Projection Based Models for High Dimensional Data

A thesis presented for the degree of

Doctor of Philosophy of Imperial College London

September, 2011

Brian Victor Parulian McWilliams

Department of Mathematics

Imperial College London

180 Queen's Gate

London SW7 2BZ

I certify that this thesis, and the research to which it refers, are the product of my own work, and that any ideas or quotations from the work of other people, published or otherwise, are fully acknowledged in accordance with the standard referencing practices of the discipline.

Signed:

Copyright

Abstract

In recent years, many machine learning applications have arisen which deal with the problem of finding patterns in high dimensional data. Principal component analysis (PCA) has become ubiquitous in this setting. PCA performs dimensionality reduction by estimating latent factors which minimise the reconstruction error between the original data and its low-dimensional projection. We initially consider a situation where influential observations exist within the dataset which have a large, adverse affect on the estimated PCA model. We propose a measure of "predictive influence" to detect these points based on the contribution of each point to the leave-one-out reconstruction error of the model using an analytic PRedicted REsidual Sum of Squares (PRESS) statistic. We then develop a robust alternative to PCA to deal with the presence of influential observations and outliers which minimizes the predictive reconstruction error.

In some applications there may be unobserved clusters in the data, for which fitting PCA models to subsets of the data would provide a better fit. This is known as the subspace clustering problem. We develop a novel algorithm for subspace clustering which iteratively fits PCA models to subsets of the data and assigns observations to clusters based on their predictive influence on the reconstruction error. We study the convergence of the algorithm and compare its performance to a number of subspace clustering methods on simulated data and in real applications from computer vision involving clustering object trajectories in video sequences and images of faces.

We extend our predictive clustering framework to a setting where two high-dimensional views of data have been obtained. Often, only either clustering or

predictive modelling is performed between the views. Instead, we aim to recover clusters which are maximally predictive between the views. In this setting two block partial least squares (TB-PLS) is a useful model. TB-PLS performs dimensionality reduction in both views by estimating latent factors that are highly predictive. We fit TB-PLS models to subsets of data and assign points to clusters based on their predictive influence under each model which is evaluated using a PRESS statistic. We compare our method to state of the art algorithms in real applications in web-page and document clustering and find that our approach to predictive clustering yields superior results.

Finally, we propose a method for dynamically tracking multivariate data streams based on PLS. Our method learns a linear regression function from multivariate input and output streaming data in an incremental fashion while also performing dimensionality reduction and variable selection. Moreover, the recursive regression model is able to adapt to sudden changes in the data generating mechanism and also identifies the number of latent factors. We apply our method to the enhanced index tracking problem in computational finance.

Acknowledgements

Firstly, I wish to thank my supervisor Dr. Giovanni Montana for all the guidance, ideas and countless hours of discussions.

Thanks also to my friends and colleagues at Imperial Statistics and Edinburgh Informatics for many stimulating conversations over lunch, coffee and beers.

A special thanks to Mike McQuaid, Wei Lin Sung and Nick Houston for invaluable assistance both technical and non-technical.

An extra special thanks to Corina who made the tough times easier.

Finally, the biggest debt of gratitude is owed to my parents, Andrew and Ratna, and my brother George. This would not have been possible without your unwavering support and tireless encouragement. Your contribution to this work is immeasurable. *Terima kasih!*

Brian McWilliams

Table of contents

List of Figures

List of Tables

Glossary

CCA Canonical Correlations Analysis

iS-PLS Incremental Sparse Partial Least Squares

LOOCV Leave-one-out Cross Validation

MVPP Multi-view Predictive Partitioning

OLS Ordinary Least Squares

PCA Principal Components Analysis

PLS Partial Least Squares

PPSC Penalised Predictive Subspace Clustering

PRESS Predicted Residual Sum of Squares

ProPCA Predictive Robust Principal Components Analysis

PSC Predictive Subspace Clustering

SCC Spectral Curvature Clustering

SLBF Spectral Local Best Flats

SSC Sparse Subspace Clustering

SVD Singular Value Decomposition

TB-PLS Two-block Partial Least Squares

Chapter 1

Introduction

In recent years a large number of interesting problems in machine learning have emerged in a variety of applications, supported by many publicly available datasets. Exponential increases in computing power have allowed datasets of previously prohibitive dimensionality to become commonplace. Table 1.1 gives an example of different data types which are commonly encountered in real applications and the approximate dimensionality associated with data of that type. For example, in computational finance, the dimensionality may represent the number of constituent stocks in an index. In text mining, the dimensionality refers to the number of different, important words which appear in a document or webpage and in image recognition, the dimensionality typically refers to the number of pixels which make up each image. One of the great challenges in machine learning today is the development of methods to deal with such high-dimensional problems in an efficient manner.

Data Type	Dimensionality
Financial indices [61]	100
Web pages [10]	$1,000$
Digital photographs [33]	$10,000$
Gene expression [102]	$100,000$

Table 1.1: Examples of common problem domains and their approximate dimension. This highlights the range of problems which are considered "high-dimensional".

Broadly speaking, there are two paradigms for learning in high-dimensional data: *supervised* and *unsupervised* learning. In supervised learning, given multivariate predictor variables $\boldsymbol{X} \in \mathbb{R}^{N \times P}$, we aim to predict an observed response variable, $\boldsymbol{y} \in \mathbb{R}^{N \times 1}$. On the other hand, unsupervised learning refers to many types of problems where no response is observed. For example we may wish to partition $\boldsymbol{X}$ such that we detect naturally occurring clusters in the N rows (samples) of $\boldsymbol{X}$. In this setting, there is often no "ground truth" and so driving the learning process and objectively evaluating the results is often difficult. Throughout this work we use $\boldsymbol{X}$ to refer to a matrix of observations and $\boldsymbol{x}_i$ to refer to individual observations, rows of $\boldsymbol{X}$, as opposed to random variables.

When the number of variables, P, is large we encounter well known issues relating to the fact that many of the dimensions may be highly correlated, noisy or simply irrelevant for the task at hand. A common method of dealing with these problems is to recover a low-dimensional representation of the original data. This typically involves estimating a small number of *latent factors*, linear combinations of the original variables, which explain the important information in the data. The latent factors are estimated by projecting the data into a low-dimensional subspace which is in some way optimal such that important information for the task at hand is retained but irrelevant information such as noise is discarded. These latent factors can then be used to perform clustering or predictive modelling in place of the original data.

In this work we build on the idea of recovering low-dimensional projections of the data and extend these conceptually simple, linear models to solve more complex, non-linear problems in high-dimensions. In each case, the problems we address have emerged recently and typically have not been widely considered in the literature.

In the unsupervised case, principal component analysis (PCA) is a ubiquitous technique for performing dimensionality reduction [45]. PCA estimates a low-dimensional linear subspace oriented in the direction of maximal variance in the data and obtains the latent factors by projecting the observations onto this subspace. PCA has been successfully applied in a huge variety of application domains

spanning many branches of the physical, life and social sciences.

In Chapter 2 we briefly review penalised regression and PCA in the context of dealing with high-dimensional data and discuss several open problems in PCA regarding model selection. We examine the related questions of selecting the number of latent factors and detecting influential observations which have a detrimental effect on the PCA model fit. Since most methods for model selection involve determining the effect of removing a single observation from the PCA model, they require as many model fits as observations which is computationally expensive.

PCA is a linear method and this limits its use in real problems which may be highly non-linear. We consider one such situation where there are heterogeneous clusters present in the data. In this situation, a single PCA model does not fit the data well and instead we make the assumption that the points in each of the clusters belongs to a cluster-specific low-dimensional subspace. Here the problem then consists in estimating the subspace parameters and the cluster assignments simultaneously. Recently, a number of state-of-the-art methods have been proposed to tackle this problem. However, we recognise some important limitations common to most methods which are related to the problem of model selection.

In Chapter 3 we propose a simple, unified framework which solves the problems of model selection and identifying influential observations in a computationally efficient manner based on an closed-form expression for the predictive performance of the PCA model called the Predicted REsidual Sum of Squares (PRESS). We propose a measure for "predictive influence" based on the effect of each observation on the PRESS. We extend this framework for dealing with influence by proposing a robust PCA criterion which seeks to directly minimise the effect of influential observations by minimising the PRESS rather than the standard PCA reconstruction error. In Chapter 4 we apply our method of identifying influential observations to the subspace clustering problem. Here, we propose an iterative optimisation algorithm based on minimising the predictive influence within each cluster. We compare our *predictive* subspace clustering (PSC) algorithm with the state-of-the-art methods reviewed in Chapter 2 and find that it produces highly competitive results on simulated and real datasets. Part of the work in Chapters 3 and 4 appears

in [62].

In the second half of this work we consider a general supervised setting where the responses, $\boldsymbol{Y} \in \mathbb{R}^{N \times Q}$ are also allowed to be high-dimensional. This is sometimes referred to as the multi-view setting where each set of high-dimensional observations is considered to be a "view" of the data. In this setting, two-block partial least squares (TB-PLS) is a popular method for dimensionality reduction which can be seen as a generalisation of PCA to two views. TB-PLS estimates latent factors which explain maximal covariance between the views. However, the ultimate purpose of performing dimensionality reduction with TB-PLS is to model the predictive relationship between predictors and responses using the low-dimensional latent factors instead of the original variables. As with PCA, the issues of model selection and influential observations are important considerations. In order to address this we propose an efficient PRESS for TB-PLS

Recently, the problem of performing unsupervised clustering using multiple views jointly has emerged. In Chapter 5 we extend our predictive subspace clustering method in two views to attack the problem of *multi-view clustering* which is gaining popularity in the machine learning literature. Here, the assumption is that each pair of predictors and responses, $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}$ belongs to one of a number of heterogeneous clusters in the data. We propose a novel solution to the problem of recovering these cluster assignments which assumes that within each cluster, a predictive relationship exists between views. The problem of recovering the true clustering then amounts to one of identifying the optimal predictive models in the data which we drive by fitting TB-PLS models to the data and then minimising the predictive influence which each point exerts within each cluster. In order to validate our novel approach to multi-view clustering, we compare our proposed multi-view predictive partitioning (MVPP) algorithm with three other multi-view clustering algorithms in a variety of simulated settings and a real application to clustering web pages and academic papers. The efficient PRESS for TB-PLS appears in [60]

The final problem we tackle in Chapter 6 again concerns estimating a predictive model in high-dimensions. However, now we assume that each pair of points is indexed by time and is observed sequentially, as $\{\boldsymbol{x}_t, \boldsymbol{y}_t\}$. Furthermore, we assume

that the underlying data generating process is non-stationary and so the predictive relationship between views can change in time. The problem now amounts to estimating a temporally adaptive PLS model which is able to adjust quickly to sudden changes in the data. Since the data is high-dimensional we assume that some of the predictors do not contribute to the predictive relationship. Therefore, in order to improve our model, we also aim to identify which observations are unimportant and should be removed from the model. This amounts to solving a penalised PLS regression problem *on-line*. We propose a novel algorithm for incremental sparse PLS (iS-PLS) which is able to automatically adapt to changes in the underlying predictive model. We apply this method to the problem of financial index tracking where we attempt to identify which assets are important for predicting the movement of a financial index. We also address some of the issues relating to the ever-present problem of model selection. The work in Chapter 6 has appeared in [61].

Chapter 2

Learning linear subspaces in high-dimensions

In this chapter we discuss well known methods for learning with high-dimensional data in both supervised and unsupervised settings.

In the supervised setting, often the task is to estimate a set of coefficients which model a linear relationship between a set of predictor variables and a response. This is commonly achieved by performing ordinary least squares (OLS) regression. However, in high-dimensions there are well-known problems with the OLS solution. Penalised regression methods have become increasingly popular in the last decade. These methods are based on imposing a penalty on the size of the coefficients estimated using least squares regression which has the effect of identifying and downweighting "unimportant" variables which do not contribute to the response. The most common penalty, the Lasso [88], places a restriction on the ℓ_1 norm of the coefficients. This causes some of the variables to be removed from the model and results in a sparse solution.

Principal component analysis (PCA) [45] has become a popular technique for unsupervised learning in high-dimensional data. PCA assumes that the important variation in the data can be well represented by a low-dimensional linear subspace. It has found use in a multitude of widely ranging fields throughout the social, physical and life sciences. However, because of its ubiquity, PCA can be applied without

thought or consideration as to its appropriateness for a given problem. In these cases, two important questions can arise relating to how well the estimated PCA model fits the data. The first concerns the well known problem of selecting the number of principal components which is important for interpretibility and ensuring the model does not overfit. The second issue is the less-studied problem of identifying *influential* observations. These are points which have a large effect on the estimated model relative to other points and so care must be taken to detect the existence of such points to determine whether they should be included in the model.

If the low-dimensional structure in the data is assumed to be non-linear, fitting a single, low-dimensional subspace to the data using PCA may no longer be appropriate. We examine a special case of this problem whereby the data belong to several disjoint subsets, or clusters and each of the clusters lies on a different linear subspace. Here, a single PCA model is no longer suitable and instead, we must consider the problem of *subspace clustering*. This problem involves simultaneously recovering the clusters and estimating the low-dimensional representations within each cluster such that we perform locally linear dimensionality reduction. However, the same issues regarding model selection and influence apply in each cluster. We review the current state-of-the-art subspace clustering methods, which attempt to address these issues, and identify their limitations.

2.1 Penalised regression

In the supervised setting, we observe points, $\{\boldsymbol{x}_i\}_1^N \in \mathbb{R}^{1\times P}$ and paired response variables, $\{y_i\}_1^N$. We can represent the observed points in matrix form $\boldsymbol{X} \in \mathbb{R}^{N\times P}$, where the i^{th} row is $\boldsymbol{x}_i$ with a corresponding vector of responses $\boldsymbol{y} \in \mathbb{R}^{N\times 1}$. We model the relationship between $\boldsymbol{y}$ and $\boldsymbol{X}$ through a linear function $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \epsilon$ where $\boldsymbol{\beta} \in \mathbb{R}^{P\times 1}$ is a vector of regression coefficients. We assume that $\boldsymbol{\epsilon}$ is mean zero noise which is uncorrelated with the response. The aim of linear regression is to obtain an estimate of the coefficients, which we denote $\hat{\boldsymbol{\beta}}$, in such a way that the

residual sum of squares between $\boldsymbol{y}$ and $\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{\beta}}$ is minimised[1],

$$\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2. \tag{2.1}$$

The estimate of $\boldsymbol{\beta}$ which minimises Eq (2.1) is the ordinary least squares (OLS) solution. Provided $\boldsymbol{X}$ is of full rank, there is a unique analytic solution to this problem given by

$$\hat{\boldsymbol{\beta}} = \left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{y}. \tag{2.2}$$

However, when P is large $\boldsymbol{X}$ may be rank deficient due to highly correlated variables or a smaller number of observations than variables (the $N < P$ case). In these cases, $\boldsymbol{S} = \boldsymbol{X}^\top \boldsymbol{X}$ will not have a unique inverse and so the estimates of the least squares coefficients $\hat{\boldsymbol{\beta}}$ will not have a unique solution.

We can overcome this problem by restricting the ℓ_1 norm of the parameter so that the least squares cost function becomes

$$\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \gamma\|\boldsymbol{\beta}\|_1$$

where $\|\boldsymbol{\beta}\|_1 = \sum_{p=1}^{P} |\beta_p|$ is the ℓ_1 norm of $\boldsymbol{\beta}$. This is known as the Lasso penalty [88]. The Lasso has the property that for appropriate values of γ it forces certain coefficients to be exactly zero. This is an important property as it has the effect of performing variable selection.

Unlike OLS, there is no simple analytical solution to every Lasso problem. Therefore, determining the shrinkage effect that the Lasso penalty has on the coefficients is not so straightforward. In the special case of orthogonal predictors (i.e. $\boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{I}_P$)[2], we can write the Lasso penalised residual sum of squares as

$$\boldsymbol{\beta}^\top \boldsymbol{\beta} - 2\boldsymbol{y}^\top \boldsymbol{X}\boldsymbol{\beta} + \gamma\|\boldsymbol{\beta}\|_1.$$

[1]Throughout this work, $\|\cdot\|$ with no superscript denotes the ℓ_2 norm which for vectors is the Euclidean norm and for matrices is the spectral norm.

[2]Throughout this work, $\boldsymbol{I}_P$ denotes the $P \times P$ identity matrix.

By minimizing the penalised residual sum of squares, the solution for $\hat{\boldsymbol{\beta}}^{\text{lasso}}$ is given as a function of the ordinary least squares solution $\hat{\boldsymbol{\beta}}$ as follows

$$\hat{\beta}_p^{\text{lasso}} = \text{sign}(\hat{\beta}_p)(|\hat{\beta}_p| - \gamma)_+$$

where

$$(a)_+ \begin{cases} a & \text{if } a > 0 \\ 0 & \text{else} \end{cases}$$

is the soft-thresholding operation. Written in this form, we can see that the Lasso has the effect of truncating every coefficient by a constant, γ. Coefficients whose values change sign are assigned a value of zero.

An attractive feature of the Lasso is that it improves interpretability of the results. Since only a few variables are included in the model, these can be considered to be the important variables which can then be examined more closely. This is particularly useful in bioinformatics applications where selected variables may correspond to, for example, important genes which can then be the target of further experiments.

In general however, the Lasso is a convex optimisation problem and cannot be solved through simple soft-thresholding. Recently, efficient methods to obtain the Lasso coefficients have been developed, for example the LARS algorithm [28] and more recently, co-ordinate descent [31]. The advent of these methods have greatly increased the popularity of the Lasso in many application domains dealing with high-dimensional data analysis.

As mentioned, the number of variables included in the model depends on the parameter, γ. This dependence is highly non-linear and the number of variables included can have a large effect on both the predictive ability and the resulting interpretation of the model. Therefore, the value of γ must be chosen carefully which is an important model selection problem.

2.2 Principal component analysis

We now consider the *unsupervised* setting where we observe points, $\{\boldsymbol{x}_i\}_1^N \in \mathbb{R}^{1\times P}$ as before but do not observe a paired response variable. Here, it is often desirable to obtain a low-dimensional representation of the data which retains the important features and reduces the effect of noise. This is useful for visualisation and for identifying patterns and structure in the data which would otherwise be undetectable in P-dimensional feature space.

Principal component analysis (PCA) has emerged as a ubiquitous technique for unsupervised dimensionality reduction. PCA aims to represent each observation in terms of a small number, $R << P$, of uncorrelated latent factors. PCA can be motivated from two different viewpoints in order to find a low-dimensional representation of the data which is in some sense optimal. In each instance, the latent factors are a linear combination of the original variables; in other words, an orthogonal projection of the original variables onto a lower dimensional subspace.

Finding an optimal low-dimensional representation of the data using PCA can be motivated in several ways depending on which properties of the data we consider important. In the first instance, we could consider the variance to be an important quantity such that we wish to find the projection of each $\boldsymbol{x}_i$ onto a subspace of dimension R whose variance is maximised. To start with, we represent the sample mean of the data as $\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^N \boldsymbol{x}_i$ and the covariance matrix as $\boldsymbol{S} = \sum_{i=1}^N (\boldsymbol{x}_i - \boldsymbol{\mu})^\top(\boldsymbol{x}_i - \boldsymbol{\mu}) = \boldsymbol{X}^\top\boldsymbol{X}$ where $\boldsymbol{X} \in \mathbb{R}^{N\times P}$ is a matrix whose i^{th} row is given by $\boldsymbol{x}_i - \boldsymbol{\mu}$ and we have omitted a scaling factor of $\frac{1}{N}$ for notational convenience. We first consider a projection of each column of the centered data matrix $\boldsymbol{X}$ onto a one-dimensional subspace ($R = 1$) which we represent as a vector of unit length, $\boldsymbol{v} \in \mathbb{R}^{P\times 1}$. We can obtain a projection which maximises the variance by solving the following optimisation problem

$$\begin{aligned} &\max_{\boldsymbol{v}} \text{Var}(\boldsymbol{X}\boldsymbol{v})^2, \\ &\text{subject to } \|\boldsymbol{v}\| = 1 \end{aligned} \tag{2.3}$$

which is equivalent to

$$\max_{\boldsymbol{v}} \boldsymbol{v}^\top \boldsymbol{S}\boldsymbol{v} \quad \text{subject to } \|\boldsymbol{v}\| = 1. \tag{2.4}$$

The constraint in (2.4) is necessary to prevent $\|\boldsymbol{v}\| \to \infty$.

We can also approach the problem of identifying a low-dimensional projection of the data by considering how much information is lost. We define the *reconstruction error* of the low-dimensional representation to be the difference between $\boldsymbol{X}$ and the projection of the low-dimensional representation space back into P dimensions. We can obtain an optimal low-dimensional projection of $\boldsymbol{X}$ by ensuring the resulting reconstruction error is minimised. For a one dimensional projection, this problem can be expressed as the least squares objective function

$$\min_{\boldsymbol{v}} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{x}_i \boldsymbol{v}\boldsymbol{v}^\top\|^2, \quad \text{subject to } \|\boldsymbol{v}\| = 1. \tag{2.5}$$

The solution to both (2.4) and (2.5) is obtained by computing the singular value decomposition (SVD) of $\boldsymbol{X}$ (see for example, [36]), given by

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}^\top, \tag{2.6}$$

where $\boldsymbol{U} = [\boldsymbol{u}^{(1)}, ..., \boldsymbol{u}^{(N)}] \in \mathbb{R}^{N\times N}$ and $\boldsymbol{V} = [\boldsymbol{v}^{(1)}, ..., \boldsymbol{v}^{(P)}] \in \mathbb{R}^{P\times P}$ are orthonormal matrices whose columns are the left and right singular vectors of $\boldsymbol{X}$, respectively. The columns of $\boldsymbol{V}$ are also the eigenvectors of $\boldsymbol{S}$. $\boldsymbol{\Lambda} = \text{diag}(\lambda^{(1)}, \ldots, \lambda^{(N)}) \in \mathbb{R}^{N\times P}$ is a diagonal matrix whose entries are the singular values of $\boldsymbol{X}$, and the square root eigenvalues of $\boldsymbol{S}$, in descending order.

The solution to (2.4) and (2.5), therefore is $\boldsymbol{v} = \boldsymbol{v}^{(1)}$. The corresponding principal component is $\boldsymbol{u}^{(1)} = \frac{1}{\lambda^{(1)}}\boldsymbol{X}\boldsymbol{v}^{(1)}$, where ${\lambda^{(1)}}^2$ quantifies the variance explained by the first principal component. Subsequent principal components are given by

subsequent columns of $\boldsymbol{U}$. The projection of each $\boldsymbol{x}_i$ onto subsequent loading vectors, given by the columns of $\boldsymbol{V}$, explains a decreasing amount of the total variance, given by $\text{Tr}(\boldsymbol{\Lambda}^2)$.

2.3 Sparse PCA

Recently, several similar methods have been proposed which take a penalised regression approach to estimating *sparse* PCA loading vectors [102, 110, 80, 81]. As with penalised regression, the underlying assumption is that in high dimensions there are many noisy or irrelevant variables.

Sparse PCA (S-PCA) [80] aims to minimise the reconstruction error between the data matrix, $\boldsymbol{X}$ and the reconstruction of the principal components, $\boldsymbol{u}$ using sparse loading vectors, $\boldsymbol{v}$ by applying the Lasso penalty to the PCA objective function (2.5). Initially, for $R = 1$, sparse PCA imposes a penalty on the ℓ_1 norm of the PCA loadings, $\boldsymbol{v}$. This results in the following optimization problem

$$\begin{aligned} &\min_{\boldsymbol{u},\boldsymbol{v}} \|\boldsymbol{X} - \boldsymbol{u}\boldsymbol{v}^\top\|_F^2 + \gamma\|\boldsymbol{v}\|_1 \\ &\text{subject to } \|\boldsymbol{u}\| = 1. \end{aligned} \tag{2.7}$$

where $\|\boldsymbol{A}\|_F^2 = \text{Tr}(\boldsymbol{A}^\top\boldsymbol{A})$ denotes the squared Frobenius norm. This problem can be solved by first obtaining $\boldsymbol{u} = \boldsymbol{u}^{(1)}$ and $\boldsymbol{v} = \sigma^{(1)}\boldsymbol{v}^{(1)}$ from the SVD of $\boldsymbol{X}$ and then applying the following iterative soft thresholding procedure to the elements of $\boldsymbol{v}$:

$$\boldsymbol{v} = \text{sgn}\left(\boldsymbol{X}^\top\boldsymbol{u}\right)\left(\left|\boldsymbol{X}^\top\boldsymbol{u}\right| - \gamma\right)_+ \tag{2.8}$$

$$\boldsymbol{u} = \frac{\boldsymbol{X}\boldsymbol{v}}{\|\boldsymbol{X}\boldsymbol{v}\|} \tag{2.9}$$

Equations (2.8) and (2.9) are applied iteratively until the change in $\boldsymbol{v}$ between iterations falls below some threshold. Subsequent sparse loadings can be found by deflating the data matrix to obtain $\boldsymbol{X} = \boldsymbol{X} - \boldsymbol{u}\boldsymbol{v}^\top$ and repeating the above steps. The complexity of this procedure to extract R principal components is $O(NPR)$,

therefore the regularised SVD method provides an efficient way of obtaining sparse loading vectors for PCA. The ℓ_1 penalty term can be replaced by one of a number of other penalties by replacing the soft-thresholding step in Eq. (2.9) by the co-ordinate descent procedure of [31]. Several other sparse PCA methods have similar problem formulations and result in almost identical iterative algorithms [102].

Similar to the Lasso, S-PCA aids in the interpretability of the resulting model by identifying which variables contribute to each latent factor and therefore which variables are important for explaining the variation in the data.

2.4 Model selection and detecting influential observations

When modelling high dimensional data using PCA, the fundamental assumption we make is that the data can be well approximated by a lower-dimensional linear subspace. The validity of this assumption relies on two important considerations. The first consideration is how many components to retain which is the well known problem of model selection in PCA. The number of components has several important interpretations. Selecting the correct number of dimensions is integral to correctly interpreting results obtained through PCA. This question of model selection is important since it determines how well the estimated model generalises to unseen data from the same distribution. If too many dimensions are retained, the resulting PCA model may overfit and therefore obtain a poor reconstruction error on novel observations than if fewer components were retained.

The second consideration is the implicit assumption that all the observed data points are describing the same underlying phenomenon and therefore come from the same distribution. Of course we may observe a large degree of variability in the data which may be as a result of measurement noise. Furthermore, the data may naturally contain outliers, that is, points occuring far away from the mean. Such characteristics of the data are readily observed by examining summary statistics and simple visualisations of the data.

Of greater concern are "influential observations" [9]. Influential observations are points which have a large effect on the model parameters relative to other points.

In some cases, influential observations may arise as a case of measurement error. In other cases, the presence of influential observations might indicate that the data does not consist of a single, homogeneous population. This has important implications since the resulting model fit will be biased towards the influential points and will not explain the data well.

Although detecting influence and performing model selection have seemingly different initial motivations, it can be shown that they are actually very similar. In this section, we review approaches taken to identify influential observations in the context of OLS and PCA. We also show the connection with model selection for PCA.

2.4.1 Model selection

Learning the number of principal components, R is an important question in modelling high dimensional data. Learning the true value of R is important to ensure that all of the important structure in the data is captured whilst at the same time, ensuring that the dimensionality is low enough so as not to overfit. In some applications there are theoretical justifications for specifying a particular value of R but for most applications R is unknown.

In the case of noise-free data, the number of principal components can be determined exactly by evaluating $R = \text{rank}(\boldsymbol{X})$. However, when the data is noisy, $\boldsymbol{X}$ may be of full rank and so a more robust means of estimating R is required. The different formulations of the PCA problem allow for several different approaches for considering model selection in PCA. A class of methods relies on so-called "elbow" heuristics to estimate R. The general idea is to plot the value of the PCA objective function as a function of R and observe the value for which the change between values of r and $r + 1$ is greatest (the elbow point). Under the maximum variance interpretation of PCA, this is known as a scree plot and is achieved by plotting the cumulatative sum of eigenvalues $\lambda_1, \ldots, \lambda_R$. Alternatively, under the minimum reconstruction error interpretation, the equivalent heuristic involves identifying the point at which the change in reconstruction error between r and $r + 1$

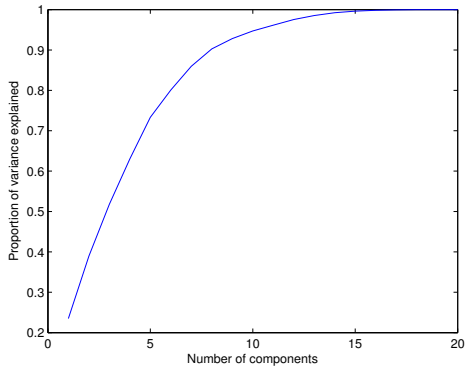
is the smallest. This can be assessed by approximating the second derivative of $\sum_{i=1}^{N} \|\boldsymbol{e}_i^{(r)}\|^2$ for $r = 1, \ldots, R$ with respect to r.

In general, both PCA objective functions vary monotonically since as R increases successively more variance is explained. Although when R is increased beyond its true value, the variance explained by subsequent latent factors is noise. There are several common heuristics used to guess the dimensionality of the data. Often, R is chosen so that some large proportion, say 95% of the variance of the data is retained. Figure 2.1 shows an example where $P = 20$ but the intrinsic dimensionality, $R = 10$. Standard Gaussian noise has been added. Figure 2.1a shows the proportion of variance explained, Figure 2.1b shows the scree plot and Figure 2.1c shows the residual error as a function of the number of components. It can be seen that each of the plots varies monotonically and their respective optimal values occur when the number of components is 20. Choosing R based on one of these common heuristic methods would result in a different value depending on which method is used. In this case, selecting R based on explaining 95% of the variance results in an overestimate and using the scree plot or the residual error plot results in an underestimate.

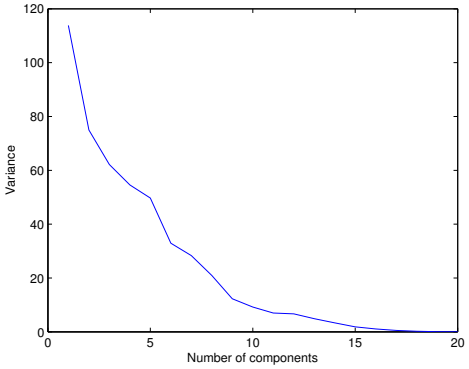
This problem has been considered in the context of OLS. The problem of overfitting can be overcome by partitioning the dataset into a separate training and test sets. Using the training set we fit the model and using the test set, we evaluate the model fit by computing the reconstruction error. In this way, the same samples are not used for both model fitting and evaluation. G-fold cross-validation (CV) is a generalization of this technique whereby the dataset is split into G equal-sized partitions and for $g = 1, \ldots, G$, $\boldsymbol{X} = [\boldsymbol{X}_1^\top, \ldots, \boldsymbol{X}_G^\top]^\top$. We fit the model using the partitions $\{1, \ldots, g-1, g+1, \ldots, G\}$ and evaluate the reconstruction error on the remaining parition, $\boldsymbol{X}_g$. The G-fold cross-validated error is then given by

$$J_{CV} = \frac{1}{G} \sum_{g=1}^{G} \|\boldsymbol{y}_{\{g\}} - \boldsymbol{X}_{\{g\}} \boldsymbol{\beta}_{\{-g\}}\|^2.$$

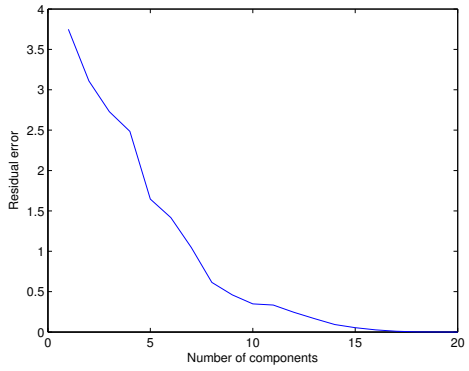
where $\boldsymbol{X}_{\{g\}}$ is a matrix whose rows are the indices in the partition $\{g\}$ and $\boldsymbol{\beta}_{\{-g\}}$ are



(a) Proportion of variance explained.



(b) Scree plot.



(c) Reconstruction error.

Figure 2.1: The PCA objective function as a function of the number of components. 2.1a shows the proportion of variance explained. 2.1b shows the scree plot. 2.1c shows the residual error. These quantities vary monotonically and the optimal value always occurs when the number of components is 20.

the OLS regression coefficients which have been estimated using all the data except for the observations corresponding to the indices in $\{g\}$. As such, the problem of overfitting is addressed by never using the same observations for both model fitting and model evaluation. The aim in using CV for model selection is to identify the model parameters which minimise the CV error. The choice of G is an open question. Values of $G = 5$ or 10 are commonly chosen for convenience with no real methodological justification. Important considerations include the number of samples and the dimensionality of the problem. If the number of samples is small, choosing G to be large will result in few cross-validation iterations meaning the resulting model may not generalise well. If either N or P is large, choosing G to be small will result in an increased computational burden.

Leave-one-out cross validation (LOOCV) occurs as a special case when $G = N$ [85]. LOOCV is performed by estimating the model N times leaving out each observation in turn and then evaluating the prediction error on the unused observation. The OLS LOOCV function is

$$J_{LOOCV} = \frac{1}{N}\sum_{i=1}^{N} \|y_i - \boldsymbol{x}_i\boldsymbol{\beta}_{-i}\|^2, \tag{2.10}$$

where $\boldsymbol{\beta}_{-i}$ denotes the estimate of the OLS regression coefficient when the i^{th} observation has been removed. This amounts to computing $\boldsymbol{\beta}_{-i} = (\boldsymbol{X}_{-i}^\top \boldsymbol{X}_{-i})^{-1}\boldsymbol{X}_{-i}^\top \boldsymbol{y}_{-i}$. LOOCV makes most efficient use of the available data since only a single observation is left out at a time. In the case of linear models, LOOCV is asymptotically equivalent to the Akaike information criterion (AIC) [86].

For OLS, it is well known that the LOOCV error can be computed analytically without the need to explicitly partition the data set. Naively computing this quantity for all $i = 1, \ldots, N$ would require the least squares model to be fit N times. Estimating $\boldsymbol{\beta}$ requires computing the inverse of the covariance matrix, $\boldsymbol{P} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1}$ which is computationally expensive. However, since each $\boldsymbol{\beta}_{-i}$ differs from $\boldsymbol{\beta}$ by only one observation, $\boldsymbol{x}_i$ we can easily compute $\boldsymbol{P}_{-i} = (\boldsymbol{X}_{-i}^\top \boldsymbol{X}_{-i})^{-1}$ from $\boldsymbol{P}$ using the matrix inversion lemma without the need to perform another $P \times P$ matrix

inversion [9]:

$$\begin{aligned}\left(\boldsymbol{X}_{-i}^{\top}\boldsymbol{X}_{-i}\right)^{-1} =& \left(\boldsymbol{X}^{\top}\boldsymbol{X} - \boldsymbol{x}_i^{\top}\boldsymbol{x}_i\right)^{-1} \\ =& \boldsymbol{P} + \frac{\boldsymbol{P}\boldsymbol{x}_i\boldsymbol{x}_i^{\top}\boldsymbol{P}}{1-h_i},\end{aligned} \tag{2.11}$$

where $h_i = \boldsymbol{x}_i^{\top}\boldsymbol{P}\boldsymbol{x}_i$. This allows the leave-one-out estimate, $\boldsymbol{\beta}_{-i}$ to be written as a function of $\hat{\beta}$ in the following way, without the need to explicitly remove any observations

$$\begin{aligned}\boldsymbol{\beta}_{-i} &= \left(\boldsymbol{X}_{-i}^{\top}\boldsymbol{X}_{-i}\right)^{-1}\left(\boldsymbol{X}^{\top}\boldsymbol{y} - \boldsymbol{x}_i^{\top}y_i\right) \\ &= \left(\boldsymbol{\beta} - \frac{(y_i - \boldsymbol{x}_i\boldsymbol{\beta})\boldsymbol{P}\boldsymbol{x}_i}{1-h_i}\right).\end{aligned} \tag{2.12}$$

The i^{th} LOOCV error, $e_{-i} = y_i - \boldsymbol{x}_i\boldsymbol{\beta}_{-i}$, is then given by

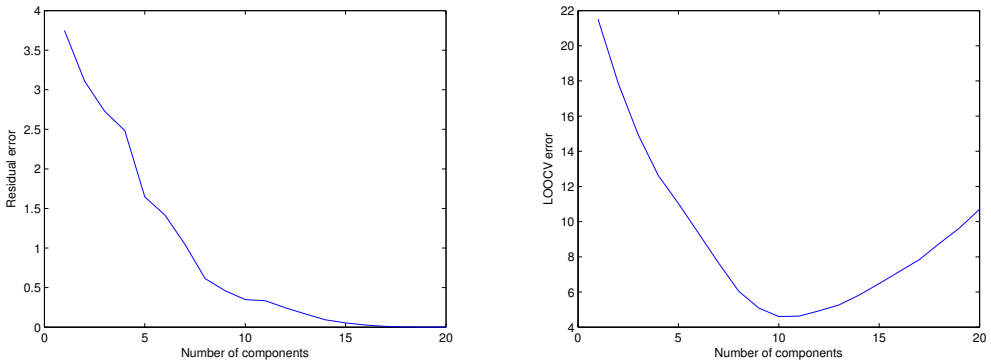
$$e_{-i} = \frac{e_i}{1-h_i}. \tag{2.13}$$

where $e_i = y_i - \boldsymbol{x}_i\boldsymbol{\beta}$ is the residual error. When expressed in this analytic form, the LOOCV for OLS is commonly known as the Predicted REsidual Sum of Squares (PRESS).

In the context of PCA, cross validation, especially LOOCV, is commonly used to estimate the number of components to retain. The LOOCV function for PCA is given by

$$J = \frac{1}{N}\sum_{i=1}^{N}\|\boldsymbol{x}_i - \boldsymbol{x}_i\boldsymbol{V}_{-i}\boldsymbol{V}_{-i}^{\top}\|^2, \tag{2.14}$$

where $\boldsymbol{V}_{-i}$ denotes the estimate of the PCA loadings when the i^{th} observation has been removed. This amounts to computing the eigendecomposition of $\boldsymbol{X}_{-i}^{\top}\boldsymbol{X}_{-i} = \boldsymbol{X}^{\top}\boldsymbol{X} - \boldsymbol{x}_i^{\top}\boldsymbol{x}_i$. Figure 2.2 demonstrates the result of performing LOOCV as a function of the number of selected components on the example in figure 2.1. Figure 2.2a again shows the residual error which decreases monotonically as the number of components increases. Figure 2.2b shows the LOOCV error. It is clear that although the overall magnitude of the LOOCV error is larger than that of the residual, it



(a) Reconstruction error.

(b) Leave-one-out error.

Figure 2.2: Model selection using cross-validation. 2.2a shows the residual error. 2.2b shows the LOOCV error. The residual error is monotonically decreasing as a function of the number of components wheras the LOOCV exhibits a global minimum when the correct number of components are added to the model.

exhibits a minimum when the number of PCA components is exactly 10.

Although PCA can be expressed as a least squares problem, the PCA parameters are estimated using a SVD. Therefore, an exact recursive formulation for the PCA LOOCV error in Eq. (2.14) is not possible. Naively computing the PCA LOOCV error requires N SVD computations which is extremely expensive when N or P are large, as is frequently the case when PCA is applied. It has been observed that in large sample sizes the LOOCV error is almost equivalent to $\frac{\lambda_R}{\sum_{r=R+1}^{P} \lambda_r}$ which requires the computation of only a single SVD, however this relationship breaks down when N is small [45]. An efficient method for the exact computation of the PRESS for PCA has been proposed [65] based on computing a rank-one downdate of the eigendecomposition obtained by PCA when a single observation is removed. Although exact and an order of magnitude more efficient than the naive approach to leave-one-out cross validation for PCA, this method has not received wide-spread attention or use in the literature.

2.4.2 Identifying influential observations

An outlying observation is typically defined as being an observation far from the mean. These can usually be identified easily. However, unlike outliers the effects of influential observations are not neccesarily apparent through simple visualisations of the data and so detecting influence is a more involved procedure. Exactly how to determine what the type and magnitude of "effect" constitutes an influential observation is an important and difficult question. For starters, the very definition of what it means for an observation to be influential is not well defined. A commonly cited definition is: [9]:

> *An influential observation is one which, either individually or together with several other observations, has a demonstrably larger impact on the calculated values of various estimates... than is the case for most other observations.*

This definition suggests that a measure of influence could determine which observations have a detrimental effect on the resulting model fit. However the vagueness of the definition suggests what exactly constitutes influence and how to measure it is an open question. Neither does this definition consider how points come to be influential in the first place which is an important issue when dealing with such observations.

Much of the existing literature concerning influential observations concentrates on identifying such points in the context of OLS. Since PCA can be considered as a least squares problem, several analogies can be drawn with PCA so it is beneficial to first examine the main approaches to identifying influence in OLS. A particular quantity of interest is the "hat matrix" [9],

$$\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^\top \boldsymbol{X})^{-1}\boldsymbol{X}^\top \in \mathbb{R}^{N\times N},$$

where $\boldsymbol{H}\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta}$, the estimated response. The term H_{ii} is known as the leverage of the i^{th} point, and determines the contribution of the i^{th} point in estimating its associated response. The partial leverage, H_{ij} is a related quantity which gives

the contribution of the j^{th} point for estimating the response associated to the i^{th} point. These quantities are used to detect influential observations which have a larger contribution to the estimated responses relative to other points. However, $\boldsymbol{H}$ does not take into account any information from $\boldsymbol{y}$ and so these leverage terms alone are not always sufficient to determine which observations are influential [9]. Alternatively, an obvious approach might be to examine the residual error given by

$$e_i = y_i - \hat{y}_i = y_i - \boldsymbol{x}_i\boldsymbol{\beta}. \tag{2.15}$$

We would perhaps expect that influential observations exhibit a large residual compared to other observations. However, it has been observed that observations which exert a large leverage on the regression may obtain relatively smaller residual errors compared to points which exert a smaller leverage [73]. This is due to the tendency of the OLS model to overfit to outlying or influential observations. This makes the use of the residual error alone unsuitable for detecting influential observations [26].

A different approach for identifying influential observations is by observing the effect on the estimated coefficient of removing each point in turn. This is achieved by removing a single observation, i from the model and computing the regression coefficients, $\boldsymbol{\beta}_{-i}$. We can then observe the relative effect of removing each observation in turn by computing the following statistic

$$\textit{DFBETA}_i = \boldsymbol{\beta} - \boldsymbol{\beta}_{-i}, \tag{2.16}$$

where $\boldsymbol{\beta}_{-i}$ is computed in the same way as in Eq. (2.12)

$$\boldsymbol{\beta}_{-i} = \left(\boldsymbol{\beta} - \frac{(y_i - \boldsymbol{x}_i\boldsymbol{\beta})\boldsymbol{P}\boldsymbol{x}_i}{1 - h_i}\right).$$

From this expression we can see that $\textit{DFBETA}_i$ has the closed form expression

$$\textit{DFBETA}_i = \boldsymbol{\beta} - \boldsymbol{\beta}_{-i} = \frac{(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{x}_i e_i}{1 - h_i}. \tag{2.17}$$

which provides an expression which combines the i^{th} residual, e_i with the leverage of the i^{th} point, h_i and $\frac{e_i}{1-h_i}$ is exactly the form of the i^{th} predicted residual used to compute the PRESS. This forms the basis of the *DFFIT* statistic which quantifies the difference between the estimated response, $\hat{y}_i$ and its leave-one-out estimate, $\hat{y}_{i,-i}$

$$\textit{DFFIT}_i = \hat{y}_i - \hat{y}_{i,-i} = \boldsymbol{x}_i(\boldsymbol{\beta} - \boldsymbol{\beta}_{-i}) = \frac{h_i e_i}{1-h_i}. \tag{2.18}$$

Several other widely used methods of identifying influential observations are also related to the predicted residual. For example, the *DFFITS* statistic has the form

$$d_i = \frac{\boldsymbol{e}_i}{\sigma_{-i}\sqrt{1-h_i}} \left[\frac{h_i}{1-h_i}\right]^{1/2},$$

where σ_{-i} is the standard deviation of the residual estimated without the i^{th} observation [9]. Similarly, *Cook's distance* can be written as [64]

$$c_i = \frac{\boldsymbol{e}_i^2}{(1-h_i)^2} \left[\frac{h_i}{P\sigma^2}\right].$$

It can be seen that both of these methods rely on scaling the i^{th} residual by a function of $1 - h_i$ which implicitly involves estimating the leave-one-out error of the regression model. It is clear then, that evaluating the prediction error is an important step in constructing a method for identifying influential observations in the context of OLS. The analytic form of this error, shown in the previous section allows these measures of influence to be computed efficiently.

It is clear then that despite being motivated in different ways, many methods of detecting influential observations are closely related to the leave-one-out prediction error. It must be noted that these methods are all designed to identify a single outlying or influential observation. When multiple influential observations are present, their effects can be more complicated. Two common effects are masking, where the effects of the influential observations are hidden and swamping where normal observations are mistaken for influential ones [75]. To solve this problem, generalised

versions of these deletion based methods have been proposed based on observing the effect of deleting groups of observations. Although this adds the combinatorial problem of choosing the number of influential observations.

In the context of PCA, single sample deletion-based methods, analogous to those for OLS, have been proposed for the purpose of detecting influential observations. Whereas in OLS, the quantity of interest is the leave-one-out estimate of the regression parameter $\boldsymbol{\beta}_{-i}$, for PCA there are two natural quantities to examine: the downdated eigenvalues, $\{\lambda_{-i}^{(r)}\}_1^R$ and eigenvectors, $\{\boldsymbol{v}_{-i}^{(r)}\}_1^R$ for all $i = 1, \ldots, N$ [65]. The first quantity is the sum of the difference between the original eigenvalues, $\boldsymbol{\Lambda}$ and their leave-one-out estimates $\boldsymbol{\Lambda}_{-i}$. The total effect of this eigenvalue downdate is computed as

$$\rho_i = \sum_{r=1}^{R} \lambda^{(r)} - \lambda_{-i}^{(r)}.$$

However, they note that this quantity is equivalent to evaluating

$$\rho_i = \frac{N}{N-1} \|\boldsymbol{x}_i - \boldsymbol{\mu}\|^2,$$

that is, the Euclidean distance from the deleted observation to the mean of the data. Therefore, examining the leave-one-out estimates of eigenvalues is only likely to identify influential points if they are also outliers. Instead, they suggest a better measure of influence to be the angle between the original eigenvector and its downdated version for each observation,

$$\cos(\theta_{\boldsymbol{v},\boldsymbol{v}_{-i}}) = \boldsymbol{v}^\top \boldsymbol{v}_{-i}.$$

This measure considers the stability of the estimates of the eigenvector when an observation has been removed. If the i^{th} observation is influential, the angle between the original eigenvector and the downdated eigenvector will be large so $\cos(\theta_{\boldsymbol{v},\boldsymbol{v}_{-i}}) \to 1$. Aside from these obvious measures which are analogous to methods used in OLS, the problem of identifying influence in PCA has not been widely developed in the literature.

However, once the influential observations have been identified, the question of how to deal with such observations is encountered. Dealing with influence in the appropriate manner requires some knowledge of why a particular observation might be influential. Often the assumption is that the influential observations do not belong to the model and so the prescribed treatment is to remove them altogether. In certain cases, influential observations may occur as a result of noise, measurement error or an extreme case of the measured phenomenon. In these cases, it may not be appropriate to remove the influential observations entirely. Furthermore, even after applying these diagnostic tests, it is not always clear which observations are influential. Many of these methods have heuristic thresholds beyond which an observation is deemed to be influential [17] although ultimately the process is very data dependent and care must be taken in any subsequent steps.

Another important question is what action should be taken when influential observations are detected. Often, the prescribed treatment is to remove such observations however, this is again data and application specific. An alternative to identifying outlying observations is to obtain an estimate by incorporating all observations using a robust model. Methods for performing robust PCA have been proposed which attempt to estimate a PCA model in the presence of noise or outliers. The goal of these methods is to implicitly identify the outlying observations and in doing so downgrade their relative importance in the resulting PCA model.

Amongst these, ROBPCA [43] combines several heuristic approaches in order to estimate robust principal components. First, ROBPCA attempts to find L least outlying observations with which to estimate an initial, robust covariance matrix. The number, L is chosen so that $N - L$ is greater than the number of outlying points. These L points are then used to construct the covariance matrix, $\boldsymbol{S}_L$ whose $M < N - 1$ largest eigenvectors are found as $\boldsymbol{V}_L = [\boldsymbol{v}_L^{(1)}, \ldots, \boldsymbol{v}_L^{(M)}]$ where M is a parameter which must be specified. The original data are then projected onto this subspace to obtain $\boldsymbol{U}_L = \boldsymbol{X}\boldsymbol{V}_L \in \mathbb{R}^{N \times M}$. The robust PCA parameters are then obtained by computing the SVD of $\boldsymbol{U}_L$.

Since the outlying points are discarded from the estimation of the initial $M < N$-dimensional subspace, when they are projected onto this subspace their contribu-

tion to the subsequent estimates of the ROBPCA loadings are reduced. As a result the estimates of the ROBPCA loadings are not biased towards the outlying points.

2.5 Subspace clustering

Up to this point we have assumed that all of the points we observe lie on the same, low-dimensional subspace. In the presence of outliers and influential observations, we have considered methods to detect these points and either remove them or, in the case of robust estimation, incorporate them into the estimate of the subspace. However, in some cases we may observe numerous outlying or influential points which belong to heterogeneous sub-populations in the data where each sub-population lies on a different low-dimensional subspace. In the following section we consider the problem of identifying these sub-populations and estimating their associated subspaces.

2.5.1 Clustering in high dimensions

As before, we observe N data points, $\{\boldsymbol{x}_i\}_1^N$. However, now we assume the observations belong to K non-overlapping clusters. The set $\mathcal{C}_k$ contains the indices of the N_k points belonging to the k^{th} cluster. In this section, we consider the task of recovering the true cluster assignments in an unsupervised fashion. The popular K-means algorithm is one of the most widely used unsupervised clustering methods [27]. K-means attempts to recover K tightly grouped clusters of points in the data where the squared Euclidean distances between points inside each cluster is smaller than the distance between clusters. Cluster assignments, $\{\mathcal{C}_k\}_1^K$ are found which minimise

$$C = \sum_{k=1}^{K} \sum_{i \in \mathcal{C}_k} \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|^2. \tag{2.19}$$

where $\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i \in C_k} \boldsymbol{x}_i$ is the sample mean of the observations assigned to cluster $\mathcal{C}_k$. K-means is an iterative algorithm which alternates between assigning points to clusters such that C is minimised and re-computing the sample mean in each cluster.

When the dimensionality of the data is large there are two well known problems related to the *curse of dimensionality* which affect the ability of methods such as K-means to recover the clusters accurately[72].

- As the number of dimensions increases, the Euclidean distance between all pairs of points tends towards uniformity.
- Many of the P dimensions in the data may be irrelevant or noisy. Therefore, computing distances between points using all P dimensions can mask the true clustering.

A solution to both problems is to first identify the relevant dimensions in the data which contain information about the difference between clusters and use only those to perform cluster analysis. These $R < P$ relevant dimensions can be thought of as low-dimensional *subspaces* of the original P dimensional space. The subspace clustering problem can be generally stated as simultaneously identifying a small number of dimensions in the data for which some measure of clustering accuracy is minimised.

Several different approaches to identifying relevant dimensions for clustering in high-dimensional data exist which broadly fall under the category of subspace clustering. One type of subspace clustering is based on feature selection [72]. That is, the similarity between points is measured on only a subset of the available variables. These methods employ a heuristic approach to identifying which subsets contain the important discriminative information between clusters. Such feature-selection based methods can be separated into two categories, **Bottom-up search** and **top-down search** methods.

Bottom-up search methods initially examine each dimension individually and identify the dimensions for which the "densest" groups of observations occur. That is, subsets of variables which contain the largest number of observations. Dimensions for which similarly dense groups of observations are found are grouped together as clusters. Bottom-up search methods make uses of the so-called *downward closure* property of density. This property states that if a dense cluster exists in R specific dimensions, it also exists in all combinations of $R-1$ of those dimensions.

The CLIQUE algorithm [2] is an example of such a method which defines a dense group as a subset of R dimensions that contains more than a specific proportion of the total number of points. CLIQUE starts from the densest single dimension and incrementally adds dimensions until all points have been assigned to a cluster. Alternatively, the ENCLUS algorithm [20] relates the density of a subset to its entropy. A dense subset will have a lower entropy and so dimensions are added to the subset such that the entropy of each subset remains below a threshold. Conversely, **top-down search** methods attempt to initially recover clusters in full P dimensional space. Dimensions are then downweighted or removed entirely until the densest possible clusters are obtained. The PROCLUS algorithm [1] proposes an extension to the K-means algorithm which assigns a weight to each variable in each cluster, whereas the clustering on a subset of attributes (COSA) algorithm [32] and more recently the sparse K-means algorithm [103] also assign a weight for each variable.

The approach to subspace clustering employed by these bottom-up and top-down search methods could more correctly be thought of as "subset clustering" since the goal is to identify a subset of the available variables in which an optimal clustering exists. However, these methods do not make any assumptions about the types of subspaces which are represented by the selected variables. As such these methods are sensitive to several tuning parameters and require expensive searches through all possible subspaces. In order to accommodate these additional parameters, complicated heuristic optimisation strategies are required.

If we make an assumption about the type of subspaces on which the clusters can lie, we can develop a more principled approach to performing subspace clustering such that heuristic search strategies are not required. We therefore concentrate on a different approach to subspace clustering which aims to identify low dimensional projections of the data which lie on *linear subspaces*. We refer to this approach as **linear subspace clustering** [94].

The use of low-dimensional projections to identify clusters can be motivated by examining the similarities between the K-means algorithm and PCA. It has long been observed that in high dimensional clustering problems, applying PCA dimension reduction before performing K-means cluster analysis improves the accuracy

of the recovered clusters. More recently the connection between K-means and PCA has been established formally. It has been shown that the eigenvectors obtained by PCA can be viewed as a continuous relaxation of the cluster assignments [25]. The K-means within cluster sum of squares objective function in Eq. (2.19) can be expressed as

$$C = \sum_{i=1}^{N} \boldsymbol{x}_i \boldsymbol{x}_i^\top - \sum_{k=1}^{K} \frac{1}{N_k} \sum_{i,j \in \mathcal{C}_k} \boldsymbol{x}_i \boldsymbol{x}_j^\top.$$

The terms within the sum over k can be replaced by a matrix of indicator variables, $\boldsymbol{U} \in \mathbb{R}^{N \times K}$ where the k^{th} column of $\boldsymbol{U}$ is

$$\boldsymbol{u}_k = (0, \ldots, 0, \overbrace{1, \ldots, 1}^{N_k}, 0, \ldots, 0)^\top / N_k^{1/2}.$$

Each row of $\boldsymbol{V}$ has a single non-zero entry. Now J can be written as

$$\begin{aligned} C &= \text{Tr}(\boldsymbol{X}\boldsymbol{X}^\top) - \sum_{k=1}^{K} \boldsymbol{u}_k{}^\top \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{u}_k \\ &= \text{Tr}(\boldsymbol{X}\boldsymbol{X}^\top) - \text{Tr}(\boldsymbol{U}^\top \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{U}). \end{aligned}$$

Since $\text{Tr}(\boldsymbol{X}\boldsymbol{X}^\top)$ is a constant, C is minimised when $\text{Tr}(\boldsymbol{U}^\top \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{U})$ is maximised.

