Figure 8.4. Other basis functions are polynomials and – more importantly - wavelets. Wavelets are families of orthogonal basis functions. These bases are generated by choosing a so-called mother-wavelet Φ and and then computing all translations and dilatations

$$\phi_{jk}(x) \quad = \quad 2^{\frac{j}{2}} \Phi\left(2^j x - k\right) \, .$$

The mother wavelet is chosen in a way that all functions are mutually orthogonal. In most applications, the mother wavelet has a compact support. An example for a mother wavelet is the Haar wavelet. Some translations and dilatations of this wavelet are shown in Figure 8.5. Wavelets are able to represent a function f

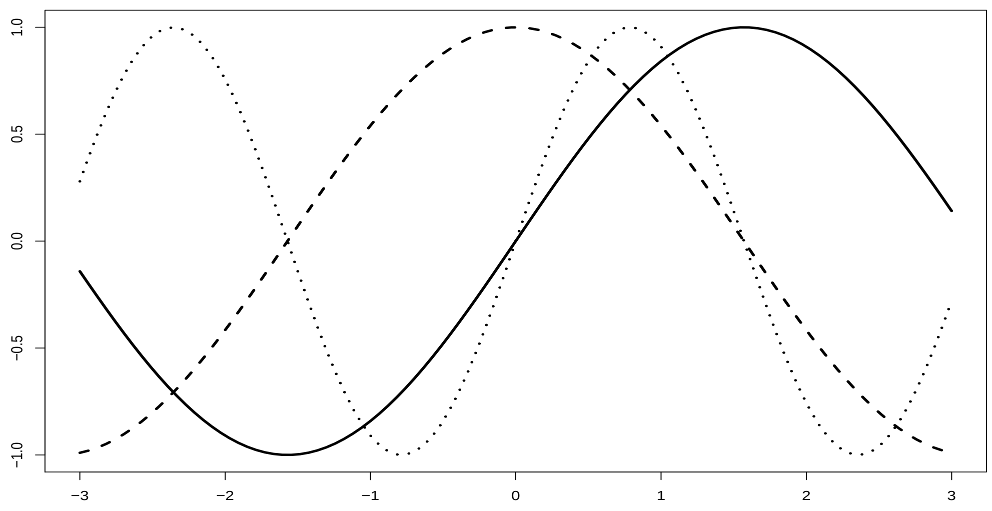

Figure 8.4: The first three Fourier basis functions.

with sharp local behavior in a very efficient way, as most wavelet coefficients γ_{jk} in

$$f(x) = \sum_{jk} \gamma_{jk}\phi_{jk}(x) \tag{8.2}$$

are zero. More details on the theory of wavelets can be found in Daubechies (1992). If f is measured without error on $p = 2^L$ equidistant points, a discrete wavelet transform (Mallat 1989) computes the coefficients of the wavelet representation (8.2). If the observations are noisy, there is a simple procedure that thresholds the wavelet coefficients (Donoho & Johnstone 1994).

A different possibility is to derive an orthogonal basis directly from the data. This can be done for instance by extending Principal Component Analysis or Partial Least Squares Regression to functional data.

It is not always necessary or even sensible to find a set of basis function and coefficients $\boldsymbol{c}_i$ that fit the data almost perfectly. We already remarked that the measurements of the curves may be noisy and we have to take care not to overfit. The possibility of plotting the discrete functions and the fitted functions is a valuable diagnostic tool to evaluate the quality of the representation. Furthermore, it might be sufficient to represent a function in terms of a few relevant basis func-

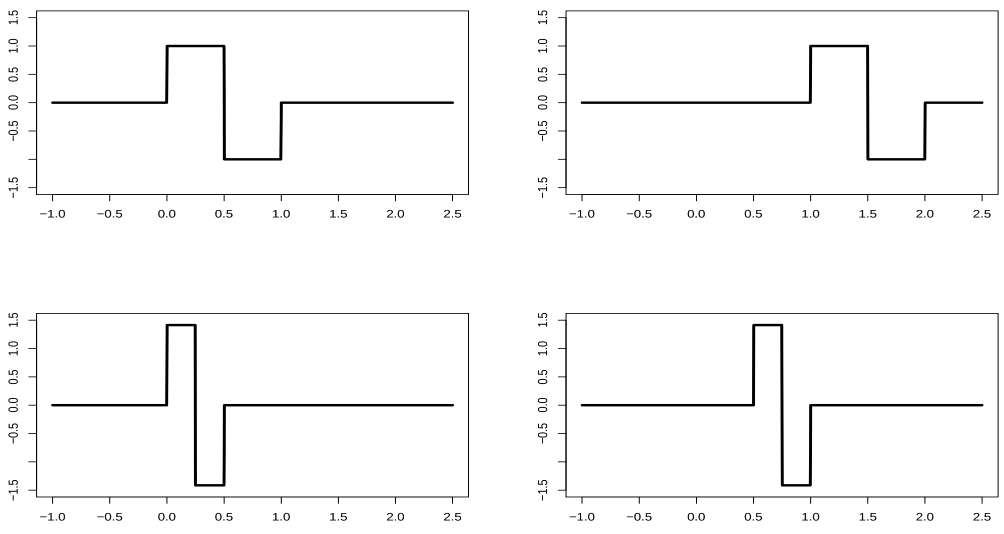

Figure 8.5: Some translations and dilatations of the Haar wavelet.

tions, which do not interpolate the data. This is illustrated in Section 9.5, where the speech recognition data are represented by a few Fourier basis functions.

8.4 Learning from Functional Data

How can we learn relationships (1.1) from functional data? For the start, we only consider linear relationships in (1.1). That is, in the regression setting ($\mathcal{Y} = \mathbb{R}$), elements $f \in \mathcal{F} = \{\mathcal{X} \to \mathbb{R}\}$ are assumed to be linear (up to an intercept) and continuous. As $\mathcal{X} = L^2(T)$ is a Hilbert space, it follows that any function $f \in \mathcal{F}$ is of the form

$$f(x(t)) \quad = \quad \beta_0 + \int_T \beta(t)x(t)dt\,. \tag{8.3}$$

In the two-class classification setting ($\mathcal{Y} = \{\pm 1\}$), we use sign(f) instead of f. As already mentioned in Chapter 1, one possibility to estimate f or β is to minimize the empirical risk (1.5). Note that this is an ill-posed problem, as there are (in general) infinitely many functions $\beta(t)$ that fit the data perfectly. There is obviously a need for regularization, in order to avoid overfitting. We can solve this problem by using a base expansion of both the predictor variable $x_i(t)$ as in

(8.1) and the function

$$\beta(t) \;=\; \sum_{l=1}^{K_\beta} b_l \phi_l(t)\,. \tag{8.4}$$

This transforms (1.5) into a parametric problem, as we need to estimate the regression coefficients

$$\boldsymbol{b} \;=\; \left(b_1, \ldots, b_{K_\beta}\right)^t.$$

If we use the quadratic loss, this is a matrix problem. We set

$$\boldsymbol{C} \;=\; \begin{pmatrix} \boldsymbol{c}_1^t \\ \vdots \\ c_n^t \end{pmatrix} \in \mathbb{R}^{n \times K_x} \quad \text{and} \quad \boldsymbol{J} \;=\; \left(\int_T \phi_i(t)\phi_j(t)dt\right)_{1 \le i \le K_x, 1 \le j \le K_\beta}.$$

It follows that (for centered data)

$$\widehat{\boldsymbol{b}} \;=\; \left(\boldsymbol{Z}^t \boldsymbol{Z}\right)^{-1} \boldsymbol{Z}^t \boldsymbol{y}\,, \tag{8.5}$$

with $\boldsymbol{Z} = \boldsymbol{C}\boldsymbol{J}$. As already mentioned, we have to regularize this problem. There are at least three possibilities. Firstly, we constrain the number of base functions in (8.4). That is, we demand that $K_\beta \ll K_x$. We show in Section 9.4 that this strategy can lead to trivial results in the Boosting setting. The second possibility is to add a penalty term $r(\beta)$ to the empirical risk (1.5). If we consider functional data, it is common to use a penalty term of the form

$$r(\beta) \;=\; \lambda \int_T \left(\beta^{(k)}(t)\right)^2 dt\,.$$

Here $\beta^{(k)}$ is the kth derivative of β – provided that this derivative exists. The choice of k depends on the data at hand and our expert knowledge on the problem. The third possibility is to apply an appropriate Boosting method. This is done in Chapter 9.

If the relationship between predictors and response are assumed to be nonlinear, there are several possibilities and we briefly discuss two of them. On the one hand, we can apply any fitting method suited for nonlinear problems to the coefficients in the representation (8.1). On the other hand, we can try to apply the kernel trick that is discussed in Section 1.4. Preda (2006) study regression problems in Reproducing Kernel Hilbert Spaces and Villa & Rossi (2005) use the kernel trick in order to extend the framework of Support Vector Machines to functional data.

It is also possible to apply linear transformations to the data prior to using a fitting method. Villa & Rossi (2005) give an illustrative example using the Tecator data set that is described in Chapter 7. Figure 8.6 shows the spectra for the data set split into two parts. Spectra that correspond to a high amount of fat tend to have two maxima instead of one. This implies that it might be worthwhile to consider the curvature of the curves. In other words, we use the second derivative of each function.

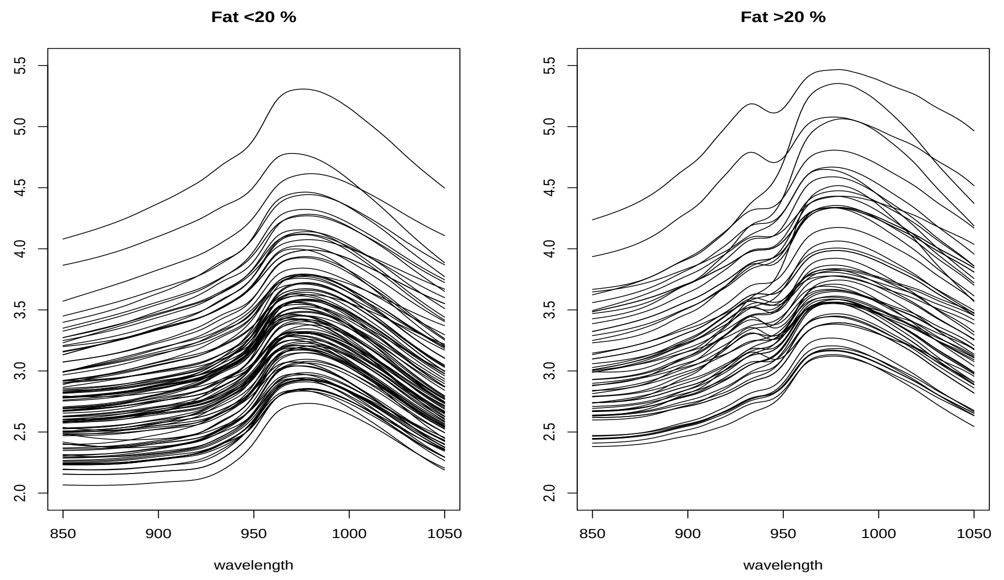

Figure 8.6: Spectra of the Tecator data set.

Finally, let us briefly mention how to model a linear relationship (1.1) if both the predictor and response variables are functional. We consider functions

$$\begin{aligned} f : L^2(T) &\rightarrow L^2(T)\,, \\ f(x(t)) &= \alpha(t) + \int_T \beta(s,t)x(s)ds\,. \end{aligned}$$

We estimate β by expanding y_i, x_i, α in terms of a basis and by representing β by

$$\beta(s,t) = \sum_{k=1}^{K_1}\sum_{l=1}^{K_2} b_{kl}\phi_k(s)\phi_l(t)\,.$$

The optimal coefficients b_{kl} are determined using the loss function

$$L(y, y') \;=\; \int_T \left(y(t) - y'(t)\right)^2 dt\,.$$

Again, we have to regularize in order to obtain smooth estimates that do not overfit.

Chapter 9

Boosting for Functional Data

In this chapter, we first introduce the main ideas of Boosting. This introduction is rather sketchy as we only want to display those results that are needed in the second part of this chapter. There, we explain how to apply these methods to functional data.

9.1 A Short Introduction to Boosting

Let us return to the learning task described in Chapter 1. Recall that a strategy to estimate relationship (1.1) is called a learner. We repeat that the choice of the learner is an important issue. If the learner is too complex, it adapts itself too much to the data at hand. If it is too weak, it is not able to detect the relevant structure of the data.

