2016-01-01

Pre-tuned Principal Component Regression and Several Variants

Pei Wang
University of Texas at El Paso, wangpeinihao@gmail.com

PRE-TUNED PRINCIPAL COMPONENT REGRESSION AND SEVERAL VARIANTS

PEI WANG

Master's Program in Mathematical Sciences

APPROVED:

Xiaogang Su, Ph.D., Chair

Behzad Djafari-Rouhani, Ph.D.

Panagis Moschopoulos, Ph.D.

Feng Yang, Ph.D.

Charles Ambler, Ph.D.
Dean of the Graduate School

to my

MOTHER and FATHER

with love

PRE-TUNED PRINCIPAL COMPONENT REGRESSION AND SEVERAL VARIANTS

by

PEI WANG

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Master's Program in Mathematical Sciences

THE UNIVERSITY OF TEXAS AT EL PASO

May 2016

Acknowledgements

I would like to express my deep-felt gratitude to my advisor, Dr. Xiaogang Su from Mathematical Sciences at University of Texas at El Paso for his advice, encouragement, enduring patience and constant support. Without his guide, I am not able to make this thesis a successful completion. He was never ceasing in his belief in me, always providing clear explanations when I was lost, constantly driving me with energy when I was tired, and always, giving me his time, in spite of anything else that was going on.

I also wish to thank the other members of my committee, Dr. Panagis Moschopoulos from the Mathematical Sciences Department, Dr. Behzad Djafari-Rouhani from the Mathematical Sciences Department as well and Dr. Feng Yang from the Kinesiology Department, all at The University of Texas at El Paso. Their guidance, comments and assistance are invaluable to the completion of this work.

Additionally, I want to thank all the professors and staff from Mathematical Sciences, The University of Texas at El Paso for all their hard work and dedication, providing me the means to complete my degree.

Last but not least, I want to thank my dear parents and sisters for their love, encouragement and firm support. It is impossible for me to continue my graduate study without them.

NOTE: This thesis was submitted to my Supervising Committee on the May 6, 2016.

Abstract

The regression coefficient estimates from ordinary least squares (OLS) have a low probability of being close to the real value when there is a multicollinearity problem in the design matrix. In order to combat this problem, many regularized methods have been introduced.

Principal components regression (PCR) is an important analysis tool for dealing with multicollinearity and high-dimensionality. In conventional PCR, the first step is to change the original predictors to orthogonal principal components (PC's) by a linear transformation. These PC's correspond to the eigenvalues which are sorted in a decreasing order. The next step is to regress the response on a number of the PC's and to compute the model selection criteria such as AIC, BIC, GCV for each model. The final step is to compare the criteria values and choose the model corresponding to the smallest value. However, the traditional way of doing PCR is quite computationally inefficient. Thus, we proposed three competitive models to overcome this problem. In these proposed methods, the number of PC's can be automatically determined. The main idea involves approximation of the indicator threshold function with a smooth sigmoid surrogate function, yet in several different ways.

In our first model PTPCR, we used the logistic function with a large fixed shape parameter and an undetermined threshold parameter to replace the indicate function. Then the selection criterion can be treated as an objective function for optimization to estimate the threshold parameter. The PC's to be included in the final model can be obtained by selecting those with eigenvalues greater than the estimated threshold parameter. This reformulation facilitates direct estimation of the best number of PC's, leading to much improved computational efficiency.

Apart from the PTPCR, we proposed another two models: PTPCR-V1 and

PTPCR-V2. In PTPCR-V1, we free the shape parameter in the logistic function. Then we optimized the criterion function with respect to both shape and threshold parameters. PTPCR-V2 is fit in a similar manner to PTPCR, except for that the preference order of PC's is now based on the regression coefficients in the PTPCR-V2.

On the basis of extensive simulation studies, all our three proposed models perform better than PCR. More specifically, PTPCR yields a similar predictive performance to PCR yet with a shorter computing time, while PTPCR-V1 and PTPCR-V2 outperform PCR not only in terms of computational efficiency, but also in prediction accuracy.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Regression analysis is concerned about exploring relationships among variables. As one of the most popular methods, linear regression assumes a linear functional association between the dependent variable (or response) and the independent variables (or explanatory variables). Linear regression has been broadly applied in a variety of areas such as geomorphology, chemistry, biology and so on. One main concern in linear regression is how to estimate the unknown regression coefficients given a set of data. The least squares method, which minimizes the sum of squared differences between the observed response values and the fitted values, is one of the most popular estimation methods. Among other appealing properties, the ordinary least square (OLS) estimates is the best linear unbiased estimate (BLUE), meaning that it yields the minimum variance among all linear unbiased estimators.

In OLS, the key to have a good estimates is that the inverse of the gram matrix exists. However, in real life, multicollinearity is very common especially in high dimensional data. When independent variables are highly correlated, the inverse of gram matrix is hard to compute numerically. Or even worse, when design matrix is not full rank, the gram matrix becomes non-invertible or singular. Under these scenarios, the estimates of regression coefficients are no longer unique or numerically unstable and the variance of the coefficient estimates tends to be inflated. In addition, the model is often over-fitted due to the redundant predictors included. This not only leads to a lack of fit but also complicates the model interpretation, rendering

the model less useful in assessing the relationship between independent variables and dependent variable. It is worth noting that the multicollinearity may not be a big concern for predictive modeling purposes only (Makridakis, Wheelwright and Hyndman[18]).

To overcome the limitations of OLS, several alternative methods were introduced. For example, Hoerl and Kennard (1970) (Hoerl and Kennard[11]) introduced the ridge regression, which adds a positive constant to the diagonal of the non-negative definite gram matrix to ensure its invertibility. Partial least square regression, invented by Wold in 1975 (Wold[40]), extracts orthogonal components by taking into consideration both the variance-covariance structure among predictors and their association with the response. Massy proposed principal component regression (PCR) in 1965(Massy[25]). As its name suggests, PCR regresses the response on the principal components, instead of the original independent variables directly. Highly competitive to the ridge and PLS regression in terms of prediction accuracy, PCR has a simpler structure that is easier to understand and extra distributional theory is available from principal component analysis (PCA; Mansfield[23]).

1.2 Principal Component Regression

Principal component regression is a great tool in many areas such as marketing, business, industry, economics, statistics, chemistry especially those where multicollinear or high dimensional data are frequently encountered. For example, PCR was used to predict the rainfall because multicollinear data are common in rainfall forecast (Sahriman, Djuraidah and Wigena[34]). Another example is chemical analysis. Because of the large number of variables and few observations, principal component regression helps handle the deficient rank problem (Mevik and Wehrens[26]). Also, in ranking sport players, principal component regression is utilized because the sports performance measures are often highly correlated (Manage and Scariano[20]).

PCR starts with extracting the principal components (PC) from the original data, which is referred to as principal components analysis (PCA). If we transform the lousy data first, the inherent relationship among variables may be better revealed and easily explored (Massy[25]). In addition, the resultant principal components, which are linear transformations of original predictors, are interpretable in the spirit of latent variables. The main objective of extracting principal components is to get uncorrelated linear combinations that explain as much information as possible from the original data.

The general criterion is to keep as much the combined variance from the original data. To this end, the principal components are routinely sorted in a decreasing order of the variance each explains. The first component contains the most variance, followed by the second component that accounts for the second largest variation, etc. The details of PCA will be introduced in later chapters. PCR then regresses the response on the resultant PCs. For this purpose, the appropriate number of principal components has to be determined. Inclusion of too many or few principal components in PCR leads to overfitting or underfitting problems and hence inaccurate prediction and complicated interpretation. Therefore, it is crucial to choose the optimal number of principal components.

Many methods have been introduced to choose the principal components. One of the methods is how much variation of the predictors we want to account for. For example, if we want to explain 90 percent of the variance, and the first four principal components satisfy the requirement, we will choose the first four principal components. In general, the last few principal components are discarded because of the least information contained. Another common way is we regress on an increasing sequence of principal components (e.g., the first principal component, the first two principal components, the first three principal components,..., all PC's) and then use model selection criteria to choose the optimal model. The commonly used criteria are the Akaike information criterion (AIC; Akaike[2]), the Bayesian information

criterion (BIC; Schwarz[35]) and the generalized cross validation (GCV; Wahba[38]). Some other methods include parallel analysis to determine the significant principal components (Franklin, et al., 1995[7]). Martinez, Liang, Zhou and Carroll (2010)[24] applied the model averaging using a Bayesian formulation to determine the number of principal components. In addition, Mansfield,Webster and Gunst[22]) proposed selecting original predictors along with the PCs and provided analytic forms of the procedure.

After choosing the number of principal components, the dependent variable is then regressed on the chosen principal component by ordinary regression methods. Since all the PC's are orthogonal, the regression coefficient for each PC are the same as what one would obtain with a simple linear regression on the PC. If all the principal components are used, the final PCR fitting result should be the same to the one obtained from OLS. For the convenience of prediction, it is often advisable to rewrite PCR model on the scale of the original variables.

1.3 Our Proposal

The traditional method for selecting PCs is essentially a best subset selection approach, where all possible increasing or decreasing sequences of PCs are tried out and compared. This discrete selection process is time-consuming especially for the high dimensional big data. This motivates us to consider a more efficient way of selecting the number of principal components in PCR. Our main idea is to obtain the best number of principal components by optimizing an approximated model selection criterion beforehand and then we regress the dependent variable on the chosen principal components. For this reason, we call the proposed method as the pre-tuned principal component regression (PTPCR). In PTPCR, we replace the indicator function with a smooth sigmoid function to quantify the complexity of a PCR model, then plug the principal component estimates into the approximated selection crite-

rion. This results in a smooth optimizing problem. Here, we use a logistic function with two parameters. One shape (or scale) parameter controls the sharpness of the approximation and the other one is threshold parameter corresponding to a cutoff point. In order to mimic a hard threshold effect, we fix the shape parameter at a relatively large value. In fact, the method shows considerable stability with respect to the shape parameters over a range of large values. By doing so, the optimization problem stays one-dimensional and can be solved efficiently. Once the cutoff value is estimated, the best number of principal components can be obtained naturally. Compared to the traditional way of regressing on every decreasing or increasing subsets of principal components, the computational time to arrive at the optimal model is greatly reduced by using PTPCR.

Furthermore, we also propose leaving the shape parameter in the sigmoid surrogated function free for estimation, leading to optimization with two decision variables. This results in a weighted PCR model, where all PC's are included but differently weighted. The weights are optimized in the sense of minimum AIC or GCV, albeit approximated. The new model would help improve the prediction accuracy. As another alternative, the sigmoid surrogated function can be applied to regression coefficients themselves, instead of the eigenvalues of the variance-covariance or correlation matrix. In addition, the proposed model enjoys other great flexibility. It can be extended to the generalized linear models(GLM). The method also allows natural incorporation of the kernel trick.

1.4 Outline of the Thesis

The remaining parts of the thesis are structured as follows. Chapter 2 contains a literature review on PCR, including its computation (e.g. singular value decomposition), traditional methods for determining the optimal number of PCs, its connection to other regression methods, and its applications. In Chapter 3, our proposed method,

pre-tuned principal component regression, is introduced in greater detail. We also discuss its variants. Chapter 4 includes the numerical results based on simulation studies and real data examples. Finally, a brief discussion concludes the thesis, where we summarize the advantages and disadvantages of the proposed PTPCR model and discuss future research avenues.

Chapter 2

Literature Review

In this chapter we provide a brief literature review on principal component regression (PCR), including its history, applications and the computational method. Then the common methods of choosing the number of principal components in PCR will be covered. At last, the connections between PCR and other linear regression methods will be discussed. These techniques are closely related to our proposed method.

2.1 Principal Component Regression

Consider the usual regression setup where the data available consist of $\{(y_i, \mathbf{x}_i) : \ i = 1, \ldots, n\}$, where both the response variable y_i and predictors $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})^T \in \mathbb{R}^p$ are continuous. Some original predictors can be categorical, in which scenario dummy numerical variables are introduced to account for them.

Let us consider a linear regression model

$$\mathbf{y} = \beta_0 \mathbf{1} + \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{2.1}$$

where $\mathbf{y} = (y_i)$ is an $n \times 1$ vector of observed responses, β_0 is the intercept, $\mathbf{1}$ an $n \times 1$ vector of ones, $\mathbf{X}$ is an $n \times p$ matrix of independent variables with sample size n, $\boldsymbol{\beta}$ is a $p \times 1$ vector of slope parameters, and $\boldsymbol{\epsilon}$ is an $n \times 1$ vector of $N(0, \sigma^2)$ random error terms. Without loss of generality (WLOG), we assume that the data have been standardized so that $\beta_0 = 0$ throughout the thesis unless otherwise indicated. Thus, the linear regression model in (2.1) reduces to

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

To estimate $\boldsymbol{\beta}$, the most commonly used method is ordinary least square (OLS) which minimizes the sum of squared errors between the observed and fitted response values. The least squares estimator $\hat{\boldsymbol{\beta}}$ is given by

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{2.2}$$

if $\mathbf{X}$ is of full column rank. The variance-covariance matrix of $\hat{\boldsymbol{\beta}}$ is

$$\text{var}(\hat{\boldsymbol{\beta}}) = \text{var}((\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}) = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}. \tag{2.3}$$

The resultant fitted value is

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \mathbf{H}\mathbf{y} \tag{2.4}$$

where $\mathbf{H}$=$\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ is called the hat matrix. One natural measure of the model fit is the minimized LS criterion, also referred to as the residual sum of squares (RSS), given by

$$\text{RSS} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \| \mathbf{y} - \hat{\mathbf{y}} \|^2 . \tag{2.5}$$

According to Equation 2.2, we know $\boldsymbol{\beta}$ is dependent on the existence of the inverse of the gram matrix $\mathbf{X}^T\mathbf{X}$. If $\mathbf{X}^T\mathbf{X}$ is singular, the inverse of $\mathbf{X}^T\mathbf{X}$ does not exist, then $\boldsymbol{\beta}$ can not be uniquely determined. One of the reasons that make the $\mathbf{X}^T\mathbf{X}$ singular is the columns of $\mathbf{X}$ are perfectly correlated. Sometimes, even the columns of $\mathbf{X}$ is highly, yet not perfectly, correlated and a unique $\boldsymbol{\beta}$ exists, the variance of $\boldsymbol{\beta}$ is tend to inflate because some diagonal entries of $(\mathbf{X}^T\mathbf{X})^{-1}$ are going to be large, rendering the OLS estimates unstable.

