Fall 12-17-2013

Clustering, Classification, and Factor Analysis in High Dimensional Data Analysis

Yanhong Wang

CLUSTERING, CLASSIFICATION, AND FACTOR ANALYSIS IN HIGH DIMENSIONAL DATA ANALYSIS

by

YANHONG WANG

Under the Direction of Xin Qi

ABSTRACT

Clustering, classification, and factor analysis are three popular data mining techniques. In this dissertation, we investigate these methods in high dimensional data analysis. Since there are much more features than the sample sizes and most of the features are non-informative in high dimensional data, dimension reduction is necessary before clustering or classification can be made. In the first part of this dissertation, we reinvestigate an existing clustering procedure, optimal discriminant clustering (ODC; Zhang and Dai, 2009), and propose to use cross-validation to select the tuning parameter. Then we develop a variation of ODC, sparse optimal discriminant clustering (SODC) for high dimensional data, by adding a group-lasso type of penalty to ODC. We also demonstrate that both ODC and SDOC can be used as a dimension reduction tool for data visualization in cluster analysis. In the second part, three existing sparse principal component analysis (SPCA) methods, Lasso-PCA (L-PCA), Alternative Lasso PCA (AL-PCA), and sparse principal component analysis by choice of norm (SPCABP) are applied to a real data set the International HapMap Project for AIM selection to genome-wide SNP data, the classification accura-

cy is compared for them and it is demonstrated that SPCABP outperforms the other two SPCA methods. Third, we propose a novel method called sparse factor analysis by projection (SFABP) based on SPCABP, and propose to use cross-validation method for the selection of the tuning parameter and the number of factors. Our simulation studies show that SFABP has better performance than the unpenalyzed factor analysis when they are applied to classification problems.

INDEX WORDS: Cluster analysis, Classification, Cross-validation, High-dimensional data, Optimal score, Principal components analysis, Tuning parameter, Variable selection, Factor Analysis

CLUSTERING, CLASSIFICATION, AND FACTOR ANALYSIS IN HIGH DIMENSIONAL DATA ANALYSIS

by

YANHONG WANG

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2013

CLUSTERING, CLASSIFICATION, AND FACTOR ANALYSIS IN HIGH DIMENSIONAL DATA ANALYSIS

by

YANHONG WANG

Committee Chair: Xin Qi

Committee: Yixin Fang

Ruiyan Luo

Yi Jiang

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

December 2013

DEDICATION

To my parents, advisers, and husband.

ACKNOWLEDGEMENTS

I would like to thank all people who have assisted and inspired me to successfully complete my doctoral study and contributed to completion of this dissertation.

I am very fortunate that I could finish my dissertation with the guidance of my two advisors, Dr. Xin Qi and Dr. Yixin Fang.

I am deeply thankful to Dr. Xin Qi, my current Ph.D. advisor, for his exceptional knowledge and clear directions. He always showed me how to approach the problems and helped me to solve them. I am especially grateful to Dr. Qi Xin for his continuous support for my research work, so that I could finish this dissertation smoothly through his patient and enlightening guidance.

I am deeply indebted to Dr. Yixin Fang, my previous advisor, for his guidance in my research during my first one and half years of Ph.D. study. Due to his creative ideas for the dissertation related topic and broad knowledge related to the research, I could establish ground work for my research and finish an important part of my thesis due to his help and advice. He also helped me to write in scientific language.

I also would like to express my gratitude to my dissertation committee members, Dr. Ruiyan Luo, Dr. Yi Jiang. They spent much precious time on correcting mistakes, and provided many insightful suggestions to improve the dissertation.

Additionally, I wish to show my great appreciation to all the professors who have taught and leaded me to the statistics field, Dr. Gengsheng Qin, Dr. Jiawei Liu, Dr. Jun Han, Dr. Xu Zhang, Dr. Yichuan Zhao, Dr. Yuanhui Xiao, and Dr. Satish Nargundkar. I would like to give my special thanks to Dr. Gengsheng Qin, as the graduate advisor, he always willing to provide advice on how to succeed in study.

Finally, I would like to thank my parents, my parents in law, my sisters, and my family. They have always supported and encouraged me whenever I needed. And a special thanks to my husband, who always support me forever no matter what happens.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

- AIC - Akaike Information Criterion
- ARI - Adjusted rand index
- BIC - Bayes Information Criterion
- CE - Clustering error
- CFA - Confirmatory factor analysis
- EFA - Exploratory factor analysis
- FA - Factor analysis
- FDA - Flexible Discriminant Analysis
- GWAS - Genome-wide association studies
- KDD - Knowledge discovery in database
- KNN - K nearest neighbor
- LDA - Linear discriminant analysis
- MLE - Maximum likelihood expectation
- ODC - Optimal discriminant clustering
- OLS - Ordinary least squares
- PCs - Principal components
- PCA - Principal component analysis
- SFA - Sparse factor analysis
- SODC - Sparse optimal discriminant clustering
- SPCA - Sparse principal component analysis
- SPCABP - Sparse principal component analysis by projection
- SVM - Support vector machine

1 INTRODUCTION

1.1 Overview of Statistical Data Mining

1.1.1 Introduction

Nowadays, the advent of computers and the information age makes it necessary to deal with large volumes of data characterized by volume, variety and velocity. As Rutherford D. Roger pointed out: "We are drowning in information and starving for knowledge." T.S. Eliot also indicated that "Where is the wisdom we have lost in knowledge? Where is the knowledge we have lost in information?" Thus many new technologies including data mining emerge to deal with "big data".

Data mining, a term came into widespread use in the 1990s, is one of many newly-popular terms in the science and mathematical fields along with many other fields like marketing where marketers try to get useful consumer information from various sources. William J Frawley, Gregory Piatetsky-Shapiro and Christopher J Matheus defined data mining as "The non trivial extraction of implicit, previously unknown, and potentially useful information from data." As the analysis step of the "Knowledge Discovery in Databases" process, or KDD (Fayyad et al., 1996), data mining is the process of extracting useful information from large data sets and transforms it into an understandable structure for further use. Data mining is an interdisciplinary subfield of computer science and statistics (Clifton, 2010; Hastie et al., 2010). It involves database and data management aspects, data prerocessing, model and inference considerations, interestingness metrics, complexity considerations, post-processing of discovered structures, visualization, and online updating (Data Mining Curriculum,2006).

Statisticians have applied these data mining techniques to many fields like biology and medicine to extract important patterns and trends from large datasets, and understand "what the data says." We call this area statistical data mining (Hastie et al., 2009).

There are two distinct types of learning methods: unsupervised and supervised learning.

In unsupervised learning methods, the number of classes or clusters and the class labels of observations are unknown, it aims to determine the number of classes and assign observations to the classes.

On the other hand, in supervised learning methods, the number of classes and the class labels for the training data are unknown. We want to find proper classification rules based on which we can assign new observations with unknown class labels to one of the classes.

1.1.2 Methodology of Data Mining

There are many different data mining techniques. In this subsection, we briefly review four popular statistical data mining methods.

1.1.2.1 Clustering

Given a collection of data points, where each data point is a p-dimensional vector, the goal is to find clustering patterns among the observations by maximizing the similarity within clusters and minimize the similarity between clusters (Hastie et al., 2009).

The similarity or dissimilarity between two objects is often measured by some similarity metrics such as the distance matrix, where the most commonly used one is the Euclidean distance. The definition of Euclidean distance is:

If $x = \{x_1, x_{2,\dots,} x_n\}$ and $y = \{y_1, y_{2,\dots,} y_n\}$ are two points in Euclidean n-space, then the distance from x to y, or from y to x is given by:

$$d(x,y) = d(y,x) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Various clustering procedures have been proposed in the literature, such as k-means clustering (MacQueen, 1967), hierarchical clustering (Johnson, 1967), k-medoids clustering (Kaufman and Rousseeuw, 1987), and spectral clustering (Shi and Malik, 2000; Ng et al., 2002). Each clustering method has its strengths and weaknesses. They will be discussed separately.

1.1.2.1.1 K-means clustering

The k-means method is one of the simplest clustering method.

Assume the number k of the clusters has been predetermined. Then the k-means algorithm has the following steps:

1) Randomly selects k centroids.
2) Assigns each data point to its closest centroid.
3) Recalculates the centroids as the average of all data points in a cluster.
4) Reassigns data points to their closest centroids.
5) Continues steps 3 and 4 until the centroids no longer changed or the maximum number of iterations is reached.

This algorithm is actually aims to solve the following optimization problem:

$$min \sum_{i=1}^{n} \sum_{j=1}^{k} \left\| (x_i^{(j)} - c_j) \right\|^2,$$

where k is the number of clusters, $x_i^{(j)}$ is the i^{th} observation in cluster j, and c_j is the cluster center for cluster j. $\left\| (x_i^{(j)} - c_j) \right\|^2$ is the squared distance between $x_i^{(j)}$ and c_j, so the above problem is to minimize the sum of the squared within-cluster distances of the n data points from their respective cluster centers.

***K*-means has the advantages and disadvantage as follows:**

Advantage:

It can process large data sets relatively efficiently.

Disadvantage:

- It requires specifying the number of clusters to extract in advance.
- The algorithm is significantly sensitive to the initial randomly selected cluster centers.
- All variables must be continuous because of the use of means.
- The approach can be severely affected by noise or outliers.
- They perform poorly in the presence of non-convex (e.g., U-shaped) clusters.

Figure 1.1 is an illustration example of the last limitation on the two moons dataset: The dataset consist of 2 clusters and 600 observations. The original dataset is plotted in the left panel, and the results of the k-means clustering are plotted in the right panel. In this example, the k-means method fails to distinguish the two clusters correctly.

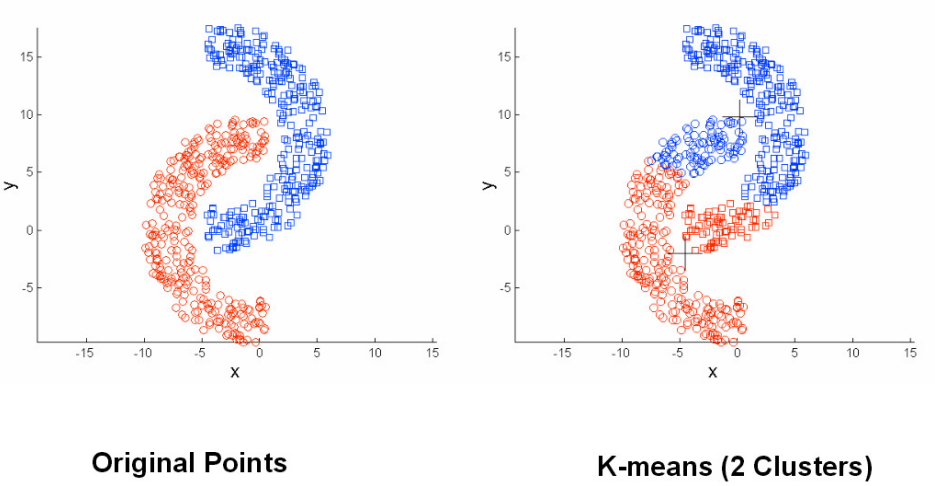

Figure 1.1 Illustration of the disadvantage of *K*-means on non-convex shapes dataset.

1.1.2.1.2 K-medoids clustering

K-medoid is another classical clustering. Both the k-means and k-medoids algorithms are partitional (breaking the dataset up into groups) and both attempt to minimize the sum of within cluster variations. In contrast to the k-means algorithm, k-medoids pick actual observations as centers to represent clusters instead of mean values and can work with an arbitrary matrix of distances in addition to the Euclidean distance.

K-medoid is more robust to noise and outliers than k-means because it minimizes the sum of pairwise dissimilarities instead of the sum of squared Euclidean distances.

The most commonly used *k*-medoid algorithm has the following steps:

1) Randomly select k of the n data points as the medoids.
2) Associate each data point to the closest medoid. ("Closest" here is defined using any valid distance metric, such as Euclidean distance, Manhattan distance or Minkowski distance).

3) For each medoid **m**:

- For each non-medoid data point **o**:
  - Swap **m** and **o** and compute the total cost of the configuration. Here cost is calculated using the same distance metric used in step 2). Use Manhattan distance to illustrate, $cost(x, c) = \sum_{i=1}^{d} |x_i - c_i|$, where *x* is any data object, *c* is the medoid, and *d* is the dimension of the object. Total cost is the summation of the cost of data object from its medoid in its cluster.

4) Select the configuration with the lowest cost.

5) Repeat steps 2 to 4 until there is no change in the medoid.

***K*- Medoids has the advantages and disadvantage as follows:**

Advantage:

- K-Medoids method is more robust than k-Means in the presence of noise and outliers.

Disadvantage:

- It does not scale well for large data sets.
- K-Medoids is more costly than the k-Means method.
- Like k-means, k-medoids requires the number of clusters k to be pre-specified.

1.1.2.1.3 Hierarchical clustering

Hierarchical clustering is a clustering method which seeks to build a hierarchy of clusters to show relations between the individual members and clusters of data based on similarity. The results of hierarchical clustering are usually presented in a dendrogram.

Strategies for hierarchical clustering generally fall into two types:

Agglomerative ("bottom up"): treat each data point as a singleton cluster from the outset and then successively merge (or agglomerate) pairs of clusters until all clusters have been merged into a single cluster that contains all data points.

Divisive ("top down"): all observations start in one cluster, and splits are performed recursively until individual data points are reached.

Hierarchical clustering has the advantages and disadvantage as follows:

Advantage:

- It only requires a measure of similarity between groups of data points.
- No need to pre-specify the number of clusters in advance.
- Generates smaller clusters which may be helpful for discovery.

Disadvantage:

- Lower efficiency.
- Interpretation of results is subjective.
- Objects may be “incorrectly” grouped at early stage. The result should be examined closely to ensure it makes sense.
- Use of different distance metrics between clusters may generate different results.

Figure 1.2 is an illustrative example of hierarchical clustering given by Mehak Aziz. Refer here.

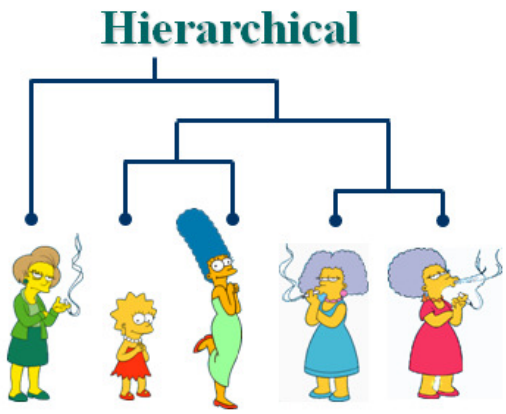

Figure 1.2 Hierarchical clustering illustration.

1.1.2.1.4 Spectral clustering

Spectral clustering (Ng et al., 2002) has become one of the most popular modern clustering algorithms in recent years. It cluster points using eigenvectors of matrices derived from the data and obtain data representation in the low-dimensional space that can be easily clustered. Spectral clustering often

outperforms traditional clustering algorithms such as the k-means algorithm. There are several versions of this method where different ways are used to extract eigenvectors. I only introduce one of them.

The algorithm of spectral clustering by Ng, Jordan, and Weiss is as follows:

1) Given a set of points $S = \{s_1, \dots, s_n\} \in R^p$.
2) Form the affinity matrix $A \in R^{n \times n}$ defined by $A_{ij} = exp\left(-\left\|s_i - s_j\right\|^2 / 2\sigma^2\right)$ if $i \neq j$, and $A_{ij} = 0$ otherwise. The matrix A consists of a quantitative assessment of the relative similarity of each pair of points in the dataset.
3) Define D to be the diagonal matrix whose (i, i)- element is the sum of A's i-th row, and $L = D^{-1/2} A D^{-1/2}$.
4) Find $x_1, x_2, \dots, x_k$, the k largest eigenvectors of L (chosen to be orthogonal to each other in the case of repeated eigenvalues), and form the matrix $X = [x_1, x_2, \dots, x_k] \in R^{n \times k}$ by stacking the eigenvectors in columns.
5) Form the matrix Y from X by renormalizing each of X's rows to have unit length, i.e. $y_{ij} = \frac{x_{ij}}{(\sum_j {x_{ij}}^2)^2}$
6) Treating each row of Y as a point in R^k, cluster them into k clusters via K-means or any other algorithm(that attempts to minimize distortion).
7) Finally, assign the original point s_i to cluster j if and only if row i of the matrix Y was assigned to cluster j.

Spectral clustering has the advantages and disadvantage as follows:

Advantage:

- Easy to implement.
- Make no assumption on the form of the data clusters by transforming the data clustering to graph partitioning problem.
- Can be solved efficiently by standard linear algebra.
- Good clustering results, specially, it is invariant to cluster shapes and densities.

- Reasonably faster for sparse data sets of several thousand elements.

Disadvantage:

- Sensitive to choice of parameters.
- Computationally expensive for large datasets.

Figure 1.3 illustrates the advantages of spectral clustering on non-convex shapes data points compared to k-means clustering. It uses the spirals dataset in the "kernlab" R package. The spirals dataset consist of 300 observations and 2 dimensions. From the figure, we can see that Spectral clustering can correctly distinguish the two clusters indicated by red color and black color respectively, while the *k*-means can't.

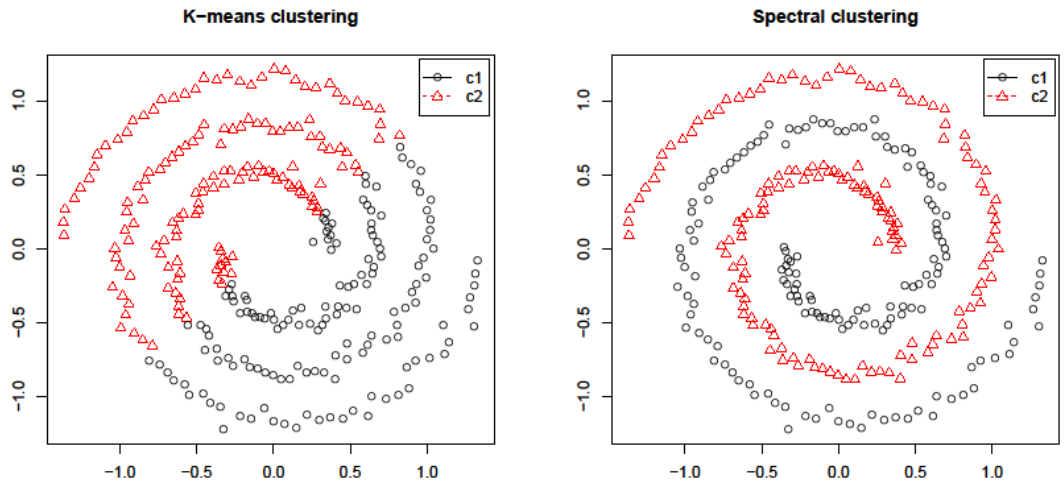

Figure 1.3 Advantages of spectral clustering in non-convex shapes data

1.1.2.2 Classification

Classification is a widely used supervised learning method. Given a training dataset containing observations whose category memberships are known, the goal is to construct a proper classification rule such when a new observation is obtained with unknown class label, we can assign it to one of the classes and make the classification error as small as possible. We often call the categories to be predicted as outcomes (dependent variables, classes), and the explanatory variables are termed predictors (independent variables) or features if they are grouped into a feature vector.

The most widely used classification algorithms includes linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), the neural network, support vector machines(SVM) (Cortes and Vapnik, 1995), k-nearest neighbours(KNN) (Altman, 1992), Gaussian mixture model, Gaussian, naive Bayes, decision tree and RBF classifiers. (Broomhead and Lowe, 1988(a); Broomhead and Lowe, 1988(b); Schwenker et al. 2001). I will only list KNN here which will be used later.

1.1.2.2.1 K-nearest neighbors (KNN)

The KNN algorithm is a widely used classification method. The algorithm is given as follows:

1) Given a point in the training data set, (say x_q), find k nearest (based on Euclidean distance or any other distance) points of x_q in the training set, say $\{x_1, x_{2,\ldots,} x_k\}$. Typically k is odd when the number of classes is 2 to avoid ties in the class assignment.
2) Return the majority vote of k observations as the class of x_q. Say $k = 5$ and there are 3 observations of C1 and 2 observations of C2. In this case, KNN says that new point has to labeled as C1 as it forms the majority. Ties which arise from finding the closest instances to x_q or from voting for the class of x_q are broken arbitrarily. If there are ties for the kth nearest vector, all candidates are included in the vote.

KNN has the advantages and disadvantage as follows:

Advantage:

- Simple to implement.
- Make no any parametric assumptions.
- Nonlinear decision surfaces.
- Quality of prediction automatically improves as the number of training data increases, especially when each class is characterized by multiple combinations of predictor values.

Disadvantage:

- Expensive: must store and search through the entire training set to classify a single test point.
- Must define a similarity measure to indicate “closeness” between objects.

- Sensitive to noisy data.
- Need to pre-specify k, different k might have different predicted class label.
- Prediction accuracy can quickly degrade when number of attributes grows.

Figure 1.4 is a visualization example of KNN. We assume that there are only two attributes, A1 and A2, the data points come from two classes with solid fill and circles with hashed fill respectively. Query q is being classified by its nearest neighbors, the solid fill class label. The distance can be measured by Euclidean distance, Hamming distance, Manhattan distance, etc.

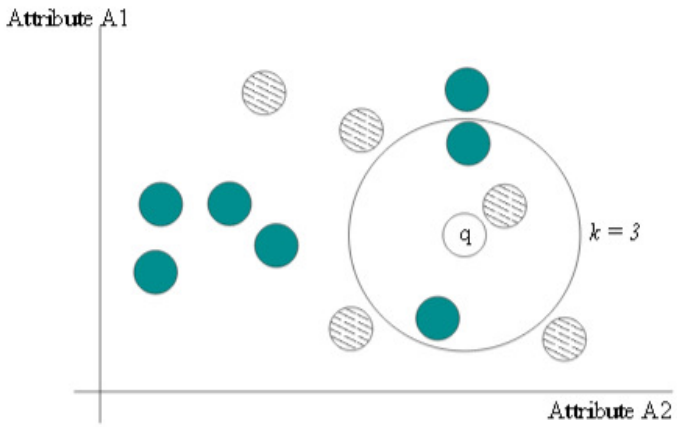

Figure 1.4 KNN example.

