STATISTICAL METHODS FOR LEARNING SPARSE FEATURES

BY

JIANJUN HU

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Statistics
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

Doctoral Committee:

Professor Feng Liang, Chair
Professor Douglas G. Simpson
Professor Xiaofeng Shao
Professor Xiaohui Chen

Abstract

With the fast development of networking, data storage, and the data collection capacity, big data are now rapidly expanding in all science and engineering domains. When dealing with such data, it is appealing if we can extract the hidden sparse structure of the data since sparse structures allow us to understand and interpret the information better. The aim of this thesis is to develop algorithms that can extract such hidden sparse structures of the data in the context of both supervised learning and unsupervised learning.

In chapter 1, this thesis first examines the limitation of the classical Fisher Discriminant Analysis (FDA), a supervised dimension reduction algorithm for multi-class classification problems. This limitation has been discussed by Cui (2012), and she has proposed a new objective function in her thesis, which is named *Complementary Dimension Analysis* (CDA) since each sequentially added new dimension boosts the discriminative power of the reduced space. A couple of extensions of CDA are discussed in this thesis, including sparse CDA (sCDA) in which the reduced subspace involves only a small fraction of the features, and Local CDA (LCDA) that handles multimodal data more appropriately by taking the local structure of the data into consideration. A combination of sCDA and LCDA is shown to work well with real examples and can return sparse directions from data with subtle local structures.

In chapter 2, this thesis considers the problem of matrix decomposition that arises in many real applications such as gene repressive identification and context mining. The goal is to retrieve a multi-layer low-rank sparse decomposition from a high dimensional data matrix. Existing algorithms are all sequential algorithms, that is, the first layer is estimated, and then remaining layers are estimated one by one, by conditioning on the previous layers. As discussed in this thesis, such sequential approaches have some limitations. A new algorithm is proposed to address those limitations, where all the layers are solved simultaneously instead of sequentially.

The proposed algorithm in chapter 2 is based on a complete data matrix. In many real appli-

cations and cross-validation procedures, one needs to work with a data matrix with missing values. How to operate the proposed matrix decomposition algorithm when there exist missing values is the main focus of chapter 3. The proposed solution seems to be slightly different from some existing work such as penalized matrix decomposition (PMD).

In chapter 4, this thesis considers a Bayesian approach to sparse principal component analysis (PCA). An efficient algorithm, which is based on a hybrid of Expectation-Maximization (EM) and Variational-Bayes (VB), is proposed and it can be shown to achieve selection consistency when both p and n go to infinity. Empirical studies have demonstrated the competitive performance of the proposed algorithm.

To my beloved parents, for their love and support.

Acknowledgments

This dissertation would never been possible without the support of many people.

I would like to express my sincere gratitude to my advisor Professor Feng Liang. Her guidance enlightened me with the first glance of research; her innovative ideas inspired me to explore new research directions perseveringly; her patience and positive attitude helped me overcome the hard time; her honesty and sense of responsibility taught me to be a better man in my lifetime.

Besides my advisor, I am also grateful to the rest of my thesis committee members: Professors Douglas Simpson, Xiaofeng Shao and Xiaohui Chen, for their insightful perspectives and suggestions on my dissertation. Meanwhile, I really appreciate Dr. Na Cui's contribution on complementary discriminant analysis algorithm in Chapter 1.

Furthermore, I want to thank Department of Statistics providing me a learning opportunity as well as a homelike atmosphere. My sincere thanks also go to Professor Yuguo Chen, Annie Qu, John Marden and Adam Martinsek who offered me orientations and advices during my master and PhD journey. Truly thanks to faculties Darren Glosemeyer and David Unger for their generous help on my TA assignments for multiple semesters and to staff Melissa Banks who gave access to useful information and logistical convenience. I also appreciate the cherish friendship coming from the fellow students in Statistics and Mathematics departments.

Last but not the least, special thanks should be given to my parents. I always gain confidence through their selfless love and generous supports when facing difficulties and enjoy the blessed moments shared with them together.

Table of Contents

List of Tables

List of Figures

Chapter 1

Sparse Dimension Reduction

1.1 Introduction

The goal of dimension reduction (DR) is to find a compact yet informative representation of the p-dimensional feature vector X via some transformation. For linear methods, it is equivalent to finding a projection matrix $\mathbf{V}_{p\times m} = [v_1, \cdots, v_m]$, which can extract the key information in X by a m-dimensional summary matrix $\mathbf{V}^t X$ where $m \ll p$. In the setting of supervised learning such as classification or regression, the projection $\mathbf{V}$ is chosen such that $\mathbf{V}^t X$ keeps the most discriminative information of the response variable Y.

Most DR algorithms are formulated as a sequential optimization problem with respect to a function $G(\cdot)$: after l directions have been retrieved, the $(l+1)$th direction is retrieved by solving

$$v_{l+1} = \arg\max_{v \perp M_l} G(v), \tag{1.1.1}$$

where M_l denotes the linear space spanned by the previously l directions: $v_1, \ldots, v_l$. For example, in Fisher's Discriminant Analysis (FDA) (Fisher, 1936), we have $G(v) = v^t \mathbf{B} v$, where $\mathbf{B}$ is between-class scatter matrix and the data have been normalized so that the within-class scatter matrix is an identity matrix. The objective function of FDA is a linear function of the L_2 norm of the data, a feature shared by many other DR algorithms. An advantage of such objective functions is that the solution is in closed form and can be solved by eigen-decomposition. The drawback, however, is that the retrieved subspace is suboptimal for multi-class classification or regression problems.

In Cui's thesis (Cui, 2012), she considered a dataset with four classes located in $\mathbb{R}^3$ (see Figure 1.1). The data are generated from a mixture of four Gaussian distributions with a common identity covariance matrix $\mathbf{I}_3$ and different mean vectors located at a, b, c, and d, where d is relatively far away from the others. The first direction chosen by FDA is roughly the Z-axis, which separates all

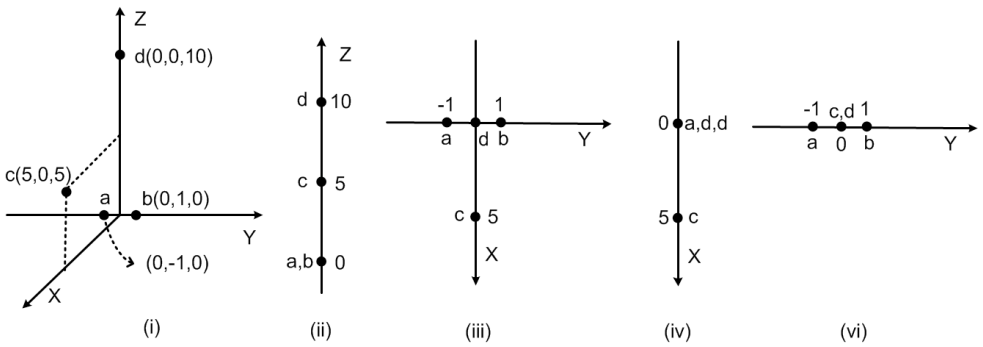

Figure 1.1: The toy example with $a = (-1, 0, 0)$, $b = (1, 0, 0)$, $c = (0, 5, 5)$ and $d = (0, 0, 10)$. The projection of the 3-dimensional data onto Z-axis, XY-plane, X-axis and Y-axis are shown in (ii), (iii), (iv) and (vi) respectively.

the classes except classes a and b. The second direction is the X-axis, however, it still leaves class a and b mixed together. Alternatively, if Y-axis were chosen as the second direction in the reduced space, then class a and b could be separated.

Cui (2012) pointed out the objective function used by FDA, as well as any objective function that is a linear function of the L_2-norm of the data, tends to: *1)* overemphasize directions that result in large between-class distances but little improvement over the classification accuracy, and *2)* overlook directions that result in a small margin of between-class distances but a big improvement over the classification accuracy.

Motivated by some earlier works such as Loog and Haeb-Umbach (2001) and Sugiyama (2006), Cui (2012) proposed a new objective function which is directly linked to the classification accuracy of the projected data $\mathbf{V}^t X$. Suppose we have a K ($K > 2$) classes dataset, for any pair of classes, i and j, the corresponding classification accuracy in the reduced subspace $\mathbf{V}^t X$ is given by

$$A(\|\mathbf{V}m_{ij}\|^2) = \frac{1}{2} + \frac{1}{2}\text{erf}\left(\frac{\|\mathbf{V}^t m_{ij}\|}{2\sqrt{2}}\right),$$

where m_{ij} denotes the pairwise mean difference between class i and class j, and

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$$

is the normal error function. Then the objective function is defined to be averaged classification

accuracy for all pairwise classes,

$$G(\mathbf{V}) = \sum_{i,j=1}^{K} p_i p_j A(\|\mathbf{V}^t m_{ij}\|^2)), \tag{1.1.2}$$

where p_i is the prior of class i.