Because of the flaws of OLS, Massy (1965) [25] proposed the principal component regression (PCR), where the regression analysis technically speaking is based on principal component analysis (PCA; Pearson[30]). Principal component analysis is widely used in many application fields because it is simple and helps us to extract important information from the lousy and complicated data sets. PCA is highly useful in dimension reduction, in clustering, in eliminating collinearity and

so on. The main idea in principal component analysis is to convert a set of correlated variables into a set of uncorrelated variables through a sequence of orthogonal transformations. Each variable in the transformed data set is a principal component (PC). The principal components are put into a decreasing order by the explained variation and these PCs are orthogonal and hence uncorrelated to each other. Thus, OLS can be applied to the de-correlated PCs, yielding the principal component regression. PCR is closely related to ordinary least estimation. Their relationship can be easily revealed through decomposition of the design matrix $\mathbf{X}$ or the gram matrix $\mathbf{X}^T\mathbf{X}$, i.e., singular value decomposition (SVD) of the design matrix $\mathbf{X}$ or eigenvalue decomposition of the gram matrix $\mathbf{X}^T\mathbf{X}$.

2.2 Singular Value Decomposition

Among various others, singular value decomposition (SVD) is one of the most popular matrix decomposition methods. SVD factorizes a matrix into three matrices. It is useful in many applications because it is a very powerful technique to explore the geometric structure of the data matrix. Another perspective is that SVD makes it possible for us to use a lower rank matrix to approximate a high rank matrix. SVD is often a processor for other methods and procedure.

SVD of $\mathbf{X}$ is closely related to the principal component analysis. Applying SVD, we rewrite $\mathbf{X}$ as follows,

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^T \tag{2.6}$$

where

- $\mathbf{U}$ is a $n \times p$ orthogonal matrix with p orthonormal column vectors $\mathbf{u}_j$'s, which is also called left singular matrix;
- $\mathbf{D}$ is a $p \times p$ diagonal matrix with diagonal elements $d_1 \geq d_2 \geq \cdots \geq d_p \geq 0$; and

- $\mathbf{V}$ is a $p \times p$ matrix with orthonormal column vectors $\mathbf{v}_j$'s, which is also called right singular matrix.

SVD of $\mathbf{X}$ leads directly to the spectral decomposition of gram matrix $\mathbf{X^T X}$, since

$$\mathbf{X^T X} = \mathbf{V D U}^T \mathbf{U D V}^T = \mathbf{V D}^2 \mathbf{V^T}.$$

Thus the spectral decomposition of $\mathbf{X^T X}$ is given by $\mathbf{X^T X} = \mathbf{V \Lambda V^T}$, where $\mathbf{\Lambda}$ is the $p \times p$ diagonal matrix with diagonal element $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$ satisfying $d_j = \sqrt{\lambda_j}$ for $j = 1, \ldots, p$.

In the above rewriting, it is important to note that both $\mathbf{U}$ and $\mathbf{V}$ have orthogonal columns, but different dimensions. As a result, we have

$$\mathbf{V^T V} = \mathbf{V V^T} = \mathbf{I}_p$$

$$\mathbf{U^T U} = \mathbf{I}_p \tag{2.7}$$

where $\mathbf{I}_p$ is the $p \times p$ identity matrix with diagonal elements 1 and off-diagonal elements 0. It is of note that $\mathbf{U U^T} \neq \mathbf{I}_n$.

Multiplying $\mathbf{V}$ on both sides of the singular value decomposition equation yields that

$$\mathbf{XV} = \mathbf{UDV}^T\mathbf{V} \implies \mathbf{XV} = \mathbf{UD} \tag{2.8}$$

Partitioning the matrix $\mathbf{U}$ into column vectors, we have $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_p]$, then $\mathbf{UD} = [d_1\mathbf{u}_1, d_2\mathbf{u}_2, \cdots, d_p\mathbf{u}_p]$, where $d_j\mathbf{u}_j$ is the j-th principal component for $j = 1, 2, \cdots, p$.

In the current practice of PCR, the response is always regressed on the first k leading principal components for some $1 \leq k \leq p$. For example, we regress the response $\mathbf{y}$ on the first principal component $d_1\mathbf{u}_1$ when $k = 1$; we regress the response $\mathbf{y}$ on the first two principal components $[d_1\mathbf{u}_1, d_2\mathbf{u}_2]$ when $k = 2$, etc. In its general form, $\mathbf{U}_k\mathbf{D}_k$ becomes the new design matrix, where $\mathbf{U}_k = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_k]$ is formed

by including the first k columns of $\mathbf{U}$ only and $\mathbf{D}_k \in \mathbb{R}^{k\times k}$ is the k-th order leading principal submatrix of $\mathbf{D}$. If we apply OLS to these principal components, the principal component regression coefficient estimates is given by

$$\hat{\boldsymbol{\beta}}_k = (\mathbf{D}_k^T\mathbf{U}_k^T\mathbf{U}_k\mathbf{D}_k)^{-1}\mathbf{D}_k\mathbf{U}_k^T\mathbf{y} = \mathbf{D}_k^{-1}\mathbf{U}_k^T\mathbf{y}. \tag{2.9}$$

The predicted vector from PCR is

$$\hat{\mathbf{y}}_k = \mathbf{U}_k\mathbf{D}_k\hat{\boldsymbol{\beta}}_k = \mathbf{U}_k\mathbf{D}_k\mathbf{D}_k^{-1}\mathbf{U}_k^T\mathbf{y} = \mathbf{U}_k\mathbf{U}_k^T\mathbf{y} = \sum_{j=1}^{k}(\mathbf{u}_j^T\mathbf{y})\mathbf{u}_j. \tag{2.10}$$

When $k = p$, it amounts to

$$\hat{\mathbf{y}}_p = \sum_{j=1}^{p}(\mathbf{u}_j^T\mathbf{y})\mathbf{u}_j. \tag{2.11}$$

If we apply singular value decomposition to the ordinary least square regression, we have the regression coefficient estimate and predicted value are

$$\hat{\boldsymbol{\beta}} = \mathbf{V}\mathbf{D}^{-1}\mathbf{U}^T\mathbf{y} \tag{2.12}$$

and hence

$$\hat{\mathbf{y}} = \mathbf{U}\mathbf{U}^T\mathbf{y} = \sum_{j=1}^{p}(\mathbf{u}_j^T\mathbf{y})\mathbf{u}_j. \tag{2.13}$$

From Equation (2.11) and Equation (2.13), it is clear that when all p principal components are used in PCR, it gives the same predicted vector as that obtained from OLS. In other words, if we choose all the principal components to do the regression in PCR, it becomes equivalent to OLS.

In the PCR analysis, we will fit p models as the number k of principal components included in PCR increases from 1 to p. Then the question is how to choose the best model among these p models. There are many criteria to compare models, including AIC, BIC, and GCV.

2.3 Selection Methods

Choosing the best PCR model amounts to selection of the optimal number k^*. To this end, a selection criterion has to be used. Among various others, Akaike information criterion (AIC), Bayesian information criterion (BIC), and generalized cross validation (GCV) are the most commonly used ones.

Akaike information criterion (AIC) was proposed by Akaike in 1973. It is the first model selection criterion. AIC measures the information lost when a given model is used to fit the data. When each model assumes IID normal error terms, the AIC is defined as

$$\mathrm{AIC}_k = n \ln(\mathrm{RSS}_k) \; + \; 2\,k \tag{2.14}$$

up to some irrelevant constant, where AIC_k is the AIC value for the model with k principal components, RSS_k is the residual sum of squares from fitting the kth model. Among all the candidates, the best model is identified with the minimum AIC_k value.

Bayesian information criterion (or Schwarz criterion) was put forward by Schwarz in 1978. It is partly based on the likelihood function as AIC. Under the assumption of IID normal random errors, the BIC is defined as

$$\mathrm{BIC}_k = n \ln(\mathrm{RSS}_k) \; + \; \ln(n)\,k \tag{2.15}$$

up to some constant, where BIC_k is the BIC value for the model with k principal components. The model with a smaller BIC value is preferred.

BIC is closely related to AIC. They both add a penalty term (the number of parameters) to the goodness-of-fit measure of the model because the likelihood tends to increase with more parameters added. From equations (2.14) and (2.15), both AIC and BIC increase as the number of parameters k increase. Thus, a lower AIC or BIC value implies fewer parameters that correspond to a more parsimonious model. Neither AIC nor BIC is a performance measure of a single model; they are only

suitable to compare models. Among a number of competitive models, they help find those that perform relatively better.

Despite the similarity with AIC, BIC is motivated by a different idea. BIC takes a Bayesian approach in model selection. It puts a heavier penalty on the model complexity and hence gives preference to simpler models (Hastie, Tibshirani and Friedman[9]). As a model selection criterion, BIC is asymptotically consistent and it selects the true model given a family of models with probability tending to 1 as $n \to \infty$. According to Shibata, AIC is asymptotically efficient(Shibat[36]). In addition, Yang pointed out that no criterion can have asymptotic efficiency and consistency simultaneously (Yang[41]). For the purpose of model selection, there is no clear rule to make a choice between them.

Besides, cross-validation is the most popular and simplest method for estimating prediction error (Hastie, Tibshirani and Friedman[9]) and generalized cross-validation (GCV) is the most frequently used criterion in selecting models. GCV, proposed by Wahba (1977), is given

$$\mathrm{GCV}_k = \frac{1}{n}\sum_{j=1}^{n}\left[\frac{y_j - \hat{y}_{(k)j}}{1 - \mathrm{EDF}/n}\right]^2 \tag{2.16}$$

where GCV_k is the GCV value for the model with k principal components; EDF is the number of effective degrees of freedom, defined as the trace of the hat matrix; and $\hat{y}_{(k)j}$ is the predicted value from kth model. The smallest GCV value, same as AIC and BIC, indicates the best model. The GCV has the functional relationship with AIC. It is easy to show that

$$\mathrm{GCV} \propto e^{\frac{\mathrm{AIC}}{n}} \tag{2.17}$$

up to some constant. According to Hastie, Tibshirani and Friedman[9], GCV has the computational advantage.

Mallows's C_p (Mallows[19]) is another choice and it is defined as

$$\mathrm{C}_k = \frac{\mathrm{RSS}_k}{S^2} - n + 2k \tag{2.18}$$

where S^2 is the residual mean square when regressing on all the p predictors. A small value of Mallow's C_p indicates the model is precise. Akaike also pointed that, in linear models, AIC is equivalent to Mallow's C_p (Akaike[2]).

PRESS criterion (Allen[3]) emulates cross-validation when an independent test data set is not available. PRESS criterion is defined as

$$\text{PRESS}_k = \sum_{i=1}^{n} (y_i - \hat{y}_{k_{(i,-i)}})^2 \tag{2.19}$$

where $\hat{y}_{k_{(i,-i)}}$ denotes the predicted value for the ith observation from the k-th fitted model by excluding ith observation. The smallest PRESS value indicates the best model.

2.4 Connection between PCR and Other Linear Regression Methods

PCR is closely related to other linear regression models such as ridge regression (RR), partial least square (PLS) regression. They share similar characteristics in some ways. In ridge regression, the regression coefficient estimate $\hat{\boldsymbol{\beta}}_{RR}$ is obtained by solving

$$\hat{\boldsymbol{\beta}}_{RR} = \arg\min_{\beta} \left[(\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta) + \lambda \boldsymbol{\beta}^T \boldsymbol{\beta} \right] \tag{2.20}$$

and hence given by

$$\hat{\boldsymbol{\beta}}_{RR} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}, \tag{2.21}$$

where $\lambda > 0$ is a tuning parameter. The corresponding predicted vector $\hat{\mathbf{y}}_{RR}$ is

$$\hat{\mathbf{y}}_{RR} = \mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} \tag{2.22}$$

With SVD of $\mathbf{X}$, $\hat{\mathbf{y}}_{RR}$ can be rewritten as

$$\hat{\mathbf{y}}_{RR} = \mathbf{UD}(\mathbf{D}^2 + \lambda\mathbf{I})^{-1}\mathbf{DU}^T\mathbf{y} = \sum_{j=1}^{p} \frac{d_j^2}{d_j^2 + \lambda} \mathbf{u_j}\mathbf{u_j^T}\mathbf{y}. \tag{2.23}$$

In summary, the predicted values from RR, PCR, and OLS can be put in an unified form as follow:

$$\hat{\mathbf{y}} = \sum_{j=1}^{p} \omega_j \mathbf{u_j} \mathbf{u_j^T} \mathbf{y} \tag{2.24}$$

with varying weights ω_j given by

- $\omega_j = 1$ for $j = 1 \cdots p$ in OLS,
- $\omega_j = \frac{d_j^2}{d_j^2 + \lambda}$ for $j = 1 \cdots p$ in RR
- $\omega_j = 1$ for $j = 1 \cdots k$ and $\omega_j = 0$ for $j = k+1 \cdots p$ in PCR.