The basic idea of Boosting is to proceed stepwise and to combine weak learners in such a way that the composite – boosted – learner

$$g_M(x) = \sum_{m=1}^{M} \alpha_m f_m(x) \tag{9.1}$$

(or $\operatorname{sign}(g_M)$ for classification problems) performs better than the single weak learners f_m. The single learners are usually called base learners and M is called the number of Boosting iterations. The learners f_m and the weights α_m are chosen adaptively from the data.

A generic Boosting algorithm proceeds in the following way. In each step m, a weighting D_m of the sample $\mathcal{S} = \{(x_i, y_i)|i = 1, \ldots, n\}$ is defined. We fit a function f_m by applying the weak learner to the weighted sample. Next, we determine an appropriate coefficient α_m for the function f_m. We update the weights $D_{m+1}(x_i)$. The idea is to give a higher weighting to those points (x_i, y_i) that are poorly approximated by f_m. After a suitable stopping criterion is fulfilled, we output (9.1) or – in the case of classification problems – $\text{sign}(g_M)$.

The key ingredient of this Boosting procedure is the reweighting of the sample $\mathcal{S}$. Points which are hard to approximate in step m are given more emphasis in the next iteration step. For some learners, it is not possible to compute a weighted loss. Instead, in each step we draw with replacement a sample of size n from $\mathcal{S}$ and use the weights D_m as probabilities.

AdaBoost (Freund & Schapire 1997) – the first Boosting algorithm – is designed for two-class classification problems, i.e. $y_i \in \{\pm 1\}$. The coefficients α_m are determined by first computing the weighted empirical risk

$$\epsilon_m = \frac{1}{\sum_i D_m(x_i)} \sum_{i=1}^{n} D_m(x_i) I_{\{y_i \neq f_m(x_i)\}} \tag{9.2}$$

and then setting

$$\alpha_m = \ln\left(\frac{1-\epsilon_m}{\epsilon_m}\right).$$

The weights are updated in the following way:

$$D_{m+1}(x_i) = D_m(x_i) \exp\left(-\alpha_m y_i f_m(x_i)\right).$$

The weight α_m depends on the weighted error ϵ_m. Note that the weights α_m are non-negative if and only if the weighted empirical error does not exceed 1/2. This implies that the weak learner must fulfill the following condition. For each weighting of the data, it must achieve an empirical error that is slightly better than random. This will be formalized in definition 9.2.

We have to determine a suitable stopping criterion. Some authors suggest to stop at step M, if $\epsilon_{M+1} = 0$. Other suggestions are to stop the Boosting algorithm if the empirical error of the Boosting classifier is 0. Experiments have shown that the generalization error of Boosting can continue to decrease even if the empirical error is already 0. This has lead to the optimistic assumption that Boosting does not overfit at all. This is however not true and we should determine M by one of the model selection criteria described in Chapter 2. In most applications, the test error curve (as a function of the number of Boosting iterations) stays rather flat around the optimal region. As a consequence, the choice of the optimal number of iterations is usually not a crucial task.

Let us now return to the definition of a weak learner. For any weighting $D = (D_1, \ldots, D_n)$ of the sample $\mathcal{S}$ and any function $f \in \mathcal{F}$, we define the weighted empirical risk as

$$\epsilon(f, D) = \frac{1}{\sum_i D_i} \sum_{y_i \neq f(x_i)} D_i \,. \tag{9.3}$$

Let us start with the following definition.

Definition 9.1 (Baseline learner)**.** Let D be a probability distribution on the sample $\mathcal{S}$. The baseline learner is the constant function

$$f_{bl}(x) = \text{sign}\left(\sum_{i:y_i=+1} D_i - \sum_{i:y_i=-1} D_i \right) .$$

The baseline learner does not depend on x for a given distribution D. It assigns the label $+1$ if and only if the weighted majority of all examples in the sample $\mathcal{S}$ belongs to the class $+1$. Consequently, the weighted empirical error of the baseline learner is at most $1/2$. We demand that a weak learner is always uniformly better than the base learner.

Definition 9.2. A fitting method (learner) is called a weak learner, if there exists $1/2 \geq \gamma > 0$ sucht that for any distribution D on $\mathcal{S}$, the function f that is produced by the learner fulfills

$$\epsilon(f, D) \leq \frac{1}{2}(1 - \gamma) \,, .$$

In applications, the most widely used weak learner is a stump, i.e. a classification tree with two final nodes.

Recall that the empirical risk defined in (1.5) depends on the function f evaluated at the points $x_1, \ldots, x_n$. We can rephrase the principle of empirical risk minimization in the following way. The task is to minimize

$$\widehat{R}(\boldsymbol{u}) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, u_i)$$

with respect to $\boldsymbol{u}$ under the constraint that $u_i = f(x_i)$ for a function $f \in \mathcal{F}$. It can be shown (Breiman 1998, Breiman 1999) that Boosting solves this problem in a forward stage-wise manner using gradient descent techniques. More precisely, in each step we fit a weak learner to x_i and the negative gradient $-\nabla\widehat{R}$ at the current estimate $g_m(\boldsymbol{x}_i)$. To ensure that the gradient descent method works well, we assume that the loss function is convex and differentiable in the second component. A generic Boosting algorithm proceeds in the following way (Bühlmann & Yu 2003, Friedman 2001). Set $m = 1$. We fit a function $f_1(x)$ using a base learner and set $g_1 = f_1$. Until a suitable stopping criterion is fulfilled, we determine the negative gradient vector of the empirical risk at $g_m(x_i)$. It is straightforward to show that the negative gradient is

$$u_i = -\left.\frac{\partial L(y_i, u_i)}{\partial u_i}\right|_{u_i = g_m(x_i)}, i = 1, \ldots, n\,. \tag{9.4}$$

The best greedy step towards the minimum of $\widehat{R}(\boldsymbol{u})$ is the negative gradient vector. As we restrict ourselves to an additive expansion as displayed in (9.1), the step vector must belong to the linear span of the class of functions $\mathcal{F}$. We hence have to find the best step direction under the constraint that it is a multiple of an element of $\mathcal{F}$, i.e. the step direction is of the form αf with $\alpha \in \mathbb{R}$ and $f \in \mathcal{F}$. This is done by fitting a function to the modified sample $\{(x_i, u_i)\}$ using a base learner. If we use the quadratic loss, this yields (for a fixed α),

$$f_{m+1} = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} (u_i - \alpha f)^2 \tag{9.5}$$

After determining the optimal step direction $f_{m+1}(x)$, we have to determine the optimal step size α_{m+1}:

$$\alpha_{m+1} = \arg\min_{\alpha} \sum_{i=1}^{n} L\left(y_i, g_m(x_i) + \alpha f_{m+1}(x_i)\right)\,. \tag{9.6}$$

We update the function

$$g_{m+1} = g_m(x) + \alpha_{m+1} f_{m+1}(x)\,,$$

increase m by one and proceed as described above.

It can be shown (Friedman 2001, Hastie et al. 2001) that AdaBoost corresponds to the exponential loss function

$$L(y, y') \;=\; \exp(-yy')\,,$$

if we use the quadratic loss function (9.5) for the base learner.

The connection between Boosting and gradient descent methods has lead to a wide range of new algorithms (Friedman 2001), notably for regression problems. Note that if we use the quadratic loss

$$L(y, y') \;=\; \frac{1}{2}\left(y - y'\right)^2\,,$$

the negative gradient is simply the vector of residuals, i.e. we iteratively fit the residuals using a weak learner. This method is called L_2Boost (Bühlmann & Yu 2003, Friedman 2001). In Bühlmann (2006) and Bühlmann & Yu (2006), L_2Boost with componentwise weak learners are investigated. Suppose that we have p predictor variables. In each Boosting iteration, out of all p variables $X^{(1)}, \ldots, X^{(p)}$, we select the one variable that reduces the the empirical risk (1.5) the most:

$$k_m \;=\; \arg\min_k \left\{ \frac{1}{n} \sum_{i=1}^{n} L\left(u_i, H^{(k)}\left(x_i^{(k)}\right)\right)\right\}\,. \tag{9.7}$$

Here, $H^{(k)}$ is a univariate base learner that is applied to the variable $X^{(k)}$. Examples are univariate least squares regression or univariate smoothing splines. We estimate the regression function f_m that is obtained by applying the weak learner $H^{(k_m)}$ to $\boldsymbol{x}_i$ and the residuals u_i. We use the shortcut

$$f(x) \;=\; f_{H,\boldsymbol{u}}(x) \tag{9.8}$$

to indicate that the function f is based on the learner H applied to the response

$\boldsymbol{u}$. The function g_{m-1} is then updated by $\nu f_m(x)$, with $0 < \nu \leq 1$ a predefined shrinkage parameter. Bühlmann & Yu (2006) suggest a fixed shrinkage value of $\nu = 0.1$. This Boosting algorithm often produces sparse solutions. That is, only a small fraction of the predictor variables are included in the final model. This can lead to interpretable models.

Boosting with the loss function

$$L(y, y') = \log\left(1 + exp(-yy')\right)$$

is suited for classification problems and called LogitBoost (Friedman et al. 2000).

Algorithm 9.3 (LogitBoost)**.** *For any sample $\mathcal{S}$ and any weak learner H, we initialize the probabilities $p_1(x_i) = 1/2$, set $g_0(x) = 0$ and iteratively compute*

$$\begin{array}{rcll}
D_m(x_i) & = & p_m(x_i)\left(1 - p_m(x_i)\right) & \textit{weights} \\
u_i & = & \frac{y_i - p_m(x_i)}{D_m(x_i)} & \textit{negative gradient} \\
f_m & = & f_{H,\boldsymbol{u}} & \textit{function obtained by } \textbf{weighted} \textit{ weak} \\
 & & & \textit{learner (with weights } D_m(x_i)) \\
g_m(x) & = & g_{m-1}(x) + \frac{1}{2} f_m(x) & \textit{update} \\
p_{m+1}(x_i) & = & \left(1 + exp\left(-2 f_m(x_i)\right)\right) - 1 & \textit{probabilities}
\end{array}$$

The final function g_M is an estimate of one-half of the log-odds ratio

$$\frac{1}{2} \log\left(\frac{P(Y = 1|X = x)}{1 - P(Y = 1|X = x)}\right).$$

As a consequence, this classification algorithm also produces estimates of the class probabilities $P(Y = 1|X = x)$. This can be advantageous if we have non-equal misclassification costs.

In the regression setting, there is only a loose definition of the term weak learner. We speak of a weak learner if the fitting method has a high bias compared to its variance or if it only uses a few degrees of freedom.

9.2 The Degrees of Freedom of Boosting

Let us return to the subject of a suitable stopping criterion. One possibility is to use cross validation. Depending on the data, this can lead to high computational costs. We can alternatively compute the complexity of the Boosting algorithm in terms of its degrees of freedom. If we use L_2Boost, they can be computed efficiently (Bühlmann & Yu 2003, Bühlmann & Yu 2006). Let us assume for simplicity that all weak learners $H_1, \ldots, H_m$ are linear in $\boldsymbol{y}$. As a consequence, the Boosting learner B_M obtained after M steps is also linear in $\boldsymbol{y}$. To show this, we first use the iterative definition of the residuals

$$\boldsymbol{u}^{(m)} = \boldsymbol{u}^{(m-1)} - \nu \boldsymbol{H}_m \boldsymbol{u}^{(m-1)} = (\boldsymbol{I}_n - \nu \boldsymbol{H}_m)\, \boldsymbol{u}^{(m-1)}\,.$$

We obtain

$$\boldsymbol{u}^{(m)} = (\boldsymbol{I}_n - \nu \boldsymbol{H}_m) \ldots (\boldsymbol{I}_n - \nu \boldsymbol{H}_1)\, \boldsymbol{y}\,.$$

Setting $\boldsymbol{u}^{(0)} = \boldsymbol{y}$, it follows that

$$\boldsymbol{B}_m \boldsymbol{y} = \sum_{i=1}^{m} \boldsymbol{H}_i \boldsymbol{u}^{(i-1)} = \sum_{i=1}^{m} \boldsymbol{H}_i\, (\boldsymbol{I}_n - \nu \boldsymbol{H}_{i-1}) \ldots (\boldsymbol{I}_n - \nu \boldsymbol{H}_1)\, \boldsymbol{y}$$

is the linear map that defines the hat matrix of L_2Boost. It is straightforward to show (Bühlmann & Yu 2006) that

$$\boldsymbol{B}_m \;=\; \boldsymbol{I}_n - (\boldsymbol{I}_n - \nu \boldsymbol{H}_m) \ldots\ldots (\boldsymbol{I}_n - \nu \boldsymbol{H}_1)\ ,$$

and the degrees of freedom are defined as the trace of the hat-matrix $\boldsymbol{B}_m$. This matrix can be computed recursively by using the following relationship:

$$\boldsymbol{B}_{m+1} \;=\; \boldsymbol{I}_n - (\boldsymbol{I}_n - \nu \boldsymbol{H}_{m+1})\, (\boldsymbol{I}_n - \boldsymbol{B}_m)\ .$$

If the weak learners are not linear in $\boldsymbol{y}$, it is possible to derive an unbiased estimate of the degrees of freedom by computing the first derivative of

$$B_M(\boldsymbol{y}) = \sum_{i=1}^{M} H_i(\boldsymbol{u}^{(i-1)})$$

with respect to $\boldsymbol{y}$ and by then determining its trace.