If the elements of $\boldsymbol{U}$ are allowed to be continuous, $\max_{\boldsymbol{U}} \text{Tr}(\boldsymbol{U}^\top \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{U})$ is an eigenproblem where the columns of $\boldsymbol{U}$ are the eigenvectors of $\boldsymbol{X}\boldsymbol{X}^\top$ [25]. It can be seen in the case of $K = 2$ that the columns of $\boldsymbol{U}$ contain redundancies. That is, $\boldsymbol{u}^{(1)}$ defines the cluster assignment for both clusters. Therefore, only the first $K - 1$ eigenvectors must be recovered. However, since the columns of $\boldsymbol{u}$ are no longer binary indicator vectors, this procedure does not minimise the original function, C and recover the cluster assignments. However the actual cluster assignments can be obtained by performing K-means on the columns of $\boldsymbol{U}$.

Furthermore, bounds on the minimum distance between clusters required in order for the PCA pre-processing step to separate the clusters in the projected space

have been established [93] . It has been shown that if the data come from a mixture of K isotropic Gaussians with minimum mixing proportion w_{min}, the subspace spanned by the top $K-1$ principal components is the subspace spanned by the means of the mixture. Projecting the data onto this lower dimensional space then has the effect of preserving the distance between means but shrinking the distances between points in a cluster by a factor of $\sqrt{\frac{K-1}{P}}$. This result requires the minimum separation between the means of any two clusters, $\mathcal{C}_1$ and $\mathcal{C}_2$ to be $\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\| \geq c\max\{\sigma_1, \sigma_2\}(K\log N)^{\frac{1}{4}}$ where c is a constant and σ_1 is the variance of cluster $\mathcal{C}_1$. Although this bound only holds for mixtures of isotropic Gaussians, using PCA as a preprocessing step for K-means has shown to be effective in a number of real world applications such as clustering of gene expression data and document clustering [25].

The use of PCA as a pre-processing method for clustering assumes that the data all lie on the same R-dimensional subspace. As with standard K-means clustering, these techniques assume that the clusters are well separated in terms of Euclidean distance in that subspace. However, in many applications clusters may lie in different low-dimensional subspaces, each with its own dimension. In these situations the data no longer satisfy the minimum separation requirement of [93] and so global PCA is unsuitable for recovering the correct clusters.

2.5.2 Linear subspace clustering

As before, we assume to have observed N points, $\{\boldsymbol{x}_i\}_1^N$, where each $\boldsymbol{x}_i \in \mathbb{R}^{1\times P}$ and the dimension P is usually very large. Again, each point is assumed to belong to one of K non-overlapping clusters, $\{\mathcal{C}_k\}_1^K$. However, we further assume that the points in the k^{th} cluster lie in a R_k-dimensional subspace, $\mathcal{S}_k$ where $R_k << P$. Each subspace $\mathcal{S}_k$ is defined in the following way

$$\mathcal{S}_k = \{\boldsymbol{x}_i : \boldsymbol{x}_i = \boldsymbol{\mu}_k + \boldsymbol{u}_{k,i}\boldsymbol{V}_k{}^\top\} \tag{2.20}$$

with $i \in \mathcal{C}_k$ and $k = 1, \ldots, K$, where $\boldsymbol{V}_k \in \mathbb{R}^{P\times R_k}$ is a basis for $\mathcal{S}_k$ whose columns are restricted to be mutually orthonormal. The point $\boldsymbol{u}_{k,i} \in \mathbb{R}^{1\times R_k}$ is the low di-

mensional representation of $\boldsymbol{x}_i$ and $\boldsymbol{\mu}_k \in \mathbb{R}^{1\times P}$ is an arbitrary point in $\mathcal{S}_k$, typically chosen to be $\mathbf{0}$.

When only one cluster exists, i.e. $K = 1$, a subspace of this form can be estimated by fitting a PCA model, which provides the best low-rank linear approximation of the original data. In this case $\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i$ and the columns of $\boldsymbol{V}_k$ are the first R right singular vectors of $\boldsymbol{X}$. The low dimensional representation of $\boldsymbol{x}_i$ is then $\boldsymbol{u}_{k,i} = \boldsymbol{x}_i\boldsymbol{V}_k$. When the data partition into more than one cluster, i.e. $K > 1$, but the cluster assignments are known, a subspace of form (2.20) can be estimated by fitting a PCA model independently in each cluster. However, since the cluster assignments are generally unknown, the problem of subspace clustering consists in the simultaneous identification of the number of clusters, K, the subspaces $\{\mathcal{S}_i\}_1^K$ and the cluster assignments $\{\mathcal{C}_i\}_1^K$.

There are several fundamental difficulties associated with this problem:

1. The ability to estimate the true subspaces is dependent on recovering the true clusters and vice-versa.

2. Subspaces may not be independent which implies they may intersect at several locations which causes difficulties when attempting to assign points to subspaces at these intersections. This causes standard clustering techniques such as K-means to fail.

3. The subspace parameters and the cluster assignments are depended on both the number of clusters, K and the dimensionality of their respective subspaces $\{R_k\}_1^K$. These quantities are difficult to estimate from the data and are often taken as known. However, in realistic scenarios both of these quantities could be unknown.

A variety of approaches have been proposed to solve problems 1 and 2 which we review in the remainder of this chapter. However, problem 3 is an issue of model selection which is a more difficult issue. In the noise-free case, assuming the correct clusters have been recovered, the rank of $\boldsymbol{X}$ within each subspace gives the local dimensionality of that subspace, R_k. However, in the presence of noise, two

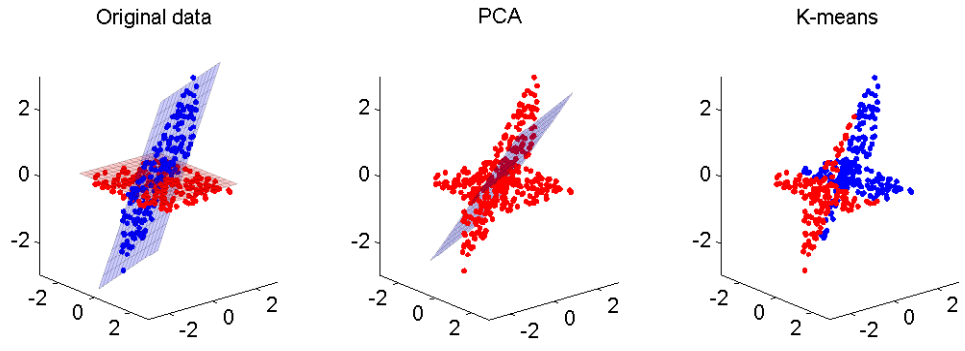


Figure 2.3: An example of points belonging to two clusters which lie in two different planes. The middle plot shows the result of fitting a single subspace using a PCA model which provides a poor fit to the data. The right-hand plot shows the result of clustering using K-Means which fails to accurately recover the clusters.

issues complicate this task. Firstly, recovering the correct clustering and therefore estimating the true subspace parameters becomes harder. Secondly, the noisy data may be of full rank. This requires the estimation of these parameters to be robust in the presence of noise.

Figure 2.3 provides an example of points in three dimensions which belong to two clusters. Each cluster consists of points which are distributed uniformly on a different two-dimensional subspace. The middle plot shows the result of fitting a single subspace to the data using a global PCA model. It is clear that the estimated subspace lies somewhere between the two true subspaces and as a result fits neither cluster well. The right-hand plot shows the result of using K-means to cluster the data. The points are assigned to clusters based on their geometric distance.

K-subspaces

Many methods for subspace clustering have been proposed which generalise the well-known K-means algorithm to K-*subspaces* [13, 91, 97]. K-subspaces aims to recover clusters such that the within cluster distances between observations, $\boldsymbol{x}_i$ and their R_k-dimensional reconstruction, $\boldsymbol{x}\boldsymbol{V}_k\boldsymbol{V}_k^\top$ is minimised. This can be expressed

in terms of the following objective function

$$C_{KSS} = \sum_{k=1}^{K} \sum_{i \in \mathcal{C}_k} \|\boldsymbol{x}_i - \boldsymbol{x}_i \boldsymbol{V}_k \boldsymbol{V}_k^\top\|^2, \tag{2.21}$$

where $\boldsymbol{V}_k \in \mathbb{R}^{P \times R_k}$ and subject to $\boldsymbol{V}_k{}^\top \boldsymbol{V}_k = I$. K-subspaces is an iterative algorithm which alternates between assigning points to clusters and estimating the subspace parameters. Within each cluster, minimizing the distance $\sum_{i \in \mathcal{C}_k} \|\boldsymbol{x}_i - \boldsymbol{x}_i \boldsymbol{V}_k \boldsymbol{V}_k{}^\top\|^2$ is equivalent to the minimum reconstruction error formulation of PCA. Therefore, for a given cluster assignment, $\{\mathcal{C}_k\}_1^K$, the K-subspaces are estimated by PCA.

Generalised PCA

Generalised PCA (GPCA) [58] takes an algebraic approach to the problem of partitioning the data and identifying the subspaces underlying those partitions. GPCA is based on the idea that points belonging to a union of K subspaces of dimension R can be represented by K polynomials of order R.

The motivating idea behind GPCA is that points lying in a subspace $\mathcal{S}_k$ can be parameterised by a vector, $\boldsymbol{b}_k \in \mathbb{R}^{P \times 1}$ which is orthogonal to $\mathcal{S}_k$ such that $\boldsymbol{x}_i \boldsymbol{b}_k = 0$ if $\boldsymbol{x}_i \in \mathcal{C}_k$. For a union of K subspaces, where each point $\boldsymbol{x}_i$ can only live in one of the subspaces, each point can be represented as

$$(\boldsymbol{x}_i \boldsymbol{b}_1)(\boldsymbol{x}_i \boldsymbol{b}_2), \ldots, (\boldsymbol{x}_i \boldsymbol{b}_K) = \prod_{k=1}^{K} (\boldsymbol{x}_i \boldsymbol{b}_k) = 0.$$

A point lying in one of the subspaces, $\mathcal{S}_k$ must satisfy one of the equations $(\boldsymbol{x}^\top \boldsymbol{b}_k = 0)$ so the subspace clustering problem becomes one of finding the roots of the polynomial.

However, in this representation, the polynomials are non-linear in the data points. This problem can be reformulated as an equivalent linear problem in terms of the monomials of $\boldsymbol{x}_i$ by means of a function, ν, a *Veronese map* which performs the mapping $\nu : \mathbb{R}^P \to \mathbb{R}^M$. The new dimensionality, M is the number of all monomi-

als of $\boldsymbol{x}_i$ of degree K which is given by the binomial coefficient

$$M = \begin{pmatrix} K+P-1 \\ K \end{pmatrix}.$$

By applying this function to each observation $\boldsymbol{x}_i$ we obtain a new vector $\nu(\boldsymbol{x}_i) \in \mathbb{R}^{1\times M}$,

$$\nu(\boldsymbol{x}_i) = \nu \begin{pmatrix} x_{i,1} \\ x_{i,2} \\ \vdots \\ x_{i,P} \end{pmatrix} = \begin{pmatrix} x_{i,1}^K \\ x_{i,1}^{K-1} x_{i,2} \\ \vdots \\ x_{i,P}^K \end{pmatrix}. \tag{2.22}$$

The problem can now be represented in the following linear expression

$$\prod_{k=1}^{K} (\boldsymbol{x}_i \boldsymbol{b}_k) = \nu(\boldsymbol{x}_i)\boldsymbol{c} = 0,$$

where $\boldsymbol{c} \in \mathbb{R}^{M\times 1}$ is the vector of coefficients. In matrix form this is

$$\begin{bmatrix} \nu(\boldsymbol{x}_1) \\ \vdots \\ \nu(\boldsymbol{x}_n) \end{bmatrix} \boldsymbol{c} = \nu(\boldsymbol{X})\boldsymbol{c} = 0.$$

The coefficients, $\boldsymbol{c}$ can then be estimated by identifying the null space of $\nu(\boldsymbol{X})$. This can be achieved by computing the right singular vectors corresponding to the zero singular values of the matrix $\nu(\boldsymbol{X})$. Finally, GPCA estimates the subspace basis vectors $\boldsymbol{b}_k$ for each of the subspaces, $\mathcal{S}_k$. Since these are defined as vectors orthogonal to their respective subspace, they are obtained by computing the derivative of the polynomial with respect to a point on that subspace.

Since GPCA remaps the points to M dimensions given by the binomial coefficient $M = \frac{(P+K-1)!}{K!(P-1)!}$, M increases exponentially with P. In practical applications, this causes the computational complexity of GPCA to become unmanageable when $P > 10$. In most real applications, global PCA is often used to reduce the dimensionality of the data to $R+1$ (where R is the maximum dimension of the subspaces).

This operation preserves the number and dimension of subspaces in the data however it requires knowledge of R. Since GPCA is an algebraic method it is sensitive to noise and outliers. Furthermore, the estimation of the subspace parameters depends on knowing the number of subspaces and their dimension in order to use the Veronese mapping function. An extension to GPCA, Robust GPCA, seeks to estimate the correct number and dimensions of subspace within the data [42] by using the Geometric information criterion (GAIC) [46].

Spectral clustering

GPCA represents each subspace using a polynomial with many parameters and as such it is not robust in the presence of noise. Rather than assigning points to subspaces based on exact algebraic techniques, a class of subspace clustering methods are based on spectral clustering. These methods consider how to measure the similarity between observations based on the subspaces they belong to. For each point, $\boldsymbol{x}_i$, these methods estimate a *local* representation of the subspace using the points nearby to $\boldsymbol{x}_i$, typically in the Euclidean sense. The distances between points are then represented as distances between local estimates of subspaces.

Spectral clustering is based on graph theoretic representation of the relationship between data points [68, 57]. Spectral clustering methods operate on an affinity matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ where the entry A_{ij} is a measure of distance between the points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. The relationship between points described in $\boldsymbol{A}$ can be considered to be a weighted graph where each A_{ij} represents a weighted edge between two vertices. The assignment of each observation to one of K clusters can be written in compact matrix form

$$\boldsymbol{L}^\top \boldsymbol{A} \boldsymbol{L}$$

where the k^{th} column of $\boldsymbol{L}$ is a binary vector which represents the assignment to the k^{th} cluster

$$\boldsymbol{l}_k = (0, \ldots, 0, \overbrace{1, \ldots, 1}^{N_k}, 0, \ldots, 0)^\top.$$

We can consider the problem of recovering clusters such that the sum of pairwise

weights between all points in each clusters are minimised. This can be expressed in the following way

$$\min_{\boldsymbol{L}} \boldsymbol{L}^\top \boldsymbol{A} \boldsymbol{L}. \tag{2.23}$$

In this representation, the problem of clustering points becomes one of partitioning the graph such that similar points are grouped together by cutting edges between vertices. Most proposed algorithms to perform these cuts are NP hard.

Spectral clustering allows the constraint on the binary structure of the columns of $\boldsymbol{L}$ to be relaxed. Under these conditions, solving Eq. (2.23) amounts to computing the smallest $K-1$ eigenvectors of $\boldsymbol{A}$. The columns of $\boldsymbol{L}$ are real-valued vectors and so a transformation is required since the cluster assignments are binary. This is typically achieved by performing K-means on $\boldsymbol{L}$.

The distance measure used to construct the affinity matrix defines the type of clusters which may be recovered. Therefore, choosing a suitable distance is crucial for subspace clustering. When the Euclidean distance is used we obtain the PCA, K-means method described in the previous section. However, when points lie in different subspaces, Euclidean distances between points are no longer a useful measure of similarity since points in different subspaces can lie close to each other. Similarly, far away points can lie on the same subspace.

Several different distance measures have been proposed which in some way quantify the notion that points may lie on different subspaces. These methods typically estimate a *local* representation of the subspace at each data point and compute some measure of distance between these local subspaces rather than the points themselves. In particular, GPCA has been used in conjunction with spectral clustering for situations where the data is noisy. In these cases, an affinity matrix is constructed by computing the distances between normal vectors at each point.

An important property of linear subspaces is that they are closed under linear transformations. This property implies that a linear combination of points belonging to a subspace, $\mathcal{S}_k$ will also belong to $\mathcal{S}_k$. This means that each point in $\mathcal{S}_k$ can also be represented as a linear combination of all other points in $\mathcal{S}_k$. This property has been exploited by several spectral subspace clustering methods to construct an

affinity matrix. Generally, these methods aim to represent a local subspace $\mathcal{S}_i$ for each point, $\boldsymbol{x}_i$, as a linear combination of its nearest neighbours.

The issue of choosing a neighbourhood is addressed by sparse subspace clustering (SSC) [29] which instead views estimating the individual subspaces as a sparse estimation problem. SSC relies the assumption that the K underlying subspaces are independent, that is each of the subspaces exist in different dimensions in the original space. SSC seeks to estimate a local subspace for each point as a linear combination of other "nearby" points. This can be viewed as estimating a sparse linear combination of every point. In the noiseless case, this amounts to solving the following ℓ_1 optimisation problem for a vector of weights, $\boldsymbol{w}_i \in \mathbb{R}^{1\times N-1}$ for each $i = 1, \ldots, N$

$$C_{SSC} = \sum_{i=1}^{N} \min \|\boldsymbol{w}_i\|_1 \tag{2.24}$$

$$\text{subject to} \quad \boldsymbol{x}_i = \boldsymbol{w}_i \boldsymbol{X}_{-i} \tag{2.25}$$

where $\boldsymbol{X}_{-i} = [\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_{i-1}^\top, \boldsymbol{x}_{i+1}^\top, \ldots, \boldsymbol{x}_N^\top]^\top$. Since the criterion to be minimised is the ℓ_1 norm of the weights, this ensures that points which do not lie in the same subspace as $\boldsymbol{x}_i$ will be assigned a zero weight. The optimal sparse representation is obtained when $\boldsymbol{x}_i$ is a combination of only other points in the same subspace.

In the presence of noise, the optimal sparse solution may contain points from other subspaces and so the SSC criterion is modified so that the ℓ_2 reconstruction accuracy of the subspaces is considered.

$$C_{SSC} = \sum_{i=1}^{N} \|\boldsymbol{w}_i\|_1 + \gamma \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{w}_i \boldsymbol{X}_{-i}\|^2$$

This criterion represents a trade-off between estimating sparse weights, $\boldsymbol{w}_i$ and minimizing the ℓ_2 reconstruction error between the original data points and the estimated subspaces. This trade-off is controlled by a regularization parameter, $\gamma \geq 0$. The sparse matrix of weights $\boldsymbol{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N]$, can be viewed as a graph where the non-sparse elements indicate observations which belong to the same subspace.

Clustering can then be performed using spectral methods. SSC only considers clusters of points which lie in linear or affine subspaces. Although the SSC approach is only provably correct when the subspaces are independent, results show that it still works well in the case of dependent subspaces.

Since the neighbourhood of points which contribute to each subspace is defined by the sparsity in $\boldsymbol{W}$, it is not necessary to define a neighbourhood of $g-$nearest neighbours. However, in the noisy case, the sparsity is affected by the tradeoff with the ℓ_2 reconstruction penalty which is controlled by the parameter, γ. Tuning this parameter is an open question and it is not clear exactly how sensitive the SSC clustering method is to different values of γ.

Spectral local best flats (SLBF) [108] estimates the cluster assignments based on minimising the distance between points and their local subspaces

$$d_{i,j} = \sqrt{\text{dist}(\boldsymbol{x}_i, \mathcal{S}_j)\text{dist}(\boldsymbol{x}_j, \mathcal{S}_i)}$$

where the function dist() is the Euclidean distance between a point and a subspace. The subspace $\mathcal{S}_i$ is estimated using PCA for the point $\boldsymbol{x}_i$ and its g-nearest neighbours. The method uses these distances to construct an affinity matrix, $\boldsymbol{A}$ with i, j^{th} entry

$$A_{i,j} = \exp(-d_{i,j}/2\sigma_j^2) + \exp(-d_{i,j}/2\sigma_i^2)$$

where σ_i^2 is a measure of how well $\mathcal{S}_j$ fits $\boldsymbol{x}_j$ and its g-nearest neighbours. Spectral clustering is performed on $\boldsymbol{A}$ to obtain the cluster assignments. SLBF has an overall computational complexity of $O((K+RP)N^2 + KPN)$ which arises from the need to compute a local PCA model for each observation.

Spectral curvature clustering (SCC) [19] assumes that the data belong to K clusters which exist in different affine subspaces each with the same dimensions, R. However, the SCC method does not attempt to estimate the subspace parameters and so does not compute a measure of reconstruction error. Instead SCC constructs an affinity matrix based on how likely different points are to exist in the same subspace. The basic idea behind SCC considers the volume of the simplex formed by selecting $R+2$ points at random. If the points are in the same subspace, this volume is zero.

SCC uses this idea as the basis for a measure of distance between groups on points.

SCC computes multi-way affinity measures between $R+1$ points, $\{\boldsymbol{x}_i\}_{i=1}^{R+2}$ selected at random by measuring the polar sine function between them. The polar sine function is related to the volume of those points however is also scale invariant. If $\text{Vol}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{R+1})$ is the volume of the simplex formed by the points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{R+1}$, the polar sine at each vertex, $\boldsymbol{x}_i$ is

$$\text{psin} = \frac{(R+1)!\text{Vol}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{R+1})}{\prod_{j \neq i} \|\boldsymbol{x}_j - \boldsymbol{x}_i\|}$$

which is zero when all the points lie in the same subspace. SCC then performs spectral clustering on the recovered distances. Since the SCC procedure constructs an affinity by computing the polar sine between points sampled randomly from the whole dataset, it is not necessary to choose a neighbourhood g as with other spectral based methods. Because of this, SCC is less sensitive to noise and intersections between subspaces. However, SCC requires $O((R+1)^2NP)$ operations per sample so a tradeoff must be made between clustering accuracy and computational time which can be achieved using heuristic sampling methods. Furthermore, SCC assumes all subspaces are of the same dimensionality which causes it to perform poorly when there are different subspaces of widely varying dimensionality.

Since spectral clustering based methods rely on globally maximizing an objective function using an eigendecomposition, they are non-iterative. As such they do not suffer from issues associated with local minima and sensitivity to starting values. Furthermore, since the clustering step relies only on an eigendecomposition, it can be computed efficiently. However, as mentioned the choice of distance is extremely important and as seen, can add significant computational cost depending on the measure used. Furthermore, the issue of model selection is still important.

2.6 Discussion

In this chapter we have examined issues relating to the modelling of high-dimensional data using PCA. Among these is the well known open question of model selection

in PCA. Many approaches to this are either based on heuristics or cross-validation, the latter of which is typically computationally expensive. We have also introduced the related problem of detecting influential observations. Similar to model selection, detecting influence using PCA is typically based on extensions of deletion based methods developed for OLS and has not been developed much beyond this. The issue of dealing with such observations, even within the context of OLS is still an open question although methods for robust estimation of PCA exist.

The field of subspace clustering has emerged in the last decade and the current state-of-the-art achieves excellent results in a number of real-world problems. However, the important questions of model selection and identifying influential observations persist and all of the methods we review have difficulties when the dimensionality of each subspace is different. Solving these issues are vital for the accurate estimation of subspaces, which as we have seen is directly related to the problem of recovering the true clustering assignments. As a result of these limitations, subspace clustering algorithms must employ computationally expensive search strategies in order to accurately estimate the underlying subspaces.

In the following chapters we propose an efficient framework for model selection and detecting influential observations in the context of PCA with the ultimate goal of developing an algorithm for subspace clustering based on the notion of influential observations which resolves the problem of model selection.

Chapter 3

Predictive methods for PCA

In this chapter we develop an efficient framework for model selection and detecting influential observations for PCA based on the concept of predictive reconstruction. We achieve this by noticing that since the PCA reconstruction error can be formulated as a least squares problem, we can derive a recursive expression for the leave-one-out estimates of the PCA loadings. This forms the basis of an efficient, approximate formulation of a PRESS statistic for PCA. We then propose a measure of predictive influence based on the effect each observation has on the PRESS under a given PCA model. We also propose a novel, robust solution to PCA by directly minimising the predictive influence. To conclude this chapter we present an example application in facial recognition.

3.1 The predictive reconstruction error

In the context of PCA, computing the leave-one-out error typically requires N model fits each using $N - 1$ data points. In the previous chapter, we reviewed a method for computing the exact PRESS for PCA efficiently [65]. However, this method still requires multiple SVD and renormalisation steps and so the resulting expression for the PRESS is not an analytic, differentiable function of the observations, $\boldsymbol{x}_i$.

In this section we propose an efficient approximation of the PRESS for PCA

which is computed at the expense of a single SVD. Importantly, unlike the method of [65], the PRESS function we propose has a closed-form recursive relationship between the PCA subspace parameters, $\boldsymbol{V}$ and their leave-one-out estimates, $\boldsymbol{V}_{-i}$. We will show in Section 3.3 that such a property is desirable as it allows us to directly minimise a measure of the *prediction* error rather than the residual error. This results in a PCA algorithm which is more robust to outliers and which generalises well to unseen observations.

3.1.1 PRESS for PCA

In this section we define an analytic, closed form of the PRESS for PCA. This formulation of the PRESS is based on the recursive formulation of the PRESS for OLS given by Eq. (2.13). However, since the PCA loadings are obtianed through the SVD, the derivation of this expression is not as straightforward and requires an approximation such that the loadings can be obtained using least squares estimation.

We initially consider the exact leave-one-out reconstruction error using a single principal component. This quantity has the following form

$$J = \frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{x}_i\boldsymbol{v}_{-i}\boldsymbol{v}_{-i}^{\top}\|^2,$$

where each $\boldsymbol{v}_{-i}$ is the first right singular vector of the SVD estimated using all but the i^{th} observation of $\boldsymbol{X}$. Therefore, computing J requires N SVD computations which is expensive when either N or P is large. We instead propose an approximate PRESS by introducing the following assumption

Assumption 3.1. *When the number of samples, N is large the estimate of the principal subspace, $\boldsymbol{v}$ does not change much if we estimate the SVD using N or $N-1$ observations. In other words we assume that $\boldsymbol{v}_{-i} \approx \boldsymbol{v}$ and therefore $\boldsymbol{x}_i\boldsymbol{v}_{-i} \approx \boldsymbol{x}_i\boldsymbol{v}$.*

We justify this assumption with intuitive and theoretical arguments in Section 3.1.2. This is known as the projection approximation subspace tracking (PAST) approximation [106]. Using this approximation and defining $d_i = \boldsymbol{x}_i\boldsymbol{v}$, we can

express the PRESS in terms of the i^{th} leave-one-out errors, $\boldsymbol{e}_{-i}$, as a quadratic function of $\boldsymbol{v}_{-i}$ in the following way

$$J = \frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{e}_{-i}\|^2 = \frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{x}_i - d_i \boldsymbol{v}_{-i}^{\top}\|^2. \tag{3.1}$$

Now $\boldsymbol{v}_{-i}$ can be obtained through least squares estimation in the following way

$$\boldsymbol{v}_{-i} = \left(\boldsymbol{d}_{-i}^{\top}\boldsymbol{d}_{-i}\right)^{-1}\left(\boldsymbol{X}_{-i}^{\top}\boldsymbol{d}_{-i}\right). \tag{3.2}$$

Since $\boldsymbol{v}_{-i}$ is an eigenvector of $\sum_{j\neq i}^{N} \boldsymbol{x}_j^{\top}\boldsymbol{x}_j$, it is constrained to be mutually orthonormal to any subsequently estimated eigenvectors. Enforcing such a constraint on $\boldsymbol{v}_{-i}$ would involve a renormalisation operation which breaks the linear relationship between $\boldsymbol{v}$ and $\boldsymbol{v}_{-i}$. Therefore, approximating each LOO estimate of $\boldsymbol{v}_{-i}$ using least squares relies on relaxing this constraint which induces a small deviation from orthonormality. It has been shown that the deviation from orthonormality in the PAST solution when initialised with a random unit vector, depends on the number of observations, N as $O(\frac{1}{N^2})$ [107]. Therefore, when $N \to \infty$, this error tends to zero. As the number of samples (and thus iterations) grows, the estimates converge to the true eigenvectors. We can now define a closed-form approximation for the PCA PRESS.

Definition 3.1. *The closed-form approximation for the PRESS for a PCA model is given by*

$$J^{(R)} \approx \frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{e}_{-i}^{(R)}\|^2. \tag{3.3}$$

where each leave-one-our error is given by

$$\boldsymbol{e}_{-i}^{(R)} = \sum_{r=1}^{R} \frac{\boldsymbol{e}_i^{(r)}}{1 - h_i^{(r)}} - (R-1)\boldsymbol{x}_i,$$

where $\boldsymbol{e}_i^{(r)} = \boldsymbol{x}_i - \boldsymbol{x}_i \boldsymbol{v}^{(r)} {\boldsymbol{v}^{(r)}}^{\top}$ *and* $h_i^{(r)} = \frac{(\boldsymbol{x}_i \boldsymbol{v}^{(r)})^2}{{\boldsymbol{v}^{(r)}}^{\top}\boldsymbol{X}^{\top}\boldsymbol{X}\boldsymbol{v}^{(r)}}$.

It can be seen that this expression depends only on quantities estimated using a single PCA model fit. The expressions in Definition 3.1 are derived by considering a recursive expression for $\boldsymbol{v}_{-i}$ in terms of the original eigenvector $\boldsymbol{v}$ using the matrix inversion lemma

$$\boldsymbol{v}_{-i} = \boldsymbol{v} - \frac{\left(\boldsymbol{x}_i^\top - d_i\boldsymbol{v}\right) Dd_i}{1 - h_i},$$

where $h_i = d_i D d_i$ and $D = \left(\boldsymbol{d}^\top \boldsymbol{d}\right)^{-1}$. Now, using this expression for $\boldsymbol{v}_{-i}$ in Eq (3.1) we obtain the i^{th} PRESS error for PCA as

$$\boldsymbol{e}_{-i} = \frac{\boldsymbol{e}_i}{1 - h_i},$$

where $\boldsymbol{e}_i = \boldsymbol{x}_i - \boldsymbol{x}_i\boldsymbol{v}\boldsymbol{v}^\top$ is the i^{th} reconstruction error. From this expression, it can be seen that the i^{th} leave one out error can be written in terms of the i^{th} reconstruction error and quantities estimated by performing PCA without any explicit leave-one-out steps. Since the contribution of subsequent latent factors to the reconstruction error is additive, we can easily obtain the leave-one-out error for $(R > 1)$ PCA components. This is achieved by simply computing the PRESS errors obtained using $r = 2, \ldots, R$ separately in the same way as for $r = 1$ and adding their contributions.

3.1.2 Approximation error

The ability to derive a recursive form for the PCA PRESS relies on making an approximation about the SVD regarding the mutual orthonormality of singular vectors and sample complexity. This approximation relies upon Assumption 3.1. In this section we describe and justify this assumption.

Assumption 3.1 implies that provided the sample size, N is sufficiently large, we are also able to estimate the PCA model accurately with $N - 1$ samples. This relates to the approximation error between the r^{th} singular value of $\boldsymbol{S}_N = \boldsymbol{X}_N^\top \boldsymbol{X}_N$ which has been estimated using N rows of $\boldsymbol{X}$, $\lambda^{(r)}(\boldsymbol{S}_N)$ and the r^{th} singular value

of $\boldsymbol{S}_{N-1}$ which has been estimated using $N-1$ rows of $\boldsymbol{X}$, $\lambda^{(r)}(\boldsymbol{S}_{N-1})$. That is,

$$\left|\lambda^{(r)}(\boldsymbol{S}_N) - \lambda^{(r)}(\boldsymbol{S}_{N-1})\right| \leq \epsilon,$$

for $1 \leq r \leq R$ where the approximation error, ϵ is arbitrarily small.

Since the rank r approximation error of the SVD is given by λ_{r+1}, if the difference between the pairs of singular values is small, it implies the difference between the corresponding pairs of singular vectors is also small. In other words, within the leave-one-out iterations, it is not necessary to recompute the SVD of $\boldsymbol{X}_{-i}^{\top}\boldsymbol{X}_{-i}$. We introduce the following theorem which details an upper bound on the maximum difference between pairs of ordered singular values of the covariance matrix of $\boldsymbol{X}$ estimated with all N observations and the covariance matrix estimated with $N-1$ observations. The value of the error term defined by the bound is dependent on the size of N and decreases as N increases.

Theorem 3.1. *Let $\boldsymbol{S}_N$ be the covariance matrix of $\boldsymbol{X}_N \in \mathbb{R}^{N\times P}$ and $\boldsymbol{S}_{N-1}$ be the sample covariance matrix of $\boldsymbol{X}_{N-1} \in \mathbb{R}^{(N-1)\times P}$. $\lambda^{(r)}(\boldsymbol{S}_N)$ is the r^{th} largest singular value of $\boldsymbol{S}_N$ then*

$$\max_r \left|\lambda^{(r)}(\boldsymbol{S}_N) - \lambda^{(r)}(\boldsymbol{S}_{N-1})\right| \leq s_{N-1},$$

where

$$s_{N-1} = c\sqrt{\frac{\log(N-1)}{N-1}}a,$$

where c is a constant and $a \geq \|\boldsymbol{x}\|$.

In what follows we provide an intuitive justification for this result and show an example using simulated data. We first establish a bound on the error between the sample covariance matrices $\boldsymbol{S}_N$ and $\boldsymbol{S}_{N-1}$ by adapting the argument in Theorem 3.1 from [79] which details an upper bound on the expected difference between the expected value of the covariance of a random vector $\boldsymbol{x}$, $\boldsymbol{S} = \mathbb{E}[\boldsymbol{x}^{\top}\boldsymbol{x}]$ and its sample covariance matrix using N independent and identically distributed realisations of $\boldsymbol{x}$, $\boldsymbol{S}_N = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i^{\top}\boldsymbol{x}_i$.

Using this result with N and $N-1$ samples we obtain the following inequalities

$$\mathbb{E}\|\boldsymbol{S} - \boldsymbol{S}_N\| \leq s_N \tag{3.4}$$

$$\mathbb{E}\|\boldsymbol{S} - \boldsymbol{S}_{N-1}\| \leq s_{N-1}. \tag{3.5}$$

Although these results only hold in expectation, we would expect the difference $\|\boldsymbol{S}_N - \boldsymbol{S}_{N-1}\|$ to be at least as small as the difference $\|\boldsymbol{S} - \boldsymbol{S}_{N-1}\|$. So, subtracting Eq (3.4) from Eq (3.5) and applying the triangle inequality we arrive at an expression for the difference between terms $\boldsymbol{S}_N$ and $\boldsymbol{S}_{N-1}$ as follows

$$\begin{aligned} |\|\boldsymbol{S} - \boldsymbol{S}_{N-1}\| - \|\boldsymbol{S} - \boldsymbol{S}_N\|| &\leq |s_N - s_{N-1}| \\ \|\boldsymbol{S}_N - \boldsymbol{S}_{N-1}\| &\leq s_{N-1} \end{aligned} \tag{3.6}$$

We now relate this result to the difference between computing the SVD of $\boldsymbol{S}_N$ and the SVD of $\boldsymbol{S}_{N-1}$ by recognising that $\boldsymbol{S}_N$ is obtained as a result of perturbing $\boldsymbol{S}_{N-1}$ by $\boldsymbol{S}_1 = \boldsymbol{x}_1^\top \boldsymbol{x}_1$ where $\boldsymbol{x}_1 \in \mathbb{R}^{1\times P}$ is the single observation missing from $\boldsymbol{S}_N$. Using a result from matrix perturbation theory, Theorem 4.11 from [84, p. 204], which details an upper bound on the maximum difference between the singular values of a covariance matrix $\boldsymbol{M}_{N-1}$ and the perturbed matrix $\boldsymbol{S}_{N-1} + \boldsymbol{S}_1$ we obtain

$$\begin{aligned} \max_r |\lambda^{(r)}(\boldsymbol{S}_{N-1} + \boldsymbol{S}_1) - \lambda^{(r)}(\boldsymbol{S}_{N-1})| &\leq \|\boldsymbol{S}_N - \boldsymbol{S}_{N-1}\| \\ &\leq s_{N-1}, \end{aligned}$$

From this we can see that the result in Theorem 3.1 holds.

This result shows that using the SVD estimated with all of the observations introduces a small error of at most $O\left(\sqrt{\frac{\log(N-1)}{N-1}}\right) \to 0$ as $N \to \infty$ into the computation of $d_i = \boldsymbol{xv}$. This justifies Assumption 3.1 and means that our efficient method of computing the PRESS is almost equivalent to LOOCV in the case of large N.

This result can be illustrated by measuring the approximation error between the leave-one-out error and our PRESS statistic. Figure 3.1 shows the approximation

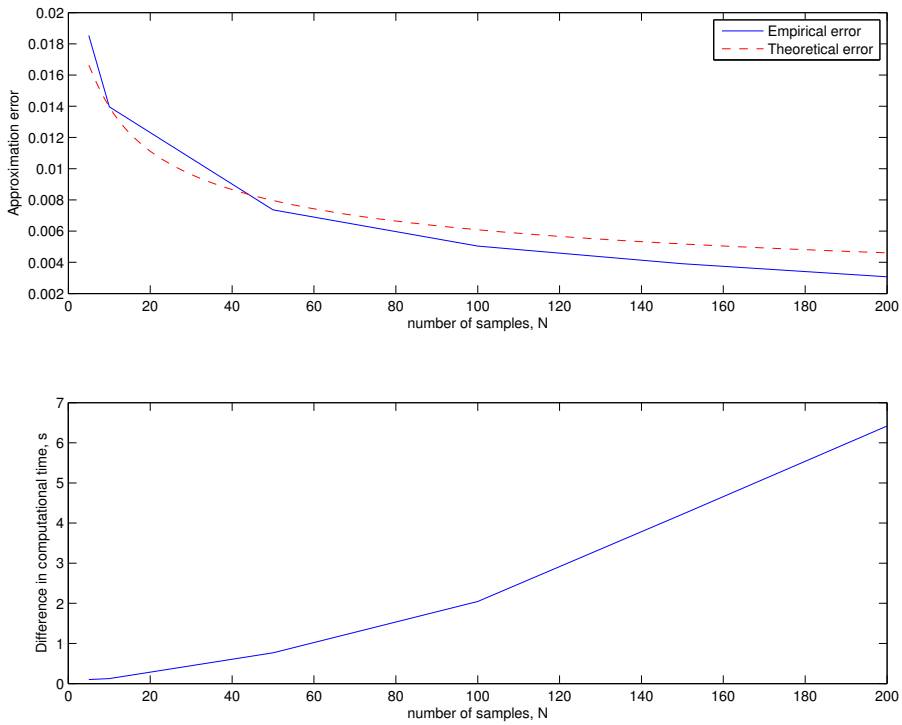


Figure 3.1: The mean squared approximation error between the leave-one-out cross validation and our analytic PRESS statistic as a function of the number of samples, N over 100 Monte Carlo simulations. It can be seen that the empirical approximation error scales according to the theoretical error, $O(\sqrt{\frac{\log N}{N}})$ shown as a dashed line. Also reported is the difference in computational time between the two methods which increases super-linearly with N.

error as a function of N using data simulated under a PCA model with $P = 500$ variables. It can be seen that the decrease in the approximation error follows the theoretical error $O(\sqrt{\frac{\log N}{N}})$ (plotted as a dashed line) and the difference in computational time increases super-linearly.

3.2 A measure of predictive influence for PCA

Using the form for the PCA PRESS derived in the previous section, we now consider how to measure the influence each point exerts on the PCA model. Since we

are interested in the predictive performance of the PCA model, we aim to identify influential points as those observations having the greatest effect on the prediction error. In order to quantify this effect, we define the *predictive influence* with respect to an observation $\boldsymbol{x}_i$ as the rate of change of the PRESS at that point whilst all other observations remain constant.

Definition 3.2. *The predictive influence of a data point $\boldsymbol{x}_i$ under the PCA model parameterised by $\boldsymbol{V}$, which we denote $\boldsymbol{\pi}_{\boldsymbol{x}_i}(\boldsymbol{V}) \in \mathbb{R}^{p\times 1}$, is the gradient of the PRESS with respect to $\boldsymbol{x}_i$ and has the following form*

$$\boldsymbol{\pi}_{\boldsymbol{x}_i}(\boldsymbol{V}) = \boldsymbol{e}_{-i}^{(R)} \left(\sum_{r=1}^{R} \frac{\left(\boldsymbol{I}_P - \boldsymbol{v}^{(r)} {\boldsymbol{v}^{(r)}}^{\top} \right)}{\left(1 - h_i^{(r)} \right)} - (R-1) \right). \tag{3.7}$$

The derivation of the predictive influence is provided in Appendix A.1. The predictive influence measures the sensitivity of the prediction error in response to an incremental change in the observation $\boldsymbol{x}_i$. The rate of change of the PRESS at this point is given by the magnitude of the predictive influence vector, $\|\boldsymbol{\pi}_{\boldsymbol{x}_i}(\boldsymbol{V})\|^2$. If the magnitude of the predictive influence is large, this implies a small change in the observation will result in a large change in the prediction error relative to other points. In this case, removing such a point from the model would cause a large improvement in the prediction error. We can then identify the most influential observations as those for which the increase in the PRESS is larger relative to other observations.

It was observed in [21] that single deletion methods for identifying influence, such as the PRESS, are only sensitive to large errors. Instead they propose weighting each observation and observing the change in the error as each weight is perturbed between 0 (equivalent to LOOCV) and 1 (equivalent to the standard residual) by computing the derivative of the residual with respect to the weight. They term this quantity *local* influence. In this context, our proposed predictive influence measure can be viewed as a measure of local influence. Here, the gradient of the PRESS with respect to each observation quantifies the effect of a perturbation in that obser-

vation. This ensures we arrive at a quantity which is more sensitive to influential observations than other methods which rely on the leave one out error alone.

It can be seen that the form of the predictive influence has more in common with the the measures of detecting influential observations for OLS, than those for PCA as reviewed in Section 2.4.2. Whereas measures of detecting influence in PCA typically involve examining the leave-one-out estimates of the eigenvalues or eigenvectors, instead our predictive influence measure consists of a function of the residual error scaled by a term analogous to the leverage of that point under a PCA model.

3.3 Predictive robust PCA (PRoPCA)

In section 2.4.2 we noted that dealing with influential observations in PCA is a challenging problem. We reviewed an approach to dealing with such observations without removing them for the model. Instead, robust PCA methods seek to identify and downweight the importance of such observations whilst estimating the PCA model. In this section we consider how such a robust approach can be incorporated into the framework for predictive influence developed in the previous section.

The predictive influence exerted by the i^{th} observation on the PCA model, given in Definition 3.2, consists of the i^{th} PCA reconstruction error weighted by its leverage under that model. This weight effectively determines the importance of each observation under a standard, non-robust PCA model. We can downgrade the importance of influential observations by taking these weights into account. Therefore, rather than obtaining loading vectors which minimise the PCA reconstruction error, we aim to obtain loadings which minimise the weighted reconstruction error, that is the predictive influence of each observation.

For $R = 1$ we can state our problem as

$$\min_{\boldsymbol{v}} \sum_{i=1}^{N} \|\boldsymbol{\pi}_{\boldsymbol{x}_i}(\boldsymbol{v})\|^2, \quad \text{subject to } \|\boldsymbol{v}\| = 1. \tag{3.8}$$

The following lemma shows that the solution to (3.8) can be obtained by iteratively re-estimating a PCA model where each observation is weighed by its leverage.

Lemma 3.1. *Solving the minimisation problem,* (3.8) *is equivalent to solving the following maximisation problem*

$$\max_{\boldsymbol{v}} \boldsymbol{v}^\top \boldsymbol{X}^\top \boldsymbol{\Xi}^{-2} \boldsymbol{X} \boldsymbol{v}, \quad \text{subject to } \|\boldsymbol{v}\| = 1. \tag{3.9}$$

where $\boldsymbol{\Xi} \in \mathbb{R}^{N \times N}$ *is a diagonal matrix with diagonal elements* $\Xi_i = (1 - h_i)^2$.

The proof of this Lemma is provided in Appendix A.2. From Lemma 3.1, the objective function, (3.9) can be recognised as an eigenproblem where each observation has been weighted by a function of its leverage under the PCA model. However, each diagonal element of $\boldsymbol{\Xi}$ is defined as a non-linear function of $\boldsymbol{v}$. Therefore, if $\boldsymbol{\Xi} \neq \boldsymbol{I}_N$, Eq. (3.9) cannot be solved analytically using a single eigendecomposition of $\boldsymbol{X}^\top \boldsymbol{\Xi}^{-2} \boldsymbol{X}$. Instead, the optimal parameters can be obtained by using the following iterative procedure which we term Predictive Robust PCA (PRoPCA)

Starting with the principal eigenvector, $\boldsymbol{v}$ of $\boldsymbol{X}^\top \boldsymbol{X}$:

1. Compute the entries of the matrix, $\boldsymbol{\Xi}$ where $\Xi_i = \left(1 - \frac{(\boldsymbol{x}\boldsymbol{v}')^2}{\boldsymbol{v}^\top \boldsymbol{X}^\top \boldsymbol{X} \boldsymbol{v}}\right)^2$.
2. Compute the principal eigenvector, $\boldsymbol{v}^*$ of $\boldsymbol{X}^\top \boldsymbol{\Xi}^{-2} \boldsymbol{X}$.
3. $\boldsymbol{v} \leftarrow \boldsymbol{v}^*$.

Repeat until $1 - \boldsymbol{v}^\top \boldsymbol{v}^* < \text{tol}$.

Successive robust principal components can be computed by deflating $\boldsymbol{X}$ by removing the contribution of the previously extracted component, $\boldsymbol{X} = \boldsymbol{X} - \boldsymbol{X}\boldsymbol{v}^*\boldsymbol{v}^{*\top}$ and repeating the above procedure.

In the following section we present results showing the ability of the predictive influence and PRoPCA to detect and deal with influential observations using data from a real application in computer vision.

Figure 3.2: The ten subjects of the Yale faces B database under ambient lighting conditions.

3.4 An example application to face images

In this section we present a series of examples on a real dataset which highlight the ability of the predictive influence to detect influential observations. We use the Yale faces B database [33] which consists of frontal images of ten individual faces taken under 64 different illumination conditions. Each image has dimensions 120×160 pixels. We represent each image by a single 19200-dimensional vector and concatenate all the images so that the full dataset is represented by a matrix $\boldsymbol{X} \in \mathbb{R}^{64K \times 19200}$. It should be noted that by representing the images in this way we make no use of the spatial information in each image. Figure 3.2 shows an example image of each of the ten subjects under ambient lighting conditions.

Under the assumption that its pose is constant, it is known that images of an object under all illumination conditions form a convex cone in image space. This cone can be well approximated by a low-dimensional, linear subspace [33]. When applied to the Yale faces database, this implies that images of a single individual can be well approximated by a standard PCA model. Figure 3.3 shows an example of an individual under ten of the different lighting conditions.

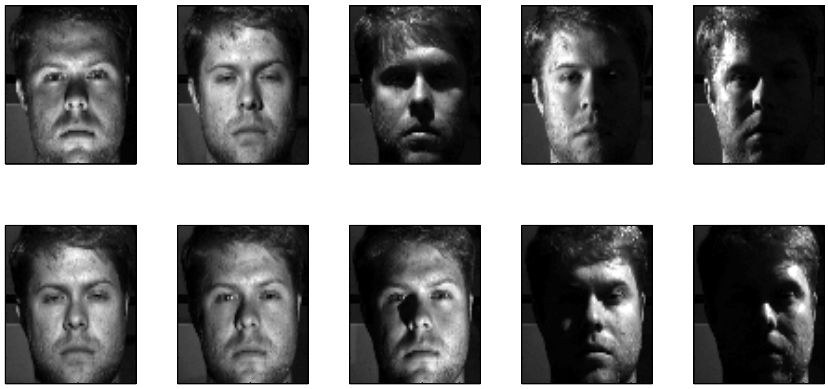


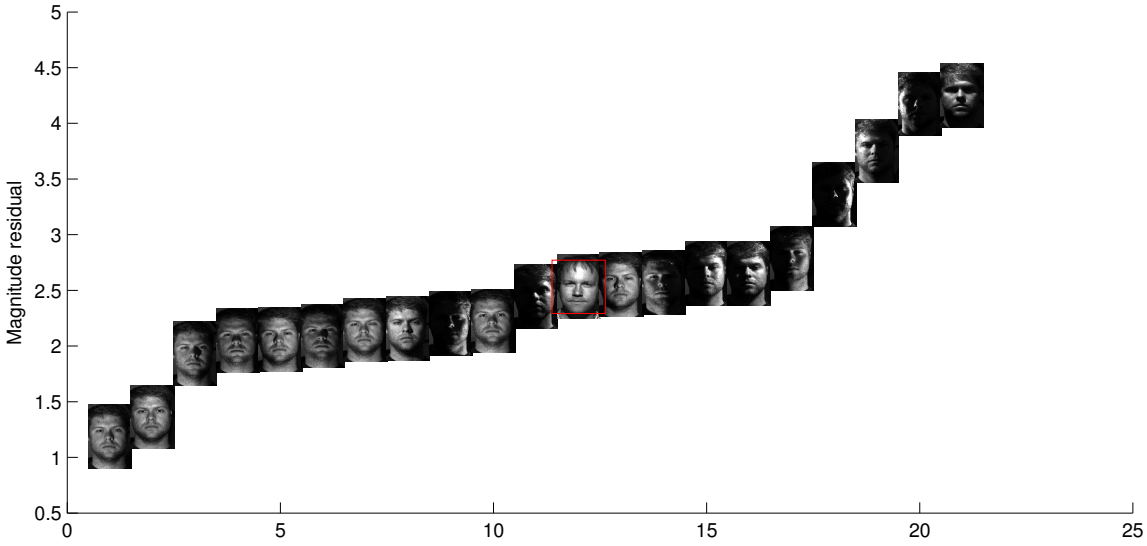
Figure 3.3: An example a subject under ten of the 64 different lighting conditions.

Detecting influence

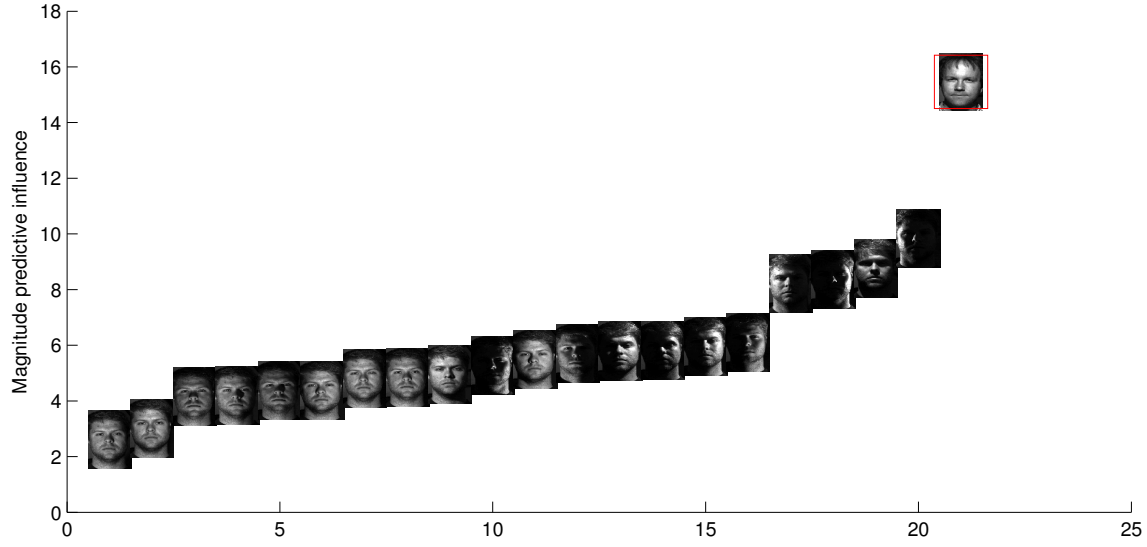
We first provide illustrative results which show the ability of the PCA predictive influence to identify influential observations compared to the PCA residual. In the context of face detection, this amounts to identifying faces of different individuals. We constructed a dataset consisting of $N_1 = 20$ faces randomly selected from subject one. We then select a single face $N_2 = 1$ randomly out of all the remaining nine subjects such that $N = N_1 + N_2 = 21$. This is considered to be the "influential" face. We train a standard PCA model on the data and examine both the residual error and predictive influence of each observation.

Figure 3.4 shows the result of a single example of this experiment. Figure 3.4a shows the magnitude residuals of each face. A single influential face (marked by a red box) exhibits a residual which is approximately the mean residual. The implies that the influential observation is not identifiable by simply examining the residual error. Figure 3.4b shows the magnitude predictive influence of each face exerted on the PCA model. The influential faces exhibits a noticably larger influence than all other faces.

We then performed a Monte Carlo study consisting of 300 simulations each with $N_1 = 20$ faces randomly selected from subject one. We then select $N_2 = 3$ influential faces randomly out of all of the remaining nine subjects so $N = 23$.



(a) PCA residual



(b) Predictive influence

Figure 3.4: An example of detecting a single influential observation (indicated by the red box) using (a) the PCA residual and (b) the predictive influence.

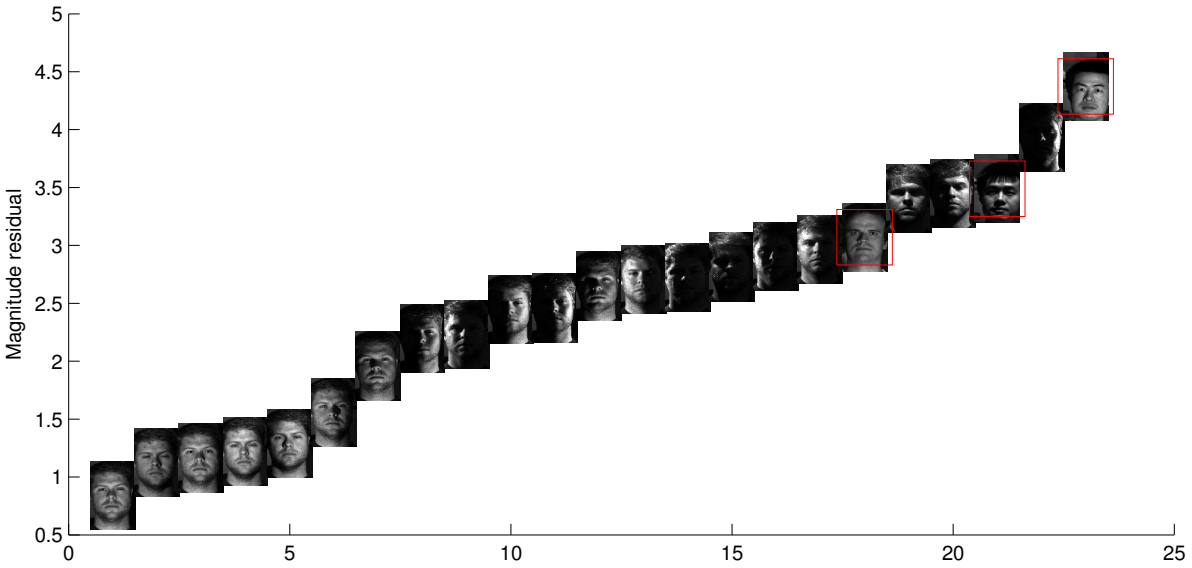
For iteration, $M = 1, \ldots, N$ we examine the M faces with the largest magnitude predictive influence and the M faces with the largest magnitude residuals. An influ-

ential observation is correctly identified if it is amongst those M selected points. All other observations amongst those M selected points are considered false positives.