1.1.2.3 Principal Component Analysis (PCA)

As we have stated that the prediction accuracy of KNN will quickly degrade when number of features grows, such as in the high-dimensional situation. In this case, dimension reduction is necessary.

PCA is one of the most widely used dimension reduction tools. Its history can be traced back to Pearson (1901) or even Cauchy (p. 416, 1829), or Jordan (1874), or Stewart (1993), or Boyer and Merzbach(1989), but its modern instantiation was formalized by Hotelling (1933) who also coined the term principal component.

Consider a n-by-p data matrix, where n is the number of observations and p is the number of features. In the case of a large p, people want to find a set of new variables called principal components (PCs) which are linear combinations of the original p variables such that the new variables contains as much

variation patterns and information in the original data set as possible. The first PC is extracted such that it has the largest variation among all possible linear combinations the coefficient vectors of which have L_2 norm 1. The second PC has largest variation among all possible linear combinations which have L_2 norm 1 coefficients and uncorrelated to the first PC. Similarly, we can calculate the third, fourth,…, PCs. Typically, we are only interested in the first few components.

1.1.2.4 Factor Analysis (FA)

FA has a long history in psychology, dating back to the work of Spearman(1904) and Pearson (1901) in the early 1900's, now it is widely used in behavioral sciences, social sciences, marketing, product management, operations research, and other applied sciences that deal with large quantities of data.

FA is a statistical method used to describe variability among observed, correlated variables in terms of a potentially lower number of unobserved variables called factors. In other words, the variations in many observed variables mainly can be explained by the variations in a fewer unobserved variables. FA tries to model the observed variables as linear combinations of the potential factors, plus "error" terms. The interdependencies information among observed variables can be used later for dimension reduction.

FA is related to PCA, but the two are not identical. Latent variable models, including FA, use regression modeling techniques to test hypotheses producing error terms, while PCA is a descriptive statistical technique (Bartholomew et al., 2008). There has been significant controversy in the field over the equivalence or otherwise of the two techniques (Suhr, 2009).

There are two types of FA: exploratory factor analysis (EFA) and confirmatory factor analysis (CFA).

EFA is a statistical method used to uncover the underlying structure of a relatively large set of variables. The researcher makes no "a priori" assumptions about relationships among factors or patterns of measured variables (Fabrigar et al., 1999). EFA procedures are more accurate when each factor is represented by multiple measured variables in the analysis (Fabrigar et al., 1999). There should be at least 3 to 5 measured variables per factor (Maccallum, 1990).

CFA is a more complex approach that tests a priori hypothesis that observed variables are associated with specific factors, i.e., it test whether the data fit a hypothesized factor model. This hypothesis model is often based on theory and/or previous analytic research (Preedy and Watson, 2009). CFA was first developed by Jöreskog (Jöreskog, 1969). CFA is characterized by allowing restrictions on factor loadings, variances, covariances, and residual variances.

1.2 Overview of Dimension Reduction

1.2.1 Introduction

Many statistical learning methods, when they applied to high dimensional datasets with tens or hundreds of thousands of variables, are often subject to "Curse of Dimensionality". For example, in classification problem, suppose each instance is described by 10 attributes out of which only 2 are relevant in determining the classification of the objective function. In this case, instances that have identical values for the 2 relevant attributes are far from one another in the 10-dimensional instance space. The remedy of the "Curse of Dimensionality" often involves some form of dimensionality reduction, either explicitly or implicitly. Dimension reduction is the process of reducing the number of random variables under consideration and can be divided into feature selection and feature extraction (variable selection).

Feature selection aims to find linear or nonlinear combinations of the original set of variables, while feature extraction aims to select a subset of variables from the original set. Developing methods for dimensionality reduction requires being clear on the goal and the setting, as methods developed for one combination of goal and setting are not generally appropriate for another. Variable selection is particularly important when the true underlying model has a sparse representation. Identifying significant predictors will enhance the prediction performance of the fitted model. The objective of dimension reduction is: providing a simplified explanation of a phenomenon for a human (possibly as part of a visualization algorithm), suppressing noise so as to make a better prediction or decision, or reducing the computational burden. These various motivations are often complementary.

The PCA and FA are both dimension reduction techniques. Regularization is a popular way used in dimension reduction. Since many statistical problems can be reformulated to a linear regression problem, we will first discuss variable selection method in ordinary least square (OLS) linear regression, and then we will discuss the regularization method in linear regression context.

1.2.2 Variable Selection Methods in OLS

There are two main approaches towards variable selection in OLS: the automatic methods (forward, backward, stepwise) and all possible regressions approach (all subsets). Automatic methods are useful when the number of predictors is large and it is not feasible to fit all possible models. The all possible regressions approach considers all possible subsets of the pool of explanatory variables and finds the model that best fits the data according to some criteria such as Adjusted R^2, Akaike Information Criterion (AIC) and Bayes Information Criterion (BIC). These criteria assign scores to each model and allow us to choose the model with the best score. They are outlined below.

Forward selection method start with a model with no predictors, then for all predictors not in the model, check their p-value and add the variable with largest F-statistics providing its corresponding p-value is less than some cut-off level. Refit with this variable. Re-compute all F statistics for adding one of the remaining variables and add variable with largest F statistics, continue until no new variable can be added.

Backward selection method is the simplest of all variable selection procedure and can be easily implemented. It start with model with all predictors, then remove variable with smallest F-statistics providing its corresponding p-value is greater than some cut-off level. Refit with this variable deleted, then re-compute all F statistics for deleting one of the remaining variables and delete variable with smallest F statistics. Continue until every remaining variable is significant at cut-off level.

Stepwise selection method is a combination of backward elimination and forward selection, it is applicable in the case where start with model with no predictors. Add variable with largest F-statistics providing its corresponding p-value is less than some cut-off level, refit with this variable. Re-compute all

F statistics for adding one of the remaining variables and add variable with largest F statistics, at each step after adding a variable, try to eliminate any variable not significant at some level (that is, do backward elimination until that stops), after doing the backwards steps take another forward step. Continue until every remaining variable is significant at cut-off level and every excluded variable is insignificant or until variable to be added is same as last deleted variable. Variables are added or removed early in the process and we want to change our mind about them later. At each stage a variable may be added or removed and there are several variations on exactly how this is done. Stepwise are relatively cheap computationally and easy to explain but they do have some disadvantages: 1) It is possible to miss the "optimal model" when dropping and add variables one at a time; 2) The p-value is not so "trustful", it may overstate the significance of results, so there are multiple testing issues; 3) results of forward and backward selection may differ; 4) Stepwise variable selection tends to pick models that are smaller than desirable for prediction purpose.

All subset method: for each subset of the set of predictors, fit the model and compute some summary statistic of the quality of the fit. Pick model which makes this summary as large (or sometimes as small) as possible. With k predictors fit 2^k models, impractical for k too large. Special best subsets algorithms work without looking at all 2^k models.

Possible summary statistics:

- R^2: adding a variable increases R^2 so this is most useful for comparing models of the same size.
- Adjusted R^2: this method adjusts R^2 to try to compensate for the fact that more variables produces larger R^2 even when the extra variables are irrelvant.
- C_p: like adjusted R^2but based on the trade off of bias and variance.
- PRESS: the sum of squares of the PRESS residuals.
- AIC: the smaller the AIC, the better the model. Its formula is $AIC = -2logL + 2p$, where L is the likelihood and p is the number of free parameter in the model.

- BIC: the smaller the BIC, the better the model. Its formula is $BIC = -2logL + 2plogn$, where L is the likelihood, p is the number of free parameters in the model, and n is the number of observations. BIC penalizes larger models more heavily and so will tend to prefer smaller models in comparison to AIC.

1.2.3 Regularization

Suppose we are given a response vector $y \in R^n$ and a predictor matrix $x \in R^{n \times p}$ with n observations and p predictor variables. Without loss of generality, we assume that the response and predictor variables have sample means equal to zeros. In OLS, we want to predict y using a linear combination of x by solving the least squares problem, $min_{\beta \in R^p} \sum_{i=1}^{n} (y_i - x_i'\beta)^2$. The prediction error can be decomposed into squared bias and variance, for the case of large p, the OLS estimate has low bias (zero bias) but suffers from high variance, which reduced the prediction accuracy. One way to improve the prediction ability is sacrificing some bias to achieve a lower variance by reducing the number of predictors included in the model. In addition, linear regression cannot identify the subset of important variables among a large number of predictors. So we need some variable selection methods to choose the "important" variables. Traditionally, statisticians use best-subset or stepwise selection methods to select significant variables, but both procedures have fundamental limitations: best-subset is computationally infeasible to do all subset selection when the number of predictors is large, and it is also extremely variable because of its inherent discreteness (Breiman, 1995; Fan and Li, 2001). Stepwise selection suffers from the high variability and in addition is often trapped into a local optimal solution rather than the global optimal solution. Furthermore, both procedures ignore the stochastic errors or uncertainty in the variable selection stage (Fan and Li, 2001; Shen and Ye, 2002). Since OLS often does poorly in both prediction and interpretation, penalization techniques have been proposed to improve OLS.

We need to find new technology for variable selection. In mathematics and statistics and particularly in the fields of machine learning, regularization refers to the process of introducing additional information in order to solve an ill-posed problem or to prevent overfitting. This information is usually of

the form of a penalty for complexity, such as restrictions for smoothness or bounds on the vector space norm.

Regularization is very important when dealing with high-dimensional data. We introduce several important regularization methods in statistical machine learning including ridge regression, lasso, and elastic net regularization.

1.2.3.1 Ridge Penalty

When building regression models for high-dimensional data which include many variables, collinearity is often a problem. One of the symptoms is that the estimate of regression coefficients may be very large, and the associated standard errors are very large as well. This means that the coefficients are not well defined. Ridge regression (Hoerl and Kennard, 1988) is designed to overcome the multicollinearity problem. It shrinks the estimated coefficients towards zero. Specifically, the ridge regression estimates are defined by

$$\hat{\beta}_{ridge} = arg\min_{\beta \in R^p} \sum_{i=1}^{n} (y_i - x_i'\beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = arg\min_{\beta \in R^p} \sum_{i=1}^{n} \|y_i - x_i'\beta\|_2^2 + \lambda \sum_{j=1}^{p} \|\beta\|_2^2$$

The penalty term is quadratic and $\lambda > 0$ is a tuning parameter which controls the strength of the penalty term. Note that we get OLS estimates when $\lambda = 0$, and $\hat{\beta}_{ridge} = 0$ when $\lambda = \infty$. The bias of the estimate increases as λ increases, while the variance decreases as λ increases. The tuning parameter is chosen to make the balance of the two trends and reduce the estimation errors.

Ridge regression performs well when some of the coefficients small, but is less dramatic when the entire coefficients are moderately large and the corresponding range for good values of λ is small.

Since ridge regression doesn't set coefficients exactly to zero unless $\lambda = \infty$, As a continuous shrinkage method, ridge regression achieves its better prediction performance through a bias–variance trade-off. However, ridge regression always keeps all the predictors in the model. Hence, it cannot perform variable selection thus does poorly in terms of offering a clear interpretation, even though it performs well in terms of prediction accuracy. However, as discussed before, in high dimensional dataset,

some variables are "noisy" and should not be included in the model to perform prediction that means we need to set their coefficients to be exactly zero. This is very important for model interpretation and prediction accuracy. The lasso penalty in next section can performs well in this situation.

1.2.3.2 Lasso Penalty

The least absolute shrinkage and selection operator (lasso), proposed by Tibshirani (1996), does both continuous shrinkage and automatic variable selection simultaneously. The lasso coefficient estimation is defined by

$$\hat{\beta}_{\text{lasso}} = arg \min_{\beta \in R^p} \sum_{i=1}^{n} \|y_i - x_i'\beta\|_2^2 + \lambda \sum_{j=1}^{p} \|\beta\|_1$$

Please note that lasso uses l_1 norm while the ridge uses l_2 norm. The l_1 penalty term is the sum of absolute values of coefficients and $\lambda > 0$ is a tuning parameter which controls the strength of the penalty term. Note that we get linear regression when $\lambda = 0$, and $\hat{\beta}_{lasso} = 0$ when $\lambda = \infty$. As in ridge penalty, the bias increases as λ increases while the variance decreases as λ increases in lasso penalty. Choosing an appropriate λ is a difficult problem.

Contrast to ridge regression which never sets coefficients to zero, lasso penalty can make some coefficient to be shrunken to zero exactly. As λ increases, more coefficients are shrunken to zero and more shrinkage is applied to those nonzero coefficients. The variable selection ability makes lasso different from ridge penalty. Compared to the classical variable selection methods such as subset selection, the LASSO has two advantages. First, the selection process in LASSO is continuous and hence more stable than the subset selection and stepwise procedure which are discrete and non-continuous. Second, the LASSO is computationally feasible for high-dimensional data. In contrast, computation in subset selection is combinatorial and not feasible when p is large. However, the lasso also has some limitations: As discussed by Tibshirani (1996), ridge regression dominates the lasso with regard to the prediction accuracy in the usual $n > p$ case if there are high correlations among the variables. Another drawback of the lasso solution is the fact that in $p > n$ situations, it selects at most n variables. Moreover, the lasso does

not group predictors as pointed out by Zou and Hastie (Zou and Hastie, 2005). If there is a group of highly correlated predictors, the lasso tends to select only some arbitrary variables from this group. Because of the above drawbacks, Lasso is inappropriate variable selection method in some situations.

The lasso penalty above forces the coefficients to be equally penalized in the penalty, sometimes it is not appropriate for some problems where different variables need to be penalized differently. (Zou, 2006) proposed an adaptive lasso penalty in which we can certainly assign different weights to different coefficients. It is defined by

$$\hat{\beta}_{\text{lasso}} = arg \min_{\beta \in R^p} \sum_{i=1}^{n} (y_i - x_i'\beta)^2 + \sum_{j=1}^{p} \lambda_j |\beta_j|$$

1.2.3.3 Elastic Net

Elastic net is a generalization of both ridge regression and the LASSO which includes both a linear and a quadratic term in the penalty. It is defined by

$$\hat{\beta}_{\text{lasso}} = arg \min_{\beta \in R^p} \sum_{i=1}^{n} (y_i - x_i'\beta)^2 + \lambda_2 \sum_{j=1}^{p} \beta_j^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j|$$

$$= arg \min_{\beta \in R^p} \sum_{i=1}^{n} \|y_i - x_i'\beta\|_2^2 + \lambda_2 \sum_{j=1}^{p} \|\beta\|_2^2 + \lambda_1 \sum_{j=1}^{p} \|\beta\|_1$$

Similar to the lasso, the elastic net simultaneously does automatic variable selection and continuous shrinkage, and it can select groups of correlated variables. The elastic net often outperforms the lasso in terms of prediction accuracy. The elastic net is particularly useful when the number of predictors (p) is much bigger than the number of observations (n).

1.3 Motivation of Dissertation

Nowadays, with the explosion of information in our world, analyzing big data makes it necessary to develop statistical technique which can uncover hidden patterns, unknown correlations and other useful information from large amounts of data.

Classification and clustering analysis are two common data mining methodology for high high-dimensional data analysis in which many variables are “noisy” or non-informative. These “noisy” variables contain redundant information due to the strong correlation or strong dependency among variables that may produce, for instance, multicollinearity. Finding structure in a high-dimensional variable space with a reduced subset of the original variables or new formed small dataset is important in both classification and cluster analysis.

Determining which variables are “important” can be a difficult task. The goal is to find a (small) subset of variables that can “explain” as much as possible of the information in the original dataset. These variables help us to better understand the multivariate structure, and, as a by-product, we find a dimension reduction procedure that can be used in a new dataset for the same problem.

The classical dimension reduction techniques (e.g., PCA or FA) often produce results that are difficult to interpret unless most of the coefficients (loadings) of the linear combination are not significant. Many new techniques are developed in literature.

1.4 Organization of the Dissertation

The remainder of this dissertation is organized as follows. In Chapter 2, ODC and SODC clustering methodology will be presented. Chapter 3 shows the comparison between two SPCA methods. SFABP is shown in Chapter 4. A brief discussion is provided in Chapter 5.

2 SPARSE OPTIMAL DISCRIMIANT CLUSTERING

2.1 Introduction

Optimal discriminant clustering (ODC) proposed by Zhang and Dai (2009) is based on optimal scoring, which was initially used in flexible discriminant analysis (FDA; Hastie et al., 1994). ODC was proposed as a clustering procedure and the tuning parameter selection problem hasn't been addressed. In this section, we advocate ODC as a dimension reduction tool for cluster analysis and propose a cross-validation method for tuning parameter selection. Furthermore, because in high-dimensional data many of the features may be non-informative for clustering, to obtain sparse solutions, we propose sparse optimal discriminant clustering (SODC) by adding a group-lasso type of penalty to ODC.

When conducting cluster analysis, we usually first draw some scatterplots for visualization purpose. However, drawing scatterplots is not an easy task for datasets having more than three features, especially when conducting cluster analysis, in which high dimensional clusters are not always visible in low dimensional plots. Very often scatterplots are drawn based on first few principal components derived from PCA. As demonstrated in Chang (1998), although PCA is a popular dimension reduction tool in many fields, it is not appropriate for cluster analysis because it does not take into account clustering structure.

An illustrative example is shown in Figure 2.1, where the simulated data consist of three clusters, each of 50 observations. The data have 10 independent and normalized features, of which the first two are informative and generated from bivariate normal distributions with centers being (2, 2), (2, -2) and (-2, 2) respectively, and the other eight are non-informative noise variables. The top left plot in Figure 1 is based on those two informative features; in practice, we don't know which features are informative and therefore we call this plot an oracle. The top right plot is based on the first two principal components from PCA, which doesn't display clearly the underlying three clusters. This failure of PCA for cluster analysis is the motivation for our reinvestigation of ODC, along with SODC in this manuscript. Both plots based on ODC and SODC in Figure 1 display three clusters clearly.

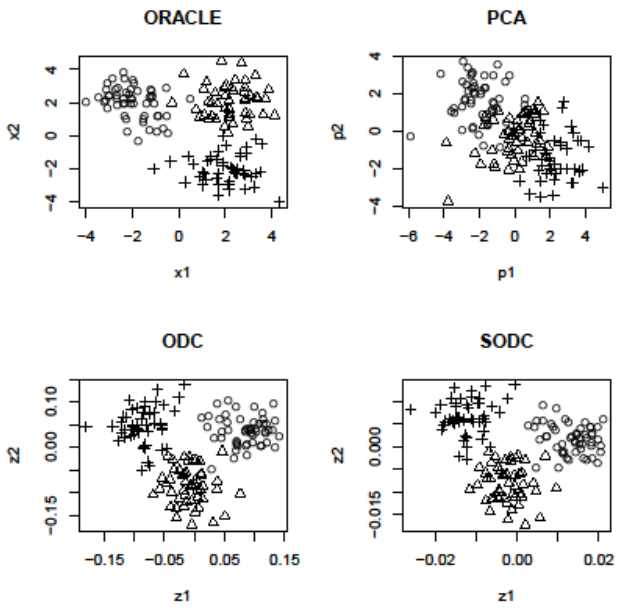

Figure 2.1 The simulated data consist of three clusters, each of 50 observations. There are 10 features, with two informative and eight non-informative. Top left: plot on those two informative features; Top right: plot on the first two principal components; Bottom left: plot on the first two ODC components; Bottom right: plot on the first two SODC components

In Section 2.2 we review ODC and propose a cross-validation method for tuning parameter selection. In Section 2.3, we propose SODC, along with a method for tuning parameter selection. In section 2.4, we examine the proposed methods through simulation studies and real data applications. In Section 2.5 there is some discussion and all the technical details are relegated to Appendix.

2.2 Optimal Discriminant Clustering

2.2.1 A Review of ODC

In this section, we review ODC proposed by Zhang and Dai (2009) and develop a method for selecting the tuning parameter in ODC. Assume that we are concerned with partitioning n p-dimensional observations $\{x_1, \dots, x_n\}$ into k clusters. Let $X = \{x_1, \dots, x_n\}'$ be the $n \times p$ design matrix and $\{x_1, \dots, x_n\}$ and $H_n = I_n - \frac{1}{n} 1_n 1_n{}'$ be the $n \times n$ centering matrix, where I_n is the $n \times n$ identity matrix and 1_n is the n-dimensional vector of ones.

To convert linear discriminant analysis (LDA) to the linear regression formulation, Hastie et al. (1994) defined a scoring matrix. Similarly, to convert cluster analysis to the linear regression formulation, Zhang and Dai (2009) defined the following scoring matrix.

Definition 1 An $n \times (k-1)$ *matrix Y is called the sample scoring matrix if it satisfies*

$$Y'Y = I_{k-1} \text{ and } 1_n'Y = 0.$$

The main part of ODC is a minimization process,

$$\widehat{W}, \hat{Y} = arg\, min_{W,Y} \|Y - H_n XW\|_F^2 + \lambda_2 \|W\|_F^2, \qquad s.t. Y'Y = I_{k-1} \text{ and } 1_n'Y = 0 \qquad (2.1)$$

Where W is a $p \times (k-1)$ matrix and $\|A\|_F = \sqrt{tr(A'A)}$ is the Frobenius norm of any matrix A. That is, ODC tries to assign low-dimensional (say, of dimension $J \leq k-1$) scoring vectors such that they can be best expressed as linear combinations of design matrix X. Close connection of ODC with two popular clustering algorithm, discriminative clustering algorithm (De la Torre and Kanade, 2006) and spectral clustering algorithm (Ng et al., 2002), was explored in the Subsection 3.3 of Zhang and Dai (2009).

After $\widehat{W}$ is obtained, scatterplots can be drawn and furthermore cluster analysis can be conducted based on $Z = H_n X\widehat{W}$, which is $n \times (k-1)$. Rewrite Z as $z^{(1)}, \ldots, z^{(k-1)}$ and call $z^{(1)}$ the first ODC component, $z^{(2)}$ the second ODC component, and so on. Therefore, by optimal scoring, the dimension is reduced from p to $k-1$, or from p to J if only the first J ODC components are used. The ODC algorithm is summarized as follows.

Optimal Discriminant Clustering (ODC) algorithm:

Step 1. Obtain $\widehat{W}$ from (2.1) for a given λ_2.