This new objective function, however, cannot be solved by eigen-decomposition to have a closed form. Cui (2012) derived an efficient Complementary Dimensionality Analysis (CDA) algorithm that sequentially solves this nonlinear objective function.

Algorithm 1.1 CDA Algorithm

1: **Step 1** At the $(l+1)$th step, given previous l solved directions, $v_1, \ldots, v_l$, form a $p \times l$ projection matrix $\mathbf{V}_l = [v_1, v_2, \ldots, v_l]$.

2: **Step 2** For any pairwise classes i and j, update

3: (a) $m_{ij(l)} = \mathbf{V}_l^t m_{ij}$ and $e_{ij(l)} = m_{ij} - m_{ij(l)}$.

4: (b) $b_{ij}^{(l)} = \frac{1}{2\|e_{ij(l)}\|^2}\left[\text{erf}\Big(\frac{\|m_{ij}\|}{2\sqrt{2}}\Big) - \text{erf}\Big(\frac{\|m_{ij(l)}\|}{2\sqrt{2}}\Big)\right]$.

5: (c) $\mathbf{S}_{l+1} = \sum_{i,j=1}^{K} p_i p_j b_{ij}^{(l)} e_{ij(l)} e_{ij(l)}^t$.

6: **Step 3** Find the 1st eigen-vector of the matrix $\mathbf{S}_{l+1}$, and set it to be v_{l+1}.

The key motivation of CDA algorithm is that each sequentially added direction should boost the discriminative power of the reduced space. Specifically, when retrieving the $l+1$-th direction v_{l+1}, it works with an updated objective function $G_{l+1}(v) = v^t \mathbf{S}_{l+1} v$ so that the solution v_{l+1} complements the previously solved directions $v_1, \cdots, v_l$ in terms of classification accuracy. This is why the new algorithm is named as *Complementary* Dimensionality Analysis.

1.2 Sparse CDA

Recall the toy example discussed before, where ideally we would like to project the data onto a two dimensional subspace, with the first direction being the Z-coordinate and the second being the X-coordinate. When we apply CDA algorithm on this simulated toy data, however, we always end up obtaining two directions which are close to, but not exactly, the Z and X coordinates.

In this section, we discuss how to retrieve sparse CDA directions. Here "sparse" means the most loadings of CDA directions are zero, i.e., the discriminant direction only involves a small fraction of

the p features. Such an extension is important for many real applications nowadays, which usually involve a large dimension of features, therefore it is desirable to have an algorithm that can do both dimension reduction and variable selection.

1.2.1 Methodology

In CDA algorithm, when retrieving the $l+1$th direction v_{l+1}, we calculate the 1st eigen vector of a each step updated matrix $\mathbf{S}_{l+1}$. There have been many works on sparse eigen-vectors on a matrix. Our work is motived by the ideas in Zou et al. (2006) and Shen and Huang (2008). In our proposed method, we first establish a connection between eigen-decomposition solution on a data matrix and an OLS model, and then we achieve sparsity by introducing an L_1 penalty on the estimated direction at each step. We start to review the connection between estimating the 1st eigen-vector and OLS in the following proposition.

Proposition 1.1. *Suppose S is a $p \times p$ positive definite matrix with an unique largest eigen value. Let α, $\beta \in \mathbb{R}^p$ and $\hat{\alpha}$, $\hat{\beta}$ are solved through*

$$(\hat{\alpha}, \hat{\beta}) = \arg\min_{\alpha,\beta} \|S - \alpha\beta^t\|_F^2 \tag{1.2.1}$$

$$\textit{subject to } \|\alpha\|_2 = 1,$$

where $\|\cdot\|_2$ is the L_2 norm for a vector, and $\|\cdot\|_F$ is the Frobenius norm for a matrix. Then $\hat{\alpha}=\frac{\hat{\beta}}{\|\beta\|_2}$ are both eigen vector corresponding to S's largest eigen value.

Proposition 1.1 is proved in the Appendix. To achieve sparsity on the direction β, we add a L_1 penalty to the objective function (1.2.1) and the optimization problem becomes

$$(\hat{\alpha}, \hat{\beta}) = \arg\min_{\alpha,\beta} \|S - \alpha\beta^t\|_F^2 + \lambda\|\beta\|_1 \tag{1.2.2}$$

$$\textit{subject to } \|\alpha\|_2 = 1.$$

where $\|\cdot\|_1$ stands for the L_1 norm for a vector.

Again, this problem does not have a closed form solution. Here, we design a two steps interactive algorithm to solve α and β. Notice that given β, we can ignore the penalty term and the solution

of α is retrieved as

$$\hat{\alpha} = \arg \min_{\|\alpha\|_2=1} \|S - \alpha\beta^t\|_F^2.$$

Using Lagrange multipliers method and take a derivative related to α from $\|S-\alpha\beta^t\|^2+\gamma(\|\alpha\|_2^2-1)$, we have

$$\begin{cases} -2(S - \alpha\beta^t)\beta + 2\gamma\alpha = 0 \\ -2S\beta + \beta^t\beta\alpha + 2\gamma\alpha = 0 \end{cases}$$

Therefore, $\alpha = \frac{S\beta}{\beta^t\beta+\gamma} = \frac{S\beta}{\|S\beta\|_2}$. When given α, there is no obvious way to find the solution of $\hat{\beta} = \arg\min_\beta \|S - \alpha\beta^t\|^2 + \lambda\|\beta\|_1$, subject to $\|\alpha\|_2 = 1$. So we first claim the following proposition, and the problem can be transformed into an easier form.

Proposition 1.2. *For a fixed α with constrain $\|\alpha\| = 1$, the penalized L_2 norm minimization problem for β:*

$$\hat{\beta} = \arg \min_{\beta} \|S - \alpha\beta^t\|^2 + \lambda\|\beta\|_1 \tag{1.2.3}$$

is equivalent to a penalized regression problem with identity design matrix:

$$\hat{\beta} = \arg \min_{\beta} \|S^t\alpha - \beta\|^2 + \lambda\|\beta\|_1 \tag{1.2.4}$$

For proposition 1.2, it shows in the Appendix that we could write $\|S - \alpha\beta^t\|^2 = \|S^t\alpha - \beta\|^2 + \phi(\alpha)$, where $\phi(\alpha)$ is a function only related to α. So for a fixed α, the solution $\hat{\beta} = \arg\min_\beta \|S - \alpha\beta^t\|^2 + \lambda\|\beta\|_1$ is the same as the solution of $\hat{\beta} = \arg\min_\beta \|S\alpha - \beta\|^2 + \lambda\|\beta\|_1$. Notice that the solution of objective function (1.2.4) is a soft thresholding rule on each element of the ordinary least square estimator $\hat{\beta}_{\text{OLS}} = S\alpha$.

Combining the above discussions, we propose a two steps iteration algorithm for sparse CDA as following.

Algorithm 1.2 Sparse CDA (sCDA) Algorithm

1: **Initialization**: Use the eigen-vector corresponding to the largest eigen value of S_l as the iteration starting value for α.

2: **Update**:

- Given α, the solution for j-th element of β is a soft thresholding of $\hat{\beta}_j$

$$\beta_j = (|\hat{\beta}_j| - \lambda/2)_+ \text{sign}(\hat{\beta}_j),$$

  where $\hat{\beta}_j$ is the j-th element of ordinary least square estimation $\hat{\beta}_{\text{OLS}} = S^t\alpha$.

- Given solved β, update the value of α by $\alpha = \frac{S\beta}{\|S\beta\|_2}$.

3: **Repetition**: Repeat the two steps in **Update** procedure until the solution converges.

4: **Normalization**: Normalize the sparsity defection, i.e. $\beta = \frac{\beta}{\|\beta\|_2}$.

In our R implementation of this algorithm, we provide two ways to control the sparsity: set the λ value, or specify the L_0 norm of β, i.e., the number of non-zero coefficients.

1.2.2 Experiments: Toy Data with Noise

We first revisit the toy example in the introduction section. To test the performance of our sparse CDA algorithm, we add three noise dimensions to make a new $\mathbb{R}^6$ toy data. We consider a projected data onto a 2-dim subspace, if the method works well, the solution should explore those three noise dimensions with 0 coefficients. We apply 6 different algorithms including our sCDA method on this data and compare all the results in Figure 1.2. Before we explain the results, here is a short summary of all the methods we use.

- **PCA**: Principle component analysis (Jolliffe, 2002) is a well know DR technique seeking the linear combinations of the original variables such that the derived variables capture maximal variance.

- **FDA**: Fisher discriminant analysis (Fisher, 1936) is a popular method utilizing the label information in finding informative projections.