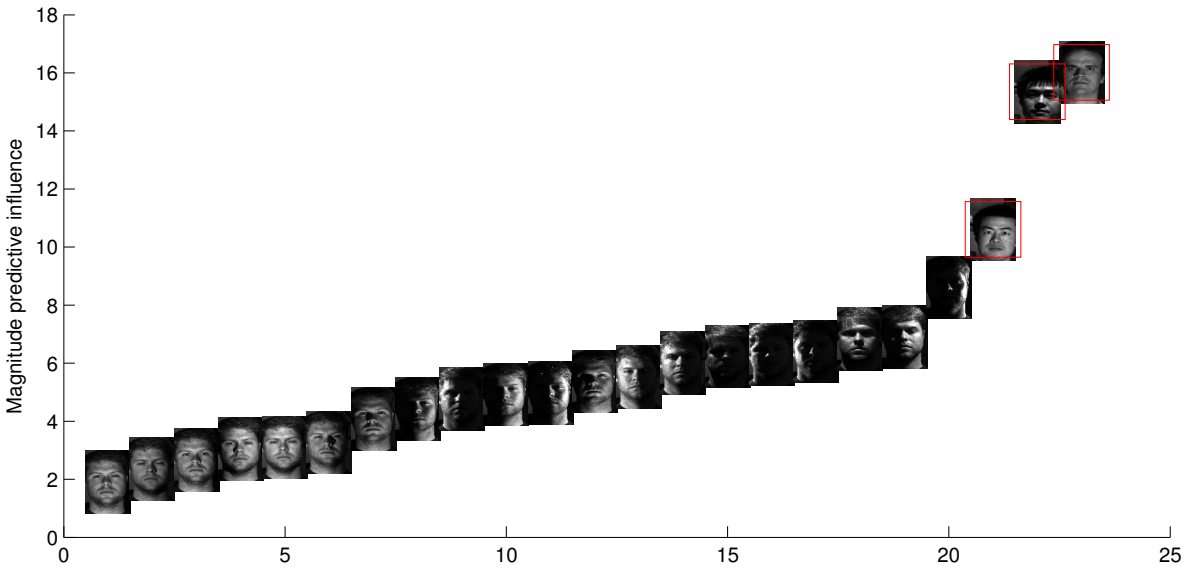
Since $N_2 = 3$, this can introduce the effects of masking and swamping which typically make it more difficult to identify groups of influential observations. Due to the effect of *masking*, single observation tests may not identify the influential points so it may be necessary to test groups of points simultaneously. However, if we test groups of observations simultaneously where the size of each group is larger than N_2, the effect of one of the influential observations may *swamp* the others and so they may be determined to be non-influential. Therefore it is helpful if we can detect influential observations by simply testing each observation individually.

Figure 3.5 reports on a single example of this study. Figure 3.5a shows the magnitude residuals of each face. The three influential faces exhibit residuals which are larger than the mean residual. However the influential faces are not identifiable by simply examining the residual error. Figure 3.5b shows the magnitude predictive influence of each face exerted on the PCA model. Two of the influential faces exhibit a noticably larger influence than all other faces. All three influential faces exhibit larger influence than the those images belonging to subject one. It has been shown that local influence measures are typically more resistant to the effects of masking and swamping [76]. Therefore, even in the presence of multiple influential observations evaluating the predictive influence at each point individually is sufficient to identify the influential observations and so the effects of masking and swamping are negated.

Figure 3.6 compares the receiver operator characteristic of the predictive influence with that of the residual error over the 300 Monte Carlo simulations. It can be seen that using the PCA residual to detect influential observations is extremely prone to identifying false positives. Maximum sensitivity is only achieved at a false positive rate of 0.97. On the other hand, the predictive influence is able to detect the influential points accurately with a true positive rate of 0.99 occurring at a false positive rate of 0.3. The above results confirm that in the presence of influential observations, the residual error tends to overfit the data. This causes the influential observations to be undetectable by simply examining the residual errors. However,



(a) PCA residual



(b) Predictive influence

Figure 3.5: An example of detecting three influential observations (indicated by the red boxes) using (a) the PCA residual and (b) the predictive influence.

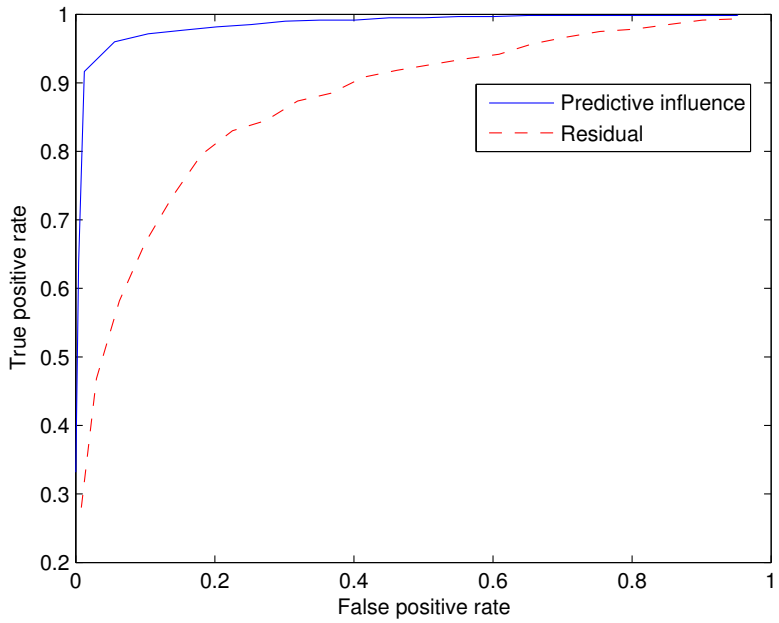


Figure 3.6: A comparison of the receiver operating characteristic curve for identifying influential observations obtained by the predictive influence and the PCA residual. The predictive influence identifies 99% of true positives at a false positive rate of 0.3 whereas using the residual, this true positive rate is achieved at a false positive rate of 0.97.

the predictive influence is able to accurately detect both single and multiple influential observations.

Robust estimation using PRoPCA

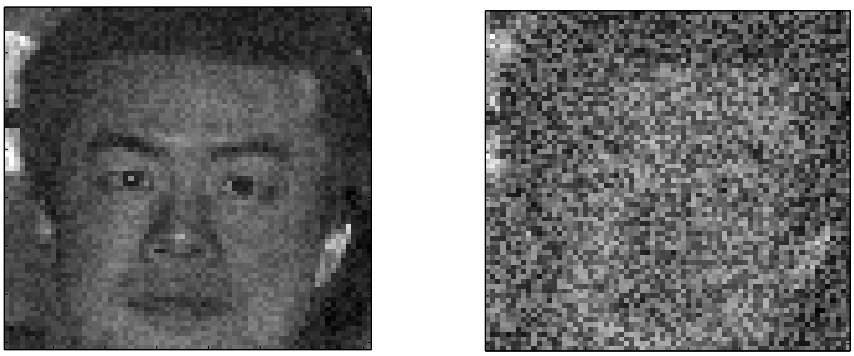
We now provide an illustrative example of using PRoPCA as a scheme for robust estimation of subspaces by incorporating influential observations instead of removing them. We construct a training set consisting of 15 images from subject one. We then add a single outlying image which has been corrupted with additive uniform noise. Two examples of these noisy images are shown in figure 3.7. Figure 3.7a shows an image which has been corrupted with low noise such that the main features of the image are still distinguishable. Figure 3.7b shows an image which has been corrupted with high noise such that any recognisable features have been

masked by the noise. These are considered the influential observations. For both PCA and ProPCA we fit the model using the training set then evaluate the mean predictive reconstruction error on the image of the first subject taken under ambient lighting conditions (see figure 3.2). In this way we evaluate the predictive performance when the model has been trained with a dataset consisting of influential observations. In this case, we would expect standard PCA to overfit so that the resulting image would be a poor representation of the true image. We also compare the result obtained by PRoPCA with that of the ROBPCA method reviewed in Section 2.4.2.

Figure 3.8 shows the comparison between the PCA reconstruction and the PRoPCA reconstruction in the presence of a single noisy influential observation. Figure 3.8a shows the result of the presence of an outlier corrupted by low noise. There is a noticible loss of detail in the image reconstructed using PCA, especially in the hair and the separation between the face and the background. On the other hand, the reconstruction using PRoPCA has retained most of the detail in the image. Figure 3.8b shows the result of an outlier corrupted with high noise. The image reconstructed using PCA retains no detail or distinguishable features. The image reconstructed using PRoPCA still retains most of the details of the original image including highlight detail especially in the eyes and forehead.

This example illustrates the extreme effect that even a single influential observation can have on the PCA model and underscores the importance of being able to reliably detect such observations. It also shows the ability of PRoPCA to estimate an accurate representation of the subspace even in the presence of such influential observations. This shows the suitability of PRoPCA as a solution for dealing with influential observations within the PCA framework.

Figure 3.9 shows the reconstruction using Robust PCA (ROBPCA). ROBPCA clearly achieves a good reconstruction of the original face in the presence of low and high noise and achieves a smaller reconstruction error than PRoPCA in both cases. However qualitatively, it can be seen that much of the “noise” removed by ROBPCA is actually signal and so all the detail in the background, forehead and hair has been removed and the separation between the subject and the background is

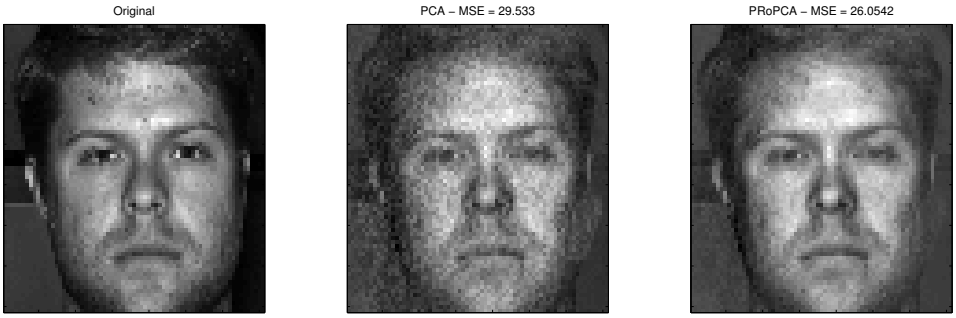


(a) Low noise. (b) High noise.

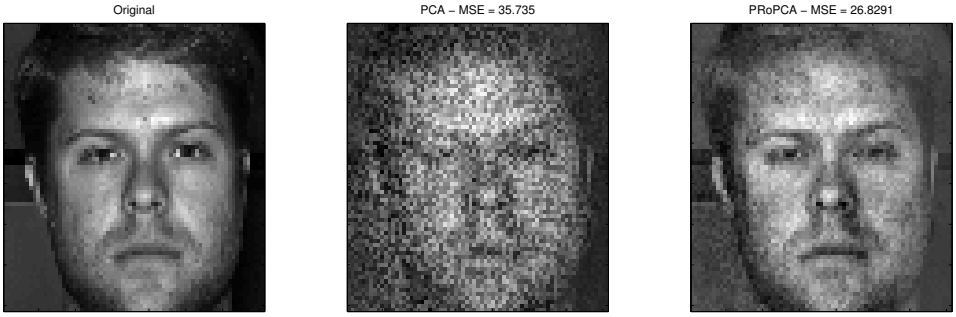
Figure 3.7: An outlying image corrupted with two different levels of additive uniform noise. A low level of noise preserves most of the features. A high level of noise removes almost all of the details.

less pronounced. This comparison highlights the importance of interpreting results obtained by PCA qualitatively as well as in terms of reconstruction error.

It should be noted that these methods are useful when only a few influential observations are present in the dataset. However, if a large number of influential observations are present, it may be more realistic to change our assumptions about the data such that instead, we assume several heterogenous sub-populations exist in the dataset whose members are not known *a priori*. In the following chapter we show that the ability to accurately identify influential observations can be successfully applied to the field of *subspace clustering*.

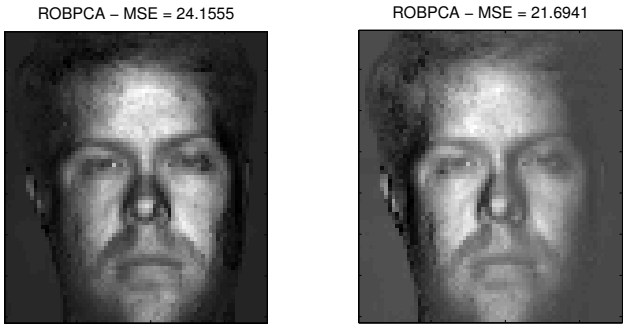


(a) Low noise.



(b) High noise.

Figure 3.8: A comparison between the PCA reconstruction and the PRoPCA reconstruction when trained on 15 examples of the target face and one example of an outlying face corrupted with additive uniform noise.



(a) Low noise. (b) High noise.

Figure 3.9: Image reconstruction using ROBPCA for the low noise setting, (a) and the high noise setting, (b). When compared to Figure 3.8, ROBPCA always achieves a smaller reconstruction error than PRoPCA. However, some of the detail in the image is sacrificed as a result.

Chapter 4

Predictive subspace clustering

So far we have developed a framework to detect influential observations and estimate robust PCA models in the presence of such observations. We now revisit the linear subspace clustering problem introduced in Section 2.5.2. It is known that learning the number of clusters, the dimension of each subspace, and the correct assignments is a challenging task. Many of the existing algorithms often perform poorly in the presence of subspaces that have different dimensions and possibly overlap, or are otherwise computationally expensive. In this chapter we present a novel approach to subspace clustering that learns the numbers of clusters and the dimensionality of each subspace in an efficient way. We assume that the data points in each cluster are well represented in low-dimensions by a PCA model so that misclustered observations can be identified as exerting a large influence on the PCA model. We drive the clustering process by assigning points to clusters such that the predictive influence exerted by each point is minimised. The proposed *predictive subspace clustering* (PSC) algorithm is assessed on both simulated data and on the popular Yale faces database where state-of-the-art performance and speed are obtained.

4.1 Clustering based on predictive reconstruction

In Chapter 2 we reviewed recent subspace clustering methods and identified several important limitations of the current state-of-the-art. The most important limitation concerns estimating subspaces of different dimensionality. In the case of GPCA, the dimensionality of each subspace must be specified in advance. In the case of successful spectral methods such as SSC, SCC and SLBF, the dimensionality of the largest subspace must be specified. This raises problems when either the exact dimensions are not known or are significantly different from each other. When the difference between $\max_k(R_k)$ and $\min_k(R_k)$ increases, these algorithms become less able to identify lower-dimensional subspaces.

SSC and SLBF are able to achieve state-of-the-art clustering performance because they adopt computationally expensive strategies to estimate a local representation of the subspace at each observation. Such strategies are necessary to ensure these methods are robust to noise, outliers and are able to correctly assign points on the intersection between subspaces. However, as P and N increase, these methods become prohibitively expensive.

In this chapter we addresses these limitations by proposing a novel approach to solving the subspace clustering problem. As with K-subspaces, we iteratively fit cluster-wise PCA models and reassign points to clusters until a certain optimality condition is met. This keeps the computational cost of the algorithm small. However, rather than trying to minimise the residuals under the individual PCA models we adopt an approach more similar to spectral clustering-based algorithms which evaluate the local properties of the subspace at each observation. We introduce an objective function based on minimising the predictive influence within each cluster such that it particularly robust to noise and outliers. This also enables the resulting algorithm to learn both the number of clusters and the dimensionality of each subspace.

Specifically, the problem we tackle is the linear subspace clustering problem as described in Section 2.5.2. That is, we observe N points, $\boldsymbol{x}_i$ where each point belongs to one of K clusters, $\{\mathcal{C}_k\}_1^K$. Points within each cluster lay on an R_k-

dimensional linear subspace, $\mathcal{S}_k$ as defined in (2.20). We consider the general case and assume that the number of dimensions in each subspace, R_k is unknown and can be different between subspaces. Our aim is therefore to simultaneously:

1. Recover the true cluster assignments, $\mathcal{C} \equiv \{\mathcal{C}_k\}_1^K$.

2. Estimate the subspace parameters, $\mathcal{S} \equiv \{\mathcal{S}_k\}_1^K$ corresponding to each cluster.

We aim to assign observations such that the reconstruction of points on the subspace is optimal in some way. As seen by the many approaches to subspace clustering, the definition of an optimal assignment rule is an open question. However, as we have seen in Section 3.4, minimising the PCA reconstruction error is prone to overfitting. For instance, the data may be corrupted by noise or lie on the intersection between subspaces and so points within clusters may be geometrically far apart. Since the PCA reconstruction error is not robust to outliers, such points may bias the estimated subspace which is a fundamental limitation for assigning points to clusters. Furthermore, since the PCA reconstruction error decreases monotonically as a function of dimensionality, in the situation where each subspace has a different intrinsic dimensionality points may be wrongly assigned to the cluster with the largest dimensionality. Such an approach therefore limits the number of dimensions to be the same in each cluster.

To avoid these problems relating to overfitting, we instead take an approach based on assigning points such that the *out-of-sample prediction* of the reconstruction error is minimised. In this way, we aim to recover clusters whose points truly belong to the underlying subspace. In each cluster we can quantify the influence exerted on the reconstruction error by each point on each PCA model. In Section 3.2 we introduced a method for evaluating the predictive influence of a point on the PCA reconstruction. We showed how using this measure, influential observations which otherwise exhibit small residual errors can be identified. In some cases, these points may not belong to the underlying PCA model and so should be removed. In the case where we have heterogeneous subsets of points lying on different subspaces, the predictive influence can be used to identify observations which may have been misclustered.

The proposed algorithm relies on the following observation. If the cluster assignments were known and a PCA model was fit to the data in each cluster, then the predictive influence of a point $\boldsymbol{x}_i$ belonging to cluster $\mathcal{C}_k$ should be small when evaluated using the correct PCA model for that cluster, and would be larger when using any of the remaining $K-1$ PCA models. In this respect, the predictive influence provides a goodness of fit measure that can be used to drive the clustering process.

The objective of the clustering algorithm is to partition the N observations $\boldsymbol{x}_i$ into one of K non-overlapping clusters such that each cluster contains exactly N_k observations and $\sum_{k=1}^{K} N_k = N$. Assuming that K is known, we recover the partitioning by ensuring each point is assigned to the cluster for which it exerts the smallest predictive influence relative to all other PCA models. This is achieved by minimising the following objective function.

Definition 4.1. *The sum of within-cluster predictive influences is given by*

$$C(\mathcal{S}, \mathcal{C}) = \sum_{k=1}^{K} \sum_{i \in \mathcal{C}_k} \|\boldsymbol{\pi}_{\boldsymbol{x}_i}(\boldsymbol{V}_k)\|^2, \tag{4.1}$$

where $\boldsymbol{\pi}_{\boldsymbol{x}_i}(\boldsymbol{V}_k)$ is the predictive influence of a point $\boldsymbol{x}_i$ under the k^{th} PCA model.

It is clear that if the expression in Eq. (4.1) is minimised, the prediction error for each PCA model will be minimised since all points in each cluster will exert minimum predictive influence. Therefore, we aim to recover clusters such that the predictive influence of points within each cluster is minimised. We refer to this approach as Predictive Subspace Clustering (PSC).

The objective function considers the predictive reconstruction error between each point and each subspace. By assigning points to clusters such that the predictive reconstruction within a cluster is minimised, the model estimation and selection problems are resolved simultaneously. If we compare Eq. (4.1) with (3.8), we see that minimising the objective function can be viewed as estimating a PRoPCA model in each cluster.

Minimising Eq. (4.1) requires determining the true partitioning of the observations and estimating PCA model parameters for those K partitions simultaneously. Since the true partitioning is unknown, there is no analytic solution to this problem. Instead, estimating both the subspaces and the optimal cluster assignments can be attacked by considering the two related optimisation problems:

1. Given K subspaces with parameters, $\mathcal{S}$ and keeping these fixed, recover the cluster assignments which solve

$$\min_{\{\mathcal{C}_1,\ldots,\mathcal{C}_K\}} C(\mathcal{S},\mathcal{C}). \tag{4.2}$$

2. Given a set of cluster assignments, $\mathcal{C}$ and keeping these fixed, estimate the parameters of the K subspaces

$$\begin{aligned} &\min_{\{\boldsymbol{V}_1,\ldots,\boldsymbol{V}_K\}} C(\mathcal{S},\mathcal{C}), \\ &\text{subject to} \quad \boldsymbol{V}_k^\top \boldsymbol{V}_k = I_P. \end{aligned} \tag{4.3}$$

In the following section we propose an iterative algorithm to solve this problem.

4.2 The PSC algorithm

Since obtaining cluster assignments by solving (4.2) changes the PCA model parameters obtained as a result of solving (4.3) and *vice versa*, these objective functions must be solved iteratively. We propose an algorithm which minimises Eq. (4.1) by alternately solving (4.2) and (4.3). At each iteration, the PSC algorithm follows three steps which are outlined below.

Initialisation (I): Given an initial partitioning of the data, $\mathcal{C}$, both the PCA model parameters and the predictive influences, $\boldsymbol{\pi}_{\boldsymbol{x}_i}(\boldsymbol{V}_k)$, are computed for all $k = 1,\ldots,K$ clusters and $i = 1,\ldots,N$ observations.

Partitioning (P): Keeping the model parameters fixed, the cluster assignments that minimise (4.2) are obtained by assigning points to the clusters, such that the set

$\mathcal{C}_k$ consists of the indices, i corresponding to the observations which exert minimal predictive influence under the k^{th} PCA model,

$$\mathcal{C}_k \leftarrow \left\{ i : \min_k \|\boldsymbol{\pi}_{\boldsymbol{x}_i}(\boldsymbol{V}_k)\|^2 \right\}_{i=1}^{N}, \text{ for } k = 1, \ldots, K. \tag{4.4}$$

Estimation (E): Keeping the cluster allocations fixed, the algorithm estimates the parameters, $\mathcal{S}$ of the PCA models using the new cluster assignments obtained during the **P** step. For all clusters, $k = 1, \ldots, K$, we estimate new PCA model parameters using $\{\boldsymbol{x}_i\}$ for all $i \in \mathcal{C}_k$.

The main computational cost involved in the PSC algorithm is the compuation of the SVD in each cluster. Every other quantity required by the algorithm is either directly computed using the SVD or is cheaply computed through a scalar operation. Therefore, each iteration of the PSC algorithm requires $O\left(KP^2 + N^2\right)$ operations. Furthermore, since all of the operations in each cluster can be computed independently for each iteration, the PSC algorithm is suited to highly parallelised implementation. This can be extended to computing each PCA PRESS component in its own thread which will result in an extremely fast implementation.

4.2.1 Convergence of PSC

In this section we demonstrate that the proposed PSC algorithm converges to a local minimum of the objective function in Eq. (4.1). We first provide a sketch of the argument and show explicitly the effect on the objective function of applying the **P** step and the **E** step. We consider the case where $R = 1$ however this holds for $R > 1$. At the beginning of each iteration we have cluster assignments $\mathcal{C}^{old}$ and corresponding PCA parameters $\mathcal{S}^{old}$. Using these values, we evaluate the objective function, Eq. (4.1), to obtain $C(\mathcal{S}^{old}, \mathcal{C}^{old})$.

We then perform Step **P** to obtain $\mathcal{C}^{new}$. Keeping the previous values of the parameters, $\mathcal{S}^{old}$, we obtain a new value of the objective function, $C(\mathcal{S}^{old}, \mathcal{C}^{new})$,

which satisfies

$$C(\mathcal{S}^{old}, \mathcal{C}^{new}) \leq C(\mathcal{S}^{old}, \mathcal{C}^{old}),$$

since following the assignment rule, (4.4) minimises the objective function by definition.

Keeping $\mathcal{C}^{new}$ fixed, we perform Step **E** to obtain $\mathcal{S}^{new}$, where the parameters are estimated independently in each cluster. In order to show the algorithm converges we must show that the new value of the objective function, $C(\mathcal{S}^{new}, \mathcal{C}^{new})$, satisfies

$$C(\mathcal{S}^{new}, \mathcal{C}^{new}) \leq C(\mathcal{S}^{old}, \mathcal{C}^{new}). \tag{4.5}$$

Since the cluster assignments are fixed, from Lemma 3.1 we know that for $R = 1$, solving (4.3) is is equivalent to solving

$$\begin{aligned} &\max_{\boldsymbol{v}} \sum_{i \in \mathcal{C}_k^{new}} \boldsymbol{v}^\top \boldsymbol{x}_i^\top \Xi_i^{-2} \boldsymbol{x}_i \boldsymbol{v} \\ &\text{subject to } \|\boldsymbol{v}\| = 1, \end{aligned} \tag{4.6}$$

within each cluster. Following this procedure yields the optimal parameters, $\mathcal{S}^* = \{\boldsymbol{v}_k^*\}_1^K$. Using these parameters we obtain a value of the objective function, $C(\mathcal{S}^*, \mathcal{C}^{new})$ which is minimised and thus satisfies (4.5). However, these optimal parameters are obtained using the iterative PRoPCA procedure in Section 3.3 which requires multiple, expensive SVD operations. Instead, we prove that estimating $\mathcal{S}^{new} = \{\boldsymbol{v}_k^{new}\}_1^K$ by fitting PCA models independently in each cluster, as in Step **E**, achieves a value of $C(\mathcal{S}^{new}, \mathcal{C}^{new})$ which satisfies (4.5).

Associated with this step is an approximation error between the PRoPCA solution and the standard PCA solution. The following lemma shows that estimating a new PCA model at each iteration always decreases this approximation error and the objective function.

Lemma 4.1. *For each cluster k, we define the approximation error between the optimal parameters $\boldsymbol{v}_k^*$ obtained by solving* (3.9)*, and the old PCA parameters $\boldsymbol{v}_k^{old}$*

as

$$E(\mathcal{S}^*, \mathcal{S}^{old}) = \sum_{i \in \mathcal{C}_k^{new}} \boldsymbol{v}_k^{*\top} \boldsymbol{x}_i^\top \Xi_{k,i}^{-2} \boldsymbol{x}_i \boldsymbol{v}_k^* - \sum_{i \in \mathcal{C}_k^{new}} \boldsymbol{v}_k^{old\top} \boldsymbol{x}_i^\top \boldsymbol{x}_i \boldsymbol{v}_k^{old}.$$

We then define the approximation error between the optimal parameters and the new PCA parameters as

$$E(\mathcal{S}^*, \mathcal{S}^{new}) = \sum_{i \in \mathcal{C}_k^{new}} \boldsymbol{v}_k^{*\top} \boldsymbol{x}_i^\top \Xi_{k,i}^{-2} \boldsymbol{x}_i \boldsymbol{v}_k^* - \sum_{i \in \mathcal{C}_k^{new}} \boldsymbol{v}_k^{new\top} \boldsymbol{x}_i^\top \boldsymbol{x}_i \boldsymbol{v}_k^{new}.$$

These error terms satisfy the inequality,

$$E(\mathcal{S}^*, \mathcal{S}^{new}) \leq E(\mathcal{S}^*, \mathcal{S}^{old}). \tag{4.7}$$

The proof of this lemma is provided in Appendix B. This lemma states that estimating a new PCA model within each cluster always yields a solution which is closer to the optimal solution than the PCA model estimated at the previous iteration.

Theorem 4.1. *Starting with any cluster configuration, $\{\mathcal{C}_k^{old}\}_1^K$, the PSC algorithm converges to a local minimum of the objective function, Eq.* (4.1).

Proof. We demonstrate that when performing a single iteration the objective function is always decreased, that is

$$C(\mathcal{S}^{new}, \mathcal{C}^{new}) \leq C(\mathcal{S}^{old}, \mathcal{C}^{new}) \leq C(\mathcal{S}^{old}, \mathcal{C}^{old}). \tag{4.8}$$

By definition, the cluster assignments, $\mathcal{C}^{new}$ obtained in Step **P** minimise the objective function and so

$$C(\mathcal{S}^{old}, \mathcal{C}^{new}) \leq C(\mathcal{S}^{old}, \mathcal{C}^{old}).$$

From Lemma 3.1, minimising the objective function by solving the minimisation problem, (4.3) is equivalent to solving the maxmisation problem, (4.6) in each cluster. This yields the optimal PSC parameters which obtain the minimum value of

the objective function, $C(\mathcal{S}^*, \mathcal{C}^{new})$ such that $C(\mathcal{S}^*, \mathcal{C}^{new}) \leq C(\mathcal{S}^{old}, \mathcal{C}^{new})$, which is the value of the objective function using the old PCA parameters. From Lemma 4.1, the difference between the optimal PSC parameters and the old PCA parameters is given by $E(\mathcal{S}^*, \mathcal{S}^{old})$.

Similarly, estimating the new PCA parameters yields a value of the objective function, $C(\mathcal{S}^{new}, \mathcal{C}^{new}) \geq C(\mathcal{S}^*, \mathcal{C}^{new})$ where the difference between the optimal parameters and the new PCA parameters is given by $E(\mathcal{S}^*, \mathcal{S}^{new})$. Since from Lemma 4.1, $E(\mathcal{S}^*, \mathcal{S}^{new}) \leq E(\mathcal{S}^*, \mathcal{S}^{old})$, in Step **E** of the PSC algorithm we estimate new PCA parameters which are closer to the optimal parameters obtained by solving (4.6) in each cluster. We therefore obtain a new value of the objective function which satisfies

$$C(\mathcal{S}^{new}, \mathcal{C}^{new}) \leq C(\mathcal{S}^{old}, \mathcal{C}^{new}).$$

Since (4.8) holds at each iteration, as a result the objective function always decreases and the PSC algorithm converges to a locally optimal solution. □

4.2.2 Model selection in PSC

Model selection is an integral part of the PSC algorithm. As mentioned in Section 2.5.2, identifying the true subspaces and thus recovering the true cluster assignments depends heavily on accurate selection of the parameters, K and R_k. The ability to learn these parameters and have R_k be different for each cluster is an open problem. The PRESS statistic provides a robust method for efficiently evaluating the fit of the PCA models within our framework. Straightforward extensions of the basic algorithm allow us to identify the optimal number of clusters, K, and the dimensionality of each subspace, $\{R_k\}_1^K$.

Assuming K is known at each iteration, using all data points in each cluster k, we evaluate all PCA PRESS statistics as in Eq. (3.3) using all values of each $R_k \in \{1, \ldots, R_{max}\}$. We select each R_k such that it minimises the PRESS in cluster k.

The number of subspaces, K is estimated using a scheme in which we dynam-

ically add and remove clusters from the model. If a cluster is not supported by the data, it is allowed to drop out of the model naturally. We add a cluster by identifying the cluster which exhibits the largest PRESS after convergence and dividing its observations between two new clusters, thereby increasing the number of clusters to $K+1$. This process continues until the overall PRESS is no longer decreased by adding a new cluster. Furthermore, the splitting operation makes the PSC algorithm less susceptible to local optimal solutions as it performs a more thorough search of the possible cluster configurations.

Our PRESS statistic for PCA allows both K and R_k to be efficiently learned from the data. Given that the SVD at each iteration and for each cluster has been computed up to dimension R_{max}, the additional computational effort required to evaluate all R_{max} values of the PRESS is negligible since all the quantities required to compute the PRESS are obtained via the SVD. Furthermore, PSC only requires the user to specify the single parameter, R_{max}. Since each one of the R_k values is typically much smaller than P, we can set R_{max} to be small relative to P so the computation of the full, P-dimensional SVD is not necessary.

4.3 Connection with K-subspaces

Our proposed PSC method for subspace clustering has noticible similarities to the K-subspaces method in that it iteratively fits K PCA models to the data and assigns points to clusters based on a measure of distance between each point and the PCA models. If we do not take into account the model selection aspect of PSC, for a fixed number of dimensions and clusters, PSC and K-subspaces differ only in the assignment step. A key question then is how much difference does using the predictive influence make as opposed to the reconstruction error. In this section we show that the predictive influence used by our PSC algorithm displays improved discrimination between observations from different subspaces when the number of dimensions, R increases. This is contrary to the behaviour of K-subspaces which becomes less discriminatory as R increases. Furthermore, this behaviour ensures that PSC is suitable for solving the subspace clustering problem when the number

of dimensions in each subspace is different.

Initially, we can see a clear connection with K-subspaces if we examine the simplest case of $R = 1$. The K-subspaces algorithm minimises the sum of squared reconstruction errors

$$C_{KSS} = \sum_{k=1}^{K} \sum_{i \in \mathcal{C}_k} \|\boldsymbol{x}_i - \boldsymbol{x}_i \boldsymbol{v}\boldsymbol{v}^\top\|^2.$$

By contrast, PSC minimises the sum of squared predictive influence within each cluster where the predictive influence exerted by a point $\boldsymbol{\pi}_{\boldsymbol{x}_i}(\boldsymbol{v})$ has the following closed-form expression

$$\|\boldsymbol{\pi}_{\boldsymbol{x}_i}(\boldsymbol{v})\|^2 = \frac{\|\boldsymbol{e}_i\|^2}{(1-h_i)^2}.$$

It can be seen that this is simply the PCA reconstruction error used by K-subspaces, scaled by the square of its leverage $\frac{\boldsymbol{x}_i}{(1-h_i)^2}$. The differences between PSC and K-subspaces become more apparent when $R > 1$. In the case when $R = 2$, the K-subspaces distance between two points when $R = 2$ is

$$C_{KSS}(i) = \|\boldsymbol{x}_i \left(\boldsymbol{I}_p - \sum_{r=1}^{2} \boldsymbol{v}^{(r)} {\boldsymbol{v}^{(r)}}^\top \right) \|^2.$$

The closed form solution for the predictive influence of a point, $\boldsymbol{x}_i$ is given by

$$\begin{aligned}
\|\boldsymbol{\pi}_{\boldsymbol{x}_i}(\boldsymbol{V})\|^2 &= \| \left(\frac{\boldsymbol{x}_i(\boldsymbol{I}_p - \boldsymbol{v}^{(1)}{\boldsymbol{v}^{(1)}}^\top)}{1-h_i^{(1)}} + \frac{\boldsymbol{x}_i(\boldsymbol{I}_p - \boldsymbol{v}^{(2)}{\boldsymbol{v}^{(2)}}^\top)}{1-h_i^{(2)}} \right) \|^2 \\
&= \frac{\|\boldsymbol{e}_i^{(1)}\|^2}{(1-h_i^{(1)})^2} + \frac{\|\boldsymbol{e}_i^{(2)}\|^2}{(1-h_i^{(2)})^2} + \frac{2\|\boldsymbol{x}_i \left(\boldsymbol{I}_p - \sum_{r=1}^{2} \boldsymbol{v}^{(r)} {\boldsymbol{v}^{(r)}}^\top \right) \|^2}{(1-h_i^{(1)})(1-h_i^{(2)})}.
\end{aligned}$$

It can be seen that the final term in the predictive influence is again equivalent to a scaled version of the K-subspaces objective function. However, the predictive influence also includes two other terms which are the distance between the projection of the point onto the principal eigenvector, $\boldsymbol{v}^{(1)}$ and the distance between the projection of the point onto the second eigenvector, $\boldsymbol{v}^{(2)}$.

As the dimensionality of the subspaces, R increases, the PCA residual decreases

monotonically since

$$\boldsymbol{x}_i(\boldsymbol{I}_p - \sum_{r=1}^{R} \boldsymbol{v}^{(r)}\boldsymbol{v}^{(r)})^\top \geq \boldsymbol{x}_i(\boldsymbol{I}_p - \sum_{r=1}^{R+1} \boldsymbol{v}^{(r)}\boldsymbol{v}^{(r)})^\top.$$

Therefore, if we increase the number of dimensions beyond $R = 1$, the distance between points becomes less discriminative. In contrast, as the components of the PSC objective function are additive, the PSC distance between two points increases as more dimensions are added, since

$$\boldsymbol{x}_i(\boldsymbol{I}_p - \boldsymbol{v}^{(r)}\boldsymbol{v}^{(r)\top}) < \boldsymbol{x}_i(\boldsymbol{I}_p - \boldsymbol{v}^{(r+1)}\boldsymbol{v}^{(r+1)\top}).$$

This implies that as the number of dimensions is increased, the predictive influence function becomes more discriminative between different clusters. Hence, the use of the predictive influence effectively imposes a penalty on higher-dimensional clusters unless they are supported by the data and therefore resolves the problem of overfitting.

4.4 Penalised PSC

In Section 2.5.1 we briefly mentioned a class of subspace clustering methods which seek to find subsets of important variables within each cluster which determine the cluster assignments. Recently, it has been observed that sometimes only certain dimensions of the data are important for determining the separation between clusters [103]. In such cases, if all variables are used for clustering, the results may be affected by the unimportant variables. It is therefore desirable to be able to identify and remove these other variables, which could also be considered as noise, from playing a role in clustering. In this section we extend the PSC algorithm to the problem of variable selection in clustering.

PCA estimates the subspaces by taking linear combinations of all P variables in $\boldsymbol{X}$. However, in high dimensional problems, some of these variables may be noisy. Furthermore in our subspace clustering application, these noisy or unimpor-

tant dimensions may also be unimportant for discriminating between clusters. In this situation, it may improve subspace estimation (and therefore clustering) if we constrain the solution such that only "important" dimensions are considered in the estimation of each subspace.

Each cluster lies on a different subspace which may consist of different important dimensions. The eigenvectors obtained by PCA can be considered as a weight vector in that they determine the importance of each variable in the subspace. By imposing a penalty on the eigenvectors obtained in each cluster, we can arrive at a framework for feature selection in subspace clustering. It should be stressed that this approach to "sparse" clustering is not the same as the compressive sensing approach of SSC, described in section 2.5.2. Instead of estimating subspaces which are sparse in the observations as SSC, we consider estimating subspaces which are sparse in the variables which until now has not been widely explored in the subspace clustering literature.

Returning to the PSC objective function in Eq. (4.1), we first consider only one-dimensional subspaces and so the sparse predictive subspace clustering problem now consists of estimating K clusters and for each cluster, a sparse subspace. This penalised PSC (PPSC) problem can be expressed in the following objective function

$$\begin{aligned} &\min_{\{\mathcal{C}_1,\ldots,\mathcal{C}_K\},\{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_K\}} C \\ &\text{subject to } ||\boldsymbol{v}_k||_1 \leq \gamma_k \text{ for } k = 1,\ldots,K \end{aligned} \tag{4.9}$$

where the parameter which controls the level of sparsity, γ_k, could be different for each cluster. It can be seen that there are K inequality constraints, one for each of the subspaces.

1. Given K penalised PCA models with parameters, $\{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_K\}$ and keeping these fixed, recover the cluster assignments which solve

$$\min_{\{\mathcal{C}_1,\ldots,\mathcal{C}_K\}} C. \tag{4.10}$$

2. Given a set of cluster assignments, $\{\mathcal{C}_1, ..., \mathcal{C}_K\}$ and keeping these fixed, estimate the parameters of the K penalised PCA models which solve

$$\min_{\{\boldsymbol{v}_1,...,\boldsymbol{v}_K\}} C, \quad \text{subject to } \|\boldsymbol{v}_k\|_1 \leq \gamma_k \text{ for } k = 1, \ldots, K \tag{4.11}$$

The PPSC algorithm proceeds in a similar way to the standard PSC algorithm. We solve (4.10) in exactly the same way as in standard PSC by assigning points to clusters such that predictive influence in each cluster is minimised. For a given set of sparsity constraints, solving (4.11) with respect to a cluster, $\mathcal{C}_k$ amounts to estimating a *sparse* PCA model using the method described in Section 2.3. This implies convergence for PPSC is guaranteed following the same arguments as in Section 4.2.1. The one-dimensional PPSC method can easily be extended to find R_k sparse components in each cluster. This introduces $\sum_{k=1}^{K} R_k$ inequality constraints which can be solved sequentially in each cluster using a standard sparse PCA algorithm.

Since PPSC estimates subspaces using only relevant subsets of the available dimensions, this approach can be seen as a combination of the "subset" clustering approach, described in section 2.5.1, and linear subspace clustering. The additional parameters, $\{\gamma_k\}_1^K$ determine the degree of sparsity in each cluster and so selecting appropriate values is important. This introduces a further issue of model selection which we discuss in Section 4.7. In the following section we present simulation results for both the standard and penalised PSC algorithms.

4.5 Simulations

We first present simple simulated examples to illustrate the type of clusters that can be detected by the proposed PSC algorithm. We generate clusters of 100 data points which are distributed uniformly on a one, two or three-dimensional linear subspace embedded in three-dimensional space. To define each subspace, we generate a set of R_k orthonormal basis vectors each of dimension $P = 3$, where each element is

sampled from a standard normal distribution. For each cluster we then sample 100 R_k-dimensional points from a uniform distribution which are then projected onto its corresponding subspace. In order to ensure our comparison of methods is fair, the method we use to generate simulated data is based on the standard GPCA test data generation software, part of the GPCA Matlab code[1] , which is commonly used to evaluate subspace clustering algorithms [19, 108].

In Figure 4.0 we consider four simulation scenarios which consist of points which lie on:

(a) two straight lines.

(b) a straight line and a 2-D plane.

(c) two 2-D planes.

(d) a straight line, a 2-D plane and a 3-D sphere.

In each of these cases, we show the original data points in P dimensions, the clustering assignments using K-means clustering in the original dimensions, and the clustering assignment using PSC. It can be noted that the subspaces intersect so points belonging to different clusters may lie close to each other. We apply the K-means algorithm, which uses the Euclidean distance between points, directly to the 3-D data and as expected it consistently fails to recover the true clusters. On the other hand, PSC correctly recovers both the true clusters and the intrinsic dimensionality of the subspaces.

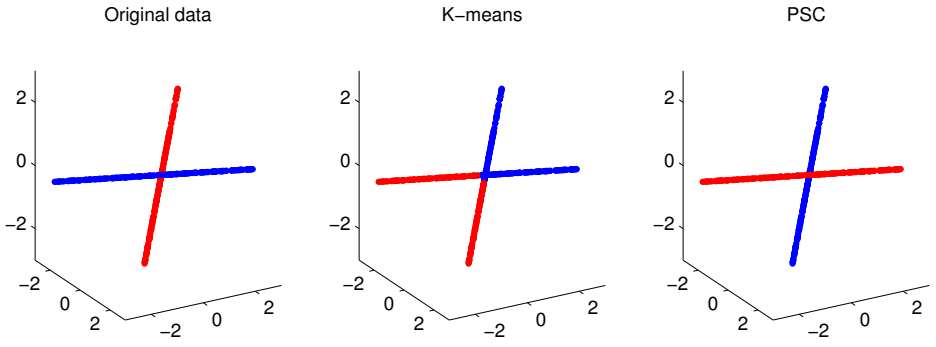
We also consider an additional scenario, (e), where $P = 200$ and $K = 4$. Here, the clusters consist of points which lie uniformly on a 5-D hyperplane, 4-D hypersphere and two lines generated as before. We then perform a comparison in the five settings described above over 200 Monte Carlo iterations between PSC and four state-of-the-art methods using available Matlab code; GPCA, SCC[2], SSC[3]

[1] `http://perception.csl.uiuc.edu/gpca`

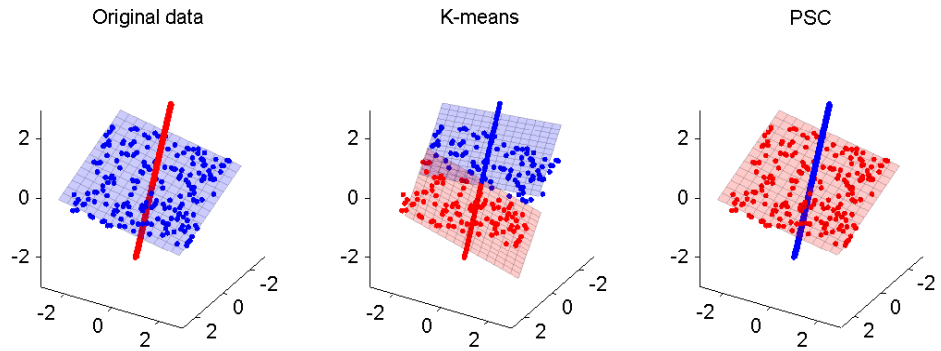
[2] `http://math.umn.edu/~lerman/scc`

[3] `http://www.cis.jhu.edu/~ehsan/ssc.html`

and SLBF[4]. Table 4.3 reports on both the mean percentage of incorrectly clustered points using the Rand index and computation time (in seconds). In scenarios (a)-(c) all competing methods achieve close to perfect clustering accuracy except GPCA which performs poorly under the scenario (b) which represents a scenario where the intrinsic dimensionality of the clusters is different. All the competing methods perform poorly under scenario (d) where the clusters exist on three subspaces of different dimensions. This is expected since SSC, SLBF and SCC all assume that the number of dimensions is the same in each cluster. Even in low-dimensions, GPCA, SSC and SLBF require at least an order of magnitude more computation time compared to PSC. In the high-dimensional scenario (e), SLBF and SSC incur further computational cost as P, K and N increases. GPCA cannot be applied in such high-dimensional settings. Our PSC algorithm accurately recovers the clusters

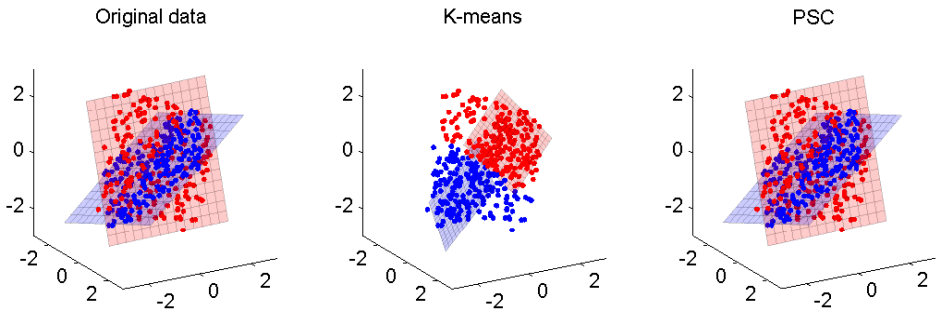


(a) Two lines

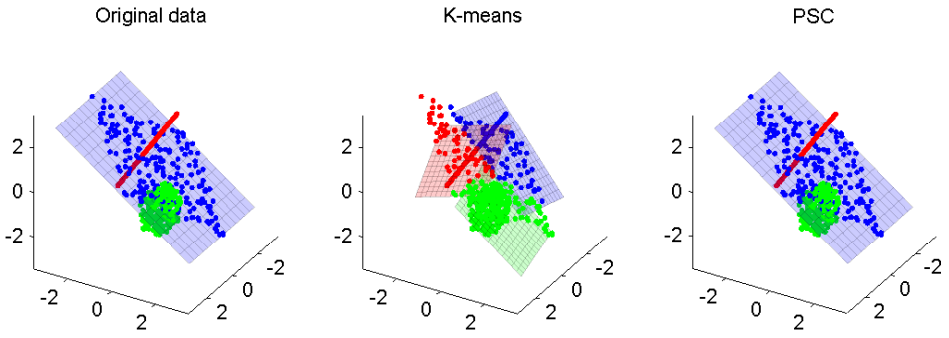


(b) Two subspaces: Line and plane

[4] http://www.math.umn.edu/~zhang620/lbf/



(c) Two planes



(d) Three subspaces: Line, plane and sphere

Figure 4.0: Example results of clustering data belonging to several different subspaces using K-means and PSC. The middle plots shows the results of clustering with K-means. the right-hand plots show that in these examples PSC consistently recovers the true cluster assignments and estimates the subspaces correctly.

in all settings with little computational cost. This is due to the ability to automatically learn the dimensionality of each subspace.

To test the penalised PSC method we use a similar simulation setting. We repeat the previously described simulations with two crucial differences. Firstly, we increase the dimensionality of each simulated dataset to $P = 200$. Secondly, in order to construct the low-dimensional subspaces, we generate *sparse* loading vectors where only ten, randomly chosen, variables out of 200 are non-zero. We add normally distributed noise with zero-mean and variance 0.4 so that the remaining 190 variables consist only of noise. Each of the $r = 1, \ldots, R_k$ loading vectors in each of the K clusters has the same number of non-zero elements however the

Setting		a	b	c	d	e
GPCA	e%	6.86	36.09	7.14	27.92	-
	t(s)	2.07	3.26	1.40	7.22	-
SCC	e%	0.00	0.00	0.00	29.49	33.05
	t(s)	0.49	0.51	0.53	0.80	3.76
SSC	e%	0.0	0.0	11.83	38.33	12.64
	t(s)	67.05	64.46	63.88	100.12	471.62
SLBF	e%	0.00	0.00	0.00	44.70	15.28
	t(s)	4.28	5.38	5.60	11.48	104.52
PSC	e%	0.00	0.03	0.12	4.09	0.78
	t(s)	0.43	0.46	0.58	1.17	24.23

Table 4.1: Comparison of clustering error (e%) and computational time in seconds, t(s), between PSC and four other state-of-the-art methods for simulated data. It can be seen that PSC achieves the smallest clustering error in the two hardest settings, d and e where the dimensionality of the subspaces is different. PSC manages to achieve these results at a relatively low computational cost.

sparsity pattern in each is different. Data simulated in this way tests the ability of the PPSC algorithm to estimate subspace parameters and simultaneously recover cluster assignments when there is a large number of noisy, irrelevant variables.

In this task, we compare PPSC with SCC, SLBF, SSC and standard PSC. We do not compare with GPCA since the dimensionality of the simulated data is too large. Importantly, we assume to know the true level of sparsity in the problem and we provide this information to the PPSC algorithm. For each algorithm, we supply the true number of clusters, K.

Table 4.2 shows the results using the sparse dataset. It can be seen that all methods achieve a larger clustering error in the sparse setting compared to the standard setting in Table 4.1. PPSC obtains the lowest error out of all methods in all scenarios apart from (a). SLBF performs comparably with PPSC in settings $(a) - (c)$ however it performs noticibly worse as the number of subspaces with mixed dimensions increases. Furthermore, in high-dimensional settings, SLBF incurs a similar high computational cost to SSC. It should also be noted that SLBF does not esti-

mate the sparse subspace parameters and so does not identify which variables are important for estimating the subspaces. This can make interpretation of the results difficult. SSC achieves the largest clustering error in every scenario which further highlights the difference with our penalised approach which induces sparsity in the variables instead of the observations.

This simulation setting highlights an important limitation of standard PSC which performs worse than SCC and SLBF. This is due to PSC estimating the subspace parameters using standard PCA which takes a linear combination of all of the variables. This causes all variables, including the noise variables, to contribute to the estimated subspace parameters.

As we expect, the performance of PPSC degrades as the dimensionality of the subspaces increases. This is due to the constraint that basis vectors of each subspace must be mutually orthonormal. Therefore, if the incorrect sparsity pattern is estimated in the first loading, all subsequent loadings are also likely to be estimated incorrectly. However, since PPSC still performs better than all other algorithms in settings (d) and (e), this further highlights the benefit of estimating the underlying subspaces using only the truly important variables.

4.6 Applications to computer vision

In this section we present a comparison of results between PSC and some state-of-the-art methods for subspace clustering on benchmark datasets from real applications in computer vision.

4.6.1 Yale faces B database

We present clustering results using the Yale faces B dataset [33], introduced in Section 3.4 and shown in Figures 3.2 and 3.3.

The Yale faces dataset has dimensionality, $\boldsymbol{X} \in \mathbb{R}^{64K \times 19200}$ which is too large for most competing subspace clustering algorithms to deal with. Therefore, following the established procedure of [108] we first use a global PCA to reduce the

Setting		a	b	c	d	e
SCC	e%	31.95	16.19	10.48	27.76	21.80
	t(s)	0.34	0.70	0.70	1.97	6.44
SSC	e%	46.63	47.60	46.78	61.92	63.13
	t(s)	138.85	142.16	142.17	248.63	400.53
SLBF	e%	12.44	7.50	3.10	35.69	56.78
	t(s)	99.11	114.72	107.93	231.59	389.90
PSC	e%	31.75	46.69	46.13	41.75	36.59
	t(s)	5.12	6.12	5.64	19.94	35.42
PPSC	e%	14.16	6.95	2.96	14.30	14.06
	t(s)	5.38	3.85	4.26	16.53	22.72

Table 4.2: Comparison of clustering error and computational time in seconds between PPSC and competing methods for sparse data. PPSC achieves the smallest clustering error in all but the first setting at a low computational cost.

dimensionality of the data to $P = 5$ for GPCA and $P = 20$ for all other methods including PSC. We then construct standard subsets of the dataset comprising of a varying number of clusters from 2 to 10 using the following individuals: [5, 8], [1, 5, 8], [1, 5, 8, 10], [1, 4, 5, 8, 10], [1, 2, 4, 5, 8, 10], [1, 2, 4, 5, 7, 8, 10], [1, 2, 4, 5, 7, 8, 9, 10] and [1, 2, 3, 4, 5, 7, 8, 9, 10] as mentioned in [94]. We again compare the clustering performance of PSC to GPCA, SCC, SSC and SLBF. We use PSC without pre-specifying K and R_k, but for all the competing methods all these parameters are fixed to be the true value of K, and the dimensionality of each subspace is set as $R = 2$. The ability to accurately and quickly cluster faces has implications in commercial implementations of facial recognition for biometric and security systems as well for entertainment purposes.

Table 4.3 compares the mean clustering error and running time for each algorithm for the settings $K = 2, \ldots, 10$. It can be seen that PSC achieves perfect clustering accuracy with all subsets in less time than the other state-of-the-art methods. Furthermore, for all values of K, PSC was able to correctly determine the true number of clusters using the PRESS. On this dataset, it is clear that PSC exhibits

state-of-the-art performance and computational speed.

Of the competing methods, SSC and SLBF achieve the next best clustering accuracy. However, each of these algorithms requires an order of magnitude more computational time than PSC. GPCA and SCC both perform well when there are fewer clusters, however as the number of clusters increases, they exhibit poorer clustering accuracy.

4.6.2 Hopkins 155 motion segmentation database

The Hopkins 155 database [90] consists of a collection of video sequences which record instances of several objects moving in a scene. Each scene consists of at most two independently moving objects. However, due to the movement of the camera, the background also moves relative to the observer. So at most there are three objects moving through the scene. For each of the important objects, several feature points are identified and tracked so that their position in each frame is recorded. The position of the point as it moves through the scene is referred to as its trajectory.

Figure 4.1 gives an example of a single frame from each of the three types of scene with three motions. Figure 4.1a shows an example of a "checker" sequence where three different objects (marked with coloured points) rotate and translate through the scene. Each of the coloured points represents a tracking point on an object. In these sequences, the objects move independently of each other and so the motion trajectories of each object is expected to lie on independent subspaces of dimension three. Figure 4.1b shows a "traffic" sequence filmed in uncontrolled conditions outdoors. Again the subspaces are independent, this time of dimension two. Figure 4.1c shows one of the "other" sequences which consists of objects translating, rotating and being carried through the scene. These objects do not move independently, and so lie on dependent subspaces which causes difficulties for most subspace clustering algorithms since the clusters are less well separated.