Bühlmann & Yu (2006) introduce a variant of L_2Boost that is called SparseL_2Boost. It is based on the effective computation of the degrees of freedom of Boosting. Instead of choosing the base learner (9.7) that reduces the empirical risk the most, we choose the base learner that reduces an appropriate information criterion the most. Bühlmann & Yu (2006) propose the generalized minimum description length criterion that is presented in Section 2.3. Note that this criterion depends on the response $\boldsymbol{y}$ and the learner $\boldsymbol{H}$, that is $gMDL = gMDL(\boldsymbol{y}, \boldsymbol{H})$.

Algorithm 9.4 (SparseL_2Boost). *For any sample $\mathcal{S}$, we set $g_0(x) = 0$, $\boldsymbol{u} = \boldsymbol{y}$ and iteratively compute*

$$\begin{array}{rcll}
u_i & = & y_i - g_m(x_i) & \textit{residuals} \\
\boldsymbol{B}^{(k)}_{m+1} & = & \boldsymbol{I}_n - \left(\boldsymbol{I}_n - \boldsymbol{H}^{(k)}_{m+1}\right)(\boldsymbol{I}_n - \boldsymbol{B}_m) & \textit{Boosting operator for the kth} \\
 & & & \textit{variable} \\
k_{m+1} & = & arg\min_k gMDL\left(\boldsymbol{y}, \boldsymbol{B}^{(k)}_{m+1}\right) & \textit{selection of the optimal variable} \\
\boldsymbol{H}_{m+1} & = & \boldsymbol{H}^{(k_m)}_{m+1} & \textit{optimal base learner} \\
f_{m+1}(x) & = & f_{\boldsymbol{H}_{m+1}, \boldsymbol{u}_i}(x) & \textit{fitting of the residuals} \\
g_{m+1}(x) & = & g_m(x) + \nu f_{m+1}(x) & \textit{update} \\
\boldsymbol{B}_{m+1} & = & \boldsymbol{I}_n - (\boldsymbol{I}_n - \nu \boldsymbol{H}_{m+1})(\boldsymbol{I}_n - \boldsymbol{B}_m) & \textit{update}
\end{array}$$

The optimal number of Boosting iterations is the one for which the generalized MDL criterion is minimal. Note that SparseL_2Boost is completely automatic in the sense that we do not have to select any external, additional model parameters.

9.3 Boosting and Partial Least Squares

Before extending the framework of Boosting to functional data, we insert a few remarks on the connections of Boosting and PLS regression. There have been approaches to use PLSR as a base learner for Boosting algorithms. Mevik et al. (2004) try to improve the performance of PLSR by averaging over several PSLR estimates that are obtained from Bootstrap samples. This general strategy is known as Bagging (Breiman 1996). However, experiments on data show that Bagging does not improve the performance of PLSR. In Boulesteix (2004) PLSR is used as a base learner for classification problems by means of the Boosting-by-reweighting technique. It is shown on several data sets that the performance of PLSR does not improve. Zhang et al. (2005) combine PLSR and L_2Boost in the regression setting. More precisely, PLSR with one latent component as a base learner is used.

Algorithm 9.5 (PLSBoost). *For any sample $\mathcal{S}$, we set $\boldsymbol{y}^{(res)} = \boldsymbol{y}$. The latent components and the regression estimates $\widehat{\boldsymbol{y}}^{(m)}$ of PLSBoost are determined by* *iteratively computing*

$$\begin{array}{rcll} \boldsymbol{t}^{(m)} & = & \boldsymbol{X}\boldsymbol{X}^t\boldsymbol{y}^{(res)} & \textit{components} \\ \widehat{\boldsymbol{y}}^{(m)} & = & \mathcal{P}_{\boldsymbol{t}^{(m)}}\boldsymbol{y}^{(res)} & \textit{estimate} \\ \boldsymbol{y}^{(res)} & = & \boldsymbol{y}^{(res)} - \widehat{\boldsymbol{y}}^{(m)} & \textit{residuals} \end{array}$$

In Zhang et al. (2005), this algorithm is compared to PLSR on different data sets. Although PLSBoost is better on all data sets, the improvement is tiny and the marginal decrease of test error is bought dearly with a substantial increase in computational costs.

We now try to give a heuristic explanation why PLSR fails as a base learner in the Boosting framework. The examples that are investigated in Zhang et al. (2005) are very high-dimensional data sets. As $\boldsymbol{X}$ is highly collinear, the Gram matrix $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{X}^t$ is very close to a rank-one matrix. It can be approximated by the first eigenvector $\boldsymbol{v}_1$ of $\boldsymbol{K}$ in the following sense:

$$\boldsymbol{K} \approx \lambda_1 \boldsymbol{v}_1 \boldsymbol{v}_1^t \,.$$

It follows that the components $\boldsymbol{t}^{(m)}$ are – up to a scaling factor c – approximately equal to $\boldsymbol{v}_1$.

$$\boldsymbol{t}^{(m)} = \boldsymbol{K}\boldsymbol{y}_{res}^{(m)} \approx c\boldsymbol{v}_1 \,.$$

Hence $\boldsymbol{t}^{(m)}$ hardly depends on the $\boldsymbol{y}$-residuals at all. This implies that the PLSR base learner equals approximately the projection onto the first eigenvector $\boldsymbol{v}_1$. As the residuals are almost orthogonal on $\boldsymbol{v}_1$, the empirical risk reduces extremely slowly. This can be seen in the examples given in Zhang et al. (2005). There, sometimes up to 6 000 Boosting iterations are needed until the algorithm is stopped.

To summarize, for highly collinear data, PLSR is not an appropriate base learner for L_2Boost, as it is too weak. It hardly depends on the response $\boldsymbol{y}$.

9.4 Functional Boosting

After this short excursion, we return to the main purpose of this chapter. How can we apply Boosting techniques to functional data? We first have to extend the notion “weak learner”. In the classification setting, we can adopt definition 9.2. A weak learner is a learner that is slightly better than random. What are examples of weak learners? Note that it is possible to apply most of the multivariate data analysis tools to functional data. We use a finite-dimensional approximation as in (8.1) and simply apply any appropriate algorithm. In this way, it is possible to use stumps (that is, classification trees with one node) or neural networks as base learners.

In the regression setting, we propose the following definition: A weak learner is a learner that has only few degrees of freedom. Examples include the two regularized least squares algorithms presented in Section 8.3 – restriction of the number of base functions in (8.4) or addition of a penalty term to (1.5). Note however that the first method leads to trivial results if we use L_2Boost. The learner is simply the projection of $\boldsymbol{y}$ onto the space that is spanned by the columns of $\boldsymbol{Z}$ (recall (8.5)). Consequently, the $\boldsymbol{y}$-residuals are orthogonal on $\boldsymbol{Z}$ and after one step, the Boosting solution does not change anymore.

The following example of a weak learner is an extension of the componentwise weak learner introduced in (9.7). It is suited for L_2Boost. We first initialize $g_0(x(t)) = 0$. In each Boosting step, we select one basis function $\phi_k(t)$ of the expansion (8.4). To select this basis function, we estimate for each function ϕ_k the regression estimates of the regression model

$$u_i = \gamma_0 + \gamma_1 \int_T x_i(t)\phi_k(t)dt + \varepsilon_i \,. \tag{9.9}$$

This defines for each basis function a linear weak learner $\boldsymbol{H}^{(k)}$. In accordance with the notation introduced in (9.8), we have

$$f_{\boldsymbol{H}^{(k)},\boldsymbol{u}}(x(t)) = \widehat{\gamma}_0 + \widehat{\gamma}_1 \int_T x(t)\phi_k(t)dt \,. \tag{9.10}$$

We now choose the basis function $\phi_{k^*}(t)$ that either minimizes the empirical risk (1.5) or that minimizes the generalized MDL criterion. If we opt for the latter approach, we call this novel method functional SparseL_2Boost. The function $g_m(x(t))$ is then updated by a small fraction ν of $f_{\boldsymbol{H}^{(k^*)},\boldsymbol{u}}(x(t))$,

$$g_{m+1}(x(t)) = g_m(x(t)) + \nu f_{\boldsymbol{H}^{(k^*)},\boldsymbol{u}}(x(t)) \,.$$

In Section 9.6, we study this algorithm on the biscuit data set introduced in Section 8.1.

Finally, let us remark that if the response variable is functional, we can adopt the same definition of weak learner as in the regression setting: A weak learner is a learner that uses only a few degrees of freedom.

9.5 Example: Speech Recognition

We illustrate the application of Boosting methods to functional data on the speech recognition data set. To this end, we first represent the time series $x_i(t)$ in terms of a Fourier basis expansion of dimension $K_x = 100$. We opt to include a generous

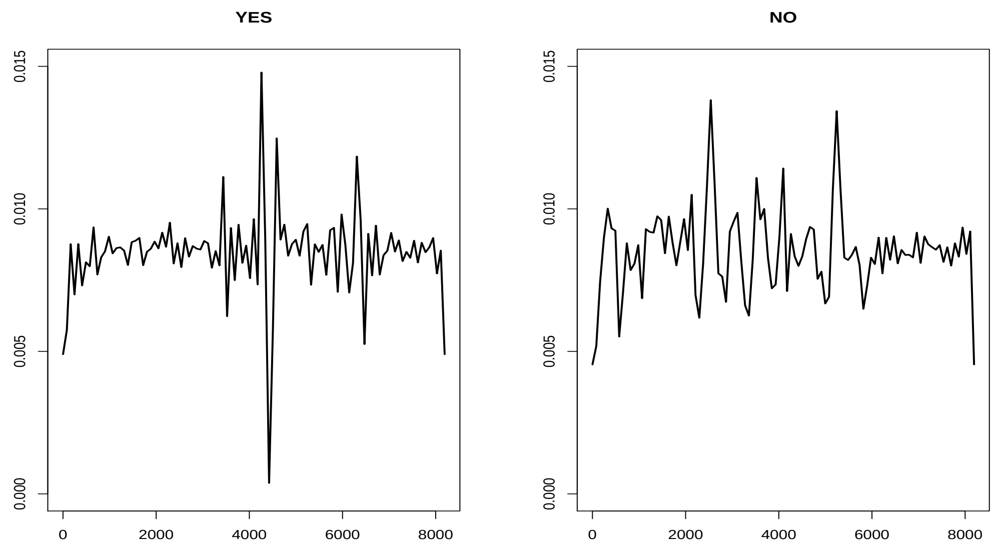

Figure 9.1: Representation of the functions in Figure 8.3 in terms of the first 100 Fourier basis functions.

amount of basis functions, as experiments indicate that the results of LogitBoost are insensitive to the addition of possibly irrelevant basis functions. We remark that the Fourier representation does not resemble the original data very much. This is illustrated in Figure 9.1.

Next, we apply the LogitBoost algorithm 9.3 to the coefficients of the Fourier expansion. For the weak learner, we choose a classification tree with two final nodes. The optimal number of components is estimated using 10fold cross-validation (cv). The minimal cv error over all Boosting iterations is 0.1, obtained after 24

Boosting iterations. This is the same error rate that is reported in Biau et al. (2005). There, a functional k-nearest-neighbor-algorithm is applied to the data.

9.6 Example: Biscuit Dough Data Set

Note that in the example of Section 8.2, the Boosting algorithm returns a function that is a linear combination of many classification trees. Functions like this are hard to interpret and this has sometime lead to the belief that Boosting is a "black box" that is only valuable for prediction but not capable to produce interpretable models. This is however not true, and we now describe how we can use Boosting to detect important features of the data.

In Section 9.4, we introduced the functional SparseL_2Boost algorithm. It is based on selecting only one basis function in each Boosting iteration. If the final model includes only a few basis functions, it might be possible to find an easy interpretation of the regression function. Recall that for the biscuit dough data, our aim is to find regions of relevant wavelengths. This implies that the estimated regression function $\beta(t)$ should be 0 on a wide range of the wavelengths. To achieve this, we first represent the data in terms of basis functions that have a very small support. Then, we apply functional SparseL_2Boost to the data using the weak learners that are defined in (9.10).