- **aPAC**: Approximation pairwise accuracy criterion is a method proposed by Loog and Haeb-Umbach (2001). They define a new objective function based on a weighted variant of the FDA,

and the weights approximate the mean accuracy among all pairs of classes.

- **SIR**: Sliced inverse regression is a dimension reduction method proposed by Li (1991). It uses the inverse regression curve to perform a weighted principal component analysis, and the effective dimension reducing directions are then estimated.

- **CDA**: Complimentary dimension analysis is proposed in Cui's thesis (Cui, 2012). It defines an objective function directly linked to the classification accuracy and it sequentially adds directions boosting the discriminative power of the reduced space.

- **sCDA**: This is our new algorithm generalizing from CDA, and it can retrieve sparse CDA directions.

Then we compare the performance of all the methods in Figure 1.2 by showing the data points in the 2-dim reduced space derived by these methods. Only CDA and SCDA successfully separates the four classes in a 2-dimension subspace. PCA, FDA, SIR and aPAC separate class c and d away from others and leave classes a and b mixed together. PCA can not achieve the final goal since it is a unsupervised learning method. FDA is not optimal due to the discrepancy between the objective function and the classification accuracy. aPAC also fails even with the revised objective function, that is because it does not incorporate the influence of the previously found directions in their algorithm. CDA borrows the idea from aPAC by reconstructing the objective function directly linked with the classification accuracy. However, different from aPAC which only keeps the coefficients same in each step, CDA updates the coefficients in the objective function sequentially by only considering the complementary directions to previous. This updated coefficients help the method to avoid digging out directions containing similar information in each step.

Meanwhile, if we take look the details of each direction, CDA cannot ignore noise dimensions and there are the retrieved two directions are not exactly the ideal Z and X axis. We then apply sCDA with $\lambda_1 = 0.01$ and $\lambda_1 = 0.0001$. The sparsity property guarantees that the estimated two directions do not contain noise dimensions. Furthermore, the larger λ_1 case ($\lambda_1 = 0.01$) leads to solution Z and X-axis, which exactly match the ideal projection directions. All the CDA and sCDA results are shown in Table 1.1.

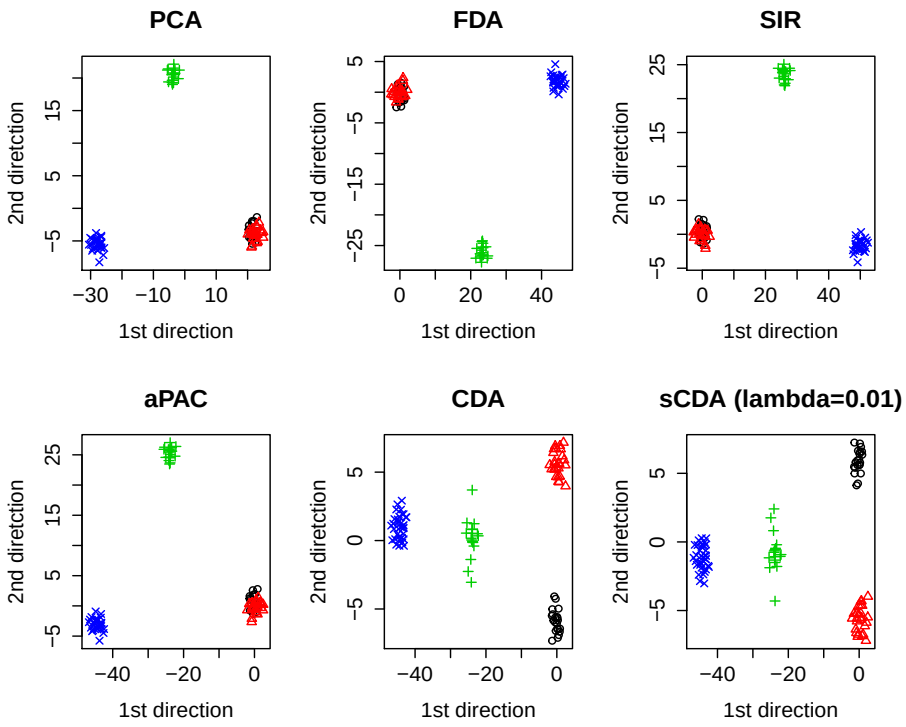

Figure 1.2: Visualization of the data points in the 2-dim reduced subspace derived by PCA, FDA, SIR, aPAC, CDA and sCDA for the toy example with noise.

	CDA		sCDA, $\lambda_1 = 0.0001$		sCDA, $\lambda_1 = 0.001$		sCDA, $\lambda_1 = 0.01$	
	1st Dir	2nd Dir	1st Dir	2nd Dir	1st Dir	2nd Dir	1st Dir	2nd Dir
x	0.1693	0.9738	-0.1618	-1	-0.0459	-1	0	-1
y	-0.2408	-0.0672	0.2486	0	0	0	0	0
z	-0.9506	0.1842	0.9550	0	0.9989	0	1	0
noise1	-0.0715	0.0627	0	0	0	0	0	0
noise2	-0.0454	0.0843	0	0	0	0	0	0
noise3	0.0494	0.0471	0	0	0	0	0	0

Table 1.1: The first two directions retrived by applying CDA and sCDA (two cases with different values of λ_1) to toy data example with three extra noise dimensions.

1.3 Swiss Roll with Local CDA

In CDA algorithm, we assume that data from each class can be well approximated by a Gaussian distribution, and the center is used to represent each class in our objective function. However, the Gaussian assumption may not be satisfied in practice. For example, consider a simple situation where data from a class following a mixture of Gaussian with two components far away from each other. Then it is no longer suitable to use the class mean to summarize data, instead, a better summary should be means for each sub-class. This simple case indicates that when data has a local structure, we may need to find other way to deal with it. There are already several DR methods dealing with local structure data, like He and Niyogi (2003) and Sugiyama (2007). In this section, we introduce an extension of CDA algorithm, called Local CDA (LCDA), which measures the accuracy based on a local classification rule. When we test LCDA on the $\mathbb{R}^3$ Swiss Roll data example (the class setting is shown in Figure 1.3), we also consider adding extra three noise dimensions to form a $\mathbb{R}^6$ new data.

1.3.1 Local CDA

To deal with data having local structure, we first introduce a new modified data set $\tilde{\mathbf{X}} = (\tilde{\mathbf{x}}_1^t, \tilde{\mathbf{x}}_2^t, ..., \tilde{\mathbf{x}}_n^t)^t$, in which the ith data point is equal to the corresponding class center, that is,

$$\tilde{\mathbf{x}}_i = m_{y_i}, \ y_i \in \{1, 2, ..., K\}$$

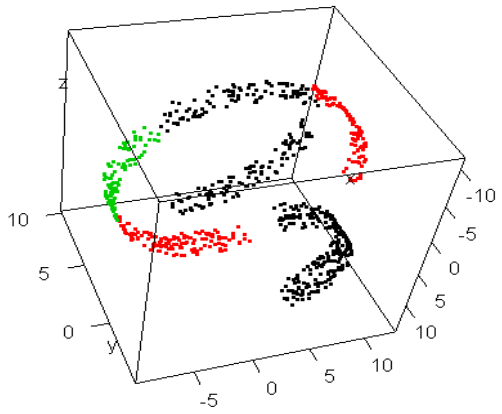

Figure 1.3: 3D plot of the Swiss Roll data. There are three colors representing three different classes. The Black and Red groups have subclasses far away from each other. Two of the subclasses in Black group also have a shift along y-axis.

Then we can rewrite our objective function (1.1.2), which is a summation of all pairwise classes, as a summation over pairwise data points for this new data matrix $\tilde{\mathbf{X}}$.

$$\begin{aligned} G(\mathbf{V}) &= \sum_{k=1}^{K-1}\sum_{k'=k+1}^{K} p_k p_{k'}\Big[\frac{1}{2}+\frac{1}{2}\mathrm{erf}\Big(\frac{\sqrt{\|\mathbf{V}^t m_{kk'}\|^2}}{2\sqrt{2}}\Big)\Big] \\ &= \frac{1}{n^2}\sum_{k=1}^{K-1}\sum_{k'=k+1}^{K} n_k n_{k'}\Big[\frac{1}{2}+\frac{1}{2}\mathrm{erf}\Big(\frac{\sqrt{\|\mathbf{V}^t m_{kk'}\|^2}}{2\sqrt{2}}\Big)\Big] \\ &= \frac{1}{n^2}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \mathbf{I}(y_i \neq y_j)\Big[\frac{1}{2}+\frac{1}{2}\mathrm{erf}\Big(\frac{\sqrt{\|\mathbf{V}^t(\tilde{\mathbf{x}}_i-\tilde{\mathbf{x}}_j\|^2}}{2\sqrt{2}}\Big)\Big] \\ &= \frac{1}{n^2}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \mathbf{I}(y_i \neq y_j)\Big[\frac{1}{2}+\frac{1}{2}g(\tilde{\mathbf{x}}_i-\tilde{\mathbf{x}}_j)\Big]. \end{aligned}$$

So we can view our CDA objective function as the pairwise classification error for any two data points from different classes, but before applying CDA, we *denoise* the data by moving each data point to its class center.