In this problem, there are N feature points tracked across $f = 1, \ldots, F$ frames. In frame f, the i^{th} feature point $\boldsymbol{a}_{fi} = \begin{bmatrix} a_{fi}^{(1)} \\ a_{fi}^{(2)} \end{bmatrix} \in \mathbb{R}^2$ is the two-dimensional (2-D) projection of a point in 3-D space. We concatenate the two dimensions of each

K		2	3	4	5	6	4	8	9	10
GPCA	e%	0.0	49.5	0.0	26.6	9.9	25.2	28.5	30.6	19.8
	t(s)	1.42	2.72	4.91	8.08	11.71	33.11	99.49	286.36	1122.50
SCC	e%	0.0	0.0	0.0	1.1	2.7	2.1	2.2	5.7	6.6
	t(s)	0.57	0.92	1.45	2.30	2.27	4.57	6.58	10.29	7.51
SSC	e%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.4	4.6
	t(s)	36.56	56.21	80.87	107.82	137.83	174.81	219.22	276.81	570.57
SLBF	e%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.2	0.9
	t(s)	3.70	7.90	14.00	28.32	43.50	63.79	118.99	179.70	249.42
PSC	e%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	t(s)	2.55	2.68	2.47	2.81	5.92	5.37	17.10	23.75	19.45

Table 4.3: Mean clustering error and computation time for Yale faces B dataset. PSC quickly achieves perfect clustering accuracy in all settings. The next best competing methods, SLBF and SSC require an order of magnitude longer to achieve similar results.

feature point in each frame such that its dimensionality is $P = 2F$. Now, each vector $\boldsymbol{x}_i = [a_{1i}^{(1)}, \ldots, a_{Fi}^{(1)}, a_{1i}^{(2)}, \ldots, a_{Fi}^{(2)}] \in \mathbb{R}^{1\times 2F}$ represents the trajectory of the i^{th} feature point as it moves in two-dimensions through each frame of the sequence. The task in motion segmentation is to cluster the trajectories into K groups such that each group represents the motion of a single object in the sequence.

We again compare against GPCA, SCC, SSC and SLBF using the standard, published results in [94]. Furthermore, we also compare against two "reference" algorithms. The standard reference algorithm (REF) [90] is the result of fitting standard PCA models to the ground truth cluster assignments and then applying a single step of the K-subspaces algorithm in order to assign points to the closest subspace. This tests the validity of the basic assumption that each of the trajectories can be well represented by a low-dimensional subspace. In a sense, the REF algorithm represents the best result possible using K-subspaces. In practise, this algorithm cannot be used since it requires knowledge of the ground truth assignments. We also define a predictive reference algorithm (P-REF) which is the result of fitting standard PCA models as in the REF algorithm, however we now assign points based on a single application of Step **P** of the PSC algorithm. This tests the suitability of using the predictive influence to assign points to clusters.

It is important to note that for GPCA, the data was first projected onto five dimensions using PCA. For SSC, each dataset was projected onto $4K$ dimensions. For SLBF, SCC and PSC all $2F$ dimensions were used for every dataset. It should be noted that in this work we only compare PSC with the state-of-the-art in motion segmentation algorithms. Comprehensive comparisons of methods on this dataset are presented in [94] and at the Hopkins155 website[5]. We apply the PSC algorithm by specifying the true value of K but we allow the algorithm to learn the values of $\{R_k\}_1^K$.

Table 4.4 reports on the mean and median percentage of points which are assigned to the wrong clusters for sequences with two motions. Firstly we note that the standard reference algorithm obtains small segmentation error in all sequences.

[5] `http://www.vision.jhu.edu/data/hopkins155`

This suggests that the clusters do lie on low-dimensional subspaces. SSC and SLBF achieve extremely small errors on all sequences. PSC performs better than GPCA and comparably to SCC for all sequences. It can be noted that PSC consistently obtains a smaller median clustering error than SCC. The P-REF performs worse than the standard REF for all but the "other" sequences.

Table 4.5 reports on the mean and median percentage of points which are assigned to the wrong clusters for sequences with three motions. Again SSC and SLBF obtain excellent results on all sequences. However, for sequences with three motions, PSC displays a relative improvement in clustering accuracy compared with the other methods and performs better than SCC in all sequences. PSC performs particularly well on the "other" sequences, which comprise of dependent subspaces, due to its ability to learn the dimensionality of the different subspaces. In the three motion setting, the P-REF also achieves a better accuracy than the standard REF for all sequences. This suggests that using the predictive influence is beneficial as the number of clusters in the dataset increases.

Figure 4.2 shows the distribution of clustering errors obtained by the PSC algorithm compared to the reference using predictive influence. Figure 4.2a shows this distribution for all sequences with two motions. It can be seen that 80% of sequences are clustered perfectly however the error distribution of the PSC algorithm exhibits heavy tails with a few sequences clustered with up to 40% error. This account for the larger mean but small median error values. The P-REF exhibits somewhat thinner tails with most of the sequences being clustered with smaller errors. However overall, PSC and P-REF perform similarly which suggests that PSC often finds the globally optimal solution. Figure 4.2b shows the distribution for sequences with three motions. Here we see that although only 70% of sequences are clustered perfectly, the maximum clustering error is 30%. This further suggests that PSC performs well in situations where there are more clusters.

	REF	GPCA	SCC	SSC	SLBF	PSC	P-REF
Checker							
Mean	2.76%	6.09%	1.31%	1.12%	1.59%	5.14%	2.98%
Median	0.49%	1.03%	0.06%	0.00%	0.00%	0.00%	0.00%
Traffic							
Mean	0.30%	1.41%	1.02%	0.02%	0.20%	1.12%	1.53%
Median	0.00%	0.00%	0.26%	0.00%	0.00%	0.00%	0.00%
Other							
Mean	1.71%	2.88%	3.21%	0.62%	0.80%	1.15%	1.66%
Median	0.00%	0.00%	0.76%	0.00%	0.00%	0.19%	0.00%
All							
Mean	2.03%	4.59%	1.41%	0.82%	1.16%	3.53%	2.47%
Median	0.00%	0.38%	0.10%	0.00%	0.00%	0.00%	0.00%

Table 4.4: Mean and median clustering errors for sequences with two motions in the Hopkins 155 data set. SSC and SLBF perform best overall. PSC exhibits comparable mean clustering error with SCC however achieves a better median error suggesting most sequences are clustered perfectly but a few achieve a relatively large error.

	REF	GPCA	SCC	SSC	SLBF	PSC	P-REF
Checker							
Mean	6.28%	31.95%	6.31%	2.97%	4.57%	6.02%	4.27%
Median	5.06%	32.93%	1.97%	0.27%	0.94%	0.74%	0.24%
Traffic							
Mean	1.30%	19.83%	3.31%	0.58%	0.38%	1.10%	0.34%
Median	0.00%	19.55%	3.31%	0.00%	0.00%	0.00%	0.00%
Other							
Mean	2.66%	16.85%	9.58%	1.42%	2.66%	0.48%	0.93%
Median	2.66%	28.66%	9.58%	0.00%	2.66%	0.48%	0.93%
All							
Mean	5.08%	28.66%	5.90%	2.45%	3.63%	4.82%	3.57%
Median	2.40%	28.26%	1.99%	0.20%	0.64%	0.27%	0.00%

Table 4.5: Mean and median clustering errors for sequences with three motions in the Hopkins 155 data set. SSC and SLBF perform best overall. However PSC exhibits comparable results in most sequences and achieves the lowest mean error in the "other" sequences. PSC consistently outperforms GPCA, SSC and even the standard reference algorithm, REF.

4.7 Discussion

In the previous chapters we have considered the related problems of model selection and influential observations in the context of PCA. We have developed a simple, unified framework to perform both tasks based on an accurate, analytic approximation for the leave-one-out predicted reconstruction error. Furthermore we have proposed a solution to the open question of dealing with influential observations within the framework of PCA. Our predictive influence measure is effective at detecting influential observations which are otherwise undetectable using the PCA residual error. As a result of minimising the predictive influence as opposed to the residual error, we obtain a robust alternative to PCA, PRoPCA which is able to incorporate influential observations and downgrade their effect so the resulting PRoPCA model is not heavily biased towards these observations. We demonstrated the ability of the predictive influence using a series of visual experiments on the widely used Yale faces benchmark dataset. The results obtained show that the predictive influence is superior to the PCA residual error for identifying influential observations. We compared PRoPCA with another robust PCA method, ROBPCA, and showed that although ROBPCA achieves a smaller reconstruction error when presented with influential training observations, PRoPCA achieves better qualitative results. Our framework for identifying influential observations has immediately obvious applications in facial recognition and biometrics amongst others.

The main application of these methods however, is to consider how PCA can be extended to solve highly non-linear problems. We apply these methods to the problem of linear subspace clustering, reviewed in Section 2.5.2. We considered a novel approach to clustering which identifies misclustered observations as those which exert a large influence within a cluster. We proposed an algorithm which assigns observations to clusters such that the within clusters sum of predictive influences is minimised. Since this procedure is based on out-of-sample prediction, it overcomes problems relating to overfitting and model selection.

The resulting PSC algorithm exhibits state of the art results on simulated data as well as real applications in computer vision. Interestingly, it is typically in the more

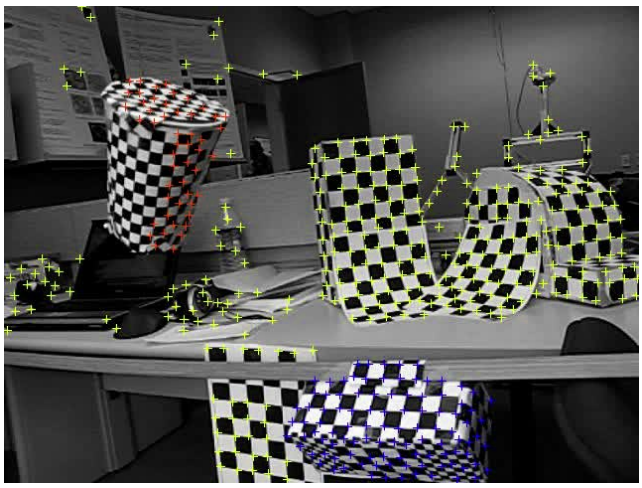
challenging circumstances (large K, varying R_k, large P, dependent subspaces) where PSC exhibits the best performance relative to competing methods. For difficult simulation settings involving subspaces of mixed dimensions, PSC outperforms all methods at low computational cost. When we apply PSC to the problem of clustering images in the Yale faces B dataset, we see that PSC achieves perfect clustering for all subsets of the data comprising of number of clusters, $K = 2, \ldots, 10$. Again, PSC achieves this result in an order of magnitude less computational time than the next best performing methods, SLBF and SSC. Finally, on the Hopkins155 dataset, we observe that PSC achieves results which are competitive with the state-of-the-art. In particular, in difficult settings where the subspaces are dependent, PSC achieves the best performance due to its ability to correctly estimate the dimensionality in each subspace. We also show that using a predictive influence based clustering criterion accurately assigns points to clusters.

We also proposed an extension to PSC, penalised PSC, which performs simultaneous subspace clustering and variable selection using sparse PCA. PPSC is able to accurately recover cluster assignments in simulated data where the data is noisy with many irrelevant variables.

There are a number of open questions for further research within the PSC framework. The most obvious concerns how to perform model selection in PPSC. The standard PSC algorithm is able to learn K and $\{R_k\}_1^K$ efficiently and accurately using the PRESS. However PPSC adds the additional parameters $\{\gamma_k\}_1^K$ which control the number of important variables in each cluster. It has been observed that prediction based methods such as the PRESS do not perform well for selecting the sparsity parameter. Recently, subset resampling methods such as *stability selection* [63] have shown promising results for accurately selecting regularisation parameters. However, implementing such a method within the PPSC framework would be computationally prohibitive. Furthermore, we are faced with the added combinatorial problem of selecting $\{R_k\}_1^K$ and $\{\gamma_k\}_1^K$ at each iteration where the cluster assignments themselves also depend on these parameters. This framework could be extended further still to allow for a different sparsity parameter for each component within each cluster. It is clear that a different optmisation strategy would need to be

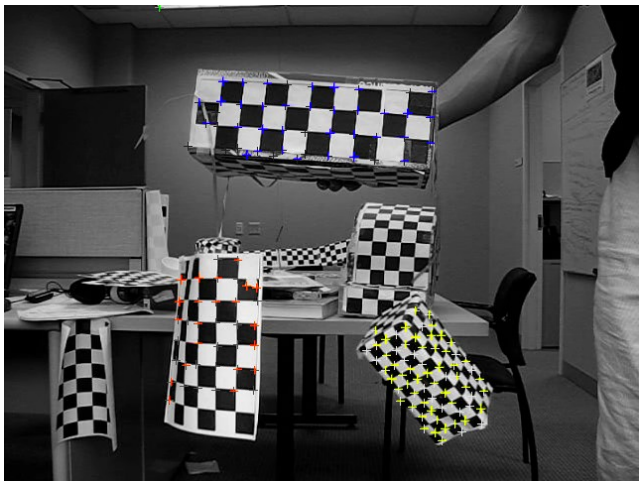
investigated.

Currently, we have only considered penalising the ℓ_1 norm of the PCA parameter. However a number of different penalties exist which have benefits in other application domains [31]. Furthermore a similar framework could also be used to obtain non-negative parameters which provide a better model in some computer vision [81] and bioinformatics [102] applications. We revisit the topic of penalisation and sparsity in Chapter 6.



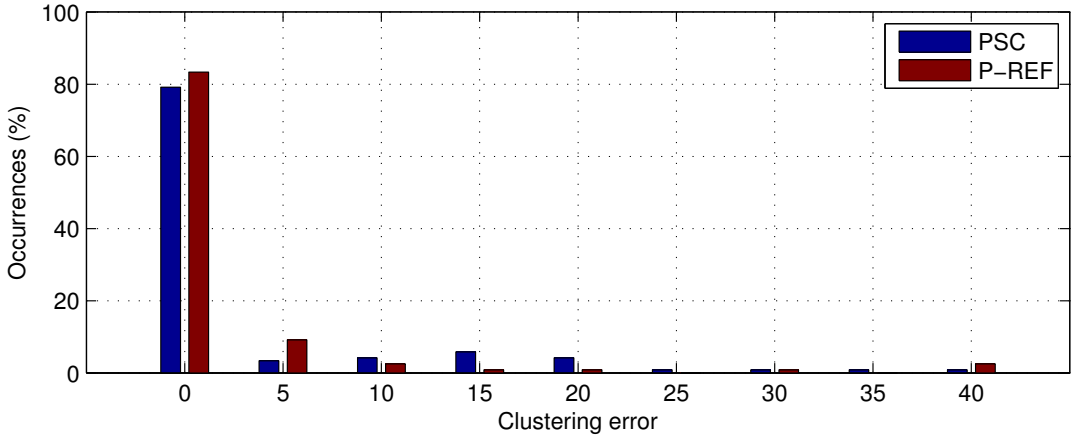
(a) Checker.

(b) Traffic.

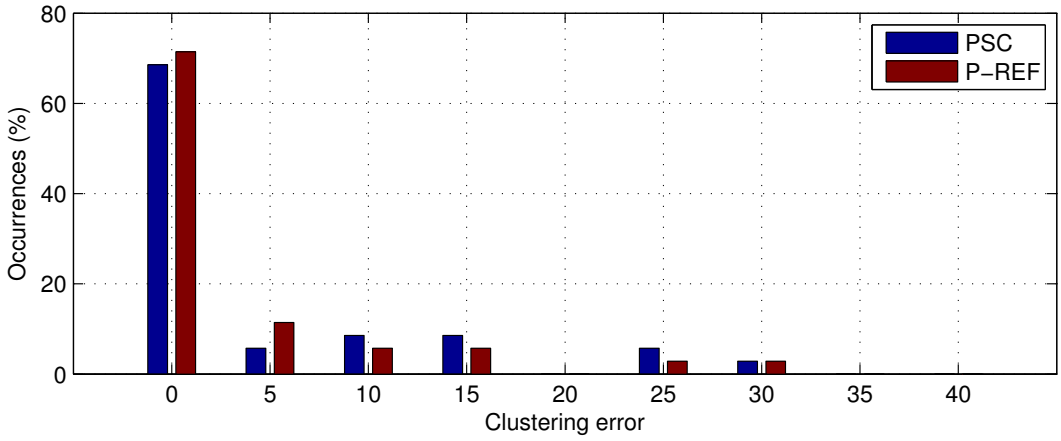


(c) Other.

Figure 4.1: Single frames from the Hopkins155 dataset showing examples of (a) "checker", (b) "traffic" and (c) "other" sequences. The coloured crosses represent the tracking points of each object whose trajectories we aim to cluster.



(a) Two motions.



(b) Three motions.

Figure 4.2: Comparison of the distribution of clustering errors obtained using PSC and the reference (P-REF) on the Hopkins155 motion segmentation dataset for (a) two motions and (b) three motions. In both settings, PSC achieves perfect clustering in most of the sequences and achieves a similar error distribution to P-REF. This suggests that the PSC algorithm often finds the optimal solution.

Chapter 5

Multi-view predictive modelling

In the previous chapters we considered the problem of performing unsupervised clustering in high-dimensions with a single view of data. In the following chapters, we consider the supervised, or multi-view, setting where we observe a random sample of N independent and identically distributed data points, $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}$, for $i = 1, \ldots, N$, where each $\boldsymbol{x}_i \in \mathbb{R}^{1\times P}$ is the "explanatory view" and each $\boldsymbol{y}_i \in \mathbb{R}^{1\times Q}$ is the "response view" observed on the i^{th} sample. The dimensions of both views, P and Q, are allowed to be very large. The N observations can then be arranged in two paired matrices, one containing all the explanatory variables observed in the samples, $\boldsymbol{X} \in \mathbb{R}^{N\times P}$, and one containing the corresponding response variables, $\boldsymbol{Y} \in \mathbb{R}^{N\times Q}$. The variables in both views are centred and scaled so as to have zero mean and unit variance.

When multiple views of data are available, the task of identifying clusters is less well defined. There may be reasons to believe that the observations should cluster in the same way under each of the available views, hence the process of inferring the true clustering can be improved by making joint use of all views.

Multi-view clustering methods have become particularly popular for a broad class of problems in web mining where data of many different types may co-occur on the same page, for example text, hyperlinks, images and video. A common application is that of clustering web pages [10]. Pages can be clustered together based on the similarity of both their text content and link structure, and the clusters iden-

tify broad subject categories. Another problem in the web mining domain involves clustering and annotating images represented as a vector of pixel intensities as well as bag of words containing the corresponding textual annotation [16]. The ability to accurately cluster these data have implications in web search and advertising.

We extend the ideas in Chapter 3 to develop an efficient PRESS statistic for TB-PLS as well as a measure of predictive influence. The resulting predictive, multi-view clustering algorithm can then be seen as an extension of the predictive subspace clustering method of Chapter 4 to the multi-view setting.

5.1 Learning in multiple views

5.1.1 High-dimensional multi-response regression

In this section we consider the problem of fitting a regression model when both the predictors and the responses are high-dimensional. In such a setting, ordinary least squares (OLS) is unsuitable for two reasons. The first is the problem of multicollinearity mentioned in Section 2.1. Secondly, when the response is high dimensional the OLS solution is equivalent to fitting as many multivariate regression models as the number of responses [38]. This means that no information about the covariance structure of the response is used.

A number of methods have been proposed for simultaneous dimensionality reduction and predictive modelling in such settings. For instance, reduced-rank regression (RRR) imposes a rank restriction on the OLS regression coefficients and effectively identifies latent factors underlying the response which explain the correlation with the predictors [44]. The regression coefficients are then estimated by performing the least squares regression of the predictors on these latent factors. Other models related to RRR also consider linear combination of the response variables [15].

5.1.2 Two block partial least squares regression

Among latent factor models for simultaneous dimensionality reduction and prediction, two-block partial least squares (TB-PLS) regression has been shown to be a particularly useful method for modelling a linear predictive relationship between two high-dimensional views [100, 77]. Unlike RRR, TB-PLS performs dimensionality reduction in both predictor and response views simultaneously by assuming that both the predictors and the responses are comprised of a set of mutually orthogonal latent factors. The TB-PLS latent factors are estimated such that they maximise the covariance between the views. This model overcomes problems relating to multicollinearity by estimating least squares regression coefficients using the low dimensional latent factors, and has been widely used in the field of chemometrics [105]. Among other applications, the model has been successfully used in genomics, where regions of DNA that are highly predictive of gene expression need to be detected, [51], and in computational finance, where the returns of several indices have been predicted using a large basket of assets [61].

The TB-PLS regression model assumes that the predictor and response views are noisy realisations of linear combinations of hidden variables, or *latent factors*, that satisfy some optimal properties and must be inferred from the data. The specific form of the TB-PLS model is given in the following definition. The TB-PLS model assumes the existence of R pairs of orthogonal latent factors, $\boldsymbol{t}^{(r)}$ and $\boldsymbol{s}^{(r)} \in \mathbb{R}^{N\times 1}$, for $r = 1, \ldots, R$ such that

$$\begin{aligned} \boldsymbol{X} &= \sum_{r=1}^{R} \boldsymbol{t}^{(r)} {\boldsymbol{p}^{(r)}}^{\top} + \boldsymbol{E}_x, \\ \boldsymbol{Y} &= \sum_{r=1}^{R} \boldsymbol{s}^{(r)} {\boldsymbol{q}^{(r)}}^{\top} + \boldsymbol{E}_y, \end{aligned} \tag{5.1}$$

where $\boldsymbol{E}_x \in \mathbb{R}^{N\times P}$ and $\boldsymbol{E}_y \in \mathbb{R}^{N\times Q}$ are matrices of residuals. For each r, the latent factors are linear combinations of the P predictors and Q responses, respectively; that is, they are found to be $\boldsymbol{t}^{(r)} = \boldsymbol{X}\boldsymbol{u}^{(r)}$ and $\boldsymbol{s}^{(r)} = \boldsymbol{Y}\boldsymbol{v}^{(r)}$ where $\boldsymbol{u}^{(r)} \in \mathbb{R}^{P\times 1}$ and $\boldsymbol{v}^{(r)} \in \mathbb{R}^{Q\times 1}$ are weight vectors of unit length. Moreover, for each r, the vectors

$\boldsymbol{p}^{(r)} \in \mathbb{R}^{P\times 1}$ and $\boldsymbol{q}^{(r)} \in \mathbb{R}^{Q\times 1}$ are the *factor loadings*, that determine the contribution of the latent factors. For any given r, each pair of latent factors $\{\boldsymbol{t}^{(r)}, \boldsymbol{s}^{(r)}\}$ provides a one-dimensional representation of both views and is obtained by identifying the directions on which the projected views have maximal covariance. Therefore, for each r, the paired latent factors satisfy the property that

$$\text{Cov}(\boldsymbol{t}^{(r)}, \boldsymbol{s}^{(r)}) = \max_{\boldsymbol{u}^{(r)},\boldsymbol{v}^{(r)}} \text{Cov}(\boldsymbol{X}\boldsymbol{u}^{(r)}, \boldsymbol{Y}\boldsymbol{v}^{(r)})^2, \tag{5.2}$$

under the constraints that $\|\boldsymbol{u}^{(r)}\| = \|\boldsymbol{v}^{(r)}\| = 1$ for all $r = 1, \ldots, R$ which ensure the weights are not arbitrarily large. For $r = 1$ this optimisation problem is equivalent to

$$\lambda^{(1)} = {\boldsymbol{t}^{(1)}}^\top \boldsymbol{s}^{(1)} = \max_{\boldsymbol{v}^{(1)},\boldsymbol{u}^{(1)}} {\boldsymbol{u}^{(1)}}^\top \boldsymbol{X}^\top \boldsymbol{Y}\boldsymbol{v}^{(1)}. \tag{5.3}$$

under the same constraints posed on the weights. Here, $\lambda^{(1)}$ is the largest singular value of $\boldsymbol{X}^\top\boldsymbol{Y}$ and the weights are the corresponding left and right singular vectors. The R weight vectors that satisfy Eq. (5.2) can then be found by computing the singular value decomposition (SVD) of $\boldsymbol{X}^\top\boldsymbol{Y}$ as

$$\boldsymbol{X}^\top\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}^\top,$$

where $\boldsymbol{U} = [\boldsymbol{u}^{(1)}, ..., \boldsymbol{u}^{(P)}] \in \mathbb{R}^{P\times P}$ and $\boldsymbol{V} = [\boldsymbol{v}^{(1)}, ..., \boldsymbol{v}^{(Q)}] \in \mathbb{R}^{Q\times Q}$ are orthonormal matrices and $\boldsymbol{\Lambda} \in \mathbb{R}^{P\times Q}$ is a diagonal matrix.Therefore $\boldsymbol{u}^{(r)}$ and $\boldsymbol{v}^{(r)}$ are taken to be the r^{th} left and right singular vectors of $\boldsymbol{X}^\top\boldsymbol{Y}$, respectively. This can be seen as a multi-view extension of the PCA maximisation problem.

The predictive relationship between the two views is driven by a linear regression model involving the R pairs of latent factors. For each r, the response latent variable depends linearly on the explanatory latent variable, as follows

$$\boldsymbol{s}^{(r)} = \boldsymbol{t}^{(r)} g^{(r)} + \boldsymbol{h}^{(r)} \tag{5.4}$$

where each $g^{(r)}$ is a scalar regression coefficient which describes the projection of the latent factor relating to the response onto the latent factor relating to the

predictors, and each $\boldsymbol{h}^{(r)} \in \mathbb{R}^{N\times 1}$ is the vector of residual errors. Since the latent factors are assumed to have zero mean, there is no intercept term. Using the inner regression models (5.4), the TB-PLS model (5.1) can now be re-written in the more familiar form

$$\begin{aligned}\boldsymbol{Y} &= \sum_{r=1}^{R} \boldsymbol{t}^{(r)} g^{(r)} \boldsymbol{q}^{(r)\top} + \boldsymbol{E} = \boldsymbol{X} \sum_{r=1}^{R} \boldsymbol{u}^{(r)} g^{(r)} \boldsymbol{q}^{(r)\top} + \boldsymbol{E} \\ &= \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{E}, \end{aligned} \tag{5.5}$$

where the regression coefficients have been defined as

$$\boldsymbol{\beta} = \sum_{r=1}^{R} \boldsymbol{u}^{(r)} g^{(r)} \boldsymbol{q}^{(r)\top} \tag{5.6}$$

and depends on the parameter sets $\{\boldsymbol{u}^{(r)}, \boldsymbol{v}^{(r)}\}_1^R$. Each one of the R factor loadings $\boldsymbol{q}^{(r)}$ are obtained by performing univariate regressions,

$$\boldsymbol{q}^{(r)} = \frac{\boldsymbol{Y}^\top \boldsymbol{s}^{(r)}}{\boldsymbol{s}^{(r)\top} \boldsymbol{s}^{(r)}}. \tag{5.7}$$

and each of the R regression coefficients $g^{(r)}$, from the inner model of Eq. (5.4), is estimated by least squares regression of $\boldsymbol{t}^{(r)}$ on $\boldsymbol{s}^{(r)}$,

$$g^{(r)} = \left(\boldsymbol{t}^{(r)\top} \boldsymbol{t}^{(r)}\right)^{-1} \boldsymbol{t}^{(r)\top} \boldsymbol{s}^{(r)}. \tag{5.8}$$

Further insights about the regression coefficients that characterise the TB-PLS model can be obtained by explaining a special case. In high-dimensional settings, such as the one we consider, it is generally appropriate to assume the data has spherical covariance within each view [18, 102], and so $\boldsymbol{X}^\top \boldsymbol{X} = I_P$ and $\boldsymbol{Y}^\top \boldsymbol{Y} = I_Q$. With only one latent factor for each view, i.e. $R = 1$, the TB-PLS regression

coefficients, $\boldsymbol{\beta}$ have the following form

$$\begin{aligned}\boldsymbol{\beta} &= \boldsymbol{u}(\boldsymbol{t}^\top\boldsymbol{t})^{-1}\boldsymbol{t}^\top\boldsymbol{s}(\boldsymbol{s}^\top\boldsymbol{s})^{-1}\boldsymbol{v}^\top\boldsymbol{Y}^\top\boldsymbol{Y} \\ &= \boldsymbol{u}(\boldsymbol{t}^\top\boldsymbol{t})^{-1}\boldsymbol{t}^\top\boldsymbol{X}\boldsymbol{\beta}^{OLS}\boldsymbol{v}(\boldsymbol{s}^\top\boldsymbol{s})^{-1}\boldsymbol{s}^\top\boldsymbol{Y} \\ &= \boldsymbol{u}\boldsymbol{t}^\top\boldsymbol{X}\boldsymbol{\beta}^{OLS}\boldsymbol{v}\boldsymbol{v}^\top.\end{aligned}$$

where $\boldsymbol{\beta}^{OLS}$ are the OLS regression coefficients. This expression is obtained by recognising that $\boldsymbol{t}^\top\boldsymbol{s} = \boldsymbol{u}^\top\boldsymbol{X}^\top\boldsymbol{Y}\boldsymbol{v}$, and then replacing the term $\boldsymbol{X}^\top\boldsymbol{Y}$ with

$$\boldsymbol{X}^\top\boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{Y}.$$

The TB-PLS regression coefficients are thus expressed in terms of the OLS regression coefficients, and by premultiplying both sides by $\boldsymbol{X}$ we obtain the estimated response view,

$$\hat{\boldsymbol{Y}}^{PLS} = \boldsymbol{X}\boldsymbol{u}\boldsymbol{u}^\top\hat{\boldsymbol{\beta}}^{OLS}\boldsymbol{v}\boldsymbol{v}^\top, \tag{5.9}$$

where $\boldsymbol{X}\boldsymbol{u}\boldsymbol{u}^\top$ is the projection of $\boldsymbol{X}$ onto its TB-PLS directions. In this form the TB-PLS solution can be viewed in terms of obtaining the OLS regression using the low-dimensional representation of $\boldsymbol{X}$ and projecting those predicted responses onto the latent factors of $\boldsymbol{Y}$.

5.1.3 Multi-view clustering

The TB-PLS regression model rests on the assumption that the N independent samples are representative of a population in which the multivariate predictors and responses have been drawn from a joint probability distribution with a $(P \times Q)$ covariance matrix $\boldsymbol{X}^\top\boldsymbol{Y}$. Under this assumption, the latent factors that determine the regression coefficients in Eq. (5.6) can be optimally estimated using all the available data. However, in many applications the observations may be representative of a number of different populations, each one characterised by a different covariance structure between the views. Failing to recognise this would lead to a biased esti-

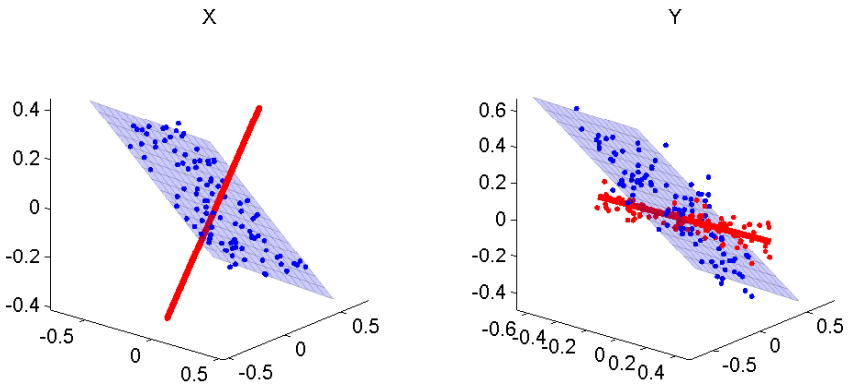
mation of the latent factors, which would in turn provide a sub-optimal predictive model.

We are interested in situations in which the observations have been sampled from K different sub-populations, where the exact value of K may be unknown. Data points belonging to a cluster $\mathcal{C}_k$, for $k = 1, \ldots, K$, have their own covariance structure, $\sum_{i \in \mathcal{C}_k} \boldsymbol{x}_i^\top \boldsymbol{y}_i$. If these cluster assignments were known, the optimal strategy would consist of fitting a separate TB-PLS model for each cluster, so that cluster $\mathcal{C}_k$ is associated with parameter set $\{\boldsymbol{u}_k, \boldsymbol{v}_k\}$ and can be referred to as a *predictive cluster*. It can be noted that in general the optimal dimension R_k is not necessarily the same across clusters. The problem involves simultaneously recovering the cluster assignments and their parameter sets, as well as learning the optimal K. Learning the optimal dimensionality in each cluster is a much harder problem which we address later.

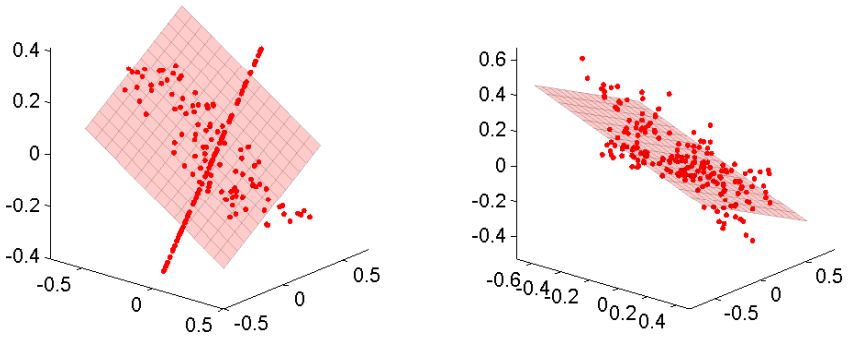
An example of this scenario is given in Figure 5.1a where $K = 2$, $R_1 = 2$, $R_2 = 1$ and $P = 3$. Here, under the $\boldsymbol{X}$ view, the points are uniformly distributed along either one of two lower dimensional subspaces, a line and a plane, both embedded in the three-dimensional space. To obtain the $\boldsymbol{Y}$ view, we have taken a linear combination of variables in the explanatory view and added some Gaussian noise; this ensures that, within each cluster, the $\boldsymbol{Y}$ view can be linearly predicted by the $\boldsymbol{X}$ view. Clearly, fitting a global TB-PLS model would be inappropriate in this setting, and this can be seen in Figure 5.1b, which shows the estimated subspaces. It can be noted that, within each view, these estimated subspaces lie somewhere between the true subspaces, so the predictive ability of the model is sub-optimal.

The example in Figure 5.1 highlights another difficulty which arises in modelling heterogeneous data: since the subspaces in each view intersect, points belonging to different clusters at the intersection may be geometrically closer than other points belonging to the same cluster. We revisit this example in Section 5.4.2 and show the results of our MVPP algorithm, for simultaneously clustering and estimating the subspaces, in Figure 5.4.

A number of multi-view clustering algorithms have been proposed to determine a common data partitioning between views. Two main approaches seem to have



(a) Simulated data.



(b) Global TB-PLS.

Figure 5.1: Figure 5.1a shows the two clusters in the $\boldsymbol{X}$ view consist of points sampled uniformly on a 1-d line and a 2-d plane embedded in three dimensions. The clusters in the $\boldsymbol{Y}$ view are noisy linear combinations of the corresponding clusters in the $\boldsymbol{X}$ view so that there is a predictive relationship between the views. Figure 5.1b shows the result of fitting a global TB-PLS model to the data. It can be seen that the resulting subspace in the $\boldsymbol{X}$ view lies between the clusters and as a result predicts the response poorly.

emerged in the literature: *late* and *early* fusion methods. The late fusion methods first recover the clusters independently from each view (e.g. by using the K-means algorithm), and then attempt to infer a "consensus" clustering by combining the partitioning obtained within each view such that some measure of disagreement between the individual partitionings is minimised [54, 50, 56, 37, 16]. Commonly,

the consensus clustering assignments are recovered by assuming the existence of latent factors that are shared between views, which are often inferred by estimating the non-negative factorisation of the combined partition matrices.

On the other hand, early fusion multi-view clustering methods start by learning any common patterns that may be shared by the views and that could yield a joint clustering [10, 92, 24]. A common assumption is that the data under each view are generated from a mixture of distributions where the mixing proportions which determine the cluster assignments are unknown but shared between views. For instance, multi-view EM algorithms have been proposed to fit mixture models in this setting [10, 92]. Several methods have been introduced which rely on a two-step approach to multi-view clustering where the clusters are ultimately defined in terms of geometric separation between points lying in low-dimensional projections. For example, multi-view CCA clustering (MV-CCA) [18] can be seen as an extension to the PCA/K-means procedure described in Section 2.5.2. First, a joint dimensionality reduction step is performed using both views, using canonical correlations analysis (CCA) which recovers latent factors explaining maximal correlation between the views; second, K-means is used to detect the clusters among data points in the lower dimensional space found in the first stage. Using CCA to initially perform dimensionality reduction has been shown to increase the separation between the clusters similarly to using PCA followed by K-means in single-view clustering. A non-linear dimensionality reduction step using kernel CCA has also been proposed [24].

Most of these multi-view clustering methods rely either explicitly or implicitly on the assumption that conditioned on the cluster assignments, the two views of data are independent. This means they do not take into account any possible predictive relationships between the two views. Indeed, the idea of multi-view clustering based on prediction has not been explored in the literature. Finite mixtures of linear regressions have been proposed to fit regression models to subsets of the data when the response is univariate [12, 83, 74]. This model assumes that the data are generated from K linear regressions where the parameters consist of the regression coefficients, the residual error and the mixing proportions which are estimated us-

ing the EM algorithm. However, as we discussed in Chapter 2, the least squares solution is prone to over-fitting and does not represent the true predictive relationship inherent between the views. Furthermore, the least squares regression applies only to a univariate response variable, and is not suitable for situations where the response is high dimensional.

Associative clustering [82] attempts to find clusters in separate views which maximise a measure of dependency between the views, in this case the mutual information. However, unlike most other multi-view clustering methods, this approach does not treat the samples in each view as paired observations and so may recover a different clustering in each view.

We propose a procedure similar to the PSC method of Chapter 4 which relies on the following observations. Given a fixed cluster allocation and model parameters, we assess whether, within each cluster, there are *influential* observations. An observation is influential if it has a large effect on the cluster-specific predictive model between views. Our strategy then consists of iterating between fitting TB-PLS models on the current data partitions and re-allocating influential observations to the most appropriate partitions until this process can be longer be improved. Similar to PSC, we first propose a computationally efficient measure of predictive influence under a TB-PLS model which is based on the PRESS statistic. We then exploit the predictive influence measure to establish an objective function for multi-view predictive partitioning.

5.2 Detecting influential observations

5.2.1 PRESS for TB-PLS

In the context of PLS regression with univariate response, the PRESS has also been used for identifying influential observations [104, 59]. However, as with the PCA PRESS, its computation requires the regression model to be fit N times, each time using $N - 1$ data points. Therefore, evaluating the PRESS for TB-PLS would require N SVD computations to be performed, each with a computational cost of

$O\left(P^2Q + Q^2P\right)$ [36]. Again, this approach is particularly expensive when the dimensions of the data in either view are large.

Recently, we proposed a closed-form expression for computing the PRESS statistic under a TB-PLS model which reduces the computational cost of explicitly evaluating the leave-one-out errors [60]. We overcome the need to recompute the SVD N times by approximating the leave-one-out estimates of the singular vectors $\{\boldsymbol{u}_{-i}, \boldsymbol{v}_{-i}\}$ with $\{\boldsymbol{u}, \boldsymbol{v}\}$. This is analogous to the approximation in Section 3.1.1.

Definition 5.1. *A closed-form approximation for the PRESS for a TB-PLS model is given by, $\boldsymbol{e}_{-i}$ is given by*

$$J \approx \frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{e}_{-i}\|^2, \tag{5.10}$$

where each leave-one-out error is given by

$$\boldsymbol{e}_{-i} \approx \frac{\boldsymbol{e}_i - t_i^2 \boldsymbol{E}_{y,i} - \boldsymbol{b}_i}{(1-t_i^2)(1-s_i^2)}. \tag{5.11}$$

Here, $\boldsymbol{e}_i = \boldsymbol{y}_i - \boldsymbol{x}_i\boldsymbol{\beta}$ is the TB-PLS residual error, and $\boldsymbol{b} = h_i s_i \boldsymbol{y}_i$, with $h_i = s_i - g t_i$ being the i^{th} residual error for the inner regression model of Eq. (5.4) *and $\boldsymbol{E}_{y,i} \in \mathbb{R}^{1\times q}$ being the i^{th} residual in the TB-PLS model in Eq.* (5.1)*.*

The derivation of Eq. (5.11) is provided in Appendix C.1. Definition 5.1 provides an approximation for the LOOCV error in terms of only the TB-PLS residual errors $\boldsymbol{e}_i$. The error introduced by approximating the leave-one-out estimates of the singular vectors is of order $O\left(\sqrt{\frac{\log(N)}{N}}\right)$. The denominator of Eq. (5.11) is a scaling term related to the contribution of each data point to the latent factors, $\boldsymbol{t}$ and $\boldsymbol{s}$. In this form, it can be seen that the TB-PLS PRESS has similarities with the PRESS for OLS regression where these scaling terms are related to the leverage each point exerts on the regression.

5.2.2 Predictive influence for TB-PLS

Using the approximation (5.11), we now consider how to measure the influence each point exerts on the TB-PLS model. Since we are interested in the predictive performance of the TB-PLS model, we aim to identify influential points as those observations having the greatest effect on the prediction error. In order to quantify this effect, we define the *predictive influence* with respect to an observation $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}$ as the rate of change of the PRESS at that point whilst all other observations remain constant.

Definition 5.2. *The predictive influence of a data point $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}$, which we denote as $\boldsymbol{\pi}_{\boldsymbol{x}_i}(\boldsymbol{u}, \boldsymbol{v}) \in \mathbb{R}^{p\times 1}$, is the derivative of the PRESS with respect to $\boldsymbol{x}_i$ which has the form,*

$$\boldsymbol{\pi}_{\boldsymbol{x}_i}(\boldsymbol{u}, \boldsymbol{v}) = \frac{\partial J}{\partial \boldsymbol{x}_i} = \frac{4\boldsymbol{e}_{-i}}{N}\left(\frac{-\boldsymbol{E}_{y,i}^{\top}}{(1-t_i^2)(1-s_i^2)} + \frac{\boldsymbol{e}_{-i}^{\top}}{(1-t_i)}\right) t_i \boldsymbol{u}^{\top}. \tag{5.12}$$

The derivation for this expression is reported in Appendix C.2. The predictive influence offers a way of measuring how much the prediction error would increase in response to an incremental change in in the observation $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}$ or alternatively, the sensitivity of the prediction error with respect to that observation. The rate of change of the PRESS at this point is given by the magnitude of the predictive influence vector, $\|\boldsymbol{\pi}_{\boldsymbol{x}_i}(\boldsymbol{u}, \boldsymbol{v})\|^2$. If the magnitude of the predictive influence is large, this implies a small change in the observation will result in a large change in the prediction error relative to other points. In this case, removing such a point from the model would cause a large improvement in the prediction error. We can then identify influential observations as those for which the increase in the PRESS is larger relative to other observations.

Some experimental results comparing the proposed predictive influence and the model residuals for the identification of influential observations are reported in Section 5.4.1. In the following Section, we develop the idea of using the predictive

influence to identify points which may belong to different TB-PLS models in heterogeneous data.

5.3 Multi-view predictive partitioning

5.3.1 The MVPP algorithm

Initially we assume that the number of clusters, K, is known. As mentioned in Section 5.1.3, the problem we aim to solve is two-fold. We aim to allocate each observation $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}$, $i = 1, \ldots, N$ into one of K non-overlapping clusters $\{\mathcal{C}_k\}_1^K$ such that each cluster contains exactly N_k observations, with $\sum_{k=1}^K N_k = N$. Simultaneously, for each cluster, we aim to estimate a predictive model parametrised by $\{\boldsymbol{u}_k, \boldsymbol{v}_k\}$ such that within each cluster the observations are similar to each other in a predictive sense.

We achieve this by fitting TB-PLS models to the data and assigning points to ensure that within each cluster they exert the smallest predictive influence on the TB-PLS model. Accordingly, we define the MVPP objective function to be minimised.

Definition 5.3. *The within clusters sum of predictive influences is given by*

$$C(\Theta, \mathcal{C}) = \sum_{k=1}^{K} \sum_{i \in \mathcal{C}_k} \|\boldsymbol{\pi}_{\boldsymbol{x}_i}(\boldsymbol{u}_k, \boldsymbol{v}_k)\|^2, \tag{5.13}$$

where $\boldsymbol{\pi}_{\boldsymbol{x}_i}(\boldsymbol{u}_k, \boldsymbol{v}_k)$ is the predictive influence of a point $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}$ under the k^{th} TB-PLS model.

Minimising Eq. (5.13) involves simultaneously determining the true partitioning of the observations, $\mathcal{C} \equiv \{\mathcal{C}_k\}_1^K$ and estimating TB-PLS model parameters, $\Theta \equiv \{\boldsymbol{u}_k, \boldsymbol{v}_k\}_1^K$ for those K partitions. If the true cluster assignments were known *a priori*, fitting these models and thus minimising the objective function would be trivial. However, since the true partitioning is unknown, there is no analytic solution to this problem. Instead, this problem of estimating the predictive models

and the cluster assignments which minimise the objective function can be solved by considering two related optimisation problems:

1. Given K TB-PLS models with parameters, Θ and keeping these fixed, recover the cluster assignments which solve

$$\min_{\{\mathcal{C}_1,\ldots,\mathcal{C}_K\}} C(\Theta, \mathcal{C}). \tag{5.14}$$

2. Given a set of cluster assignments, $\{\mathcal{C}_k\}_1^K$ and keeping these fixed, estimate the parameters of the K predictive models which solve

$$\min_{\{\boldsymbol{u}_1,\boldsymbol{v}_1\ldots,\boldsymbol{u}_K,\boldsymbol{v}_K\}} C(\Theta, \mathcal{C}). \tag{5.15}$$

Obtaining cluster assignments by solving (5.14) changes the TB-PLS model parameters obtained as a result of solving (5.15) and *vice versa*, therefore these objective functions must be solved iteratively. We propose an iterative algorithm which minimises Eq. (5.13) by alternately solving (5.14) and (5.15). At each iteration, the MVPP algorithm follows three main steps, outlined below, which are guaranteed to decrease the objective function. Therefore, MVPP is guaranteed to converge to a local minimum. After the initialisation step, the MVPP algorithm iterates between the partitioning and estimation steps, as detailed below, until convergence is achieved.

Initialisation (I): Given an initial partitioning of the data, $\mathcal{C}$, both the TB-PLS model parameters and the predictive influences, $\boldsymbol{\pi}_{\boldsymbol{x}_i}(\boldsymbol{u}_k, \boldsymbol{v}_k)$, are computed for all $k = 1, \ldots, K$ clusters and $i = 1, \ldots, N$ observations.

Partitioning (P): Keeping the model parameters fixed, the cluster assignments that minimise (5.14) are obtained by assigning points to the clusters, such that the set $\mathcal{C}_k$ consists of the indices, i corresponding to the observations which exert minimal predictive influence under the k^{th} TB-PLS model,

$$\mathcal{C}_k \leftarrow \left\{ i : \min_k \|\boldsymbol{\pi}_{\boldsymbol{x}_i}(\boldsymbol{u}_k, \boldsymbol{v}_k)\|^2 \right\}_{i=1}^{N}, \text{ for } k = 1, \ldots, K. \tag{5.16}$$

Estimation (E): Keeping the cluster allocations fixed, the parameters $\{\boldsymbol{u}_k, \boldsymbol{v}_k\}$ that minimise (5.15) are estimated using the data points $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}$ for all $i \in \mathcal{C}_k$ for each $k = 1, \ldots, K$ and according to Eq. (5.2).

5.3.2 Algorithm convergence

In this section we demonstrate that the proposed MVPP algorithm converges to a local minimum of the objective function in Eq. (5.13). Since the MVPP algorithm shares similarities with PSC, the convergence of the algorithm can be demonstrated in a similar manner.

Since the cluster assignments are fixed, the minimisation problem of (5.15) can be solved by independently solving the following minimisation problem in each cluster,

$$\min_{\boldsymbol{u}_k, \boldsymbol{v}_k} \sum_{i \in \mathcal{C}_k^{new}} \|\boldsymbol{\pi}_{\boldsymbol{x}_i}(\boldsymbol{u}_k, \boldsymbol{v}_k)\|^2. \tag{5.17}$$

In the following lemma we show that although (5.17) has no closed-form solution, it can be reformulated as a familiar maximisation problem.

Lemma 5.1. *The minimisation problem of* (5.17) *is equivalent to the following maximisation problem*

$$\max_{\boldsymbol{u}_k, \boldsymbol{v}_k} \sum_{i \in \mathcal{C}_k^{new}} \Xi_{k,i}^{-1} \boldsymbol{u}_k{}^\top \boldsymbol{x}_i^\top \boldsymbol{y}_i \boldsymbol{v}_k, \tag{5.18}$$

where the weight terms $\Xi_{k,i} = (1 - (\boldsymbol{x}_i \boldsymbol{u}_k)^2)(1 - (\boldsymbol{y}_i \boldsymbol{v}_k)^2) \leq 1$ *scale each observation by its leverage under the* k^{th} *TB-PLS model.*

The proof is provided in Appendix C.3. The maximisation problem is similar to the optimisation required to fit the TB-PLS model, as in (5.3). However, since the weights are also functions of the terms $\boldsymbol{u}_k$ and $\boldsymbol{v}_k$, the solution cannot be obtained as a result of estimating a single SVD within each cluster. Instead, a solution is

obtained by iteratively computing $\boldsymbol{u}_k$ and $\boldsymbol{v}_k$ for fixed values of $\Xi_{k,i}$ using the SVD and computing new values of $\Xi_{k,i}$. This procedure requires multiple SVD computations which is computationally expensive. Furthermore, minimising (5.18) yeilds a solution where the predictive influence of incorrectly assigned observations within each cluster are downweighted by their leverage, $\Xi_{k,i}$. However for assigning points to clusters, we wish to detect terms which have a large predictive influence.

Therefore, in Step **E** of the MVPP algorithm, rather than solving the weighted problem (5.18) exactly, we instead compute the standard TB-PLS solution given by (5.3) within each cluster using the new cluster assignments. This is equivalent to solving

$$\max_{\boldsymbol{u}_k, \boldsymbol{v}_k} \sum_{i \in \mathcal{C}_k^{new}} {\boldsymbol{u}_k}^\top \boldsymbol{x}_i^\top \boldsymbol{y}_i \boldsymbol{v}_k. \tag{5.19}$$

As a result there is an approximation error between the optimal MVPP solution, obtained by solving (5.18), and the TB-PLS solution, obtained by solving (5.19).

The following lemma quantifies the difference in this approximation error between using the old TB-PLS parameters, Θ^{old} and the new TB-PLS parameters Θ^{new}.The lemma states that estimating new TB-PLS parameters using the new cluster assignments always results in a smaller approximation error.

Lemma 5.2. *For each cluster k, we define the approximation error between the optimal parameters Θ^* obtained by solving* (5.18)*, and the old TB-PLS parameters as*

$$E(\Theta^*, \Theta^{old}) = \sum_{i \in \mathcal{C}_k^{new}} \Xi_{k,i}^{-1} {\boldsymbol{u}_k^*}^\top \boldsymbol{x}_i^\top \boldsymbol{y}_i \boldsymbol{v}_k^* - \sum_{i \in \mathcal{C}_k^{new}} {\boldsymbol{u}_k^{old}}^\top \boldsymbol{x}_i^\top \boldsymbol{y}_i \boldsymbol{v}_k^{old}.$$

We then define the approximation error between the optimal parameters and the new TB-PLS parameters obtained by solving (5.19) *as*

$$E(\Theta^*, \Theta^{new}) = \sum_{i \in \mathcal{C}_k^{new}} \Xi_{k,i}^{-1} {\boldsymbol{u}_k^*}^\top \boldsymbol{x}_i^\top \boldsymbol{y}_i \boldsymbol{v}_k^* - \sum_{i \in \mathcal{C}_k^{new}} {\boldsymbol{u}_k^{new}}^\top \boldsymbol{x}_i^\top \boldsymbol{y}_i \boldsymbol{v}_k^{new}.$$

These error terms satisfy the inequality,

$$E(\Theta^*, \Theta^{new}) \leq E(\Theta^*, \Theta^{old}). \tag{5.20}$$

The proof of this lemma is provided in Appendix C.4. Inequality (5.20), states that estimating new TB-PLS models after reassigning clusters always decreases the approximation error relative to the TB-PLS models estimated at the previous iteration. Therefore, performing Step **E** of the MVPP algorithm always improves the objective function. We are now ready to present our theorem which explicitly states the convergence of the MVPP algorithm.

Theorem 5.1. *Starting with any cluster configuration, $\mathcal{C}$, the MVPP algorithm converges to a local minimum of the objective function, Eq.* (5.13)*.*

The proof of this theorem follows an argument analogous to the proof of Theorem 4.1.

Proof. We denote by $C(\Theta^{old}, \mathcal{C}^{old})$ the objective function, Eq. (5.13), evaluated using the cluster assignments $\mathcal{C}^{old}$ and TB-PLS model parameters Θ^{old}. We demonstrate that when performing a single iteration of the MVPP algorithm the objective function is always decreased, that is

$$C(\Theta^{new}, \mathcal{C}^{new}) \leq C(\Theta^{old}, \mathcal{C}^{new}) \leq C(\Theta^{old}, \mathcal{C}^{old}) \tag{5.21}$$

By definition, the cluster assignments, $\mathcal{C}^{new}$, obtained in Step **P** always minimise the objective function and so

$$C(\Theta^{old}, \mathcal{C}^{new}) \leq C(\Theta^{old}, \mathcal{C}^{old})$$

From Lemma 5.1, the optimal MVPP parameters yield a value of the objective function, $C(\Theta^*, \mathcal{C}^{new})$ such that $C(\Theta^*, \mathcal{C}^{new}) \leq C(\Theta^{old}, \mathcal{C}^{new})$. From Lemma 5.2, the difference between the optimal MVPP parameters and the old TB-PLS parameters is given by $E(\Theta^*, \Theta^{old})$.

Similarly, estimating the new TB-PLS parameters yields a value of the objective function, $C(\Theta^{new}, \mathcal{C}^{new}) \geq C(\Theta^*, \mathcal{C}^{new})$ where the difference between the optimal parameters and the new TB-PLS parameters is given by $E(\Theta^*, \Theta^{new})$. Since from

Lemma 5.2, $E(\Theta^*, \Theta^{new}) \leq E(\Theta^*, \Theta^{old})$, the new value of the objective function satisfies

$$C(\Theta^{new}, \mathcal{C}^{new}) \leq C(\Theta^{old}, \mathcal{C}^{new}).$$

Since (5.21) holds at each iteration, as a result the objective function always decreases and the MVPP algorithm converges to a locally optimal solution. □

5.3.3 Total predictive influence

The definition of the predictive influence, $\boldsymbol{\pi}_{\boldsymbol{x}_i}(\boldsymbol{u}, \boldsymbol{v})$ assumes $\boldsymbol{y}_i$ is constant and therefore does not consider how variation in $\boldsymbol{y}_i$ affects the PRESS. However, according to Eq. (5.1), TB-PLS models systematic noise in both views in the form of the residual matrices $\boldsymbol{E}_x$ and $\boldsymbol{E}_y$. Assuming $\boldsymbol{y}_i$ is constant ignores the variation in $\boldsymbol{E}_y$ and so the predictive influence is sensitive to noise in the response. In order to obtain a measure of predictive influence which is robust to noise in both views it is important to explicitly quantify the effect of $\boldsymbol{y}_i$ on the predictive ability of the given TB-PLS model. Therefore we introduce a measure of total predictive influence with respect to the pair of points $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}$.