Step 2. Calculate $Z = (z_1, \ldots, z_n)' = H_n X\widehat{W}$.

Step 3. Perform k-means on z_i, $i = 1, \ldots, n$..

Step 4. Return the partition of z_i as the partition of x_i.

We conclude this brief review by remarking on some points which were not discussed in Zhang and Dai (2009). First, the problems of selecting tuning parameter λ_2 and number of clusters k were not

addressed in Zhang and Dai (2009). These two problems are discussed in Subsections 2.2 and 2.3, respectively.

Second, ODC components are linear combinations of the original features. Consideration of linear combinations provides better interpretation and is mathematically more convenient than other projection methods such as invariant coordinate selection (Tyler et al., 2009) and projection pursuit (Friedman and Tukey, 1974; Jones and Sibson, 1987). However, ODC may be less powerful than these well-known methods.

Third, ODC is affine equivariant, and this property is shared with invariant coordinate selection (Tyler et al., 2009). In addition, ODC belongs to the class of projection pursuit (Jones and Sibson, 1987), if we treat the object function in (1) as a projection index.

Fourth, k-means in Step 3 can be replaced by any other clustering procedure such as hierarchical clustering, k-medoids clustering, and spectral clustering. Choice of a clustering procedure may be suggested by scatterplots based on the first few ODC components.

2.2.2 Tuning Parameter Selection

The problem of selecting tuning parameter λ_2 was not discussed in Zhang and Dai (2009), although it was pointed out there that "the parameter λ_2 has a significant impact on the performance of ODC". Here we propose a cross-validation method for selecting λ_2 in ODC. First we randomly split the data into C roughly equal-sized parts (say $C = 5$), $X_1, \dots, X_C$ whose sizes are $n_1, \dots, n_C$. For a given λ_2 and a given $c \in (1, \dots C)$, let $\widehat{W}_{(-c)}^{\lambda_2}$ be the estimated W from (2.1) using the data leaving out X_C, and let

$$\hat{Y}_{(c)}^{\lambda_2} = arg\,min \left\| Y - H_{n_c} X_{(c)} \widehat{W}_{(-c)}^{\lambda_2} \right\|_F^2, \qquad \text{s.t.}\, Y'Y = I_{k-1} \;\; and \;\; 1_{n_c}' Y = 0$$

Now we can describe the cross-validation method as follows.

Cross-validation for selecting λ_2:

Step 1. Select a list of λ_2.

Step 2. Randomly split the data into $X_1, \dots, X_C$.

Step 3. Calculate $(\lambda_2) = \frac{1}{C}\sum_{c=1}^{C} \left\| \hat{Y}_{(c)}^{\lambda_2} - H_{n_c} X_{(c)} \widehat{W}_{(-c)}^{\lambda_2} \right\|_F^2$.

Step 4. Obtain $\hat{\lambda}_2 = argminCV(\lambda_2)$.

An illustrative example is shown in the left panel of Figure 3, "where one-standard-error" rule (e.g., Hastie et al., 2009, p. 244) is applied to detect the elbow point.

2.2.3 Selection of the Number of Clusters

Selection of number of clusters k is an important problem in cluster analysis. Many selection methods have been proposed and most of them are based on between-cluster and/or within-cluster sum of squared distances; to name just a few, Calinski and Harabasz (1974), Hartigan (1975), and Krzanowski and Lai (1988). Recently proposed methods include the silhouette statistic by Kaufman and Rousseeuw (1990), the gap statistic by Tibshirani et al. (2001), the jump statistic by Sugar and James (2003), and the stability selection by Wang (2010). Any of these methods can be applied to ODC.

We briefly describe the routine of selecting number of cluster k in ODC. For any given $k \in (1, \dots k_{max})$, where k_{max} is the largest number of clusters to be considered and some method may start with $k = 2$, execute ODC with λ_2 selected via the proposed crossvalidation, compute the value of the index under consideration (say, the gap statistic), and select $\hat{k}$ as the maximizer (or minimizer, depending on the index under consideration) of these computed indexing values. An illustrative example is shown in Figure 2.2, where the simulated dataset is the same as the one used for Figure 2.1. Both the gap statistic (Tibshirani et al., 2001; R function "clusGap") and the stability selection (Fang and Wang, 2012; R function "nselectboot" written by Prof Christian Hennig) correctly select $\hat{k} = 3$. This example also demonstrates that either k-means or any other clustering (say, hierarchical clustering) can be used in Step 3 in the ODC algorithm.

2.3 Sparse Optimal Discriminant Clustering

2.3.1 SODC

In this section, we are concerned with high-dimensional data where p is large. As suggested by ODC, scatterplots and further clustering are based on $Z = H_n X\widehat{W}$ after $\widehat{W}$ is obtained from (1). For simplicity, denote $\tilde{X} = H_n X$ as the centered design matrix. Let $W = (w_{ij})_{p\times(k-1)} = (w^{(1)}, \dots, w^{(k-1)}) = (w_1, \dots, w_p)'$, and use similar notation for other matrices such as $\widehat{W}$, $\tilde{X}$ and Z. Then $z_i = \sum_{j=1}^{p} \widehat{w}_j \tilde{x}_{ij}$; that is, the dimension-reduced observation z_i's are linear combinations of $\tilde{x}_{i1}, \dots \tilde{x}_{ip}$. However, it is hard to interpret z_i when it involves all the p features. Moreover, in situations where many features are not informative for clustering, including such features may cause poor performance of ODC. Therefore, we propose to add some penalty in (1) to obtain sparse solution of $\widehat{W}$.

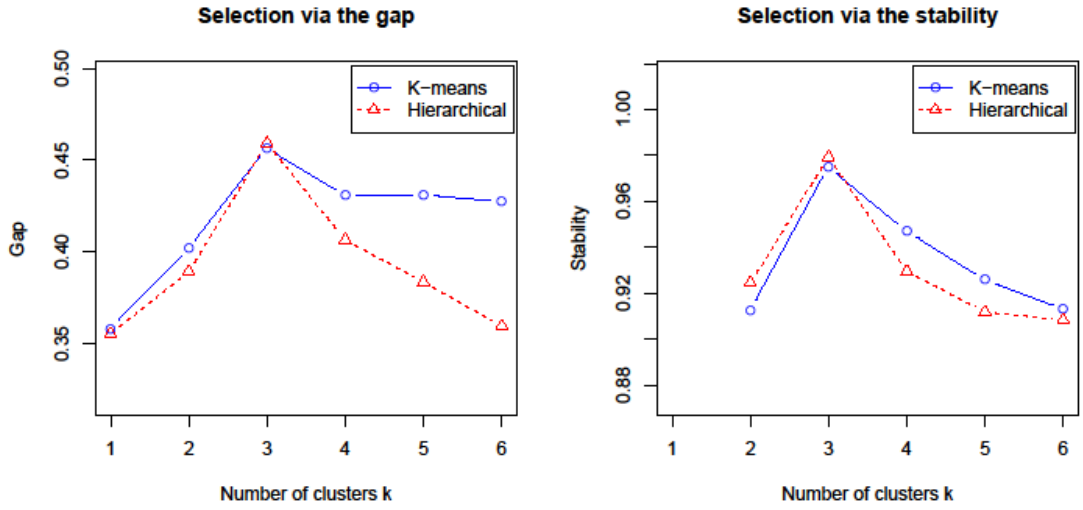

Figure 2.2 The dataset used is the same as the one used in Figure 2.1. K-means clustering and hierarchical clustering are applied in Step 3 of ODC, respectively. Both the gap statistic and the stability selection correctly select $\widehat{k} = 3$.

Note that if we want to exclude the influence of the jth feature, we should exclude it in all the $k-1$ ODC components, making all the components of $\widehat{w}_j$ zero. This is an all-in-all-out fashion, and therefore we propose to add a group-lasso type of penalty (Yuan and Lin, 2006) to ODC. Therefore, the minimization process in ODC becomes

$$\widehat{W},\hat{Y} = arg\min_{W,Y}\|Y - H_nXW\|_F^2 + \lambda_2\|W\|_F^2 + \lambda_1\sum_{j=1}^{p}\left\|\mathrm{w_j}\right\|_2 , \quad \text{s.t.}\, Y'Y = I_{k-1} \ and\ 1_n'Y = 0 \qquad (2.2)$$

Where $\left\|w_j\right\|_2$ is the Euclidean norm of w_j . After sparse $\widehat{W}$ is obtained from (2.2), scatterplots can be drawn and furthermore cluster analysis can be conducted based on $Z = H_nX\widehat{W}$, which is $n \times (k - 1)$. Rewrite Z as $(z^{(1)}, \dots, z^{(k-1)})$ and call $z^{(1)}$ the first SODC component, $z^{(2)}$ the second SODC component, and so on.

We should point out that recently many methods for integrated clustering and variable selection, either model-based or non-model-based, have been proposed. The long list includes methods proposed by Friedman and Meulman (2004), Raftery and Dean (2006), Steinley and Brusco (2008), Maugis et al. (2009), Witten and Tibshirani (2010), Bouveyron and Brunet (2012), and Sun et al. (2012a). When the main purpose is variable selection in cluster analysis, we suggest consider one of these existing methods.

Nonetheless, because the implicit linear formation in ODC offers good interpretation and is mathematically convenient, it is worthwhile to develop SODC to improve the performance of ODC. By the same reason, sparse principal component analysis was proposed by Zou et al. (2006) based on the linear formation of PCA, sparse canonical correlation analysis was proposed by Sun et al. (2008) based on the linear formation of CCA, and sparse discriminant analysis was proposed by Clemmensen et al. (2011) based on the linear formation of LDA.

2.3.2 Implementation

We propose an iterative minimization algorithm for (2.2). First, given W, the minimization with respect to Y is given in the following theorem. The proof is given in Appendix A.1.

Theorem 1 *Given W, the minimizer of (2.2) with respect to Y is $\hat{Y} = UV'$ where U and V are from the singular value decomposition $H_nXW = UDV'$.*

We propose to use a block-wise coordinate descent procedure (Yuan and Lin, 2006) to obtain the minimizer of (2.2) with respect to W for a given Y. For this aim, stack the columns of Y into an $n(k-1)$-vector, $\mathbf{Y}$, and let $\widetilde{\mathbf{X}}_j$ be the $n(k-1) \times (k-1)$ matrix diag($\tilde{x}^{(j)}, \dots, \tilde{x}^{(j)}$), where $\tilde{x}^{(j)}$ is the jth column

of $\tilde{X}, j = 1, \ldots, p$. Then, given Y (equivalently, given $\mathbf{Y}$), the minimization of (2.2) with respect to W becomes

$$\widehat{W} = arg\min_{W} \left\| \mathbf{Y} - \sum_{j=1}^{p} \widetilde{\mathbf{X}}_j w_j \right\|_F^2 + \lambda_2 \|w_j\|_2^2 + \lambda_1 \sum_{j=1}^{p} \|w_j\|_2 , \quad \text{s.t.} Y'Y = I_{k-1} \text{ and } 1_n' Y = 0 \qquad (2.3)$$

This minimization can be achieved by the block-wise coordinate descent procedure described in the following theorem. The proof is in Appendix A.2.

Theorem 2 *Given* $\mathbf{Y}$*, let the minimizer of (2.3) be* $\widehat{W} = (\widehat{w}_1, \ldots, \widehat{w}_p)'$*. Then*

$$\widehat{w}_j = \frac{(\|V_j\|_2 - \lambda_1/2)_+}{(1+\lambda_2)\|V_j\|_2} V_j$$

where $(a)_+ = a$ *if* $a > 0$ *and = 0 otherwise, and* $V_j = \widetilde{\mathbf{X}}_j(\mathbf{Y} - \sum_{l \neq j} \widetilde{\mathbf{X}}_l \widehat{w}_l)$.

From Theorem 2, $\widehat{w}_j = 0$ if $\lambda_1 \geq 2\|V_j\|_2$, so we can achieve sparse $\widehat{W}$ when λ_1 is large enough. Based on these two theorems, starting with some initializations $\hat{Y}_0$ and $\widehat{W}_0$, we can iteratively update $\widehat{W}$ and $\hat{Y}$ until they converge. Because the objective function in (2.3) is convex, the solution to (2.3) is unique and in our numerical studies the algorithm converged quickly after a few iterations.

In practice, we can compute a solution path along a list of λ_1, say $\{10^{-\tau + log_{10}(\lambda_{max}) \times l/L}\}$, $l = \{0, \ldots, L\}$, which ranges from a very small value $\lambda_1^0 = 10^{-\tau}$ to λ_1^{max}, Here λ_1^{max}, whose formula is in Appendix A.2, is the smallest value of λ_1 such that all features are excluded (i.e., $\widehat{w}_j = 0$, for $j = 1, \ldots, p$. When compute the solution for $\lambda_1 = \lambda_1^0$, we use the solution $\hat{Y}_0$ and $\widehat{W}_0$ obtained from ODC as initializations; when compute the solution for next λ_1 in the list, we use the solution obtained most recently as initializations; and so on.

2.3.3 Tuning Parameter Selection

Instead of selecting λ_2 and λ_1 at the same time, we fix λ_2 as the one selected by the cross-validation method in Subsection 2.2.2 and focus on selecting λ_1. Assume that in high dimensional data, only some features are informative for clustering and ideally those noninformative features should be ex-

cluded. It is a feature selection problem and the group lasso penalty in SODC plays an important role. For supervised learning, many feature selection criteria, such as AIC and BIC, have been proposed. However, for unsupervised learning, because there is not a clearly defined loss function, it is very hard to conduct feature selection.

Recently, the idea of clustering stability has been successfully applied to select the number of clusters. See, e.g., Fowlkes and Mallows (1983), Gnanadesikan (1997), Ben-Hur et al. (2002), Lange et al. (2004), Ben-David et al. (2006), Wang (2010), and Fang and Wang (2012). However, it has not been studied for feature selection in cluster analysis. In this section, we develop a stability selection method for selecting λ_2 in SODC.

The method is motivated by the kappa selection proposed by Sun et al. (2012b), where they were concerned with feature selection in linear regression. Because of the sparsity property of the group-lasso penalty (Theorem 2), for a given λ, an active subset of features will be selected (i.e., those features with non-zero w_j) when we apply SODC to a training dataset. Due to the randomness of data generation, different training datasets may produce different active subsets of features. The rationale behind the kappa selection is that an appropriate λ should produce stable active subsets of feature when we apply SODC to repeatedly generated training datasets.

Hypothetically, assume that there are two independent training datasets, producing two active subsets of features via SODC given λ_1, $\hat{A}_{1\lambda_1}$ and $\hat{A}_{2\lambda_1}$, respectively. The agreement of these two active subsets can be measured by Cohen's kappa coefficient (Cohen, 1960),

$$k\left(\hat{A}_{1\lambda_1}, \hat{A}_{2\lambda_1}\right) = \frac{Pr(a) - Pr(e)}{1 - Pr(e)},$$

Where $Pr(a) = \left(\left|\hat{A}_{1\lambda_1} \cap \hat{A}_{2\lambda_1}\right| + \left|\hat{A}^c_{1\lambda_1} \cap \hat{A}^c_{2\lambda_1}\right|\right)/p$ and $Pr(e) = \left(\left|\hat{A}_{1\lambda_1}\right|\left|\hat{A}_{2\lambda_1}\right| + \left|\hat{A}^c_{1\lambda_1}\right|\left|\hat{A}^c_{2\lambda_1}\right|\right)/p^2$

But in practice we have only one training dataset. In order to calculate the stability associated with λ_1, we repeatedly randomly split the data into two halves. In addition, assuming that there is at least one informative feature and one non-informative feature (excluding the two degenerate cases),

$k(\hat{A}_{1\lambda_1}, \hat{A}_{2\lambda_1})$ will be set as -1 if both $\hat{A}_{1\lambda_1}$ and $\hat{A}_{2\lambda_1}$ are empty or both are full. We describe the kappa selection method for λ_1 in the following.

Kappa selection method for selecting λ_1:

Step 1. Select a list of λ_1.

Step 2. Repeatedly randomly split the data into two halves X_1^{b*} and X_2^{b*} , $b = 1, \dots, B$.

Step 3. For any λ_1, apply SODC to $X_1^{b*}$and X_2^{b*} and obtained $\hat{A}_{1\lambda_1}^{b*}$and $\hat{A}_{2\lambda_1}^{b*}$.

Step 4. Calculate $k(\lambda_1) = \frac{1}{B}\sum_{b=1}^{B} k(\hat{A}_{1\lambda_1}^{b*}, \hat{A}_{2\lambda_1}^{b*})$.

Step 5. Select $\hat{\lambda}_1 = argmax k(\lambda_1)$.

We conclude this section with some remarks. First, the list of λ_1 in Step 1 can be decided according to the rule discussed at the end of the preceding subsection. Second, because estimating the average stability in Step 4 is accurate for a moderately large B, in our numerical studies, we use $B = 20$ and it works well.

Third, here we suggest select λ_2 first and then select λ_1, because SODC is proposed to improve the performance of ODC. In the first layer, we attempt to select an appropriate $\lambda_2$leading to "good" linear combinations of the original features, while in the second layer, we attempt to select an appropriate λ_1 leading to "sparse" linear combinations. Fourth, we can select number of clusters k and two tuning parameters λ_1 and λ_2 simultaneously by some criterion (say the gap statistic). However, searching tuning parameters over a two-dimensional or three-dimensional space is computationally intensive.

2.4 Numerical Results

2.4.1 Simulation Studies

We have created R package "SODC"" to implement the proposed methods, using it for the following numerical studies. Assume each simulated dataset consists of 100 p-dimensional observations $x_i, i = 1, \dots, 100$, which are generated as follows. First, C_i's are uniformly sampled from $\{1,2,3\}$, indicat-

ing the cluster memberships of the observations. Then for each i, the first q informative features are generated from $N(m(C_i), I_q)$, where q is even and

$$m(C_i) = \mu\left(-1_{\frac{q}{2}}', 1_{\frac{q}{2}}'\right) I(C_i = 1) + \mu 1_q I(C_i = 2) + \mu\left(1_{\frac{q}{2}}', -1_{\frac{q}{2}}'\right) I(C_i = 3),$$

and the remaining $p - q$ non-informative features are generated from $N(0_{p-q}, I_{p-q})$. Clearly, the three clusters are separated when μ is large, and overlapped when μ is small. When the number of noise variables $p - q$ increases, it becomes harder to distinguish the three clusters.

Table 2.1 Selection of number of clusters k via the gap statistic. Each simulation setting is replicated 50 times and the frequency distribution of the selected number of clusters kˆis reported.

p	μ	Distribution of $\hat{k}$					
		1	2	3	4	5	6+
10	2.0	0	0	49	0	1	0
	2.2	0	0	45	1	0	4
	2.4	0	0	48	1	1	0
50	2.0	0	0	50	0	0	0
	2.2	0	0	50	0	0	0
	2.4	0	0	50	0	0	0
100	2.0	0	12	38	0	0	0
	2.2	0	0	50	0	0	0
	2.4	0	0	50	0	0	0
200	2.0	27	23	0	0	0	0
	2.2	7	34	9	0	0	0
	2.4	0	7	43	0	0	0

First, we examine the performance of the gap statistic for selecting number of clusters k in ODC. Set $q = 2$, $p = 10, 50, 100, or\ 200$, and $\mu = 2.0, 2.2, or\ 2.4$. Tuning parameter λ_2 is selected over a grid of values $\{10^{-3+3\times l/19}\}$, where $l = 1, \ldots, 19$, using 5-fold cross-validation. Each simulation setting is replicated 50 times and the frequency distribution of the selected number of clusters $\hat{k}$ is reported in Table 2.1. It seems the gap statistic works very well when $p = 10, 50, and\ 100$. But it doesn't work well when $p = 200$ and $\mu = 2.0\ and\ 2.2$, which also indicates the necessity of conducting variable selection in cluster analysis.

Hereafter, we assume that the true number of clusters is known as 3 and we use the above settings excluding the settings with $p = 100$, along with letting $B = 20$ for selecting λ_1 in SODC over a grid of

values f$\{10^{-3+\lambda_1^{max}\times l/19}\}$, where $l = 1, \ldots, 19$ and λ_1^{max} is provided in Appendix A.2. All clustering algorithms are randomly started 100 times to overcome their dependence on the initialization. Each simulation setting is replicated 20 times.

Now we compare the performance of six clustering procedures: (1) k-means using only those q informative features (referred as ORACLE); (2) k-means using all features (ALL); (3) k-means using only the principal components from PCA whose corresponding eigenvalues are greater than the average (PCA; this rule was suggested by Cattell, 1966); (4) ODC with λ_2 selected via the proposed cross-validation method (ODC); (5) SODC with λ_1 selected via the proposed bootstrap method (SODC); and (6) k-means using the subset of features selected by SODC (SODC*). The results are summarized in Table 2.2. The clustering performance is evaluated by two typical measurements: adjusted rand index (ARI; Rand, 1971) and clustering error (CE; Zhang and Dai, 2009), which measure the agreement and disagreement between the resultant cluster memberships and the true cluster memberships, respectively. The larger NMI or smaller CE, the better the performance.

From Table 2.2, we see that, as a dimensional-reduction tool, both ODC and SODC perform much better than PCA, which is not working well when p is large. We also see that SODC improves ODC a lot when p is large. Moreover, we find that SODC* performs similarly with ORACLE, meaning that SODC performs well in terms of feature selection. Note that sometimes ODC performs worse than the regular k-means including all the features. This fact is actually the motivation of proposing SODC. In addition, it seems that SODC* always outperforms SODC, because when SODC shrinkages those non-informative effects to zero, it also introduces estimation bias to those informative effects.