We choose Daubechies wavelets for this application. They are illustrated in Figure 9.2. We repeat that the SparseL_2Boost algorithm does not rely on any external model parameters that have to be estimated. In the case of functional data, the number of basis functions however constitutes an additional parameter. We now show how to determine the optimal number of basis functions and the optimal number of Boosting iterations using the generalized MDL criterion.

From a technical point of view, in order to compute a wavelet transformation, it is necessary to have observations at 2^L equidistant points. To fulfill this requirement, we fit the initial $p = 700$ observations using a lot of B-splines basis functions (in this case, 65) and evaluate these functions at 2^L equidistant points. Afterwards, it is possible to estimate the coefficients of the wavelet transformation that constitutes of 2^L wavelet basis functions. Hence, L determines the number of basis functions and has to be estimated from the data. This is done by running

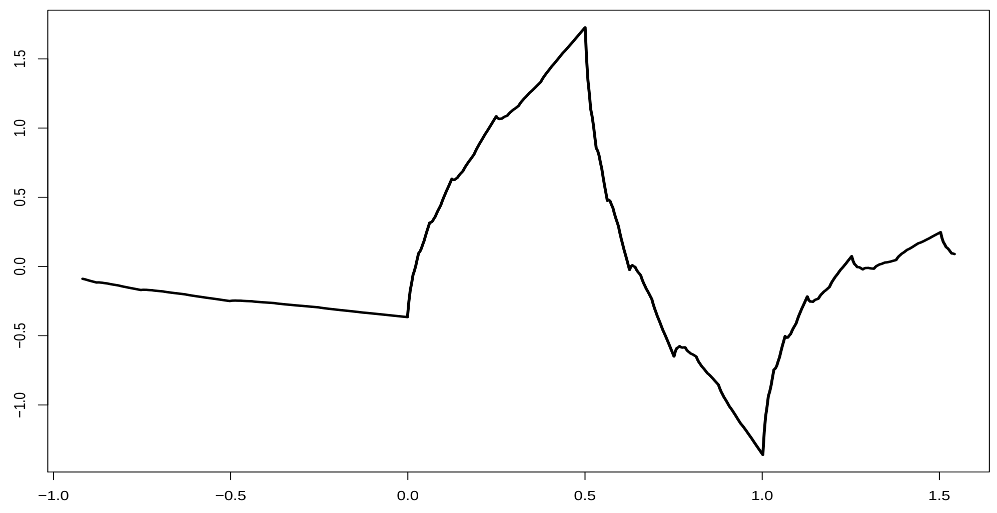

Figure 9.2: Daubechies wavelet.

functional SparseL_2Boost for different values of L in the range of $3, 4, \ldots, 10$. The optimal number value of L is obtained by comparing their generalized MDL criterion (for their respective optimal number of Boosting iterations).

In order to compare functional SparseL_2Boost to PLSR, we randomly split the whole data set (that consists of $39+31 = 70$ observations) into a training set of size 39 and a test set of size 31. The optimal number of PLSR components is estimated on the training set using 5fold cross-validation. The optimal parameters for the Booosting algorithm (i.e. L and the number of Boosting iterations m) are found by minimizing the generalized MDL criterion on the training set. The optimal models are then evaluated on the test set. The random splitting is repeated 50 times. The boxplot of the test errors are depicted in Figure 9.3. We also conduct a Wilcoxon rank sum test to test the alternative hypothesis that the test error of the Boosting algorithm is lower than the test error of PLSR. The median test error for the two methods, their optimal model parameters (estimated on the original training set) and the p-value for the Wilcoxon rank sign test are displayed in Table 9.1. The regression coefficients that we obtain from functional SparseL_2Boost are displayed in Figure 9.4. Here, we compute the optimal model on the original training set with the optimal model parameters that are displayed in Table 9.1. We see that the two methods are compatible. We can clearly distinguish a region of relevant wavelengths in the range of $\approx 1600 - 1900$ nanometers. In addition,

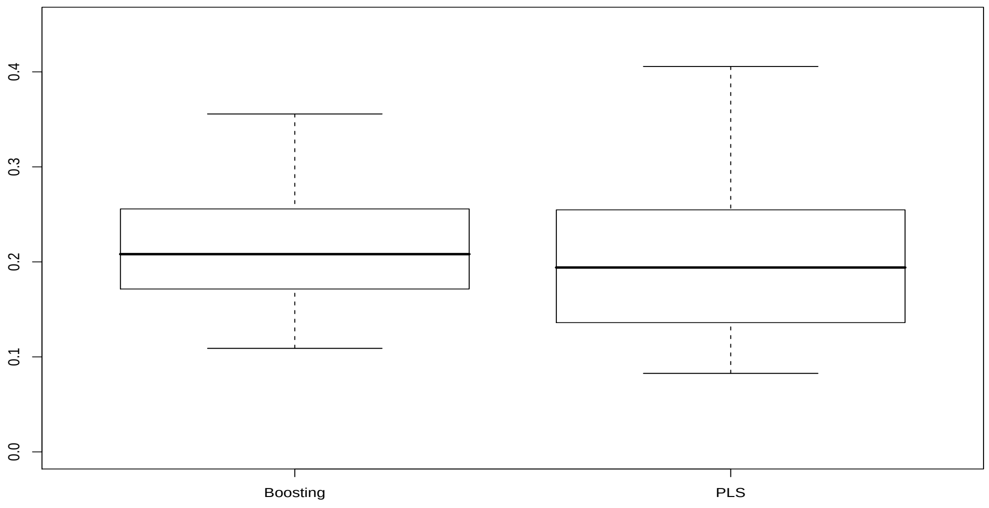

Figure 9.3: Test errors for PLSR and functional SparseL_2Boost.

	median test error	optimal model parameters	p-value
PLSR	0.194	$m = 13$	0.1067
Boosting	0.208	$L = 7, m = 70$	–

Table 9.1: Median test error, optimal parameter values and p-value for the biscuit dough data set.

there is a – somewhat less pronounced - region around $\approx$ 1400 nanometers.

9.7 Conclusion

The extension of Boosting methods to functional data is straightforward. After choosing a base algorithm (which we call a weak learner), we iteratively fit the data by either applying this algorithm to reweighted samples or by using a gradient descent technique. In many applications, we use a finite-dimensional expansion of the functional examples in terms of base functions. This finite-dimensional representation can then be plugged into existing algorithms as LogitBoost or L_2Boost. In addition, it is possible to extract sparse models from the data. We proposed a method that is based on an extension of Boosting algorithms (that perform variable selection) to functional data.

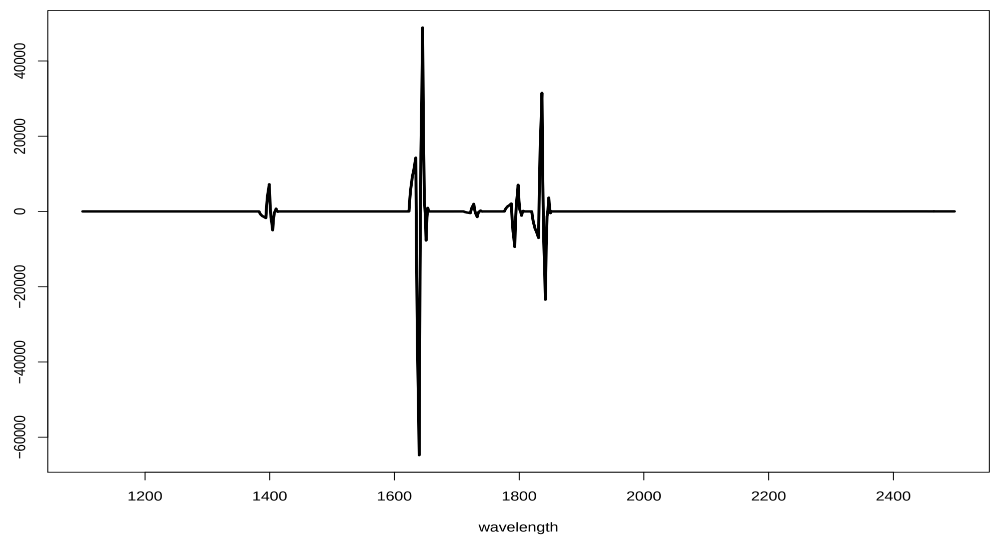

Figure 9.4: Standardized regression coefficients of functional SparseL_2Boost for 2^7 wavelet basis functions and 70 Boosting iterations.

Chapter 10

Summary and Outlook

In this work, we studied different methods for the analysis of high-dimensional data. In this chapter, we briefly review the main results and discuss some open problems and possible future research directions.

We provided two negative results on Partial Least Squares path models. On the one hand, we illustrated that the PLS algorithms in mode B do not necessarily converge to the maximum of the associated optimization problem. More severely, we showed that the algorithms with at least one block in mode A are not even attached to any (sufficiently smooth) optimization problem! Although we suggested a modification of mode A, there still remain a lot of unanswered questions. Do the optimization problems in mode B and (modified) mode A really reflect the outer PLS model as illustrated in Figure 3.3? Which scheme is the appropriate one? How do we specify the PLS model at all? There is no rule on how to determine the presence or the direction of the arrows in the inner model. In applications, this is usually done ad-hoc. Another serious problem that is prevalent in a lot of applications is the interpretation of the model. It is common to analyze the quality of the PLS model in terms of the size of the weight vectors and there are statistical tests (based on Bootstrap techniques) to determine their significance. These approaches are often only justified by heuristic arguments. To summarize, we advise to meet results based on PLS path modeling techniques with a portion of skepticism.

These reservations are however not valid for Partial Least Squares for Regression. PLSR is in fact a well-founded and established tool in the analysis of high-dimensional data, and it has been used successfully in a broad range of applications. It is computationally fast and it can be easily extended to nonlinear problems with the help of the kernel trick. Its close connection to methods from linear algebra (as illustrated in Chapter 6) has lead to further theoretical results on PLSR. In particular, a lot of statistical properties of PLSR depend on this relationship. As an example, we explained in Chapter 7 how to compute the shrinkage factors of PLSR by exploiting its relationship to Krylov methods. The interrelation between numerical linear algebra and biased multivariate regression techniques is truly fascinating. We hope that – based on this – it is possible to gain additional insight into existing statistical methods or even to establish new ones.

As a matter of fact, we introduced one new statistical method that can be justified in terms of numerical linear algebra. The penalized PLSR approach developed in Chapter 5 is proven to be equal to a preconditioned conjugate gradient algorithm. A different motivation for this novel technique is given in terms of the kernel trick. In addition, penalized PLS in combination with a B-splines transformation can be successfully applied to the estimation of very high-dimensional generalized additive models. In the examples that are presented in Chapter 5, the novel method outperforms two other methods for modeling GAM's. However, there is still space for improvement. We already mentioned that in the penalized PLSR approach, the degree of smoothness is the same for each variable. This is a drawback compared to other methods. Recall that we illustrated in Section 5.5 that the smoothness depends on the number of penalized PLSR components that are included in the model. In order to assign different degrees of smoothness to each variable, we might therefore allow different numbers of components for each variable. As this would lead to an infeasible, high-dimensional model selection problem, we suggest a slightly different approach based on Boosting techniques. In the spirit of componentwise L_2Boost, we proceed stepwise. In each step, we increase the number of penalized PLSR components for only one selected predictor variable. The selection might be based on the maximal reduction of the empirical risk. If the estimation of the degrees of freedom of (penalized) PLSR in its current form was more reliable, it would even be possible to adapt SparseL_2Boost to this particular problem.

To conclude our comments on Partial Least Squares, we remark that it was not our primary goal to advertise truncated PLSR as a pioneering new regression method. There have been discussions in the literature whether the shrinkage behavior of PLSR leads to inferior statistical properties or not, and our aim was mainly to investigate empirically these statistical properties.

In Chapter 9, we analyzed functional data with the help of Boosting techniques. It is perhaps not surprising that the extension of Boosting methods to functional data is straightforward. The achievement of this chapter is not merely the combination of these two concepts, but rather the introduction of methods that produce sparse functional regression models. It is definitely worthwhile to study further the potentials of functional SparseL_2Boost, and its illustration on one single data set is surely insufficient. Topics that might be investigated are e.g. the influence of the type of basis functions, the influence of the model selection criterion and the reliability of this method – for instance in terms of (Bootstrap) confidence intervals.

Bibliography

Akaike, H. (1973), Information Theory and an Extension of the Maximum Likelihood Principle, *in* 'Second International Symposium on Information Theory', pp. 267–281.