Definition 5.4. *The predictive influence of a data point $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}$, which we denote as $\boldsymbol{\pi}_{\boldsymbol{x}_i, \boldsymbol{y}_i}(\boldsymbol{u}, \boldsymbol{v}) \in \mathbb{R}^{(P+Q)\times 1}$, is the total derivative of the PRESS with respect to the P variables in $\boldsymbol{x}_i$ and the Q variables in $\boldsymbol{y}_i$,*

$$\boldsymbol{\pi}_{\boldsymbol{x}_i, \boldsymbol{y}_i}(\boldsymbol{u}, \boldsymbol{v}) = \left[\frac{\partial J}{\partial x_{i,1}}, \ldots, \frac{\partial J}{\partial x_{i,P}}, \frac{\partial J}{\partial y_{i,1}}, \ldots, \frac{\partial J}{\partial y_{i,Q}} \right]$$

$$= \left[\frac{\partial J}{\partial \boldsymbol{x}_i}, \frac{\partial J}{\partial \boldsymbol{y}_i} \right]. \tag{5.22}$$

The derivation of $\frac{\partial J}{\partial \boldsymbol{y}_i}$ follows the same argument as the derivation of $\frac{\partial J}{\partial \boldsymbol{x}_i}$ in Appendix C.2. The magnitude of the total predictive influence vector $\boldsymbol{\pi}_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathbb{R}^{(P+Q)\times 1}$ describes the rate of change in the PRESS function in response to an incremental change in the predictor $\boldsymbol{x}_i$ and the response $\boldsymbol{y}_i$. As such, the total

derivative is more robust to noise than the individual gradients since it considers the effect of both $\boldsymbol{E}_x$ and $\boldsymbol{E}_y$ on the PRESS. We demonstrate this property of the total predictive influence with experiments in Section 5.4.

Throughout the rest of this chapter, all of the results have been obtained by computing the predictive influence with respect to $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}$ as in Eq. 5.4.

5.3.4 Model selection

Model selection in both clustering and TB-PLS are challenging problems which have previously only been considered separately. Within our framework, the PRESS statistic provides a robust method for efficiently evaluating the fit of the TB-PLS models to each cluster. A straightforward application of the PRESS allows us to identify the optimal number of clusters, K. We also apply a similar intuition to attempt to learn the number of latent factors of each TB-PLS model, $R_1, \ldots, R_K$.

Since our algorithm aims to recover predictive relationships on subsets of the data, the number of clusters is inherently linked to its predictive performance. If K is estimated correctly, the resulting prediction error should be minimised since the correct model has been found. We therefore propose a method to select the number of clusters by minimising the out-of-sample prediction error which overcomes the issue of over-fitting as we increase K. The strategy consists in running the MVPP algorithm using values of K between 1 (no clusters) and K_{max}, the maximum number of cluster that we think can be supported by the data. We then select the value of K which minimises the mean PRESS value over the K clusters. This is possible due to our computationally efficient formulation of the PRESS for TB-PLS and the fact that we aim to recover clusters which are maximally predictive. The performance of this approach using simulated data is discussed in Section 5.4.3.

In the case where there is little noise in the data, the number of latent factors can be learned by simply evaluating the PRESS in each cluster at each iteration. Therefore, in the k^{th} cluster, the value of R_k is selected such that the PRESS is minimised. Since we select the value of R_k which minimises the PRESS, this also guaranteed to decrease the objective function. However, as the amount of noise

in the data increases, selecting each optimal R_k value becomes a more difficult task due to the iterative nature of the algorithm. In this case, setting $R = 1$ tends to capture the important predictive relationships which define the clusters whereas increasing each R_k can actually be detrimental to clustering performance. This issue is discussed in Section 5.4.3.

5.4 Performance evaluation using simulated data

5.4.1 Identifying influential observations

Initially we assess the ability of our criterion for detecting influential observations under a TB-PLS model, and demonstrate why using model residuals only is unsuitable. For this assessment, we assume an homogeneous population consisting of bivariate points under each view, so $P = Q = 2$. We also assume that one latent factor only is needed to explain a large portion of covariance between the views.

In order to generate data under the TB-PLS model, we first create the paired vectors $\{\boldsymbol{t}, \boldsymbol{s}\}$ by simulating $N = 100$ elements from a bivariate normal distribution with zero mean, unit variances and off diagonal elements 0.9. The corresponding factor loadings $\boldsymbol{p}$ and $\boldsymbol{q}$ are simulated independently from a uniform distribution, Unif$(0, 1)$. We then randomly select three observations in the $\boldsymbol{X}$ view and add standard Gaussian noise to each so that the between-view predictive relationship for those observations are perturbed. Figure 5.2a shows a plot of the predictors $\boldsymbol{X}$ and the responses $\boldsymbol{Y}$. The three influential observations are circled in each view. Since these observations are only different in terms of their predictive relationships, they are undetectable by visually exploring this scatter plot.

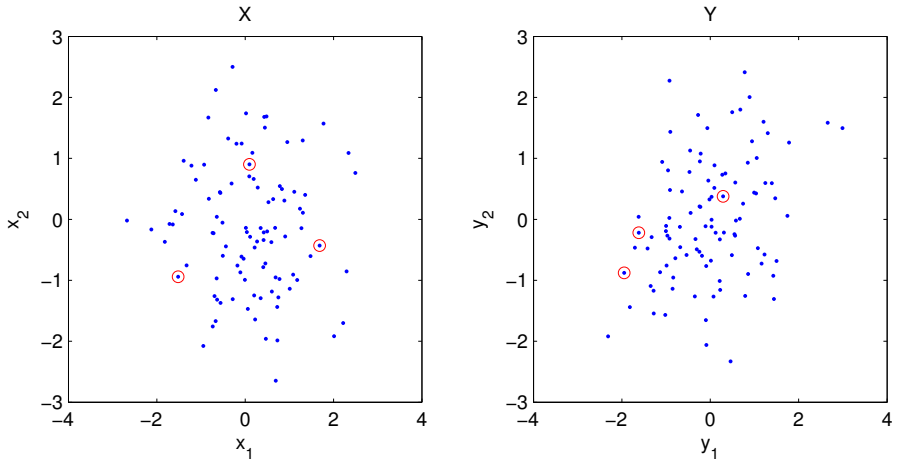
Using all 100 points, we fit a TB-PLS model with $R = 1$ and compute both the residual error and the predictive influence of each observation. In Figure 5.2b, the observations in $\boldsymbol{X}$ are plotted against their corresponding residuals (shown in the left-hand plot) and predictive influences (shown in the right-hand plot). Since TB-PLS aims to minimise the residual error of all observations, including the influential observations results in a biased model fit; although the influential observations ex-

hibit large residuals, this is not sufficient to distinguish them from non-influential observations. On the other hand, the predictive influence of each point is computed by implicitly performing leave-one-out cross validation and, as a consequence of this, the predictive influence of those points is larger than that of any of the other points. This simple example provides a clear indication that the influential observations can be identified by comparing the relative predictive influence between all points.

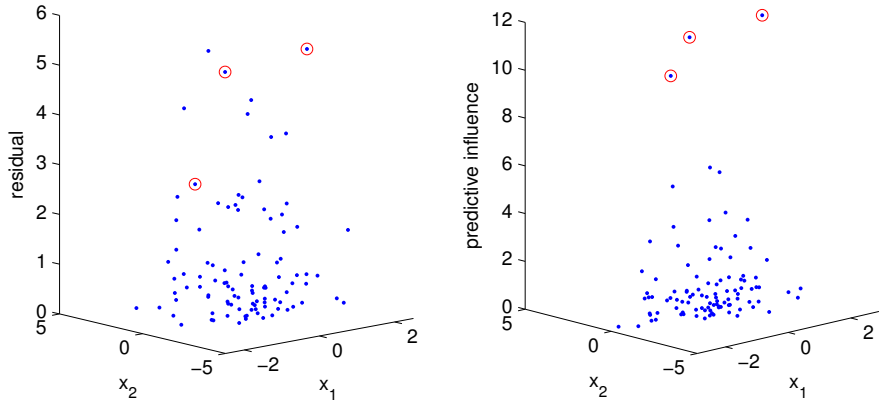
We also perform a more systematic and realistic evaluation in higher dimensions. For this study, we simulate 300 independent data sets, whereby each data set has $P = Q = 200$, $N = 100$ and three influential observations. We follow a similar simulation procedure as the one described before, and set $R = 1$. Once the TB-PLS has been fit, all points are ranked in decreasing order, from those having the largest predictive influence and largest residual. We then select the first top m ranked observations (with $m = 1, \ldots, N$) and define a true positive as any truly influential observation that is among the selected ones; all other observations among those m are considered false positives.

Figure 5.3 compares the ROC curve obtained using the predictive influence and the residual error for this task. This figure shows that the predictive influence consistently identifies the true influential observations with fewer false positives than when the residual is used. Using the predictive influence we detect all influential observations with a false positive rate of 0.34 whereas using the residual we detect almost as many false positives as influential observations. Maximum sensitivity occurs with a false positive rate of 0.97. This suggests that using the residuals to detect influential observations in high dimensions is approximately equivalent to a random guess, and clearly demonstrates the superiority of the proposed predictive influence measure for this task.

Here we revisit the illustrative example described in Section 5.1.3. The predictors shown in Figure 5.1 consist of three-dimensional points sampled uniformly along a line and a plane, and these two subspaces intersect. The response consists of a noisy linear combination of the points in each cluster. Using the same simulated data, we can explore the performance of both our MVPP algorithm and a different



(a) Two dimensional predictors and responses generated under the TB-PLS model. The influential observations are circled.



(b) Two dimensional predictors plotted against their corresponding magnitude residual error and predictive influence respectively.

Figure 5.2: 5.2a shows the two dimensional predictors, $\boldsymbol{X}$ and responses, $\boldsymbol{Y}$ with the influential observations circled. It is clear that the influential observations cannot be identified by simply examining these scatter plots. 5.2b shows the magnitude residual (left-hand plot) and predictive influence (right-hand plot) for each observation in $\boldsymbol{X}$. The predictive influence of the influential observations is much larger than that of all other observations so that these points are clearly identified. The same degree of separation is not evident by examining the magnitude residual error.

multi-view clustering algorithm, MV-CCA [18]. MV-CCA makes two assumptions about the data: firstly, it fits a single, global CCA model which assumes all points belong to the same low dimensional subspace; secondly, it recovers the clusters us-

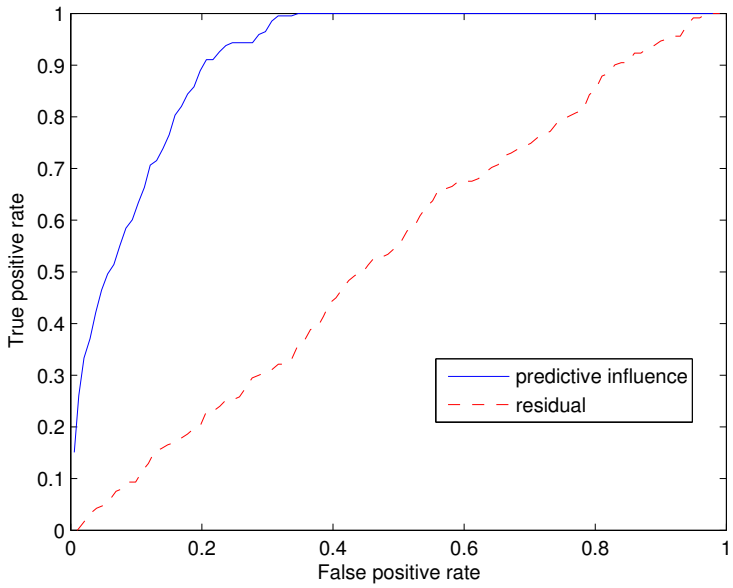


Figure 5.3: ROC curve which compares the ability to detect outliers of the predictive influence and the residual. The results are averaged over 300 Monte Carlo simulations. Using the predictive influence to detect influential observations consistently identifies more true positives for a given false positive rate than using the residual. The predictive influence detects all influential observations with a false positive rate of 0.34 whereas the residual consistently identifies almost as many false positives as true positives.

ing K-means which assumes that the clusters are well separated geometrically in this subspace. An important aspect of MV-CCA is that since geometric clusters are assumed to exist in both views, K-means is performed using the latent factor corresponding to only one of the views. In these experiments we use the latent factor corresponding to the X view.

Figure 5.4 shows the clustering results on this example data set using both MV-CCA and MVPP. The result of clustering using MV-CCA shown in Figure 5.4a highlights the weaknesses of using a global, geometric distance-based method since the existence of clusters is only apparent if local latent factors are estimated using the correct subset of data. MV-CCA fits a single plane to the data which is similar to the one estimated by a global TB-PLS model, as in Figure 5.1b. The points are then clustered based on their geometric separation on that plane which results in an

incorrect cluster allocation.

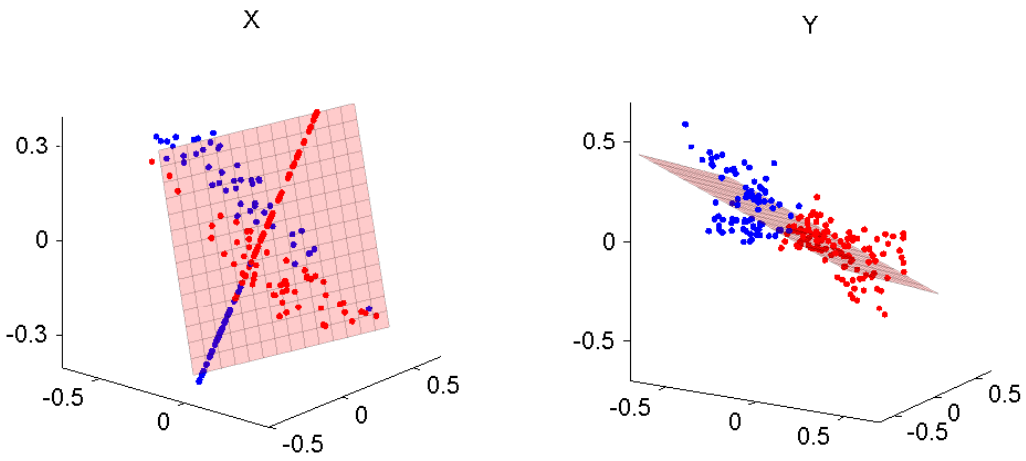
In comparison, Figure 5.4b shows the result of clustering with MVPP showing how the ability to recover the true clusters, and therefore deal with the confounding geometric structures, by inferring the true underlying predictive models. Moreover, since the noise in the data is low, in this example we are able to let MVPP learn the true number of latent factors in each cluster using the procedure described in Section 5.3.4.

5.4.2 Simulation settings

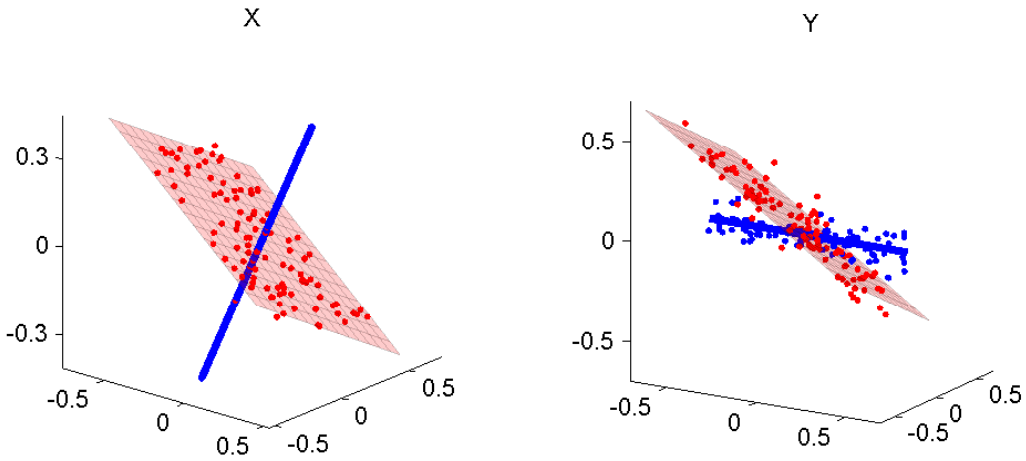
In order to extensively evaluate the performance of predictive partitioning and compare it fairly to other multi-view clustering methods, we devise two different simulation settings which are designed to highlight situations both where current approaches to multi-view clustering are expected to succeed and fail.

The rationale behind the two scenarios we construct comes from considering what it means for observations to cluster in two views. Commonly, in a single view, clusters are considered to be formed by geometrically distinct groups of data points. This notion of geometric distance is also encountered implicitly in mixture models. Separation conditions have been developed for the exact recovery of mixtures of Gaussian distributions, for instance, for which the minimum required separation between means of the clusters is proportional to the cluster variances [47]. In the extension to two views, clusters in each view can also be defined as groups of data points which are well separated geometrically such that similar conditions apply [18].

Therefore, in scenario A, we construct clusters according to the assumption that data points have a similar geometric structure under both views which should be recovered by existing multi-view clustering algorithms. However, in our experiments we assess the performance as a function of the signal to noise ratio. As the level of noise is increased, the between-cluster separation diminishes to the point that all clusters are undetectable using a notion of geometric distance. In this scenario, it will be clear that a clustering approach based on predictive influence is expected to



(a) Clustering using MV-CCA



(b) Clustering using MVPP

Figure 5.4: Plot (a) shows the result of clustering the example dataset introduced in Figure 5.1 using the MV-CCA method. It can be seen that MV-CCA fits a single plane to the data and assigns points to clusters based on geometric distances between points on that plane so the resulting clustering is incorrect. Plot (b) shows the result of clustering using the MVPP algorithm which models the predictive relationship within each cluster. As a result, the true subspaces and cluster assignments are recovered.

be more robust against noise.

In scenario B we consider a situation where the clustering is not defined by

geometric structure. We simulate data under cluster-wise regression models where the geometric structure in the predictor view is different to that in the response view. In this situation, clustering based on geometric separation is expected to perform poorly regardless of the signal to noise ratio. In all of these settings we set the number of latent factors, $R = 1$ and the number of clusters, $K = 2$.

Scenario A: geometric clusters

The first simulation setting involves constructing K "geometric clusters" (up until the addition of noise). We simulate each pairs of latent factors $\boldsymbol{t}_k$ and $\boldsymbol{s}_k$, with $k = 1, \ldots, K$, from a bivariate normal distribution. Each $i = 1, \ldots, N_k$ element, where $N_k = 50$, is simulated as

$$(t_{k,i}, s_{k,i}) \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}), \tag{5.23}$$

where the means of the latent factors, $\boldsymbol{\mu}_k$ defines the separation between clusters. The covariance matrix is given by $\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix}$.

In order to induce a covariance structure in the $\boldsymbol{X}$ loadings, we first generate a vector $\boldsymbol{w}_k$ of length $P = 200$ where each of the $p = 1, \ldots, P$ elements is sampled from a uniform distribution

$$w_{k,p} \sim \begin{cases} \text{Unif}[0, 1], & \text{if } p = 1, \ldots, P/2 \\ \text{Unif}[1, 2], & \text{if } p = P/2 + 1, \ldots, p \end{cases}$$

The P elements of the $\boldsymbol{X}$ loadings and the Q elements of the $\boldsymbol{Y}$ loadings are then simulated in the following way

$$\boldsymbol{u}_k \sim \mathcal{N}\left(0, \boldsymbol{w}_k {\boldsymbol{w}_k}^\top\right), \tag{5.24}$$

$$\boldsymbol{v}_k \sim \text{Unif}[0, 1]. \tag{5.25}$$

We then normalise the vectors so that $\|\boldsymbol{u}_k\| = \|\boldsymbol{v}_k\| = 1$. Finally, for $K = 2$, each

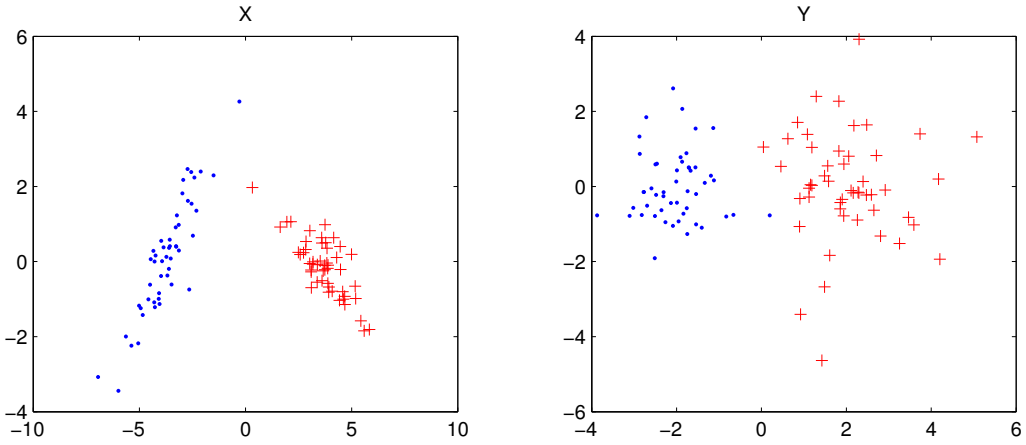


Figure 5.5: An example of data generated in scenario A where the clusters are "geometric clusters" i.e. the Euclidean distance between points within clusters is small compared to points between clusters. The predictors $\boldsymbol{X} \in \mathbb{R}^{100\times 200}$ and response $\boldsymbol{Y} \in \mathbb{R}^{100\times 200}$ have been plotted in the projected space.

pair of observations is generated from the TB-PLS model in the following way

$$\boldsymbol{x}_i = \begin{cases} t_{1,i}\boldsymbol{u}_1{}^\top + \boldsymbol{E}_{x,i}, & \text{if } i \in \mathcal{C}_1 \\ t_{2,i}\boldsymbol{u}_2{}^\top + \boldsymbol{E}_{x,i}, & \text{if } i \in \mathcal{C}_2 \end{cases} \tag{5.26}$$

$$\boldsymbol{y}_i = \begin{cases} s_{1,i}\boldsymbol{v}_1{}^\top + \boldsymbol{E}_{y,i}, & \text{if } i \in \mathcal{C}_1 \\ s_{2,i}\boldsymbol{v}_2{}^\top + \boldsymbol{E}_{y,i}, & \text{if } i \in \mathcal{C}_2 \end{cases} \tag{5.27}$$

where each element of $\boldsymbol{E}_{x,i} \in \mathbb{R}^{1\times P}$ and $\boldsymbol{E}_{y,i} \in \mathbb{R}^{1\times Q}$ are sampled i.i.d from a normal distribution, $\mathcal{N}(0, \sigma^2)$. The signal to noise ratio (SNR), and thus the geometric separation between clusters, is decreased by increasing σ^2.

Figure 5.5 shows an example of data points generated under this simulation setting; the SNR is large and the geometric clusters are well separated. As the SNR decreases, the geometric clusters become less well separated and so this setting tests the suitability of the predictive influence for clustering when the data is noisy.

Scenario B: predictive clusters

The second setting truly tests the predictive nature of the algorithm by breaking the link between geometric and predictive clusters. In this setting, the geometric position of the clusters in $\boldsymbol{X}$ and the predictive relationship between $\boldsymbol{X}$ and $\boldsymbol{Y}$ is no longer related. We start by constructing the data as before according to Eqs. (5.23) - (5.27) for $K = 2$. However, we now split the first cluster in $\boldsymbol{X}$ space into three equal parts and translate each of the parts by a constant c_1. For all $i \in \mathcal{C}_1$

$$\boldsymbol{x}_i = \begin{cases} \boldsymbol{x}_i + c_1 & \text{if } j = 1, \ldots, N_k/3 \\ \boldsymbol{x}_i & \text{if } i = N_k/3 + 1, \ldots, 2N_k/3 \\ \boldsymbol{x}_i - c_1 & \text{if } i = 2N_k/3 + 1, \ldots, N_k. \end{cases}$$

We then split the second cluster in $\boldsymbol{X}$ space into two equal parts and perform a similar translation operation with a constant c_2. For all $i \in \mathcal{C}_2$

$$\boldsymbol{x}_i = \begin{cases} \boldsymbol{x}_i + c_2 & \text{if } i = 1, \ldots, N_k/2 \\ \boldsymbol{x}_i & \text{if } i = N_k/2, \ldots, N_k. \end{cases}$$

The result is that there are now four distinct geometric clusters in $\boldsymbol{X}$ space but still only two clusters which are predictive of the points in $\boldsymbol{Y}$ space. Parametrising the data simulation procedure to depend on the constants c_1 and c_2 means that we can generate artificial data sets where one of the geometric clusters in $\mathcal{C}_1$ are geometrically much closer to $\mathcal{C}_2$ however the predictive relationship remains unchanged. We call these structures "confounding clusters".

Figure 5.6 shows an example of this simulation setting when the SNR is large. In this setting, noise is only added in the response which preserves the confounding geometric clusters in $\boldsymbol{X}$ but removes the separation between clusters in $\boldsymbol{Y}$. Therefore we expect methods which do not take into account predictive influence to fail to recover the true clusters and instead only recover the confounding geometric clusters.

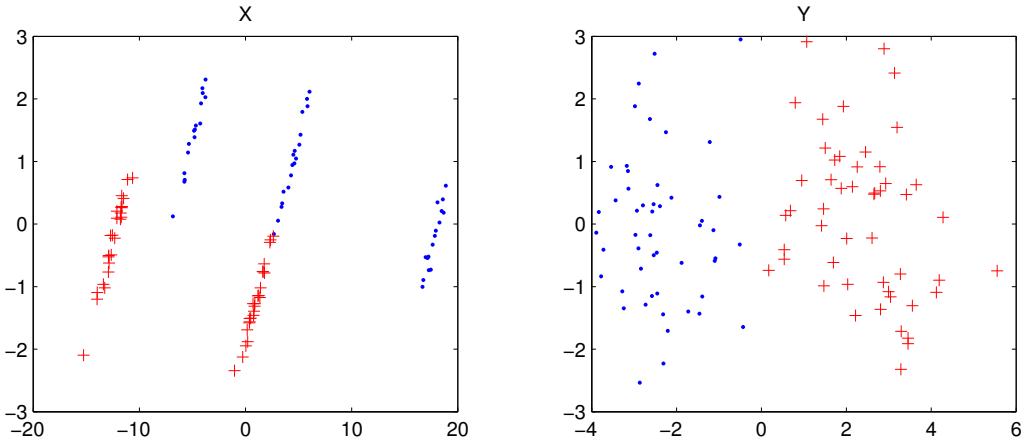


Figure 5.6: An example of data generated in scenario B. Points in the predictive clusters in $\boldsymbol{X}$ have been translated to create four clear geometric clusters. In this case, in the $\boldsymbol{X}$ view, the distance between cluster 2 (crosses) and two of the geometric clusters from cluster one (dots) is smaller than the distance between the points in cluster one. This implies that Euclidean distance based clustering will fail to recover the true clusters. The predictors $\boldsymbol{X} \in \mathbb{R}^{100\times 200}$ and response $\boldsymbol{Y} \in \mathbb{R}^{100\times 200}$ have been plotted in the projected space.

5.4.3 Experimental results

We first demonstrate the superiority of the predictive influence with respect to both views over computing the predictive influence with respect to the $\boldsymbol{X}$-view only. We can show this by considering the ratio between the sum of predictive influences with respect to both $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ and the sum of predictive influences with respect to $\boldsymbol{x}_i$ only within a cluster which we define as

$$\kappa = \frac{\sum_{i\in\mathcal{C}_k} \|\boldsymbol{\pi}_{\boldsymbol{x}_i,\boldsymbol{y}_i}(\boldsymbol{u}_k,\boldsymbol{v}_k)\|^2}{\sum_{i\in\mathcal{C}_k} \|\boldsymbol{\pi}_{\boldsymbol{x}_i}(\boldsymbol{u}_k,\boldsymbol{v}_k)\|^2}.$$

Since our aim is to minimise the predictive influence within each cluster, κ gives an expression for the relative discriminative ability of using these two different methods for computing the predictive influence for TB-PLS. Figure 5.7 shows these ratios as a function of increasing additive noise (decreasing SNR) in both views. We report on the mean ratios under scenario A over 200 Monte Carlo simulations. It can be seen that when the noise in both views is small, $\kappa \approx 1$ and so both

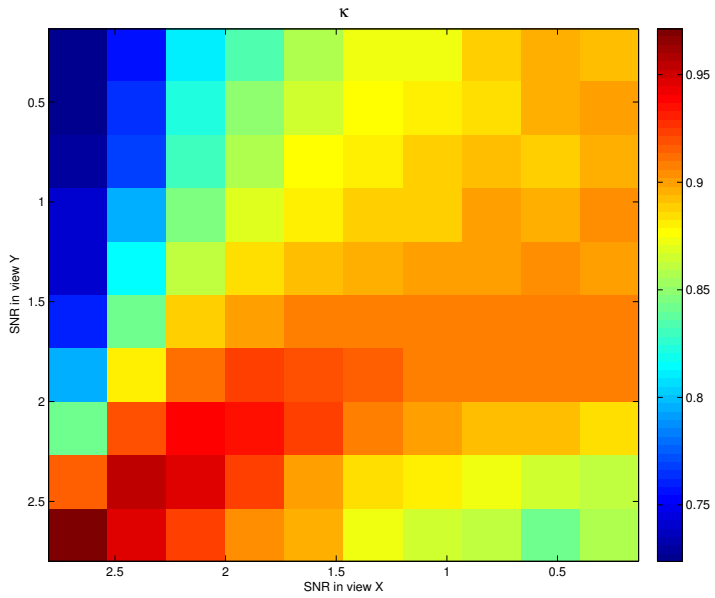


Figure 5.7: The ratio, κ between the value of the objective function obtained using the predictive influence with respect to $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ and the predictive influence with respect to $\boldsymbol{x}_i$. The ratio reported is the mean over 200 Monte-Carlo simulations. As the noise is increased in both views, using the predictive influence with respect to both views improves the relative separation between between the clusters.

predictive influences are approximately equivalent. However, as noise in the response increases, κ decreases which implies $\|\boldsymbol{\pi}_{\boldsymbol{x}_i}(\boldsymbol{u}_k, \boldsymbol{v}_k)\|$ increases relative to $\|\boldsymbol{\pi}_{\boldsymbol{x}_i,\boldsymbol{y}_i}(\boldsymbol{u}_k, \boldsymbol{v}_k)\|$. This suggests that when the response is noisy, computing the predictive influences with respect to both $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ is less sensitive to noise and therefore more discriminative for clustering.

Using data simulated under scenarios A and B, we assess the mean clustering and predictive performance of the MVPP algorithm in comparison to some multi-view clustering algorithms over 200 Monte Carlo simulations. In each simulation, the latent factors, loadings and noise are randomly generated as described in section 5.4.2. We also examine issues relating to model selection in the MVPP algorithm.

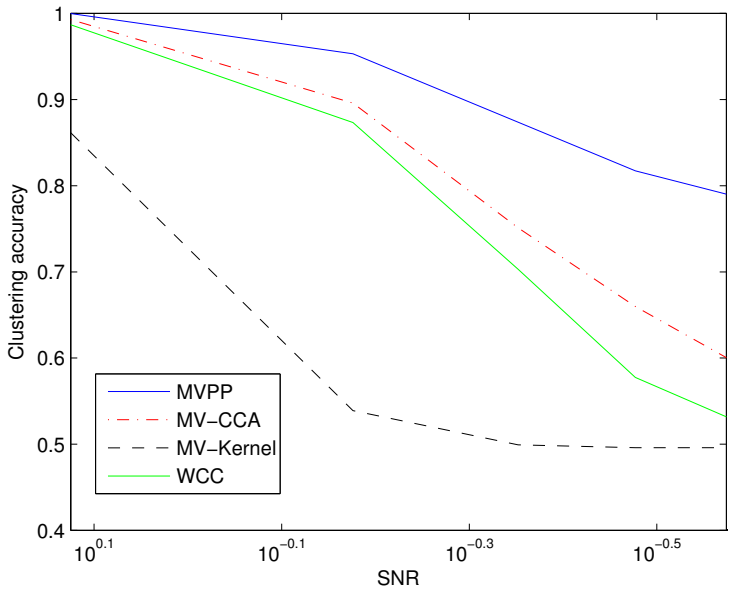


Figure 5.8: Comparing the mean clustering accuracy of different methods for $K = 2$ in simulation setting A over 200 Monte Carlo simulations. When the SNR is high, MVPP achieves maximum accuracy and as the noise increases, the decrease in performance is small relative to the other methods.

Clustering performance

Figure 5.8 shows the result of the comparison of clustering accuracy between methods when $K = 2$ in scenario A. A SNR of $10^{0.1}$ indicates that signal variance is approximately 1.3 times that of the noise variance and so the clusters in both views are well separated whereas a SNR of $10^{-0.5}$ indicates that the clusters overlap almost completely. It can be seen that when the noise level is low, MVPP is able to correctly recover the true clusters. As the noise increases, and the geometric separation between clusters is removed, the clustering accuracy of the competing methods decreases at a faster rate than MVPP.

Since MV-CCA assumes that the clusters are well separated geometrically, as the noise increases the estimated latent factor is biased which decreases the separation between the clusters. Another reason for the difference in performance between MV-CCA and MVPP lies with how the multiple views are used for clustering. Although MV-CCA clustering derives a low dimensional representation of the data

using both views, the actual clustering is performed using the latent factors of only one view. MVPP considers the important predictive contribution from both views in constructing the predictive influences and so clustering occurs jointly between the views.

The MV-kernel method [24] relies on the Euclidean distance between points in constructing the similarity matrix. This method works well only when the clusters are well separated in each view. Computing the Euclidean distance between points in high dimensions before performing dimensionality reduction means that the MV-kernel method is affected by the curse of dimensionality. As such, its performance degrades rapidly as the SNR decreases.

WCC [54] clusters each view separately using K-means and combines the partitions to obtain a consensus. Since it does not take into account the relationship between the two views, when the data is noisy this can result in two extremely different partitions being recovered in each view and therefore a poor consensus clustering.

Figure 5.9 shows the result of the comparison between methods in scenario B. It can be seen that MVPP consistently clusters the observations correctly in this challenging setting and is extremely robust to noise due to the implicit use of cross-validation. Since none of the other methods takes into account the predictive relationship between the clusters and instead only find geometric clusters, they all consistently fail to identify the true clusters. The similar performance for low levels of noise corresponds to these methods consistently misclustering the points based on their geometric position. As the noise increases, the performance of WCC, MV-CCA and MV-kernel remains fairly constant. This confirms that these methods are not correctly utilising the important information in the second view of data even when the predictive clusters in the response are well separated.

Predictive performance

Since only MVPP considers the predictive performance of the clustering by evaluating the PRESS error in each cluster, in order to test the predictive performance

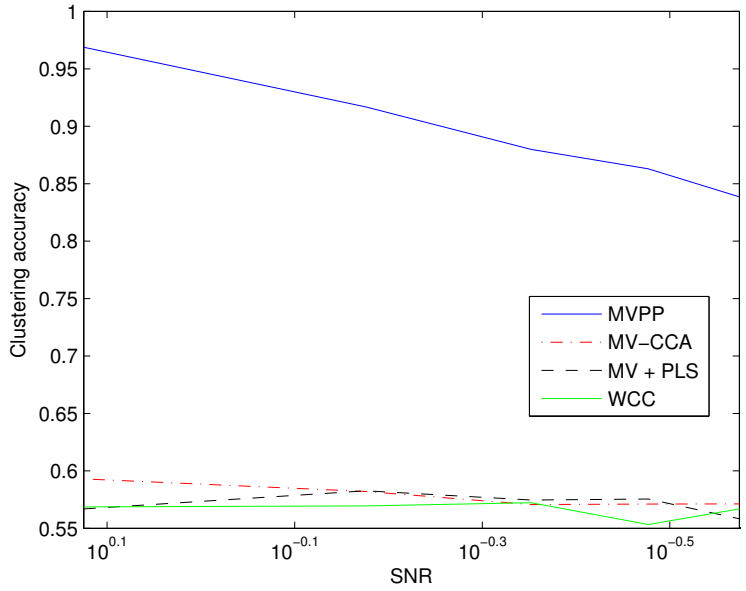


Figure 5.9: Comparing the mean clustering accuracy in simulation setting B over 200 Monte Carlo simulations. MVPP achieves a high clustering accuracy for all levels of noise whereas the competing methods perform poorly even when the SNR is large, since they recover clusters based on the confounding geometric structure in the X view.

of the competing multi-view clustering algorithms we must perform clustering and prediction in two steps. Therefore we first perform clustering with each of the methods on the full dataset and then train a TB-PLS model in each of the obtained clusters. We then test the predictive ability by evaluating the leave-one-out cross validation error within each cluster. For comparison, we also evaluate the LOOCV error of a global TB-PLS model which we fit using all of the data.

Figure 5.10 shows the result of predictive performances under scenario A. This figure shows that MVPP achieves the lowest prediction error amongst the multi-view clustering methods. This is to be expected since the clusters are specifically obtained such that they are maximally predictive through implicit cross validation. The prediction error of the competing multi-view methods is larger than MVPP which indicates that these methods are really not selecting the truly predictive clusters. As the noise increases, the prediction performance of all methods decreases

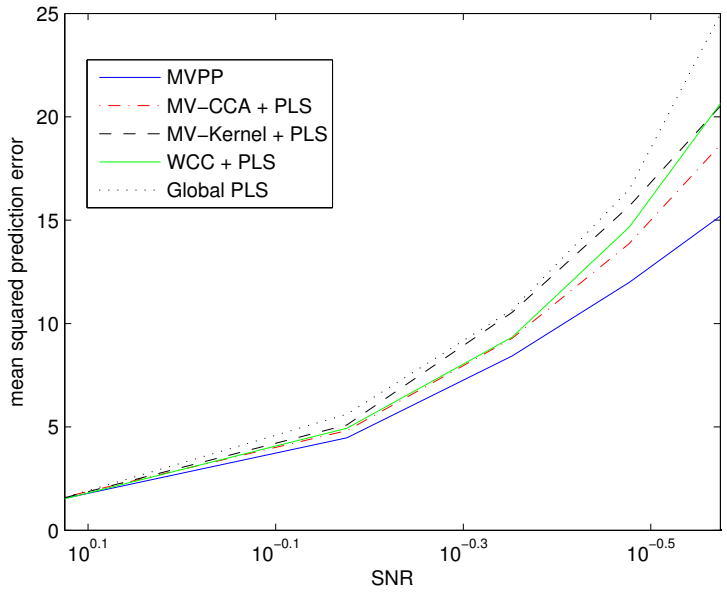


Figure 5.10: Comparing the mean leave-one-out prediction error over 200 Monte Carlo simulations of the clusters obtained by different methods for $K = 2$ in simulation setting A. MVPP consistently achieves the lowest prediction error of the multi-view clustering methods due to the clusters being selected based on their predictive ability. Similarly to the clustering performance, as the noise increases the relative difference between MVPP and the other methods also increases. It can be seen that all clustering methods achieve better prediction than a global PLS model.

however as MVPP is more robust to noise than the competing methods, its relative decrease in performance is smaller. It can be noted that for low levels of noise the global predictive model performs worst of all. This further supports the notion of attempting to uncover locally predictive models within the data.

Figure 5.11 shows the prediction performance in scenario B. Since MVPP is able to accurately recover the predictive clusters, it displays the lowest prediction error amongst the multi-view clustering methods. As noted above, the other multi-view clustering methods only recover the geometric clusters and so their prediction performance is worse. The relative performance difference between competing methods stays similar as noise increases however, since MVPP is affected by noise in $\boldsymbol{Y}$, its predictive performance decreases relative to the other methods.

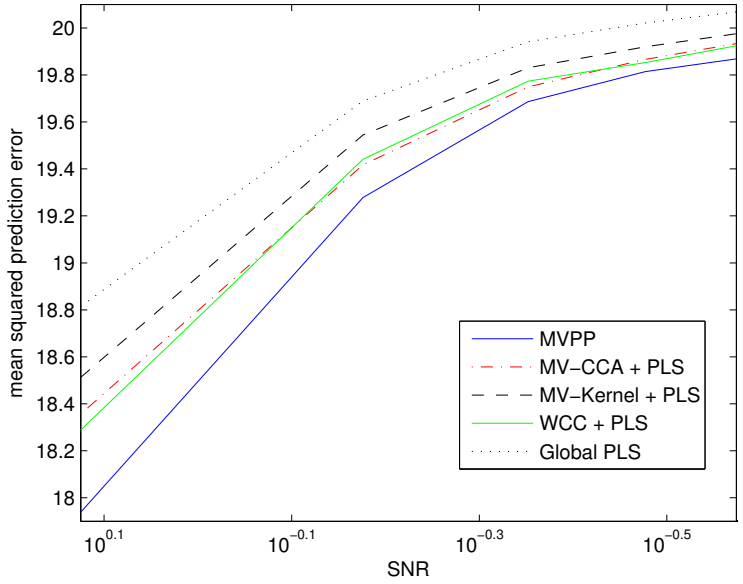


Figure 5.11: Comparing the mean leave-one-out prediction error of the clusters obtained in simulation setting B over 200 Monte Carlo simulations. MVPP achieves the best prediction performance of the multi-view clustering methods. Since as noise increases, the relative clustering clustering performance between MVPP and the competing methods decreases, this relative predictive performance of MVPP also decreases. Again, global PLS achieves the worst prediction accuracy of all methods.

Model selection results

The ability of MVPP to learn the true number of clusters in the data is assessed using the procedure in Section 5.3.4. In this experiment, the data was simulated under setting A and the true number of clusters was set as $K = 2$. Figure 5.12 shows a comparison between the PRESS prediction error and the objective function for different values of K over 200 Monte Carlo simulations. As expected, the objective function decreases monotonically as K is increased whereas the PRESS exhibits a global minimum at $K = 2$.

In the above simulation settings, the number of latent factors was fixed to be $R = 1$. According to the TB-PLS model in Section 5.1.2 the first latent factor is the linear combination of each of the views which explains maximal covariance between $\boldsymbol{X}$ and $\boldsymbol{Y}$. Therefore, the first latent factor is the most important for predic-

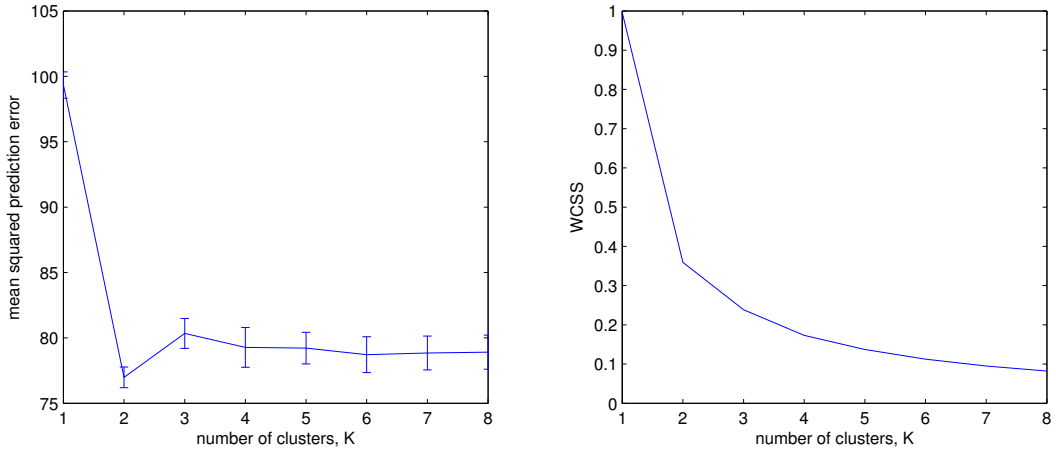


Figure 5.12: Comparing the prediction error with the objective function for different values of K in the first simulation setting where the true value of $K = 2$. It can be seen that as K increases the global minimum of the PRESS occurs when $K = 2$, whereas the objective function decreases monotonically as it begins to overfit the data. The error bars also show that the standard deviation of the PRESS is smallest when $K = 2$. This allows us to use the prediction error to select the true number of clusters.

tion. Each successive latent factor explains a decreasing amount of the covariance between the views and so contributes less to the predictive relationship.

Figure 5.13 shows the effect of the number of latent factors, R on the clustering accuracy of MVPP in scenario A. It can be seen that for low levels of noise, when the clusters are well separated, increasing R has little effect on the clustering accuracy. As the noise increases, the first latent factor appears to capture all of the important predictive relationships in the data whereas subsequent latent factors only fit the noise which causes a detrimental effect on the clustering accuracy as more latent factors are added.

5.5 Applications to web data

In this section we perform the comparison of methods using two real world data sets. The first is the WebKB[1] dataset. WebKB is a collection of interconnected

[1] `http://www.cs.cmu.edu/~webkb`

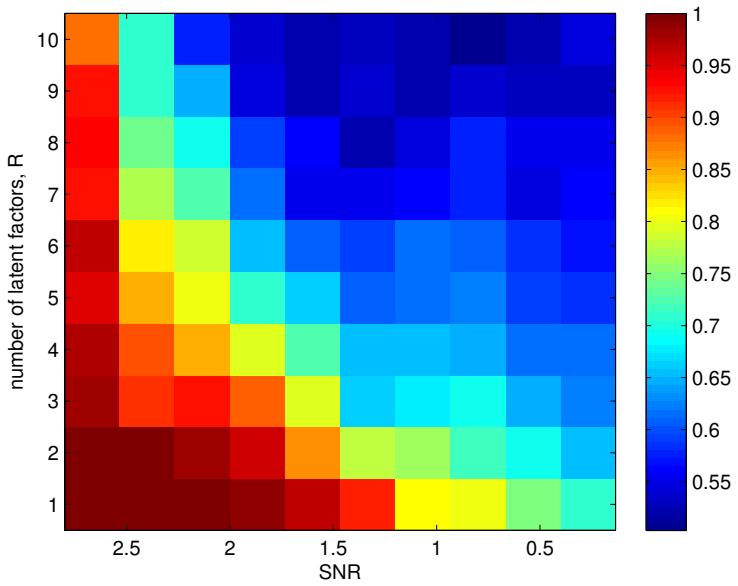


Figure 5.13: The effect of the number of latent factors, R on the clustering accuracy. For low levels of noise, increasing R has little effect on the clustering accuracy. However, as the noise increases, it can be seen that the first latent factor explains all of the signal in the data and increasing R has a detrimental effect on the clustering accuracy.

web pages taken from the computer science departments of four universities: Cornell, Texas, Washington and Wisconsin. This dataset is commonly used to test multi-view clustering algorithms [10, 41, 24]. Each web page can be represented using two views: the text content of the page and the anchor text of the hyperlinks pointing to that page. There are two separate problems associated with the WebKB dataset. The first problem which we denote WebKB-2 involves clustering the pages into two groups consisting of "course" and "non-course" related pages respectively. The second problem, WebKB-4, involves clustering the pages into four groups consisting of "course", "student", "staff" and "faculty" related pages. Both views of the pages consist of a bag of words representation of the important terms where the stop words have been removed. Each word has been normalised according to its frequency in each page and the inverse of the frequency at which pages containing that word occur (term frequency-inverse document frequency normalization). We

University	Observations				View 1	View 2
	Course	Student	Staff	Faculty	P	Q
Cornell	83	18	38	32	1703	694
Texas	103	18	33	31	1703	660
Washington	106	19	65	27	1703	715
Wisconsin	116	22	70	34	1703	745

Table 5.1: A summary of the number of observations and variables in the different configurations of the WebKB dataset.

treat each web page as an observation, i where the predictor vector, $\boldsymbol{x}_i$ is the page text and the response vector, $\boldsymbol{y}_i$ is the link text. The dimensions of these vectors for each university is given in table 5.1. It is known that a predictive relationship exists between views [87] and so we expect the results obtained by MVP to reflect the ability to exploit that relationship in order to correctly identify clusters.

We also evaluate the clustering and prediction performance of MVP and competing methods on a second benchmark dataset, the Citeseer data set [11]. The Citeseer dataset consists of scientific publications $(N = 3312)$ belonging to one of six classes of approximately equal sizes. The predictor view, $\boldsymbol{x}_i$, consists of a bag of words representation of the text of each publication in the same form as the WebKB dataset $(P = 3703)$. We perform two analyses. In the first, the response view, $\boldsymbol{y}_i$ comprises of a binary vector of the incoming references between a paper and the other publications in the dataset ($Q = 2316$). In the second, the response view comprises of a binary vector of the outgoing references from each paper ($Q = 1960$).

For the WebKB-2 clustering problem there are two true clusters of approximately equal size. We again compare MVP with the WCC, MV-CCA and MV-Kernel clustering methods. For each method, we then evaluate the leave-one-out prediction error for the previously recovered clusterings. We also evaluate the leave-one-out prediction error for global PLS which has been estimated using all the data.

Table 5.2 shows the results of clustering and prediction on the WebKB-2 dataset. In all cases, MVP achieves almost 100% clustering accuracy whereas the other

University	Global PLS	WCC	MV-CCA	MV-Kernel	MVPP
Cornell					
Acc	-	0.50	0.56	0.65	0.96
Error	163.69	46.71	137.65	159.93	37.35
Texas					
Acc	-	0.50	0.57	0.71	0.95
Error	177.50	40.90	132.01	173.79	33.74
Washington					
Acc	-	0.87	0.79	0.69	0.97
Error	209.40	46.44	106.86	109.16	31.53
Wisconsin					
Acc	-	0.67	0.76	0.59	0.98
Error	234.16	68.86	171.58	244.85	55.72

Table 5.2: The clustering accuracies (Acc) and mean squared leave-one-out prediction error on the WebKB-2 dataset. MVPP consistently accurately recovers the true clusters and therefore also obtains the best prediction accuracy. The large variance in prediction accuracy between the other methods demonstrates the importance of fitting the correct local models.

methods achieve between $50 - 87\%$ accuracy which suggests that there is a predictive relationship between the text view of the webpage and the incoming links which MVP is able to exploit to recover the true clusterings. MVP also achieves a much lower prediction error than the other clustering methods which vary widely. This suggests that since the dimensionality of the problem is large, a small error in cluster assignment can lead to fitting a poor predictive model.

For the WebKB-4 clustering problem there are four true clusters where one of the clusters is much larger than the others. This poses a particularly challenging scenario since K-means based techniques favour clusters which are of a similar size. Table 5.3 details the results on this dataset. Again, in all cases, MVP achieves the highest clustering accuracy. In this dataset, the clustering accuracy for MVP is approximately 15% lower than for $K = 2$ due to the irregular cluster sizes and the poorer separation between clusters. The other methods also generally achieve poorer clustering accuracy however the relative decrease is not as large. Similarly for the previous dataset, the better clustering performance of the multi-view meth-

University	Global PLS	WCC	MV-CCA	MV-Kernel	MVPP
Cornell					
Acc	-	0.70	0.69	0.44	0.83
Error	163.69	35.43	19.77	105.04	17.89
Texas					
Acc	-	0.58	0.68	0.41	0.86
Error	177.50	54.35	26.21	141.34	18.97
Washington					
Acc	-	0.70	0.68	0.53	0.75
Error	209.40	36.14	33.26	98.58	17.34
Wisconsin					
Acc	-	0.69	0.74	0.53	0.85
Error	234.16	61.61	31.27	110.89	21.13

Table 5.3: The clustering accuracies (Acc) and mean squared leave-one-out prediction error on the WebKB-4 dataset. MVPP again achieves the best clustering and prediction performance. Although the clustering accuracy is worse than in the WebKB-2 configuration, the improved prediction performance suggests that fitting four clusters is a more accurate model of the data.

ods does not necessarily achieve better prediction performance. Despite achieving a relatively poorer clustering accuracy, fitting four clusters instead of two greatly improves the prediction performance of all clustering methods.

Table 5.4 shows the results for clustering and prediction using the Citeseer dataset. It can be seen that in both configurations, MVP achieves the highest clustering accuracy although the relative difference is not as large as for the WebKB dataset. In this case, MVP achieves the lowest prediction error of all methods. The large variance in prediction error between the multi-view clustering methods despite their similar clustering accuracy again suggests that incorrectly clustering observations can severely affect the prediction performance due to the high dimensionality of the data.

5.6 Discussion

In this chapter, we have considered the increasingly popular situation in machine learning of identifying clusters in data by combining information from multiple

Configuration	Global PLS	WCC	MV-CCA	MV-Kernel	MVPP
Text + Inbound					
Acc	-	0.76	0.76	0.73	0.81
Error	344.06	70.62	76.50	110.30	39.51
Text + Outbound					
Acc	-	0.76	0.76	0.72	0.87
Error	278.53	110.46	84.50	73.95	52.96

Table 5.4: The clustering accuracies (Acc) and mean squared leave-one-out prediction error on the Citeseer dataset. MVPP achieves the best clustering accuracy and prediction error whereas the other methods all achieve a similar clustering accuracy.

views. We have highlighted some cases where the notion of a predictive cluster can better uncover the true partitioning in the data. In order to exploit this, our work consolidates the notion of predictive and cluster analysis which were previously mostly considered separately in the multi-view learning literature.

In order to identify the true predictive models in the data, we have developed a novel method for assessing the predictive influence of observations under a TB-PLS model. We then perform multi-view clustering based on grouping together observations which are similarly important for prediction. The resulting algorithm, MVPP, is evaluated on data simulated under the TB-PLS model such that the true clusters are predictive rather than geometric. The results demonstrate how geometric distance based multi-view clustering methods are unable to uncover the true partitions even if those methods explicitly assume the data is constructed using latent factors. On the other hand, MVPP is able to uncover the true clusters in the data to a great degree of accuracy even in the presence of noise and confounding geometric structure. Furthermore, the clusters obtained by MVPP provide the basis of a better predictive model than the clusters obtained by the competing methods. An application to real web page and academic paper data show similar results.

We have also attempted to unify the difficult issues of model selection in clustering and TB-PLS which have previously only been considered seperately. We have shown that our prediction based clustering criterion can be used to learn the true number of clusters. However, we have also seen that learning the number of latent

factors in each of the TB-PLS models remains a difficult problem due to the effects of noise and the iterative nature of the algorithm.

Chapter 6

On-line variable selection in streaming data

Throughout this work we have considered so called *batch* methods for dealing with high-dimensional data which require access to all N observations simultaneously. In this chapter we consider situations where the data arrives sequentially as a data stream. In such situations, batch methods are often not appropriate due to reasons of computational efficiency and the possibility of dealing with non-stationary data.

Streaming data arise in several application domains, including web analytics [67], healthcare monitoring [69] and asset management [109], among others. In all such contexts, large quantities of data are continuously collected, monitored and analyzed over time. When dealing with data streams, a common and important task consists of learning a regression function that explains the linear relationship between a number of incoming streams, regarded as predictive variables or covariates, and a number of co-evolving data streams, regarded as responses. In this chapter we consider a real-time system observes P predictor and Q response data streams at discrete time points. The input data vector observed at time t is denoted by $\boldsymbol{x}_t \in \mathbb{R}^{1\times P}$ where the subscript refers to the time stamp and the dimension P may be very large. The output streams are collected in a vector $\boldsymbol{y}_t \in \mathbb{R}^{1\times Q}$, which may also be very high-dimensional, although Q is generally much smaller than P. Our objective is to recursively estimate a regression function of form $y_t = \boldsymbol{\beta}_t \boldsymbol{x}_t + \epsilon_t$

where each ϵ_t is an independent noise component. Our fundamental assumption is that, at any given time, only a few selected components of $\boldsymbol{x}_t$ contain enough predictive power, and only those should be selected to build the regression model.

There are a number of challenges arising in this setting. Firstly, a decision has to be made on how to select the truly important predictive components of the input data streams that best explain the multivariate response in a computationally efficient manner. We embrace a sparse regression approach where the unimportant variables are excluded from the model by forcing their coefficients to be exactly zero, as described in Section 2.1. Secondly, since the components of $\boldsymbol{x}_t$ may be highly correlated, variable selection arises in an ill-posed problem and special care is needed in order to deal with this difficulty. As will be clear later, we take a dimensionality reduction approach. Thirdly, the relationship between input and output streams is expected to change quite frequently over time, with the frequency of change depending on the specific application domain and nature of the data. This aspect requires the development of adaptive methods that are able to deal with possible non-stationarities and the notion of *concept drift*, that is the time-dependency of the underlying data generating process [98].