Naturally, when data in each class cannot be well-approximated by a Gaussian distribution, the class mean will not be a good summary for each class. Then we should *denoise* the data by moving each data point to its local mean, for example, an average over its k nearest neighbors within its class, and then apply CDA on this new data set. Our Local CDA (LCDA) algorithm uses each kind of idea and its steps are summarized in the following procedure.

Algorithm 1.3 Local CDA (LCDA) Algorithm

1: For each sample (x_i, y_i), compute

$$\tilde{\mu}_{i,loc} = \frac{1}{k}\sum_{j\in s_i} x_j$$

where s_i={j: x_j belongs to the k nearest neighbors of x_i in group y_i }.

2: Calculate the objective function

$$G(\mathbf{V}) = \frac{1}{n^2}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\mathbf{I}(y_i \neq y_j)\big[\frac{1}{2} + \frac{1}{2}g(\tilde{\mu}_{i,loc} - \tilde{\mu}_{j,loc})\big].$$

3: Use the sequential method and linear function approximation to get direction v at each step, i.e., at step $l+1$, write $\tilde{\mu}_{ij} = \tilde{\mu}_{i,loc} - \tilde{\mu}_{j,loc}$, $\tilde{\mu}_{ij(l)} = \mathbf{V}_l^t\tilde{\mu}_{ij}$ and $\tilde{e}_{ij(l)} = \tilde{\mu}_{ij} - \tilde{\mu}_{ij(l)}$,

$$\begin{aligned}
G_{l+1}(v) &= G([\mathbf{V}_l, v]) \\
&= \frac{1}{n^2}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\mathbf{I}(y_i \neq y_j)\Big[\frac{1}{2} + \frac{1}{2}\mathrm{erf}\Big(\frac{\sqrt{\|[\mathbf{V}_l, v]^t(\tilde{\mu}_{i,loc} - \tilde{\mu}_{j,loc})\|^2}}{2\sqrt{2}}\Big)\Big] \\
&= \frac{1}{n^2}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\mathbf{I}(y_i \neq y_j)\Big[\frac{1}{2} + \frac{1}{2}\mathrm{erf}\Big(\frac{\sqrt{\|\mathbf{V}_l^t\tilde{\mu}_{ij}\|^2 + \|v^t\tilde{\mu}_{ij}\|^2}}{2\sqrt{2}}\Big)\Big] \\
&= \frac{1}{n^2}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\mathbf{I}(y_i \neq y_j)\Big[\frac{1}{2} + \frac{1}{2}\mathrm{erf}\Big(\frac{\sqrt{\|\tilde{\mu}_{ij(l)}\|^2 + \|v^t\tilde{e}_{ij(l)}\|^2}}{2\sqrt{2}}\Big)\Big] \\
&\approx \frac{1}{2n^2}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\mathbf{I}(y_i \neq y_j)\Big[a_{ij}^{(l)} + b_{ij}^{(l)}v^t\tilde{e}_{ij(l)}\tilde{e}_{ij(l)}^t v\Big]
\end{aligned}$$

$$v = \arg\max_v v^t S_{l+1} v, \ \textit{where}\ \mathbf{S}_{l+1} = \frac{1}{2n^2}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\mathbf{I}(y_i \neq y_j)b_{ij}^{(l)}\tilde{e}_{ij(l)}\tilde{e}_{ij(l)}^t$$

where $b_{ij}^{(l)}$ is solved using similar method in algorithm 1.1, and $a_{ij}^{(l)}$ is not relevant since the matrix $\mathbf{S}_{l+1}$ at each step only depends on $b_{ij}^{(l)}$.

Similarly to sparse CDA, we can also retrieve a sparse solution for the $(l+1)$th direction in LCDA's third procedure by adding a L_1 penalty after $\mathbf{S}_{l+1}$ is calculated. We call this method Sparse LCDA (sLCDA).

1.3.2 Experiment Results

Simulation shows CDA does not completely separate three groups with a classification error 0.216 on the projected testing data. As we mentioned before, CDA uses the class mean to summarize the data and the three black binds in swiss roll are far away from each other although they are assigned in the same group. Therefore, using the whole group's mean will misinterpret the structure of the data. Moreover, CDA also does not have the ability to identify the noise directions so that every noise dimension has non-zero loadings. We also try sparse LCDA (sLCDA) with penalty parameter $\lambda_1 = 0.00001$, 0.0001, and 0.001 to the same data. All the results identify noise directions and larger λ_1 provides sparser direction estimations. Especially for $\lambda_1 = 0.001$, the two projection directions are exactly X and Z coordinate. It also convinces us that the testing data is better separated onto the 2-dim subspace by sLCDA than CDA from Figure 1.4.

	CDA		sLCDA, $\lambda_1 = 0.00001$		sLCDA, $\lambda_1 = 0.0001$		sLCDA, $\lambda_1 = 0.001$	
	1st Dir	2nd Dir	1st Dir	2nd Dir	1st Dir	2nd Dir	1st Dir	2nd Dir
x	0.1693	0.9738	-0.9645	0	-0.9997	0	-1	0
y	-0.2408	-0.0672	0	-0.4846	0	0	0	0
z	-0.9506	0.1842	0.2641	-0.8747	0.0237	-1	0	-1
noise1	-0.0715	0.0627	0	0	0	0	0	0
noise2	-0.0454	0.0843	0	0	0	0	0	0
noise3	0.0494	0.0471	0	0	0	0	0	0

Table 1.2: The first two directions retrieved by CDA and sLCDA for swiss roll data with three extra noise dimensions.

1.4 Penicillium Data

We now analyze a real high-dimensional data related to three species of Penicillium fungi: Melanoconidium, Polonicum, and Venetum. This data has 36 samples (first 12 are P. Melanoconidium species, 13-24 are P. Polonicum species, and the last 12 are P. Venetum species) with 3754 variables extracted from multi-spectral images of the three species. The data was analyzed before by Clemmensen et al. (2011) where they proposed a Sparse Discriminant Analysis (SDA) method.

In our simulation study, we first normalize the data to have unit variance for each feature, and delete features with all 0's. Then we divide the data into training (24 samples) and testing (12 samples), and apply our sCDA method on the 24 by 3536 training data matrix to retrieve a 2-