Biau, G., Bunea, F. & Wegkamp, M. (2005), 'Functional Classification in Hilbert Spaces', *IEEE Transactions on Information Theory* **47**, 116–128.

Boulesteix, A.-L. (2004), 'PLS Dimension Reduction for Classification with Microarray Data', *Statistical Applications in Genetics and Molecular Biology* **3**.

Boulesteix, A.-L. (2006), 'Maximally Selected Chi-Square Statistics for Ordinal Variables', *Biometrical Journal* .

Boulesteix, A.-L. & Strimmer, K. (2006), 'Partial Least Squares: A Versatile Tool for the Analysis of High-Dimensional Genomic Data', *Briefings in Bioinformatics* . to appear.

Breiman, L. (1996), 'Bagging Predictors', *Machine Learning* **24**, 123–140.

Breiman, L. (1998), 'Arcing Classifiers (with Discussion)', *The Annals of Statistics* **26**(3), 801–849.

Breiman, L. (1999), 'Prediction Games and Arcing Classifiers', *Neural Computation* **11**, 1493–1517.

Brown, P., Fearn, T. & Vannucci, M. (2001), 'Bayesian Wavelet Regression on Curves with Application to a Spectroscopic Calibration Problem', *Journal of the American Statistical Association* **96**, 398–408.

Bühlmann, P. (2006), 'Boosting for High-Dimensional Linear Models', *The Annals of Statistics* **34**, 559–583.

Bühlmann, P. & Yu, B. (2003), 'Boosting with the L2-Loss: Regression and Classification', *Journal of the American Statistical Association* **98**, 324–339.

Bühlmann, P. & Yu, B. (2006), 'Sparse Boosting', *Journal of Machine Learning Research* **7**, 1001–1024.

Burnham, K. & Anderson, D. (2004), 'Multimodel Inference: Understanding AIC and BIC in Model Selection', *Sociological Methods and Research* **33**, 261–304.

Butler, N. & Denham, M. (2000), 'The Peculiar Shrinkage Properties of Partial Least Squares Regression', *Journal of the Royal Statistical Society, Series B* **62**.

Chu, M. T. & Watterson, J. (1993), 'On a Multivariate Eigenvalue Problem: I. Algebraic Theory and a Power Method', *SIAM Journal of Scientific Computing* **14**, 1089–1106.

Daubechies, I. (1992), *Ten Lectures on Wavelets*, Siam Philadelphia.

de Bie, T., Christianini, N. & Rosipal, R. (2005), *Handbook of Geometric Computing: Applications in Pattern Recognition, Computer Vision, Neuralcomputing, and Robotics*, Springer, chapter Eigenproblems in pattern recognition.

de Boor, C. (1978), *A Practical Guide to Splines*, Springer.

de Jong, S. (1993), 'SIMPLS: an Alternative Approach to Partial Least Squares Regression', *Chemometrics and Intelligent Laboratory Systems* **18**, 251–263.

de Jong, S. (1995), 'PLS Shrinks', *Journal of Chemometrics* **9**, 323–326.

Donoho, D. & Johnstone, I. (1994), 'Ideal Spatial Adaption by Wavelet Shrinkage', *Biometrika* **81**, 425–455.

Drazin, M. (1968), 'Pseudoinverses in Associate Rings and Semigroups', *The American Mathemtaical Monthly* **65**, 506–514.

Durand, J. F. (1993), 'Generalized Principal Component Analysis with Respect to Instrumental Variables via Univariate Spline Transformations', *Computational Statistics and Data Analysis* **16**, 423–440.

Durand, J. F. (2001), 'Local Polynomial Additive Regression Through PLS and Splines: PLSS', *Chemometrics and Intelligent Laboratory Systems* **58**, 235–246.

Durand, J. F. & Sabatier, R. (1997), 'Additive Splines for Partial Least Squares Regression', *Journal of the American Statistical Association* **92**, 1546–1554.

Eilers, P. & Marx, B. (1996), 'Flexible Smoothing with B-Splines and Penalties', *Statistical Science* **11**, 89–121.

Frank, I. & Friedman, J. (1993), 'A Statistical View of some Chemometrics Regression Tools', *Technometrics* **35**, 109–135.

Freund, Y. & Schapire, R. (1997), 'A Decision-Theoretic Generalization of Online Learning and an Application to Boosting', *Journal of Computer and System Science* **55**(1), 119–139.

Friedman, J. (2001), 'Greedy Function Approximation: a Gradient Boosting Machine', *The Annals of Statistics* **29**, 1189–1232.

Friedman, J., Hastie, T. & Tibshirani, R. (2000), 'Additive Logistic Regression: a Statistical View of Boosting (with Discussion)', *The Annals of Statistics* **28**, 337–407.

Goutis, C. (1996), 'Partial Least Squares Yields Shrinkage Estimators', *The Annals of Statistics* **24**, 816–824.

Goutis, C. & Fearn, T. (1996), 'Partial Least Squares Regression on Smooth Factors', *Journal of the American Statistical Association* **91**, 627–632.

Hanafi, M. (2006), 'PLS Path Modelling: Computation of Latent Variables with the Estimation Mode B', *Computational Statistics* . to appear.

Hanafi, M. & Qannari, E. (2005), 'An Alternative Algorithm to the PLS B problem', *Computational Statistics and Data Analysis* **48**, 63 – 67.

Hansen, M. & Yu, B. (2001), 'Model Selection and Minimum Descripion Length Principle', *Journal of the American Statistical Association* **96**, 746–774.

Hastie, T. & Tibshirani, R. (1990), *Generalized Additive Models*, Chapman and Hall.

Hastie, T., Tibshirani, R. & Friedman, J. (2001), *The Elements of Statistical Learning*, Springer.

Helland, I. (1988), 'On the Structure of Partial Least Squares Regression', *Communications in Statistics, Simulation and Computation* **17**(2), 581–607.

Hestenes, M. & Stiefel, E. (1952), 'Methods for Conjugate Gradients for Solving Linear Systems', *Journal of Research of the National Bureau of Standards* **49**, 409–436.

Horst, P. (1961), 'Relations Among m Sets of Measures', *Psychometrika* **26**, 129–149.

Horst, P. (1965), *Factor Analysis of Data Matrices*, Holt, Rineheart and Winston.

Hotelling, H. (1936), 'Relations Between two Sets of Variates', *Biometrika* **28**.

Hurvich, C. & Tsai, C.-L. (1989), 'Regression and Time Series Model Selection in Small Samples', *Biometrika* **76**, 297–307.

Ibsen, I. & Meyer, C. (1998), 'The Idea behind Krylov Methods', *The American Mathematical Monthly* **105**, 889–899.

Kockelkorn, U. (2000), *Lineare Statistische Methoden*, Oldenbourg.

Kondylis, A. & Whittaker, J. (2006), 'Preconditioning Krylov Spaces and Variable Selection in PLSR', *preprint* .

Krämer, N. (2006), An Overview on the Shrinkage Properties of Partial Least Squares Regression. to appear in Computational Statistics.

Lanczos, C. (1950), 'An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators', *Journal of Research of the National Bureau of Standards* **45**, 225–280.

Lingjaerde, O. & Christopherson, N. (2000), 'Shrinkage Structures of Partial Least Squares', *Scandinavian Journal of Statistics* **27**, 459–473.

Lohmöller, J.-B. (1989), *Latent Variable Path Modeling with Partial Least Squares*, Physica-Verlag, Heidelberg.

Magnus, J. & Neudecker, H. (1988), *Matrix Differential Calculus with Applications in Statistics and Economentrics*, Wiley.

Mallat, S. (1989), 'A Theory of Multiresolution Signal Decomposition: The Wavelet Representation', *IEEE Transactions of Pattern Analysis and Machine Intelligence* **11**(7), 674–693.

Manne, R. (1987), 'Analysis of Two Partial-Least-Squares Algorithms for Multivariate Calibration', *Chemometrics and Intelligent Laboratory Systems* **2**, 187–197.

Martens, H. & Naes, T. (1989), *Multivariate Calibration*, Wiley, New York.

Mathes, H. (1993), Global Optimisation Criteria of the PLS-Algorithm in Recursive Path Models with Latent Variables, *in* K. Haagen, D. Bartholomev & M. Deister, eds, 'Statistical Modelling and Latent Variables', Elsevier Science.

Mevik, B.-H., Segtan, V. & Naes, T. (2004), 'Ensemble Methods and Partial Least Squares Regression', *Journal of Chemometrics* **18**, 498–507.

Osborne, B., Fearn, T., Miller, A. & Douglas, S. (1994), 'Application of Near Infrared Reflectance Spectroscopy to Compositional Analysis of Biscuits and Biscuits Dough', *Journal of the Science of Food and Agriculture* **35**, 99–105.

Parlett, B. (1998), *The Symmetric Eigenvalue Problem*, Society for Industrial and Applied Mathematics.

Phatak, A. & de Hoog, F. (2002), 'Exploiting the Connection between PLS, Lanczos, and Conjugate Gradients: Alternative Proofs of Some Properties of PLS', *Journal of Chemometrics* **16**, 361–367.

Phatak, A., Rilley, P. & Penlidis, A. (2002), 'The Asymptotic Variance of the Univariate PLS Estimator', *Linear Algebra and its Applications* **354**, 245–253.

Preda, C. (2006), 'Regression Models for Functional Data by Reproducing Kernel Hilbert Spaces', *submitted* .

R Development Core Team (2005), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.
*http://www.R-project.org

Ramsay, J. & Silverman, B. (2005), *Functional Data Analysis*, second edn, Springer.

Rännar, S., Lindgren, F., Geladi, P. & Wold, S. (1994), 'A PLS Kernel Algorithm for Data Sets with many Variables and Fewer Objects, Part I: Theory and Applications', *Journal of Chemometrics* **8**, 111–125.

Rosipal, R. & Krämer, N. (2006), Overview and Recent Advances in Partial Least Squares, *in* 'Subspace, Latent Structure and Feature Selection Techniques', Lecture Notes in Computer Science, Springer, pp. 34–51.

Rosipal, R. & Trejo, L. (2001), 'Kernel Partial Least Squares Regression in Reproducing Kernel Hilbert Spaces', *Journal of Machine Learning Research* **2**, 97–123.

Rosipal, R., Trejo, L. & Matthews, B. (2003), Kernel PLS-SVC for Linear and Nonlinear Classification, *in* 'Proceedings of the Twentieth International Conference on Machine Learning', Washington, DC, pp. 640–647.

Schölkopf, B. & Smola, A. (2002), *Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond.*, The MIT Press.

Serneels, S., Lemberge, P. & Espen, P. V. (2004), 'Calculation of PLS Prediction Intervals Using Efficient Recursive Relations for the Jacobian Matrix', *Journal of Chemometrics* **18**, 76–80.

Stein, C. (1981), 'Estimation of the Mean of a Multivariate Normal Distribution', *The Annals of Statistics* **9**, 1135–1151.

Stone, M. (1974), 'Cross-validatory Choice and Assessment of Statistical Predictions', *Journal of the Royal Statistical Society, Series B* **36**, 111–147.

Stone, M. (1977), 'An Asymptotic Equivalence of Choice of Model by Cross-validation and Akaike's Criterion', *Journal of the Royal Statistical Society, Series B* **39**, 44–47.

Stone, M. & Brooks, R. (1990), 'Continuum Regression: Cross-Validated Sequentially Constructed Prediction Embracing Ordinary Least Squares, Partial Least Squares and Principal Components Regression (with Discussion)', *Journal of thhe Royal Statistical Society, Series B* **52**, 237–269.

Villa, N. & Rossi, F. (2005), Support Vector Machine for Functional Data Classification, *in* 'Proceedings of ESANN 2005', pp. 467–472.

Wold, H. (1975), Path models with Latent Variables: The NIPALS Approach, *in* H. B. et al., ed., 'Quantitative Sociology: International Perspectives on Mathematical and Statistical Model Building', Academic Press, pp. 307–357.

Wold, H. (1982), Soft Modelling: the Basic Design and some Extensions, *in* K. Jöreskog & H. Wold, eds, 'Systems under Indirect Observation', Vol. 2, North Holland, Amsterdam.

Wold, H. (1985), Partial Least Squares, *in* 'Encyclopedia of Statistical Sciences', J. Wiley and Sons.

Wold, S., Ruhe, H., Wold, H. & III, W. D. (1984), 'The Collinearity Problem in Linear Regression. The Partial Least Squares (PLS) Approach to Generalized Inverses', *SIAM Journal of Scientific and Statistical Computations* **5**, 735–743.

Wood, S. (2000), 'Modelling and Smoothing Parameter Estimation with Multiple Quadratic Penalties', *Journal of the Royal Statistical Society, Series B* **62**, 413–428.