Therefore, PCR, RR and OLS are closely related to each other. They share the same formula yet with different weights. Figure 2.1 illustrates the relationship among them. RR can be treated as a shrunk version of OLS. The regression coefficients estimate of OLS is shrunk by the rate $\frac{d_j^2}{d_j^2+\lambda}$. PCR can be treated as a truncated version of OLS. The first k large eigenvalue components are kept and the last $p-k$ small eigenvalue components are discarded. Also, the RR can be treated as a smooth version of PCR because it weights the principal components by a factor instead of keeping the first k principal components and discarding the rest (Bair, Hastie, Paul and Tibshirani[5]).

Stone[37] proposed the continuum regression framework by considering a particular objective criterion in a general sequential procedure to construct regressors in least squares regression. By adjusting a continuum parameter, it generates a continuous spectrum of models where OLS and PCR stand at the two ends and the PLS lies in-between. The framework we consider in (2.24) is even more general than continuum regression since the principal components could be replaced with any orthogonal components and different models can be obtained by applying different weight functions.

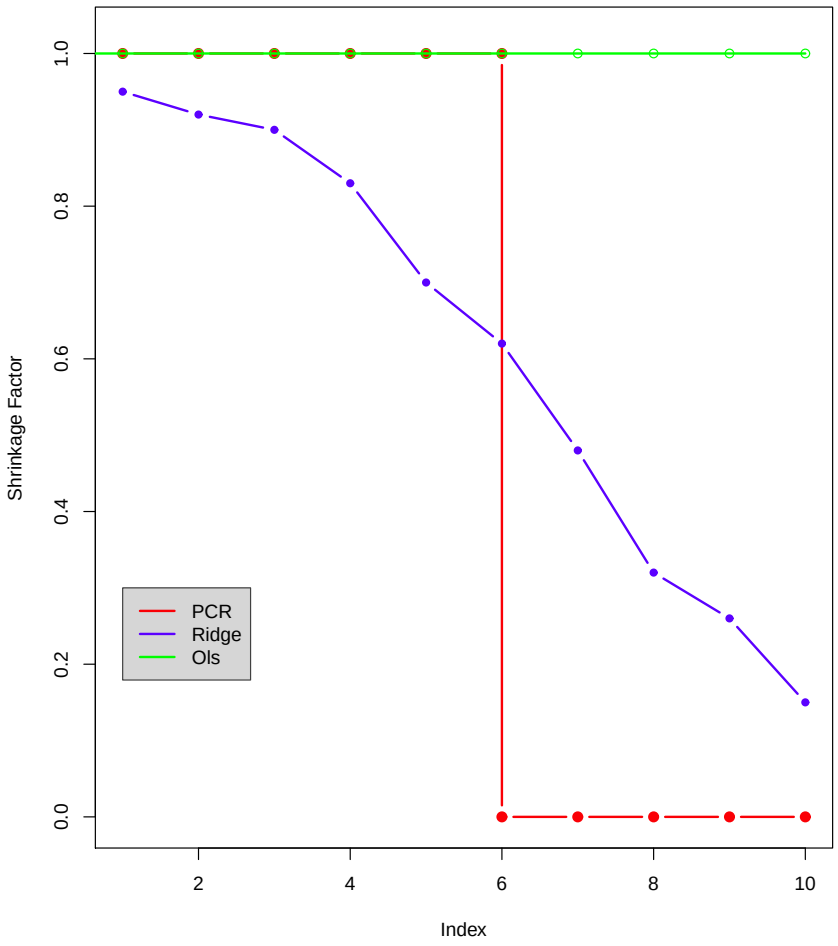


Figure 2.1: Plot of RR, PCR, OLS regression coefficient shrinkage. RR shrinks the regression coefficient of OLS by the factors $\frac{d_j^2}{d_j^2+\lambda}$ while PCR truncates them.

Chapter 3

Pre-Tuned Principal Component Regression

Nowadays, the big data problem is a hotly discussed issue among scientists from different areas. In statistics, statisticians are seeking better methods to model big data. Big data are often characterized with four V's: velocity, volume, variety, and veracity. The multicollinearity problem is often seen in high dimensional data. PCR is an excellent alternative method to OLS in linear regression because it overcomes the multicollinearity problem and improves the prediction accuracy. The main issue associated with PCR is to identify the best subset of principal components. However, with the ordinary PCR, it is very time consuming because we have to fit p models. This motivates us to develop a computationally efficient algorithm for conducting PCR that is applicable even in high-dimensional scenario. One prominent feature with these principal components is that they are naturally ranked in some way, compared to a set of arbitrary predictors. The ranking could be based on the eigenvalues, which correspond to their explained percentage of variations in observed predictors, or the regression coefficients, which correspond to their associations with the response. This interesting feature suggests a monotone weighting function for PCR.

This chapter is organized in the following manner. In Section 3.1, we will present our proposed method, pre-tuned principal component regression (PTPCR), in detail. The proposed method mimics PCR in many ways. Then we develop several variants of PRPCR along the similar lines in Section 3.2. Section 3.3 concludes the chapter

by presenting all those variants in a unified form, together with OLS, PCR, and ridge regression.

3.1 PTPCR

In the common practice of PCR, we obtain the PCR coefficient estimator $\hat{\boldsymbol{\beta}}_k = \mathbf{D}_k^{-1}\mathbf{U}_k^T\mathbf{y}$ for the original predictors $\mathbf{X}$ and hence the predicted vector $\hat{\mathbf{y}}_k = \mathbf{X}\hat{\boldsymbol{\beta}}_k = \mathbf{U}_k\mathbf{U}_k^T\mathbf{y}$ for each model with $k = 1 \cdots p$ in the way we discussed in the previous chapter. In chapter 2, we have rewritten the PCR predicted vector in a weighted form

$$\hat{\mathbf{y}} = \sum_{j=1}^{p} \omega_j \mathbf{u_j}\mathbf{u_j^T}\mathbf{y} = \sum_{j=1}^{p} \omega_j (\mathbf{u_j^T}\mathbf{y})\mathbf{u_j} \tag{3.1}$$

with $\omega_j = 1$ for $j = 1 \cdots k$ and $\omega_j = 0$ for $j = k+1 \cdots p$, if the first k principal components are selected. If we denote the regression coefficients for the principal components as $\alpha_j = \mathbf{u_j^T}\mathbf{y}$, then the predicted vector can be written as

$$\hat{\mathbf{y}} = \sum_{j=1}^{p} \delta(\lambda_j; c)\alpha_j \mathbf{u_j} \tag{3.2}$$

where the weights are supplied via the indicator function $\delta(\lambda_j; c) = I\{\lambda_j \geq c\} = I\{d_j \geq \sqrt{c}\}$ and $c = \lambda_k = d_k^2$ is the cutoff point or cutpoint in short. Then we compute the model selection criterion for each model and choose the optimal model. In other words, the number of principal components k plays the role of the tuning parameter in PCR.

If, for example, prediction accuracy is of central concern, GCV, given by,

$$\mathrm{GCV}_k = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{y_i - \hat{y}_i}{1 - \mathrm{EDF}/n}\right]^2 = \frac{1}{n}\frac{\mathrm{RSS}_k}{(1 - \mathrm{EDF/n})^2} = \frac{n}{(n-k)^2}\mathrm{RSS}_k \tag{3.3}$$

is often used, where $\mathrm{RSS}_k = \|\mathbf{y} - \hat{\mathbf{y}}_k\|^2$ and EDF is trace of the hat matrix $\mathbf{H}_{PCR} = \mathbf{U}_k\mathbf{U}_k^T$

$$\mathrm{EDF} = \mathrm{tr}(\mathbf{U}_k\mathbf{U}_k^T) = \mathrm{tr}(\mathbf{U}_k^T\mathbf{U}_k) = \mathrm{tr}(\mathbf{I}_k) = k = \sum_{j=1}^{p}\delta(\lambda_j; c) \tag{3.4}$$

If we donate the optimal number of principal components k^*, then

$$k^* = \arg\min_k GCV_k \tag{3.5}$$

k^* corresponding to the smallest GCV value.
This is the conventional procedure for fitting PCR where p models are fit in order to determine the optimal PCR model. Clearly this is computationally inefficient especially when p is large.

To improve the computational efficiency, our first proposal of PTPCR is to avoid the selection of the tuning parameter. The main idea is to plug an approximated predicted vector to the model selection criteria directly and minimize the criteria function with respect to a threshold parameter that corresponds to the cutoff value. Then we select the principal components that are associated with the eigenvalues larger than the estimated threshold.

To start, we first replace the indicator function $\delta(x; c)$ in (3.2) with a smooth sigmoid surrogate function $s(x; c)$. It is nature to use the expit or logistic function while many other choices are available. The logistic function is defined as

$$s(x; c) = s(x; a, c) = \text{expit}\{a(x - c)\} = \frac{1 + \tanh(a(x - c)/2)}{2}, \tag{3.6}$$

where $\tanh(\cdot)$ is the hyperbolic tangent function and the shape parameter a controls the sharpness of the approximation. Figure 3.1 plots $\text{expit}\{a(x - c)\}$ as a function with $c = 0$ for different choices of $a = 1, \cdots, 200$; also plotted in the blue line is the indicator function $I(x \geq 0)$. The plot shows that the expit function is getting closer to the indicator function as the shape parameter a increases. Thus, a relatively large a is needed in order to have a good approximation. In addition, it is nature to have $a = a_n$ such that $\lim_{n\to\infty} a_n = \infty$ and thus $\lim_{n\to\infty} s(x; c, a_n) = \delta(x; c)$. It turns out that the performance of our proposed method is rather stable with different choices of a, and this will be explored in the next chapter.

Replacing the indicator function $\delta(\lambda_j; c)$ with logistic function $s(\lambda_j; c)$, we will have the approximated predicted vector and effective degrees of freedom (EDF).

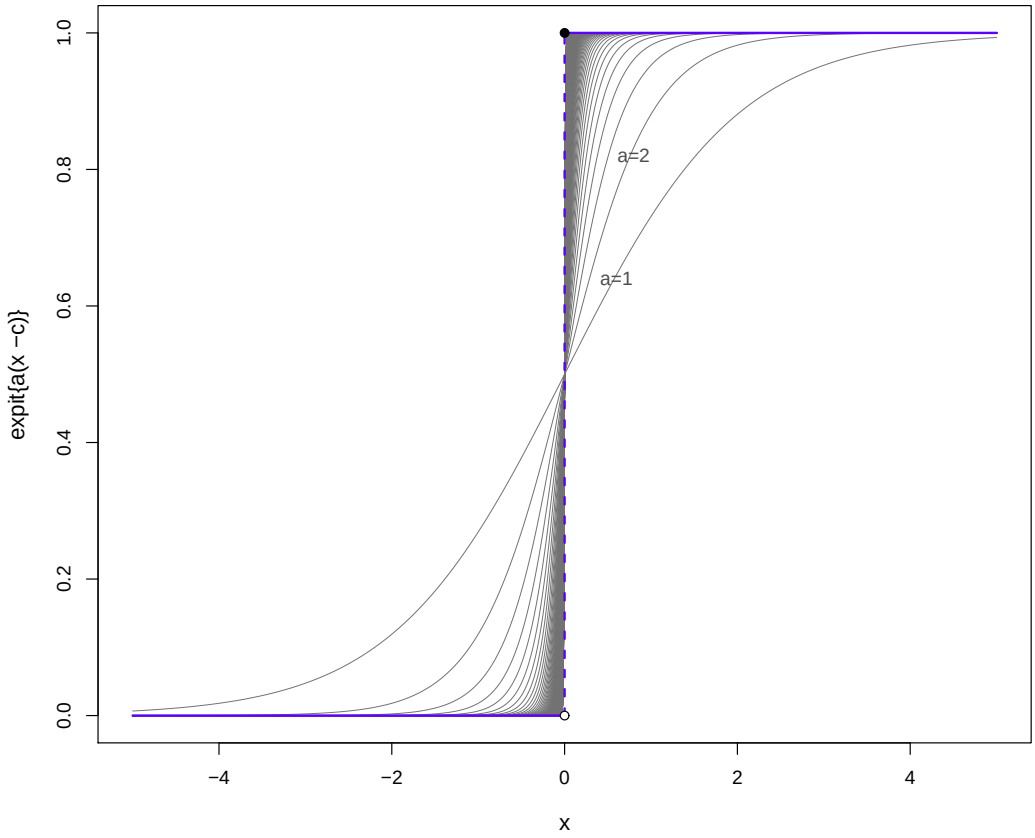


Figure 3.1: Plot of the Expit Function expit$\{a(x-c)\}$ with $c = 0$ and $a = 1, \cdots, 200$