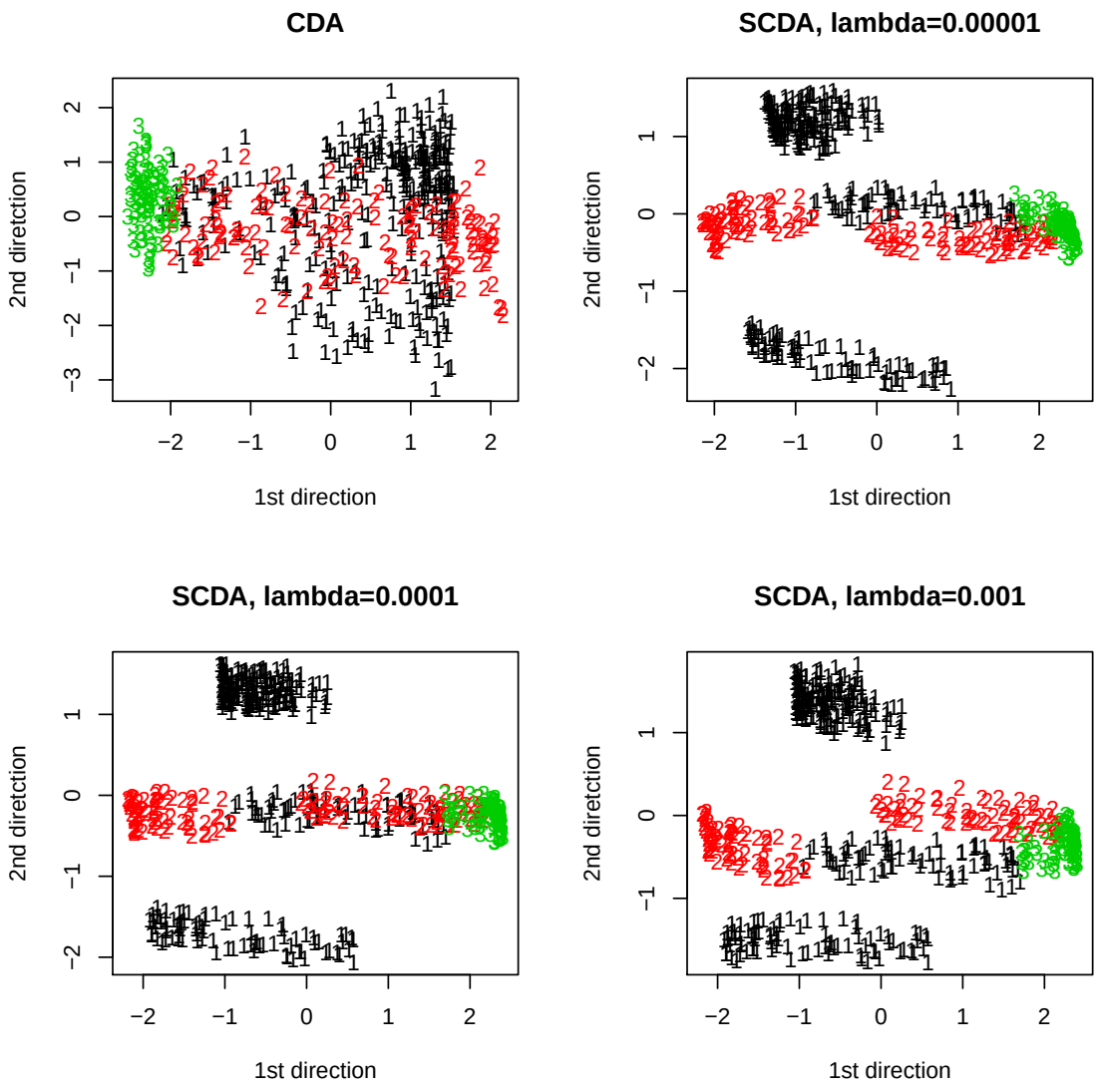

Figure 1.4: Swiss Roll testing data projected onto the 2-dimension subspace by CDA and sLCDA with different λ_1. CDA cannot separate the data well, and sparse LCDA with $\lambda_1 = 0.001$ has the best performance to recognize the same group with subclass.

dimension subspace. We repeat this experiment 10 times, each time training and testing sample are randomly split; the projection matrix is learned through sCDA on training; the testing data's group labels are predicted through 1 nearest neighbor(1-NN) between projected training sample with true label and projected testing sample. We also consider solutions in two different setups: *1.* each projection direction has only one active variable (one non-zero coefficient); *2.* each direction has two active variables. As a comparison, we also apply SDA method introduced by Clemmensen et al. (2011).

The classification error are summarized in Table 1.3. sCDA nearly has no errors on testing for all the iterations. And SDA has a worse performance in 2nd iteration of both setups with error rate 0.25. We also plot the the projected data (combination of training and testing) in a 2-dimensional subspace for the 2nd iteration in Figures 1.5 - 1.6. The projecting plots clearly indicate that our sCDA method has a better performance than SDA. SDA seems to overfit the training data and fails to correctly predict one group (three red triangle points) in testing.

	iter 1	iter 2	iter 3	iter 4	iter 5	iter 6	iter 7	iter 8	iter 9	iter 10
SDA error (setup I)	0	0.25	0	0	0	0	0.083	0.083	0	0
sCDA error (setup I)	0	0.083	0	0	0	0	0.083	0	0	0
SDA error (setup II)	0	0.25	0	0.167	0	0	0.083		0	0
sCDA error (setup II)	0	0.083	0	0	0	0	0	0	0.083	0

Table 1.3: Classification error on testing set for SDA and sCDA.

Moreover, SDA does not have a stable solution: it could return completely different directions when training different random samples. Based on the variable selection Tables 1.4 and 1.5, we find that SDA ends up with many different chosen directions for the first 6 iterations, for both setting *1* and *2*. For sCDA, although there is also no guarantee to have the same directions in each iteration, 936 and 1221 will be more possibly picked up as the 1st direction.

	iteration 1	iteration 2	iteration 3	iteration 4	iteration 5	iteration 6
SDA active ID(1st)	377	1220	377	420	936	1221
SDA active ID(2nd)	444	3220	444	375	2346	2003
sCDA active ID(1st)	1221	1220	1221	1219	1221	377
sCDA active ID(2nd)	1261	1582	1261	1586	2955	422

Table 1.4: Non-zero coefficient ID of projection directions for SDA and sCDA with setup *1.* 1st and 2nd represent the first and second retrieved directions.

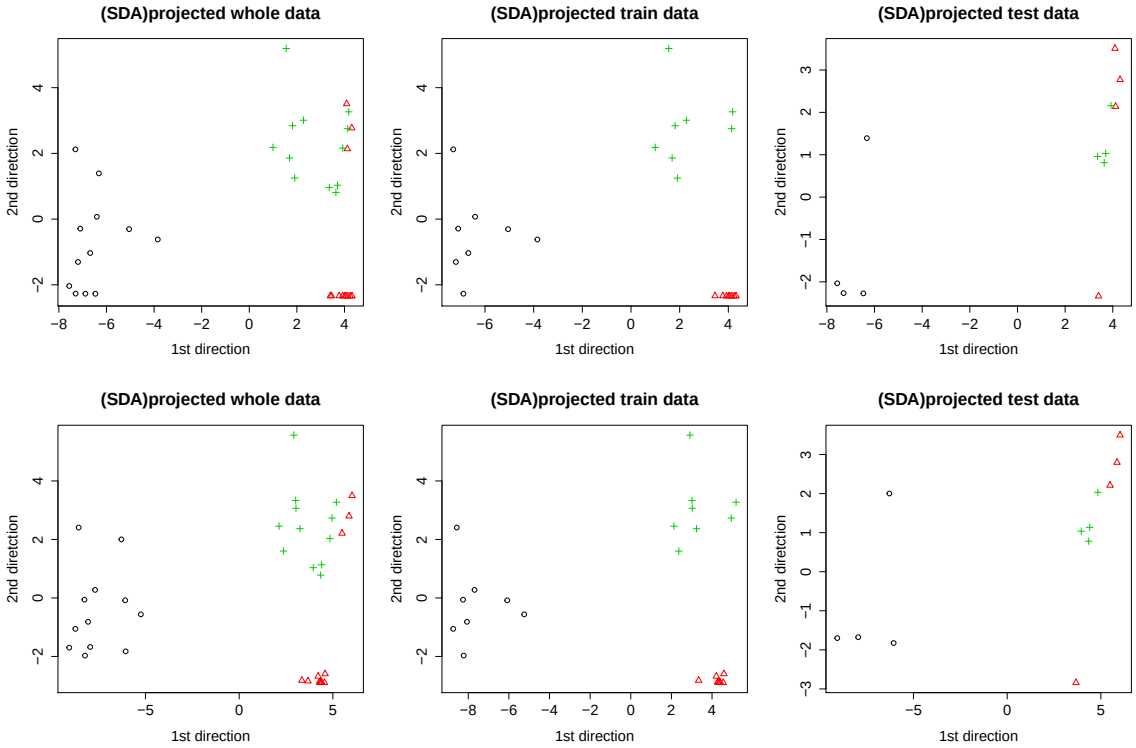

Figure 1.5: Projected testing data onto a 2 dimension subspace by using SDA with setup *1* and *2* in the 2nd experiment. The top three figures are projected whole, training, and testing data respectively with setup *1*. The bottom three figures are projected whole, training, and testing data respectively with setup *2*. Different groups are represented by points with different colors and shapes.