As we have seen, when both predictors and responses are high-dimensional it is plausible to assume the existence of *latent factors* that explain a large proportion of the covariance between them. For instance, in computational finance, latent factor models have been extensively used to explain and model asset prices [49]. The methodology we propose builds on a variant of Partial Least Squares (PLS) regression for the efficient estimation of such latent factors.

6.1 Multivariate methods for data streams

In the online setting we assume the data arrives at discrete time points as a vector $\boldsymbol{x}_t \in R^{1\times P}$ with the output, denoted as the vector $\boldsymbol{y}_t \in R^{1\times Q}$. At any time t we only have access to the data points up to the current time. In this section we briefly review *online* implementations of PCA and PLS with an aim to apply these methods to develop an online variable selection algorithm. When considering updating model

parameters in an incremental fashion, there are three important aspects that must be taken into account

- In the case where the observed data is sampled from a stationary distribution, the solution obtained from the online algorithm must converge to the solution given by the corresponding batch algorithm.
- In the case where the distribution of the observed data changes in time, the algorithm should be able to adapt to those changes. This can be achieved using a variety of ways, the most common being the use of *forgetting factors* which is a method of attaching less importance to older data. Therefore, when the distribution changes, data which was sampled from the previous distribution will not contribute as much (or at all) to the current solution. The use of forgetting factors will be described in Section 6.1.1.
- The online algorithm should typically have a lower computational complexity than the corresponding batch algorithm. Since an iteration of the online algorithm must be executed every time a new data point is received, if this is not the case, the batch algorithm may simply be applied over a sliding data window. Therefore, it is desirable to be able to update the solution incrementally using the previously computed solution and the new data without having to re-estimate the solution using all of the available data.

6.1.1 Recursive Least Squares

The key step in PCA and PLS is the computation of the SVD. We have seen in Chapter 2 how this can be reformulated as a regression problem. One potential method of understanding how PCA and PLS may be applied to streaming data is to first examine how online linear regression problems may be solved in an efficient manner. Recursive least squares (see for example [55]) is a well known technique which utilises the standard least squares criterion for linear regression with a uni-

variate response

$$RSS(t)_{LS} = \sum_{i=1}^{t} \|y_i - \boldsymbol{x}_i \boldsymbol{\beta}_t\|^2.$$

The criterion is modified so that we set out to optimise the weighted least squares criterion

$$RSS(t)_{WLS} = \sum_{i=1}^{t} \omega^{t-i} \|y_i - \boldsymbol{x}_i \boldsymbol{\beta_t}\|^2.$$

where $0 \leq \omega \leq 1$ is a forgetting factor which allows exponential discounting of past data so that the algorithm is able to adapt to changes in the data [1] . In the case where $\omega = 1$, no discounting of past data takes place and when $\omega = 0$ the effective sample size is reduced to the current data point only. The solution of the weighted least squares problem is given by

$$\hat{\boldsymbol{\beta}}_t = (\sum_{i=1}^{t} \omega^{t-i} \boldsymbol{x}_i^\top \boldsymbol{x}_i)^{-1} \sum_{i=1}^{t} \omega^{t-i} \boldsymbol{x}_i^\top y_i.$$

From this, it can be seen that the only two quantities required to solve the least squares problem at time t are the weighted covariance matrix of the data $\boldsymbol{S}_t = \sum_{i=1}^{t} \omega^{t-i} \boldsymbol{x}_i^\top \boldsymbol{x}_i$ and the weighted covariance between the data and the response $\boldsymbol{M}_t = \sum_{i=1}^{t} \omega^{t-i} \boldsymbol{x}_i^\top y_i$. Using this formulation, we can update the covariance matrix $\boldsymbol{S}_t$ as

$$\boldsymbol{S}_t = \omega \boldsymbol{S}_{t-1} + \boldsymbol{x}_t^\top \boldsymbol{x}_t. \tag{6.1}$$

Similarly, $\boldsymbol{M}_t$ can be updated as

$$\boldsymbol{M}_t = \omega \boldsymbol{M}_{t-1} + \boldsymbol{x}_t^\top y_t. \tag{6.2}$$

We can then find the new least squares estimate at each time point by calculating $\hat{\boldsymbol{\beta}}_t = \boldsymbol{M}_t \boldsymbol{S}_t^{-1}$ where $\boldsymbol{S}_t^{-1}$ can be efficiently updated at each time point using the

[1] It should be noted that in the on-line learning literature, the forgetting factor is conventionally denoted by λ which is the notation we use in [61]. However, to maintain consistency with this document where λ represents a singular value, we use the unconventional symbol ω.

Sherman-Morrison formula as

$$\boldsymbol{S}_t^{-1} = (\omega \boldsymbol{S}_{t-1} + \boldsymbol{x}_t^\top \boldsymbol{x}_t)^{-1} = \frac{1}{\omega}\left(\boldsymbol{S}_{t-1}^{-1} - \frac{\boldsymbol{S}_{t-1}^{-1}\boldsymbol{x}_t^\top \boldsymbol{x}_t \boldsymbol{S}_{t-1}^{-1}}{\omega + \boldsymbol{x}_t^\top \boldsymbol{S}_{t-1}^{-1}\boldsymbol{x}_t}\right). \tag{6.3}$$

The forgetting factor has the effect of controlling the effective sample size of the algorithm. In the most common case, ω is chosen to be close to one. The amount of forgetting can be quantified by the memory time constant T_0. For such a value of ω it is given by

$$T_0 = \frac{1}{1-\omega},$$

which means that observations older than T_0 are weighted less than e^{-1} of the most recent observation. The selection of a suitable value for ω is an important choice which determines how well the algorithm tracks changes in the data. Typically, a value between 0.98 and 0.995 is chosen [55]. However using a too large a value for ω can mean that the algorithm tracks changes too slowly. Conversely, if ω is too small, the algorithm will be sensitive to noise. We examine the problem of using a time-varying forgetting factor in Section 6.3.3.

In Section 2.2 we saw how PCA can be expressed as a regression problem. Using the same method, a recursive form of the PCA objective function can be derived as

$$RSS_{PCA}(t) = \sum_{i=1}^{t} \omega^{t-i} \|\boldsymbol{x}_i - \boldsymbol{x}_i \boldsymbol{V}_t \boldsymbol{V}_t^\top\|^2.$$

However, as mentioned in Section 3.1.1, this term is fourth-order in $\boldsymbol{V}_t$ and so cannot be solved directly using ordinary least squares. Therefore in order to simplify the cost function, the term $\boldsymbol{x}_t \boldsymbol{V}_t$ is approximated using the estimate at the previous time point, $\boldsymbol{u}_i = \boldsymbol{x}_i \boldsymbol{V}_{t-1}$. This is easily computed since $\boldsymbol{V}_{t-1}$ is already known at the time point t. The resulting incremental PCA cost function is

$$RSS'_{PCA}(t) = \sum_{i=1}^{t} \omega^{t-i} \|\boldsymbol{x}_i - \boldsymbol{u}_i \boldsymbol{V}_t^\top\|^2. \tag{6.4}$$

Provided the data is slowly changing, the approximation error introduced by this step is small although the extracted eigenvectors will not be completely orthogonal.

Using this approximation, we obtain a least squares criterion which is second order in $\boldsymbol{V}_t$. This is the same PAST approximation we considered when computing the leave-one-out reconstruction error for PCA in Section 3.1.1. Eq. (6.4) is minimised using the PAST algorithm [106] which computes $\boldsymbol{V}_t$ using a procedure similar to standard RLS.

6.1.2 The power method and adaptive SIM

The power method (see, for example [36]) is a simple, iterative algorithm to find the largest normalised eigenvector and corresponding eigenvalue of a correlation matrix. The algorithm involves continuously multiplying a vector $\boldsymbol{v}$ by the covariance matrix $\boldsymbol{S} = \boldsymbol{X}^\top \boldsymbol{X}$ and normalising as $\boldsymbol{v} = \boldsymbol{S}\boldsymbol{v}/\|\boldsymbol{S}\boldsymbol{v}\|$. If, at the start of the sequence, $\boldsymbol{v} = \boldsymbol{v}_0$ is initialized to have unit norm, it will converge to the largest eigenvector of $\boldsymbol{S}$.

The simultaneous iterations method (SIM) [36] extends the power method by operating on the columns of the matrix $\boldsymbol{V} = \left[\boldsymbol{v}^{(1)}, ..., \boldsymbol{v}^{(R)}\right]$ to find the first R eigenvectors of $\boldsymbol{S}$. In the SIM algorithm, each of the vectors, $\boldsymbol{v}^{(r)}$ are extracted sequentially in the same manner as in the power method. Without further modifications, each sequence of $\boldsymbol{S}\boldsymbol{v}^{(r)}$ will converge to the subspace spanned by the largest eigenvector. To ensure that they converge to different eigenvectors, the basis vectors must be orthogonalised using a standard method such as the Gram-Schmidt procedure [36].

In order to use SIM in the problem of adaptive, incremental PCA some modifications must be made. Adaptive SIM [30] allows the SIM algorithm to adapt to a changing covariance matrix by updating the estimate of $\boldsymbol{S}_t$ as each new data point arrives using a forgetting factor, ω in the usual manner

$$\boldsymbol{S}_t = \omega \boldsymbol{S}_{t-1} + \boldsymbol{x}_t^\top \boldsymbol{x}_t.$$

This allows the adaptive SIM algorithm to track changes in the eigenstructure of the covariance matrix. At each time point, one iteration of the SIM algorithm is performed to update the estimate of $\boldsymbol{V}_t$, the eigenvectors of $\boldsymbol{S}_t$.

6.1.3 Online PLS

The literature on incremental methods for PLS is quite limited. One reason for this may be that many of the online algorithms for PCA originate from the signal processing and neural networks communities and PLS is not as prevalent there as PCA.

A recursive PLS algorithm has been proposed for applications in chemical process control [23, 22] In fact, this algorithm is not recursive in the sense of recursive least squares where the parameter is directly estimated using a weighted version of the previous parameter and the new data. In this method, the covariance matrices $\boldsymbol{S} = \boldsymbol{X}^\top \boldsymbol{X}$ and $\boldsymbol{M} = \boldsymbol{X}^\top \boldsymbol{Y}$ which are required to extract the PLS weight vectors and the $\boldsymbol{x}$-loading vectors are updated using forgetting factors as in Equations (6.1) and (6.2). As in the standard PLS algorithm, for R PLS components, each PLS weight must be found by computing the SVD of the covariance between $\boldsymbol{X}$ and $\boldsymbol{Y}$. Therefore, when the algorithm is applied in an online setting, the SVD must be computed R times. This suggests there are no significant computational improvements compared to the batch improved PLS algorithm. The only benefit of this recursive PLS algorithm over the batch algorithm is the ability to adapt to changes and the reduced data storage requirements which arise from the manner in which the covariance matrices are updated.

A simple online PLS algorithm was proposed by [95] as part of an efficient locally weighted learning algorithm locally weighted projection regression (LWPR). LWPR incrementally learns and approximates non-linear functions in high dimensional spaces and was developed for use in learning inverse dynamics in humanoid robotics applications. At each time point, t the LWPR algorithm calculates the weight vector $\boldsymbol{u}_t = \boldsymbol{x}_t^\top y_t$ which is simply the covariance between the data and the response. The latent factor, t_t is computed as in the usual way as the projection of the data $\boldsymbol{x}_t$ onto the weight vector $\boldsymbol{u}_t$. As the weight vector is not computed using the SVD, this method has no facility for dealing with a multivariate response.

6.1.4 Online variable selection

To date, the problem of selecting variables on-line has been somewhat less studied. To the best of our knowledge, only two relatively recent works address this issue within a penalised regression framework. The recursive LARS method for on-line Lasso simply performs a single iteration of the LARS algorithm as a new data point becomes available [48]. More recently an alternative approach to on-line Lasso, named adaptive Lasso (aLasso), has been developed based on RLS [6]. They solve the Lasso problem using coordinate descent combined with an adaptive RLS algorithm that can adapt to changes in the data over time. However, neither approach considers a multivariate response or high-dimensional predictors which are two key issues addressed in this chapter.

6.2 PLS regression

In section 5.1.2 we introduced TB-PLS regression for multivariate regression with high-dimensional response. In this section we introduce a different intepretation of PLS which is motivated by regression problems where the predictors are high-dimensional but the response is low-dimensional or even univariate.

In this interpretation of PLS, the predictors and responses are controlled by a single set of R common latent factors, $[\boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(R)}]$[2] in the following way

$$\boldsymbol{X} = \sum_{r=1}^{R} \boldsymbol{s}^{(r)} {\boldsymbol{p}^{(r)}}^{\top} + \boldsymbol{E}_x, \quad \boldsymbol{Y} = \sum_{r=1}^{R} \boldsymbol{s}^{(r)} {\boldsymbol{q}^{(r)}}^{\top} + \boldsymbol{E}_y \,,$$

where $\boldsymbol{s}^{(r)} \in \mathbb{R}^{N \times 1}$ are the latent factors and $\boldsymbol{p}^{(r)} \in \mathbb{R}^{P \times 1}$ and $\boldsymbol{q}^{(r)} \in \mathbb{R}^{Q \times 1}$ are loading vectors of $\boldsymbol{X}$ and $\boldsymbol{Y}$, respectively. In this setting, $Q << P$ and sometimes, $Q < R$. $\boldsymbol{E}_x$ and $\boldsymbol{E}_y$ are residuals for which we do not assume a distribution. If we compare with the TB-PLS model in Eq. (5.1) we can consider PLS regression as a special case of TB-PLS where the parameter which defines the inner relationship, $g = 1$ and therefore $\boldsymbol{t}^{(r)} = \boldsymbol{s}^{(r)}$. The PLS regression algorithm them aims to find the

[2] We denote the common latent factor as $\boldsymbol{s}$ rather than $\boldsymbol{t}$ to avoid confusion with the time index, t.

R latent factors of $\boldsymbol{X}$ such that $\boldsymbol{s}^{(r)} = \boldsymbol{X}^{(r)}\boldsymbol{u}^{(r)}$ where $\boldsymbol{u}^{(r)}$ is a vector of weights estimated by solving the following optimization problem:

$$\begin{aligned} &\max_{\boldsymbol{u}} \operatorname{cov}(\boldsymbol{X}^{(r)}\boldsymbol{u}, \boldsymbol{Y})^2, \\ &\text{subject to } \|\boldsymbol{u}\| = 1. \end{aligned} \tag{6.5}$$

which is equivalent to solving

$$\begin{aligned} &\max_{\boldsymbol{u}} \boldsymbol{u}^\top \boldsymbol{X}^\top \boldsymbol{Y}\boldsymbol{Y}^\top \boldsymbol{X}\boldsymbol{u}, \\ &\text{subject to } \|\boldsymbol{u}\| = 1. \end{aligned} \tag{6.6}$$

The rank of $\boldsymbol{M} = \boldsymbol{X}^\top \boldsymbol{Y}$ is given by the number of non zero singular values of $\boldsymbol{M}$. If $Q < R$ is is not possible to extract all R PLS directions simultaneously using a single SVD. In this case it is normally necessary to deflate $\boldsymbol{X}$ by removing the contribution of the previous latent factor and recomputing the SVD using the deflated matrix.

We address this situation by noting that a similar problem arises in ordinary least squares fitting problems: when $\boldsymbol{X}$ is ill-conditioned, $\boldsymbol{S} = \boldsymbol{X}^\top \boldsymbol{X}$ is rank deficient and therefore cannot be inverted. This situation is generally resolved using *ridge regression* (see for example [38, p. 59]) which applying a small positive constant to the diagonal of $\boldsymbol{S}$ which has the effect of reducing the variance in the solution by adding some bias. We use a similar technique in order to regularise $\boldsymbol{M}\boldsymbol{M}^\top$ by adding a small constant term to the covariance matrix, in the following way:

$$\boldsymbol{G} = \boldsymbol{X}^\top \left(\alpha \boldsymbol{I}_N + (1-\alpha)\boldsymbol{Y}\boldsymbol{Y}^\top\right) \boldsymbol{X}, \tag{6.7}$$

where $0 \leq \alpha \leq 1$. It can be noticed that this has the effect of adding a small constant to the diagonal of $\boldsymbol{Y}\boldsymbol{Y}^\top$. With this modification, it follows that $\operatorname{rank}(\alpha \boldsymbol{I}_N + (1-\alpha)\boldsymbol{Y}\boldsymbol{Y}^\top) = \operatorname{rank}(\boldsymbol{X}^\top \boldsymbol{X})$, and this prevents H from becoming rank deficient [34].

Rearranging Eq. (6.7), we obtain a new covariance matrix

$$\boldsymbol{G} = \alpha \boldsymbol{S} + (1 - \alpha)\boldsymbol{M}\boldsymbol{M}^\top, \tag{6.8}$$

In this form, it can be noted that H is a weighted sum of the covariance matrix of $\boldsymbol{X}$ and the covariance matrix of $\boldsymbol{X}$ and $\boldsymbol{Y}$. The parameter α has a clear interpretation: when $\alpha = 0$, this yields regular PLS; when $\alpha = 1$, this yields PCA; when $0 < \alpha < 1$, we obtain a trade-off between PLS and PCA. Therefore, using this new covariance matrix can be thought of as biasing the PLS solution (which takes into consideration the response in deriving the latent factors and therefore yields better predictive performance) towards the PCA solution (which obtains latent factors in an unsupervised manner). Since $\boldsymbol{M}\boldsymbol{M}^\top$ is larger than $\boldsymbol{S}$, provided α is small, the contribution of $\boldsymbol{M}\boldsymbol{M}^\top$ to $\boldsymbol{G}$ is larger than the contribution of $\boldsymbol{S}$. Therefore the specific value chosen for α, as long as this is not too close to 1, will not have any noticeable effect on the solution; see also Section 6.4.4 for further comments and a sensitivity analysis. However, this simple modification will ensure that $\boldsymbol{G}$ is always full-rank and the PLS weights can then be obtained in a single step by solving the following optimization problem

$$\begin{aligned} &\max_{\boldsymbol{U}} \boldsymbol{U}^\top \boldsymbol{G} \boldsymbol{U}, \\ &\text{subject to } \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_P. \end{aligned} \tag{6.9}$$

so that $\boldsymbol{U} = [\boldsymbol{u}^{(1)}, \ldots, \boldsymbol{u}^{(R)}]$ are the first R eigenvectors of $\boldsymbol{G}$. The latent factors are then computed as $\boldsymbol{s}^{(r)} = \boldsymbol{X}\boldsymbol{u}^{(r)}$ and corresponding $\boldsymbol{Y}$ loadings, $\boldsymbol{Q} = [\boldsymbol{q}^{(1)}, \ldots, \boldsymbol{q}^{(R)}]$ are given by $\boldsymbol{q}^{(r)} = ({\boldsymbol{s}^{(r)}}^\top \boldsymbol{s}^{(r)})^{-1} {\boldsymbol{s}^{(r)}}^\top \boldsymbol{Y}$. The final PLS regression coefficients are $\boldsymbol{\beta} = \boldsymbol{U}\boldsymbol{Q}^\top$.

As we will now see, this formulation of PLS suits our purposes since we must only find a single set of weight vectors.

6.3 Sparse PLS regression

6.3.1 Off-line learning

We now observe that the PLS weights can be made sparse by using a penalised form of the SVD which leads to a novel and efficient method of variable selection based on the PLS framework. Sparse matrix factorization methods have recently been introduced by [80] and [103]. Specifically, [80] formulates a sparse matrix factorization using the best low rank approximation property of the SVD; this is achieved by reformulating the SVD problem as a regression where the aim is to minimise the sum of squared errors between X and its best low rank approximation. [103] propose a more general framework for sparse matrix factorization which includes the sparse SVD method of [80] as a special case.

Recently, a sparse PLS algorithm based on the sparse PCA method mentioned in Section 2.3 has been proposed by [52] which computes the PLS weights using the standard PLS algorithm described in Section 6.2. Since this method does not use a regularised covariance matrix to extract the PLS weights, the problems of rank-deficiency are still present and so R separate SVD computations are required to extract all R latent factors. In this section we use sparse SVD in order to achieve an efficient variable selection algorithm within the PLS framework described in the previous section.

Since the SVD problem of (6.9) can viewed as an ordinary least squares problem for each $r = 1, \ldots, R$, we can obtain a sparse solution by imposing a penalty on the ℓ_1 norm of $\boldsymbol{u}$,

$$\begin{aligned} &\min_{\boldsymbol{u},\boldsymbol{v}} \|\boldsymbol{G} - \boldsymbol{u}\boldsymbol{v}^\top\|^2 + \gamma^{(1)}\|\boldsymbol{u}\|_1 \\ &\text{subject to} \quad \|\boldsymbol{v}\| = 1, \end{aligned} \tag{6.10}$$

where $\gamma^{(1)}$ is a parameter which controls the sparsity of the solution. If $\gamma^{(1)}$ is large enough, it will force some variables to be exactly zero. The problem of Eq. (6.10) can be solved in an iterative fashion by first setting $\boldsymbol{u} = \boldsymbol{u}^{(1)}$ and $\boldsymbol{v} = \boldsymbol{v}^{(1)}$ as

before. The Lasso penalty can then be applied as an iterative component-wise soft thresholding operation on the elements of $\boldsymbol{u}$ in the following way

$$\boldsymbol{u} = \text{sgn}\left(\boldsymbol{G}\boldsymbol{v}\right)\left(|\boldsymbol{G}\boldsymbol{v}| - \gamma^{(1)}\right)_+, \tag{6.11}$$

$$\boldsymbol{v} = \boldsymbol{G}\boldsymbol{u}/\|\boldsymbol{G}\boldsymbol{u}\|. \tag{6.12}$$

Assuming known sparsity parameters $\{\gamma^{(r)}\}_1^R$, the procedure above allows all the PLS weight vectors to be extracted and made sparse at once without the need to recompute an SVD for each dimension. The remaining steps of the PLS algorithm are unchanged.

The number R of PLS components to be retained and the sparsity parameters $\{\gamma^{(r)}\}_1^R$ must be pre-selected. As usually done in off-line PLS, we suggest to determine the initial R by performing K-fold cross-validation using historical data (CV) [52]. In Section 6.3.2 we discuss how the proposed on-line algorithm is able to track changes in the both the number of important latent factors and their weights, over time.

The other parameters to be selected are the $\{\gamma^{(r)}\}_1^R$, which control the amount of sparsity in the input. In several applications, the number of variables to include in the model is controlled by the user, because this yields results that can be more easily interpreted; for instance, in genomics applications [103] and in some financial applications such as index tracking and portfolio optimization (see Section 6.5). When the user wishes to select a specific number of variables for the rth PLS component, the corresponding $\gamma^{(r)}$ can be obtained by finding the roots of the following function related to the following thresholding function

$$f(\gamma^{(r)}) = \sum_{i=1}^{p} \mathbb{I}\left(\text{sgn}(\boldsymbol{u}_i^{(r)})(|\boldsymbol{u}_i^{(r)}| - \gamma^{(r)})_+ > 0\right) - \theta,$$

where $\mathbb{I}$ is an indicator function which finds the non-zero elements of u after the threshold has been applied. The desired value of $\gamma^{(r)}$ is such that $f(\gamma^{(r)}) = 0$. In other applications, the objective is to identify as many important variables as possible in order to achieve accurate predictions. When the degree of sparsity has

to be inferred from historical data, K-fold cross validation can again be used prior to on-line learning; then the optimal value of $\gamma^{(r)}, r = 1, \ldots, R$ minimises the cross-validated prediction error. We next discuss how our on-line algorithm can automatically track the most important latent factors and, within each, the most important variables.

6.3.2 On-line learning

In an on-line learning setting, we no longer assume we have access to the full data matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$. Instead the data stream arrives sequentially at each time point, t, as $\boldsymbol{x}_t \in \mathbb{R}^{1\times P}$. Similarly, the response is observable only at discrete time points as $\boldsymbol{y}_t \in \mathbb{R}^{1\times Q}$. The use of on-line methods for prediction in a streaming data setting is desirable as off-line methods require a full model fit to be performed every time a new data point arrives which can be computationally expensive.

There are three important issues involved in performing sparse and adaptive incremental learning with PLS regression: (i) latent factors must be recursively computed because they may be changing with time; (ii) important predictors must be recursively extracted from the most important latent factors; (iii) adaptation to changes in the data generating mechanism must be data-driven. In this section, we propose an on-line learning procedure which combines all three of the above criteria in the context of Sparse PLS. We call the resulting algorithm incremental Sparse PLS (iS-PLS). To our knowledge, this is the first such method which combines tracking of latent factors with variable selection in an adaptive fashion for data streams.

Before moving on to the description of the algorithm, we state two assumptions about the nature of the data stream:

Assumption 6.1. *The number of important latent factors underlying the data as well as their weights can evolve over time.*

Assumption 6.2. *The important variables to be retained within each latent factors can evolve over time, but their number does not change.*

Assumption 6.1 is required in order to allow for time-dependent changes in the covariance structures of the data streams. Assumption 6.2 also specifies a time-varying functional relationship between the input and the output. However, in order to simplify the estimation procedure, we impose that the number of important variables to be extracted from each latent factor does not change over time; that is, the parameters $\{\gamma^{(r)}\}_1^R$ are time-invariant.

In order to track the latent factors, we must first compute the PLS weights which are found as a result of solving the optimization problem (6.9). In the on-line setting, this amounts to updating the SVD of the time-varying covariance matrix, $\boldsymbol{H}_t$. When a new data point $\boldsymbol{x}_t$ and its corresponding response $\boldsymbol{y}_t$ arrives, we update the individual covariance matrices as follows:

$$\boldsymbol{S}_t = \omega \boldsymbol{S}_{t-1} + \boldsymbol{x}_t^\top \boldsymbol{x}_t, \qquad \boldsymbol{M}_t = \omega \boldsymbol{M}_{t-1} + \boldsymbol{x}_t^\top \boldsymbol{y}_t, \tag{6.13}$$

where $0 \leq \omega \leq 1$ is a forgetting factor which downweights the contribution of past data points. We elaborate further on the use of the forgetting factor in Section 6.3.3. We then construct the PLS covariance matrix $\boldsymbol{G}_t$ of Eq. (6.8), by summing the weighted PCA and PLS covariance matrices $\boldsymbol{S}_t$ and $\boldsymbol{M}_t\boldsymbol{M}_t^\top$, which leads to:

$$\boldsymbol{G}_t = \alpha \boldsymbol{S}_t + (1-\alpha)\boldsymbol{M}_t\boldsymbol{M}_t^\top \ . \tag{6.14}$$

A problem we face with applying the sparse PLS algorithm to streaming data consists in updating the SVD of $\boldsymbol{G}_t$ recursively. Since $\boldsymbol{G}_t$ is a weighted sum between two covariance matrices we are unable to find its eigenvectors using a standard RLS method. RLS requires as input the current estimate of the inverse covariance matrix and the new data observation whereas we essentially only have access to a time-varying covariance matrix, $\boldsymbol{G}_t$.

We resolve this by applying the adaptive SIM algorithm described in Section 6.1.2. At each time point, the estimate of the eigenvectors of the covariance matrix, $\boldsymbol{G}$ are updated by performing a single power iteration as $\boldsymbol{U}_t = \boldsymbol{G}_t\boldsymbol{U}_{t-1}$. Without further modification, each individual sequence $\boldsymbol{G}_t\boldsymbol{u}_t^{(r)}$ will converge to the pri-

mary eigenvector therefore, after each iteration, each column of $\boldsymbol{U}_t$ must be orthogonalised with respect to the other columns to ensure that they converge to different eigenvectors. This requirement is necessary as the true eigenvectors of $\boldsymbol{G}$ form an orthogonal basis. The updated estimate of the eigenvectors is obtained by $\boldsymbol{U}_t = \text{orth}(\boldsymbol{U}_t)$ where $\text{orth}(\boldsymbol{U}_t)$ is any orthogonalization procedure which causes the columns of the matrix $\boldsymbol{U}_t$ to be mutually orthonormal. This ensures the columns of $\boldsymbol{U}_t$ will converge to different eigenvectors of $\boldsymbol{G}$. Specifically, in our implementation we use the Gram-Schmidt orthogonalisation procedure which has a computational complexity of $O(PR^2)$ [36].

Once the weight vectors $\boldsymbol{U}_t$ have been updated, they are made sparse using a modified version of the iterative regularised SVD algorithm used for Sparse PLS in Section 6.3. Since our algorithm is on-line and the solution is updated when a new data point arrives, we no longer iteratively apply the thresholding operation and instead apply it directly to the sparse estimate of the eigenvector found at the previous time point. When the data stream is stationary, this procedure quickly converges to the optimal solution as we effectively perform the iterations in time; simulation studies showing that the convergence takes place are presented in Section 6.4. In the case of a regime change, the proposed algorithm is able to quickly adapt to changes and quickly converges to the new solution (see Sections 6.3.3 and 6.4).

The final steps of the PLS algorithm proceed as in the off-line case. The latent factors $\boldsymbol{s}^{(r)}$ are computed as $\boldsymbol{s}_t^{(r)} = \boldsymbol{x}_t \boldsymbol{u}_t^{(r)}$. Since the number of observations at each update is effectively one, the $\boldsymbol{Y}$-loadings can be computed as $\boldsymbol{q}_t^{(r)} = \boldsymbol{y}_t^\top \boldsymbol{s}_t^{(r)} / ({\boldsymbol{s}_t^{(r)}}^\top \boldsymbol{s}_t^{(r)})$. The sparse PLS regression coefficients are $\boldsymbol{\beta}_t = \boldsymbol{U}_t \boldsymbol{Q}_t^\top$ so that the estimated response at time t is $\hat{\boldsymbol{y}}_t = \boldsymbol{x}_t \boldsymbol{U}_t \boldsymbol{Q}_t^\top$. Algorithm (1) details the resulting iS-PLS procedure in full.

Initialise $\boldsymbol{U}_0 = \boldsymbol{I}$, $\boldsymbol{M}_0 = \mathbf{0}$, $\boldsymbol{S}_0 = \mathbf{0}$;

Data: Input $\boldsymbol{x}_t$ and output $\boldsymbol{y}_t$ at time t

Result: Sparse regression coefficients $\boldsymbol{\beta}_t$ at time t

$\omega_t \longleftarrow \texttt{selfTune}(\boldsymbol{x}_t, \boldsymbol{y}_t, \boldsymbol{\beta}_{t-1})$;

$\boldsymbol{M}_t \longleftarrow \omega_t \boldsymbol{M}_{t-1} + \boldsymbol{x}_t^\top \boldsymbol{y}_t$;

$\boldsymbol{S}_t \longleftarrow \omega_t \boldsymbol{S}_{t-1} + \boldsymbol{x}_t^\top \boldsymbol{x}_t$;

$\boldsymbol{G}_t \longleftarrow \alpha \boldsymbol{S}_t + (1-\alpha)\boldsymbol{M}_t \boldsymbol{M}_t^\top$;

for $r \leftarrow 1$ **to** R **do**

$\boldsymbol{a}^{(r)} \leftarrow \boldsymbol{G}_t \boldsymbol{u}^{(r)}$;

$\boldsymbol{w}^{(r)} \leftarrow \left[\boldsymbol{I}_P - \sum_{k=1}^{r-1} \boldsymbol{u}^{(k)} \boldsymbol{u}^{(k)\top}\right] \boldsymbol{a}^{(r)}, \quad \boldsymbol{w}^{(1)} \leftarrow \boldsymbol{a}^{(1)}$;

$\boldsymbol{u}^{(r)} \leftarrow \boldsymbol{w}^{(r)} / \|\boldsymbol{w}^{(r)}\|$;

$\gamma^{(r)} \leftarrow \texttt{findRoot}(\boldsymbol{u}^{(r)})$;

$\boldsymbol{u}^* \leftarrow \text{sgn}\left(\boldsymbol{u}^{(r)}\right)\left(|\boldsymbol{u}^{(r)}| - \gamma^{(r)}\right)_+$;

$\boldsymbol{u}^* \leftarrow \frac{\boldsymbol{u}^*}{\|\boldsymbol{u}^*\|}$;

$\boldsymbol{u}_t^{(r)} \leftarrow \boldsymbol{u}^*$;

$\boldsymbol{s}_t^{(r)} \leftarrow \boldsymbol{x}_t \boldsymbol{u}_t^{(r)}$;

$\boldsymbol{q}_t^{(r)} \leftarrow \frac{\boldsymbol{y}_t \boldsymbol{s}^{(r)}}{\boldsymbol{s}^{(r)\top} \boldsymbol{s}^{(r)}}$;

end

$\boldsymbol{\beta}_t \leftarrow \boldsymbol{U}_t \boldsymbol{Q}_t^\top$;

Algorithm 1: The iS-PLS Algorithm

A few comments about the initialization of the algorithm are in order. We set $\boldsymbol{U}_0 = [\boldsymbol{u}_0^{(1)}, ..., \boldsymbol{u}_0^{(R)}] = \boldsymbol{I}_{P \times R}$ to ensure that the initial estimates of the eigenvectors are mutually orthogonal. We also initialise $\boldsymbol{M}_0 = \mathbf{0}$ and $\boldsymbol{S}_0 = \mathbf{0}$. In the off-line case, the complexity introduced by the penalisation function is $O(RNP)$. In the on-line case we operate only on a single data point at a time this reduces the complexity of the penalization function at each time point to $O(RP)$.

6.3.3 Adaptive behaviour using self-tuning forgetting factors

The problem of adapting the learning algorithm to changes in the data has often been addressed in the context of recursive least squares (RLS) [40]. The motivation for introducing adaptive behaviour within iS-PLS is to allow us to detect changes in the latent factors underlying the data by tracking some measure of the performance of the algorithm and noticing when there is a decrease in performance signalling a change. When a change is detected, the forgetting factor is modified so that the currently computed latent factors are discarded and the soft-thresholding algorithm starts recursively computing the new solution starting from the current data point.

In order to introduce such adaptive behaviour, we modify a self-tuning forgetting factor algorithm proposed by Paleologu et al. [70] to be used within our iS-PLS algorithm. This procedure for self-tuning of the forgetting factor is motivated by the aim of correctly identify the true least squares regression coefficients and additive noise when the algorithm converges. This is achieved by formulating an expression for the forgetting factor in terms of the ratio between two error quantities estimated using RLS: the variance of the *a priori* error and the variance of the *a posteriori* error at each time point. The *a priori* error, defined as $e_t = y_t - \boldsymbol{x}_t\boldsymbol{\beta}_{t-1}$, is the prediction error of the current time point using the previously estimated coefficient whereas the *a posteriori* error, defined as $\epsilon_t = y_t - \boldsymbol{x}_t\boldsymbol{\beta}_t$, is the residual error using the regression coefficient estimated at the current time point. The self-tuning forgetting factor, ω_t can then be updated at each time point as

$$\omega_t = \frac{\sigma_h \sigma_\epsilon}{\sigma_e - \sigma_\epsilon}, \tag{6.15}$$

where σ_ϵ^2 is the variance of the *a posteriori* error and σ_e^2 is the variance of the *a priori* error. σ_h^2 is the variance of the quantity $h_t = \boldsymbol{x}_t^\top \boldsymbol{S}_t^{-1} \boldsymbol{x}_t$ which is analogous to the leverage at time t. When the data is stationary or changing slowly, the difference between the *a priori* error and the *a posteriori* error is small. In this case, the numerator dominates the ratio in Eq. (6.15) which leads to a larger value of ω_t. In the case where the data is non-stationary, the difference between the *a priori* error and the *a posteriori* error will increase. This causes the denominator to dominate

the ratio in Eq. (6.15) which leads to a smaller value of ω_t. Since inverting $\boldsymbol{S}_t$ at each time point is a computationally expensive operation, Eq. (6.3) can be used instead to efficiently obtain $\boldsymbol{S}_t^{-1}$ as each new data point arrives. In order to ensure ω_t remains between 0 and 1 we take $\omega_t = \min\{\omega_t, 0.999\}$. All the estimates of the noise variances featuring in Eq. (6.15) can be updated recursively using the following equations:

$$\sigma_{h,t}^2 = a\sigma_{h,t-1}^2 + (1-a)h_t^2,$$

$$\sigma_{e,t}^2 = a\sigma_{e,t-1}^2 + (1-a)e_t^2,$$

$$\sigma_{\epsilon,t}^2 = b\sigma_{\epsilon,t-1}^2 + (1-b)e_t^2,$$

where $0 < a < 1$ and $a < b < 1$ are constant terms which determine the effective sample size for the recursive noise estimates.

The parameters a and b are selected so that the system noise is estimated using a relatively long exponential window, as suggested in [53]. The *a priori* error is the current prediction error and is tracked recursively using a short exponential time window which assigns small weights to previous estimates; in this way, the *a priori* error is sensitive to sudden changes in the current prediction error. Accordingly, the algorithm is able to adjust the forgetting factor. Assuming stationary data, in the limit $t \to \infty$, the *a posteriori* error is the true system noise when the estimate of $\boldsymbol{\beta}_t$ has converged to its true value. In this limit, the *a priori* error tends towards the *a posteriori* error. Based on this observation, we can estimate the *a posteriori* error as the *a priori* error over a longer exponential window to provide a stable estimate of the error which will not be affected by sudden changes in the current prediction error. It can be seen that the hyper-parameters a and b control how quickly the error estimates converge to their true values; provided that $b > a$, their specific values do not greatly affect the sensitivity of the self-tuning algorithm after the convergence period. If $a = b$, then $\sigma_{e,t}^2 = \sigma_{\epsilon,t}^2$ which implies the data is stationary and so ω_t is prevented from adapting. If $a > b$, the error estimates will not converge to their true values and the algorithm will attempt to continuously adjust the forgetting factor.

In our experiments we set $a = 0.5$ which is a short exponential window allowing

$\sigma^2_{e,t}$ to accurately track the *a priori* error; we also set $b = 0.9$ thus ensuring that $\sigma^2_{\epsilon,t}$ will converge to the *a posteriori* error. Experimental results in Section 6.4 show that the performance of the algorithm is not affected greatly by the specific choice of these parameters as long as the constraints are obeyed. We modify the self-tuning forgetting factor algorithm for use in the iS-PLS algorithm by computing e_t and ϵ_t using the PLS regression coefficients. The self-tuning forgetting factor is described in full in [70].

6.3.4 Detecting changes in the number of important latent factors

The weights determining the PLS latent factors can be time-dependent and the algorithm presented so far, coupled with an adaptive forgetting factor, is able to detect and adapt to changes over time. According to our Assumption 6.1, the number of the most important factors that are being retained for prediction is also allowed to change over time. As pointed out before, off-line cross-validation over a training period can be generally used to obtain an initial value R_0, prior to on-line learning.

In this section we describe how the algorithm adjusts R_t on-line. At each time step, we suggest to recursively update the mean prediction errors

$$E_t^{(r)} = \omega_t E_{t-1}^{(r)} + \|\boldsymbol{y}_t - \hat{\boldsymbol{y}}_t^{(r)}\|^2,$$

where $\hat{\boldsymbol{y}}_t^{(r)}$ is the prediction of $\boldsymbol{y}_t$ obtained at time $t-1$ using r latent factors. In our procedure, we use $r = R_t - 1, R_t, R_t + 1$, in order to determine when to decrease or increase the current value R_t by one. It can be noted that past errors are included in $\boldsymbol{E}_t^{(r)}$ but these are downweighted using the adaptive forgetting factor whilst more weight is given to the current prediction error term.

The error ratios $Q_t^{(R_t-1)} = E_t^{(R_t-1)}/E_t^{(R_t)}$ and $Q_t^{(R_t+1)} = E_t^{(R_t+1)}/E_t^{(R_t)}$ can then be computed. Values of these ratios above 1 indicate that the current R_t is appropriate and should not be changed. Conversely, values below 1 suggest that a change in R_t is needed, as this adaptation will improve the prediction performance. Accordingly, we define two truncated error ratios, $A_t^{(R_t-1)} = \min(1, Q_t^{(R_t-1)})$ and $A_t^{(R_t+1)} = \min(1, Q_t^{(R_t+1)})$, and trigger a change in R_t when either one deviates

substantially away from one.

6.4 Experimental results with simulated data

In this section we report on simulation experiments designed to demonstrate the performance of the sparse PLS regression algorithm as variable selection method in both the off-line case, where all the data has been observed, and the on-line case, where the data is assumed to arrive sequentially at discrete time points. Both univariate and multivariate responses will be considered.

6.4.1 Ability to track the important explanatory variables

First, we propose a simulation setting whereby groups of explanatory variables are generated from three distinct latent factors. In this setting we impose that, each time step, the response depends on only two out of the three existing latent factors, which we call here the *active* factors. This is obtained by setting the regression coefficients of the variables corresponding to the remaining group of variables, associated to the *inactive* factor, exactly to zero. The non-zero regression coefficients associated to the active variables are simulated so that one of the two sets of variables is more strongly correlated with the response than the other. A similar simulation framework has been described in [7].

A more precise description of the simulation scheme follows. In order to generate data that are evolving over time, the three hidden factors are assumed to follow an autoregressive (AR) process of first order, so that factor r is given by

$$f_t^{(r)} = \delta_j f_{t-1}^{(r)} + \epsilon_t,$$

with $t = 2 \ldots, 400$ and starting with an arbitrary initial value at $t = 1$. This is done independently for each $r = 1, \ldots, 3$. The parameter δ_j is the autoregressive coefficient for factor r, and we use $\delta_1 = 0.1$, $\delta_2 = 0.4$, $\delta_3 = 0.2$. The error terms in each factor follow a normal distribution with different means but same variance. We create three groups of data streams by collecting the indices representing the vari-

ables into three non-overlapping sets, $\mathcal{F}_1 = \{1, \ldots, 100\}$, $\mathcal{F}_2 = \{101, \ldots, 200\}$, and $\mathcal{F}_3 = \{201, \ldots, 300\}$. Each explanatory variable is then generated as

$$X_{t,p} = \begin{cases} f_t^{(1)} + \eta_{t,p}, & p \in \mathcal{F}_1 \\ f_t^{(2)} + \eta_{t,p}, & p \in \mathcal{F}_3 \\ f_t^{(3)} + \eta_{t,p}, & p \in \mathcal{F}_3, \end{cases}$$

for $t = 1, \ldots, 400$ and $p = 1, \ldots, 300$. Here η_t is a standard normal variable. Finally, the univariate response is simulated as

$$y_t = \boldsymbol{x}_t \boldsymbol{\beta} + \epsilon_t,$$

where the regression coefficients, $\boldsymbol{\beta}$ are sampled from normal distributions in the following way

$$\beta_p \sim \begin{cases} \mathcal{N}(10, 0.25), & p \in \mathcal{F}_1 \\ \mathcal{N}(5, 0.25), & p \in \mathcal{F}_2, \\ 0, & p \in \mathcal{F}_3, \end{cases}$$

so that the third factor does not contribute to the response. Using the simulated data streams, we are able to show that the off-line sparse PLS regression procedure accurately selects the correct active variables. In the on-line case, for which we propose an incremental soft-thresholding procedure, we are also able to demonstrate how, when the underlying data is stationary, the on-line solution converges to the off-line solution. The sparsity parameter, which determines how many variables are included in the regression model, is selected using the ten-fold cross-validation procedure described in Section 6.3.

Based on a Monte Carlo simulation study involving 500 data sets, the cross-validated mean number of active variables in the first component turned out to be 105.37 with a standard error of 19.50. We also compared the performance of our algorithm with that of two variable selection methods: standard Lasso regression (using the LARS algorithm of [28]) and Sparse PCA (S-PCA) using the sparse

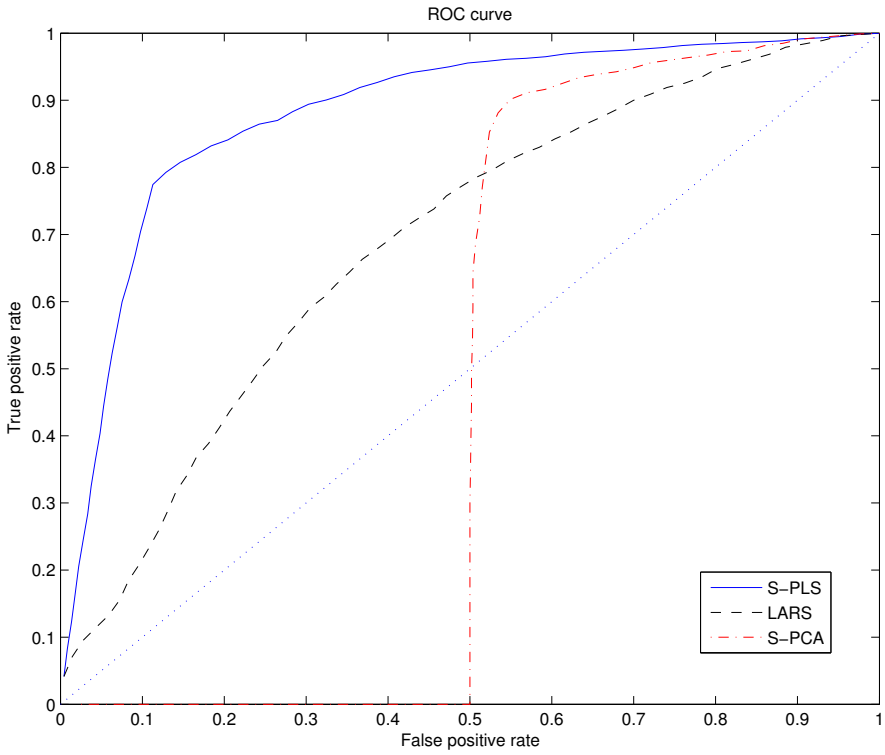


Figure 6.1: ROC curve obtained by S-PLS, LARS and S-PCA, reported are averages for a Monte Carlo simulation of 500 runs. For the simulated data, S-PLS (solid line) has a larger sensitivity for smaller values of specitivity than LARS (dashed line) which means it selects the correct variables and selects fewer incorrect variables. In the portion where S-PCA (dot-dashed line) has zero sensitivity, it is selecting the variables in X with the largest variance which by design are not the variables which are most correlated with the response.

SVD method of [80] described in Section 6.3. Figure 6.1 compares the ROC curve obtained by S-PLS with that of LARS and S-PCA. As before, we report on average results based on a Monte Carlo simulation consisting of 500 simulated data sets. The sensitivity measure is defined as the proportion of correctly selected variables, whereas the specificity measure is defined as the proportion of variables which are correctly assigned a zero coefficient and excluded from the model. It can be seen that S-PLS (solid line) performs better than LARS (dashed line) in cases, such as ours, where the explanatory variables are highly correlated. As expected, S-PCA (dot-dashed line) exhibits zero sensitivity for a portion of the curve. This is because S-PCA is selecting the variables in X whose projection explains a large proportion of the input variance; by construction, these variables are not those most correlated with the output. More generally, we expect S-PCA to exhibit much lower sensitivity than S-PLS except in the case where the variables associated with largest variance in the input are also able to explain a large proportion of variability in the output.

6.4.2 Convergence of the incremental soft-thresholding update

In order to test the convergence of the iS-PLS algorithm to the off-line algorithm, we consider the same simulation setting introduced in Section 6.4.1 and assume that all observations arrive sequentially. Figure 6.2 shows the result of a Monte Carlo simulation consisting of 500 runs of the on-line iS-PLS algorithm with a forgetting factor $\omega = 1$ (since the data stream is stationary and no forgetting is required). The shaded area shows the Monte Carlo error. It can be seen that the sensitivity of the on-line algorithm reaches its maximum value after observing 25 data points. These results, as well other extensive experimental results (not shown here) suggest that, in the presence of stationary data, the iS-PLS solution converges to the solution found by iterative soft-thresholding in an off-line setting only after a brief learning period. In the next section we show how the on-line algorithm is also able to adapt to changes.

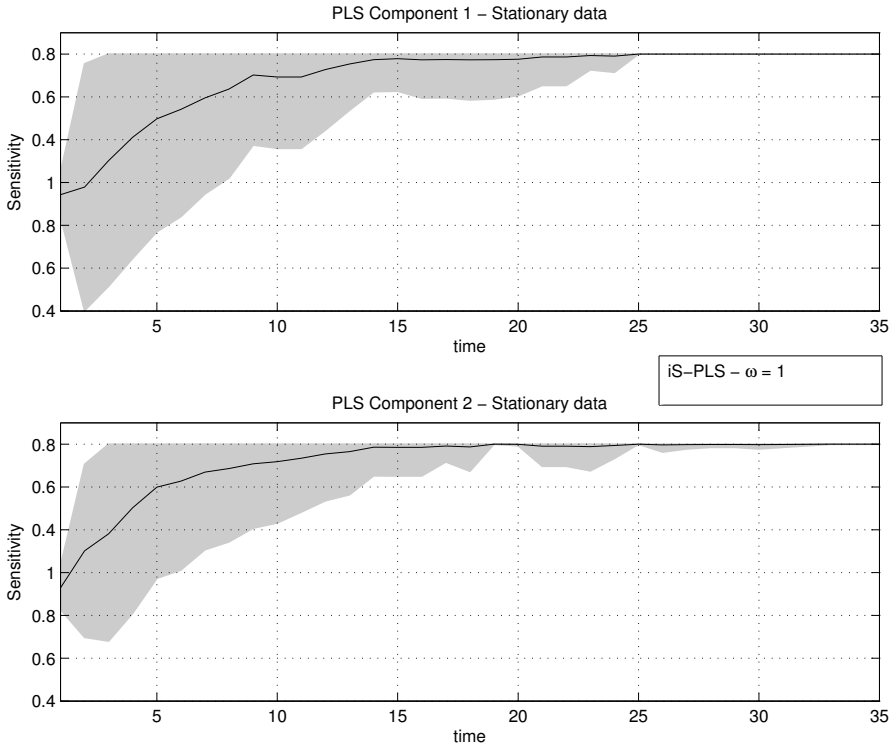


Figure 6.2: Sensitivity of the iS-PLS algorithm. The shaded area shows the Monte Carlo error of the sensitivity index. It can be seen that after $t = 25$ data points, the iS-PLS result achieves maximum sensitivity and converges to the same solution obtained by off-line learning, when all data are available.

6.4.3 Ability to adapt to changes

In order to test the adaptive behaviour of the iS-PLS algorithm in high dimensions, we generated a non-stationary output by introducing time-dependent regression coefficients, according to the following scheme: For $t = 1, \ldots, 100$,

$$\beta_{t,p} \sim \begin{cases} \mathcal{N}(10, 0.25), & p \in \mathcal{F}_1 \\ \mathcal{N}(5, 0.25), & p \in \mathcal{F}_2 \\ 0 & \text{otherwise.} \end{cases}$$

For $t = 101, \ldots, 300$,

$$\beta_{t,p} \sim \begin{cases} \mathcal{N}(5, 0.25), & p \in \mathcal{F}_1 \\ \mathcal{N}(10, 0.25), & p \in \mathcal{F}_2 \\ 0 & \text{otherwise.} \end{cases}$$

For $t = 301, \ldots, 400$,

$$\beta_{t,p} \sim \begin{cases} \mathcal{N}(5, 0.25), & p \in \mathcal{F}_2 \\ \mathcal{N}(10, 0.25), & p \in \mathcal{F}_3 \\ 0 & \text{otherwise.} \end{cases}$$

In this way, the important predictors change over time and we expect these changes to be picked up in real-time by the algorithm.

The coefficient values are represented graphically in Figure 6.3: in plot (a) the black shaded area represents variables with a large coefficient, the gray shaded area represents variables with a smaller coefficient and the unshaded area represents variables with a zero coefficient (inactive variables). In this setting, we set $R = 2$ and the sparsity parameters $\gamma^{(r)}$ are chosen so that, at any given time, exactly 100 variables are selected. The forgetting factor ω is updated to ensure a rapid adjustment when the coefficients switch while also keeping the switching frequency low when the data is stationary to gain stability in the selected variables. Figure 6.3 also shows the results of a single run of this experiment. Clearly, the first PLS component is

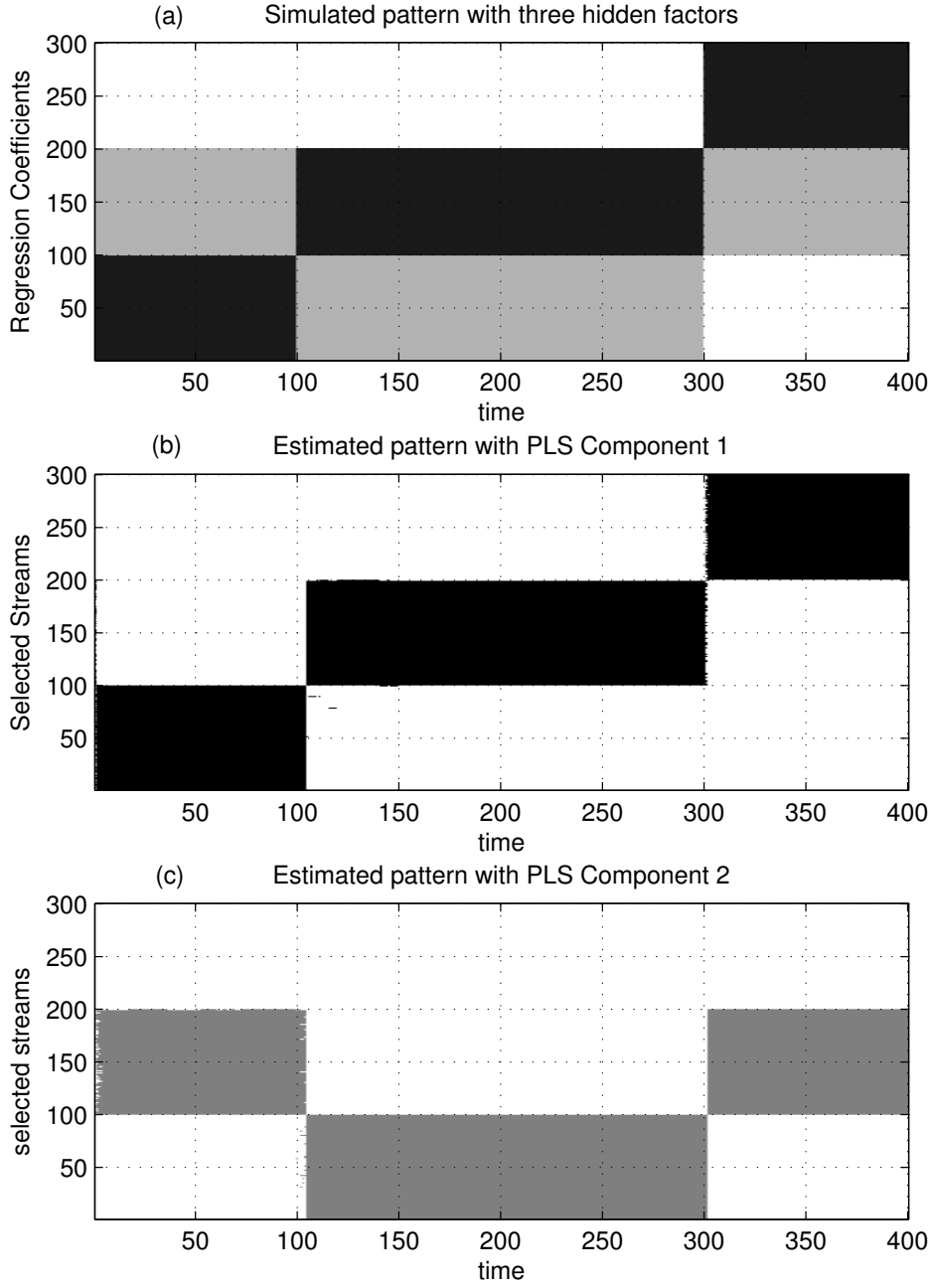


Figure 6.3: Figure (a) shows a block-wise representation of input data streams: active streams having larger coefficients (black blocks), smaller regression coefficients (gray blocks), and inactive streams (white blocks) which only contributes to noise. Each block is related to a different hidden factor. Figures (b) and (c) shows the data streams selected on-line by the first and second PLS component, respectively in a typical run. Noise occurs in the solution between $t = 0$ and $t = 10$ which occurs before the iS-PLS solution has converged as described in Figure 6.2. There is also noise present when the factors switch at $t = 100$ before the algorithm adjusts.