The predicted vector $\hat{\mathbf{y}}$ in (3.2) is approximated by

$$\tilde{\mathbf{y}} = \sum_{j=1}^{p} s(\lambda_j; c)\alpha_j \mathbf{u_j} \tag{3.7}$$

and the EDF k in equation 3.4 is approximated by

$$\tilde{k} = \sum_{j=1}^{p} s(\lambda_j; c) \tag{3.8}$$

Bringing $\tilde{\mathbf{y}}$ and $\tilde{k}$ into GCV_k yields the approximated GCV_k given as follows

$$\mathrm{GCV}_c = \frac{n}{(n-\tilde{k})^2}\|\mathbf{y} - \tilde{\mathbf{y}}\|^2 \tag{3.9}$$

For the implementation convenience, we rewrite the GCV_c in matrix notation. First, define $\mathbf{\Delta} = \text{diag}\{\delta(\lambda_j; c)\}$ and $\mathbf{S} = \text{diag}\{s(\lambda_j; c)\}$. It should be noticed that $\mathbf{\Delta}^2 = \mathbf{\Delta}$ since $\mathbf{\Delta}$ is a diagonal matrix with diagonal elements either 1 or 0. The predicted vector is $\hat{\mathbf{y}} = \mathbf{U\Delta U}^T\mathbf{y}$. If we replace $\mathbf{\Delta}$ with $\mathbf{S}$, then the predicted vector becomes $\tilde{\mathbf{y}} = \mathbf{USU}^T\mathbf{y}$ and hence $\text{RSS}_k = \|\mathbf{y} - \tilde{\mathbf{y}}\|^2 = \mathbf{y}^T(\mathbf{I} - \mathbf{USU}^T)\mathbf{y}$ and $\text{EDF} = \text{tr}(\mathbf{USU^T}) = \text{tr}(\mathbf{S})$. Putting together, the approximated GCV is

$$\text{GCV}_\lambda(c) = \frac{n\mathbf{y}^T(\mathbf{I} - \mathbf{USU}^T)\mathbf{y}}{(n - \text{tr}(\mathbf{S}))^2}, \tag{3.10}$$

where c enters GCV through the matrix $\mathbf{S}$.

The approximated GCV in (3.10) can be treated as an objective function of c. The best cutoff point c^* is the one that minimizes the approximated GCV, i.e.,

$$c^* = \text{argmin}_c \text{GCV}_\lambda(c). \tag{3.11}$$

Solving (3.11) is a one-dimensional smooth optimization problem. We use function `optimize()` in R software to get the c^*. Once $c^\star$ is identified, we include only those principal components that correspond to eigenvalues greater than $c^\star$. In other words, the best number k of principal components is given by

$$k = \sum_{j=1}^{p} \delta(\lambda_j; c^*) \tag{3.12}$$

With the optimal number k of PCs, the final PCR model is found by regressing the response on the first k PCs.

3.2 PTPCR Variants

3.2.1 PTPCR Variant 1

In PTPCR, we assume that the shape parameter a is a fixed large number in order to ensure that the logistic function facilitates a good approximation to the indicator

function. On the other hand, it is nature to leaving a as a free parameter. This would lead to a different model that is highly competitive to PCR. We call this model pre-tuned principal component regression variant 1(PTPCR-V1).

To estimate a, rewrite the logistic function $s(\lambda_j; c)$ as $s(\lambda_j; a, c)$. With the same steps as in PTPCR, we can obtain the same form for the approximated GCV as in (3.10). The only difference lies in the diagonal matrix $\mathbf{S}$. In PTPCR, $\mathbf{S}$ involves only c while two parameters $\{a, c\}$ enter GCV through $\mathbf{S}$ in PTPCR-V1. The approximated GCV in (3.10) can be rewritten into

$$\mathrm{GCV}_\lambda(c, a) = \frac{n\mathbf{y}^T(\mathbf{I} - \mathbf{U}\mathbf{S}\mathbf{U}^T)\mathbf{y}}{(n - \mathrm{tr}(\mathbf{S}))^2}. \tag{3.13}$$

Namely, GCV is now an objective function for both a and c. The parameters a and c can be estimated by minimizing (3.13), i.e.,

$$\{c^*, a^*\} = \mathrm{argmin}_{c,a}\mathrm{GCV}_\lambda(c, a). \tag{3.14}$$

This boils down to a two-dimensional smooth optimization problem. It is worth noting that we no longer truncate principal components in PTPCR-V1. Instead, all of them will be included in the model, however, their individual contribution to the model will be regulated via the weights given by $s(\lambda_j, \hat{a}, \hat{c})$.

The relationship among PCR, PTPCR and PTPCR-V1 is illustrated in the figure 3.2. The blue line corresponds to the logistic function $s(\lambda_j; c)$ with a large fixed $a = 80$ in PTPCR, which hence approximates the indicator function marked in red as used in PCR. The green line shows a logistic function $s(\lambda_j; c, a)$ with both a and c being free, the scenario in PTPCR-V1. Clearly, PTPCR-V1 enjoys more flexibility in fitting PCR models and could lead to improved prediction accuracy. One drawback, however, is that the final model is no longer as interpretable as PCR or PTPCR where only the first k PCs are included. Nevertheless, interpretability is not an important issue with PCR models in general.

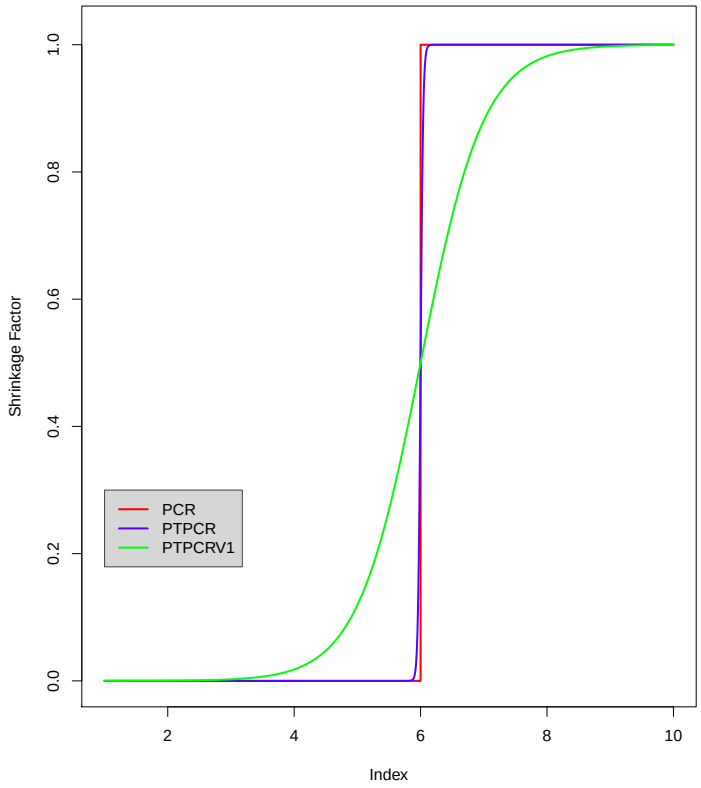


Figure 3.2: Comparsion between PCR, PTPCR and PTPCR-V1

3.2.2 PTPCR Variant 2

In PTPCR, we use the logistic function $s(\lambda_j; c)$ to replace the indicator function in PCR, then change the problem into a one-dimensional smooth optimization problem. The basic idea involved here is selecting the PCs by the ordered eigenvalues of the sample variance and co-variance matrix.

Using the top principal components for regression is justified in some way by several authors. With extensive simulation studies, Li (2007) [17] affirmed the conjecture that the first few principal components have the stronger correlation to the

response than the low rank principal components if one arbitrarily selects a covariance matrix for the predictors matrix and coefficients for the regression. Artemiou and Li (2009) [4] presented a probabilistic explanation that the response is more likely to have a higher correlation with the leading principal components. Namely, $P(\rho_j^2 \geq \rho_{j'}^2) > \frac{1}{2}$ whenever $j < j'$ where ρ_j denotes the correlation between the response and the j-th principal component. Ni (2011) [29] reinforced the results of Artemiou and Li with some new perspective of this idea.

However, Jollife (1982) [14] used several data examples to illustrate that a principal component corresponding to a smaller eigenvalue could have a higher correlation with the response; Hadi and Ling (1998) [8] presented the similar results with an example in which only the principal component corresponding to the smallest eigenvalue is correlated to the response. In addition, Hwang and Nettleton (2003)[12] pointed out that discarding the principal components with small eigenvalues but closely related to the response $\mathbf{y}$ may induce large biases.

Motivated by these papers, it is nature to ask if we can choose the PCs by the magnitude of the coefficients $\alpha_j = \mathbf{u_j^T y}$. The idea is use $s(\alpha_j^2; a, c)$ to replace the indicator function $\delta(\alpha_j^2; a, c)$. Computing the predicted vector and the number of effective degrees of freedom in the same manner as in PTPCR-V1, we rewrite the approximated GCV in (3.10) as

$$\mathrm{GCV}_\alpha(a, c) = \frac{n\mathbf{y}^T(\mathbf{I} - \mathbf{USU}^T)\mathbf{y}}{(n - \mathrm{tr}(\mathbf{S}))^2} \tag{3.15}$$

The $\mathrm{GCV}_\alpha(a, c)$ is different from $\mathrm{GCV}_\lambda(a, c)$ in the diagonal matrix S.

Same as before, there are two options in computing the final prediction. One option is fix a at a large value and include only PCs with the coefficients greater than c^* in the model. In this approach, the logistic function is shard due to the large a value and hence c^* is used as a threshold for selection of principal components. We call this variant of PTPCR as principal component regression variant 2 (PTPCR-V2).

More specifically, the objective function (3.15) is reduced to

$$\mathrm{GCV}_\alpha(c) = \frac{n\mathbf{y}^T(\mathbf{I} - \mathbf{USU}^T)\mathbf{y}}{(n - \mathrm{tr}(\mathbf{S}))^2}. \tag{3.16}$$

Compared to equation (3.10) where the diagonal element of matrix $\mathbf{S}$ is $s(\lambda_j; c)$, the diagonal element of $\mathbf{S}$ in equation (3.16) now becomes $s(\alpha_j^2; c)$. The coefficient cutoff point c^* is given by

$$c^* = \mathrm{argmin}_c \mathrm{GCV}_\alpha(c). \tag{3.17}$$

The second option is to leave both a and c for estimation as is in (3.15). This may lead to a model with potentially better predictive ability.

In either scenario, the optimization problem involved is smooth, and either one-dimensional or two-dimensional. The result PTPCR-V2 models may outperform PCR since they take the association with the response into consideration in selecting or weighting the principal components.

With a fixed, solving the cutoff point in methods PTPCR and PTPCR-V2, as presented above, are one-dimensional optimization problems. We have used the function `optimize()` in R software to solve this optimization problem. This function implement the Brent's method [6], which is a hybrid method that searches over a given interval for the optimum value of the objective function. The lower bound and the upper bound for the search interval can be conveniently determined by the smallest and largest values of either λ_j or α_j^2. Brent's method combines golden section search and successive parabolic interpolation. Its convergence rate is guaranteed superlinear for solving the PTPCR problems. Golden section search (1953) [6] [15] is a useful technique to find the minimum or maximum of a function by recursively narrowing the range to an interval which includes the extreme value. The first evaluation is computed at the point $a_1 = a_0 + (1 - \phi) \cdot (b_0 - a_0)$ where $[a_0, b_0]$ is the interval in which the objective function is optimized and ϕ is the golden rate 0.618. The second evaluation is computed at $b_1 = a_0 + \phi \cdot (b_0 - a_0)$ and the local optimum is found in $[a_1, b_1]$. Successive parabolic interpolation is another technique to find

the minimum or maximum of a function by using parabolas to the function at three unique points. Combining these two methods yields advantages in both speed and stability for finding the solution.

Solving PTPCR-V1 is a two-dimensional optimization problem. The function `optim()` in R implements several algorithms for multivariate optimization. The default method is using Nelder and Mead (1965)[27] which only use function values and works well for non-differentiable objective functions. Other optional methods include BFGS (a quasi-Newton method), conjugate gradient, and simulated annealing.

3.3 Unified Form

Looking back to the four alternative methods to PCR, it is worth noting that they all involve the replacement of the indicator function with a logistic function and an optimizing problem connected to GCV. For better exposition, we can put them in the following unified form in terms of the predicted vector:

$$\hat{\mathbf{y}} = \sum_{j=1}^{p} \omega_j \alpha_j \mathbf{u}_j \tag{3.18}$$

with

$$\omega_j = \begin{cases} 1 & \text{OLS,} \\ \delta(\lambda_j; c) & \text{PCR} \\ \frac{d_j^2}{d_j^2+\lambda} & \text{RR} \\ \delta(\lambda_j; c) & \text{PTPCR} \\ s(\lambda_j; c, a) & \text{PTPCR} - \text{V1} \\ \delta(\alpha_j^2; c) & \text{PTPCR} - \text{V2} \\ s(\alpha_j^2; a, c) & \text{PTPCR} - \text{V3} \end{cases},$$

where constants a, c are the parameters to be determined. Equation (3.18) shows that all the seven models share the same formula for the fitted vector. In (3.18),

the summand involved in the fitted vector is the product of the weight function, the regression coefficient, and the **u** vector. The only difference among these seven models lies in the weight function. By rewriting all the seven models in a unified form, the relationship and connections among these models become manifest.