Table 2.2 Comparing six clustering procedures: (1) using only informative features; (2) using all features; (3) using some PCA components; (4) using the first two ODC components; (5) using the first two SODC components; and (6) using only the features selected by SODC.

p	Method	$\mu = 2$		$\mu = 2.2$		$\mu = 2.4$	
		ARI (SD)	CE (SD)	ARI (SD)	CE (SD)	ARI (SD)	CE (SD)
10	ORACLE	0.90(0.05)	0.02(0.01)	0.93(0.04)	0.01(0.01)	0.96(0.03)	0.01(0.01)
	ALL	0.90(0.05)	0.02(0.01)	0.93(0.04)	0.02(0.01)	0.96(0.03)	0.01(0.01)
	PCA	0.70(0.09)	0.07(0.02)	0.73(0.09)	0.06(0.02)	0.82(0.06)	0.04(0.01)
	ODC	0.85(0.05)	0.03(0.01)	0.91(0.04)	0.02(0.01)	0.94(0.03)	0.01(0.01)
	SODC	0.88(0.06)	0.03(0.01)	0.93(0.05)	0.02(0.01)	0.95(0.03)	0.01(0.01)
	SODC*	0.90(0.05)	0.02(0.01)	0.93(0.04)	0.01(0.01)	0.96(0.03)	0.01(0.01)
50	ORACLE	0.91(0.03)	0.02(0.01)	0.95(0.03)	0.01(0.01)	0.96(0.03)	0.01(0.01)
	ALL	0.88(0.03)	0.03(0.01)	0.92(0.03)	0.02(0.01)	0.96(0.03)	0.01(0.01)
	PCA	0.64(0.10)	0.08(0.02)	0.72(0.08)	0.06(0.02)	0.79(0.06)	0.05(0.01)
	ODC	0.82(0.05)	0.04(0.01)	0.87(0.05)	0.03(0.01)	0.94(0.03)	0.01(0.01)
	SODC	0.86(0.08)	0.03(0.02)	0.93(0.04)	0.02(0.01)	0.96(0.03)	0.01(0.01)
	SODC*	0.91(0.03)	0.02(0.01)	0.95(0.03)	0.01(0.01)	0.96(0.03)	0.01(0.01)
200	ORACLE	0.92(0.05)	0.02(0.01)	0.94(0.02)	0.01(0.00)	0.96(0.03)	0.01(0.01)
	ALL	0.78(0.07)	0.05(0.02)	0.86(0.04)	0.03(0.01)	0.90(0.04)	0.02(0.01)
	PCA	0.59(0.11)	0.09(0.02)	0.67(0.07)	0.07(0.02)	0.75(0.08)	0.05(0.02)
	ODC	0.65(0.11)	0.08(0.02)	0.78(0.06)	0.05(0.01)	0.82(0.08)	0.04(0.02)
	SODC	0.84(0.07)	0.04(0.02)	0.92(0.03)	0.02(0.01)	0.93(0.04)	0.02(0.01)
	SODC*	0.92(0.05)	0.02(0.01)	0.94(0.02)	0.01(0.00)	0.96(0.03)	0.01(0.01)

Finally, we evaluate the feature selection performance of SODC. For this aim, set $q = 2$, $p = 10, 50, or\ 200$, and $\mu = 1.2, 1.4, or\ 1.6$. As discussed in Subsection 3.1, there are many existing methods for variable selection in cluster analysis. Here we only compare SODC with two representative and recent methods: (1) a model-based method proposed in Raftery and Dean (2006) and a non-model-based method proposed in Witten and Tibshirani (2010). For Raftery and Dean's method, we use R package “clustervarsel", where two optional algorithms are available, ‘greedy’ and ‘headlong’. Because selecting ‘greedy’ makes R running very slow, we select ‘headlong’. For Witten and Tibshirani's method, we use R package “sparcl”, where both sparse k-means clustering and sparse hierarchical clustering are available. Here we consider k-means for both Witten and Tibshirani's method and SODC, although we can also consider any other clustering procedure for SODC. The performance of sparse k-means depends on the selection of argument ‘wbound’. Because it doesn't perform well if the default option of argument ‘wbound’ is used, we set ‘wbound’ as seq(1.001, 1.1, 20).

The results of feature selection are summarized in Table 2.3, showing the percentage of selecting exactly the true subset of informative features among 20 replications, the average number of incorrectly selected noise variables, the average number of incorrectly excluded informative features, and the average size of the selected subset of features. Note that μ in these settings takes on smaller values than those in Table 2.2, because otherwise all methods will select the true best subset of features trivially.

From Table 2.3, we see that these three methods are comparable and very efficient in selecting the true subset including only the informative features when p is moderately large such as $p = 10$, or 50. It is still efficient for SODC and Witten and Tibshirani's method when $\mu = 1.4$ or $\mu = 1.6$ and p is large such as 200. We also see that the false negative rate is very small in all of the cases and the false positive rate is also small in most of the cases.

To illustrate the selection processes of λ_2 and λ_1, we randomly select one replication from the setting where $(n, \mu, q, p) = (100, 2.4, 2, 10)$ and display the estimated $CV(\lambda_2)$ via the cross-validation method and the estimated $k(\lambda_1)$ via the kappa method, respectively. In the left panel of Figure 2.3, the horizontal line is based on "one-standard-error" rule (e.g., Hastie et al., 2009, p. 244) and the vertical line indicates the location of the largest $\lambda_2(\hat{\lambda}_2 = 10^{1.8})$ whose $CV(\lambda_2)$ value is below the horizontal line. In right panel of Figure 2.3, we see that the kappa function $k(\lambda_1)$ is small when λ_1 is large or small, and achieves its maximum at $\hat{\lambda}_1 = 10^{0.18}$.

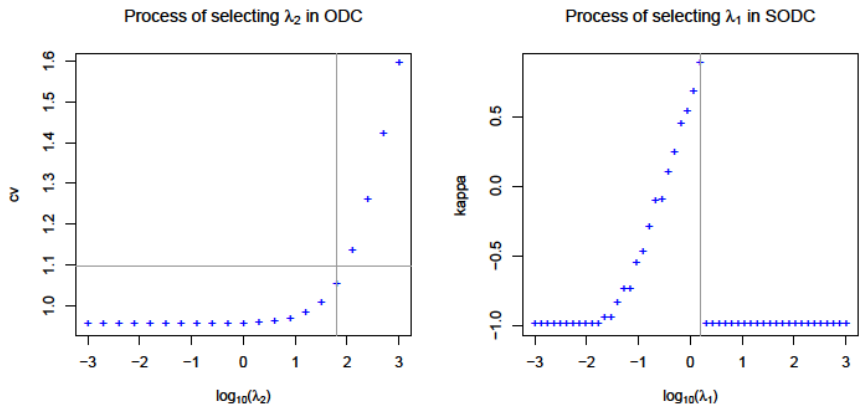

Figure 2.3 The processes of selecting λ_2 using cross-validation method and λ_1 using kappa method based on one realization. The selected tuning parameters are $\hat{\lambda}_2 = 10^{1.8}$ and $\hat{\lambda}_1 = 10^{0.18}$.

Table 2.3 Feature selection by SODC, compared with Raftery and Dean's model-based clustering with headlong algorithm (MCLH) and Witten and Tibshirani's sparse k-means clustering (SKM). True: the percentage of selecting exactly the true subset of informative features; FP: the average number of incorrectly selected noise variables; FN: the average number of incorrectly excluded informative features; Size: the average size of the selected subset.

		$\mu = 1.2$			$\mu = 1.4$			$\mu = 1.6$		
p	Method	True	FP/FN	Size	True	FP/FN	Size	True	FP/FN	Size
10	SODC	85%	0.15/0.00	2.15	95%	0.05/0.00	2.05	100%	0.00/0.00	2.00
	SKM	65%	2.05/0.00	4.05	100%	0.00/0.00	2.00	100%	0.00/0.00	2.00
	MCLH	70%	0.10/0.20	1.90	100%	0.00/0.00	2.00	100%	0.00/0.00	2.00
50	SODC	75%	0.30/0.00	2.30	95%	0.05/0.00	2.05	100%	0.00/0.00	2.00
	SKM	35%	3.40/0.00	5.40	100%	0.00/0.00	2.00	95%	0.05/0.00	2.05
	MCLH	65%	0.70/0.25	2.45	90%	0.15/0.00	2.15	90%	0.15/0.00	2.15
200	SODC	0%	3.15/0.10	5.05	70%	0.30/0.05	2.25	100%	0.00/0.00	2.00
	SKM	25%	6.70/0.00	8.70	70%	2.00/0.00	4.00	90%	0.40/0.00	2.40
	MCLH	25%	4.75/0.70	6.05	30%	1.55/0.00	3.55	35%	1.25/0.00	3.25

2.4.2 Real Dataset Application

We examine the proposed methods using five real datasets downloaded from UCI website http://archive.ics.uci.edu/ml/datasets.html: (1) Wine; (2) Wisconsin Breast Cancer (WBC); (3) Libras Movement (LM); (4) Dermatology (DERM); and (5) Semeion Handwritten Digit (SHD). Some descriptions of these datasets are in Table 2.4 and see the UCI website for further descriptions. We apply the ODC algorithm with λ_2 selected by the cross-validation method and the SODC algorithm with λ_1 selected by the kappa method, along with the k-means algorithm. We assume that the available class assignments are actually the cluster assignments and assume the number of clusters are known. The results are summarized in Table 2.5.

Table 2.4 Some information of the five UCI datasets, where k is the number of classes (treated as clusters here), p is the number of features, and n is sample size.

Dataset	k	p	n
Wine	3	13	178
WBC	2	30	569
LM	15	90	360
DERM	6	33	366
SHD	10	256	1593

To illustrate the real application, we examine Wine data in more detail. The selection processes of selecting number of clusters k via the gap statistics and the stability are shown in the top panel of Figure 2.4. When hierarchical clustering is applied, both selection methods correctly select $\hat{k}$ as 3. When k-

means clustering is applied, the stability selection correctly selects $\hat{k}$ as 3, and the elbow point of the gap curve is also 3. Scatterplots based on the first two ODC components and the first two SODC components are shown in the bottom panel of Figure 2.4. Both scatterplots display three clusters clearly, and in the SODC scatterplot three clusters are completely separated.

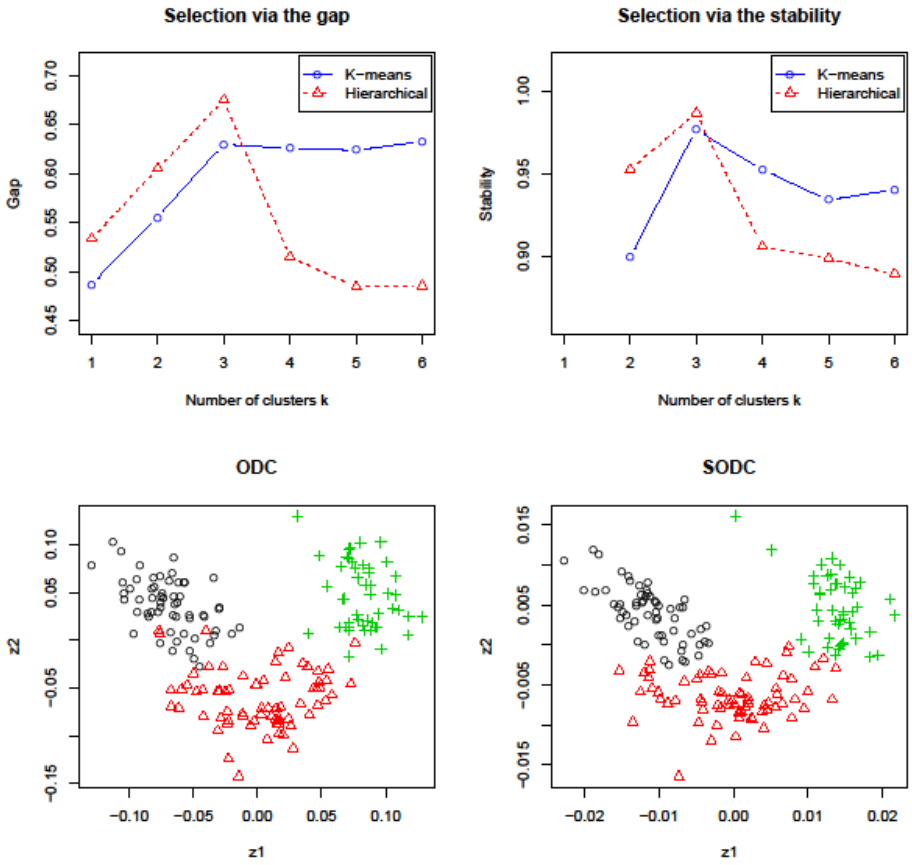

Figure 2.4 Wine data. The selection processes of selecting k via the gap statistics and the stability are shown in the top panel; both k-means and hierarchical clusterings are applied. Scatterplots based on first two ODC components and first two SODC components a re in bottom panel.

The application of the proposed methods to the five UCI datasets are summarized in Table 2.5. It seems ODC performs similarly as k-means for the first four datasets. For SHD data, where both k and p are large, ODC performs worse than k-means, but SODC improves ODC significantly. For all the five datasets, SODC* performs similarly as k-means, but SODC* uses much fewer features. This achievement is worthwhile, especially in medical studies, because it is safer, cheaper, and faster to make a diagnostics

using fewer features (or diagnostic markers). For example, for WBC data, SODC and SODC*, which use only 16 features to classify breast cancer types, perform similarly as k-means using 30 features.

Table 2.5 The results of applying different algorithms to those five UCI datasets. K-means clustering and ODC use all the available features. q^is the number of features selected by SODC. SODC* performs k-means using only the features selected by SODC.

Dataset	Method	ARI	CE	$\hat{q}$
Wine	k-means	0.90	0.02	13
	ODC	0.91	0.02	13
	SODC	0.83	0.04	7
	SODC*	0.85	0.03	7
WBC	k-means	0.68	0.08	30
	ODC	0.66	0.08	30
	SODC	0.65	0.09	16
	SODC*	0.66	0.08	16
LM	k-means	0.32	0.05	90
	ODC	0.35	0.05	90
	SODC	0.25	0.05	36
	SODC*	0.30	0.05	36
DERM	k-means	0.71	0.05	33
	ODC	0.71	0.05	33
	SODC	0.72	0.04	23
	SODC*	0.73	0.04	23
SHD	k-means	0.35	0.06	256
	ODC	0.28	0.07	256
	SODC	0.31	0.07	146
	SODC*	0.33	0.07	146

2.5 Discussion

Here we reinvestigate an existing method, ODC, and advocate it as a dimension reduction tool for cluster analysis. We propose a cross-validation method for selecting the tuning parameter in ODC. We also examine the performance of using existing methods such as the gap statistic and the stability selection to select the number of clusters in ODC. As a dimension reduction tool for cluster analysis, ODC performs much better than PCA, which does not take into account the clustering structure.

Furthermore, we propose SODC by adding a group-lasso type of penalty on ODC to conduct cluster analysis and feature selection simultaneously. The proposal of SODC is parallel to that of spare principal component analysis by Zou et al. (2006), sparse canonical correlation analysis by Sun et al. (2008), and sparse discriminant analysis (SDA) by Clemmensen et al. (2011). Take the last one as an ex-

ample. On the one hand, FDA in Hastie et al. (1994) is a method for classification analysis using optimal scoring, and then SDA is a "sparse" version of FDA by adding a lasso type of penalty. On the other hand, ODC is a method for cluster analysis using optimal scoring, and then SODC is a "sparse" version of ODC by adding a group-lasso type of penalty.

However, the selection of tuning parameters in SDA is different from that in SODC. For classification, the selection of tuning parameters in SDA can be guided by misclassification error. For cluster analysis, the problem of selecting the tuning parameter in SODC is much harder. Here we propose a method called kappa selection. The kappa selection is proved to be asymptotically consistent for linear regression, but the consistency hasn't be addressed under the current setting of cluster analysis.

Finally, both ODC and SODC, along with many other existing tools such as projection pursuit, can be used as a dimension reduction tool for drawing scatterplots before conducting analysis. Drawing such scatteplots are very important in that by examining such plots we can have rough ideas about some important issues such as: if there is no true clustering (i.e, $k = 1$), if the clusters are normally distributed, if the clusters are convex, and if some clusters are much bigger than the others. In the literature of cluster analysis, such issues have been addressed to some extent and many special methods dealing with these issues have been proposed. In practice, we should apply several different clustering procedures to a same dataset to see if the results are stable and the findings are consistent.

3 SPARSE PRINCIAL COMPONENT CLASSIFICATION

3.1 Introduction

As we discussed in chapter 1, PCA is a popular feature extraction and dimension reduction tool, seeks the linear combinations of the original variables called PCs such that the derived PCs capture maximal variance and guarantee minimal information loss. However, the classic PCA has major practical and theoretical drawbacks when it is applied to high-dimensional data. The classic PCA produces inconsistent estimates in high-dimensional situations. The loadings of PCs are typically nonzero. This often makes it difficult to interpret the PCs and identify important variables. The formula for standard PCA is:

Given a data matrix with n observations and p variables, the data are first centered on the means of each variable.

The first principal component is calculated to account for the largest possible variance in the dataset and can be written by:

$$PC1 = arg \max_{\sum_{j=1}^{p} a_{1j}^2 = 1} a_1' XX' a_1$$

The second PC is calculated in the same way, with the condition that it is uncorrelated with PC1 and account for the next highest variance.

Many SPCA methods were raised to solve the curse of dimensionality problem, such as (Zou et al., 2006; Lee et al., 2012). In this chapter, three existing SPCA methods are reviewed and a comparison of their application to a real world dataset, ancestry-informative markers in genome-wide association studies (GWAS), is given. This dataset is also used by Lee et al., (2012).

3.2 PCA, L-PCA, and AL-PCA

For a dataset with n rows of observations by p columns of variables, Lee et al., (2012) formulates the classical PCA as alternating-regression algorithm by trying to get the PCs sequentially. Then modify the alternating regression algorithm for standard PCA by adding either a lasso or alternative lasso penalty on the PCs to perform sparse PCA.

3.2.1 Reformulation of Standard PCA

The steps to get first principal components PC1:

Let PC1 denoted by $v = (v_1, \dots, v_p)'$ where v_j denotes the j-th variable contributing to the PC1 and denoted PCscore1 by $a = (a_1, \dots, a_n)'$. Then we use an iterative process to minimize the following sum of squares to estimate v and a.

$$\sum_{i=1}^{n} \sum_{j=1}^{p} (x_{ij} - a_i v_j)^2 \tag{3.1}$$

The idea is:

1) Given an initial value for a.
2) Estimate v_j for each j as the slope of a linear regression model with no intercept using $x_{ij}'s (i = 1, \dots, n)$ as the response and $a_i's$ as the independent variable. Thus we get $v_j = \sum_{i=1}^{n} x_{ij} a_i / \sum_{i=1}^{n} {a_i}^2$ for $(j = 1, \dots, p)$.
3) Fixing v and normalizing the v vector to have unit length, we estimate each a_i as the slope from a linear regression model with no intercept using $x_{ij}'s$ as the response and $v_j's$ as the independent variable. Thus we get $a_i = \sum_{j=1}^{p} x_{ij} v_j$ for $(i = 1, \dots, n)$.
4) Repeat (2) (3) until convergence.

The steps to get other principal components PCs:

After we get PC1 and PCscore1, we replace x_{ij} by $x_{ij} - a_i v_j$ and use formula (3.1) to identify the second PC and its scores as PC2 and PCscore2. We can continue this process until we estimate the desired number of PCs.

3.2.2 Sparse PCA with LASSO (L-PCA)

L-PCA method tries to estimate PCs and their corresponding PC scores by minimizing the following modified criterion from (3.1):

$$\sum_{i=1}^{n} \sum_{j=1}^{p} (x_{ij} - a_i v_j)^2 + 2\lambda \sum_{j=1}^{d} |v_j| \tag{3.2}$$

Where λ is a positive penalty parameter that controls sparsity of the derived PC. The larger λ, the sparser PC. The solution of L-PCA is the same as the traditional PCA when $\lambda = 0$.

L-PCA uses the same regression model described in section 3.2.1 to estimate a_i given v, and estimate v_j using soft-threshold solution:

$$v_j = \frac{1}{\sum_{i=1}^{n} {a_i}^2} \cdot sgn\left(\sum_{i=1}^{n} x_{ij} a_i\right)\left\{\left|\sum_{i=1}^{n} x_{ij} a_i\right| - \lambda\right\}_+ \tag{3.3}$$

Where sgn(•) denote the sign function and { • }+ denote a truncation function that returns its argument if it is nonnegative or 0 if it is negative.

(3.3) shows that the estimate of v_j is shrunk to 0 if the magnitude of $\sum_{i=1}^{n} x_{ij} a_i$ is smaller than the penalty parameter λ . Consequently, that particular variable will not contribute to the loading of the PC. We iteratively solve for a and v using ordinary and lasso regression, respectively, until convergence.

3.2.3 Sparse PCA with Alternative LASSO (AL-PCA)

AL-PCA method tries to estimate PCs and their corresponding PC scores by minimizing the following modified criterion from (3.1):

$$\sum_{i=1}^{n} \sum_{j=1}^{p} \left(x_{ij} - a_i v_j\right)^2 + 2\lambda \sum_{j=1}^{p} \frac{|v_j|}{|\hat{v}_j|} \tag{3.4}$$

(3.4) show that the penalty function for AL-PCA depends on $|\hat{v}_j|$. $|\hat{v}_j|$ is now shrunk by an variable-specific threshold value $\frac{\lambda}{|\hat{v}_j|}$, so the variable with larger loading on the PC will be associated a smaller penalty than those with a variable with smaller loading and thus the AL-PCA will have more penalty on those insignificant variables. AL-PCA often identifies a smaller set of variables than L-PCA with larger loadings.

AL-PCA uses the same regression model described in section 3.2.1 to estimate a_i given v, and estimates v_j using the following soft-threshold solution:

$$v_j = \frac{1}{\sum_{i=1}^{n} {a_i}^2} \cdot sgn\left(\sum_{i=1}^{n} x_{ij} a_i\right)\left\{\left|\sum_{i=1}^{n} x_{ij} a_i\right| - \frac{\lambda}{|\hat{v}_j|}\right\}_+ \tag{3.5}$$

Lee et al., (2012) proposes a likelihood-based procedure by calculating BIC to select the tuning parameter λ for L-PCA and AL-PCA .

The algorithm for L-PCA and AL-PCA is:

1) Scaling and centering the original $n \times d$ matrix X.
2) Set the initial value of a. We propose to set the initial value of a using Xv, where v is the first right-singular vector of X (equivalently the first PC score from PCA of $X'X$).
3) Find PC v using soft thresholding (3.3) or (3.5) and then normalize it.
4) Compute PC score $a = Xv$.
5) Repeat 3–4 until convergence. This step may be conducted multiple times on grids of λ for penalty parameter selection.
6) $X = X - \hat{a}\hat{v}'$ for the next PC. Here â and v̂ are derived from standard PCA without penalty.
7) Repeat (2)–(6) k times for obtaining k PCs.