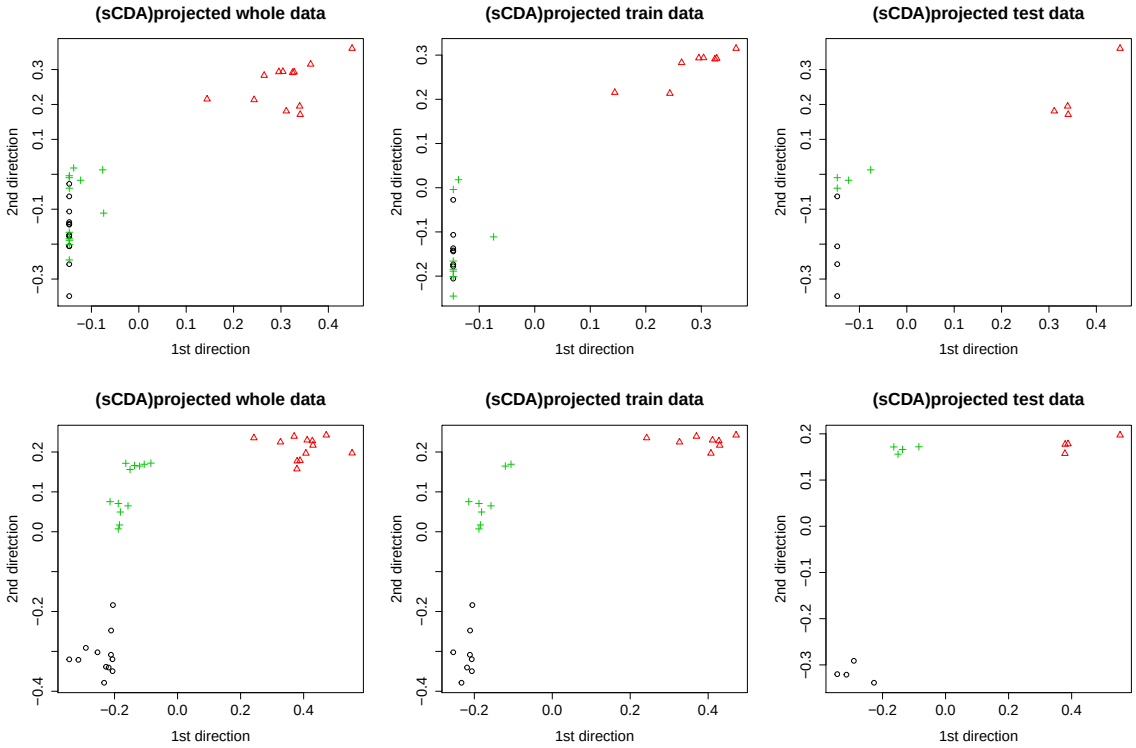

Figure 1.6: Projected testing data onto a 2 dimension subspace by using sCDA with setup *1* and *2* in the 2nd experiment. The top three figures are projected whole, training, and testing data respectively with setup *1*. The bottom three figures are projected whole, training, and testing data respectively with setup *2*. Different groups are represented by points with different colors and shapes.

	iteration 1	iteration 2	iteration 3	iteration 4	iteration 5	iteration 6
SDA active ID(1st)	444, 2150	200, 467	377, 3396	255, 259	263, 937	422, 2152
SDA active ID(2nd)	377, 2541	468, 937	444, 2151	810, 850	467, 2345	377, 937
sCDA active ID(1st)	936, 1221	1112, 1220	936, 1221	934, 1219	936, 1221	936, 1221
sCDA active ID(2nd)	1588, 3219	1582, 3220	2552,2718	1586, 3108	1588, 2955	2542, 2613

Table 1.5: Non-zero coefficient ID of projection directions for SDA and sCDA with setup *2*. 1st and 2nd represent the first and second retrieved directions.

1.5 Discussion

Most DR algorithms are formulated as an optimization problem, with an objective function which is a linear function of squared L_2 norm of between class distance. Minimizing such an objective function, however, does to lead to directions which can produce good classification results, due to the discrepancy between classification accuracy and the L_2 distance. In this chapter, we reviewed a new DR algorithm CDA, whose objective function is directly related to the classification accuracy. There is also an efficient algorithm to retrieve the directions sequentially. The CDA algorithm can be viewed as a weighted FDA algorithm, where the weights vary from class to class, and got updated from step to step to quantify the new contribution of a direction to the classification accuracy, in addition to the previously retrieved directions. This is why it is called Complementary Dimensionality Analysis, since any newly retrieved direction is making non-redundant contribution to the classification task.

We then mainly proposed two extensions of CDA algorithm. We extend our algorithm, by using an L_1 penalty, to retrieve sparse directions. This method named sparse CDA (sCDA) can identify noise directions and make each direction easier to explain. Moreover, the original CDA is derived based on a Gaussian distribution assumption for data within each class and such an assumption may fail in practice. Another extension, Local CDA (LCDA), is designed to handle this case. Furthermore, based on the similar idea of sCDA, our sparse LCDA method starts from LCDA and it can not only deal with local structure data but also provide sparse solutions to identify noise directions.

The empirical performances of our algorithms are promising, both in terms of accuracy and computation speed. Currently, we are trying to explore extensions of our work to other related areas, such as metric learning, subspace learning and recommendation system.

Chapter 2

Sparse Matrix Decomposition: A Regularization Method

2.1 Introduction

Principal component analysis (PCA) is the foundations for many methods of multivariate analysis. In quite a few real applications, in order to better understand the complex system, a natural approach is to break the data down into simpler components. Thus, matrix decomposition / representation of complex systems and data becomes an inspiring and challenging topic. Several methods were developed further on such as non-negative matrix factorization (NMF), sparse and low-rank matrix decomposition (SLRMD) and sparse singular value decomposition (SSVD). These methodologies usually rely on different assumptions and they are designed for various purposes.

One property for PCA is that the principle component vectors have both positive and negative coefficients. However, in many data-processing tasks, negative values are meaningless. For instance, when the columns of the input matrix were word counts from documents, negative values cannot be properly interpreted into different semantic categories. NMF addresses this issue by adding an non-negativity constraint on the matrix decomposition. The idea first came from Paatero and Tapper (1994) and Lee and Seung (1999) also independently introduced the NMF concept on unsupervised learning. Since the problem is not exactly solvable in general, it is commonly approximated with numerical methods. Paatero and Tapper (1994) proposed a constrained alternating least squares (ALS) algorithm to solve the problem, and some successful algorithms are based on alternating non-negative least squares: include the projected gradient descent methods (Lin (2007a), Lin (2007b)), the active set method (Kim and Park (2008), Gemulla et al. (2011)) and the optimal gradient method (Guan et al., 2012), e.t.c. Lee and Seung's multiplicative update rule also has been a popular method due to the simplicity of implementation.

Through PCA, the underlying data is approximately rotated on a low-dimensional linear subspace. However, the entries of the matrix could often be corrupted by errors or noise, some of the

entries could even be missing. Classical PCA fails in this case due to highly sensitive to sparse errors of high magnitude. This question can be considered as a low-rank matrix recovery problem, in which it aims to recover a low-rank matrix L from the corrupted data matrix $M = L + S$. Unlike the small noise term in classical PCA, the entries in S may have arbitrarily large magnitude as well as sparse structure. Candès et al. (2011) proposed a Robust Principal Component Analysis (RPCA) method to solve this problem. Clearly, obtaining the exact solution is NP-hard for arbitrary sparse and low-rank matrix. Chandrasekaran et al. (2009) proved that with some suitable assumptions, the recovery for both matrix components was available. Their approach reduced the original problem to solve a semi-definite programming (SDP) problem. It was also shown in Chandrasekaran et al. (2011) that the recovery can be achieved via convex relaxation where a L_1-norm and a nuclear norm are used to induce sparse and low-rank structures, respectively. Afterwards, more and more researchers kept working on the problem of sparse and low-rank matrix decomposition (SLRMD). Lots of applications also applied to different areas including model selection in statistics, system identification in engineering and matrix rigidity in computer science.

In term of getting practical results, the applications from SVD and PCA are usually interchangeable. However, as the size of the data grows larger and larger, statistical inference with SVD and PCA turns to be very hard without assumptions of strong structure in the data. For instance, in a large noisy matrix, the significant structure is open concentrated in a small subset, and the noise can overwhelm the signal to a high level of degree that estimates using SVD or PCA loadings could be far away from the truth. And due to the accumulation of noise from the majority of structureless cells, the classical algorithm will produce estimates with large variances (Shabalin and Nobel, 2013). Many other research also pointed out the similar issues that in very high dimensional settings, classical SVD and PCA may have poor statistical properties (Paul (2007), Johnstone and Lu (2012)). Shen and Huang (2008) used the connection of PCA and SVD of the data matrix and extracted the PCs through solving a low rank matrix approximation problem. Similarly, in high dimensional matrix decomposition problems, there usually involve assumptions such as low rank and sparsity. By imposing sparsity restriction on SVD, it may shave of the noise cells and therefore dig out the "checkerboard" patterns representing biclustering structure (Lee et al. (2010) and Sill et al. (2011)). More details to this filed will be introduced later. Moreover, solving sparse solutions will gain computational benefits since the time cost of computing numerically precise SVD or PCA solutions is huge. There are more references dealing with sparse PCA as well as SVD solutions

on high-dimensional data, including Lu (2002), Zou et al. (2006), Shen et al. (2013), Witten et al. (2009), Paul and Johnstone (2012), Huang et al. (2012), Ma et al. (2013), and Allen et al. (2014).

In this chapter, we will focus on the last situation above to deal with low rank sparsity approximation on a matrix. Consider a data matrix $A \in \mathbb{R}^{m\times n}$, and we are interested in learning a multi-layer representation of this matrix,

$$A \approx \sum_{i=1}^{r} d_i \mathbf{u}_i \mathbf{v}_i^t, \tag{2.1.1}$$

where $\mathbf{u}_i$'s are m-by-1, $\mathbf{v}_i$'s are n-by-1 vectors, $d_i > 0$ and $r \in \mathbb{Z}^+$ usually with a small value. Such a problem may arise in bioinformatics, where the rows correspond to genes, the columns correspond to samples and A_{ij}'s are the measured expression levels for the i-th gene in j-th sample. It may also arise in text mining, where the rows may correspond to documents, the columns correspond to words, and A_{ij}'s are the word frequencies appearing in each document. Although both m and n may be large, we assume the intrinsic structure of the data matrix is of low-dimension as described in (2.1.1).