Wood, S. (2006), *Generalized Additive Models: An Introduction with R*, Chapman and Hall.

Yamanishi, Y., Vert, J.-P., Nakaya, A. & Kanehisa, M. (2003), 'Extraction of Correlated Gene Clusters from Multiple Genomic Data by Generalized Kernel Canonical Correlation Analysis', *Bioinformatics* **19**, 323–330.

Zhang, M., Xu, Q. & Masssart, D. (2005), 'Boosting Partial Least Squares', *Analytical Chemistry* **77**, 1423–1431.

Appendix A

Mathematical Background

In this chapter, we briefly summarize some background material from mathematics that is needed throughout this work.

A.1 Matrix Differential Calculus

This section contains results from Magnus & Neudecker (1988).

Definition A.1 (First derivative of vector functions)**.** Let $f : \mathbb{R}^p \to \mathbb{R}^q$ be a function and $\boldsymbol{x} \in \mathbb{R}^p$. If there is a $p \times q$ matrix $\boldsymbol{A}(\boldsymbol{x})$ such that for all $\boldsymbol{h} \in \mathbb{R}^p$ with $\|\boldsymbol{h} - \boldsymbol{c}\| < \varepsilon$

$$f(\boldsymbol{x} + \boldsymbol{h}) \quad = \quad f(\boldsymbol{x}) + \boldsymbol{A}(\boldsymbol{x})\boldsymbol{h} + r_{\boldsymbol{x}}(\boldsymbol{h})$$

and

$$\lim_{\boldsymbol{h} \to 0} \frac{r_{\boldsymbol{x}}(\boldsymbol{h})}{\|\boldsymbol{h}\|} = 0\,,$$

then f is differentiable at point $\boldsymbol{x}$. The matrix $\boldsymbol{A}(\boldsymbol{x})$ is called the first derivative of f at $\boldsymbol{x}$ and is denoted by $\frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x})$.The linear function

$$\begin{aligned} df(\boldsymbol{x}) : \mathbb{R}^p &\to \mathbb{R}^q \\ df(\boldsymbol{x})(\boldsymbol{h}) &= \frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x})\boldsymbol{h} \end{aligned}$$

is called the first differential of f at $\boldsymbol{x}$.

We now have to extend the notion of differentiability to functions defined on matrix spaces.

Definition A.2. Let $\boldsymbol{A}$ be a $m \times n$ matrix and $\boldsymbol{A}_{,j}$ its jth column. Then vec($\boldsymbol{A}$) is defined as the vector

$$\mathrm{vec}(\boldsymbol{A}) = \left(\boldsymbol{A}_{,i}^t, \boldsymbol{A}_{,2}^t, \ldots, \boldsymbol{A}_{,n}^t\right)^t$$

of length mn.

Definition A.3 (First derivative of matrix functions)**.** Let $f : \mathbb{R}^{p\times m} \to \mathbb{R}^{q\times l}$ be a function and $\boldsymbol{X} \in \mathbb{R}^{p\times m}$. f is differentiable at $\boldsymbol{X}$ if and only if the vector-valued function

$$\begin{aligned} F : \mathrm{vec}(\mathbb{R}^{p\times m}) &\to \mathbb{R}^{ql} \\ F(\mathrm{vec}(\boldsymbol{X})) &= \mathrm{vec}(f(\boldsymbol{X})) \end{aligned}$$

is differentiable at vec($\boldsymbol{X}$). The differential

$$df(\boldsymbol{X}) : \mathbb{R}^{p\times m} \to \mathbb{R}^{q\times l}$$

of f at $\boldsymbol{X}$ is a linear function that is defined via the relationship

$$\mathrm{vec}\left(df(\boldsymbol{X})(\boldsymbol{H})\right) = \left(\frac{\partial F}{\partial vec(\boldsymbol{X})}(vec(\boldsymbol{X}))\right) vec(\boldsymbol{H})\,. \qquad \text{(A.1)}$$

$\frac{\partial F}{\partial vec(\boldsymbol{X})}(vec(\boldsymbol{X}))$ is called the derivative of f at $\boldsymbol{X}$.

This theorem is needed in Chapter 3.

Theorem A.4 (Theorem of Schwartz)**.** *Suppose that $f : \mathbb{R}^p \to \mathbb{R}$ is twice differentiable on an open subset $\mathcal{U}$ of $\mathbb{R}^p$. For all $\boldsymbol{x} \in \mathcal{U}$, the Hessian matrix*

$$\boldsymbol{H}_f(\boldsymbol{x}) = \frac{\partial}{\partial \boldsymbol{x}}\frac{\partial}{\partial \boldsymbol{x}} f(\boldsymbol{x}) \in \mathbb{R}^{p\times p}$$

of f is a symmetric matrix.

Proof. The proof can be found in any introductory book on calculus or in Magnus & Neudecker (1988). □

Most of the rules on differential calculus for real-valued functions are also valid for matrix-valued functions and we summarize some of them.

Proposition A.5. *Suppose that f and g are differentiable functions.*

1. *(Product rule)*

   $$d((fg)(\boldsymbol{x})) \;=\; (df(\boldsymbol{x}))g(\boldsymbol{x}) + f(\boldsymbol{x})(d(g(\boldsymbol{x}))\,.$$

   Here, the expression on the right hand side is a short-cut for the map

   $$\boldsymbol{h} \;\mapsto ((df(\boldsymbol{x}))(\boldsymbol{h}))\, g(\boldsymbol{x}) + f(\boldsymbol{x})\,((d(g(\boldsymbol{x}))(\boldsymbol{h}))\;. \tag{A.2}$$

2. *(Differential of an inverse) If $f(\boldsymbol{x})$ is a regular matrix for all $\boldsymbol{x}$, then*

   $$\boldsymbol{d}(f^{-1})(\boldsymbol{x}) \;=\; -(f(\boldsymbol{x}))^{-1}d(f)(\boldsymbol{x})(f(\boldsymbol{x}))^{-1}\,.$$

3. *(Differential of the transpose)*

   $$d(f^t(\boldsymbol{x})) \;=\; (d(f(\boldsymbol{x})))^t\;.$$

These rules are needed in the proof of proposition A.11.

A.2 Optimization under Constraints

In this section, we briefly recapitulate how to optimize with side constraints. Suppose that we want to compute

$$\arg\max \qquad f(\boldsymbol{w})\,, \tag{A.3}$$
$$\text{subject to} \qquad g_k(\boldsymbol{w}) = 0\,,\; k = 1\,\ldots, K\,. \tag{A.4}$$

Here $f, g_1, \ldots, g_k$ are real-valued functions $\mathbb{R}^p \to \mathbb{R}$. We assume that f and $g_1, \ldots, g_K$ are differentiable. We define the Lagrangian function

$$L(\boldsymbol{w}) \;=\; f(\boldsymbol{w}) - \sum_{k=1}^{K} \lambda_k g_k(\boldsymbol{w})\,. \tag{A.5}$$

The values λ_k are called Lagrangian multipliers.

Proposition A.6. *Any solution $\boldsymbol{w}^*$ of (A.3) and (A.4) fulfills the Lagrangian* *equations*

$$\left(\frac{\partial f}{\partial \boldsymbol{w}}\right)(\boldsymbol{w}^*) = \sum_{i=1}^{k} \lambda_i \left(\frac{\partial g_i}{\partial \boldsymbol{w}}\right)(\boldsymbol{w}^*), \tag{A.6}$$

$$g_i(\boldsymbol{w}^*) = 0. \tag{A.7}$$

A.3 Eigen Problems and the Singular Value Decomposition

Any matrix $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ defines a linear map $\mathbb{R}^p \to \mathbb{R}^n$ via $\boldsymbol{w} \mapsto \boldsymbol{X}\boldsymbol{w}$. The singular value decomposition of $\boldsymbol{X}$ is a representation of this map in terms of orthonormal basis vectors for both $\mathbb{R}^p$ and $\mathbb{R}^n$ such that the map defined by $\boldsymbol{X}$ is as simple as possible.

Proposition A.7 (Singular Value Decomposition)**.** *For any matrix $\boldsymbol{X} \in \mathbb{R}^{n\times p}$, there is an orthonormal basis $\boldsymbol{u}_1, \ldots \boldsymbol{u}_p$ of $\mathbb{R}^p$ and a set of orthonormal vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_p \in \mathbb{R}^n$ such that*

$$\boldsymbol{X}\boldsymbol{u}_i = \sigma_i \boldsymbol{v}_i\,,\ \sigma_i \geq 0\,.$$

The quantities σ_i are called the singular values of $\boldsymbol{X}$ and are numbered in decreasing order. In matrix notation, we have

$$\boldsymbol{X} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^t \tag{A.8}$$

with

$$\boldsymbol{V}^t\boldsymbol{V} = \boldsymbol{I}_p \quad \textit{and} \quad \boldsymbol{U}^t\boldsymbol{U} = \boldsymbol{I}_p\,.$$

It follows immediately that the rank of $\boldsymbol{X}$ equals the number of nonzero singular values (counted with multiplicities). We note that $\boldsymbol{V}$ is a basis of the column space of $\boldsymbol{X}$ and $\boldsymbol{U}$ is a basis of the row space of $\boldsymbol{X}$. We can extend the vectors $\boldsymbol{v}_i$ to an orthonormal basis of $\mathbb{R}^n$.

Definition A.8. A vector $\boldsymbol{u} \in \mathbb{R}^p \setminus \{0\}$ is called an eigenvector of a quadratic matrix $\boldsymbol{A} \in \mathbb{R}^{p\times p}$ if there is a scalar $\lambda \in \mathbb{R}$ such that $\boldsymbol{A}\boldsymbol{u} = \lambda\boldsymbol{u}$. We call λ an eigenvalue of $\boldsymbol{A}$.

The eigendecomposition of $\boldsymbol{A}$ is a representation of the form

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1}\,.$$

For some matrices, there is no eigendecomposition. If $\boldsymbol{A}$ is however symmetric, we have an orthogonal eigendecomposition

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^t \quad , \quad \boldsymbol{U}^t\boldsymbol{U} = \boldsymbol{I}_p \, .$$

The eigenvectors of a symmetric matrix can be computed with the help of the so-called power method.

Algorithm A.9 (Power method)**.** *For a symmetric matrix $\boldsymbol{A}$ and an initial vector $\boldsymbol{b}_0$, the power method computes iteratively*

$$\begin{array}{rcll} \widetilde{\boldsymbol{b}}_{k+1} & = & \boldsymbol{A}\boldsymbol{b}_k & \textit{matrix multiplication} \\ \boldsymbol{b}_{k+1} & = & \frac{1}{\|\widetilde{\boldsymbol{b}}_{k+1}\|}\widetilde{\boldsymbol{b}}_{k+1} & \textit{normalization} \end{array}$$

The power algorithm converges to the eigenvector $\boldsymbol{u}$ for which the corresponding eigenvalue has the greatest absolute value, if this eigenvalue is dominant (in absolute terms) and if the starting vector $\boldsymbol{b}_0$ is not orthogonal on the eigenvector $\boldsymbol{u}$.

A.4 Projections

Let us consider a general Hibert space $\mathcal{V}$. For a subspace $\mathcal{U}$ and any vector $\boldsymbol{v} \in \mathcal{V}$, we define the following optimization problem:

$$\begin{array}{rl} \arg\min & \|\boldsymbol{v} - \boldsymbol{u}\| \, , \\ \text{subject to} & \boldsymbol{u} \in \mathcal{U} \, . \end{array}$$

As we assume that $\mathcal{V}$ is a Hilbert space, the solution exists if $\mathcal{U}$ is a closed subspace. We call the unique solution the (orthogonal) projection of $\boldsymbol{v}$ onto $\mathcal{U}$ and denote it by $\mathcal{P}_{\mathcal{U}}\boldsymbol{v}$.

If $\mathcal{U}$ is finite-dimensional, we can give a short representation of the projection operator. Denote by $\boldsymbol{U} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k)$ any set of vectors that generate the subspace $\mathcal{U}$. For any other set $\boldsymbol{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_l)$ of vectors we define the $k \times l$ matrix

$$\langle \boldsymbol{U}, \boldsymbol{V} \rangle \;=\; (\langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle) \; .$$

Furthermore, we define the (symbolic) multiplication of $\boldsymbol{U}$ with a vector $\boldsymbol{\alpha} \in \mathbb{R}^k$ as

$$\boldsymbol{U}\boldsymbol{\alpha} = \sum_{i=1}^{k} \alpha_i \boldsymbol{u}_i \,.$$

The projection map is then

$$\mathcal{P}_{\boldsymbol{U}}\boldsymbol{v} = \boldsymbol{U}\left(\langle \boldsymbol{U}, \boldsymbol{U}\rangle\right)^{-} \langle \boldsymbol{U}, \boldsymbol{v}\rangle \,. \tag{A.9}$$

We now list some properties of projection operators.