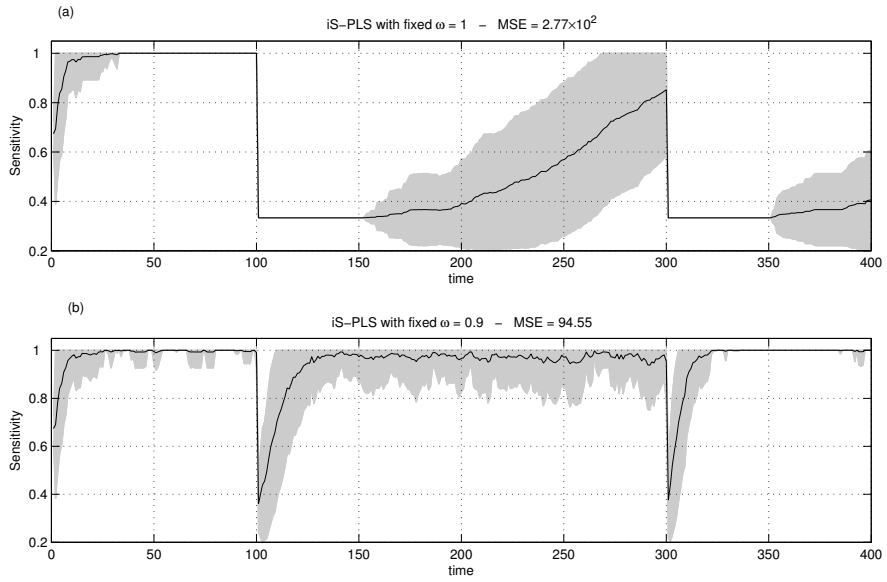


Figure 6.4: Sensitivity of iS-PLS for different values of ω averaged over 500 Monte Carlo simulations where the shaded area shows the Monte Carlo error. In plot (a), $\omega = 1$ which causes the solution to converge with low variance up until the coefficients change. It then is not able to quickly or accurately adjust to the new important factors. In plot (b), $\omega = 0.9$ which allows the solution to rapidly adjust to changes, however there is a lot of variance in the solution during periods when the coefficients are not changing due to the small forgetting factor. Despite the large variance, when $\omega = 0.9$ the algorithm achieves greater sensitivity throughout the simulation which achieves a smaller prediction error as more of the correct variables are being selected compared to the case where there is no forgetting.

able to accurately select the most important group of variables. The second component always selects the second most important group of variables whilst mostly ignoring the group of variables selected by the first component. Neither component selects the inactive variables suggesting the algorithm is correctly able to distinguish important predictors from noise. As the regression coefficients switch, the algorithm only requires few data points before it detects the changes and adapts.

Figure 6.4 reports on the mean sensitivity of the iS-PLS algorithm in a Monte Carlo simulation consisting of 500 runs of this experiment for different values of ω. The solid line shows the mean percentage of correctly selected variables by the first and second PLS components. The shaded area shows the Monte Carlo error. This

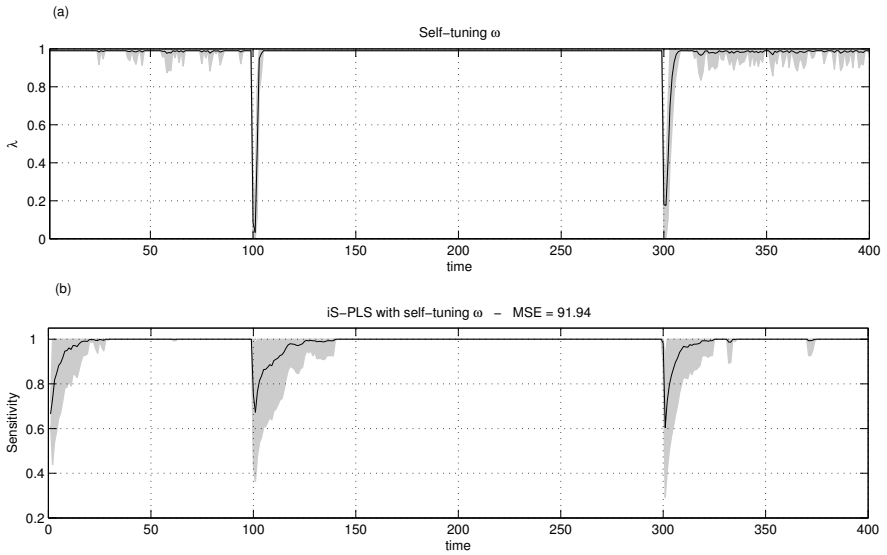


Figure 6.5: Sensitivity of iS-PLS when using a self-tuning ω averaged over 500 Monte Carlo simulations where the shaded area shows the Monte Carlo error. Plot (a) shows how the value of the self-tuning forgetting factor changes over time. Plot (b) shows that when ω is dynamically adjusted using the variable forgetting factor algorithm, the solution adjusts rapidly and with little error to changes in the coefficients. During periods when the data is stationary, the variance in the solution is small.

result clearly illustrates the limitations of using a fixed ω. Plot (a) shows the result when $\omega = 1$. Here, during the first stationary period, the selected components are stable. However, when the coefficients suddenly transit to another stationary period, the algorithm adapts very slowly. Plot (b) shows the result when $\omega = 0.9$. Although the algorithm is able to adapt quickly to changes in the coefficients, during stationary periods the estimated coefficients have higher variance. This can be seen by observing the larger Monte Carlo error during the periods of stationary data. However, the reported Mean Squared Error (MSE) shows that, despite the higher variability, the ability to adapt to changes in the data (using a forgetting factor $\omega = 0.9$, in this case) results in a smaller prediction error.

Figure 6.5 reports on the sensitivity of the iS-PLS algorithm when using the adaptive forgetting factor introduced in Section 6.3.3. Plot (a) shows the value of ω over time as a result of a self-tuning algorithm. During stationary periods, ω

is kept very close to 1, as we would expect. When the coefficients "jump", ω is automatically set to a very small value for a short period of time and then set back to its previous value. This allows iS-PLS to adjust very quickly to sudden changes in the data whilst allowing a low-variance solution during stationary periods. Plot (b) shows the sensitivity measure obtained by iS-PLS when ω is adjusted using the self-tuning forgetting factor and takes the values reported in Plot (a). It is clear that in the stationary periods, iS-PLS correctly selects the important variables with very little error. In response to a change in the important factors, the percentage of correctly selected variables instantly decreases and quickly adapts to the new data. The algorithm eventually selects the correct variables after a short settling time. However, during this time the estimation variance increases. The reported MSE (91.94) shows that the ability to dynamically adapt the forgetting factor results in a smaller prediction error compared with the fixed ω case (Figure 6.4).

In order to evaluate the performance of the iS-PLS algorithm in a more challenging setting, we devised a simulation where the number of change points is a random outcome governed by a geometric distribution with a parameter 0.1, so that the mean number of change points is 10 over a period of 1000 data points. When a change point occurs, the active variables also switch randomly. Furthermore, we restrict the number of variables which contribute to the response to be some percentage of the active variables, ranging between 1% (3 variables) and 30% (90 variables). In this case, many of the variables are effectively noise. Our results are benchmarked against the Recursive LARS (R-LARS) algorithm of [48] and the adaptive Lasso (aLasso) algorithm of [6]. In order to make the comparison meaningful and simple to interpret, we let the number of variables to be selected by each algorithm to be equal to the true number of active variables.

In Figure 6.6, Plot (a) reports on the mean sensitivity, averaged over 1000 data points and over 500 Monte Carlo simulations. The shaded, colored areas represent the Monte Carlo errors for each competing algorithm. It can be seen that iS-PLS always achieves higher sensitivity compared to R-LARS or aLasso, even when the number of active variables is small. This suggests that the ability to identify important latent factors is beneficial, especially in cases where a large number of the

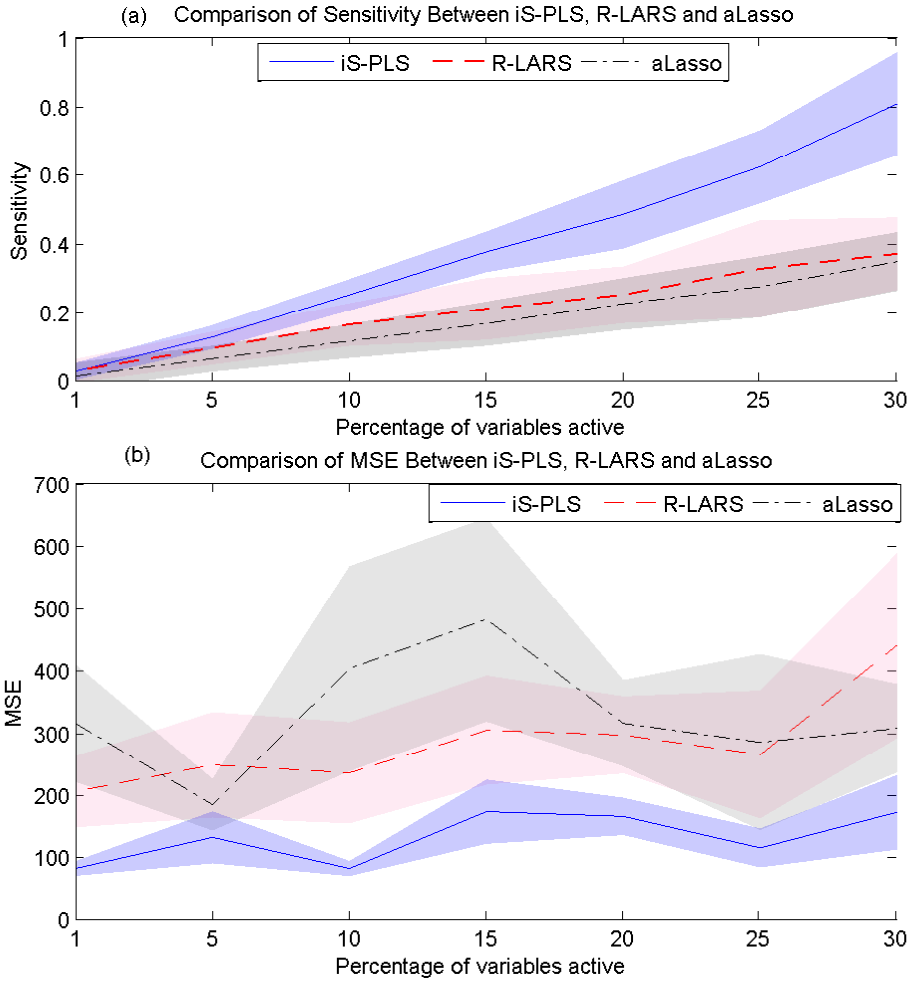


Figure 6.6: Comparison of mean sensitivity (a) and MSE (b) over 1000 data points between iS-PLS, R-LARS and aLasso averaged over 500 Monte Carlo simulations where the shaded areas show the Monte Carlo error. (a) shows that for fewer active variables in the simulation, iS-PLS achieves a higher sensitivity than R-LARS and aLasso. (b) shows that iS-PLS achieves a smaller MSE with lower variance than R-LARS and aLasso however the error is not greatly affected by an increase in the sensitivity, which suggests that iS-PLS is better suited to situations where there are many correlated variables.

variables only contribute to noise and do not affect the response. Furthermore, these results illustrate the ability of iS-PLS to adapt to changes in the latent factors. As the number of variables increases, iS-PLS approaches quickly reaches maximum sensitivity, whereas the other methods do not perform well. Plot (b) reports on the corresponding MSE. The MSE achieved by iS-PLS is smaller, and has lower variance, compared to other methods; however the MSE is not greatly affected by an increase in the sensitivity.

6.4.4 Sensitivity analysis

In order to study how the algorithm depends on the hyper-parameters, we carried out a sensitivity analysis. First, we examined the effect of α which regularises the covariance matrix so that the full PLS solution may be extracted using only one SVD. Based on 500 Monte Carlo replicates, when α varies from 10^{-5} and 10^{-1}, the corresponding average MSEs (and their standard deviations) were $1.551(2.253)$ and $1.624(2.393)$, respectively; when α takes larger values, such as 0.5 and 1, the MSE gets as large as $7.899(32.551)$ and $32.331(187.559)$, respectively. Clearly, as α is kept very small, there is very little effect on the overall prediction error; however, as the solution becomes closer to the PCR solution, the MSE and standard deviation increase dramatically. Every value of α tested yields a covariance matrix $\boldsymbol{G}$ which is always of full rank; hence it is always sensible to choose the smallest value of α to achieve a solution as close as possible to the true PLS solution.

Secondly, we look at the effect of the hyper-parameters a and b controlling the effective memory length in the self-tuning forgetting factor algorithm. Figure 6.7 shows the sensitivity of the iS-PLS algorithm using 500 Monte Carlo simulations as a function a and b, for $b \leq 1$ and $a \leq b$. The overall sensitivity ranges between 0.80 and 0.91. When $b > a$, the range of sensitivity is between 0.85 and 0.91. In the case where $a = b$, the *a priori* error and the *a posteriori* error in Eq. (6.15) become equal which prevents ω_t from adapting. Similarly when $b = 1$, the estimate for the *a posteriori* error does not discount past data which in turn stops ω_t from adapting. At these extreme cases, the algorithm achieves its lowest sensitivity of

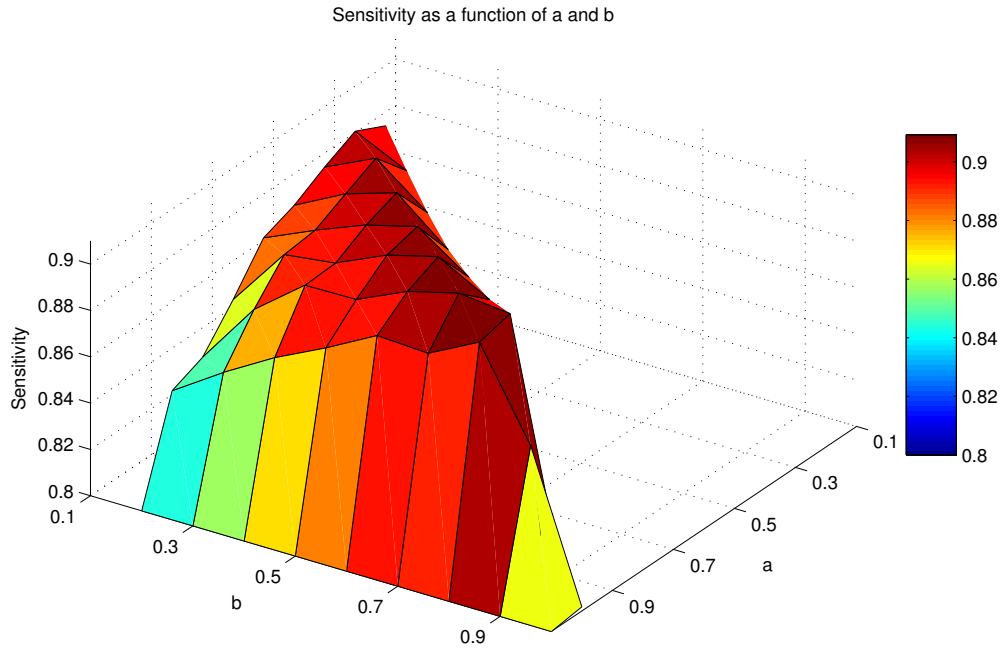


Figure 6.7: The mean sensitivity of the iS-PLS algorithm after 500 Monte Carlo simulations as a function of the hyper-parameters a and b which control the memory length in the forgetting factor algorithm. It can be seen that the algorithm is not very sensitive to changes in the values of these parameters provided $1 > b > a$. No self-tuning takes place at the extreme values when $b = 1$ and $b = a$ and so the algorithm achieves minimum sensitivity of 0.80. Maximum sensitivity of 0.91 occurs around $a = 0.5$, $b = 0.9$.

0.80. In our simulations we set $a = 0.5$ and $b = 0.9$ which lies close to the region of maximum sensitivity; we expect this to be the case because, in order to obtain an accurate estimate of the *a posteriori* error, we must use a long exponential time window whereas the *a priori* error is an instantaneous estimate of the current error and can be obtained using an extremely short window.

To conclude our sensitivity analysis, we demonstrate how the sparsity parameter and the noise level affect the performance. Figure 6.8 contains a heatmap of the mean sensitivity of the S-PLS algorithm as a function of both the number of selected variables and the signal-to-noise ratio. For simplicity, we only consider the first latent factor. It clearly emerges that when the signal-to-noise ratio is high, the

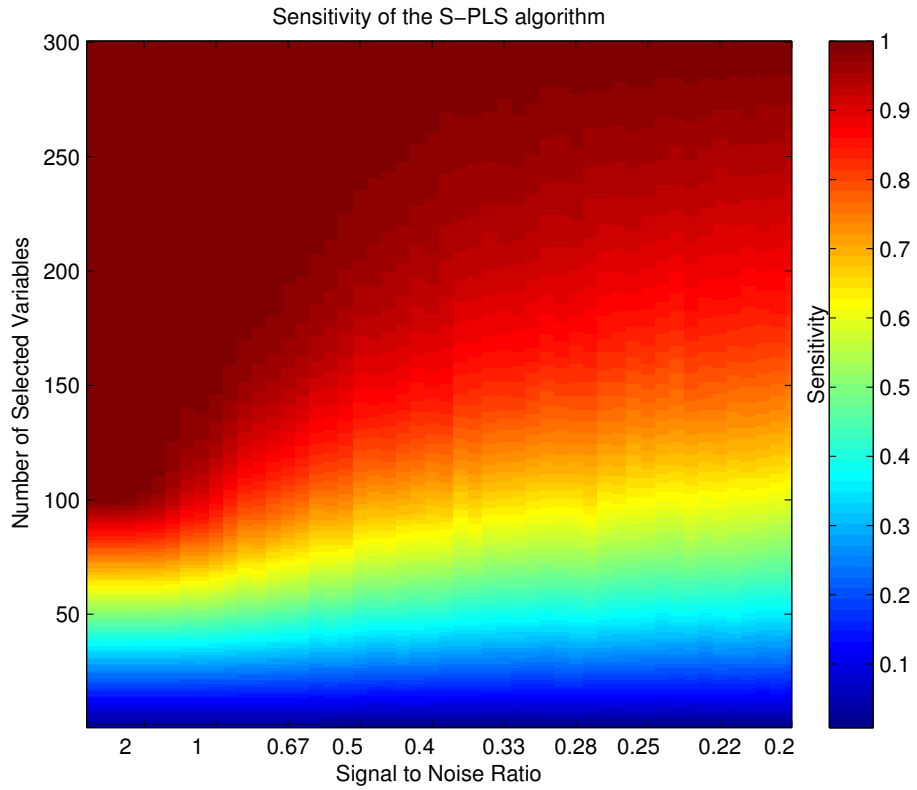


Figure 6.8: Sensitivity of the S-PLS algorithm as a function of the number of variables selected and the signal to noise ratio in the first latent factor over 500 Monte Carlo simulations. When the signal to noise ratio is high, S-PLS achieves maximum sensitivity when approximately 100 variables are selected (where 100 is the true number of active variables). As the noise increases to the point where the variance of the noise is four times that of the signal, the sensitivity decreases so that maximum sensitivity is only achieved when all variables are selected.

algorithm always correctly selects variables from the active group when less than 100 variables are selected (the true number of active variables in the first latent factor). As the noise level increases, the sensitivity of the algorithm decreases to the point where maximum sensitivity is only reached when all 300 variables are selected; this implies that S-PLS is no longer able to distinguish the active variables from the inactive variables. The optimal degree of sparsity should be selected using cross-validation and, following our Assumption 6.2, will be kept fixed over time.

6.4.5 Performance with high-dimensional responses

In order to test the ability of iS-PLS to track important variables in a setting where multiple correlated responses are generated by the same underlying processes, we set up the following simulation. Two independent latent factors $\boldsymbol{f}^{(1)}$ and $\boldsymbol{f}^{(2)}$ each with $T = 400$ time points were simulated from a bivariate normal distribution with means $\mu^{(1)} = 3$ and $\mu^{(2)} = 5$ and identity covariance. Two $\boldsymbol{Y}$-loading vectors $\boldsymbol{q}^{(1)}$ and $\boldsymbol{q}^{(2)}$ of length 50 were sampled from a standard normal distribution and orthogonalised with respect to each other using the QR decomposition. The 50 responses were generated as

$$\boldsymbol{Y} = \sum_{r=1}^{2} \boldsymbol{f}^{(r)} {\boldsymbol{q}^{(r)}}^{\top} + \boldsymbol{\epsilon},$$

where the errors are i.i.d. standard normals. Two time-varying $\boldsymbol{X}$-loading vectors, $\boldsymbol{u}_t^{(1)}$ and $\boldsymbol{u}_t^{(r)}$ each with $P = 300$ variables were generated such that the non-zero elements in each vector were simulated in the following way: for $t = 1, ..., 100$, $u_{t,p}^{(1)} \sim \mathcal{N}(10, 0.25)$ when $p \in \mathcal{F}_1$ and $u_{t,p}^{(2)} \sim \mathcal{N}(5, 0.25)$ when $p \in \mathcal{F}_2$. All the non-zero loadings in the two groups of active variables are swapped at $t = 101$ and a final change of variables occurs at time $t = 301$. The predictor variables were then generated as

$$X_{t,p} = \begin{cases} f_t^{(1)} \boldsymbol{u}_t^{(1)} + \boldsymbol{\eta}_t, & p \in \mathcal{F}_1 \\ f_t^{(2)} \boldsymbol{u}_t^{(2)} + \boldsymbol{\eta}_t, & p \in \mathcal{F}_2, \end{cases}$$

where $\boldsymbol{\eta}_t \in \mathbb{R}^{1 \times P}$ is standard normal noise. In this way, only two groups of variables are correlated with the response through the latent factors, f_t at any one time.

We again compare the performance of iS-PLS with R-LARS and aLasso. Since these are univariate techniques, we run R-LARS and aLasso for each response separately and obtain the mean result. Figure 6.9a shows the mean sensitivity of iS-PLS compared with R-LARS and aLasso as a result of 500 Monte Carlo simulations. It can be seen that iS-PLS is able to accurately select the important variables during periods where the loading coefficients are stationary, achieving close to maxi-

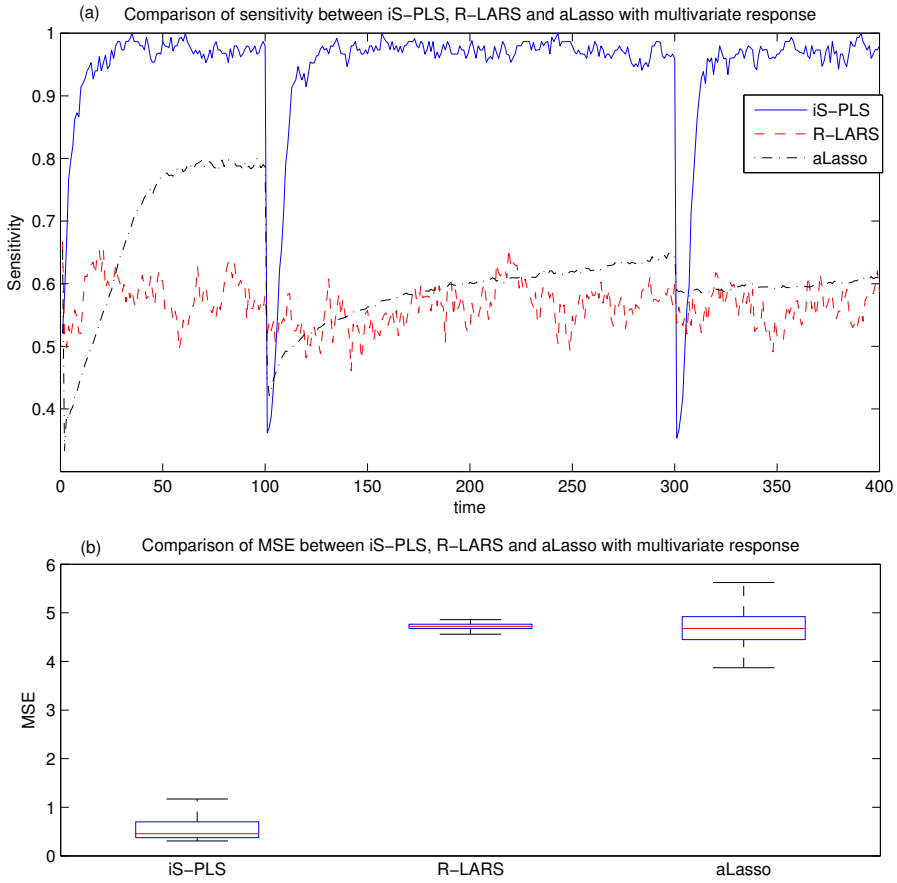


Figure 6.9: (a) reports on a comparison of mean sensitivity between iS-PLS, R-LARS and aLasso averaged over 500 Monte Carlo simulations in a setting where there are 50 correlated responses. iS-PLS achieves close to maximum sensitivity. As the important variables change, iS-PLS is able to quickly adapt. Neither R-LARS or aLasso are able to uncover the important variables as they do not assume the existence of latent factors and are unable to use information in all 50 responses simultaneously. (b) reports on the distribution of MSE of the different methods over the Monte Carlo simulations. The iS-PLS algorithm achieves a consistently lower error than either R-LARS or aLasso.

mum sensitivity. The iS-PLS algorithm is able to quickly adapt when the loadings change. Both R-LARS and aLasso achieve much lower sensitivity than iS-PLS and are much slower to adapt in response to changes in the active latent factor. This is expected as neither method is able to estimate the latent factors which affect the important variables. Figure 6.9b shows the distribution of mean squared errors for

the three methods. It can be seen that iS-PLS achieves a consistently lower error than either of the univariate techniques which is expected considering the accuracy of the tracking displayed by iS-PLS compared with R-LARS and aLasso.

We also report on the mean computational time required to run each algorithm once for 50 responses averaged over the same 500 Monte Carlo simulations. A key advantage of iS-PLS is its ability to handle high dimensional inputs and responses simultaneously with relatively low computational cost. For a univariate response, the computational complexity of R-LARS and aLasso per time point is $O(P)$ and $O(P^2)$ respectively, compared with a complexity of $O(R^2P)$ for iS-PLS. However, iS-PLS has a further advantage in that it can handle multivariate response with no further increase in complexity whereas both other methods will scale linearly in the number of responses. In this setting iS-PLS took 8.31s whereas R-LARS took 556.9s and aLasso took 2391s. These timings confirm that iS-PLS is significantly faster than both other methods when dealing with multivariate responses.

6.4.6 Ability to track the number of important latent factors

Finally, we test the ability of iS-PLS to adaptively tune the number of latent factors R. Using the same simulation framework, we simulate a third uncorrelated latent factor in the same way as before and simulate loadings such that at time $t = 1, ..., 100$, only the first loading contains non-zero elements. At time $t = 101, ..., 300$, the first two factors contain non-zero elements and at $t = 301, ..., 400$ all three loading vectors contain non-zero elements.

In Figure 6.10, plot (a) shows the mean estimated value of R_t through time compared with the true value of R_t. It can be seen that the iS-PLS can quickly and accurately adjust the number of latent factors active in the data as R_t increases from 1 to 3 and then decreases again. Plot (b) shows the mean value of the truncated error ratios $A_t^{(R_t-1)}$ and $A_t^{(R_t+1)}$. A sufficient negative gain in either ratio indicates when the algorithm should increment or decrement the value of R_t. As more factors are added, the resulting negative gains in the error ratios gets smaller. This is because each subsequent latent factor added by iS-PLS accounts for progressively less of the

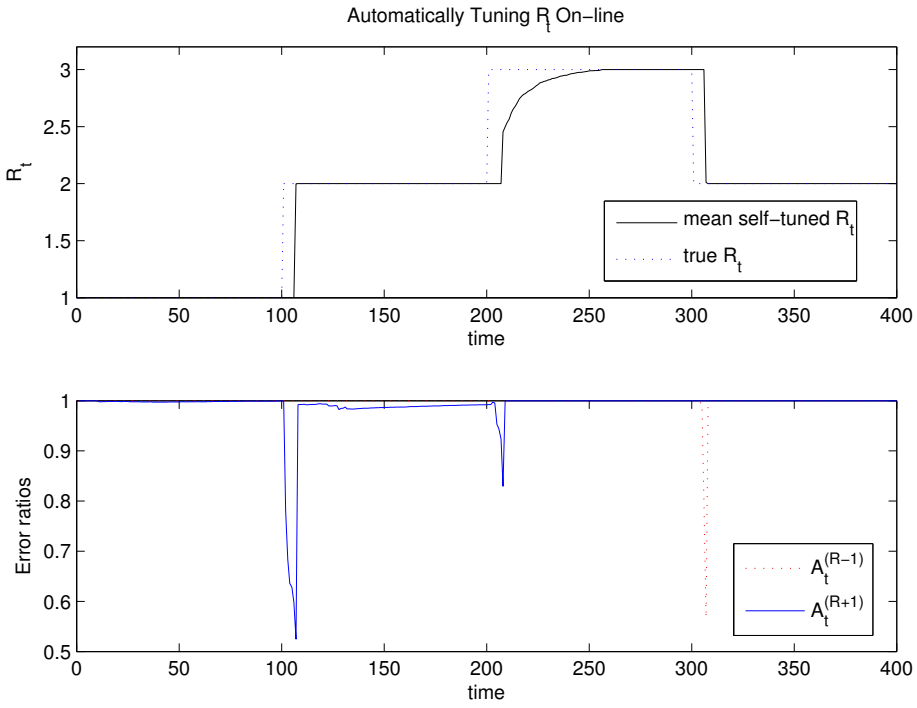


Figure 6.10: Self-tuning the number of latent factors. (a) shows how the mean value of R_t is adjusted in response to changes in the number of latent factors in the data over 500 Monte Carlo simulations. Starting with one important latent factor, at $t = 100$ and $t = 200$ we increment the number of important latent factors by 1 and at $t = 300$ we decrement the number of latent factors by 1. (b) shows the truncated error ratios which determines when to update R_t. iS-PLS is able to quickly adapt as the number of latent factors in the data changes.

total variance between the explanatory variables and the response. This accounts for the slower adjustments made when R_t increases from 2 to 3.

6.5 An application to index tracking

We have applied the iS-PLS algorithm to the problems of both univariate and multivariate index tracking. A financial index consists of a portfolio of constituent stocks whose daily price is determined as a weighted sum of the prices of those constituents. A commonly used measure of performance of the index is the daily index return, y_t which is defined as the percentage gain (or loss) of the index price

each day. Similarly, the daily portfolio returns, $\boldsymbol{x}_t$, are the corresponding daily percentage gains in the individual stock prices. The makeup of an index is determined by a number of factors such as market capitalization and asset price and as such, stocks may be added or removed from an index based on whether they meet the criteria for inclusion in the index. Similarly, the weights assigned to each asset are also frequently adjusted. These changes in the index cause its returns to be non-stationary.

There are two interconnected problems associated with index tracking: *asset selection* and *asset allocation*. Asset selection involves selecting a subset of p out of P available assets to construct a tracking portfolio. The asset allocation problem involves investing a proportion of the total available capital in each one of the p selected assets by estimating the index weight assigned to that asset. The overall goal of performing index tracking is to reproduce as accurately as possible the returns of the index. The constituents and weights of the tracking portfolio can be selected by attempting to minimise the *tracking error* [66], that is the error between the index returns y_t and the tracking portfolio returns $\hat{y}_t$,

$$\frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{y}_t)^2, \tag{6.16}$$

where T is the period over which the returns are observed. Following this setting, the problem of asset allocation becomes a standard regression problem with the portfolio weights being the parameters to be estimated. Traditionally, the problem of selecting a small number of assets from a large index such as the S&P 500 has been performed using search algorithms such as simulated annealing [8, 35]. However, this is prohibitively expensive. Therefore we must look at methods of automatically selecting assets at low computational cost. Automatic asset selection is not a prevalent topic in the literature with most methods either falling into the category of full-index replication (no asset solution), computationally intensive search or proprietary algorithms.

Our motivation for using a sparse PLS algorithm for index tracking is that a

number of studies of financial markets have suggested the existence of latent factors. For instance, [5, 4] use standardised principal component weights as portfolio weights. They motivate their approach by providing evidence that the principal eigenvector of the covariance matrix of asset returns generally captures the underlying *market factor*. Latent factor models related to principal components have also been suggested for the construction of an index tracking portfolio by [78] and [39]. However, to our knowledge PLS has not been used in the context of financial index tracking. Given the evidence for a latent factor underlying the market, we expect that PLS is better suited for index tracking applications as it takes into account the covariance between the constituent asset returns and the index returns as opposed to just the asset returns as in PCA. This means PLS should identify the assets which are contributing most importantly to the variation in the index returns.

For this application, we have used publicly available data sets from the S&P 100 and Nikkei indices, as previously described in [8]. The S&P 100 index has 98 constituents and the Nikkei has 225. To motivate the need for an incremental variable selection algorithm for index tracking, we start by presenting an example of tracking with two off-line variable selection methods, our Sparse PLS algorithm using one latent factor and the LARS algorithm of [28]. We also compare these methods to an Ordinary Least Squares (OLS) fit which does not perform variable selection. We performed *enhanced* tracking of the S&P 100 index where the aim is to overperform the index returns by a certain percentage. This is achieved by creating a new target asset by adding a percentage of its annual returns to the index, we then aim to track this *enhanced* index [3]. We enhance the S&P 100 by an additional 15% annual returns.

Figure 6.11 shows the in-sample results of enhanced tracking of the S&P 100 index using a static portfolio of 10 stocks selected from 98 using S-PLS and LARS and full index replication using OLS. Despite having access to all of the data, it is clear that using a static portfolio for a long period of time leads to poor tracking performance and in all three cases the artificial portfolios underperform the index. This result confirms that a scheme for periodically rebalancing the portfolio is necessary to produce better tracking performance. The OLS tracking performance exhibits

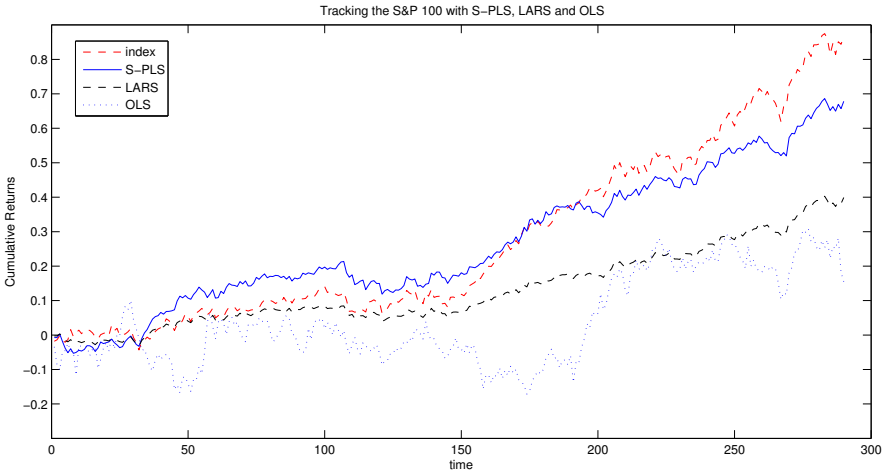


Figure 6.11: Comparison of enhanced tracking (+15% annual returns) of the S&P 100 index using a static portfolio of 10 stocks chosen out of 98 using S-PLS and LARS. S-PLS performs well during the first half of the tracking period, however as the entire period is used for training, a change in the "market factor" at around $t = 150$ is not picked up by the off-line S-PLS algorithm which causes tracking performance to degrade over the second half of the tracking period. Similarly, the performance of LARS gets worse after $t = 150$. The OLS portfolio exhibits poor tracking performance with high variance due to the correlated nature of the asset returns.

high variance due to the many correlated data assets which further suggests that a latent factor approach which represents correlated assets using a single factor is appropriate.

We tested the iS-PLS algorithm in the multivariate case where two indices, the S&P 100 and the Nikkei, are simultaneously tracked. Both benchmark indices have been *enhanced* by adding 15% annual returns as previously described. The total combined number of available stocks is 323 and we set the portfolio size to 10. We constrain the selected stocks to be associated to the main latent factor only, so that $R = 1$, as in suggested in [5]. In order to assess whether iS-PLS selects the important variables over time more accurately than simple random guessing, we compared its performance with the average returns obtained from a population of 1000 random portfolios of the same size, with each portfolio being made of a randomly selected subset of assets. To make sure that the comparison is fair, the

portfolio weights are also time-varying and are obtained by using a RLS method with the same ω parameter. Figure 6.12 shows the results of this test. It can be seen that iS-PLS consistently overperforms both indices and selects a small portfolio achieving the target annual returns of $+15\%$. In comparison, the mean returns of the 1000 random portfolios underperforms the S&P index by 32.07% and the Nikkei by 8.42%.

Our empirical results suggest that the importance of certain stocks in the index is not constant over time so the ability to detect and adapt to these changes is certainly advantageous. Using a model that assumes a time-varying latent factor driving the asset returns is also advantageous in this setting, since its existence in real markets has been heavily documented in the financial literature. Figure 6.12c is a heatmap illustrating how the make-up of the portfolio selected by iS-PLS changes during the entire period. Specifically, it shows the existence of a few important stocks that are held for the majority of the period whereas other assets are picked and dropped more frequently throughout the period, further suggesting that it is advantageous to be able to adapt the constituents of a tracking portfolio.

Finally, we also compared the iS-PLS algorithm against the R-LARS and aLasso algorithms. The purpose of this comparison is again to determine whether assuming a time varying latent factor which explains covariance between the data and the response is beneficial compared to two on-line variable selection techniques which do not make this assumption. Figure 6.13 compares the performance of the iS-PLS algorithm against that of R-LARS and aLasso for the same bivariate index tracking problem. Since both R-LARS and aLasso are univariate techniques, they cannot perform bivariate index tracking. Therefore we compared iS-PLS performing bivariate index tracking against univariate tracking performed by R-LARS and aLasso on each of the indices separately. We set the portfolio size for each R-LARS and aLasso portfolio equal to 10. It can be seen that R-LARS and aLasso underperforms both indices suggesting that assuming a latent factor which explains the important covariation between the asset returns and the index returns is beneficial to performing accurate enhanced index tracking. It can be seen that aLasso performs slightly better than R-Lars which suggests that the use of a self-tuning

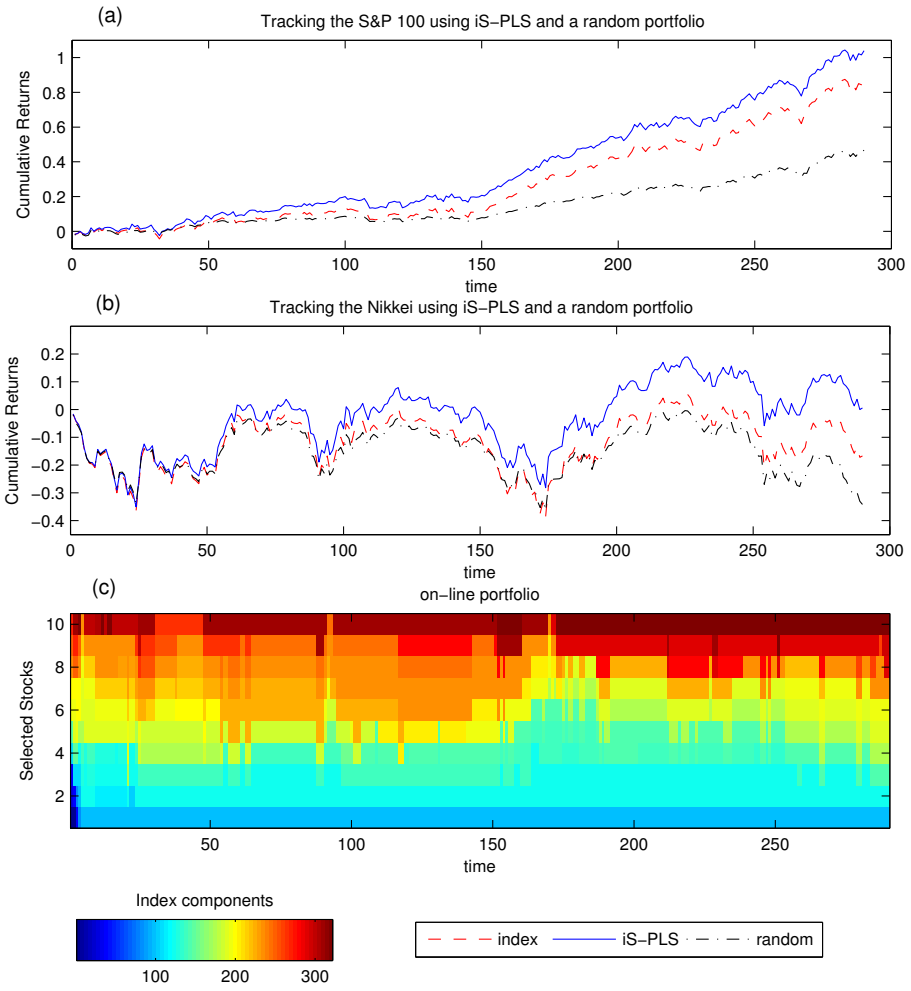


Figure 6.12: A comparison between iS-PLS and an averaged random portfolio performing bivariate enhanced tracking (+15% annual returns) of the S&P 100 (a) and Nikkei (b) indices using a dynamic portfolio of 10 stocks out of 323. The dashed red line shows the returns of the index over the tracking period. The solid blue line and the dashed black shows the returns of the iS-PLS and random portfolios, respectively. For both indices, iS-PLS achieves the target returns whereas the random portfolio underperforms the index. (c) shows how the stocks selected by iS-PLS change over the tracking period, notably at $t = 150$ in response to the S&P 100 and at $t = 95$ and $t = 110$ in response to the Nikkei.

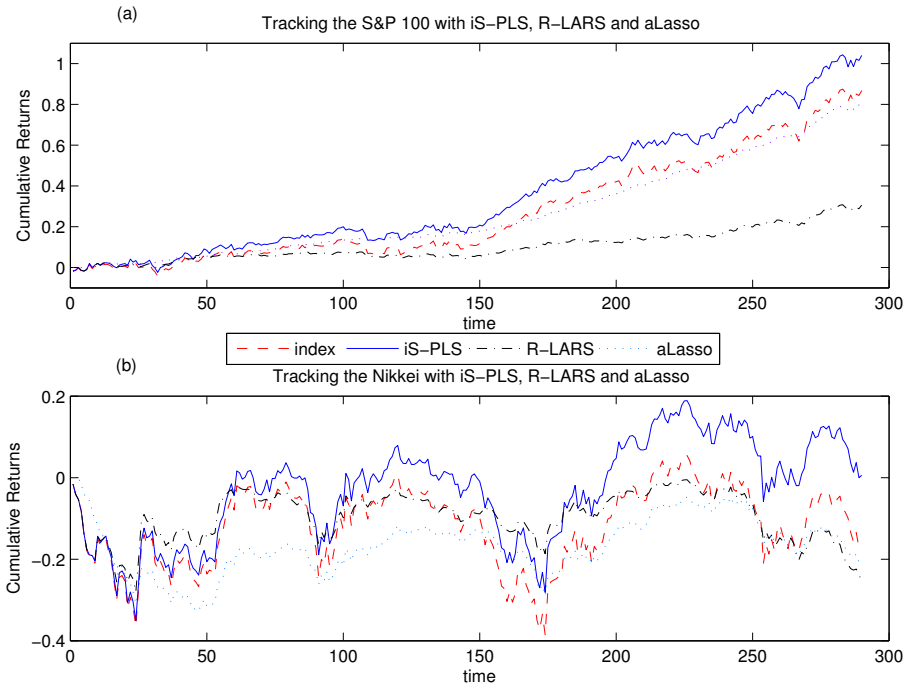


Figure 6.13: A comparison of bivariate enhanced tracking (+15% annual returns) between iS-PLS, Recursive LARS (R-LARS) and adaptive Lasso (aLasso) tracking the S&P 100 (a) and Nikkei indices (b). R-LARS and aLasso underperform both indices which suggests that the ability of iS-PLS to identify latent factors which contribute to the response is important in performing index tracking.

forgetting factor is also beneficial to performing accurate tracking however it is not as important as the ability to identify the important latent factors.

6.6 Discussion

In this chapter we have presented an efficient on-line learning algorithm for variable selection in a multivariate regression context based on streaming data. The proposed method can be interpreted as an on-line and adaptive version of two-block PLS regression with sparsity constraints. As far as we are aware, this is the first such algorithm which combines dimensionality reduction and variable selection for data streams in a unified framework. iS-PLS has advantages, over other incremen-

tal algorithms, in that it is able to deal with high dimensional and multicollinear data. The algorithm is also able to self-adjust some essential parameters, such as the number of important latent factors that need to be retained to obtain accurate predictions, and the forgetting factor needed to detect changes in the underlying data generating mechanism. Extensive simulation results show that the algorithm is able to accurately detect and track the important variables that should enter the regression model, even when sudden changes occur in the data. It also outperforms competing methods for on-line variable selection that do not assume the existence of latent factors.

The iS-PLS algorithm requires the specification of some parameters. Of these, the ridge parameter α and the forgetting factor memory length parameters a and b have been shown not to play a critical role in the performance of the algorithm. The iS-PLS algorithm is then able to accurately track any time-varying change in the covariance matrices and self-adjust the number and weights of latent factors. The user is only required to select the initial number of important PLS components to be retained. Prior to on-line learning, this value may be selected using domain-specific knowledge or prior information. For instance, in the financial index tracking application, it is well known that the first latent factor is representative of the entire market [5]; in other areas, such as genomic [103] and chemical process control [22], the selection of R is also guided by external information. Alternatively, the optimal R can be obtained using cross-validation [52].

In the existing literature, other approaches for on-line tracking of latent factors assume that the number of factors is fixed and these are updated in time without removing or adding new factors [106, 101, 14]. Recently there have been some attempts to update R_t on-line in the context of PCA; for instance, in [71, 80], the proportion of variance explained by the current number of factors is estimated and drives the tuning of R_t. However, these approaches are not immediately applicable to PLS regression. Other than tacking the PLS weights, our proposed iS-PLS algorithm is also able to improve on the prediction error by decreasing and increasing R_t as needed to achieve good predictive performance; this is obtained by means of an on-line approximation to cross-validation.

The only parameters that are assumed to be time-independent are $\{\gamma^{(r)}\}_1^R$, which controls the degree of sparsity in the PLS solutions. As with R, these parameters can be either selected using domain-specific information or, prior to on-line learning, using cross-validation. For instance, [96] present a cross validation method to select the sparsity parameter which achieves the lowest mean squared prediction error; however, they also suggest that some parameter tuning is done by the user based on the number of desired variables. Recently, [103] present an application to genomic data where both the number of latent factors and the degree of sparsity is also selected by the user. In the relevant on-line learning literature, very little work has been done for adapting the degree of sparsity in recursive regression problems, especially in high-dimensional settings. To our knowledge, the only method of adapting this parameter over time has been proposed by [6], but they deal with a simpler on-line regression problem in which the response is univariate and the explanatory variables are uncorrelated. In their work, the sparsity parameter is selected in order to minimises a running estimate of the AIC criterion which quantify the goodness of fit of the regression model. Although a similar solution may be investigated for our setting, this seems much harder to achieve in practice because our PLS model assumes R_t latent factors, which may change over time; the algorithm would then need to track the optimal sparsity parameter $\gamma_t^{(r)}$ for each factor. More work needed to be done towards the development a fully automated procedure.

We have also presented an application of iS-PLS to the bivariate index tracking problem in computational finance. On this task, iS-PLS overperforms both target indices by the a pre-selected amount. It also improves upon two competing on-line variable selection methods based on penalised regression, namely recursive LARS [48] and adaptive Lasso [6], which consistently underperform the target indices. It should be noted that this is not a completely realistic simulation of the index tracking problem. Associated with re-balancing a portfolio are *transaction costs* which make re-estimating the sparse regression coefficients, $\boldsymbol{\beta}_t$ at each time point prohibitively costly. In order to develop a realistic solution to the index tracking problem, the tracking error must be minimised subject to a constraint on the transaction costs.

Chapter 7

Conclusions and further work

In this work we have introduced three main novel methods for both supervised and unsupervised learning in high-dimensions based on identifying low-dimensional projections of the data. Throughout this work we have included discussion of the successes and limitations of these methods in the relevant chapters. In this chapter we summarise the main contributions and avenues for further research.

In our first contribution, we developed a framework for detecting observations which are influential under a PCA model based on the out-of-sample reconstruction error. We then used this as a goodness-of-fit measure in the context of subspace clustering. The resulting subspace clustering method achieved state-of-the art clustering accuracy and speed on both simulated and real datasets. We then briefly proposed an extension to PSC based on the penalised regression framework to obtain sparse loadings.

The PSC approach is inherently limited since it only considers clusters which lie in linear subspaces. Whilst this works well for certain applications such as images of faces and motion tracking, it may be desirable to develop a more general framework for identifying clusters which do not necessarily lie in linear subspaces. Recently, the field of multiple manifold clustering has emerged where the aim is to find clusters which lie in non-linear manifolds of which, subspace clustering is a special case. A recent method for spectral clustering on multiple manifolds [99] has been shown to achieve excellent results on both simulated datasets and the

Hopkins155 dataset compared with linear subspace clustering methods.

In our second major contribution, we proposed a new approach to multi-view clustering which takes a step towards consolidating supervised and unsupervised learning in the multi-view setting. This allows us to model more complicated dependencies between the views than the usual conditional independence assumption allows. Our approach can be viewed as an extension of subspace clustering in two-views and so carries with it the same benefits of subspace clustering compared to geometric-distance based clustering.

The field of multi-view clustering where there is a predictive relationship between the views has not been well developed and so our work represents a significant and novel contribution which consolidates supervised and unsupervised approaches to multi-view learning. Since this method has many conceptual and algorithmic similarities with PSC, an obvious extension would be to develop a penalised MVPP algorithm which recovers sparse TB-PLS weights. However, the proposed MVPP optimisation algorithm does not perform well when $R > 1$. Therefore, a more robust approach to model selection must be developed.

Our final major contribution is a method for tracking important variables in streaming data. As we have seen, the field of variable selection in data streams has not been widely investigated and so our iS-PLS method represents a novel contribution to the field. We highlighted one possible application in financial index tracking for which iS-PLS out-performs both batch methods and other on-line variable selection methods. Earlier we noted that our index tracking application does not obey the usual constraints imposed by considering transaction fees. Therefore, an application-specific extension of iS-PLS would be to incorporate such constraints. This could be achieved by considering a different penalty on the weights such as the *fused-lasso* [89] which, alongside the standard Lasso penalty, imposes a constraint on the ℓ_1 difference between each pair of coefficients, $\sum_{p=2}^{P} |\beta_p - \beta_{p-1}|$. This encourages sparsity in the coefficients as well as sparsity in the difference between coefficients so that non-zero elements of $\boldsymbol{\beta}$ are piecewise constant. An on-line modification would rather impose a penalty on $\sum_{p=1}^{P} |\beta_{p,t} - \beta_{p,t-1}|$ so as to encourage a piecewise constant regression coefficient throughout time.

Since the framework for the efficient PRESS for both PCA and TB-PLS is based on recursive least squares, a possible avenue of further research is extend the work in Chapters 3, 4 and 5 to the on-line setting. In Chapter 6 we used a variable forgetting factor algorithm, detailed in Eq. (6.15), which monitors a function of the ratio between the residual error and the leverage at each time point. It can be recognised that both of these terms are commonly used in a variety of methods for identifying influential observations including the predictive influence measures we propose. Therefore, an interesting direction for future research would be to develop an on-line predictive influence measure for change-point detection for streaming data. A natural extension would then be to develop a fully on-line subspace and multi-view clustering algorithm for identifying clusters in high-dimensional data streams. Such methods could have applications in clustering objects in video streams. A practical benefit of an on-line clustering algorithm is a reduced storage complexity which allows clustering to be performed efficiently on data where there is a very large number of observations, such as large collections of images and web-pages on the internet without having to store them all in memory simultaneously which may be impracticable if not impossible.

Appendix A

Derivations and proofs for Chapter 3

A.1 Derivation of Definition 3.2

Using the chain rule, the gradient of the PRESS with respect to $\boldsymbol{x}_i$ has the following form

$$\frac{\partial J}{\partial \boldsymbol{x}_i} = \frac{1}{2}\frac{\partial}{\partial \boldsymbol{x}_i}\|\boldsymbol{e}_{-i}\|^2 = \frac{1}{2}\boldsymbol{e}_{-i}\frac{\partial}{\partial \boldsymbol{x}_i}\boldsymbol{e}_{-i}.$$

Using the quotient rule, the partial derivative of the i^{th} leave-one-out error has the following form

$$\frac{\partial \boldsymbol{e}_{-i}}{\partial \boldsymbol{x}_i} = \frac{\frac{\partial \boldsymbol{e}_i}{\partial \boldsymbol{x}_i}(1-h_i) + \boldsymbol{e}_i\frac{\partial h_i}{\partial \boldsymbol{x}_i}}{(1-h_i)^2}$$

which depends on the partial derivatives of the i^{th} reconstruction error and h_i which have the following forms respectively

$$\begin{aligned}\frac{\partial \boldsymbol{e}_i}{\partial \boldsymbol{x}_i} &= \frac{\partial}{\partial \boldsymbol{x}_i}\boldsymbol{x}_i\left(\boldsymbol{I}_P - \boldsymbol{v}\boldsymbol{v}^\top\right)\\ &= \left(\boldsymbol{I}_P - \boldsymbol{v}\boldsymbol{v}^\top\right),\end{aligned}$$

and

$$\begin{aligned}\frac{\partial h_i}{\partial \boldsymbol{x}_i} &= \frac{\partial}{\partial \boldsymbol{x}_i}\boldsymbol{x}_i \boldsymbol{v} D \boldsymbol{v}^\top \boldsymbol{x}_i^\top \\ &= 2\boldsymbol{v} D d_i.\end{aligned}$$

Finally, we can evaluate the derivative of the PRESS, J with respect to $\boldsymbol{x}_i$ as

$$\begin{aligned}\frac{1}{2}\frac{\partial \|\boldsymbol{e}_{-i}\|^2}{\partial \boldsymbol{x}_i} &= \boldsymbol{e}_{-i}\frac{\partial \boldsymbol{e}_{-i}}{\partial \boldsymbol{x}_i} \\ &= 2\boldsymbol{e}_{-i}\frac{\left(\boldsymbol{I}_P - \boldsymbol{v}\boldsymbol{v}^\top\right)(1-h_i) + 2\boldsymbol{e}_i \boldsymbol{v} D d_i}{(1-h_i)^2}\end{aligned}$$

However, examining the second term in the sum, $\boldsymbol{e}_i \boldsymbol{v} D d_i$, we notice

$$\begin{aligned}\boldsymbol{e}_i \boldsymbol{v} D d_i &= (\boldsymbol{x}_i - \boldsymbol{x}_i \boldsymbol{v}\boldsymbol{v}^\top)\boldsymbol{v} D d_i \\ &= \boldsymbol{x}_i \boldsymbol{v} D d_i - \boldsymbol{x}_i \boldsymbol{v}\boldsymbol{v}^\top \boldsymbol{v} D d_i \\ &= 0.\end{aligned}$$

The gradient for a single PCA component with respect to $\boldsymbol{x}_i$ is given by

$$\begin{aligned}\frac{1}{2}\frac{\partial \|\boldsymbol{e}_{-i}\|^2}{\partial \boldsymbol{x}_i} &= 2\boldsymbol{e}_{-i}\frac{\left(\boldsymbol{I}_P - \boldsymbol{v}\boldsymbol{v}^\top\right)(1-h_i)}{(1-h_i)^2} \\ &= 2\boldsymbol{e}_{-i}\frac{\left(\boldsymbol{I}_P - \boldsymbol{v}\boldsymbol{v}^\top\right)}{(1-h_i)}.\end{aligned}$$

Since the contributions of successive components are additive, for $R > 1$, we arrive at the expression in Definition 3.2.