Chapter 4

Numerical Result and Discussion

In this chapter, we will present the numerical results from analyses of both simulated data and real data examples based on our proposed models. In simulation section, different scenarios were used to evaluate the performance of our methods and compare them to others such as original PCR and OLS. Subsequently, we tested our models by using two real data sets.

When comparing the models, three performance criteria were used: computation time, prediction accuracy, and selected number of principal components when appropriate. In our programming, we have used the function `system.time()` in R to obtain the CPU computing time. The prediction accuracy was found by calculating $\sum_{j=1}^{p}(y_j - \hat{y}_j)^2/n$. Both our simulating and real data examples are performed in software R. When evaluate the models, the one with less simulation time and higher prediction accuracy is preferred.

The remaining of the chapter is organized in the following manner. Section 4.1 presents the model settings for simulation studies and the results from the simulated experiments. Section 4.2 contains two real data examples.

4.1 Simulation Results

In this section, we outline the simulation setting. All the models involve one-dimensional continuous quantitative response Y and a number of continuous quantitative predictors X's. In order to evaluate the performance of different methods, several simulation settings are considered. Each simulation setting involves several

parameters for controlling different aspects.

4.1.1 Simulation Settings

In all simulation studies, the predictors $\mathbf{x} \in \mathbb{R}^p$ is generated from a multivariate normal distribution with mean $\boldsymbol{\mu}_x = \mathbf{0}$ and variance-covaraince matrix $\boldsymbol{\Sigma} = \left(\rho^{|j-j'|}\right)$. The parameter ρ is hence used to control the correlation among predictors. Let n denote the sample size and k is the true number of principal components used to be regressed. Let $\boldsymbol{\alpha}$ denote the regression coefficients when regressing Y on the PC's.

We considered two types of model configurations. In PTPCR and PTPCRV1, the response $\mathbf{y}$ is generated as the product of the first k column vectors of matrix $\mathbf{U}$, or $\mathbf{U}_k$ from SVD of X and a given set of values for $\boldsymbol{\alpha}$ plus the noise vector $\boldsymbol{\epsilon}$. In PTPCRV2, $\mathbf{y}$ is the product of the matrix $\mathbf{U}$ and $\boldsymbol{\alpha}$ plus the noise, where $\boldsymbol{\alpha}$ could have sparsity anywhere in its components. That is, $k = p$ in the latter case, however the response could be arbitrarily associated with any of the PC's in any order. The noise vector $\boldsymbol{\epsilon}$ here is simulated from a multivariate normal distribution with mean $\mathbf{0}$ and VCOV matrix $\sigma^2\,\mathbf{I}_n$, where the parameter σ^2 can be adjusted to control the signal strength.

In order to evaluate the performance of the proposed models, we consider seven different simulation settings that vary in sample size, dimension, correlation, signal strength, and coefficients. The detailed specifications are listed in table 4.1.

Table 4.1: The Simulation Settings

Settings	p	n	k	ρ	σ	α	a
Setting I	50	1000	6	0.5	0.05	$(k:1)$	1/200
Setting II	20	1000	$\mathbf{k}_\alpha$	0.5	0.05	$\boldsymbol{\alpha}_k$	1/200
Setting III	50/100/200	500/1000	6	0.5	0.05	$(k:1)$	5
Setting IV	100/300/500	1000	7	0.3/0.6/0.9	0.05	$(k:1)$	200
Setting V	100/300/500	1000	7	0.3/0.6/0.9	0.05	$\frac{(k:1)}{k}$	200
Setting VI	100/300/500	2000	7	0.3/0.6/0.9	0.05	$\frac{(k:1)}{k}$	200
Setting VII	300	1000	7	0.3/0.6/0.9	0.05/1	$\frac{(k:1)}{k}$	200

The $\mathbf{k}_\alpha$ in the table is a vector with elements 1, 5, 10, 15, 20 and the $\boldsymbol{\alpha}_k$ is the vector with the elements 2.9, 3.8, 5, 3.8, 2.9 corresponding to the index in $\mathbf{k}_\alpha$ and 0 in all the other indexes. The slash / in the table means 'either or'. For example, 0.05/1 means either 0.05 or 1.

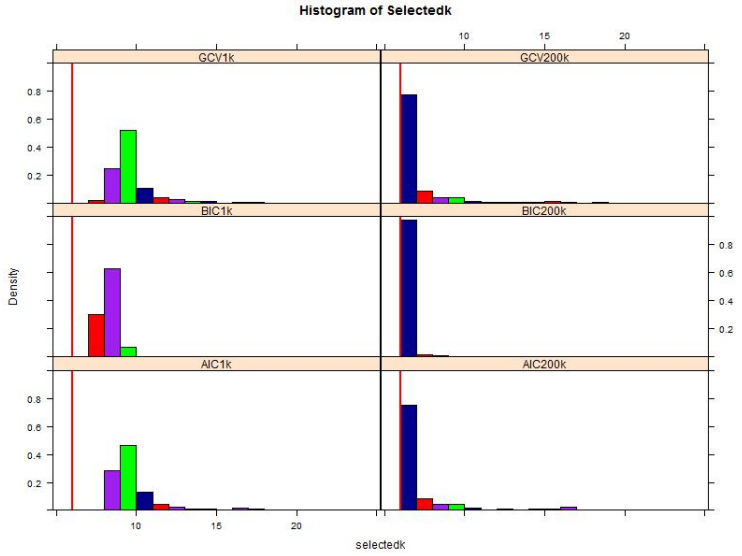
The histograms of the number of selected PC's from model PTPCR and PTPCR-V2 were drawn under the setting I and the setting II. The computing time was calculated under the setting setting III. The prediction error was computed under the setting IV, setting V, setting VI and setting VII. These settings represent the strong and weak signals.

4.1.2 Numerical Results

In this section, we will present the simulation results including the histograms of the number of selected PC's, the computing time and the prediction error. In addition, we will give the representative discussion after each result.

First, we provide the histograms of the number of selected PC's from model PTPCR by different criteria. The criteria are AIC, BIC and GCV. On setting I, the true number of PC's was set as 6 and the shape parameter was either 1 or 200. For each model, the count was given by 500 times run.

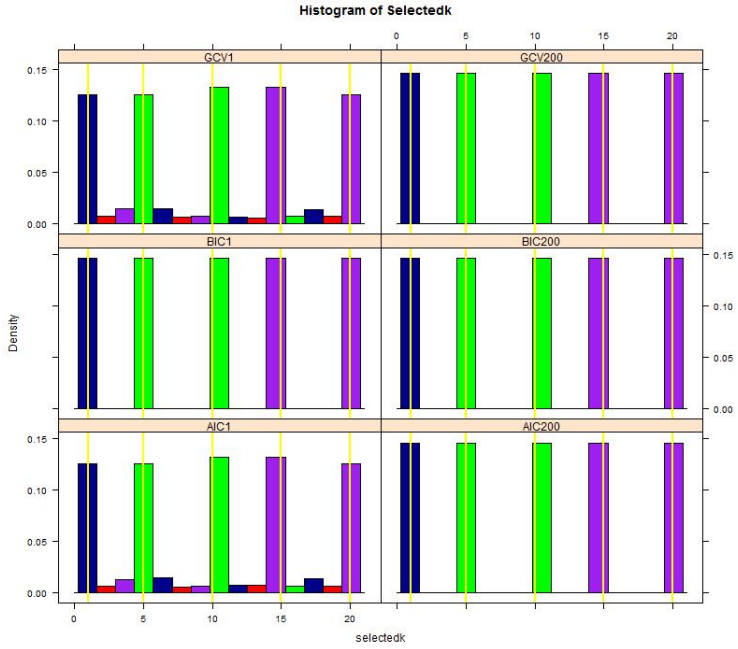
Figure 4.1: Histograms of the number of the selected PC's.



In figure 4.1, the left column is the results from $a = 1$ and the right column is the results from $a = 200$. The red vertical line in the figures is the true number of PC's and it was set as 6 in our simulation work. Comparing the bars in the right column with the ones in the left column, the right ones are more concentrated and closer to the red line while the left ones are more disperse. This means the model with a large constant a has a better performance than the one with a small a because the former one is more likely to choose the true number of PC's. In terms of the criteria, GCV and AIC have the similar performance because they have the semblable pattern in the figure. The BIC criterion has the best performance since the bars are gathering closest to the red line no matter what a value is.

Then, we show the histograms of the indexes corresponding to the selected PC's from model PTPCR-V2 by different criteria with shape parameter 1 and 200. According to setting II, the true k was set as 1, 5, 10, 15, 20 and α was set as a vector with 20 elements. The 1st, 5th, 10th, 15th and 20th element have the value 2.9, 3.8, 5, 3.8, 2.9 and all the others are 0. For each model, 500 times run were taken to get the count.

Figure 4.2: Histograms of the index corresponding to the selected PC's.



The model with $a = 1$ resulted in the histograms on the left side in figure 4.2 while $a = 200$ had the results on the right side. The yellow vertical lines in the figures are corresponding to the true index of the selected PC's. In term of the performance of different criteria, BIC is the best which is the same result as the previous figure. In addition, the model with $a = 200$ has the better result than the model with $a = 1$.

Next, we compare the simulation time between the original PCR and our proposed method PTPCR under the setting III. In setting III, the sample size was set as either 500 or 1000 with dimension 50, 100 and 200. Then, we recorded the computing time for these two models. Table 4.2 and 4.3 shows the results.

Table 4.2: The computing time for m simulation run when $n = 500$

m = 10	p=50	$p = 100$	$p = 200$
PCR	2.31	7.55	25.78
PTPCR	0.19	0.46	1.64
m = 50	p=50	$p = 100$	$p = 200$
PCR	11.26	35.13	130.23
PTPCR	0.98	2.39	7.61
m = 100	p=50	$p = 100$	$p = 200$
PCR	22.58	70.07	251.94
PTPCR	1.98	4.84	15.90
m = 500	p=50	$p = 100$	$p = 200$
PCR	112.78	351.49	1275.26
PTPCR	10.09	24.69	78.20

Table 4.3: The computing time for m simulation run when $n = 1000$

m = 10	p=50	$p = 100$	$p = 200$
PCR	4.63	12.43	42.38
PTPCR	0.39	0.93	2.54
m = 50	p=50	$p = 100$	$p = 200$
PCR	22.20	61.85	209.74
PTPCR	1.79	4.65	12.59
m = 100	p=50	$p = 100$	$p = 200$
PCR	43.56	122.46	420.55
PTPCR	3.64	9.69	24.68
m = 500	p=50	$p = 100$	$p = 200$
PCR	223.97	646.05	2098.58
PTPCR	19.52	45.71	119.76

Table 4.2 and table 4.3 show the computing time for model PCR and PTPCR under the sample size $n = 500$ and $n = 1000$. As is shown in the table, the PTPCR method took much less time to compute the model compared with the traditional way of computing PCR. In addition, the computing time for PCR model increases dramatically while our PTPCR model increases gently and the difference between these two models becomes more and more obvious when the dimension is getting large. Thus, our PTPCR model beat PCR model in term of the computing time.

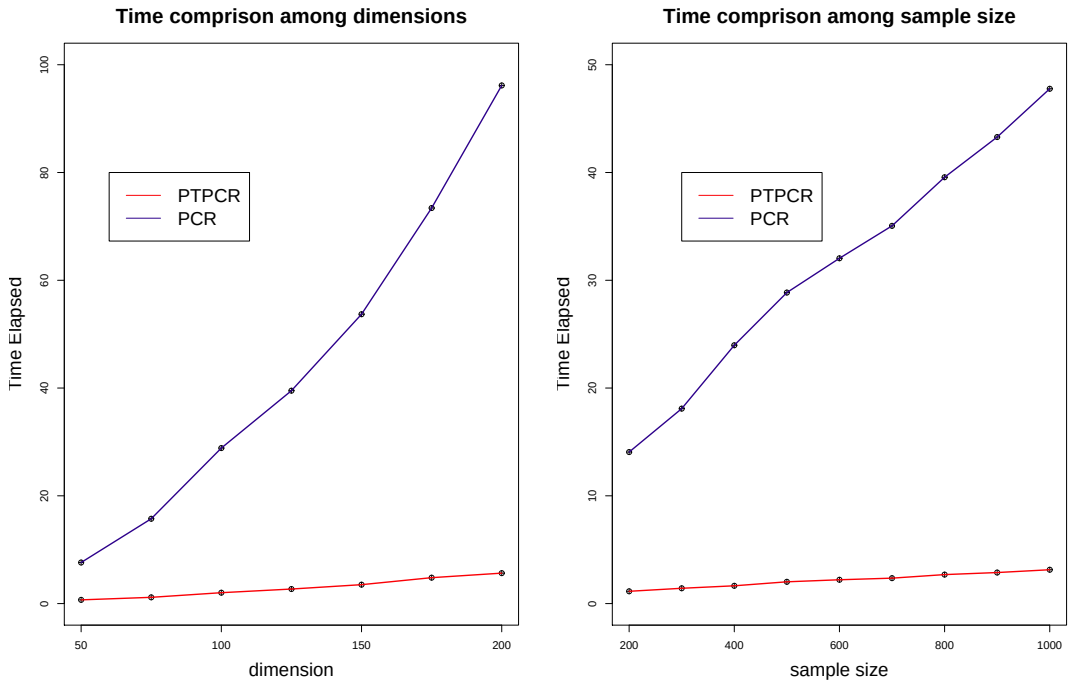


Figure 4.3: Figure of time elapsed with different dimensions and sample size

In figure 4.3, the left graph shows the relationship between the computing time and dimension when the sample size was set as 500, the right graph shows the computing time and sample size when the dimension was set as 100. The computing time was recorded basing on 50 times run. According to the figure, the computing time has the higher slope with the dimension and the sample size in PCR than PTPCR.