Proposition A.10. *Denote by $\mathcal{P}_{\mathcal{U}}$ the projection onto the subspace $\mathcal{U}$.*

1. *$\mathcal{P}_{\mathcal{U}}$ is a symmetric map.*

2. *The projection operator is idempotent, $\mathcal{P}_{\mathcal{U}}^2 \equiv \mathcal{P}_{\mathcal{U}}$.*

3. *If the space $\mathcal{U}^{\perp}$ that is orthogonal on $\mathcal{U}$ is a closed subspace, then $(Id_{\mathcal{V}} - \mathcal{P})$ is the projection onto that space.*

4. *If $\mathcal{V}$ is finite-dimensional and $\mathcal{P}_{\mathcal{U}}$ can be represented by a matrix $\boldsymbol{P}$, then trace$(\boldsymbol{P}) = \dim \mathcal{U}$.*

In Chapter 4, we need the first derivative of a projection operator. We now present this result. Let us assume that both vectors $\boldsymbol{v} = \boldsymbol{v}(\boldsymbol{y}), \boldsymbol{z} = \boldsymbol{z}(\boldsymbol{y}) \in \mathbb{R}^n$ depend on a vector $\boldsymbol{y}$. The projection of $\boldsymbol{z}$ onto $\boldsymbol{v}$ is defined as (see (A.9))

$$\mathcal{P}_{\boldsymbol{v}}\boldsymbol{z} = \boldsymbol{v}\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}\boldsymbol{v}^t\boldsymbol{z} \,.$$

For any function f that depends on $\boldsymbol{y}$, we use $df = df(\boldsymbol{y})$ as a shortcut. Using proposition A.5, we have

$$\begin{aligned}
d\left(\mathcal{P}_{\boldsymbol{v}}\boldsymbol{z}\right) &= d\left(\boldsymbol{v}\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}\boldsymbol{v}^t\boldsymbol{z}\right) \\
&= \left(d\boldsymbol{v}\right)\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}\boldsymbol{v}^t\boldsymbol{z} + \boldsymbol{v}\left(d\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}\right)\boldsymbol{v}^t\boldsymbol{z} + \boldsymbol{v}\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}d\left(\boldsymbol{v}^t\boldsymbol{z}\right) \\
&= \left(d\boldsymbol{v}\right)\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}\boldsymbol{v}^t\boldsymbol{z} - \boldsymbol{v}\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}d\left(\boldsymbol{v}^t\boldsymbol{v}\right)\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}\boldsymbol{v}^t\boldsymbol{z} \\
&\quad + \boldsymbol{v}\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}\left[d\left(\boldsymbol{v}^t\right)\boldsymbol{z} + \boldsymbol{v}^t d\boldsymbol{z}\right] \\
&= \left(d\boldsymbol{v}\right)\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}\boldsymbol{v}^t\boldsymbol{z} - \boldsymbol{v}\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}d\left(\boldsymbol{v}^t\right)\boldsymbol{v}\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}\boldsymbol{v}^t\boldsymbol{z} \\
&\quad - \boldsymbol{v}\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}\boldsymbol{v}^t d\boldsymbol{v}\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}\boldsymbol{v}^t\boldsymbol{z} + \boldsymbol{v}\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}d\left(\boldsymbol{v}^t\right)\boldsymbol{z} + \boldsymbol{v}\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}\boldsymbol{v}^t d\boldsymbol{z} \\
&= \left(d\boldsymbol{v}\right)\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}\boldsymbol{v}^t\boldsymbol{z} - \boldsymbol{v}\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}d\left(\boldsymbol{v}^t\right)\mathcal{P}_{\boldsymbol{v}}\boldsymbol{z} - \mathcal{P}_{\boldsymbol{v}}d\boldsymbol{v}\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}\boldsymbol{v}^t\boldsymbol{z} \\
&\quad + \boldsymbol{v}\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}d\left(\boldsymbol{v}^t\right)\boldsymbol{z} + \mathcal{P}_{\boldsymbol{v}}d\boldsymbol{z} \\
&= \left(d\boldsymbol{v}\right)\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}\boldsymbol{v}^t\boldsymbol{z} - \boldsymbol{v}\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}d\left(\boldsymbol{v}\right)^t\mathcal{P}_{\boldsymbol{v}}\boldsymbol{z} - \mathcal{P}_{\boldsymbol{v}}d\boldsymbol{v}\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}\boldsymbol{v}^t\boldsymbol{z} \\
&\quad + \boldsymbol{v}\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}d\left(\boldsymbol{v}\right)^t\boldsymbol{z} + \mathcal{P}_{\boldsymbol{v}}d\boldsymbol{z}
\end{aligned}$$

Using (A.2), this is equivalent to the following. For all $\boldsymbol{h} \in \mathbb{R}^n$,

$$\begin{aligned}
\left(d\left(\mathcal{P}_{\boldsymbol{v}}\boldsymbol{z}\right)\right)\boldsymbol{h} &= \left(\left(d\boldsymbol{v}\right)\boldsymbol{h}\right)\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}\boldsymbol{v}^t\boldsymbol{z} - \boldsymbol{v}\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}\left(d\left(\boldsymbol{v}\right)\boldsymbol{h}\right)^t\mathcal{P}_{\boldsymbol{v}}\boldsymbol{z} \\
&\quad - \mathcal{P}_{\boldsymbol{v}}\left(d\boldsymbol{v}\boldsymbol{h}\right)\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}\boldsymbol{v}^t\boldsymbol{z} + \boldsymbol{v}\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}\left(d\left(\boldsymbol{v}\right)\boldsymbol{h}\right)^t\boldsymbol{z} \\
&\quad + \mathcal{P}_{\boldsymbol{v}}d\boldsymbol{z}\boldsymbol{h}\,.
\end{aligned}$$

This can be further simplified by factoring out the expression $\left(\boldsymbol{v}^t\boldsymbol{v}\right)^{-1}$ and rearranging some terms. We obtain

$$\begin{aligned}
\left(d\left(\mathcal{P}_{\boldsymbol{v}}\boldsymbol{z}\right)\right)\boldsymbol{h} &= \frac{1}{\boldsymbol{v}^t\boldsymbol{v}}\left[\boldsymbol{v}^t\boldsymbol{z}\left(\left(d\boldsymbol{v}\right)\boldsymbol{h}\right) - \boldsymbol{v}\boldsymbol{z}^t\mathcal{P}_v^t\left(\left(d\boldsymbol{v}\right)\boldsymbol{h}\right) - \boldsymbol{v}^t\boldsymbol{z}\mathcal{P}_v\left(\left(d\boldsymbol{v}\right)\boldsymbol{h}\right) + \boldsymbol{v}\boldsymbol{z}^t\left(\left(d\boldsymbol{v}\right)\boldsymbol{h}\right)\right] \\
&\quad + \mathcal{P}_{\boldsymbol{v}}d\boldsymbol{z}\boldsymbol{h} \\
&= \frac{1}{\boldsymbol{v}^t\boldsymbol{v}}\left[\boldsymbol{v}^t\boldsymbol{z} - \boldsymbol{v}\boldsymbol{z}^t\mathcal{P}_v - \boldsymbol{v}^t\boldsymbol{z}\mathcal{P}_v + \boldsymbol{v}\boldsymbol{z}^t\right]\left(\left(d\boldsymbol{v}\right)\boldsymbol{h}\right) + \mathcal{P}_{\boldsymbol{v}}d\boldsymbol{z}\boldsymbol{h}\,.
\end{aligned}$$

Finally, we use the definition of the first derivative A.1 and obtain the following result.

Proposition A.11. *The first derivate of the projection operator is*

$$\frac{\partial\mathcal{P}_{\boldsymbol{v}}\boldsymbol{z}}{\partial\boldsymbol{y}} = \frac{1}{\boldsymbol{v}^t\boldsymbol{v}}\left[\boldsymbol{v}\boldsymbol{z}^t\left(\boldsymbol{I} - \mathcal{P}_{\boldsymbol{v}}\right) + \boldsymbol{v}^t\boldsymbol{z}\left(\boldsymbol{I} - \mathcal{P}_{\boldsymbol{v}}\right)\right]\frac{\partial\boldsymbol{v}}{\partial\boldsymbol{y}} + \mathcal{P}_{\boldsymbol{z}}\frac{\partial\boldsymbol{z}}{\partial\boldsymbol{y}}\,.$$

A.5 The Moore-Penrose Inverse

The contents of this section can be found e.g. in Kockelkorn (2000). If a matrix $\boldsymbol{A}$ is not quadratic or is not of full rank, we have to find a suitable surrogate for its inverse. In this work, we use the Moore-Penrose inverse.

Proposition A.12 (Moore-Penrose Inverse)**.** *For any matrix $\boldsymbol{A} \in \mathbb{R}^{p\times l}$, there is a unique matrix $\boldsymbol{A}^- \in \mathbb{R}^{l\times p}$ such that*

$$\begin{aligned} \boldsymbol{A} &= \boldsymbol{A}\boldsymbol{A}^-\boldsymbol{A}\,, \\ \boldsymbol{A}^- &= \boldsymbol{A}^-\boldsymbol{A}\boldsymbol{A}^-\,, \\ \left(\boldsymbol{A}\boldsymbol{A}^-\right)^t &= \boldsymbol{A}\boldsymbol{A}^-\,, \\ \left(\boldsymbol{A}^-\boldsymbol{A}\right)^t &= \boldsymbol{A}^-\boldsymbol{A}\,. \end{aligned}$$

Proposition A.13. *If $\boldsymbol{A}$ is a symmetric matrix with eigendecomposition*

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^t\,,$$

the Moore-Penrose inverse of $\boldsymbol{A}$ is defined in the following way. Set

$$\left(\boldsymbol{\Lambda}^-\right)_{ij} = \begin{cases} 0 & i \neq j \\ \frac{1}{\lambda_i} & i = j \text{ and } \lambda_i \neq 0 \\ 0 & i = j \text{ and } \lambda_i = 0 \end{cases} .$$

Then

$$\boldsymbol{A}^- = \boldsymbol{U}\boldsymbol{\Lambda}^-\boldsymbol{U}^t\,.$$

Proof. It follows readily from the definiton of $\boldsymbol{\Lambda}^-$ that

$$\boldsymbol{\Lambda}\boldsymbol{\Lambda}^- = \boldsymbol{\Lambda}^-\boldsymbol{\Lambda} = \mathrm{diag}(\underbrace{1,\ldots,1}_{\mathrm{rk}(\boldsymbol{A})-\mathrm{times}},0,\ldots,0)\,.$$

This implies that $\boldsymbol{\Lambda}^-$ is indeed the Moore-Penrose inverse of $\boldsymbol{\Lambda}$, as the properties in proposition A.12 are fulfilled. It follows that

$$\boldsymbol{A}\boldsymbol{U}\boldsymbol{\Lambda}^-\boldsymbol{U}^t\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{\Lambda}^-\boldsymbol{\Lambda}\boldsymbol{U}^t = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^t = \boldsymbol{A}$$

and

$$\boldsymbol{U}\boldsymbol{\Lambda}^{-}\boldsymbol{U}^{t}\boldsymbol{A}\boldsymbol{U}\boldsymbol{\Lambda}^{-}\boldsymbol{U}^{t} = \boldsymbol{U}\boldsymbol{\Lambda}^{-}\boldsymbol{\Lambda}\boldsymbol{\Lambda}^{-}\boldsymbol{U}^{t} = \boldsymbol{U}\boldsymbol{\Lambda}^{-}\boldsymbol{U}^{t}\,.$$

Finally, we remark that the matrix

$$\boldsymbol{A}\boldsymbol{U}\boldsymbol{\Lambda}^{-}\boldsymbol{U}^{t} = \boldsymbol{U}\boldsymbol{\Lambda}^{-}\boldsymbol{U}^{t}\boldsymbol{A} = \boldsymbol{U}\mathrm{diag}(1,\ldots,1,0,\ldots,0)\boldsymbol{U}^{t}$$

is symmetric. □

Proposition A.14. *The system of linear equations*

$$\boldsymbol{A}\boldsymbol{x} \;=\; \boldsymbol{b}$$

has a solution if and only if $\boldsymbol{x}^{*} = \boldsymbol{A}^{-}\boldsymbol{b}$ *is a solution. Any solution of these linear equations has the form*

$$\boldsymbol{x} \;=\; \boldsymbol{x}^{*} + \left(\boldsymbol{I} - \boldsymbol{A}^{-}\boldsymbol{A}\right)\boldsymbol{v}$$

for any vector $\boldsymbol{v}$*. The two components of* $\boldsymbol{x}$ *are orthogonal.*