3.3 Sparse Principal Component by Choice of Norm (SPCABP)

Qi et al., (2012) propose a new criterion-based sequential sparse PCA method (SPCABP) by replacing the l_2-norm in traditional eigenvalue problems with a new norm, which is a convex combination of l_1 and l_2 norms. The optimization problems in his methods are natural extensions of those in classic PCA and have relatively simple forms. An efficient iterative algorithm was proposed to solve these optimization problems. He also proves the convergence of this iterative algorithm and provides the detailed characterization of the limits. With this method, we can efficiently obtain uncorrelated PCs or orthogonal loadings, and achieve the goal of explaining high percentage of variations with sparse loadings.

Qi et al., (2012) defines the following "mixed norm":

$$\|v\|_\lambda = [(1-\lambda)\|v\|_2^2 + \lambda\|v\|_1^2]^{\frac{1}{2}}, \quad \forall\, v \in R^p\,. \tag{3.6}$$

This norm becomes to the l_2-norm when $\lambda = 0$, and l_1-norm when $\lambda = 1$.

And the first PC is given as follows:

$$\max_{v \in R^p, \|v\|_2 = 1} \frac{v' \Sigma v}{\|v\|_{\lambda_1}^2} \qquad (3.7)$$

(3.7) is equivalent to the following problem:

$$\max_{u \in R^p} u' \Sigma u \quad , subject\ to\ \|u\|_{\lambda_1} \leq 1 \qquad (3.8)$$

The higher order sparse PCs are defined either by (3.9) or by (3.10) as follows:

$$\max_{\substack{\|v\|_2 = 1,\ v \perp v_j, \\ j = 1, \ldots, k-1}} \frac{v' \Sigma v}{\|v\|_{\lambda_k}^2} \qquad (3.9)$$

Where λ_k is the tuning parameter for v_k.

$$\max_{\substack{\|v\|_2 = 1,\ v \perp \Sigma v_j, \\ j = 1, \ldots, k-1}} \frac{v' \Sigma v}{\|v\|_{\lambda_k}^2} \qquad (3.10)$$

When (3.9) is used, the obtained PCs are orthogonal to each other, and when (3.10) is used, the obtained PC scores are uncorrelated with each other.

3.4 Numerical Results

3.4.1 HapMap II study

We compare L-PCA, AL-PCA, and SPCABP by applying them to a real high dimensional data set, the International HapMap Project (The International HapMap Consortium, 2005) for AIM selection to genome-wide SNP data.

The dataset consists of 90 subjects of European ancestry (Utah residents with ancestry from northern and western Europe; CEU) from 30 parent-offspring trios, 90 subjects of African ancestry (Yoruba in Ibadan, Nigeria; YRI) from 30 parent- offspring trios, and 90 unrelated subjects of Asian ancestry (45 Han Chinese in Beijing, China; CHB, and 45 Japanese in Tokyo, Japan; JPT). We use unrelated subjects only and so exclude the CEU and YRI off- spring from analysis. Consequently, the final data set used for the comparison of sparse PCA methods consists of 210 subjects (60 CEU, 60 YRI, 45 CHB, 45 JPT).

I use the preprocessed dataset provided by Lee et al., (2012). The dataset was reduced from the initial 3,976,554 number of SNPs to 24,395 independent SNPs for AIM selection by applying quality-control procedures to exclude problematic SNPs from the analysis.

Lee et al., (2012) indicate that the top two PCs were sufficient for explaining the genetic variability within the sample due to ancestry. So I apply standard PCA, L-PCA, AL-PCA, and SPCABP to obtain the top two PCs to identify AIMs. Table 3.1 lists the number of nonzero loadings in principal components 1(PC1) and principal components 2 (PC2) obtained from different methods. Figure 3.1 suggests SPCA methods provide similar cluster patterns of the population structure as the traditional PCA, but the SPCA methods use much less SNPs than the latter method for constructing PCs. Specially, L-PCA and AL-PCA only use 1/3 of the total snps, and SPCABP only use 1/40 of the total snps, while PCA use all the total 24395 snps. We also present the scatterplots of individual scores from traditional PCA, L-PCA, AL-PCA, and SPCABP in Figure 3.1. The results show that the scores from traditional PCA and the three sparse PCA methods are almost identical and can clearly separate the three distinct ethnic groups.

Table 3.1 Number of non zero loadings for Principal components 1 (PC1) and Principal components 2 (PC2)

Method	PC1	PC2
PCA	24395	24395
L-PCA	8926	7829
AL-PCA	8810	7201
SPCABP	595	582

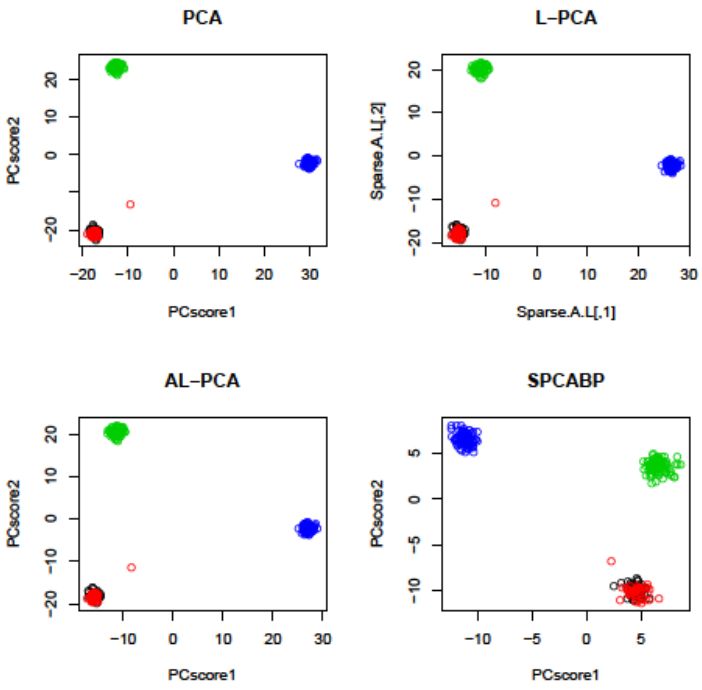

Figure 3.1 Scatterplots of the individual scores from traditional PCA, L-PCA, AL-PCA, and SPCABP.

3.5 Discussion

This chapter focuses on using three sparse PCA to identify AIMs and show that the method can identify a small set of markers that capture genetic ancestry as efficiently as genome-wide SNP data can. We can also apply sparse PCA directly to genome-wide marker data for the purpose of population-stratification adjustment in GWAS studies and whole-genome/whole genome sequencing studies. The results show that SPCABP can get similar classification results with even less number of markers than L-PCA and AL-PCA. All the three sparse PCA method outperforms the traditional PCA.

4 SPARSE FACTOR ANALYSIS BY PROJECTION

4.1 Introduction

The presence of collinearity is both a blessing and curse for data analysis, it means that only a small number of variables are needed to capture most of the variations in the data set, but we have to select the variables very carefully.

Factor analysis is useful tool when there is a high degree of cross-correlation between variables. It is a general purpose technique for dimensionality reduction with applications in diverse areas including psychometric, economics and computational biology. Factor analysis (Gorsuch, 1983) models the observed multivariate random variables as linear combinations of some unobserved (hidden or latent) factors plus error terms. Factor analysis was first introduced by (Spearman, 1904) to support his psychological theory of intelligence.

As discussed in chapter 1, there are two major kinds of FA: EFA in which researcher does not know the factor structure prior to running the analysis and CFA in which researcher "knows" the factor structure prior to the analysis. EFA can be done in two ways:

R-mode factor analysis: A variable based factor analysis aiming to simplify a matrix of variables by forming a smaller number of latent factors, k, that are linear combinations of the original p variables in the data set. The methods seeks to preserve the variance of the original variables in the new linear combinations (R-mode factors), and thus explain the maximum amount of variance in the data set. The first important step in this method is to transform the n sample by p variable data matrix X, into the associated variable covariance or correlation matrix, R, which is done by pre-multiplying the data matrix X by its transpose: $R = X'X$. Here R is a square $(p\ x\ p)$ matrix whose dimensions are determined by the number of variables (p) in the data set.

Q-Mode factor analysis: A sample based factor analysis aiming to simplify a large matrix of variables measured on many samples. The methods seeks to preserve the "information" within the samples of the original data set in the new linear weighting of contributions from the various factors(Q-mode fac-

tors) that are determined from the data. The first important step in this method is to transform the n sample by p variable data matrix X, into the sample similarity matrix, Q, which is formed by post-multiplying the data matrix X by its transpose: $Q = XX'$, Here Q is a square $(n\ x\ n)$ matrix whose dimensions are determined by the number of samples (n) in the data set.

In this dissertation, we only discuss the EFA and R-mode factor analysis.

4.1.1 Rotation Technique in FA

In FA, in order to get better interpretation of the factors, we often rotate the estimated factor loading matrix. As Yaremko et al., (1986) said: "In factor or principal-components analysis, rotation of the factor axes (dimensions) identified in the initial extraction of factors, in order to obtain simple and interpretable factors."

Two main types of rotation are used: orthogonal rotation and oblique rotation. Orthogonal rotation assumes that the factors in the analysis are orthogonal (uncorrelated). Gorsuch (1983) lists four different orthogonal methods: varimax, quartimax, equimax, and orthomax. In contrast, oblique rotation methods assume that the factors are correlated. They are defined as:

Varimax Rotation: maximizes the squared factor loadings in each factor, i.e., simplifies the columns of the factor loading matrix. In each factor the large loadings are increased and the small ones are decreased so that each factor only has a few variables with large loadings (but possibly the same variables can be relevant to explain two or more factors).

Quartimax Rotation: maximizes the variance of the squared factor loadings in each variable, i.e., simplifies the rows of the factor loading matrix. In each variable the large loadings are increased and the small ones are decreased so that each variable will only load on a few factors (but one factor may need to be interpreted by a large number of variables).

Equimax Rotation: Combines the objectives of both varimax and quartimax rotations. equamax maximizes a weighted sum of the varimax and quartimax criteria, reflecting a concern for simple structure

within variables as well as within factors. The number of variables that load highly on a factor and the number of factors needed to explain a variable are minimized.

Orthomax Rotation: Like Equimax, orthomax Rotation is a compromise between Varimax and Quartimax, but the researcher need to decide the relative weight of the Varimax component versus the Quartimax one. Parsimax is a specific case of Orthomax rotation where the relative weights of the Varimax and Quartimax rotations depends on the number of variables and factors.

Oblique Rotation: will produce factors that are not independent. There are major implications in using oblique rotation methods. Although they can lead to further simplification of the factor matrix, it becomes less straightforward to evaluate the contribution of each factor in explaining the original variability.

4.1.2 Formation of Classic FA

Assume that we are concerned with forming p n-dimensional variables $\{x_1, \dots, x_p\}$ into k factors where $x_i's, i = 1, \dots, p,$ are independent and identically distributed (i.i.d.). Let $X = \{x_1, \dots, x_p\}'$ be the $p \times n$ matrix where each row is a variable and each column is an observation. Without loss of generality, assume the mean of x_i is zero and the covariance of X is Σ. The factor model is represented by

$$X_{p\times n} = \Lambda_{p\times k} F_{k\times n} + \varepsilon_{p\times n} \tag{4.1}$$

Where Λ is a $p \times k$ matrix of factor loadings, $F = \{f_1, \dots, f_k\}' \in R^n$ and $\varepsilon = \{\varepsilon_1, \dots, \varepsilon_p\}' \in R^n$ are called common factors and unique factors, respectively, and F and ε are multivariate-normally distributed and should satisfy the assumption $E(F) = 0, E(\varepsilon) = 0, E(FF') = I_k, E(\varepsilon\varepsilon') = \Psi,$ and $E(F\varepsilon') = O$, i.e., F and ε are independent. Then we have $\Sigma = \Lambda\Lambda' + \Psi$. The proportion of variance of a given variable that is explained by the common factors is communality, and uniqueness is the variability of a variable minus its communality.

The factor model (4.1) is invariant under an orthogonal rotation, thus we can use the orthogonal rotation technique presented in section 4.1.1 such as varimax to yield either large or small loadings. Often, small loadings are further truncated at some threshold (e.g. 0.01), for zero loadings greatly enhance the

interpretability. The communalities are unchanged from the unrotated to the rotated solution. The goal of rotation is to obtain a simpler factor loading pattern that is easier to interpret than the original factor pattern.

4.1.3 Motivation of Sparse factor analysis (SFA)

Sparse factor analysis is a natural extension that can be motivated by the observation that sparse features tend to generalize better, or justified based on a priori beliefs about the underlying generative model of the observed data.

In this chapter we propose a novel sparse factor analysis model which expands on our previous work on sparse principal components analysis by projection models. We study its performance both on simulated data, and on a data set regarding snp that has been previously analyzed in the literature. The numerical results prove that SFABP performs much better when applied to high dimensional dataset.

4.2 Sparse Factor Analysis by Choice of Norm

4.2.1 SFABP

We propose a novel sparse factor analysis model called SFABP for multinomial data which was based on SPCABP in section 3.3. SFABP model provides a parsiminious lower-dimensional representation of multivariate continuous data by imposing structure upon the covariance matrix of a latent normal parameter. This confers a number of advantages over traditional models. First, the factor can be used as a powerful exploratory tool for investigating underlying structure in continuous data. Second, it can be used to create well-behaved shrinkage estimators that make the multivariate model viable even when the number of variables is large relative to the number of observations. The use of sparsity contributes additional regularization, and also provides a natural framework for investigating the number of factors driving the observed covariation. Finally, the factor model offers significant computational gains, as it circumvents the need to do further analysis like classification or clustering directly from high-dimensional dataset.

4.2.2 Implementation

In (75), Section 4.3 derives the relationship between PCA and FA. i.e. we can take the eigenvectors (of unit length) and weight them with the square root of the corresponding eigenvalue.

We illustrate this idea as follows:

For a given data matrix X with n observation and p variables, denote the SVD of X as $X = UDV'$, where U is an $n \times r$ orthnormal (i.e. $UU' = I$) column-wise matrix, contain the r eigenvectors of $Q = XX'$, V is an $p \times r$ orthnormal column-wise matrix, contain the r eigenvectors of $R = X'X$, and D is a real, positive, diagonal $r \times r$ matrix with diagonal as the singular values of X sorted in descending order.

$R = X'X = D^2 = Id' \;\; \Lambda = I\sqrt{d'}$

The factor loadings for each variable on each component can be obtained by multiply the eigenvectors times their singular values(i.e. the square roots of the eigenvalues) .

The principal component score is XV, and the factor scores is XVD. It is clear that they are essentially the same thing except that the factor scores have been scaled by the magnitude of the singular values (i.e. the square root of the length of the eigenvectors).

Based on the relationship between PCA and FA, we proposed SFABP based on SPCABP. SPCABP only gives its PCs when do analysis, we propose the following methods to derive eigenvalues given PCs. Suppose the first k PCs is sufficient to account for the variance of X, For ith eigenvector $e_i, j = 1, \dots, k$. $Cov(X) = \sum_{j=1}^{k} d_j e_j \, e_j' + \psi$. $e_i' Cov(X) e_i = e_i' (\sum_{j=1}^{k} d_j e_j \, e_j') e_i + e_i' \psi e_i$, since $e_i' e_j = 1$ if $i = j$, 0 otherwise, and we consider $e_i' \psi e_i \approx 0$ in a good FA model. Thus $d_i = e_i' Cov(X) e_i$, and $\psi = Cov(x) - \sum_{j=1}^{k} d_j e_j \, e_j'$.

So the algorithm for SFABP is as follows:

1) Use SPCABP to get first *k* orthogonal PCs which can be indicated as $\hat{P}$.
2) Use $d_i = e_i' Cov(X) e_i, \;\; i = 1, \dots, k,$ to get the corresponding first k eigenvalues.
3) Get factor scores using $\hat{F}^{\lambda} = \left(\hat{\Lambda}' \; \hat{\Lambda} \right)^{-1} \hat{\Lambda}' X'$, where $\hat{\Lambda} = \hat{P} D$, D is the a real, positive, diagonal $k \times k$ matrix which are the first k singular values of X.

4) Use KNN or LDA to do classification on the factor scores.

The alternative methods for calculating factor scores are regression, Bartlett, and Anderson-Rubin.

4.2.3 Tuning Parameter Selection

Xin (2009) proposed a method of selecting tuning parameter λ_j, however, that method is not so efficient. Since I would like to do classification on the factor scores, so here we propose a cross-validation method by evaluating the classification error (CE) to choose λ_j and the number of factors k in SFABP, and we consider λ_j the same for all PCs (say λ) for purpose of simplicity. First we randomly split the data into C roughly equal-sized parts (say $C = 5$), $X_1, \dots, X_{(C)}$, whose sizes are $n_1, \dots, n_{(C)}$. For a given $\lambda_j = \lambda$ and a given $c \in \{1, \dots, C\}$, let $\hat{P}^{\lambda}_{(-c)}$, $\hat{\Lambda}^{\lambda}_{(-c)}$ and$\hat{F}^{\lambda}_{(-c)}$ be the estimated PCs from algorithm SFABP, the estimated factor loading and factor score respectively using the data leaving out $X_{(c)}$, and $\hat{\Lambda}^{\lambda}_{(-c)} = \hat{P}^{\lambda}_{(-c)}\hat{D}^{\lambda}_{(-c)}$, $\hat{F}^{\lambda}_{(c)}$ be the estimated factor score using $\hat{F}^{\lambda}_{(c)} = \left(\hat{\Lambda}^{\lambda}_{(-c)}{}' \hat{\Lambda}^{\lambda}_{(-c)}\right)^{-1} \hat{\Lambda}^{\lambda}_{(-c)}{}' X_c{}'$, Let $\widehat{CE}^{\lambda}_{(c)}$ be the estimated classification error by applying any classification method on $\hat{F}^{\lambda}_{(c)}$.

Now we can describe the cross-validation method as follows.

Cross-validation for selecting λ:

1) Select a list of λ.
2) Randomly split the data into $X_1, \dots, X_{(C)}$.
3) Calculate $\hat{F}^{\lambda}_{(c)}$ using $\hat{F}^{\lambda}_{(c)} = \left(\hat{\Lambda}^{\lambda}_{(-c)}{}' \hat{\Lambda}^{\lambda}_{(-c)}\right)^{-1} \hat{\Lambda}^{\lambda}_{(-c)}{}' X_c{}'$.
4) Applying KNN or linear discriminate analysis (LDA) to do classification on $\hat{F}^{\lambda}_{(c)}$, and get the $\widehat{CE}^{\lambda}_{(c)}$.
5) Calculate $CV\ (\lambda) = \frac{1}{C}\sum_{c=1}^{C} \widehat{CE}^{\lambda}_{(c)}$.
6) Obtain $\hat{\lambda} = arg\ min\ CV(\lambda)$.

4.2.4 The Number of Factors Selection

One of the hardest things when conducting FA is to determine how many factors to settle on. There are several more commonly used criteria to help with the selection. They are as follows:

Kaiser's stopping rule: only considers the number of factors with eigenvalues over 1.00 should in the analysis.

Cattell's Scree test (Cattell, 1966): The researcher examines the scree plot which is a graphic visualization of the relationship between eigenvalues and number of factors and decides where the line stops descending precipitously and levels out (Bryant and Yarnold, 1995, pp. 103-104).

Number of non-trivial factors: Trivial factors are usually defined as those that do not have two or three variables loading above the cut-point (often .30) on them. It is worth considering that triviality is a matter of degrees and may be in the eye of the beholder.

A priori criterion: If a researcher has created a set of test or questionnaire items to contain a specific number of subtests or scales, it would make sense to set that same number of factors in the factor analysis of those items. These are known as a priori criteria for determining the number of factors.

Percent of cumulative variance: An approach that is closely related to Kaiser's stopping rule and the scree plot, however, percents of cumulative variance are harder to interpret than the other two. As a result interpreting the percentages of cumulative variance is more a matter of keeping an eye on the amount of cumulative variance being accounted for by various other stopping rules.

To choose the number of factors in SFABP, we also use the CV method based on CE. For any given $k \in \{1, \dots, K_{max}\}$, where K_{max} is the largest number of factors to be considered, execute SFABP with λ selected via the proposed CV method, compute the CE, and select $\hat{k}$ as the minimize of these average computed CE values.

One thing should be pointed out here is that we can choose the tuning parameter λ and the number of factor k coincidently, or choose it separately if the other one is known.

4.3 Numerical Results

The proposed SFA modeling strategy is investigated through the simulations and real data analysis.

4.3.1 Simulation Studies

In the simulation study, the following model is used.

Model A:

$$\Lambda = \begin{pmatrix} 0.9 * 1_q & 0_q & 0_q \\ 0_q & 0.8 * 1_q & 0_q \\ 0_q & 0_q & 0.7 * 1_q \end{pmatrix},$$

$$diag(\varphi) = (0.19 * 1_q^{'},\ 0.36 * 1_q^{'},\ 0.51 * 1_q^{'})' ,$$

Where 1_q is a q -dimensional vector with each element being 1, and 0_q is a q-dimensional zero vector. For each model, the data sets were generated with $X = N(0, \Lambda\Lambda^T + \Psi)$.

Suppose for each model, 50 data sets were generated and the number of samples was $n = 100, 200, 300, 400, and\ 500$. Table 4.1 show the mean squared error of Λ and Ψ defined by

$$MSE_\Lambda = \frac{1}{50}\sum_{i=1}^{50}\left\|\Lambda - \hat{\Lambda}^{(i)}\right\|^2 \text{ and } MSE_\Psi = \frac{1}{50}\sum_{i=1}^{50}\left\|\Psi - \widehat{\Psi}^{(i)}\right\|^2$$

Where $\hat{\Lambda}^{(i)}$ and $\widehat{\Psi}^{(i)}$ are the estimates for Λ and Ψ respectively using ith dataset.