Last but not least, we present the average prediction error from five models (PTPCR, PTPCR-V2, PTPCR-V1, PCR, OLS) under setting IV, V, VI and VII. The average prediction error was basing on 300 times run. The results are shown in following.

Average prediction error under setting IV (strong signal)

Table 4.4: n=1000; k=7; $\alpha = k : 1$; $\sigma = 0.05$;

p=100	PTPCR	PTPCRV2	PTPCRV1	PCR	OLS
ρ=0.3	1.72115	1.69591	1.72130	1.72123	1.72146
ρ=0.6	1.84668	1.80976	1.84667	1.84691	1.84835
ρ=0.9	2.01842	1.98477	2.01841	2.01841	2.01985
p=300	PTPCR	PTPCRV2	PTPCRV1	PCR	OLS
ρ=0.3	1.56168	1.54277	1.56180	1.56438	1.56820
ρ=0.6	1.68531	1.65718	1.68682	1.68612	1.69230
ρ=0.9	1.90349	1.87199	1.90360	1.90359	1.91024
p=500	PTPCR	PTPCRV2	PTPCRV1	PCR	OLS
ρ=0.3	1.44310	1.42866	1.44316	1.44915	1.46089
ρ=0.6	1.62190	1.59951	1.62207	1.62381	1.63834
ρ=0.9	1.94668	1.91987	1.94696	1.94731	1.96493

Average prediction error under setting V (weak signal)

Table 4.5: n=1000; k=7; $\alpha = (k:1)/k$; $\sigma = 0.05$;

p=100	PTPCR	PTPCRV2	PTPCRV1	PCR	OLS
ρ=0.3	1.40401	1.37618	1.41501	1.40906	1.45130
ρ=0.6	1.49015	1.46446	1.49122	1.49182	1.53918
ρ=0.9	1.53031	1.48848	1.53002	1.53060	1.57727
p=300	PTPCR	PTPCRV2	PTPCRV1	PCR	OLS
ρ=0.3	1.31516	1.29071	1.31509	1.32662	1.50940
ρ=0.6	1.38657	1.36014	1.38669	1.39284	1.58547
ρ=0.9	1.43424	1.39854	1.43528	1.43479	1.63071
p=500	PTPCR	PTPCRV2	PTPCRV1	PCR	OLS
ρ=0.3	1.23589	1.21813	1.23975	1.25615	1.70644
ρ=0.6	1.33314	1.30847	1.33295	1.34159	1.78658
ρ=0.9	1.46082	1.42747	1.46053	1.46124	1.92208

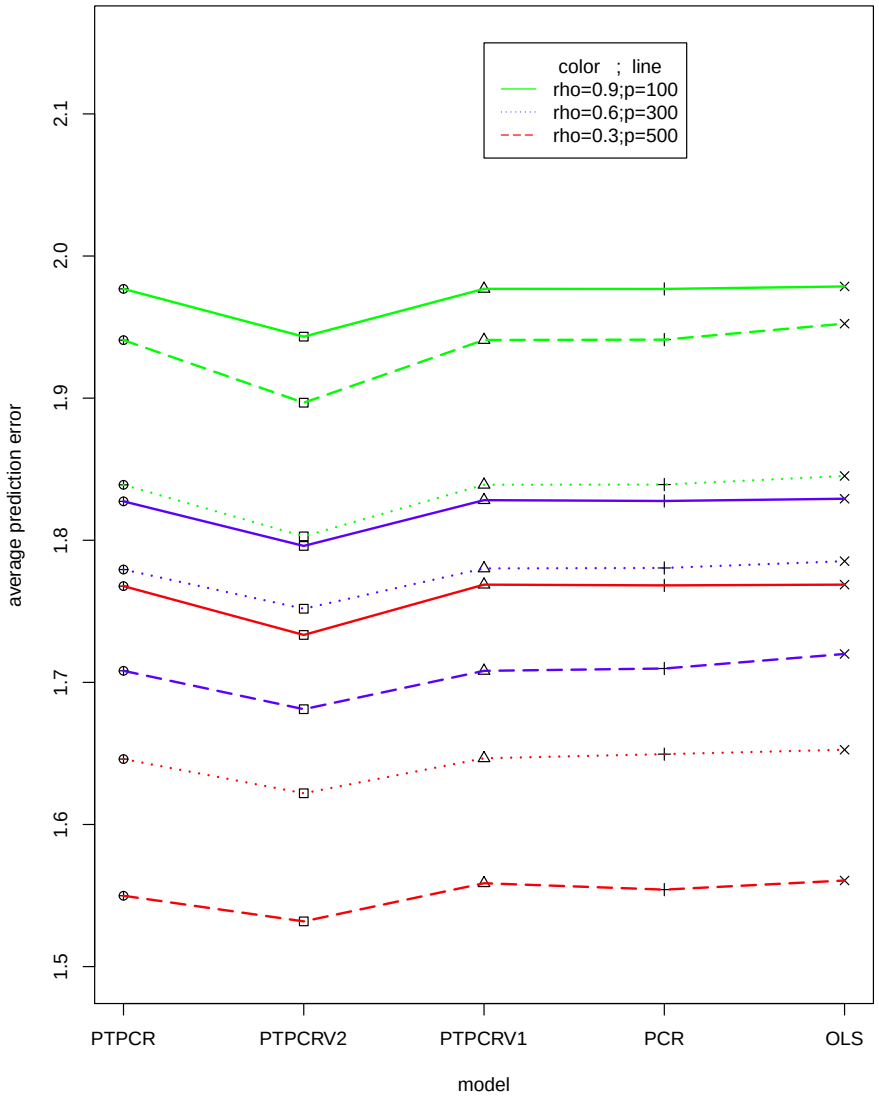


Figure 4.4: Plot of average prediction error under setting VI

Table 4.4 and 4.5 show the average prediction error from the five models under setting IV and V. It can be seen that model PTPCR-V2 has the smallest average prediction error, model OLS has the largest average prediction error and model PTPCR, PCR, PTPCR-V1 stand in the middle. It means that PTPCR-V2 performs

best and OLS performs worst with the others in the middle in terms of the average prediction error. Figure 4.4 is the average prediction error under setting VI. It shows the same results as table 4.4 and 4.5.

Average prediction error under setting VII

Table 4.6: n=1000; k=7; p=300;α=(k:1)/k

σ=0.05	PTPCR	PTPCRV2	PTPCRV1	PCR	OLS
ρ=0.3	1.31516	1.29071	1.31509	1.32662	1.50940
ρ=0.6	1.38657	1.36014	1.38669	1.39284	1.58547
ρ=0.9	1.43424	1.39854	1.43528	1.43479	1.63071
σ = 1	PTPCR	PTPCRV2	PTPCRV1	PCR	OLS
ρ=0.3	1.07129	1.16064	0.99919	1.07612	1.44350
ρ=0.6	1.09323	1.19054	0.99988	1.09489	1.46187
ρ=0.9	1.12364	1.21060	1.00174	1.12334	1.48882

In table 4.6, the average prediction error under settings VII is presented. In terms of the performance of each model, the ranking varies as the noise changes. When the noise is relatively small, σ equals to 0.005 here, model PTPCRV2 performs best. However, when the noise σ equals to 1, model PTPCRV1 performs best. It makes sense because PTPCR-V2 chooses the PC's by considering the correlation between the response and the predictors while PTPCR-V1 chooses the PC's by considering the correlation among the predictors only. Thus, the performance of model PTPCR-V2 is lowered when the correlation between the response and the predictors is weakened.

4.2 Data Example

In this section, two real data examples were used to evaluate our proposed models. One is the large real data set and the other one is the moderate data set. In the large data set, we separated it to training data and test data with the same sample size.

The training data set was used to build the model and the test data set was used to test the model. In the moderate data set, we divided it into 10 folds. Then, we used nine folds to build the model and the remaining one to test the model sequentially. Again, the evaluation was basing on the number of selected PC's, computing time and prediction error.

4.2.1 Real Data Example 1

The first data set is the friedman1 data set from R package `mlbench`. The function `mlbench.friedmen1` is used to produce the data set. The basic idea is simulating the design matrix, generating the response basing on the design matrix and plus noise at the end. The detail process is as follows. Firstly, the ten input variables are simulated by uniform distribution on the interval [0,1] independently. Secondly, the first five variables will be chosen to generate the output variable by the formula $y = 10sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5$. At last, the noise ϵ is added to the output variable, where ϵ is normally distributed with mean 0 and the standard deviation sd.

This data set is frequently used in regression problems. It has ten predictors and one response. In order to intensify the noise of the data set, we added another 30 columns. These columns were simulated via normal distribution with the column mean 0 and the standard deviation sd. In addition, we assigned the sample size n to 2000 for the train data set and the test data set as well. Then we have a train data set and a test data set both with 2000 rows and 41 columns. The results of all the five models are presented as follows,

Results of real data example 1

Table 4.7: Pmse, computing time and the number of selected PC's from different models

PMSE	PTPCR	PTPCRV2	PTPCRV1	PCR	OLS
	0.26168	0.26166	0.26224	0.26168	0.26168
Running time	PTPCR	PTPCRV2	PTPCRV1	PCR	OLS
	0.41857	0.37857	0.40429	0.88428	0.03286
Selected PC's	PTPCR	PTPCRV2	PTPCRV1	PCR	OLS
	40	37	40	40	40

In table 4.7, the prediction error, the computing time and the number of selected PC's are shown. Among all the five models, our proposed model PTPCR-V2 has the smallest prediction error and it means that PTPCR-V2 performs best. All the other four models perform similarly in terms of prediction error. In addition, OLS has the shortest computing time because there is not variable selection involved in the model and all the variables will be used. Apart from OLS, model PTPCRV2 used the shortest time and model PCR used the longest time. In terms of selected PC's, PTPCR-V2 used 37 PCs with the 14th, 35th and 39th PC were excluded and the other four models used all the 40 PCs.

4.2.2 Real Data Example 2

In the second real data example, the data set is from UCI Machine Learning Repository. It is denoted as Concrete Compressive Strength Data Set (CCSDS). Professor I-Cheng Yeh from Department of Information Management Chung-Hua University provided this data set. CCSDS is used to test the compressive strength of the concrete, one of the most important materials in civil engineering.

This data set consists of 1030 observations and has one quantitative output vari-

able and 8 quantitative input variables. The output variable (response) is the concrete compressive strength (ccs) with the unit MPa and the input variables (independent variables) are cement, blast furnace slag (bfs) , fly ash (fa), water, superplasticizer (sp), coarse aggregate (ca), fine aggregate (fae) and age. Except the age has the unit day, all the other independent variables have the unit kg in a m^3 mixture. CCSDS has no missing value and it is always used in regression problems.

In order to reinforce the collinearity for the data set, we added more columns. These columns were created by multiplying two variables together. First, we multiplied every predictor to itself. Then, we multiplied the first predictor cement to every other predictor. In addition, every original variable were cubed. At last, we made the cement into power of 4th. The final variables are listed in table 4.8,

Table 4.8: Predictors in data set 2

Original	Squared	Intersecting	Cubic	Biquadrate
cement	$cement^2$	N.A.	$cement^3$	$cement^4$
bfs	bfs^2	cement·bfs	bfs^3	N.A.
fa	fa^2	cement·fa	fa^3	N.A.
water	$water^2$	cement·water	$water^3$	N.A.
sp	sp^2	cement·sp	sp^3	N.A.
ca	ca^2	cement·ca	cat^3	N.A.
fae	fae^2	cement·fae	fae^3	N.A.
age	age^2	cement·age	age^3	N.A.

Then, there are 32 predictors in the data set with sample size 1030. Since the sample size is not large, the 10 folds cross validation method was used to get the prediction error for the five models. We equally and randomly divided the data set into 10 data subsets. 1 out of the 10 subsets was chosen to be the test data set and the remaining 9 subsets were combined as the train data set. Then we fitted the five models using the train data set and did the prediction by using the test data

set. A sum of squared distance between the predicted response and true response was calculated. The test data set was given by the order of the 10 subsets from the first one to last one sequentially and the train data set was given by the remaining 9 subsets. In this manner, we fitted the model ten times. The prediction error (PMSE) was obtained by summing the sum of squared errors and divided it by the sample size. In the meantime, we recorded the computing time for each model. The results for this data example are given in following table,

Results of real data example 2

Table 4.9: Pmse, computing time and the number of selected PC's from different models

PMSE	PTPCR	PTPCRV2	PTPCRV1	PCR	OLS
	0.16367	0.16341	0.17379	0.16409	0.16425
Running time	PTPCR	PTPCRV2	PTPCRV1	PCR	OLS
	1.2843	0.8471	0.9171	3.1214	0.1957
Selected k	PTPCR	PTPCRV2	PTPCRV1	PCR	OLS
	31	25	32	31	32

Table 4.9 shows the prediction error, computing time and the selected number of PC's from the real data example. As is shown in the table, PTPCRV2, PTPCR, PCR, OLS and PTPCRV1 are ordered in an increasing order in terms of PMSE. It means our proposed model PTPCRV2 performs best. In addition, basing on the running time, except OLS, the PTPCRV2 spent the least time to fit the model. In terms of the number of selected PC's, PTPCR-V1 and OLS used all the PC's, PTPCR and PCR used 31 PC's and the PTPCR-V2 used 25 PC's with the 12th, 18th, 19th, 20th, 24th, 31st and 32nd PC's were excluded.