Suppose the number of layers r is fixed. Then it is natural to solve $\mathbf{u}_i$'s and $\mathbf{v}_i$'s by minimizing the approximation error as follows:

$$\min_{\text{rank}(X_{m\times n})=r} \|A - X\|_F^2, \quad X = \sum_{i=1}^{r} d_i \mathbf{u}_i \mathbf{v}_i^t, \tag{2.1.2}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. It is known from Horn and Johnson (1985) that the solution is given by the top r components from SVD of A, namely,

$$A = UDV^t = \sum_{i=1}^{p} d_i \mathbf{u}_i \mathbf{v}_i^t,$$

where $p \leq \min(n, m)$ is the rank of A, $U_{m\times p} = [\mathbf{u}_1, \mathbf{u}_2, ...\mathbf{u}_p]_{m\times p}$ and $V_{n\times p} = [\mathbf{v}_1, \mathbf{v}_2, ...\mathbf{v}_p]$ are orthogonal matrices, i.e., $U^tU = V^tV = I_p$, and $D_{p\times p}$ is a diagonal matrix with K positive values (i.e., the singular values), $d_1 \geq d_2 \geq ... \geq d_K > 0$, and $d_{K+1} = \cdots = d_p = 0$. In others words, the first r components from the SVD decomposition give the best approximation of A in the sense of the Frobenius norm.

Of interest nowadays is the low-rank approximation with $\mathbf{u}_i$'s and $\mathbf{v}_i$'s being sparse. The sparsity

here leads to a block structure for each layer $d_i \mathbf{u}_i \mathbf{v}_i^t$, and it's due to that only a subset of the columns and rows have non-zero estimations. Identifying such block structures is appealing for many real applications. For instance, in gene expression data, such structures may reveal special pathways that are only present in some sub-populations, or for the word frequency data, such structures may reveal associations between words that are only present in some topics.

We start with a review on two existing algorithms on sparse rank-one approximation which are most relevant to our work. Both algorithms can be used to extract more than one layers by applying the methods sequentially. We will also discuss the limitation of such a sequential approach later.

- *Biclustering via Sparse Singular Value Decomposition*

  Biclustering is an important approach in DNA microararry data. The word refers to the "simultaneously clustering" of both rows and columns of a data matrix. Using biclustering method, we can interpret gene features through complicated expression patterns under different conditions of samples. This type of analysis was first introduced by Hartigan (1972). After that, Cheng and Church (2000) brought this concept to gene expression data analysis, and Ben-Dor et al. (1999), Tanay et al. (2002) and Abdullah and Hussain (2006) developed the approach by linking to graph based models. Lazzeroni and Owen (2002) defined what they called plaid model to decompose the data into multiple layers through analysis of variance, which correspond to biclusters. Further on, plenty of research explored singular value decomposition for visualization of gene expression data (Kluger et al. (2003), Liu et al. (2004)). Based on SVD, bicluster problem on gene data matrix was transformed into two global clustering problems. After biclustering, there were distinctive "checkerboard" patterns in data representing which genes are functionally related.

  As high dimensionality rapidly becomes a common feature for the data, it offers additional statistical challenges in high-dimension or even high-dimension and low sample size setting where relying on classical analysis may not be suitable. Lee et al. (2010) introduced sparse singular value decomposition (SSVD) as a tool for biclustering in the new data environment. It sought a low-rank, "checkerboard" structured matrix approximation to data matrices. To obtain sparse loadings, they imposed sparsity-inducing penalties on both $\mathbf{u}$ and $\mathbf{v}$ when dealing

with the first layer estimation,

$$\min_{\mathbf{u},\mathbf{v},d} \left(\|A - d\mathbf{u}\mathbf{v}^t\|_F^2 + \lambda_1 d\|\mathbf{u}\|_1 + \lambda_2 d\|\mathbf{v}\|_1 \right) \tag{2.1.3}$$
$$\text{subject to } \|\mathbf{u}\|_2 = 1,\ \|\mathbf{v}\|_2 = 1,\ d > 0$$

where λ_1 and λ_2 are tuning parameters that balance the trade of between estimation accuracy and sparsity level, and $\|\cdot\|_1$ and $\|\cdot\|_2$ denote the L_1 and L_2 norm, respectively.

- *A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis*

  Witten et al. (2009) also defined a rank-one approximation with sparse constraints as a penalized term in the objective function to perform matrix decomposition. The decomposition built upon a variety of existing matrix decompositions, such as the SVD, the NMF (Lee and Seung (1999), Lee and Seung (2001)) and the plaid model (Lazzeroni and Owen (2002)). Different from the objective function in Lee et al. (2010), they put on a penalty directly on $\mathbf{u}$ and $\mathbf{v}$, but not related to the scalar d,

$$\min_{\mathbf{u},\mathbf{v},d} \|A - d\mathbf{u}\mathbf{v}^t\|_F^2 \tag{2.1.4}$$
$$\text{subject to } \|\mathbf{u}\|_2 = 1,\ \|\mathbf{v}\|_2 = 1,\ d > 0,\ \|\mathbf{u}\|_1 \le c_1,\ \|\mathbf{v}\|_1 \le c_2.$$

  Here, c_1 and c_2 are tuning parameters equivalent to λ_1 and λ_2 in (2.1.3). Then they developed an algorithm that iteratively updated $\mathbf{u}$, $\mathbf{v}$ and d until convergence. The formula for updating $\mathbf{u}$ given $\mathbf{v}$ is

$$\mathbf{u} = \frac{S(A\mathbf{v}, \Delta)}{\|S(A\mathbf{v}, \Delta)\|_2}, \tag{2.1.5}$$

  where $S(a, c)$ is the soft-thresholding operator, i.e.,

$$S(a, c) = \text{sgn}(a)(|a| - c)_+,$$

  where x_+ is equal to x if $x > 0$ and 0 if $x \le 0$. The way to determine Δ in (2.1.5) is that $\Delta = 0$ if it makes $\|\mathbf{u}\|_1 \le c_1$, otherwise Δ is chosen to be a positive number such that $\|\mathbf{u}\|_1 = c_1$.

Moreover, this penalized matrix decomposition (PMD) method also unified the regularized low-rank matrix approximation approach of Shen and Huang (2008) with the maximum variance criterion from Jolliffe et al. (2003) and the SPCA method from Zou et al. (2006). Jolliffe et al. (2003) pointed out a modified principal component technique based on the LASSO (SCoTLASS) which was the most simple and natural way to define the notion of sparse principal components. Unfortunately, the objective function is not convex which leads difficulties in computations. And the special case in PMD with L_1 constraint on columns but not in rows yields a more efficient solution to SCoTLASS for finding the first sparse principal component. In addition, when the methodology is applied to a cross-products matrix, it results in the same method for penalized canonical correlation analysis (CCA) (Parkhomenko et al. (2009)).

However, a drawback of PMD algorithm is that there is no close form solution for Δ. Instead, the appropriate value for Δ such that $\|\mathbf{u}\|_1 = c_1$ is obtained through the binary search algorithm (Cormen et al., 1990). Part of the algorithm we propose in this chapter is a different method to solve the same objective function in (2.1.4), and the calculation process is much easier with closed form solutions related to different values of tuning parameters.