We compare the performance of four factor analysis procedures: (1) principal factor analysis (PFA); (2) maximum likelihood expectation (MLE) factor analysis; (3) SFABP setting $\lambda = 0$ (FABP); and (4) SFABP setting $\lambda \neq 0$ (SFABP). We set $q = 10$, 20, and $n = 100, 200, 300, 400, and\ 500$. Table 4.1 shows the results.

Table 4.1 Mean squared error of the factor loadings and uniquenesses, and their standard deviation(in parenthesis).

n	Method	$q = 10$	$q = 20$
		Lambda(SD) Phi(SD)	Lambda(SD) Phi(SD)
100	PFA	7.95(3.80) 0.14(0.05)	23.16(6.28) 0.20(0.03)
	MLE	7.50(3.78) 0.12(0.02)	23.60(7.74) 0.20(0.04)
	FABP	7.95(3.81) 0.14(0.05)	23.16(6.28) 0.20(0.03)
	SFABP	1.51(0.42) 0.24(0.07)	5.64(1.36) 0.49(0.14)
200	PFA	3.74(1.09) 0.08(0.03)	10.27(1.16) 0.11(0.03)
	MLE	3.09(0.87) 0.06(0.02)	9.80(1.21) 0.10(0.02)
	FABP	3.74(1.09) 0.08(0.03)	10.27(1.16) 0.11(0.03)
	SFABP	0.79(0.27) 0.13(0.05)	1.88(0.53) 0.14(0.04)
300	PFA	2.49(0.74) 0.07(0.02)	8.79(2.93) 0.07(0.02)
	MLE	2.00(0.70) 0.03(0.01)	8.40(2.82) 0.07(0.02)
	FABP	2.49(0.74) 0.07(0.02)	8.79(2.93) 0.07(0.02)
	SFABP	0.62(0.13) 0.09(0.02)	1.15(0.40) 0.09(0.02)
400	PFA	2.22(0.76) 0.07(0.02)	6.78(1.68) 0.06(0.01)
	MLE	1.76(0.66) 0.03(0.01)	6.41(1.73) 0.05(0.02)
	FABP	2.22(0.76) 0.07(0.02)	6.78(1.68) 0.06(0.01)
	SFABP	0.66(0.19) 0.09(0.02)	0.98(0.26) 0.08(0.01)
500	PFA	1.41(0.22) 0.06(0.01)	4.98 (1.22) 0.06(0.02)
	MLE	1.05(0.36) 0.02(0.01)	4.59(1.17) 0.04(0.01)
	FABP	1.41(0.22) 0.06(0.01)	4.98(1.22) 0.06(0.02)
	SFABP	0.60(0.14) 0.08(0.02)	0.95(0.35) 0.07(0.02)

From table 4.1, we can see that

1) The value of MSE becomes smaller when the sample size n increases, becomes larger when q increases.
2) SFABP performs best compared to the other three methods according to the MSE of Λ, but performs worst according to the MSE of Ψ. The reason might be the large bias introduced by the methodology for estimating λ_i.
3) The FABP and PFA performs exactly same, that also indicate that FABP becomes principal factor analysis when $\lambda = 0$.
4) The MLE performs between FABP and SFABP.

We choose one setting $(n, q) = (300, 2)$ as an illustrative example to show the factor loadings we get using SFABP, and set $= 0.1$, the factor loadings results is shown in Table 4.2, we can see that SFABP can correctly get the sparse solution of factor loadings. The larger λ, the sparser the factor loadings we will get.

Table 4.2 The estimated factor loadings from SFABP when $n = 300, q = 2$, and $\lambda = 0.1$.

Factor1	Factor2	Factor3
0.9562781	0.0000000	0.0000000
0.9562717	0.0000000	0.0000000
0.0000000	-0.9000128	0.0000000
0.0000000	-0.8999815	0.0000000
0.0000000	0.0000000	-0.8643205
0.0000000	0.0000000	-0.8643882

Assume each simulated dataset consists n p-dimensional observations $x_i, i = 1, \dots, n$, which are generated as follows. First, we generate $3q$ variables using the model A, and generated with $X = N(0, \Lambda\Lambda^T + \Psi)$. For the remaining $nvar$ variables where $nvar = p - 3q$, similar to SODC simulation study, suppose C_i's are uniformly sampled from {1,2,3}, indicating the cluster memberships of the obser-

vations. Then for each i, the $nvar$ features which accounting for the classification identification are generated from $N(m(C_i), I_{nvar})$, where nvar is even and

$$m(C_i) = \mu\left(-1'_{\frac{nvar}{2}}, 1'_{\frac{nvar}{2}}\right) I(C_i = 1) + \mu 1_{nvar} I(C_i = 2) + \mu\left(1'_{\frac{nvar}{2}}, -1'_{\frac{nvar}{2}}\right) I(C_i = 3),$$

Clearly, the three groups are separated when μ is large, and overlapped when μ is small. For classification purpose, the number of noise variables $3q$ increases, it becomes harder to distinguish the three groups.

We set the testing dataset sample size as 1500, and compare the performance of five classification procedures: (1) k-means using all features (ALL); (2) k-means using the factors from MLE factor analysis with no rotation (MLENO); (3) k-means using the factors from MLE factor analysis with varimax rotation (MLEVAR); (4) k-means using the factors from SFABP setting $\lambda = 0$ (FABP); and (5) k-means using the factors from SFABP setting $\lambda \neq 0$ (SFABP).

First, set $q = 100$, $nvar = 4$, $n = 120, 150, 180, 360, 480, or\ 900$, and $\mu = 1.5, 1.6, or\ 1.7$. Tuning parameter λ is selected over a range of values $\{0.00001, 0.0001, 0.001, 0.01, 0.05, 0.1, 0.2, 0.3\}$. 5-fold cross-validation is applied to select the number of factors. Each simulation setting is replicated 20 times and the classification error rate is reported in Table 4.3. It shows that SFABP works very well in all settings, FABP, MLENO, MLEVAR, and FABP perform similarly when $n = 360, 480, and\ 900$ where $n > p$, however, MLENO and MLEVAR does not work at all when $n = 120, 150, and\ 180$ where $n < p$. ALL performs worst in all settings, which also indicates the necessity of conducting dimension reduction in classification analysis. As to the two methods KNN or LDA, LDA outperforms over KNN in all settings. Table 4.3 also show that the classification error decrease when u increase or n increase.

Table 4.3 Comparing five classification procedures: (1)using all features (ALL); (2) using factors from MLE factor analysis with no rotation (MLENO); (3) using the factors from MLE factor analysis with varimax rotation (MLEVAR); (4) using the factors from SFABP setting $\lambda = 0$ (FABP); and (5) using the factors from SFABP setting $\lambda \neq 0$ (SFABP).

n	Method	$\mu = 1.5$		$\mu = 1.6$		$\mu = 1.7$	
		KNN(SD)	LDA(SD)	KNN(SD)	LDA(SD)	KNN(SD)	LDA(SD)
120	ALL	0.39(0.01)	0.14(0.01)	0.37(0.01)	0.12(0.01)	0.34(0.02)	0.10(0.01)
	MLENO	n/a	n/a	n/a	n/a	n/a	n/a
	MLEVAR	n/a	n/a	n/a	n/a	n/a	n/a
	FABP	0.25(0.02)	0.15(0.01)	0.23(0.01)	0.13(0.01)	0.20(0.01)	0.11(0.01)
	SFABP	0.23(0.02)	0.11(0.01)	0.21(0.02)	0.08(0.01)	0.20(0.02)	0.07(0.01)
150	ALL	0.39(0.02)	0.14(0.01)	0.36(0.01)	0.12(0.01)	0.34(0.02)	0.10(0.01)
	MLENO	n/a	n/a	n/a	n/a	n/a	n/a
	MLEVAR	n/a	n/a	n/a	n/a	n/a	n/a
	FABP	0.23(0.01)	0.13(0.01)	0.21(0.01)	0.12(0.01)	0.19(0.01)	0.10(0.01)
	SFABP	0.21(0.02)	0.10(0.01)	0.20(0.01)	0.08(0.01)	0.17(0.02)	0.07(0.00)
180	ALL	0.39(0.01)	0.14(0.01)	0.36(0.01)	0.12(0.01)	0.34(0.01)	0.10(0.01)
	MLENO	n/a	n/a	n/a	n/a	n/a	n/a
	MLEVAR	n/a	n/a	n/a	n/a	n/a	n/a
	FABP	0.22(0.02)	0.13(0.01)	0.20(0.02)	0.11(0.01)	0.16(0.01)	0.09(0.01)
	SFABP	0.20(0.01)	0.10(0.01)	0.18(0.02)	0.08(0.01)	0.16(0.02)	0.07(0.01)
360	ALL	0.39(0.01)	0.14(0.01)	0.36(0.01)	0.12(0.01)	0.33(0.01)	0.10(0.00)
	MLENO	0.18(0.01)	0.11(0.01)	0.16(0.01)	0.09(0.01)	0.14(0.01)	0.08(0.00)
	MLEVAR	0.18(0.01)	0.11(0.01)	0.16(0.01)	0.09(0.01)	0.14(0.01)	0.08(0.00)
	FABP	0.19(0.02)	0.11(0.01)	0.17(0.01)	0.09(0.01)	0.15(0.01)	0.08(0.00)
	SFABP	0.16(0.02)	0.09(0.01)	0.14(0.01)	0.08(0.01)	0.12(0.01)	0.06(0.00)
480	ALL	0.38(0.02)	0.14(0.01)	0.37(0.01)	0.12(0.01)	0.34(0.01)	0.10(0.01)
	MLENO	0.18(0.01)	0.10(0.01)	0.15(0.01)	0.09(0.01)	0.14(0.01)	0.07(0.01)
	MLEVAR	0.18(0.01)	0.10(0.01)	0.15(0.01)	0.09(0.01)	0.14(0.01)	0.07(0.01)
	FABP	0.18(0.01)	0.10(0.01)	0.16(0.01)	0.09(0.01)	0.14(0.01)	0.07(0.01)
	SFABP	0.15(0.01)	0.09(0.01)	0.13(0.01)	0.08(0.01)	0.11(0.01)	0.06(0.01)
900	ALL	0.39(0.01)	0.14(0.01)	0.36(0.01)	0.12(0.01)	0.33(0.01)	0.10(0.01)
	MLENO	0.17(0.01)	0.10(0.01)	0.15(0.01)	0.08(0.01)	0.12(0.02)	0.07(0.01)
	MLEVAR	0.17(0.01)	0.10(0.01)	0.15(0.01)	0.08(0.01)	0.12(0.02)	0.07(0.01)
	FABP	0.17(0.01)	0.10(0.00)	0.15(0.01)	0.08(0.01)	0.13(0.02)	0.07(0.01)
	SFABP	0.14(0.01)	0.09(0.01)	0.12(0.01)	0.07(0.01)	0.10(0.01)	0.06(0.00)

First, set $q = 10, 100, 200, or\ 300$, $nvar = 4, 6, 8, or\ 10$, $n = 150$, and $\mu = 1.5$, the other settings are the same as in previous paragraph for getting Table 4.3. Table 4.4 is a summarization of the classification error rate, it also show that the classification error increases when q increases and decreases when $nvar$ increases. It also shows that SFABP performs better in high dimensional data setting.

Table 4.4 Comparing five classification procedures.

var	Method	$q = 10$		$q = 100$		$q = 200$		$q = 300$	
		KNN(SD)	LDA(SD)	KNN(SD)	LDA(SD)	KNN(SD)	LDA(SD)	LDA(SD)	LDA(SD)
4	ALL	0.21(0.01)	0.09(0.01)	0.39(0.02)	0.14(0.01)	0.45(0.02)	0.20(0.01)	0.48(0.01)	0.28(0.01)
	MLENO	0.16(0.01)	0.09(0.01)	n/a	n/a	n/a	n/a	n/a	n/a
	MLEVAR	0.16(0.01)	0.09(0.01)	n/a	n/a	n/a	n/a	n/a	n/a
	FABP	0.16(0.01)	0.09(0.01)	0.23(0.01)	0.13(0.01)	0.29(0.01)	0.18(0.01)	0.35(0.02)	0.22(0.01)
	SFABP	0.19(0.01)	0.10(0.01)	0.21(0.02)	0.10(0.01)	0.22(0.01)	0.11(0.01)	0.26(0.02)	0.12(0.01)
6	ALL	0.12(0.01)	0.05(0.01)	0.30(0.02)	0.08(0.01)	0.38(0.01)	0.13(0.01)	0.43(0.02)	0.21(0.02)
	MLENO	0.09(0.01)	0.05(0.01)	n/a	n/a	n/a	n/a	n/a	n/a
	MLEVAR	0.09(0.01)	0.05(0.01)	n/a	n/a	n/a	n/a	n/a	n/a
	FABP	0.09(0.01)	0.05(0.01)	0.14(0.01)	0.07(0.01)	0.17(0.02)	0.09(0.01)	0.22(0.01)	0.11(0.01)
	SFABP	0.13(0.01)	0.05(0.01)	0.15(0.01)	0.05(0.01)	0.15(0.02)	0.05(0.01)	0.17(0.02)	0.06(0.01)
8	ALL	0.08(0.01)	0.02(0.01)	0.24(0.01)	0.04(0.00)	0.32(0.01)	0.08(0.01)	0.37(0.02)	0.15(0.01)
	MLENO	0.06(0.01)	0.02(0.00)	n/a	n/a	n/a	n/a	n/a	n/a
	MLEVAR	0.06(0.01)	0.02(0.00)	n/a	n/a	n/a	n/a	n/a	n/a
	FABP	0.05(0.01)	0.02(0.00)	0.08(0.01)	0.03(0.00)	0.11(0.00)	0.05(0.00)	0.13(0.02)	0.06(0.00)
	SFABP	0.07(0.02)	0.02(0.01)	0.10(0.01)	0.02(0.00)	0.11(0.01)	0.03(0.00)	0.12(0.03)	0.03(0.01)
10	ALL	0.05(0.01)	0.01(0.00)	0.19(0.01)	0.03(0.00)	0.27(0.01)	0.05(0.01)	0.33(0.01)	0.11(0.01)
	MLENO	0.04(0.01)	0.01(0.00)	n/a	n/a	n/a	n/a	n/a	n/a
	MLEVAR	0.04(0.01)	0.01(0.00)	n/a	n/a	n/a	n/a	n/a	n/a
	FABP	0.04(0.00)	0.01(0.00)	0.05(0.01)	0.02(0.00)	0.06(0.01)	0.02(0.00)	0.09(0.01)	0.03(0.01)
	SFABP	0.04(0.01)	0.02(0.00)	0.08(0.01)	0.01(0.00)	0.08(0.02)	0.01(0.00)	0.08(0.01)	0.01(0.00)

To illustrate the selection processes of the number of factors, we randomly select one replication from the setting where $(n, q, u, nvar) = (1200, 10,3,4)$, and set $\lambda = 0.1$, 10-fold cross-validation and display the estimated factors in figure 4.1.

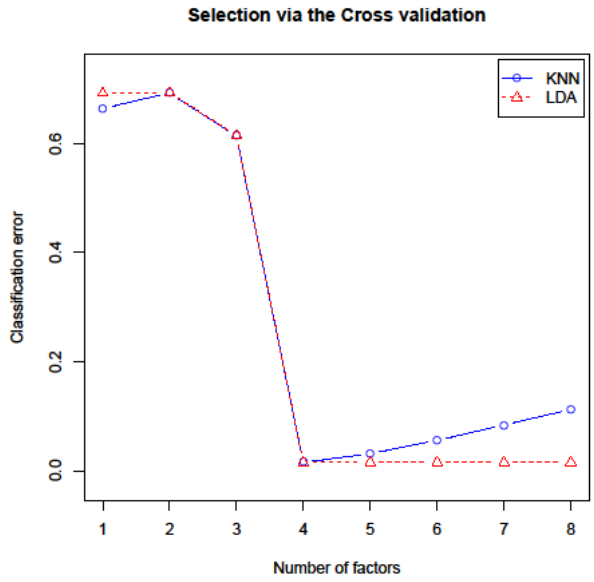

Figure 4.1 KNN classification and LDA classification are applied in Step 4 of SFABP, respectively. The cross-validation selection correctly select the number of factors as 4.

We can see that both the KNN and LDA can be used in step 4 and both can correctly select the number of factors as 4.

Now we still use setting $(n, q, u, nvar) = (1200, 10, 3, 4)$, but now λ is chosen from the range $\lambda \in \{(0.00001, 0.0001, 0.001, 0.01, 0.05, 0.1, 0.2, 0.3\}$. Table 4.5 shows the results when selecting λ and k coincidently. The minimal CE can be found when $k = 4$ and $\lambda = 0.1$.

Table 4.5 Selecting k and λ coincidently.

k	$\lambda = 0.00001$	$\lambda = 0.0001$	$\lambda = 0.001$	$\lambda = 0.01$	$\lambda = 0.05$	$\lambda = 0.1$	$\lambda = 0.2$	$\lambda = 0.3$
1	0.68490	0.68020	0.67790	0.66550	0.66890	0.67370	0.66940	0.66190
2	0.66870	0.66680	0.66880	0.68950	0.68610	0.69160	0.55115	0.28100
3	0.67360	0.67970	0.68420	0.68120	0.68040	0.68000	0.28200	0.01125
4	0.01775	0.01775	0.01725	0.01700	0.01700	0.01675	0.01750	0.01725
5	0.03500	0.03575	0.03250	0.03830	0.03225	0.02925	0.03250	0.03350
6	0.04950	0.04925	0.05000	0.06705	0.06050	0.05975	0.05875	0.06050
7	0.07905	0.08180	0.08030	0.08180	0.08760	0.08100	0.08480	0.08590
8	0.10530	0.10265	0.10365	0.13450	0.11860	0.11625	0.11840	0.11445

4.4 Discussion

We compared the PFA, MLE, FABP, SFABP methods using simulated studies, and get the conclusion that as the number of samples n increases, the value of the MSE becomes smaller, and the true model tends to be selected in all models, and SFABP has the smallest MSE for Λ. We also compare the classification performance of five methods (ALL, MLENO, MLEVAR, FABP, and SFABP), the simulation results showed that SFABP outperforms the other methods in high dimensional data, specially, MLENO, MLEVAR does not work at all when the number of observations less than the number of variables ($n < p$).

5 DISCUSSION AND FUTURE WORK

In this dissertation, we investigate clustering, classification, and factor analysis methods which can be used in high dimensional dataset analysis.

For clustering, we reinvestigate an existing method, ODC, and advocate it as a dimension reduction tool for cluster analysis. We propose a cross-validation method for selecting the tuning parameter in ODC. We also examine the performance of using existing methods such as the gap statistic and the stability selection to select the number of clusters in ODC. As a dimension reduction tool for cluster analysis, ODC performs much better than PCA, which does not take into account the clustering structure. Furthermore, we propose SODC by adding a group-lasso type of penalty on ODC to conduct cluster analysis and feature selection simultaneously. We propose a kappa method for selecting turning parameter in SODC.

For classification, we studied two existing SPCA methods, and apply them to snp dataset. We do comparison between the results and made a conclusion that SPCABP performs better than L-PCA and AL-PCA methods in dimension reduction ability.

For factor analysis, we propose a novel FA method called SFABP based on the idea of SPCABP. We propose to use cross-validation method to choose tuning parameters. After applying SFABP to simulated dataset and real dataset, we proved that SFABP can perform well for high dimensional dataset compared to the some other FA methods.

For future work, we will focus on developing other methodology to derive eigenvalue thus to reduce the MSE of Ψ in SFABP. We will also validate our method on the real dataset. We might also consider to apply SODC, SPCABP, SFA on a same dataset to compare their performance.

REFERENCES

[1] Altman, N. S. (1992), "An Introduction to Kernel and Nearest-neighbor Nonparametric Regression," The American Statistician, 46 (3), 175–185.

[2] Bartholomew, D.J.; Steele, F.; Galbraith, J.; Moustaki, I. (2008), "Analysis of Multivariate Social Science Data," Statistics in the Social and Behavioral Sciences Series (2nd ed.). Taylor & Francis.

[3] Ben-David, S., Von Luxburg, U., and Pal, D. (2006), "A Sober Look at Clustering Stability,"19th Annual Conference on Learning Theory (COLT 2006), 4005, 5-19.

[4] Ben-Hur, A., Elissee_, A., and Guyon, I. (2002), "A Stability Based Method for Discovering Structure in Clustered Data," Pacific Symposium on Biocomputing, 7, 6-17.

[5] Bouveyron, C., and Brunet, C. (2012), "Simultaneous Model-based Clustering and Visualization in the Fisher Discriminative Subspace, "Statistics and Computing, 22(1), 301-324.

[6] Boyer, C., Merzbach U. (1989), "A History of Mathematics," 2nd ed. New York, John Wiley & Sons.

[7] Breiman, L. (1995), "Better Subset Regression Using the Nonnegative Garotte," Technometrics, 37, 373–384.

[8] Broomhead, D. S., Lowe, D. (1988a). "Multivariable Functional Interpolation and Adaptive Networks," Complex Systems, 2, 321–355.

[9] Broomhead, D. S., and Lowe, D. (1988b), "Radial Basis Functions, Multi-variable Functional Interpolation and Adaptive Networks," Technical report, RSRE, 4148.

[10] Bryant, F. B., and Yarnold, P. R. (1995), "Principal-Components Analysis and Confirmatory Factor Analysis," In L. G. Grimm & P. R. Yarnold (Eds.), Reading and understanding multivariate statistics. Washington, DC: American Psychological Association.

[11] Calinski, R. B., and Harabasz, J. (1974), “A Dendrite Method for Cluster Analysis," Communications in Statistics - Simulation and Computation, 3(1), 1-27.

[12] Cattell, R. B. (1966), “The Scree Test for The Number of Factors," Multivariate Behavioral Research, 1(2), 245-276.

[13] Cauchy A.L. (1829),

“l′equation a l′aide de laquelle on determine les inegalites seculaires des mouvements des planètes Oeuvres Complètes(IIème Série),″ Paris, Blanchard.

[14] Chang, W. (1998), “On Using Principal Components before Separating a Mixture of Two Multivariate Normal Distributions," Applied Statistics, 32(3), 267-275.

[15] Clemmensen, L., Hastie, T., Witten, D. M., and Ersboll, B. (2011), “Sparse Discriminant Analysis," Technometrics, 53(4), 406-413.