Chapter 5

Conclusion and Future Work

This chapter, the last chapter of this thesis, is devoted to present the conclusion of the work we have done and point out some the future research avenues. In this thesis work, we have proposed three methods for conducing principal components regression analysis: pre-tuned principal component regression (PTPCR), pre-tuned principal component regression variant 1 (PTPCR-V1) and pre-tuned principal component regression variant 2 (PTPCR-V2). Then we illustrated the relationship among our proposed methods and some other methods including ridge regression (RR) and partial least squares regression (PLSR). In addition, we compared our proposed methods to some traditional methods such as ordinary least squares (OLS) and principal component regression (PCR) via simulation.

The remainder of this chapter is structured as follows. Section 5.1 is a summary of the thesis work. In this section, we will summarize our work, conclude the simulation results in short, and state the advantages of our methods as well. Section 5.2 discusses some future work of our project.

5.1 Summary

It is widely known that OLS regression coefficient estimates perform poorly when there is a multi-collinearity problem among the predictors in the design matrix. The coefficient estimates may have the large probability of being far away from the true regression coefficients. PCR is one of the most popular proposals to overcome this problem. The idea associated with PCR is regressing the response on several leading

principal components (PC's), each given by a linear transformation of the original predictors. The PCR, however, is quite computationally inefficient since it has to examine a number of models.

Herein, we proposed our first model PTPCR which determines the number of PC's beforehand. The main idea in PTPCR is to replace the indicator function by an approximating smooth sigmoid function. To approximate well, we fix the shape parameter at a large value in the smooth sigmoid function. Then, in our second proposed model, PTPCR-V1, we leave the fixed large shape parameter free so that it is adaptively estimated from the data. This would naturally lend us more predictive power. At last, we proposed our third model, PTPCR-V2, which orders the PC's by the magnitude of the regression coefficients with the response. This will help deal with scenarios where the response is more correlated with PCs that come last and hence give us even more predictive power. In this third model, we fix the shape parameter as a large constant.

Both the simulation and real data examples are used to investigate and compare the performance of the three models we proposed with the available methods. In terms of computational time and prediction accuracy, we have the following conclusions based on empirical results: First, all the three proposed models perform better than the conventional principal component regression in general. Secondly, PTPCR performs similarly to the conventional PCR in term of the prediction error. However, the computational time of PTPCR is obviously reduced compared with PCR especially when the dimensions is large. Thirdly, PTPCR-V1 and PTPCR-V2 not only reduce the computational time compared to PCR, but also decrease the prediction error. These two models outperform PCR and PTPCR under various experimental settings. Fourthly, in terms of comparison between PTPCR-V1 and PTPCR-V2, PTPCR-V2 performs better under most of the experimental settings and the real data example. However, when the noise parameter σ changes from 0.05 to 1 with all other parameters fixed, PTPCR-V1 has a smaller prediction error than PTPCR-

V2. Fifthly, our methods change the discrete nature of selecting PC's in PCR to a smooth process. Lastly, our proposed methods are more flexible since we can choose the PC's based on either eigenvalue λ_j or regression coefficient α_j.

5.2 Future Work

In the future work, we will continue our research efforts in extending our current results and deepening our study of the proposed methods in theory. These include

1. In model PTPCR-V2, we have fixed the shape parameter a as a large constant. It is natural to extend the mode by leaving a free, as in PTPCR-V1. This modification is expected to help further reduce the prediction error.

2. Our current proposed methods use a logistic function with either one or two parameters to substitute the indicator function. The logistic function here has a symmetrical S shape. We can extend the models by using the more flexible generalised logistic function [31] that allows for unsymmetrical S shapes. It has the form as follows

$$Y(t) = A + \frac{K - A}{(C + Qe^{-Bt})^{1/\nu}}, \tag{5.1}$$

   where A is the lower asymptote, K is the upper asymptote, B is the growth rate, Q is related to $Y(0)$ and controls the symmetric, and C is typically 1. By adjusting the parameters in equation (5.1), the generalised logistic function can be used to replace the indicator function to generate new PTPCR models.

3. The kernel trick has been used in the principal component regression by many authors including Rosipal et al. (2001)[32][33], Jade et al. (2003)[13], Hoegaerts et al. (2005)[10] and Wibowol et al. (2012)[39]. It is clear that the kernel trick can be combined into our proposed methods.

4. Our work can also be extended to the generalized linear models (GLM; Nelder and Wedderburn, 1972 [28] and Agresti, 2007 [1]) for other types of responses.

5. We did not consider the scenario in which the dimension of predictor matrix p is greater than the number of observations n in this work. In our future work, it is one of the aspects we will focus on.

6. We will study conditions under which our proposed methods are guaranteed to have the smaller prediction error than OLS or PCR theoretically.

References

[1] Agresti, A. (2007), An Introduction to Categorical Data Analysis, *Wiley Series in Probability and Statistics*,2nd Edition.

[2] Akaike, H. (1973), Information theory and an extension of the maximum likelihood principle, *In Proceedings of Second International Symposium on Information Theory*, (eds B. N. Petrov and F. Csaki). Budapest: Akademiai Kiado, 267–281.

[3] Allen, D. M. (1974), The relationship between variable selection and data augmentation and a method for prediction, *Technometrics*, Vol. 16, pp. 125–127.

[4] Artemiou, A. and Li, B. (2009), On Principal Components and Regression: A Statistical Explanation of a Natural Phenomenon, *Statistica Sinica*, Vol. 19, pp. 1557–1565.

[5] Bair, E., Hastie, T., Paul, D. and Tibshirani, R. (2006), Prediction by supervised principal components, *Journal of the American Statistical Association*, Vol. 101, No. 473, pp. 119–137.

[6] Brent, R. (1973), Algorithms for Minimization without Derivatives, *Englewood Cliffs N.J.: Prentice-Hall.*

[7] Franklin, S. B., Gibson, D. J, Robertson, P. A., Pohlmann, J. T and Fralish, J. S. (1995), Parallel Analysis: A method for determining significant principal components, *Journal of Vegetation Science*, Vol. 6, pp. 99-106.

[8] Hadi, A. S., and Ling, R. F. (1998), Some Cautionary Notes on the Use of Principal Components Regression, *The American Statistician*, Vol. 52, pp. 15–19.

[9] Hastie, T., Tibshirani, R. and Friedman, J. (2009), The Elements of Statistical Learning, *Data Mining, Inference and Prediction*, 2nd Edition. Springer Series in Statistics.

[10] Hoegaerts, L., Suykens, J. A. K., Vandewalle, J. and De Moor, B. (2005), Subset based least squares subspace in reproducing kernel Hilbert space, *Neurocomputing*, 293-323.

[11] Hoerl, A. E. and Kennard, R. W. (1970), Ridge regression: biased estimation for non-orthogonal problems, *Technometrics*, Vol. 12, pp. 55–67.

[12] Hwang, J. T. G. and Nettleton, D. (2003), Principal Components Regression with Data-Chosen Components and Related Methods, *Technometrics*, Vol. 45, pp. 70–79.

[13] Jade, A. M., Srikanth, B., Kulkari, B. D., Jog, J. P. and Priya, L. (2003) Feature extraction and denoising using kernel pca, *Chem. Eng. Sci.*, 58, 4441-4448.

[14] Jollife, I. T. (1982), A Note on the Use of Principal Components in the Regression, *Applied Statistics*, Vol. 31, pp. 300–303.

[15] Kiefer, J. (1953), Sequential minimax search for a maximum, *American Mathematical Society*, 4(3), pp. 502–506.

[16] LeBlanc, M. and Tibshirani, R. (1996), Combining estimates in regression and classification, *Journal of the American Statistical Association*, Vol. 91, No. 436, pp. 1641–1650.

[17] Li, B. (2007), Comment: Fisher Lecture: Dimension Reduction in Regression, *Statistical Science*, Vol. 22, No. 1, pp. 32–35.

[18] Makridakis, S. G., Wheelwright, S. C. and Hyndman, R. J. (1998), Forecasting: Methods and Applications, 3rd edition.

[19] Mallows, C. L. (1973), Some Comments on C_p, *Technometrics*, Vol. 15, No. 4, pp. 661–675.

[20] Manage, A. B. W. and Scariano, S. M. (2013), An Introductory Application of Principal Components to Cricket Data, *Journal of Statistics Education*, Vol. 21.

[21] Mandel, J. (1982), Use of the Singular Value Decomposition in Regression Analysis, *The American Statistician*, Vol. 36, No. 1, pp. 15-24.

[22] Mansfield, E. R., Webster, J. T. and Gunst, R.F. (1977), An Analytic Variable Selection Technique for Principal Component Regression, *Applied Statistics*, Vol. 26, pp. 34–40.

[23] Mansfield, E. R. (1978), PCR: Principal component regression analysis, *Journal of Marketing Research*, Vol. 15, No. 3, pp. 471–472.

[24] Martinez, J. G., Liang, F., Zhou, L. and Carroll, R. J. (2010), Longitudinal Functional Principal Component Modelling via Stochastic Approximation Monte Carlo, *The Canadian Journal of Statistics*, Vol. 38, pp. 256-270.

[25] Massy, W. F. (1965), Principal Components Regression in Exploratory Statistical Research, *Journal of The American Statistical Association*, Vol. 60, pp. 234–256.

[26] Mevik, B. and Wehrens, R. (2007), The pls Package: Principal Component and Partial Least Squares Regression in R, *Journal of Statistical Software*, Vol. 18.

[27] Nelder, J. A. and Mead, R. (1965), A simplex algorithm for function minimization, *Computer Journal*, 7, pp. 308-313.

[28] Nelder, J. and Wedderburn, R. W. M.(1972), Generalized linear models, *Journal of the Royal Statistical Society* 135, pp. 370–384

[29] Ni, L. (2011), Principal Component Regression Revisited, *Statistica Sinica*, Vol. 21, pp.741–747.

[30] Pearson, K. (1901), On Lines and Planes of Closest Fit to Systems of Points in Space, *Philosophical Magazine*, Vol. 2(11), pp. 559–572.

[31] Richards, F. J. (1959), A Flexible Growth Function for Empirical Use, *Journal of Experimental Botany*, 10(2), pp. 290-300.

[32] Rosipal, R., Girolami, M., Trejo, L. J. and Cichoki, A. (2001) Kernel pca for feature extraction and de-noising in nonlinear regression, *Neural Computing and Applications*, 231-243.

[33] Rosipal, R., Trejo, L. J. and Cichoki, A. (2001) Kernel principal component regression with em approach to nonlinear principal component extraction, *Technical Report, University of Paisley, UK*.

[34] Sahriman, S., Djuraidah, A. and Wigena, A. H. (2014), Application of Principal Component Regression with Dummy Variable in Statistical Downscaling to Forecast Rainfall, *Open Journal of Statistics*, Vol. 4, pp. 678–686.

[35] Schwarz, G. (1978), Estimating the Dimension of a Model, *The Annals of Statistics*, Vol. 6, No.2, pp. 461–464.

[36] Shibata, R. (1980), Asymptotically effcient selection of the order of the model for estimating parameters of a linear process, *The Annuals of Statistics*, Vol. 8, No.1, pp. 147–164.

[37] Stone, M. and Brooks, R. J. (1990), Continuum regression: cross-validated sequentially constructed prediction embracing ordinary least squares, partial least squares and principal components regression, *Journal of Royal Statistical Society*, Vol. 52, No.2, pp. 237–269.

[38] Wahba, G. (1977), Practical Approximate Solutions to Linear Operator Equations When the Data Are Noisy, *SIAM Journal Numerical Analysis*, Vol. 14, No. 4, pp. 651–667.

[39] Wibowol, A. and Yamamoto, Y.(2012), A Note on Kernel Principal Component Regression, *Computational Mathematics and Modeling*, Vol. 23, No. 3, pp. 350–367.

[40] Wold, H. (1975), Soft Modeling by Latent Variables: The Nonlinear Iterative Partial Least Squares(NIPALS) Approach, *Perspectives in Probability and Statistics, in Honor of M. S. Bartlett*, pp. 117-144.

[41] Yang, Y. (2005), Can the strengths of AIC and BIC be shared? A conflict between model identification and regression estimation, *Biometrika*, 92(4), pp. 937–950.