Appendix B

Results of the Simulation Study

We display the results of the simulation study that is described in Section 7.3. The following tables show the MSE-RATIO for $\widehat{\boldsymbol{\beta}}$ as well as for $\widehat{\boldsymbol{y}}$. In addition to the MSE-RATIO, we display the optimal number of components for each method. It is interesting to see that the two quantities are the same almost all of the times.

collinearity	no	no	no	med.	med.	med.	high	high	high
stnr	1	3	7	1	3	7	1	2	7
1	0.833	0.861	0.676	0.958	1.000	0.993	1.000	0.999	1.000
2	0.980	0.976	0.975	0.995	0.938	0.864	0.847	0.965	0.866
3	1.000	0.993	1.001	0.969	0.960	0.993	0.954	0.980	0.967
4	1.000	1.001	0.999	0.988	1.000	1.002	0.997	0.993	0.992
m_{PLS}^{opt}	2	5	2	2	4	3	1	2	5
m_{TRN}^{opt}	2	5	2	2	4	3	1	2	5

Table B.1: MSE-RATIO of $\widehat{\boldsymbol{\beta}}$ for $p = 5$. The first two rows display the setting of the parameters. The rows entitled 1-4 display the MSE ratio for the respective number of components.

collinearity	no	no	no	med.	med.	med.	high	high	high
stnr	1	3	7	1	3	7	1	2	7
1	0.775	0.780	0.570	0.919	1.000	0.970	1.004	0.995	0.999
2	0.978	0.972	0.9697	0.994	0.882	0.786	0.828	0.951	0.823
3	1.001	0.990	1.001	0.969	0.967	0.992	0.960	0.977	0.973
4	1.000	1.001	0.999	0.990	1.000	1.001	0.997	0.996	0.993
m_{PLS}^{opt}	3	5	3	2	4	4	1	2	5
m_{TRN}^{opt}	2	5	2	2	4	4	1	2	3

Table B.2: MSE-RATIO of $\widehat{\boldsymbol{y}}$ for $p = 5$.

collinearity	no	no	no	med.	med.	med.	high	high	high
stnr	1	3	7	1	3	7	1	2	7
1	0.929	0.963	0.972	0.98	0.998	0.989	1.000	1.000	1.000
2	0.938	0.959	0.977	0.922	0.91	0.978	0.789	0.793	0.792
3	0.907	0.952	0.981	0.875	0.91	0.945	0.849	0.843	0.849
4	0.905	0.933	0.971	0.879	0.913	0.912	0.857	0.864	0.868
5	0.901	0.942	0.954	0.879	0.924	0.898	0.870	0.883	0.879
6	0.898	0.942	0.945	0.878	0.915	0.891	0.882	0.890	0.893
7	0.892	0.926	0.949	0.887	0.906	0.891	0.891	0.895	0.898
8	0.899	0.926	0.956	0.892	0.904	0.895	0.897	0.897	0.903
9	0.908	0.933	0.955	0.897	0.910	0.895	0.903	0.902	0.904
10	0.913	0.938	0.951	0.900	0.916	0.898	0.902	0.899	0.901
11	0.913	0.937	0.947	0.902	0.919	0.907	0.906	0.901	0.902
12	0.917	0.931	0.944	0.909	0.919	0.917	0.908	0.904	0.906
13	0.924	0.932	0.946	0.919	0.92	0.925	0.913	0.907	0.914
14	0.933	0.939	0.946	0.927	0.917	0.936	0.921	0.911	0.922
15	0.94	0.945	0.95	0.933	0.916	0.936	0.928	0.916	0.931
16	0.949	0.945	0.951	0.935	0.918	0.941	0.938	0.922	0.936
17	0.956	0.945	0.954	0.939	0.922	0.945	0.944	0.926	0.936
18	0.961	0.944	0.959	0.943	0.931	0.95	0.946	0.930	0.935
19	0.968	0.946	0.964	0.946	0.934	0.958	0.953	0.939	0.936
20	0.973	0.951	0.973	0.949	0.935	0.962	0.961	0.947	0.939
21	0.977	0.958	0.977	0.954	0.936	0.966	0.968	0.955	0.943
22	0.98	0.965	0.981	0.961	0.94	0.973	0.972	0.962	0.948
23	0.984	0.97	0.984	0.968	0.945	0.98	0.976	0.967	0.950
24	0.987	0.976	0.988	0.975	0.948	0.983	0.98	0.970	0.953
25	0.989	0.98	0.99	0.978	0.953	0.987	0.981	0.973	0.959
26	0.992	0.985	0.993	0.982	0.959	0.991	0.984	0.977	0.966
27	0.994	0.989	0.996	0.986	0.966	0.992	0.987	0.981	0.975
28	0.995	0.991	0.997	0.988	0.973	0.994	0.99	0.985	0.984
29	0.996	0.993	0.998	0.99	0.978	0.995	0.993	0.988	0.988
30	0.997	0.994	0.999	0.992	0.982	0.996	0.995	0.991	0.99
31	0.998	0.995	0.999	0.994	0.985	0.997	0.996	0.992	0.993
32	0.998	0.996	0.999	0.996	0.99	0.998	0.996	0.994	0.995
33	0.999	0.997	1.000	0.996	0.991	0.999	0.997	0.995	0.996
34	0.999	0.998	1.000	0.997	0.993	0.999	0.998	0.996	0.997
35	0.999	0.999	1.000	0.999	0.994	0.999	0.998	0.997	0.998
36	1.000	1.000	1.000	0.999	0.996	0.999	0.999	0.998	0.998
37	1.000	1.000	1.000	0.999	0.998	1.000	0.999	0.998	0.999
38	1.000	1.000	1.000	1.000	0.999	1.000	0.999	0.999	0.999
39	1.000	1.000	1.000	1.000	0.999	1.000	1.000	0.999	1.000
m^{opt}_{PLS}	1	3	5	1	2	2	1	1	1
m^{opt}_{TRN}	1	3	5	1	2	2	1	1	1

Table B.3: MSE-RATIO of $\widehat{\boldsymbol{\beta}}$ for $p = 40$.

collinearity	no	no	no	med.	med.	med.	high	high	high
stnr	1	3	7	1	3	7	1	2	7
1	0.781	0.797	0.791	0.877	0.983	0.924	1.013	1.004	1.001
2	0.870	0.857	0.853	0.785	0.702	0.868	0.673	0.684	0.680
3	0.853	0.899	0.914	0.776	0.818	0.853	0.778	0.772	0.778
4	0.874	0.891	0.896	0.818	0.836	0.838	0.81	0.818	0.822
5	0.889	0.92	0.893	0.839	0.891	0.846	0.835	0.855	0.856
6	0.898	0.942	0.921	0.844	0.884	0.859	0.862	0.881	0.881
7	0.897	0.938	0.929	0.876	0.902	0.88	0.886	0.898	0.898
8	0.923	0.941	0.943	0.886	0.898	0.896	0.9	0.906	0.914
9	0.924	0.944	0.960	0.904	0.916	0.901	0.915	0.92	0.917
10	0.935	0.958	0.961	0.913	0.93	0.915	0.915	0.914	0.921
11	0.943	0.967	0.959	0.922	0.937	0.916	0.924	0.92	0.927
12	0.954	0.967	0.958	0.929	0.942	0.938	0.932	0.926	0.931
13	0.959	0.967	0.965	0.941	0.95	0.942	0.939	0.933	0.937
14	0.961	0.961	0.966	0.948	0.949	0.954	0.947	0.942	0.942
15	0.97	0.969	0.977	0.954	0.953	0.96	0.953	0.948	0.949
16	0.975	0.971	0.976	0.964	0.962	0.967	0.961	0.954	0.957
17	0.979	0.976	0.983	0.968	0.962	0.974	0.967	0.957	0.957
18	0.982	0.981	0.985	0.972	0.966	0.979	0.968	0.960	0.966
19	0.986	0.985	0.988	0.976	0.969	0.980	0.974	0.965	0.970
20	0.989	0.987	0.991	0.977	0.970	0.983	0.979	0.972	0.974
21	0.991	0.99	0.992	0.980	0.973	0.985	0.984	0.977	0.978
22	0.993	0.99	0.994	0.984	0.979	0.988	0.988	0.982	0.981
23	0.995	0.992	0.996	0.987	0.98	0.991	0.990	0.986	0.983
24	0.996	0.993	0.997	0.989	0.982	0.993	0.992	0.987	0.984
25	0.996	0.995	0.997	0.99	0.983	0.994	0.993	0.989	0.985
26	0.997	0.996	0.998	0.992	0.986	0.996	0.994	0.991	0.987
27	0.998	0.997	0.999	0.994	0.990	0.997	0.995	0.993	0.989
28	0.999	0.997	0.999	0.995	0.991	0.998	0.996	0.994	0.991
29	0.999	0.998	0.999	0.996	0.992	0.998	0.997	0.996	0.994
30	0.999	0.999	1.000	0.997	0.993	0.999	0.998	0.997	0.994
31	0.999	0.999	1.000	0.998	0.994	0.999	0.998	0.997	0.996
32	0.999	0.999	1.000	0.998	0.995	0.999	0.999	0.998	0.997
33	1.000	0.999	1.000	0.998	0.996	0.999	0.999	0.998	0.998
34	1.000	0.999	1.000	0.999	0.997	1.000	0.999	0.999	0.998
35	1.000	1.000	1.000	0.999	0.998	1.000	0.999	0.999	0.999
36	1.000	1.000	1.000	0.999	0.998	1.000	1.000	0.999	0.999
37	1.000	1.000	1.000	1.000	0.998	1.000	1.000	0.999	0.999
38	1.000	1.000	1.000	1.000	0.999	1.000	1.000	0.999	1.000
39	1.000	1.000	1.000	1.000	0.999	1.000	1.000	1.000	1.000
m_{PLS}^{opt}	1	3	9	1	1	2	1	1	1
m_{TRN}^{opt}	1	3	4	1	2	2	1	1	1

Table B.4: MSE-RATIO of $\widehat{\boldsymbol{y}}$ for $p = 40$.

collinearity	no	no	no	med.	med.	med.	high	high	high
stnr	1	3	7	1	3	7	1	2	7
1	0.845	0.763	0.825	0.862	0.839	0.906	1.017	1.002	1.000
2	0.813	0.892	0.864	0.753	0.832	0.847	0.695	0.695	0.694
3	0.863	0.884	0.881	0.788	0.837	0.859	0.806	0.808	0.812
4	0.908	0.898	0.918	0.852	0.852	0.864	0.866	0.861	0.865
5	0.933	0.940	0.954	0.889	0.888	0.900	0.900	0.903	0.903
6	0.954	0.953	0.960	0.900	0.905	0.915	0.926	0.931	0.931
7	0.964	0.967	0.979	0.927	0.935	0.930	0.951	0.953	0.953
8	0.976	0.972	0.988	0.942	0.942	0.950	0.968	0.968	0.970
9	0.984	0.982	0.993	0.959	0.963	0.961	0.979	0.968	0.979
10	0.99	0.990	0.995	0.969	0.970	0.970	0.980	0.979	0.981
11	0.994	0.991	0.998	0.976	0.979	0.978	0.987	0.987	0.988
12	0.996	0.994	0.998	0.982	0.987	0.984	0.991	0.992	0.993
13	0.997	0.995	0.999	0.988	0.991	0.990	0.994	0.994	0.995
14	0.998	0.997	0.999	0.991	0.994	0.992	0.996	0.997	0.997
15	0.999	0.998	1.000	0.994	0.995	0.994	0.997	0.998	0.998
16	0.999	0.999	1.000	0.996	0.997	0.996	0.998	0.998	0.999
17	1.000	0.999	1.000	0.997	0.998	0.997	0.999	0.999	0.999
18	1.000	0.999	1.000	0.998	0.999	0.998	0.999	0.999	0.999
19	1.000	1.000	1.000	0.999	0.999	0.999	0.999	1.000	1.000
20	1.000	1.000	1.000	0.999	0.999	0.999	1.000	1.000	1.000
m_{PLS}^{opt}	1	2	5	1	1	2	1	1	1
m_{TRN}^{opt}	1	1	3	1	1	2	1	1	1

Table B.5: MSE-RATIO of $\widehat{\boldsymbol{y}}$ for $p = 100$. We only display the results for the first 20 components, as the MSE-RATIO equals 1 (up to 4 digits after the decimal point) for the remaining components.