[16] Clifton, C. (2010), "Encyclopædia Britannica: Definition of Data Mining," Retrieved 2010-12-09.

[17] Cohen, J. (1960), “A Coefficient of Agreement for Nominal Scales, "Educational and Psychological Measurement, 20(1), 37-46.

[18] Cortes, C., and Vapnik, V. (1995), "Support-Vector Networks,", Machine Learning, 20, 273-297.

[19] "Data Mining Curriculum". ACM SIGKDD. 2006-04-30. Retrieved 2011-10-28.

[20] De la Torre, F., and Kanade, T. (2006), “Discriminative cluster analysis, "In The 23rd International Conference on Machine Learning, 241-248.

[21] Fabrigar, L. R., Wegener, D. T., MacCallum, R. C., and Strahan, E. J. (1999), "Evaluating the Use of Exploratory Factor Analysis in Psychological Research," Psychological Methods, 4(3), 272-299.

[22] Fan, J., and Li, R. (2001), “Variable Selection via Nonconcave Penalized Likelihood and Its Oracle Properties,” Journal of the American Statistical Association, 96, 1348–1360.

[23] Fang, Y., and Wang, J. (2012), “Selection of the Number of Clusters via the Bootstrap Method, "Computational Statistics and Data Analysis, 56(3), 468-477.

[24] Fayyad, U., Piatetsky-Shapiro, G., Smyth, P. (1996), "From Data Mining to Knowledge Discovery in Databases," American Association for Artificial Intelligence, 0738-4602, 37-54.

[25] Fowlkes, E. B., and Mallows, C. L. (1983), “A Method for Comparing Two Hierarchical Clusterings, "Journal of the American Statistical Association, 78(383), 553-584.

[26] Friedman, J. H., and Meulman, J. J. (2004), “Clustering Objects on Subsets of Attributes (with Discussion),"Journal of the Royal Statistical Society, Series B, 66(4), 815-849.

[27] Friedman, J. H., and Tukey, J. W. (1974), “A Projection Pursuit Algorithm for Exploratory Data Analysis, "IEEE Trans. Comput., C-23(9), 881-890.

[28] Gnanadesikan, R. (1997), “Methods for Statistical Data Analysis of Multivariate Observations," 2nd Edition, John Wiley & Sons, Inc., New York.

[29] Gorsuch, R. (1983), “Factor Analysis,” Hillsdale, NJ: Lawrence Erl-baum.

[30] Hartigan, J. A. (1975), “Clustering Algorithms," Wiley, New York.

[31] Hastie, T., Tibshirani, R., and Buja, A. (1994), “Flexible Discriminant Analysis by Optimal Scoring," Journal of the American Statistical Association, 89, 1255-1270.

[32] Hastie, T., Tibshirani, R., Friedman, J. (2009), “The Elements of Statistical Learning: Data Mining, Inference, and Prediction," Second Edition Springer Series in Statistics.

[33] Hoerl, A., and Kennard, R. (1988), “Ridge Regression,” In Encyclopedia of Statistical Sciences, New York: Wiley, 8, 129–136.

[34] Hotelling, H. (1933), “Analysis of a complex of statistical variables into principal components,” Journal of Educational Psychology, 24, 417-441.

[35] Johnson, S. C. (1967), “Hierarchical Clustering Schemes, " Psychometrika, 2, 241-254.

[36] Jones, M. C., and Sibson, R. (1987), “What is Projection Pursuit?," Journal of the Royal Statistical Society, Series A, 150(1), 1-37.

[37] Jordan C.(1874), “**Memóire sur** les forms **bilinéaires** ,”J Math Pure Appl, 19, 35-54.

[38] Jöreskog, K. G. (1969), “A General Approach to Confirmatory Maximum Likelihood Factor Analysis,” Psychometrika, 34(2), 183-202.

[39] Kaufman, L., and Rousseeuw, P. (1990), “Finding Groups in Data: An Introduction to Cluster Analysis," Wiley, New York.

[40] Krzanowski, W. J., and Lai, Y. T. (1988), “A Criterion for Determining the Number of Groups in a Data Set Using Sum-of-Squares Clustering," Biometrics, 44(1), 23-34.

[41] Lange, T., Roth, V., Braun, M., and Buhmann, J. (2004), “Stability-based Validation of Clustering Solutions," Neural Computation, 16(6), 1299-1323.

[42] Lee S., Epstein M.P., Duncan R., Lin X. (2012), “Sparse principal component analysis for identifying ancestry-informative markers in genome-wide association studies,” Genet Epidemiol, 36(4):293-302.

[43] Maccallum, R. C. (1990), "The Need for Alternative Measures of Fit in Covariance Structure Modeling," Multivariate Behavioral Research, 25(2), 157-162.

[44] MacQueen, J. B. (1967), “Some Methods for Classification and Analysis of Multivariate Observations," Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, 1, 281-297.

[45] Maugis, C., Celeux, G., and Martin-Magniette, M. L. (2009), “Variable Selection in Model-based Clustering: A General Variable Role Modeling," Computational Statistics and Data Analysis, 53(11), 3872C3882.

[46] Ng, A., Jordan, M., and Weiss, Y. (2001), “On Spectral Clustering: Analysis and an Algorithm," Advances in Neural Information Processing Systems, 14, 849-856.

[47] Pearson, K. (1901), “On lines and planes of closest fit to systems of points in space,” Philosophical Magazine A, 2(6), 559-572.

[48] Preedy, V. R., and Watson, R. R. (2009), “Handbook of Disease Burdens and Quality of Life Measures,” New York: Springer.

[49] Qi, X., Luo, R. and Zhao, H. (2013), “Sparse principal component analysis by choice of norm,” Journal of Multivariate Analysis, 114, 127-160.

[50] Raftery, A. E., and Dean, N. (2006), “Variable Selection for Model-Based Clustering," Journal of the American Statistical Association, 101(473), 168-178.

[51] Rand, W. M. (1971), “Objective Criteria for the Evaluation of Clustering Methods," Journal of the American Statistical Association (American Statistical Association), 66(336), 846-850.

[52] Schwenker, F., Kestler, H.A., and Palm, G. (2001), "Three Learning Phases for Radial-basis-function Networks," Neural Networks, 14, 439–458.

[53] Shen, X., and Ye, J. (2002), “Adaptive Model Selection,” Journal of the American Statistical Association, 97, 210–221.

[54] Shi, J., and Malik, J. (2000), “Normalized Cuts and Image Segmentation," IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8), 888-905.

[55] Spearman, C. (1904), “General Intelligence, Objectively Determined and Measured,” The American Journal of Psychology, 15, 201–293.

[56] Steinley, D., and Brusco, M. J. (2008), “A New Variable Weighting and Selection Procedure for K-means Cluster Analysis," Multivariate Behavioral Research, 43(1), 77-108.

[57] Stewart, G.W. (1993), “on the early history of the singular value decomposition,” SIAM Review, 35(4), 551-566.

[58] Sugar, C., and James, G. (2003), “Finding the Number of Clusters in a Data Set: an Information Theoretic Approach," Journal of American Statistical Association, 98(463), 750-763.

[59] Suhr, D.D. (2009), "Principal Component Analysis vs. Exploratory Factor Analysis," Statistics and Data Analysis. SUGI 30, Proceedings, 203-30.

[60] Sun, L., Ji, S., and Ye, J. (2008), “A Least Squares Formulation for Canonical Correlation Analysis," In The 25th Intl Conf. Machine Learning, 1024-1031.

[61] Sun, W., Wang, J., and Fang, Y. (2012a), “Regularized k-means Clustering of High dimensional Data and its Asymptotic Consistency, " Electron. J. Statist, 6, 148-167.

[62] Sun, W., Wang, J., and Fang, Y. (2012b), “Consistent Selection of Tuning Parameters via Variable Selection Stability," arXiv:1208.3380v1.

[63] International HapMap Consortium. Comment in Nature. 2005 Oct 27;437(7063):1241-2

[64] Tibshirani, R. (1996), “Regression Shrinkage and Selection via the Lasso,” Journal of the Royal Statistical Society B, 58, 267–288.

[65] Tibshirani, R., Walther, G., and Hastie, T. (2001), “Estimating the Number of Clusters in a Data Set via the Gap Statistic, " Journal of Royal Statistical Society, Series B, 63(2), 411-423.

[66] Tyler, D.E., Critchley, F., D umbgen, L., and Oja, H. (2009), “Invariant Co-ordinate Selection (with discussion)," Journal of the Royal Statistical Society, Series B, 71(3), 549-592.

[67] Wang, J. (2010), "Consistent Selection of the Number of Clusters via Crossvalidation," Biometrika, 97(4), 893-904.

[68] Witten, D. M., and Tibshirani, R. (2010), "A Framework for Feature Selection in Clustering," Journal of the American Statistical Association, 105(490), 713-726.

[69] Yaremko, R. M., Harari, H., Harrison, R. C., and Lynn, E. (1986), "Handbook of research and quantitative methods in psychology," for students and professionals. Hillsdale, NJ: Lawrence Erlbaum Associate.

[70] Yuan, M., and Lin, Y. (2006), "Model Selection and Estimation in Regression with Grouped Variables," Journal of the Royal Statistical Society, Series B, 68(1), 49-67.

[71] Zhang, Z., and Dai, G. (2009), "Optimal Scoring for Unsupervised Learning," Advances in Neural Information Processing Systems 23, 12, 2241-2249.

[72] Zou, H. (2006), "The Adaptive Lasso and Its Oracle Properties," Journal of the American Statistical Association, 101(476), 1418-1429.

[73] Zou, H., and Hastie, T. (2005), "Regularization and Variable Selection Via the Elastic Net," Journal of the Royal Statistical Society B, 67, 301–320.

[74] Zou, H., Hastie, T., and Tibshirani, R. (2006), "Sparse Principal Component Analysis," Journal of Computational and Graphical Statistics, 15(2), 265C286.27.

[75] Unknow author, "Principle Component and Factor Analysis," ref here.

APPENDIX

Appendix A: Proof of Theorem for SODC

A.1 Proof of Theorem 1

We use the method of Lagrange multipliers to find the minimize of (2) with respect to Y given W. Let $X^* = H_n XW$ and $k^* = k - 1$. Consider the objective function,

$$L(Y, X^*, \Lambda, b) = \frac{1}{2} tr(Y'Y) - tr(Y'X^*) + \frac{1}{2} tr\left(X^{*\prime} X^*\right) - \frac{1}{2} tr\left(\Lambda(Y'Y - I_{k^*})\right) - tr(b'Y'1_n),$$

Where Λ is a $k^* \times k^*$ symmetric matrix of Lagrange multipliers and b is a $k^* \times 1$ vector of Lagrange multipliers. Letting the differentiation of L be zero, we have

$$\frac{\partial L}{\partial Y} = Y - X^* - Y\Lambda - 1_n b' = 0$$

Pre-multiplying both sides of the above equation by $1_n'$, we have $b = 0$ and then $Y - X^* - Y\Lambda = 0$. Likewise, pre-multiplying by Y', we have $I_{k^*} - Y'X^* = \Lambda$.

If $n \geq k^*$, denote the SVD of X^* as $X^* = UDV'$, where $U \subset R^{n \times k^*}$, $D \subset R^{k^* \times k^*}$, $V' \subset R^{k^* \times k^*}$. Because $1_n' X^* = 0$, we have $1_n' X^* V D^{-1} = 1_n' U = 0$. Let $\hat{Y} = UV'$, which satisfies $1_n' \hat{Y} = 1_n' UV' = 0$ and $\hat{Y}'\hat{Y} = VU'UV' = I_{k^*}$. Moreover, we have $I_{k^*} - \hat{Y}'X^* = I_{k^*} - VU'UDV' = I_{k^*} - VDV' = \Lambda$. This implies that $\hat{Y}$ is a minimizer of problem (2.2).

If $n < k^*$, denote the SVD of X^* as $X^* = UDV'$, where $U \subset R^{n \times n}$, $D \subset R^{n \times n}$, $V' \subset R^{n \times k^*}$. Using this SVD, we can show the theorem similarly.

A.2 Proof of Theorem 2

Given $\mathbf{Y}$, the sub-gradient equations for (3) of W is

$$-2\widetilde{\mathbf{X}}_j' \left(\mathbf{Y} - \sum_{j=1}^{p} \widetilde{\mathbf{X}}_j w_j \right) + 2\lambda_2 w_j + \lambda_1 \frac{w_j}{\|w_j\|_2} = 0, j = 1, \ldots, p$$

Let the minimize of (2.3) be $\widehat{W} = (\widehat{w}_1, \ldots, \widehat{w}_p)'$. If

$$\left\| \widetilde{\mathbf{X}}_j' (\mathbf{Y} - \sum\nolimits_{l \neq j} \widetilde{\mathbf{X}}_l \widehat{w}_l \right\|_2 < \left\| \frac{\lambda_1}{2} \right\|$$

Then $\widehat{w}_j = 0$. (Therefore, $\widehat{w}_j = 0$ for any j if $\lambda > \lambda_{max} = max_j \left\| \widetilde{\mathbf{X}}_j' \mathbf{Y} \right\|$.) Otherwise,

$$\widehat{w}_{j=}(\widetilde{\mathbf{X}}_{\mathrm{j}}'\widetilde{\mathbf{X}}_{\mathrm{j}} + \lambda_2 + \frac{\lambda_1}{2\|w_j\|_2})^{-1} V_j.$$

Where

$$V_j = \widetilde{\mathbf{X}}_{\mathrm{j}}(\mathbf{Y} - \sum\nolimits_{l \neq \mathrm{j}} \widetilde{\mathbf{X}}_l \, \widehat{w}_l)$$

Note that $\widetilde{\mathbf{X}}_{\mathrm{j}}'\widetilde{\mathbf{X}}_{\mathrm{j}}$ is actually a diagonal matrix where diagonal terms are sample variances of features. If we conduct standardization on the design matrix at the beginning, we have $\widetilde{\mathbf{X}}_{\mathrm{j}}'\widetilde{\mathbf{X}}_{\mathrm{j}} = I_{k-1}$ then the above equation becomes

$$\widehat{w}_j = \frac{2\left\|\widehat{w}_j\right\|_2}{\lambda_1 + 2(1+\lambda_2)\left\|\widehat{w}_j\right\|_2} V_j$$

The Euclidean norm is $\left\|\widehat{w}_j\right\|_2 = \frac{2\|V_j\|_2 - \lambda_1}{2(1+\lambda_2)}$. Plugging this norm to the above formula of $\widehat{w}_j$, we get the formula of $\widehat{w}_j$ stated in the theorem.

Appendix B: Core Code for SODC

```
library(magic) # for adiag function
library(ppls)  # for normalize.vector
library(psych)  # for tr() trace function
library(MASS)  # for mvrnorm  function

#### Function: cluster analysis using spectral cluster
hclust.wrap=function(x,centers) {
  hc <- hclust(dist(x))

  res=cutree(hc, k = centers) #k = 1 is trivial
  return(res)
}

##############################################################################
######################### ODC algorithm ################################
##############################################################################

### Function: Get Y, W, Z as in ODC paper for given dataset, #of clusters k and sigma^2
odc.cv=function(data,k,lambda2){

   x=data.matrix(data)
   n=nrow(x)
   p=ncol(x)

   ####  get centered matrix hn
   n1=as.vector(rep(1,n))
   hn=diag(n)-1/n*n1%*%t(n1)

   #### get matrix Hnx

   hnx=hn %*% x

   #### get Y, W, Z

   if(lambda2==0){

    lambda2=10^(-10)

   }



   tmpmat = ginv((t(x)%*% hnx+lambda2*diag(p)), tol=exp(-25)  )%*%t(x)%*%hn  ####use
ginv, not use solve() in order to prevent the singular error messsage.
   if(all(is.finite(tmpmat)))
   s=hnx %*% tmpmat
   else
   stop("infinite or missing values in return frin ginv fuction")
```

```
    eig=eigen(s, symmetric = TRUE)######use symmetric = TRUE to prevent some error mes-
sage that eigen vector include some complex value

    yhat=eig$vectors[,1:(k-1)]

    what=tmpmat%*%yhat
    Z= hnx %*% what

    return(list(yhat=yhat, what=what,Z=Z,s=s, hnx=hnx))


}


### Function: get optimal sigma^2 (lambda_2 in SODC) and use this optimal sigma^2 get Z, do
kmeans clustering on Z.

odc.clust = function(x,centers,cv.num=5,l2=-1,clus=kmeans,l2.idx=seq(-3, 3, by=6/20)) {

 if(l2 == -1){
   rlt = odc.optimallambda2(x, centers,cv.num,l2.idx)

   rlt.odc = odc.cv(x,centers,rlt$opt.lambda2)
   opt.lambda2 = rlt$opt.lambda2
  }
  else{
     rlt.odc = odc.cv(x,centers,l2)

  }
  res = clus(rlt.odc$Z,centers)
  if(l2==-1)
  return(list(res=res, opt.lambda2=opt.lambda2))
  else
  return(res)

}



### Function: Cross validation on average (objective function) to get optimal sigma^2 (lambda_2
in SODC) in the case of known centers
odc.optimallambda2=function(data, centers,  cv.num=5,lambda2.idx=seq(-3, 3, by=6/20)) {


    x=data.matrix(data)
    n=nrow(x)


    sigma=10^lambda2.idx
    nsigma = length(sigma)
```

```
n1=as.integer(n/cv.num)

cv.r=matrix(NA,3, nsigma)
cv.s=matrix(NA,cv.num, nsigma)
df.s=matrix(NA,cv.num, nsigma)



for (cv.i in 1:cv.num) {

    x.tr1=x[-((cv.i-1)*n1+(1:n1)),]
    x.val=x[(cv.i-1)*n1+(1:n1),]
    ####
    for (nsigma in 1:nsigma) {


        tr1.cv=odc.cv(x.tr1,centers,sigma[nsigma])

        #### get # of rows and columns of validation samples
        nval=nrow(x.val)
        pval=ncol(x.val)

        ####  get centered matrix hn.val for validation samples
        n1val=as.vector(rep(1,nval))
        hn.val=diag(nval)-1/nval*n1val%*%t(n1val)

        #### get matrix Hnx.val for validation samples

        hnx.val=hn.val %*% x.val


        #### get prediction of Y of validation samples from estimated W of training samples
        s <- svd(hnx.val %*% tr1.cv$what)
        yhat.val=s$u %*% t(s$v)

        #### objective function Euclidean norm  ||Y-HnXW||
        obj=yhat.val-hnx.val %*% tr1.cv$what
        cv.s[cv.i, nsigma]=tr(t(obj) %*% obj)
        #cv.s[cv.i, nsigma]=tr(t(yhat.val-hnx.val %*% tr1.cv$what) %*% (yhat.val-hnx.val %*%
tr1.cv$what))


        #### get df of s=hnX(x^T+sigma^2Ip)^(-1)  please refer to ridge regression of elemen-
tary statistics of data mining
        tmpmat = ginv((t(x.val)%*% hnx.val+sigma[nsigma]*diag(pval)), tol=exp(-
25)  )%*%t(x.val)%*%hn.val
        s=hnx.val %*% tmpmat
        df=tr(s)
        df.s[cv.i, nsigma]=df
    }
```

```
    }

    sigma.mean = apply(cv.s, 2, mean,na.rm=TRUE)
    sigma.sd = apply(cv.s, 2, sd,na.rm=TRUE)

    df.mean = apply(df.s, 2, mean,na.rm=TRUE)

    cv.r[1,] = sigma.mean
    cv.r[2,] = sigma.sd
    cv.r[3,] = df.mean


  y.hl=min(cv.r[1,],na.rm =TRUE)+cv.r[2,which(cv.r[1,] == min(cv.r[1,],na.rm =TRUE), arr.ind
= TRUE)]

    meanord=order(cv.r[1,])

    tmp=which(cv.r[1,] < y.hl)

    opt.lambda2=sigma[tmp[length(tmp)]]


    return(list(cv.r=cv.r, sigma=sigma,opt.lambda2=opt.lambda2))
  }



  ##########################################################################
  ################################### SODC algorithm ########################
  ##########################################################################

  #### Function: get matrix Hnx, B, and initial value of W
  get.hnx.B.initialW=function(data,k,lambda2){     ###### initially use this to do simulation, but it
cost time, so change to the following version

    x=data.matrix(data)
    p=ncol(x)

    odcrlt = odc.cv(data,k,lambda2)

    #### extend orthonormalized matrix HnX to a nx(c-1) by px(c-1) block diagonal matrix
    #### normalize a vector   normalize.vector {ppls}

      hnx.new=NULL
      for(j in 1:p)
      hnx.new=cbind(hnx.new,normalize.vector(odcrlt$hnx[,j]))

    tmp = hnx.new

    if(k>2){
```

```
            for(i in 1:(k-2)){

             tmp=adiag(tmp,hnx.new)

            }
            B=tmp
          } else   B=hnx.new



          y.oldvec = as.vector(odcrlt$yhat)
          w.oldvec = as.vector(t(odcrlt$what))

          return(list(y.initial=y.oldvec, w.initial=w.oldvec, what=odcrlt$what,yhat=odcrlt$yhat,B=B,
hnx=odcrlt$hnx))


        }



        ####Function: get correspoinding predictors of Wj group
        get.bj=function(c, B, p,j){

            Bj=NULL
              for(k in 1:c-1){
               if(c>2 & j<=p*(c-1)){

                Bj=cbind(Bj,B[,j])
                j=j+p

               }
               if(c==2){
                 Bj=B[,j]
               }

             }

          return(Bj)
        }



        ####Function: get correspoinding predictors matrix in the order of wj group   #### newly
changed version of get.B.inorder function
        get.B.inorder=function(c, B, p){

            B.inorder=NULL
            s=NULL
            for(j in 1: p){
```

```
        s = c(s, seq(j,,p,c-1))
        }

       B.inorder=B[,s]

    return(B.inorder)
}

####Function: get correspoinding parameters matrix after removing the jth group
get.w.remove.j=function(w,j){
trans.w=t(w)
trans.w[,j]=0
return(as.vector(trans.w))

}


#### Function:SODC algorithm
my.lasso.classify=function(data,c,lambda1,lambda2, tol=10^(-10),iter.max = 50){


   p = ncol(data.matrix(data))
   w.init=get.hnx.B.initialW( data, c,lambda2)
   B.inorder=get.B.inorder(c, w.init$B, p)

   w.old = matrix(10000, nrow(w.init$what),ncol(w.init$what))
   count <- 0


   while(min(t(w.init$what-w.old)%*%(w.init$what-w.old), t(w.init$what+w.old) %*%
(w.init$what+w.old)) > tol){

     count <- count + 1
     if(count > iter.max) break;
     for(j in 1: p){

       Bj=get.bj(c, w.init$B, p,j)
       w.remove.j=get.w.remove.j(w.init$what,j)
       res=w.init$y.initial-B.inorder%*%(w.remove.j)

       vlnorm=sqrt(sum((t(Bj)%*%(res))^2))
       wjnorm=sqrt(sum((w.init$what[j,])^2))


      if(wjnorm!=0){


        if (vlnorm <= lambda1/2){

          wjhat= 0
        }else
```

```
            {
              wjhat = ((1-lambda1/2*vlnorm)/(1+lambda2)) * (t(Bj) %*% res)

            }


            w.old = w.init$what
            w.init$what[j,]=wjhat

            s <- svd(w.init$hnx %*% w.init$what)
            yhat.new=s$u %*% t(s$v)
            w.init$y.initial = as.vector(yhat.new)
            w.init$w.initial = as.vector(w.init$what)

          }


        }



      }
      nvarselected=0
      nvarselectedset=NULL
      for(i in 1: p){
        wjnormtemp=sqrt(sum((w.init$what[i,])^2))

        if(wjnormtemp!=0){
          nvarselected=nvarselected+1

          nvarselectedset=c(nvarselectedset,i)
          #if(nvarselected<=1)
          #stop("Please use a smaller lambda1"))
        }
      }
      Z= w.init$hnx %*% w.init$what



       return(list(Z=Z, varset=nvarselectedset, what=w.init$what, nvarselected=nvarselected))


    }

    ##################################################
    #### Function: cluster analysis using odc.lasso.coordinate clustering. If l2=-1, select the optimal
lambda2 and corresponding winit
    sodc.clust = function(x,centers,l1=-1,l2=-1,cv.num=5,clus=kmeans,boot.num=20,l2.idx=seq(-3, 3,
by=6/20),l1.idx=seq(-3, 3, by=6/20)) {
      if(l2==-1){
```

```
    rlt=odc.optimallambda2(x, centers,cv.num,l2.idx)

    l2=rlt$opt.lambda2


  }
  if(l1==-1){

   rlt = sodc.optimallambda1.boot.all(x, centers, boot.num,l1.idx)
   l1=rlt$opt.lambda1
  }


  rlt.sodc = my.lasso.classify( x,centers,l1,l2)

  cl=NULL
  clvar=NULL
  if(length(unique(rlt.sodc$Z))>=centers ){

    cl=clus(rlt.sodc$Z,centers)

    clvar=clus(x[,rlt.sodc$varset],centers)

  }

  return(list(cl=cl, clvar=clvar, opt.lambda1=l1, opt.lambda2=l2))
}


#####################################################
#### SODC algorithm tuning parameter selection####
#####################################################

#### Function: SODC selecting optimal lambda1 using clust.kappa method (SODC)  use dependent generated bootstrap pairs, all lambda1 values
sodc.optimallambda1.boot.all=function(data,centers,boot.num=20,l1.idx=seq(-3,3,by=6/20)){
     lambda1=10^l1.idx
     n.lambda1 = length(lambda1)
      x=data.matrix(data)
   n=nrow(x)
   ncol=ncol(x)


   n1=as.integer(n*(1/2))
   boot.r=matrix(NA, 1, n.lambda1)
   boot.s=matrix(NA,boot.num, n.lambda1)
     for (boot.i in 1:boot.num) {

        x.perm=x[sample(1:n,n,replace=F),]
        x.boot1=x.perm[1:n1,]
        x.boot2=x.perm[(n1+1):n,]
```

```
        rlt1=odc.optimallambda2(x.boot1, centers,cv.num=5)
        rlt2=odc.optimallambda2(x.boot2, centers,cv.num=5)


       for (nlambda1 in 1:n.lambda1) {



          boot1.clus=my.lasso.classify(x.boot1,centers,lambda1[nlambda1],rlt1$opt.lambda2)
          boot2.clus=my.lasso.classify(x.boot2,centers,lambda1[nlambda1],rlt2$opt.lambda2)

          boot.s[boot.i, nlambda1]=clust.kappa(boot1.clus$varset,boot2.clus$varset,ncol)

        }
       print (boot.i)

     }

   lambda1.mean = apply(boot.s, 2, mean,na.rm=TRUE)


   boot.r[1, ] = lambda1.mean
   if(all(boot.r[1,]==-1))
   print("for this dataset, SODC always choose the exactly the true model.")
   opt.lambda1=lambda1[which(boot.r[1,] ==max(boot.r[1,]))][1]
  return(list(boot.r=boot.r, boot.s=boot.s, lambda1=lambda1,opt.lambda1=opt.lambda1))
}




#### Function: output the kappa score between two clustering assignments: clus1 and clus2
#### p is the total number of variables of a dataset on which do clustering
clust.kappa=function(clus1,clus2,p) {

  com=seq(1,p,by=1)
  n11=length(intersect(clus1,clus2))
  s1com=setdiff(com,clus1)
  s2com=setdiff(com,clus2)
  n12=length(intersect(clus1,s2com))
  n21=length(intersect(s1com,clus2))
  n22=length(intersect(s1com,s2com))

  pr_a=(n11+n22)/p

  pr_e=(n11+n12)*(n11+n21)/p^2+(n12+n22)*(n21+n22)/p^2


  if(length(clus1)==0||length(clus2)==0||length(clus1)+length(clus2)==2*p) ka=-1 else
  ka=(pr_a-pr_e)/(1-pr_e)
```

```
  return(ka)
}


#### Function: generate multimal norm data set with 3 clusters
#### u: contral the overlap degree among clusters. The cluster centers are (-u,u),(u,u),(u,-u) re-
spectively,
#### nvar: number of informative variable,should be even number; nobs: number of observations;
#### p: total number of variables. p-nvar: number of noise variable
my.normdata.gen=function(u,p,nobs,nvar){

  vec25=as.vector(rep(1,nvar/2))
  vec50=as.vector(rep(1,nvar))
  idenmat50=diag(nvar)



  mean.y1=c(-u*vec25,u*vec25)
  mean.y2=c(u*vec50)
  mean.y3=c(u*vec25,-u*vec25)
  #mean.y4=c(-u*vec50)



  x=matrix(0, nrow=nobs, ncol=p)


  y = sample(c(1,2,3), nobs, replace = TRUE)

  for(i in 1:nobs){


      if(y[i]== 1)
      x[i,1:nvar]= mvrnorm(n=1, mean.y1, idenmat50)
      else if(y[i]== 2)
      x[i,1:nvar]= mvrnorm(n=1, mean.y2, idenmat50)
      else if(y[i]== 3)
      x[i,1:nvar]= mvrnorm(n=1, mean.y3, idenmat50)

      if(p>nvar)
        x[i,(nvar+1):p] = rnorm(n=(p-nvar), mean = 0, sd = 1)


   }
   return (list(x=x,y=y))


}
```