Appendix A

The R Code

```
# ================================
# FUNCTION rdat() GENERATES DATA
# ================================

library(MASS)
library(pls)

# ================================
# GENERATES DATA FOR PTPCR BY VAR
# ================================
rdat.PCR <- function(n, alpha= (3:1)/3, mu.x=rep(0, 10), rho.x=.5, sigma=1)
{
p <- length(mu.x);
k <- length(alpha)
if (k >p) stop("The length of alpha (k) must be no larger than the length
of mu.x (p)!")

# GENERATE X
S <- matrix(1, p, p)
for (i in 1:p){
for (j in 1:p){
```

```
S[i, j] <- rho.x^(abs(i-j))
}};
X <- mvrnorm(n=n, mu=mu.x, Sigma=S, tol=1e-6, empirical=F)
X <- scale(X, center=TRUE, scale=TRUE)  # STANDARDIZE
U <- svd(X)$u; # print(U)

# GENERATE y
y <- U[, 1:k]%*%alpha + rnorm(n, mean = 0, sd = sigma)

# OUTPUT
dat <- data.frame(cbind(X, y))
colnames(dat) <- c(paste("x", 1:p, sep=""), "y")
return(list(dat=dat, k0=k))
}

# ==================================
# GENERATES DATA FOR PTPCR BY COEF
# ==================================
rdat.coef <- function(n, alpha, mu.x=rep(0, 10), rho.x=.5, sigma=0.05)
{
p <- length(mu.x);
if (length(alpha) != p) stop("The length of alpha must be the same as the
length of mu.x (p)!")

# GENERATE X
S <- matrix(1, p, p)
for (i in 1:p){
for (j in 1:p){
```

```
S[i, j] <- rho.x^(abs(i-j))
}};
X <- mvrnorm(n=n, mu=mu.x, Sigma=S, tol=1e-6, empirical=F)
X <- scale(X, center=TRUE, scale=TRUE)  # STANDARDIZE
U <- svd(X)$u; # print(U)

# GENERATE y
y <- U%*%alpha + rnorm(n, mean = 0, sd = sigma)

# OUTPUT
dat <- data.frame(cbind(X, y))
colnames(dat) <- c(paste("x", 1:p, sep=""), "y")
return(list(dat=dat))
}

#=================================================
# Generate data with free a and c
#=================================================
rdatac.PCR <- function(n,alpha=(3:1)3,
mu.x=rep(0,50),rho.x=0.5,sigma=0.1,a=10,c=median(alpha))
{
p <- length(mu.x);
k <- length(alpha)
if (k > p) stop("The length of alpha (k) must be no large than
the length of mu.x (p)!")
# generate X
S <- matrix(1,p,p)
for (i in 1:p){
```

```
for (j in 1:p){
S[i,j] <- rho.x^(abs(i-j))
}}
X <- mvrnorm(n=n,mu=mu.x,Sigma=S,tol=1e-6,empirical=F)
X <- scale(X,center=TRUE,scale=TRUE)
U <- svd(X)$u;

#generate y
W <- diag(a*(alpha -c))
y <- U[,1:k]%*%W%*%alpha + rnorm(n,mean=0,sd=sigma)
# output
dat <- data.frame(cbind(X,y))
colnames(dat) <- c(paste("x",1:p,sep=""),"y")
return(list(dat=dat,k0=k))
}

# ============================================
# THE OBJECTIVE FUNCTION - GCV, AIC, AND BIC
# ============================================

cexpit <- function(x) (1+tanh(x/2))/2

Q.obj <- function(c, a,
ss.y, d, gamma, n,
criterion=c("AIC", "BIC", "GCV"),
method=c("singular.value", "coefficient"))

{
```

```
if (method=="singular.value") s <- cexpit(a*(d-c))
else if (method=="coefficient") s <- cexpit(a*(c(t(gamma)^2)-c))
else stop("Wrong argument for method=!")
SSE <- ss.y - t(gamma)%*%diag(s)%*% gamma
EDF <- sum(s)
if (criterion=="GCV") Q.n <- SSE/((n - EDF)^2)
else {
lambda0 <- ifelse(criterion=="BIC", log(n), 2) # penalty for AIC or BIC
Q.n <- n*log(SSE) + lambda0*EDF
}
return(Q.n)
}


#=========================================
# The objective function for free a and c
#=========================================
cexpit <- function(x){(1+tanh(x/2))/2}

Q.objac <- function(c, ss.y,d,gamma,n,criterion=c("AIC", "BIC", "GCV"))
{
c1 <- c[1]
c2 <- c[2]
s <- cexpit(c1*(d-c2))
SSE <- ss.y-t(gamma)%*%diag(s)%*%gamma
EDF <- sum(s)
if (criterion=="GCV") Q.n <- SSE/((n-EDF)^2)
else {
```

```
lambda0 <- ifelse(criterion=="BIC", log(n),2) #penalty for AIC or BIC
Q.n <- n*log(SSE) + lambda0*EDF
}
return(Q.n)
}

# ============================
# PRE-TUNED PCR by var and coe
# ============================

pretuned.PCR <- function(formula, data, split,
a=NULL, criterion=c("AIC", "BIC", "GCV"),
method=c("singular.value", "coefficient"),
details=F)
{
call <- match.call()
# CHECK THE data= ARGUMENT
if (missing(data)) data <- environment(formula)
# OBTAIN THE DESIGN MATRIX X AND RESPONSE y

mf <- match.call(expand.dots = FALSE)
m <- match(c("formula", "data", "subset", "weights", "na.action"),
names(mf), 0L)
mf <- mf[c(1L, m)]
mf$drop.unused.levels <- TRUE
mf[[1L]] <- quote(stats::model.frame)
mf <- eval(mf, parent.frame())
mt <- attr(mf, "terms")
```

```
Y <- model.response(mf, "any")
if (length(dim(Y)) == 1L) {
nm <- rownames(Y); dim(Y) <- NULL
if (!is.null(nm)) names(Y) <- nm
}

# ADOPTED FROM FUNCTION lm.ridge()
X <- if (!is.empty.model(mt)) model.matrix(mt, mf, contrasts)
else matrix(, NROW(Y), 0L)
n <- NROW(X); p <- NCOL(X)
if (Inter <- attr(mt, "intercept")) {
Xm <- colMeans(X[, -Inter]); p <- p - 1
# EVEN IF THE FORMULA HAS AN INTERCEPT TERM, INTERACEPT WILL BE REMOVED.
X <- scale(X[, -Inter], center=Xm, scale=F)
} else {
Xm <- colMeans(X)
X <- scale(X, center=Xm, scale=F)
}
Ym <- mean(Y); Y <- Y - Ym
Xscale <- drop(rep(1/n, n) %*% X^2)^0.5  # SD OF X
X <- scale(X, center=FALSE, scale=Xscale )
Xnames <- colnames(X)
if (details) print(head(X))

# SVD OF X
fit.svd <- svd(X)
d <- fit.svd$d; U <- fit.svd$u; V <- fit.svd$v
```

```
# FIND OPTIMAL c
# ---------------------
if (details) print(cbind(dim(U), dim(Y)))
gamma <- t(U)%*%Y; ss.y <- sum(Y^2)
if (is.null(a)) a <- n
L0 <- ifelse(method=="singular.value", min(d)-0.001, min(gamma^2)-0.001)
U0 <- ifelse(method=="singular.value", max(d), max(gamma^2))

if(missing(split)) split=1
cseq <- c()
objectiveseq <- c()
ed <- (U0-L0)/split
seq <- 0
for (i in 1:split){
lower = L0 + seq*ed
upper = L0 + (seq+1)*ed
optimization <- optimize(f=Q.obj,lower = lower, upper = upper,
maximum=F, tol=.Machine$double.eps^0.25,
a=a, ss.y=ss.y, d=d, gamma=gamma, n=n,criterion=criterion,
method=method)
cseq[i] <- optimization$minimum
objectiveseq[i] <- optimization$objective
seq <- seq + 1
}
IC.min <- min(objectiveseq)
c.star <- cseq[which.min(objectiveseq)]

# COMPUTE PCR DIRECTLY
```

```
# ------------------------------------------
if (method=="singular.value") {
k.star <- sum(d > c.star)
U.k <- U[, d > c.star]
}
else if (method=="coefficient") {
k.star <- (1:length(gamma))[gamma^2 > c.star]
U.k <- U[, gamma^2 > c.star]
}
else stop("Wrong argument for method=!")
fitted.pcr <- U.k%*%t(U.k)%*%Y + Ym
if(method=="singular.value") {
coeff <- V[,1:k.star]%*%solve(diag(d)[1:k.star,1:k.star])%*%gamma[1:k.star]}
else{
coeff <- V[,k.star]%*%solve(diag(d)[k.star,k.star])%*%gamma[k.star]}
# USING {pls} PACKAGE
# fitted.pcr <- pcr(formula, data=data, ncomp=k.star)
out <- list(c.star=c.star, IC.min=IC.min, k.star=k.star, fitted.pcr=fitted.pcr,
U=U, V=V, d=d, gamma=gamma, coeff=coeff,
formula=formula, method=method, criterion=criterion,
sd.X = Xscale, mean.y = Ym, mean.X = Xm)
class(out) <- "pretuned.pcr"
return(out)
}

# ============================
# PRE-TUNED PCR by freed a
# ============================
```

```
pretuned.PTPCRV1 <- function(formula, data,
criterion=c("AIC", "BIC", "GCV"), details=F)
{
call <- match.call() #check the data=ARGUMENT
if (missing(data)) data <- environment(formula)
mf <- match.call(expand.dots=FALSE)
m <- match(c("formula","data", "subset", "weights", "na.action"),
names(mf), 0L)
mf <- mf[c(1L, m)]
mf$drop.unused.levels <- TRUE
mf[[1L]] <- quote(stats::model.frame)
mf <- eval(mf, parent.frame())
mt <- attr(mf, "terms")
Y <- model.response(mf, "any")
if (length(dim(Y)) == 1L) {
nm <- rownames(Y); dim(Y) <- NULL
if (!is.null(nm)) names(Y) <- nm
}
# ADOPTED FROM FUNCTION lm.ridge()
X <- if (!is.empty.model(mt)) model.matrix(mt, mf, contrasts)
else matrix(, NROW(Y), 0L)
n <- NROW(X); p <- NCOL(X)
if (Inter <- attr(mt, "intercept")) {
Xm <- colMeans(X[, -Inter]); p <- p - 1
# EVEN IF THE FORMULA HAS AN INTERCEPT TERM, INTERACEPT WILL BE REMOVED.
X <- scale(X[, -Inter], center=Xm, scale=F)
} else {
```

```
Xm <- colMeans(X)
X <- scale(X, center=Xm, scale=F)
}
Ym <- mean(Y); Y <- Y - Ym   # NOTE THAT Y IS CENTERED ONLY;
Xscale <- drop(rep(1/n, n) %*% X^2)^0.5  # SD OF X
X <- scale(X, center=FALSE, scale=Xscale )
Xnames <- colnames(X)
if (details) print(head(X))  ################
 #SVD of X
fit.svd <- svd(X)
d <- fit.svd$d; U <- fit.svd$u; V <- fit.svd$v
# find optimal c and a
if (details) print(cbind(dim(U), dim(Y)))
gamma <- t(U)%*%Y; ss.y <- sum(Y^2)
out <- optim(par=c(0.5,0.5), fn=Q.objac,method="Nelder-Mead",ss.y=ss.y, d=d,
gamma=gamma, n=n,criterion=criterion)
c.star <- out$par[2]
a.star <- out$par[1]

# COMPUTE PCR DIRECTLY
#----------------------

k.star <- sum(d > c.star)
U.k <- U[, d > c.star]

coeff <-  V %*% solve(diag(d))%*% diag(as.vector(cexpit(a.star*((d)-c.star))))
%*% gamma
fitted.pcr <- U.k%*%t(U.k)%*%Y + Ym
```

```
# USING {pls} PACKAGE
# fitted.pcr <- pcr(formula, data=data, ncomp=k.star)
out <- list(c.star=c.star, a.star=a.star, k.star=k.star,fitted.pcr=fitted.pcr,
U=U, V=V, d=d, gamma=gamma,coeff=coeff,
formula=formula, criterion=criterion,
sd.X = Xscale, mean.y = Ym, mean.X = Xm)
class(out) <- "pretuned.pcr"
return(out)
}


#============================
# predicition function by var
#============================
predictPTPCR<- function(object.PTPCR, newdata, cols.x){
fit <- object.PTPCR
p <- length(fit$mean.X)
X0 <- newdata[, 1:cols.x]
if (p!=NCOL(X0)) stop("The number of predictors in the new data does not
match!")

# standardize new data using mean and SD from the training
X <- scale(X0, center = fit$mean.X, scale = fit$sd.X)
y0 <- newdata[,cols.x+1]
pred <- mean(y0) + X%*%fit$coeff
return(pred)
}
```

Curriculum Vitae

Pei Wang was born on September 10, 1990 at Zhejiang Province in China. He is the first son of Zaihua Wang and Aili Xu. Both his parents are farmers and working industriously on the farmland. After finishing the middle school in town in 2006, he was admitted to Shengzhou No.1 Middle School to continue the high school. In three years, in the fall of 2009, he was admitted to Zhejiang University of Technology where he got his bachelor's degree in Information and Computing Science. He started his internship in Taikang Life Insurance Co., Ltd. when he was a senior student. During this time, he decided to continue study for a graduate degree.

In the spring of 2014, he entered University of Texas at El Paso as a graduate student majoring in Statistics. While in University of Texas at El Paso, after the first semester, he was funded as a teaching assistant until May of 2016. After getting his Master's Degree, he plans to continue study for a Ph.D. degree in the fall of 2016.

Permanent address: Unit 2, Building 16, Baihehuayuan, Shiji Square, Paojiang
Shaoxing, Zhejiang, China 312074