A.2 Proof of Lemma 3.1

From Definition 3.2, for $R = 1$, the predictive influence of a point $\boldsymbol{\pi}_{\boldsymbol{x}_i}(\boldsymbol{v})$ is

$$\begin{aligned}\boldsymbol{\pi}_{\boldsymbol{x}_i}(\boldsymbol{v}) &= \boldsymbol{e}_{-i}\frac{(\boldsymbol{I}_p - \boldsymbol{v}\boldsymbol{v}^\top)}{1-h_i} \\ &= \frac{\boldsymbol{e}_i}{(1-h_i)^2}\end{aligned} \tag{A.1}$$

This is simply the i^{th} leave-one-out error scaled by $1 - h_i$. If we define a diagonal matrix $\boldsymbol{\Xi} \in \mathbb{R}^{N\times N}$ with diagonal entries $\Xi_i = (1-h_i)^2$, we can define a matrix $\boldsymbol{\Pi} \in \mathbb{R}^{N\times P}$ whose rows are the predictive influences, $\boldsymbol{\pi}_{\boldsymbol{x}_i}(\boldsymbol{v})$, $\boldsymbol{\Pi} = [\boldsymbol{\pi}_{\boldsymbol{x}_1}(\boldsymbol{v})^\top, \ldots, \boldsymbol{\pi}_{\boldsymbol{x}_N}(\boldsymbol{v})^\top]^\top$. This matrix has the form

$$\boldsymbol{\Pi} = \boldsymbol{\Xi}^{-1}\left(\boldsymbol{X} - \boldsymbol{X}\boldsymbol{v}\boldsymbol{v}^\top\right).$$

Now, solving (3.8) is equivalent to minimising the squared Frobenius norm, $\|\boldsymbol{\Pi}\|_F^2 = \text{Tr}\left(\boldsymbol{\Pi}^\top\boldsymbol{\Pi}\right)$,

$$\begin{aligned}&\min_{\boldsymbol{v}} \text{Tr}\left(\left(\boldsymbol{X} - \boldsymbol{X}\boldsymbol{v}\boldsymbol{v}^\top\right)^\top \boldsymbol{\Xi}^{-2}\left(\boldsymbol{X} - \boldsymbol{X}\boldsymbol{v}\boldsymbol{v}^\top\right)\right) \\ &\text{subject to } \|\boldsymbol{v}\| = 1.\end{aligned} \tag{A.2}$$

Expanding the terms within the trace we obtain

$$\begin{aligned}&\text{Tr}\left(\left(\boldsymbol{X} - \boldsymbol{X}\boldsymbol{v}\boldsymbol{v}^\top\right)^\top \boldsymbol{\Xi}^{-2}\left(\boldsymbol{X} - \boldsymbol{X}\boldsymbol{v}\boldsymbol{v}^\top\right)\right) \\ &= \text{Tr}\left(\boldsymbol{X}^\top\boldsymbol{\Xi}^{-2}\boldsymbol{X}\right) - 2\text{Tr}\left(\boldsymbol{v}\boldsymbol{v}^\top\boldsymbol{X}^\top\boldsymbol{\Xi}^{-2}\boldsymbol{X}\right) + \text{Tr}\left(\boldsymbol{v}\boldsymbol{v}^\top\boldsymbol{X}^\top\boldsymbol{\Xi}^{-2}\boldsymbol{X}\boldsymbol{v}\boldsymbol{v}^\top\right).\end{aligned}$$

By the properties of the trace, the following equalities hold

$$\text{Tr}\left(\boldsymbol{v}\boldsymbol{v}^\top\boldsymbol{X}^\top\boldsymbol{\Xi}^{-2}\boldsymbol{X}\right) = \boldsymbol{v}^\top\boldsymbol{X}^\top\boldsymbol{\Xi}^{-2}\boldsymbol{X}\boldsymbol{v},$$

and

$$\begin{aligned}\text{Tr}\left(\boldsymbol{v}\boldsymbol{v}^\top \boldsymbol{X}^\top \boldsymbol{\Xi}^{-2} \boldsymbol{X}\boldsymbol{v}\boldsymbol{v}^\top\right) &= \text{Tr}\left(\boldsymbol{\Xi}^{-1}\boldsymbol{X}\boldsymbol{v}\boldsymbol{v}^\top \boldsymbol{v}\boldsymbol{v}^\top \boldsymbol{X}^\top \boldsymbol{\Xi}^{-1}\right) \\ &= \text{Tr}\left(\boldsymbol{\Xi}^{-1}\boldsymbol{X}\boldsymbol{v}\boldsymbol{v}^\top \boldsymbol{X}^\top \boldsymbol{\Xi}^{-1}\right) \\ &= \boldsymbol{v}^\top \boldsymbol{X}^\top \boldsymbol{\Xi}^{-2} \boldsymbol{X}\boldsymbol{v},\end{aligned}$$

since $\boldsymbol{\Xi}$ is diagonal and $\boldsymbol{v}^\top \boldsymbol{v} = 1$. Therefore, (A.2) is equivalent to

$$\begin{aligned}&\min_{\boldsymbol{v}} \text{Tr}\boldsymbol{X}^\top \boldsymbol{\Xi}^{-2} \boldsymbol{X} - \boldsymbol{v}^\top \boldsymbol{X}^\top \boldsymbol{\Xi}^{-2} \boldsymbol{X}\boldsymbol{v}, \\ &\text{subject to } \|\boldsymbol{v}\| = 1.\end{aligned} \tag{A.3}$$

It can be seen that under this constraint, Eq. (A.3) is minimised when $\boldsymbol{v}^\top \boldsymbol{X}^\top \boldsymbol{\Xi}^{-2} \boldsymbol{X}\boldsymbol{v}$ is maximised which, for a fixed $\boldsymbol{\Xi}$ is achieved when $\boldsymbol{v}$ is the eigenvector corresponding to the largest eigenvalue of $\boldsymbol{X}^\top \boldsymbol{\Xi}^{-2} \boldsymbol{X}$.

Appendix B

Proof of Lemma 4.1

In this section we provide a proof of Lemma 4.1 which states that estimating the PCA parameters using the newly assigned clusters always introduces an error compared to the optimal parameters, however this error is always smaller than if the previously estimated PCA parameters are used. As an additional consequence of this proof, we develop an upper bound for the approximation error which can be shown to depend on the leverage terms. We derive this result for a single cluster, $\mathcal{C}^{new}$ however it holds for all clusters.

We represent the assignment of points $i = 1, \ldots, N$ to a cluster, $\mathcal{C}^{new}$ using a binary valued diagonal matrix $\boldsymbol{A}$ whose diagonal entries are given by

$$A_i = \begin{cases} 1, & \text{if } i \in \mathcal{C}^{new} \\ 0, & \text{otherwise,} \end{cases} \tag{B.1}$$

where $\text{Tr}(\boldsymbol{A}) = N_k$. We have shown in Lemma 3.1 that for a given cluster assignment, the parameters which optimise the objective function can be estimated by computing the SVD of the matrix

$$\sum_{i \in \mathcal{C}_k^{new}} \boldsymbol{x}_i^\top \Xi_i^{-2} \boldsymbol{x}_i = \boldsymbol{X}^\top \boldsymbol{\Xi}^{-2} \boldsymbol{A} \boldsymbol{X}, \tag{B.2}$$

within each cluster where the i^{th} diagonal element of $\boldsymbol{\Xi}$ is $\Xi_i = (1 - h_i)^2 \leq 1$, so that $\Xi_i^{-2} \geq 1$. We can then represent $\boldsymbol{\Xi}^{-2} = \boldsymbol{I}_N + \boldsymbol{\Phi}$ where $\boldsymbol{\Phi} \in \mathbb{R}^{n \times n}$ is a diagonal

matrix with entries $\Phi_i = \phi_i \geq 0$. Now, we can represent Eq. (B.2) at the next iteration as

$$\boldsymbol{M} = \boldsymbol{X}^\top \boldsymbol{A}(\boldsymbol{I}_N + \boldsymbol{\Phi})\boldsymbol{X}. \tag{B.3}$$

We can quantify the difference between the optimal parameter, $\boldsymbol{v}^*$ obtained by solving (3.9) using $\boldsymbol{M}$ and the new PCA parameter estimated at iteration $\tau + 1$, $\boldsymbol{v}^{new}$ as,

$$E(\mathcal{S}^*, \mathcal{S}^{new}) = \boldsymbol{v}^{*\top} \boldsymbol{M}^{new} \boldsymbol{v}^* - \boldsymbol{v}^{new\top} \boldsymbol{X}^\top \boldsymbol{A}\boldsymbol{X}\boldsymbol{v}^{new},$$

where $\boldsymbol{v}^{new}$ is obtained through the SVD of $\boldsymbol{X}^\top \boldsymbol{A}\boldsymbol{X}$. We can express $E(\mathcal{S}^*, \mathcal{S}^{new})$ in terms of the spectral norm of $\boldsymbol{M}$. Since the spectral norm of a matrix is equivalent to its largest singular value, we have $\boldsymbol{v}^{new\top} \boldsymbol{X}^\top \boldsymbol{A}\boldsymbol{X}\boldsymbol{v}^{new} = \|\boldsymbol{X}^\top \boldsymbol{A}\boldsymbol{X}\|$ Since $\boldsymbol{\Phi}$ is a diagonal matrix, its spectral norm, $\|\boldsymbol{\Phi}\| = \max(\boldsymbol{\Phi})$. Similarly, $\boldsymbol{A}$ is a diagonal matrix with binary valued entries, so $\|\boldsymbol{A}\| = 1$.

$$\begin{aligned} E(\mathcal{S}^*, \mathcal{S}^{new}) &\leq \|\boldsymbol{M} - \boldsymbol{X}^\top \boldsymbol{A}\boldsymbol{X}\| \\ &\leq \|\boldsymbol{X}^\top \boldsymbol{A}\boldsymbol{\Phi}\boldsymbol{X}\| \\ &\leq \max(\boldsymbol{\Phi})\|\boldsymbol{X}^\top \boldsymbol{X}\|. \end{aligned} \tag{B.4}$$

Where the triangle and Cauchy-Schwarz inequalities have been used. In a similar way, we now quantify the difference between the optimal parameter and the old PCA parameter $\boldsymbol{v}^{old}$.

$$E(\mathcal{S}^*, \mathcal{S}^{old}) = \boldsymbol{v}^{*\top} \boldsymbol{M}\boldsymbol{v}^* - \boldsymbol{v}^{old\top} \boldsymbol{X}^\top \boldsymbol{A}\boldsymbol{X}\boldsymbol{v}^{old}.$$

Since $\boldsymbol{v}^{new}$ is the principal eigenvector of $\boldsymbol{X}^\top \boldsymbol{A}\boldsymbol{X}$, by definition, $\boldsymbol{v}^{new\top} \boldsymbol{X}^\top \boldsymbol{A}\boldsymbol{X}\boldsymbol{v}^{new}$ is maximised, therefore we can represent the difference between the new parameters and the old parameters as

$$E(\mathcal{S}^{new}, \mathcal{S}^{old}) = \boldsymbol{v}^{new\top} \boldsymbol{X}^\top \boldsymbol{A}\boldsymbol{X}\boldsymbol{v}^{new} - \boldsymbol{v}^{old\top} \boldsymbol{X}^\top \boldsymbol{A}\boldsymbol{X}\boldsymbol{v}^{old} \geq 0.$$

Using this quantity, we can express $E(\boldsymbol{v}^{old})$ in terms of the spectral norm of $\boldsymbol{S}$ as

$$\begin{aligned} E(\mathcal{S}^*, \mathcal{S}^{old}) \leq & \|\boldsymbol{M}\| - \boldsymbol{v}^{old^\top} \boldsymbol{X}^\top \boldsymbol{A} \boldsymbol{X} \boldsymbol{v}^{old} \\ \leq & \|\boldsymbol{X}^\top \boldsymbol{\Phi} \boldsymbol{A} \boldsymbol{X}\| + \|\boldsymbol{X}^\top \boldsymbol{A} \boldsymbol{X}\| - \boldsymbol{v}^{old^\top} \boldsymbol{X}^\top \boldsymbol{A} \boldsymbol{X} \boldsymbol{v}^{old} \\ \leq & \max(\boldsymbol{\Phi}) \|\boldsymbol{X}^\top \boldsymbol{X}\| + E(\mathcal{S}^{new}, \mathcal{S}^{old}), \end{aligned} \tag{B.5}$$

From Eq. (B.5) and (B.4) it is clear that

$$E(\mathcal{S}^*, \mathcal{S}^{new}) \leq E(\mathcal{S}^*, \mathcal{S}^{old}). \tag{B.6}$$

This proves Lemma 4.1.

The inequality in Eq. (B.6) implies that estimating the SVD using $\boldsymbol{X}^\top \boldsymbol{A} \boldsymbol{X}$ obtains PCA parameters which are closer to the optimal values than those obtained at the previous iteration. Therefore, estimating a new PCA model after each cluster re-assignment step never increases the objective function. Furthermore, as the recovered clustering becomes more accurate, by definition there are fewer influential observations within each cluster. This implies that $\max(\boldsymbol{\Phi}) \to 0$, and so $E(\mathcal{S}^*, \mathcal{S}^{new}) \to 0$.

Appendix C

Derivations and proofs for Chapter 5

C.1 Derivation of Definition 5.1

For a single latent factor we can write the i^{th} leave one out error, $\boldsymbol{e}_{-i}$ as

$$\boldsymbol{e}_{-i} = \boldsymbol{y}_i - \boldsymbol{x}_i\boldsymbol{\beta}_{-i},$$

where $\boldsymbol{\beta}_{-i}$ is estimated using all but the i^{th} observation. Since $\boldsymbol{\beta} = \boldsymbol{u}g\boldsymbol{q}^\top$, we can write $\boldsymbol{e}_{-i}$ as

$$\boldsymbol{e}_{-i} = \boldsymbol{y}_i - \boldsymbol{x}_i\boldsymbol{u}_{-i}g_{-i}\boldsymbol{q}_{-i}^\top.$$

The difference between the singular vectors estimated using all the data and the leave-one-out estimate, $\|\boldsymbol{u} - \boldsymbol{u}_{-i}\|$ is of order $O\left(\sqrt{\frac{\log(n-1)}{n-1}}\right)$ [60] so that if n is large, we can write

$$\boldsymbol{e}_{-i} = \boldsymbol{y}_i - \boldsymbol{x}_i\boldsymbol{u}g_{-i}\boldsymbol{q}_{-i}^\top$$

Using the matrix inversion lemma, we can obtain recursive update equations for g_{-i} which only depends on g and does not require an explicit leave-one-out step in the

following way

$$\begin{aligned} g_{-i} &= \left(\boldsymbol{t}^\top \boldsymbol{t}\right)_{-i}^{-1} \boldsymbol{t}_{-i}^\top \boldsymbol{s}_{-i} \\ &= \left(\boldsymbol{t}^\top \boldsymbol{t} - t_i t_i\right)^{-1} \left(\boldsymbol{t}^\top \boldsymbol{s} - t_i s_i\right) \\ &= g - \frac{(s_i - t_i g)(\boldsymbol{t}^\top \boldsymbol{t})^{-1} t_i}{1 - t_i^2}, \end{aligned}$$

where the expression for g is given by Equation (5.8). In the same way we derive an expression for $\boldsymbol{q}_{-i}$ which only depends on $\boldsymbol{q}$

$$\boldsymbol{q}_{-i} = \boldsymbol{q} - \frac{\left(\boldsymbol{y}_i - s_i \boldsymbol{q}^\top\right)^\top (\boldsymbol{s}^\top \boldsymbol{s})^{-1} s_i}{1 - s_i^2},$$

where the expression for $\boldsymbol{q}$ is given by Equation (5.7). Equation (5.11) is then obtained by using these values for g_{-i} and $\boldsymbol{q}_{-i}$ in $\boldsymbol{\beta}_{-i} = \boldsymbol{u} g_{-i} \boldsymbol{q}_{-i}^\top$ and simplifying.

C.2 Derivation of Definition 5.4

In this section we provide a derivation of the predictive with respect to an observation, $\boldsymbol{x}_i$. We take the partial derivative of the TB-PLS PRESS function, J with respect to $\boldsymbol{x}_i$,

$$\frac{\partial J}{\partial \boldsymbol{x}_i} = \frac{1}{N} \frac{\partial \|\boldsymbol{e}_{-i}\|^2}{\partial \boldsymbol{x}_i} = \frac{2\boldsymbol{e}_{-i}}{N} \frac{\partial \boldsymbol{e}_{-i}}{\partial \boldsymbol{x}_i}.$$

Taking derivatives of the constituent parts of $\boldsymbol{e}_{-i}$ in Equation (5.11) with respect to $\boldsymbol{x}_i$ we obtain

$$\frac{\partial}{\partial \boldsymbol{x}_i} \boldsymbol{e}_i = -2\boldsymbol{\beta}^\top, \tag{C.1}$$

$$\frac{\partial}{\partial \boldsymbol{x}_i} t_i^2 \boldsymbol{E}_{y,i} = 2\boldsymbol{E}_{y,i}^\top t_i \boldsymbol{u}^\top, \tag{C.2}$$

$$\frac{\partial}{\partial \boldsymbol{x}_i}\boldsymbol{b}_i = -2\boldsymbol{\beta}^\top, \tag{C.3}$$

and

$$\frac{\partial}{\partial \boldsymbol{x}_i}(1-t_i^2)(1-s_i^2) = -2(1-s_i^2)t_i\boldsymbol{u}^\top. \tag{C.4}$$

We can now obtain $\frac{\partial \boldsymbol{e}_{-i}}{\partial \boldsymbol{x}_i}$ by combining Equations (C.1), (C.2), (C.3) and (C.4)

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{x}_i}\boldsymbol{e}_{-i} =& 2\frac{\left(-\boldsymbol{\beta}^\top - \boldsymbol{E}_{y,i}^\top t_i\boldsymbol{u}^\top + \boldsymbol{\beta}^\top\right)(1-t_i^2)(1-s_i^2)}{\left((1-t_i^2)(1-s_i^2)\right)^2} \\ &+ 2\frac{\left(\boldsymbol{e}_i - t_i^2\boldsymbol{E}_{y,i} - \boldsymbol{b}_i\right)^\top (1-s_i^2)t_i\boldsymbol{u}^\top}{\left((1-t_i^2)(1-s_i^2)\right)^2}.\end{aligned}$$

From the definition of the PRESS in Eq. (5.11) we know that $\frac{\boldsymbol{e}_i - t_i^2\boldsymbol{E}_{y,i} - \boldsymbol{b}_i}{(1-t_i^2)(1-s_i^2)} = \boldsymbol{e}_{-i}$, so

$$\frac{\partial}{\partial \boldsymbol{x}_i}\boldsymbol{e}_{-i} = 2\frac{\left(-\boldsymbol{E}_{y,i}^\top t_i\boldsymbol{u}^\top\right)}{(1-t_i^2)(1-s_i^2)} + 2\frac{\boldsymbol{e}_{-i}^\top t_i\boldsymbol{u}^\top}{(1-t_i^2)}.$$

Finally, the predictive influence with respect to $\boldsymbol{x}_i$ is given by

$$\boldsymbol{\pi}_{\boldsymbol{x}_i}(\boldsymbol{u},\boldsymbol{v}) = \frac{\partial J}{\partial \boldsymbol{x}_i} = \frac{4\boldsymbol{e}_{-i}}{N}\left(\frac{-\boldsymbol{E}_{y,i}^\top}{(1-t_i^2)(1-s_i^2)} + \frac{\boldsymbol{e}_{-i}^\top}{(1-t_i^2)}\right)t_i\boldsymbol{u}^\top. \tag{C.5}$$

C.3 Proof of Lemma 5.1

In this section we provide a proof of Lemma 5.1 which states that minimising the sum of squared predictive influences as in Eq. (5.17),

$$\min_{\boldsymbol{u},\boldsymbol{v}} \sum_{i=1}^{N} \|\boldsymbol{\pi}_{\boldsymbol{x}_i}(\boldsymbol{u},\boldsymbol{v})\|^2, \tag{C.6}$$

is equivalent to solving the following maximisation problem,

$$\max_{\boldsymbol{u},\boldsymbol{v}} \sum_{i=1}^{N} \Xi_i^{-1}\boldsymbol{u}^\top\boldsymbol{x}_i^\top\boldsymbol{y}_i\boldsymbol{v}.$$

We show this by initially rewriting the objective function in Eq. (5.17) in matrix form for a single cluster k using the form of the predictive influence given in Eq. (C.5) in Appendix C.2. We then optimise the objective function with respect to the parameters, $\boldsymbol{u}$ and $\boldsymbol{v}$.

In order to write the predictive influence in matrix form, we first construct a matrix, $\boldsymbol{\mathcal{E}} \in \mathbb{R}^{N\times Q}$ whose rows are the leave-one-out errors, $\boldsymbol{e}_{-i} = \frac{\boldsymbol{e}_i - t_i^2 \boldsymbol{E}_{y,i} - \boldsymbol{b}_i}{(1-t_i^2)(1-s_i^2)}$, from Eq. (5.11), so that $\boldsymbol{\mathcal{E}} = [\boldsymbol{e}_{-1}^\top, \ldots, \boldsymbol{e}_{-N}^\top]^\top$. Defining diagonal matrices $\boldsymbol{\Xi} \in \mathbb{R}^{N\times N}$ with diagonal entries $\Xi_i = (1-t_i^2)(1-s_i^2)$, and $\boldsymbol{H} \in \mathbb{R}^{N\times N}$ with elements $H_i = \Xi_i^{-1} t_i^2$, we can write $\boldsymbol{\mathcal{E}}$ as

$$\begin{aligned}
\boldsymbol{\mathcal{E}} =& \boldsymbol{\Xi}^{-1}\left(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta} - (\boldsymbol{s} - g\boldsymbol{t})\,\boldsymbol{s}^\top \boldsymbol{Y}\right) - \boldsymbol{H}\boldsymbol{E}_y \\
=& \boldsymbol{\Xi}^{-1}\left(\boldsymbol{Y} - g\boldsymbol{X}\boldsymbol{u}\boldsymbol{v}^\top - \boldsymbol{Y}\boldsymbol{v}\boldsymbol{v}^\top + g\boldsymbol{X}\boldsymbol{u}\boldsymbol{v}^\top\right) - \boldsymbol{H}\left(\boldsymbol{Y} - \boldsymbol{Y}\boldsymbol{v}\boldsymbol{v}^\top\right) \\
=& \left(\boldsymbol{\Xi}^{-1} - \boldsymbol{H}\right)\boldsymbol{Y} - \left(\boldsymbol{\Xi}^{-1} - \boldsymbol{H}\right)\boldsymbol{Y}\boldsymbol{v}\boldsymbol{v}^\top \\
=& \left(\boldsymbol{\Xi}^{-1} - \boldsymbol{H}\right)\boldsymbol{Y}\left(\boldsymbol{I}_q - \boldsymbol{v}\boldsymbol{v}^\top\right). \qquad \text{(C.7)}
\end{aligned}$$

We also define a scaled version of $\boldsymbol{\mathcal{E}}$ in the same way as Eq. (C.7), which we denote $\tilde{\boldsymbol{\mathcal{E}}} \in \mathbb{R}^{N\times Q}$ whose rows are $\frac{\boldsymbol{e}_{-i}}{1-t_i^2}$. This matrix has the form

$$\tilde{\boldsymbol{\mathcal{E}}} = \left(\boldsymbol{\Xi}^{-1}\tilde{\boldsymbol{\Xi}}^{-1} - \tilde{\boldsymbol{H}}\right)\boldsymbol{Y}\left(\boldsymbol{I}_Q - \boldsymbol{v}\boldsymbol{v}^\top\right), \qquad \text{(C.8)}$$

where $\tilde{\boldsymbol{\Xi}} \in \mathbb{R}^{N\times N}$ is a diagonal matrix with diagonal entries $\tilde{\Xi}_i = (1-t_i^2)$ and $\tilde{\boldsymbol{H}} \in \mathbb{R}^{N\times N}$ is a diagonal matrix with entries $\tilde{H}_i = \Xi_i^{-1}\tilde{\Xi}_i^{-1} t_i^2$. Using these quantities, we can now define a matrix $\boldsymbol{\Pi} = [\boldsymbol{\pi}_{\boldsymbol{x}_1}(\boldsymbol{u},\boldsymbol{v})^\top, \ldots, \boldsymbol{\pi}_{\boldsymbol{x}_N}(\boldsymbol{u},\boldsymbol{v})^\top]^\top \in \mathbb{R}^{N\times P}$ whose rows are the predictive influence terms given in Eq. (C.5),

$$\boldsymbol{\Pi} = \frac{4}{N}\boldsymbol{\mathcal{E}}\underbrace{\left(-\boldsymbol{E}_y^\top \boldsymbol{\Xi}^{-1} + \tilde{\boldsymbol{\mathcal{E}}}^\top\right)}_{\boldsymbol{B}}\boldsymbol{X}\boldsymbol{u}\boldsymbol{u}^\top,$$

where for notational convenience we also define the matrix $\boldsymbol{B} \in \mathbb{R}^{Q\times N}$. Therefore,

solving (C.6) is equivalent to minimising the squared Frobenius norm,

$$\min_{\boldsymbol{u},\boldsymbol{v}} \text{Tr}\left(\boldsymbol{\Pi}^\top \boldsymbol{\Pi}\right) = \min_{\boldsymbol{u},\boldsymbol{v}} \boldsymbol{u}^\top \boldsymbol{X}^\top \boldsymbol{B}^\top \boldsymbol{\mathcal{E}}^\top \boldsymbol{\mathcal{E}} \boldsymbol{B} \boldsymbol{X} \boldsymbol{u}. \tag{C.9}$$

The equality in Eq. (C.9) holds due to the following property of the trace,

$$\text{Tr}\left(\boldsymbol{\Pi}^\top \boldsymbol{\Pi}\right) = \text{Tr}\left(\boldsymbol{\Pi}\, \boldsymbol{\Pi}^\top\right),$$

and using $\boldsymbol{u}^\top \boldsymbol{u} = 1$. First, in order to solve Eq. (C.9), since it is explicitly a function of $\boldsymbol{u}$, we evaluate its partial derivative with respect to $\boldsymbol{u}$ and set the resulting expression equal to zero,

$$\frac{\partial}{\partial \boldsymbol{u}}\, \boldsymbol{u}^\top \boldsymbol{X}^\top \boldsymbol{B}^\top \boldsymbol{\mathcal{E}}^\top \boldsymbol{\mathcal{E}} \boldsymbol{B} \boldsymbol{X} \boldsymbol{u} = 2\boldsymbol{u}^\top \boldsymbol{X}^\top \boldsymbol{B}^\top \boldsymbol{\mathcal{E}}^\top \boldsymbol{\mathcal{E}} \boldsymbol{B} \boldsymbol{X} = \boldsymbol{0},$$

for which a solution occurs when the vector

$$\boldsymbol{u}^\top \boldsymbol{X}^\top \boldsymbol{B}^\top = \boldsymbol{0} \in \mathbb{R}^{1\times q}. \tag{C.10}$$

Now, using Eq. (C.8) in Eq. (C.10) we have

$$\boldsymbol{u}^\top \boldsymbol{X}^\top \left(\boldsymbol{\Xi}^{-1}\tilde{\boldsymbol{\Xi}}^{-1} - \tilde{\boldsymbol{H}}\right) \boldsymbol{Y} \left(\boldsymbol{I}_Q - \boldsymbol{v}\boldsymbol{v}^\top\right) = \boldsymbol{u}^\top \boldsymbol{X}^\top \boldsymbol{\Xi}^{-1} \boldsymbol{Y} \left(\boldsymbol{I}_Q - \boldsymbol{v}\boldsymbol{v}^\top\right), \tag{C.11}$$

so a "trivial" solution occurs when both sides of Eq. (C.11) are equal to zero, that is

$$\boldsymbol{u}^\top \boldsymbol{X}^\top \boldsymbol{\Xi}^{-1} \boldsymbol{Y} \left(\boldsymbol{I}_Q - \boldsymbol{v}\boldsymbol{v}^\top\right) = \boldsymbol{0}.$$

If we define the scalar $\lambda_\Xi = \boldsymbol{u}^\top \boldsymbol{X}^\top \boldsymbol{\Xi}^{-1} \boldsymbol{Y} \boldsymbol{v}$ the resulting expression is

$$\boldsymbol{u}^\top \boldsymbol{X}^\top \boldsymbol{\Xi}^{-1} \boldsymbol{Y} = \lambda_\Xi \boldsymbol{v}^\top. \tag{C.12}$$

Therefore, for fixed $\boldsymbol{\Xi}$ a solution for the optimal $\boldsymbol{u}$ and the corresponding value of $\boldsymbol{v}$ can be obtained using the SVD where $\boldsymbol{u}$ is the left singular vector of the matrix $\boldsymbol{X}^\top \boldsymbol{\Xi}^{-1} \boldsymbol{Y}$ with corresponding singular value λ_Ξ and right singular vector, $\boldsymbol{v}$. These

singular vectors can be obtained by solving the following maximisation problem,

$$\max_{\boldsymbol{u},\boldsymbol{v}} \sum_{i=1}^{N} \Xi_i^{-1} \boldsymbol{u}^\top \boldsymbol{x}_i^\top \boldsymbol{y}_i \boldsymbol{v},$$

which proves the lemma.

C.4 Proof of Lemma 5.2

In this section we provide a proof of Lemma 5.2. This proof follows the same arguments as the proof in Appendix B. We derive this result for a single cluster, $\mathcal{C}^{new}$ however it holds for all clusters. We represent the assignment of points $i = 1, \ldots, N$ to a cluster, $\mathcal{C}^{new}$ using a binary valued diagonal matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ whose diagonal entries are given by

$$A_i = \begin{cases} 1, & \text{if } i \in \mathcal{C}^{new} \\ 0, & \text{otherwise,} \end{cases} \tag{C.13}$$

where $\text{Tr}(\boldsymbol{A}) = N_k$. We have shown in Eq. (C.12) that for a given cluster assignment, the parameters which optimise the objective function can be estimated by computing the SVD of the matrix

$$\sum_{i \in \mathcal{C}^{new}} \boldsymbol{x}_i^\top \Xi_i^{-1} \boldsymbol{y}_i = \boldsymbol{X}^\top \boldsymbol{\Xi}^{-1} \boldsymbol{A} \boldsymbol{Y}, \tag{C.14}$$

within each cluster where the i^{th} diagonal element of $\boldsymbol{\Xi}$ is $\Xi_i = (1-t_i^2)(1-s_i^2) \leq 1$, so that $\Xi_i^{-1} \geq 1$. We can then represent $\boldsymbol{\Xi}^{-1} = \boldsymbol{I}_n + \boldsymbol{\Phi}$ where $\boldsymbol{\Phi} \in \mathbb{R}^{N \times N}$ is a diagonal matrix with entries $\Phi_i \geq 0$. Now, we can represent Eq. (C.14) as

$$\boldsymbol{M} = \boldsymbol{X}^\top \boldsymbol{A} (\boldsymbol{I}_n + \boldsymbol{\Phi}) \boldsymbol{Y}. \tag{C.15}$$

We can quantify the difference between the optimal parameters, $\Theta^* = \{\boldsymbol{u}^*, \boldsymbol{v}^*\}$ obtained by solving (5.18) using $\boldsymbol{M}$ and the new TB-PLS parameters, $\Theta^{new} =$

$\{\boldsymbol{u}^{new}, \boldsymbol{v}^{new}\}$ as,

$$E(\Theta^*, \Theta^{new}) = \boldsymbol{u}^{*\top}\boldsymbol{M}\boldsymbol{v}^* - \boldsymbol{u}^{new\top}\boldsymbol{X}^\top\boldsymbol{A}\boldsymbol{Y}\boldsymbol{v}^{new},$$

Taking spectral norms we have $\boldsymbol{u}^{*\top}\boldsymbol{M}\boldsymbol{v}^* = \|\boldsymbol{M}\|$ and $\boldsymbol{u}^{new\top}\boldsymbol{X}^\top\boldsymbol{A}\boldsymbol{Y}\boldsymbol{v}^{new} = \|\boldsymbol{X}^\top\boldsymbol{A}\boldsymbol{Y}\|$. If we define $\max(\boldsymbol{\Phi}) = \max_i(\sqrt{\Phi_i})$, we have $\|\boldsymbol{\Phi}\| = \max(\boldsymbol{\Phi})$. Similarly, $\|\boldsymbol{A}\| = 1$. We can then express $E(\Theta^*, \Theta^{new})$ in terms of the spectral norm,

$$\begin{aligned} E(\Theta^*, \Theta^{new}) &= \|\boldsymbol{M}\| - \|\boldsymbol{X}^\top\boldsymbol{A}\boldsymbol{Y}\| \\ &\leq \|\boldsymbol{X}^\top\boldsymbol{A}(\boldsymbol{I}_n + \boldsymbol{\Phi})\boldsymbol{Y} - \boldsymbol{X}^\top\boldsymbol{A}\boldsymbol{Y}\| \\ &\leq \max(\boldsymbol{\Phi})\|\boldsymbol{X}^\top\boldsymbol{Y}\|. \end{aligned} \tag{C.16}$$

In a similar way, we now quantify the difference between the optimal parameters and the old TB-PLS parameters $\Theta^{old} = \{\boldsymbol{u}^{old}, \boldsymbol{v}^{old}\}$,

$$E(\Theta^*, \Theta^{old}) = \boldsymbol{u}^{*\top}\boldsymbol{M}\boldsymbol{v}^* - \boldsymbol{u}^{old\top}\boldsymbol{X}^\top\boldsymbol{A}\boldsymbol{Y}\boldsymbol{v}^{old}.$$

Since $\boldsymbol{u}^{new}$ and $\boldsymbol{v}^{new}$ are the first left and right singular vectors of $\boldsymbol{X}^\top\boldsymbol{A}\boldsymbol{Y}$, by definition, $\boldsymbol{u}^{new}\boldsymbol{X}^\top\boldsymbol{A}\boldsymbol{Y}\boldsymbol{v}^{new}$ is maximised, therefore we can represent the difference between the new parameters and the old parameters as

$$E(\Theta^{new}, \Theta^{old}) = \boldsymbol{u}^{new\top}\boldsymbol{X}^\top\boldsymbol{A}\boldsymbol{Y}\boldsymbol{v}^{new} - \boldsymbol{u}^{old\top}\boldsymbol{X}^\top\boldsymbol{A}\boldsymbol{Y}\boldsymbol{v}^{old} \geq 0. \tag{C.17}$$

Using this quantity we can express $E(\Theta^*, \Theta^{old})$ in terms of the spectral norm of $\boldsymbol{M}$ as

$$\begin{aligned} E(\Theta^*, \Theta^{old}) =& \|\boldsymbol{M}\| - \boldsymbol{u}^{old\top}\boldsymbol{X}^\top\boldsymbol{A}\boldsymbol{Y}\boldsymbol{v}^{old} \\ \leq& \|\boldsymbol{X}^\top\boldsymbol{\Phi}\boldsymbol{A}\boldsymbol{Y}\| + \|\boldsymbol{X}^\top\boldsymbol{A}\boldsymbol{Y}\| - \boldsymbol{u}^{old\top}\boldsymbol{X}^\top\boldsymbol{A}\boldsymbol{Y}\boldsymbol{v}^{old} \\ \leq& \max(\boldsymbol{\Phi})\|\boldsymbol{X}^\top\boldsymbol{Y}\| + E(\Theta^{new}, \Theta^{old}). \end{aligned} \tag{C.18}$$

From Eq. (C.18) and (C.16) it is clear that

$$0 \leq E(\Theta^*, \Theta^{new}) \leq E(\Theta^*, \Theta^{old}). \tag{C.19}$$

This proves Lemma 5.2.

Appendix D

Linear Algebra

Some of the main results from linear algebra which we use in this thesis are presented in here. These results are discussed in more detail in [36].

D.1 Singular Value Decomposition

The singular value decomposition of $\boldsymbol{X}$ is

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}^\top, \tag{D.1}$$

where $\boldsymbol{U} = [\boldsymbol{u}^{(1)}, ..., \boldsymbol{u}^{(N)}] \in \mathbb{R}^{N\times N}$ and $\boldsymbol{V} = [\boldsymbol{v}^{(1)}, ..., \boldsymbol{v}^{(P)}] \in \mathbb{R}^{P\times P}$ are orthonormal matrices whose columns are the left and right singular vectors of $\boldsymbol{X}$, respectively. $\boldsymbol{\Lambda} = \text{diag}(\lambda^{(1)}, \ldots, \lambda^{(N)}) \in \mathbb{R}^{N\times P}$ is a diagonal matrix whose entries are the singular values of $\boldsymbol{X}$.

The SVD is also related to the eigenvalue decomposition (EVD) in the following way

$$\begin{aligned} S = \boldsymbol{X}^\top\boldsymbol{X} &= (\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}^\top)\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}^\top \\ &= \boldsymbol{V}\boldsymbol{\Lambda}^2\boldsymbol{V}^\top. \end{aligned}$$

So that the the columns of $\boldsymbol{V}$ are also the eigenvectors of $\boldsymbol{S}$ and the entries of $\boldsymbol{\Lambda}$ are the corresponding square root eigenvalues.

The first R components of the SVD constitute the best rank R approximation of $\boldsymbol{X}$ under the Frobenius norm.

D.2 Woodbury Matrix Identity

For matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ of the appropriate sizes we can write the inverse of $\boldsymbol{A} + \boldsymbol{BC}$ in terms of the inverse of $\boldsymbol{A}$ using the following identity,

$$(\boldsymbol{A} + \boldsymbol{BC})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{B}(\boldsymbol{I} + \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B})\boldsymbol{C}\boldsymbol{A}^{-1}.$$

We can use this identity to efficiently update the inverse of a positive semi-definite matrix, $\boldsymbol{A} \in \mathbb{R}^{P\times P}$ after a rank-1 update by a vector $\boldsymbol{x} \in \mathbb{R}^{1\times P}$,

$$(\boldsymbol{A} + \boldsymbol{x}^\top \boldsymbol{x})^{-1} = \boldsymbol{A}^{-1} - \frac{\boldsymbol{A}^{-1}\boldsymbol{x}^\top \boldsymbol{x}\boldsymbol{A}^{-1}}{1 + \boldsymbol{x}\boldsymbol{A}^{-1}\boldsymbol{x}^\top}.$$

Similarly, if we perform a rank-1 downdate of $\boldsymbol{A}$ we have,

$$(\boldsymbol{A} - \boldsymbol{x}^\top \boldsymbol{x})^{-1} = \boldsymbol{A}^{-1} + \frac{\boldsymbol{A}^{-1}\boldsymbol{x}^\top \boldsymbol{x}\boldsymbol{A}^{-1}}{1 - \boldsymbol{x}\boldsymbol{A}^{-1}\boldsymbol{x}^\top}.$$

References

[1] C. Aggarwal, J. Wolf, P. Yu, C. Procopiuc, and J. Park. Fast algorithms for projected clustering. *Sigmod Record*, 28:61–72, 1999.

[2] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. *Sigmoid Record*, 27:94–105, 1998.

[3] C. Alexander and A. Dimitriu. The cointegration alpha: enhanced index tracking and long-short equity market neutral strategies. *Social Science Research Network Working Paper Series*, 2002.

[4] C. Alexander and A. Dimitriu. Equity indexing: optimising passive investments. *Quantitative Finance*, 4:30–33, 2004.

[5] C. Alexander and A. Dimitriu. Sources of over-performance in equity markets: mean reversion, common trends and herding. Technical report, ISMA Center, University of Reading, UK, 2005.

[6] C. Anagnostopoulos, D. Tasoulis, D. J. Hand, and N. M. Adams. Online optimisation for variable selection on data streams. *Proceedings of the 18th European Conf. on Artificial Intelligence*, pages 132–136, 2008.

[7] E. Bair, T. Hastie, D. Paul, and R. Tibshirani. Prediction by supervised principal components. *Journal of the American Statistical Association*, 101:119–137, 2006.

[8] J. Beasley, N. Meade, and T. J. Chang. An evolutionary heuristic for the index tracking problem. *European Journal of Operational Research*, 148:621–643, 2003.

[9] D. Belsley and E. Kuh. *Regression diagnostics: Identifying influential data and sources of collinearity*. Wiley, New York, USA, 1 edition, 2004.

[10] S. Bickel and T. Scheffer. Multi-view clustering. In *IEEE International Conference on Data Mining*, pages 19–26. Citeseer, 2004.

[11] S. Bickel and T. Scheffer. Estimation of mixture models using co-em. In *The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 35–46, 2005.

[12] C. Bishop. *Pattern recognition and machine learning*. Springer, New York, 2006.

[13] P. Bradley and O. Mangasarian. k-Plane clustering. *Journal of Global Optimization*, 16:23–32, 2000.

[14] M. Brand. Fast online svd revisions for lightweight recommender systems. Technical report, Mitsubishi Electric Research Laboratory, 2003.

[15] L. Breiman and J. Friedman. Predicting Multivariate Responses in Multiple Linear Regression. *Journal of the Royal Statistical Society Series B*, 59:3–54, 1997.

[16] E. Bruno. Multiview Clustering: A Late Fusion Approach Using Latent Models. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 736–737, 2009.

[17] S. Chatterjee and A. Hadi. Influential Observations, High Leverage Points, and Outliers in Linear Regression. *Statistical Science*, 1:379–393, 1986.

[18] K. Chaudhuri, S. Kakade, K. Livescu, and K. Sridharan. Multi-view clustering via canonical correlation analysis. *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 129–136, 2009.

[19] G. Chen and G. Lerman. Spectral Curvature Clustering (SCC). *International Journal of Computer Vision*, 81:317–330, 2008.

[20] C.-H. Cheng, A. W. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 84–93, 1999.

[21] R. D. Cook. Assessment of local influence. *Journal of the Royal Statistical Society Series B*, 48:133–169, 1986.

[22] B. S. Dayal and J. F. Macgregor. Improved PLS algorithms. *Journal of Chemometrics*, 11:73–85, 1997.

[23] B. S. Dayal and J. F. Macgregor. Recursive exponentially weighted pls and its applications to adaptive control and prediction. *Journal of Process Control*, 7:169–179, 1997.

[24] V. de Sa, P. Gallagher, J. Lewis, and V. Malave. Multi-view kernel construction. *Machine learning*, 79:47–71, 2010.

[25] C. Ding and X. He. K-means clustering via principal component analysis. *Proceedings of the 21st international conference on Machine learning*, 2004.

[26] N. Draper and J. John. Influential observations and outliers in regression. *Technometrics*, 23:21–26, 1981.

[27] R. Duda and P. Hart. *Pattern classification*. Wiley-Interscience, 2 edition, 2000.

[28] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

[29] E. Elhamifar and R. Vidal. Sparse subspace clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2797, 2009.

[30] S. Erlich and K. Yao. Convergences of adaptive block simultaneous iteration method for eigenstructure decomposition. *Signal Processing*, 37:1–13, 1994.

[31] J. Friedman, E. Hastie, H. Höfling, and R. Tibshirani. Pathwise Coordinate Optimization. *The Annals of Applied Statistics*, 1:302–332, 2007.

[32] J. H. Friedman and J. J. Meulman. Clustering objects on subsets of attributes. *Journal of the Royal Statistical Society Series B*, 66:815–849, 2004.

[33] A. Georghiades, P. Belhumeur, and D. Kriegman. From few to many: illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:643–660, 2001.

[34] L. Gidskehaug, H. Stødkilde-Jørgensen, M. Martens, and H. Martens. Bridge-pls regression: two-block bilinear regression without deflation. *Journal of Chemometrics*, 18:208–215, 2004.

[35] M. Gilli and E. Këllezi. Threshold accepting for index tracking. Technical report, Department of Econometrics, University of Geneva, Switzerland, 2001.

[36] G. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1996.

[37] D. Greene and P. Cunningham. A Matrix Factorization Approach for Integrating Multiple Data Views. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part I*, pages 423–438, 2009.

[38] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[39] R. A. Haugen and N. Baker. Dedicated stock portfolios. *Journal of Portfolio Management*, 16:17–22, 1990.

[40] S. Haykin. *Adaptive Filter Theory*. Prentice Hall, 2001.

[41] C. Hou, C. Zhang, Y. Wu, and F. Nie. Multiple view semi-supervised dimensionality reduction. *Pattern Recognition*, 43:720–730, 2010.

[42] K. Huang, Y. Ma, and R. Vidal. Minimum effective dimension for mixtures of subspaces: a robust GPCA algorithm and its applications. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 631–638, 2004.

[43] M. Hubert, P. Rousseeuw, and K. Vanden Branden. ROBPCA: a new approach to robust principal component analysis. *Technometrics*, 47:64–79, 2005.

[44] A. Izenman. *Modern multivariate statistical techniques: regression, classification, and manifold learning*, chapter 6, pages 107–190. Springer Verlag, 2008.

[45] I. T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer-Verlag, New York, 2 edition, 2002.

[46] K. Kanatani. Geometric information criterion for model selection. *International Journal of Computer Vision*, 26:171–189, 1998.

[47] R. Kannan and S. Vempala. Spectral Algorithms. *Foundations and Trends in Theoretical Computer Science*, 4:157–288, 2008.

[48] S.-P. Kim, Y. Rao, D. Erdogmus, and J. Principe. Tracking of multivariate time-variant systems based on on-line variable selection. In *IEEE Workshop on Machine Learning for Signal Processing*, pages 123–132, 2004.

[49] J. Knight and S. Satchell, editors. *Linear Factor Models in Finance*. Butterworth-Heinemann, 2004.

[50] T. Lange and J. Buhmann. Fusion of similarity data in clustering. *Advances in Neural Information Processing Systems 18*, pages 723–730, 2006.

[51] K. Lê Cao, P. Martin, C. Robert-Granié, and P. Besse. Sparse canonical methods for biological data integration: application to a cross-platform study. *BMC bioinformatics*, 10, 2009.

[52] K. Lê Cao, D. Rossouw, C. Robert-Granié, and P. Besse. Sparse PLS: variable selection when integrating omic data. Technical report, INRA, 2008.

[53] S.-H. Leung and C. So. Gradient-based variable forgetting factor RLS algorithm in time-varying environments. *IEEE Transactions on Signal Processing*, 53:3141–3150, 2005.

[54] T. Li and C. Ding. Weighted consensus clustering. In *Proceedings of 2008 SIAM International Conference on Data Mining (SDM2008)*, 2008.

[55] L. Ljung. *System Identification: Theory for the User*. Prentice Hall, 2nd edition, 1998.

[56] B. Long, P. Yu, and Z. Zhang. A general model for multiple view unsupervised learning. In *Proceedings of the 8th SIAM International Conference on Data Mining (SDM'08), Atlanta, Georgia, USA*, 2008.

[57] U. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17:395–416, Aug. 2007.

[58] Y. Ma. *Generalized Principal Component Analysis: Modeling & Segmentation of Multivariate Mixed Data*. 2006.

[59] H. Martens. Modified Jack-knife estimation of parameter uncertainty in bilinear modelling by partial least squares regression (PLSR). *Food Quality and Preference*, 11:5–16, 2000.

[60] B. McWilliams and G. Montana. A PRESS statistic for two-block partial least squares regression. In *Proceedings of the 10th Annual Workshop on Computational Intelligence*, 2010.

[61] B. McWilliams and G. Montana. Sparse partial least squares regression for on-line variable selection with multivariate data streams. *Statistical Analysis and Data Mining*, 3:170–193, 2010.

[62] B. McWilliams and G. Montana. Predictive subspace clustering. In *Machine Learning and Applications (ICMLA), 2011 Tenth International Conference on*, pages 247–252, December 2011.

[63] N. Meinshausen and P. Bühlmann. Stability selection. *Journal of the Royal Statistical Society Series B*, 72:417–473, 2010.

[64] M. Meloun. Detection of single influential points in OLS regression model building. *Analytica Chimica Acta*, 439:169–191, 2001.

[65] B. Mertens, T. Fearn, and M. Thompson. The efficient cross-validation of principal components applied to principal component regression. *Statistics and Computing*, 5:227–235, 1995.

[66] J. Miao and C. L. Dunis. Volatility filters for dynamic portfolio optimization. *Applied Financial Economics Letters*, 1:111–119, 2005.

[67] O. Nasraoui, C. Rojas, and C. Cardona. A framework for mining evolving trends in web data streams using dynamic learning and retrospective validation. *Journal of Computer Networks - Special Issue on Web Dynamics*, 50:1425–1652, 2006.

[68] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856, 2001.

[69] S.-K. Ng, G. McLachlan, and A. Lee. An incremental em-based learning approach for on-line prediction of hospital resource utilization. *Artificial Intelligence in Medicine*, 36:257–267, 2006.

[70] C. Paleologu, J. Benesty, and S. Ciochina. A robust variable forgetting factor recursive least-squares algorithm for system identification. *IEEE Signal Processing Letters*, 15:597–600, 2008.

[71] S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming pattern discovery in multiple time-series. *Proceedings of the 31st International Conference on Very Large Data Bases*, pages 697 – 708, 2005.

[72] L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: a review. *SIGKDD Explor. Newsl*, 6:90–105, 2004.

[73] D. Pregibon. Logistic regression diagnostics. *The Annals of Statistics*, 9:705–724, 1981.

[74] L. Qin and S. Self. On comparing the clustering of regression models method with K-means clustering. *Memorial Sloan-Kettering Cancer Center Department of Epidemiology and Biostatistics Working Paper Series*, page 14, 2007.

[75] A. Rahmatullah Imon. Identifying multiple influential observations in linear regression. *Journal of Applied Statistics*, 32:929–946, 2005.

[76] M. M. S. Rancel and M. A. G. Sierra. A connection between local and deletion influence. *Sankhyā: The Indian Journal of Statistics, Series A*, 62:144–149, 2000.

[77] R. Rosipal and N. Krämer. Overview and Recent Advances in Partial Least Squares. In *Subspace, Latent Structure and Feature Selection*, pages 34–51. Springer, 2006.

[78] A. Rudd. Optimal selection of passive portfolios. *Financial Management*, 1:57–66, 1980.

[79] M. Rudelson and R. Vershynin. Sampling from large matrices. *Journal of the ACM*, 54, 2007.

[80] H. Shen and J. Huang. Sparse principal component analysis via regularized low rank matrix approximation. *Journal of Multivariate Analysis*, 99:1015–1034, 2008.

[81] C. D. Sigg and J. M. Buhmann. Expectation-maximization for sparse and non-negative PCA. In *Proceedings of the 25th international conference on Machine learning - ICML '08*, pages 960–967, New York, New York, USA, 2008. ACM Press.

[82] J. Sinkkonen, J. Nikkilä, L. Lahti, and S. Kaski. Associative clustering. In *The European Conference on Machine Learning*, pages 396–406, 2004.

[83] H. Späth. Clusterwise linear regression. *Computing*, 22:367–373, 1979.

[84] G. Stewart and J.-g. Sun. *Matrix Perturbation Theory*. Academic Press Inc., San Diego, USA, first edition, 1990.

[85] M. Stone. Cross-validation and multinomial prediction. *Biometrika*, 61(3):509–515, 1974.

[86] M. Stone. An asymptotic equivalence of choice of model by cross-validation and Akaike's criterion. *Journal of the Royal Statistical Society Series B*, 39:44–47, 1977.

[87] Y. Tian, T. Huang, and W. Gao. Robust Collective Classification with Contextual Dependency Network Models. In *Advanced Data Mining and Applications*, pages 173–180, 2006.

[88] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B*, 58(1):267–288, 1996.

[89] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society Series B*, 67:91–108, 2005.

[90] R. Tron and R. Vidal. A Benchmark for the Comparison of 3-D Motion Segmentation Algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[91] P. Tseng. Nearest q-flat to m points. *Journal of Optimization Theory and Applications*, 105:249–252, 2000.

[92] G. Tzortzis and A. Likas. Convex mixture models for multi-view clustering. In *International Conference on Artificial Neural Networks*, pages 205–214, 2009.

[93] S. Vempala and G. Wang. A spectral algorithm for learning mixtures of distributions. *Journal of Computer and System Sciences*, 68:841–860, 2002.

[94] R. Vidal. Subspace Clustering. *IEEE Signal Processing Magazine*, 28:52–68, 2011.

[95] S. Vijayakumar, A. D'Souza, and S. Schaal. Incremental Online Learning in High Dimensions. *Neural Computation*, 17:2602–2634, 2005.

[96] S. Waaijenborg and A. Zwinderman. Penalized canonical correlation analysis to quantify the association between gene expression and dna markers. *BMC Proceedings*, 1, 2007.

[97] D. Wang, C. Ding, and T. Li. K-Subspace Clustering. In *Machine Learning and Knowledge Discovery in Databases*, pages 506–521. Springer, 2009.

[98] H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept -drifting data streams using ensemble classifiers. *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 226–235, 2003.

[99] Y. Wang, Y. Jiang, Y. Wu, and Z.-H. Zhou. Spectral clustering on multiple manifolds. *IEEE Transactions on Neural Networks*, 22:1149 –1161, 2011.

[100] J. Wegelin. A Survey of Partial Least Squares (PLS) Methods, with Emphasis on the Two-Block Case. Technical report, University of Washington, 2000.

[101] J. Weng, Y. Zhang, and W. Hwang. Candid covariance-free incremental principal component analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:1034–1040, 2003.

[102] D. Witten. *A Penalized Matrix Decomposition, and its Applications*. PhD thesis, Stanford University, 2010.

[103] D. Witten and R. Tibshirani. A framework for feature selection in clustering. *Journal of the American Statistical Association*, 105:713–726, 2010.

[104] S. Wold, A. Ruhe, H. Wold, and W. J. Dunn. The Collinearity Problem in Linear Regression. The Partial Least Squares (PLS) Approach to Generalized Inverses. *SIAM journal on Scientific Computing*, 5:735–743, 1984.

[105] S. Wold, M. Sjöström, and L. Eriksson. PLS-regression: a basic tool of chemometrics. *Chemometrics and Intelligent Laboratory Systems*, 58:109–130, 2001.

[106] B. Yang. Projection Approximation Subspace Tracking. *IEEE Transactions on Signal Processing*, 43:95–107, 1995.

[107] B. Yang. Asymptotic convergence analysis of the projection approximation subspace tracking algorithms. *Signal processing*, 50:123–136, 1996.

[108] T. Zhang, A. Szlam, Y. Wang, and G. Lerman. Hybrid linear modeling via local best-fit flats. Arxiv preprint, 2010.

[109] Y. Zhu and D. Shasha. Statstream: statistical monitoring of thousands of data streams in real time. *Proceedings of the 28th VLDB Conference*, 2002.

[110] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. 2004.