Appendix C: Core Code for SPCABP

```
constMin1 <- function(a, mu)
# max a'x such that ||x||_mu \e 1.
{
nvar <- length(a)
x <- rep(1,nvar)
aa <- abs(a)
aa.sort <- sort(aa, decreasing=T, index.return=T,method="quick")
aas <- aa.sort$x
aas.sum <- sum(aas)
aas.cum <- cumsum(aas)

flag <- ( (aas[-1] <= mu*aas.cum[-nvar]/(1+(1:(nvar-1)-1)*mu)) & (mu*aas.cum[-
nvar]/(1+(1:(nvar-1)-1)*mu) < aas[-nvar]) )
if(sum(flag)>0){
mc <- which(flag>0)[1]
}else{
mc <- nvar
}
tmp <- mu * aas.cum[mc] / (1+(mc-1)*mu)
x <- c(aas[1:mc] - tmp, rep(0, nvar-mc))

b <- sort(aa.sort$ix, index.return=T,method="quick")

x <- x[b$ix]

norm= sqrt(mu*(sum(abs(x)))^2+(1-mu)*sum(x^2))
I=aa.sort$ix[1:mc]
x[I] <- (sign(a[I]) * x[I]) / norm

return(list(x=x, norm.lambda=norm, m=mc))
}




constOpt2 <- function(a, Psi, mu, tol,tol.1)
{nvar=length(a)
 if(is.null(Psi))
  {ret <- constMin1(a,mu)
   v<- ret$x
   return(v)
  }else
   {Psi=as.matrix(Psi)
    dim.old <- dim(Psi)[2]
    t=rep(0,dim.old)
    ret <- constMin1(a,mu)
    x<- ret$x # correspond to x(t) in proof of theorem 2.6
    nu <- ret$norm / 2
    m <- ret$m
```

```
h <- as.vector((2*nu/(1-mu))*t(x)%*%Psi)
H <- sum(h^2)
alpha <- mu / (1+(m-1)*mu)
count1 <- 0

while(H>tol){
 #print(c("count1=",count1,H))
 count1 <- count1+1
 if(count1 > 400){
    # print("big count1");
    #print(c(count1, H));
    break;
    }
 I=which(x!=0)
 signx <- sign(x[I])
 if(length(I)<(nvar/2))
  { NNPsi <- Psi[I,]
    K <- NNPsi - alpha * (signx %*% (t(signx) %*% NNPsi))
    A=(t(K)%*%NNPsi/(1-mu))
  }else{
     NNPsi <- Psi[-I,]
     if((nvar-length(I))==1){NNPsi=(t(as.matrix(NNPsi)))}
     temp=t(Psi[I,])%*%signx
     A=((diag(dim.old)-t(NNPsi)%*%NNPsi-alpha*temp%*%t(temp))/(1-mu))
    }


 if(H>1e-5)
  {deriv.1 <- 2*A%*%h
   deriv.2 <- 2*A%*%t(A)
   eig.hess <- (eigen(deriv.2))$values
   positive.thresh <- max(c(0, abs(eig.hess[eig.hess<1e-6]))) # value 0 is used to avoid NULL
in the latter part
   positive.thresh=positive.thresh +1e-7
   #positive.thresh= positive.thresh+min(positive.thresh, 1e-3)
   if(H<1e-7){positive.thresh=H}
    # print(c("H",H))
   Hess <- deriv.2 + positive.thresh * diag(dim.old)
   t.inc <- - solve(Hess, deriv.1)
  }else{
   fit=svd(A)
   I=which(fit$d>tol.1)
   if(length(I)==1){t.inc=-fit$v[,I]*(1/(fit$d[I]))*(fit$u[,I])*h}else
     {t.inc=-fit$v[,I]%*%diag(1/(fit$d[I]))%*%t(fit$u[,I])%*%h}
   }


 t.new <- as.vector(t + t.inc)
 # print(c("H",H))
 temp=as.vector(a+Psi %*% t.new)
 ret <- constMin1(temp,mu)
```

```
x.new<- ret$x # correspond to x(t) in proof of theorem 2.6
nu.new <- ret$norm / 2
m.new <- ret$m
alpha.new <- mu / (1+(m.new-1)*mu)
h.new <- as.vector((2*nu.new/(1-mu))*t(x.new)%*%Psi)
H.new <- sum(h.new^2)
count2=0
while((H.new>tol)&(H.new>H+2*(1e-4)*t(t.inc)%*%(A%*%h))){
  count2 <- count2+1
  #print(c(count2, H,t.inc))
  if(count2 > 4){
    #print("big count2.")
    #print(count2)
    temp=rnorm(length(t),0,1)
    t.new=t+0.1*temp/sqrt(sum(temp^2))
    break;
  }

  t.inc <- 0.3 * t.inc
  t.new <- as.vector(t + t.inc)
  temp=as.vector(a+Psi %*% t.new)
  ret <- constMin1(temp,mu)
  x.new<- ret$x # correspond to x(t) in proof of theorem 2.6
  nu.new <- ret$norm / 2
  m.new <- ret$m
  alpha.new <- mu / (1+(m.new-1)*mu)
  h.new <- as.vector((2*nu.new/(1-mu))*t(x.new)%*%Psi)
  H.new <- sum(h.new^2)
 }
t <- t.new
h=h.new
nu=nu.new
H <- H.new
x <- x.new
m=m.new
alpha <- alpha.new
} # end of while(H>tol)
#print(c("count1=",count1,H))
return(x)
}
}
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
spca.BP=function(y, ncomponent,lambda,nrepeat=3,covariance=F, corr=F,orthogonal=F,tol=10^{-12},tol.1=10^{-8})
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# y is either a data matrix with rows being samples and columns being variables or a covariance matrix;
```

ncomponent is the number of PCs you want to extract;
lambda is the sparse tuning parameter;
nrepeat is the number of repeats for each components in order to get the global maximum of the objective function.
covariance indicates whether y is a covariance matrix;
corr indicates whether the correlation matrix is used instead of the covariance matrix;
orthogonal indicates whether the components are orthogonal or uncorrelated;
#%%

```
{mu=lambda
 PCs=NULL
 Psi=NULL
 if(sum(is.na(y))!=0)
   {print("There are missing values!")
   return(NULL)}else
  {
 if(covariance)
  {nvar=dim(y)[1]
  if(corr)
   {mtx=cov2cor(y)
   }else
    {mtx=y}
  for(ncomp in 1:ncomponent)
   {
    qq=cc=rep(0,nrepeat)
    mqq=0
    for(j in 1:nrepeat)
     {
      v <- rnorm(nvar,0,30)
      v.old <- rnorm(nvar,0,5)
      count <- 0
      value=t(v)%*%(mtx%*%v)
      value.old=0
      while((value>value.old)&(min(t(v-v.old)%*%(v-v.old), t(v+v.old) %*% (v+v.old)) >
sum(v^2)*tol.1))
       {
        count <- count + 1
        if(count > 60)
        {#print(c("count",count))
         break}
        v.old <- v
        a <- mtx%*%v
        if(!is.null(Psi)){a <- a - Psi %*% (t(Psi) %*% a)} # get orthogonal of a
        if (sum(abs(a))==0) stop
         v <- constOpt2(a/sqrt(sum(a^2)), Psi, mu, tol,tol.1)
        value=t(v)%*%(mtx%*%v)
        #print(c("value=",value))
        #if(!is.null(Psi)){print(t(Psi)%*%v)}
       }
```

```
        qq[j]=t(v)%*%(mtx%*%v)
        #print(c("qq=",qq[j],1))

        if((mqq<qq[j]))
          {vc=v
           mqq=qq[j]
          }
        }
       print(c("The component",ncomp))
       PCs=cbind(PCs,vc/sqrt(sum(vc^2)))
       if(orthogonal)
        {Psi <-cbind(Psi, vc)}else
         {
          u <-mtx%*%vc
          if(!is.null(Psi)){u=u-Psi%*%(t(Psi)%*%u)}
          Psi=cbind(Psi, u/sqrt(sum(u^2)))
         }

     }
   }else
   {x=y
    nsample=dim(x)[1]
    nvar =dim(x)[2]
    mux<- drop(rep(1,dim(x)[1]) %*% x)/dim(x)[1]
    temp=scale(x,mux,scale =F)
    if(corr)
     {
      normx <- sqrt(drop(rep(1,dim(x)[1]) %*% (temp^2)))
      z.x=scale(x,mux,normx)
     }else
       {z.x=scale(x,mux,rep(max(abs(x)),dim(x)[2]))}
    for(ncomp in 1:ncomponent)
     {
      qq=cc=rep(0,nrepeat)
      mqq=0
      for(j in 1:nrepeat)
       {
        v <- rnorm(nvar,0,30)
        v.old <- rnorm(nvar,0,5)
        count <- 0
        value=sum((z.x%*%v)^2)
        value.old=0
        while((value>value.old)&(min(t(v-v.old)%*%(v-v.old), t(v+v.old) %*% (v+v.old)) >
sum(v^2)*tol.1))
        {
         count <- count + 1
          if(count > 60)
            {#print(c("count",count))
             break}
         v.old <- v
         temp=(z.x%*%v)/nsample
```

```
      a <- t(z.x)%*%temp
      if(!is.null(Psi)){a <- a - Psi %*% (t(Psi) %*% a)} # get orthogonal of a
      if (sum(abs(a))==0) stop
      v <- constOpt2(a/sqrt(sum(a^2)), Psi, mu, tol,tol.1)
      value=sum((z.x%*%v)^2)
      #print(c("value=",value))
      #if(!is.null(Psi)){print(t(Psi)%*%v)}
     }
    qq[j]=sum((z.x%*%v)^2)
     #print(c("qq=",qq[j],2))
    if((mqq<qq[j]))
      {vc=v
       mqq=qq[j]
      }
    }
  print(c("The component",ncomp))

  PCs=cbind(PCs,vc/sqrt(sum(vc^2)))
  if(orthogonal)
   {Psi <-cbind(Psi, vc)}else
      {temp=(z.x%*%vc)/nsample;
       u <-t(z.x)%*%temp;
       if(!is.null(Psi)){u=u-Psi%*%(t(Psi)%*%u)}
       Psi=cbind(Psi, u/sqrt(sum(u^2)))
      }
 }
 if(corr)
   {PCs=t(scale(t(PCs),center=F,normx))
    PCs=scale(PCs,center=F)/sqrt(dim(PCs)[1]-1)
   }
}

return(PCs)
}
}
```

Appendix D: Core Code for SFABP

```
#### FUNCTION: use spcaBP to do factor analysis, return factor loadings and factor scores.

spcaBP.fa = function(x, k, lambda){  ###x is a data matrix, centers is the number of groups., group is the group factor indicating what class is the observation in


        loadings=NULL

        sdx=scale(x, center = TRUE, scale=TRUE)

        PCs=spca.BP(sdx, k,lambda, orthogonal=T) #x the n*p data matrix with n samples and p variables

            for(j in 1:k){ # k is the numbers of factors

               loadings = cbind(loadings,sqrt(t(PCs[,j]) %*% (cor(x)) %*% PCs[,j]) * PCs[,j])

            }

        scores= t(solve(t(loadings) %*% loadings) %*% t(loadings) %*% t(sdx)) ####get factor scores  n*k


      colnames(loadings) <- paste("Factor", 1:k, sep = "")
      rownames(loadings) <- colnames(x)
      colnames(scores) <- paste("SC", 1:k, sep = "")

      fit = matrix(0,3,k)
      fit[1,]=colSums(loadings^2)
      fit[2,]= fit[1,]/ncol(x)

      for(i in 1: k){
        fit[3,i]= sum(fit[2, 1:i])
      }

      colnames(fit) =colnames(loadings)
      names(fit[2,]) = paste("Proportion Var")
      return(list( loadings=loadings, scores=scores, fit=fit))
}

#### FUNCTION: return classification error rate
misclassification.rate=function(predict,grouping){
tab = table(predict,grouping)
num1=sum(diag(tab))
denom1=sum(tab)
signif(1-num1/denom1,3)

#ce= sum(tab[row(tab) != col(tab)]) / sum(tab)
}
```

FUNCTION: do classification, KNN-cv or LDA-cv, return classification error rate and the classification group factors.

spcaBP.classify = function(x, grouping, method="KNN"){ ###x is a data matrix or a factor score matrix, centers is the true number of groups., grouping is the group factor indicating what class is## ##the ##observation in

```
    ####use knn.cv to do classification and see the classification error rate

    cr=factor(grouping)
    if(method=="KNN"){
       cluster = knn.cv(x, cr)


    }
    else if (method=="LDA"){

        cl = lda(x, cr,CV = TRUE)
        cluster = cl$class


    }


     ce = misclassification.rate(cluster,cr)


    return(list(ce=ce, cluster=cluster))
}

##### FUNCTION: SFABP: CV to choose optimal factors and lambda
SPCA.BP.optimal.k.lambda.cv=function(data, cl, cv.num=10, KK=4,ratio=1/3,  method="KNN" )
{

   x=data.matrix(data)
   n=nrow(data)
   ncol=ncol(data)

   n1=as.integer(n*ratio)

   #boot.s=matrix(NA,boot.num, KK)

   lambda=c(0.00001, 0.0001,0.001, 0.01, 0.05, 0.1, 0.2, 0.3 ) ### for colon data
   #lambda=c(0.001, 0.01, 0.05, 0.1, 0.2, 0.3 )
   cv.s = array(0 ,dim=c(KK,length(lambda),cv.num))
   cv.r = matrix(NA, KK, length(lambda))
     for (cv.i in 1:cv.num) {
       sn = sample(1:n,n,replace=F)
       x.perm=x[sn,]
       x.tr=x.perm[1:(2*n1),]
```

```
    x.val=x.perm[-(1:(2*n1)),]
    cl.perm=cl[sn]
    cl.tr=cl.perm[1:(2*n1)]
    cl.val=cl.perm[-(1:(2*n1))]

    sdx.tr=scale(x.tr, center = TRUE, scale=TRUE)
    #sdx.val=scale(x.val, center = TRUE, scale=TRUE)

    for (factors in 1:KK)  {
        for(nl in 1: length(lambda)){

            #### Qi's SPCA
                PCs=spcaBP.fa(sdx.tr, factors, lambda=lambda[nl])

            scores.tr = PCs$scores

            scores.val = spcaBP.predict.score(PCs, x.val)
            if(method=="KNN"){
                precl = spcaBP.predict.cl(scores.tr, scores.val, cl.tr, cl.val,  method="KNN")
            }
            else if (method=="LDA"){
                precl = spcaBP.predict.cl(scores.tr, scores.val, cl.tr, cl.val,  method="LDA")
            }

            #### release memory,which is important for large dataset
            rm(PCs)
            gc()


                cv.s[factors, nl, cv.i] = precl$ce


        }

    }
     print (cv.i)

}


cv.r = apply(cv.s, c(1,2), mean,na.rm=TRUE)
pos=which(cv.r==min(cv.r), arr.ind = TRUE)

opt.k = pos[1,1]
opt.lambda=lambda[pos[1,2]]
return(list(cv.r=cv.r, cv.s=cv.s, lambdas=lambda,k=opt.k, lambda=opt.lambda))
}
```