2.1.1 Limitations

With the above mentioned sparse matrix decomposition methods for one layer, it is often suggested to further decompose a matrix into multiple layers by sequently applying those methods. However, the sequential decomposition method may not acquire estimation on each layer correspond to the actual sparse structure. We provide a toy example here to give a better explanation. Suppose we generate a two-layers data matrix A through

$$A = d_1\mathbf{u}_1\mathbf{v}_1^t + d_2\mathbf{u}_2\mathbf{v}_2^t + E,$$

where $E = \{e_{ij}\}$ with $e_{ij} \overset{\text{iid}}{\sim} \mathsf{N}(0,1)$, $\mathbf{u}_{1:2}$ and $\mathbf{v}_{1:2}$ are sparse $\mathbb{R}^{10}$ vectors taking values as follows,

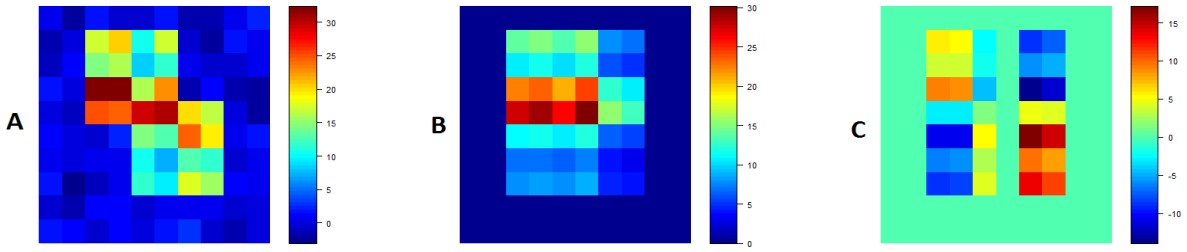

Figure 2.1: Two layer sparse matrix decomposition by sequential approach. A is the two-layers toy data, B is the first layer explored by PMD sparse matrix decomposition algorithm, C is the second layer sequentially solved by PMD algorithm.

$$
\begin{aligned}
\mathbf{u}_1 &= (0, 2, 1.5, 3, 2.5, 0, 0, 0, 0, 0)^t, \\
\mathbf{u}_2 &= (0, 0, 0, 0, 2.5, 2.5, 1.5, 2, 0, 0)^t, \\
\mathbf{v}_1 &= (0, 0, 2.5, 2.5, 1.5, 2, 0, 0, 0, 0)^t, \\
\mathbf{v}_2 &= (0, 0, 0, 0, 2, 1.5, 3, 2.5, 0, 0)^t,
\end{aligned}
$$

and $d_1 = 4$ and $d_2 = 3$. If we use 256 pixels to plot the matrix, the top left piece (A) in Figure 2.1 shows the matrix structure.

If we apply PMD method in Witten et al. (2009) to sequentially solve for the top two layers decomposition of A, the estimation result is provided as part B and C in Figure 2.1. Although the estimated layers given by PMD are sparse which are different from the results solved by SVD, and the summation of the first two layers is also pretty closed to the true data, yet the two layers separately do not catch true structure of A. Results are similar if we apply the algorithms from Lee et al. (2010). The culprit is the sequential approach. Without the sparsity constraint, the rank-r approximation problem in (2.1.2) can be solved sequentially through a rank-one approximation algorithm. That is, if the best rank-one approximation of A is $d_1\mathbf{u}_1\mathbf{v}_1^t$, then $d_2\mathbf{u}_2\mathbf{v}_2^t$ is the best rank-one approximation of the residual matrix $A - d_1\mathbf{u}_1\mathbf{v}_1^t$, and so on. However, this result will not hold any more with the sparsity constraint. Therefore, another purpose of this section is to propose an algorithm that estimate the multiple layers simultaneously, instead of sequentially.

2.2 Multi Layers Sparse Decomposition

2.2.1 Objective Function

Let's propose the problem in the following way. Given a $m \times n$ matrix A, we want to find a r layers sparse decomposition with each layer a rank-one matrix, i.e.,

$$A_{m\times n} \approx \sum_{i=1}^{r} d_i \mathbf{u}_i \mathbf{v}_i^t,$$

where $\mathbf{u}_i$=$(u_{i1}, ..., u_{im})$ is a L_2 norm one vector in $\mathbb{R}^m$, $\mathbf{v}_i$=$(v_{i1}, ..., v_{in})$ is a L_2 norm one vector in $\mathbb{R}^n$ and d_i is a positive scalar for $i = 1, 2, ..., r$. The "sparse" means that for $\mathbf{u}_i$ and $\mathbf{v}_i$, most of their loadings equal to zero. We can obtain this property by imposing penalties on $\mathbf{u}_i's$ and $\mathbf{v}_i's$ when solving the estimations. It normally chooses the common L_1 penalty, such that the objective function can be formulated as

$$(\widetilde{\mathbf{u}}_{1:r}, \widetilde{\mathbf{v}}_{1:r}, \widetilde{d}_{1:r}) = \arg\min \left(\|A - \sum_{i=1}^{r} d_i \mathbf{u}_i \mathbf{v}_i^t\|_F^2 + \sum_{i=1}^{r} \lambda_{i1} \|\mathbf{u}_i\|_1 + \sum_{i=1}^{r} \lambda_{i2} \|\mathbf{v}_i\|_1 \right), \tag{2.2.1}$$
$$\text{subject to } \|\mathbf{u}_i\|_2 = 1,\ \|\mathbf{v}_i\|_2 = 1,\ i = 1:r,\ d \geq 0.$$

where λ_{i1} and λ_{i2} are all tuning parameters for $i = 1, 2, ..., r$.

When $r = 1$, this objective function is different from (2.1.3) as what is used in Lee et al. (2010), since we put penalty directly on $\mathbf{u}$ and $\mathbf{v}$ without related to the scalar d. And it is equivalent to the idea from Witten et al. (2009) as listed in (2.1.4), since there is a one to one mapping between λ_1 and c_1, λ_2 and c_2. By using this objective function, we claim that it can be solved in a different way which there is no need to use the binary search algorithm and thus it's solution is easier to explain and understand.

2.2.2 A Generic Tool

The r-layers matrix decomposition problem in (2.2.1) is not easy to solve directly, since it involves $3r$ parameters. We choose to iteratively update only one of them at each step, and repeat the procedure until all the parameters converge, i.e., at a particular step, expect for the ith layer, estimations of other layers are assumed to be fixed; and within the ith layer, only one parameter among $\mathbf{u}_i$, $\mathbf{v}_i$ and d_i is going to be estimated given fixed values of the other two. Therefore, we can transform the

objective function (2.2.1) into three generic optimization problems,

$$\widetilde{\mathbf{v}}_i = \arg\min_{\|\mathbf{v}_i\|_2=1} \left(\|Y - d_i \mathbf{u}_i \mathbf{v}_i^t\|_F^2 + \lambda_2 \|\mathbf{v}_i\|_1 \right) \tag{2.2.2}$$

$$\widetilde{\mathbf{u}}_i = \arg\min_{\|\mathbf{u}_i\|_2=1} \left(\|Y - d_i \mathbf{u}_i \mathbf{v}_i^t\|_F^2 + \lambda_1 \|\mathbf{u}_i\|_1 \right) \tag{2.2.3}$$

$$\widetilde{d}_i = \arg\min \|Y - d_i \mathbf{u}_i \mathbf{v}_i^t\|_F^2 \tag{2.2.4}$$

where $Y = A - \sum_{j \neq i} d_j \mathbf{u}_j \mathbf{v}_j^t$ is the residual for estimating the ith layer. Also as we explained, in (2.2.2), d_i and $\mathbf{u}_i$ are given, in (2.2.3), d_i and $\mathbf{v}_i$ are given, in (2.2.4), $\mathbf{u}_i$ and $\mathbf{v}_i$ are given respectively. The first two problems can be considered as the same one and without loss of generality, we provide the solution of (2.2.2) in the following theorem.

Theorem 2.1. *Given $Y \in \mathbb{R}^{m \times n}$, $\boldsymbol{u} = \boldsymbol{u}_0 \in \mathbb{R}^m$ with $\|\mathbf{u}_0\|^2 = 1$ and $d = d_0 > 0$, let $\boldsymbol{b} = Y^t \boldsymbol{u}_0 = (b_1, ..., b_n) \in \mathbb{R}^n$, then the solution of*

$$\widetilde{\boldsymbol{v}} = \arg\min_{\|\boldsymbol{v}\|_2=1} \left(\|Y - d_0 \boldsymbol{u}_0 \boldsymbol{v}^t\|_F^2 + \lambda_1 \|\boldsymbol{v}\|_1 \right) \tag{2.2.5}$$

can be written as

- *If $\lambda_1 - 2d_0|b_j| \geq 0$ for $j = 1, 2, ..., n$, then $v_k = 1$, $v_j = 0$ for all $j \neq k$, where $k = \arg\max_k |b_k|$.*
- *If $\exists\ j$ such that $\lambda_1 - 2d_0|b_j| < 0$, let $\mathcal{H} = \{h : \lambda_1 - 2d_0|b_h| < 0\}$, then $v_j = 0$ for $j \notin \mathcal{H}$ and for $j \in \mathcal{H}$, it satisfies*

$$\begin{cases} \frac{\left|\lambda_1 - 2d_0|b_j|\right|}{|v_j|} = c, \text{ for a constant } c \neq 0 \\ \sum_{j \in \mathcal{H}} v_j^2 = 1 \\ sgn(v_j) = sgn(b_j) \end{cases}$$

To combine all the constrains, we have,

$$\widetilde{\boldsymbol{v}} = \frac{S(Y^t \boldsymbol{u}_0, \frac{\lambda_1}{2d_0})}{\|S(Y^t \boldsymbol{u}_0, \frac{\lambda_1}{2d_0})\|_2}.$$

We provide the proof in the Appendix B. Here is an intuition of the Lemma **??**. The problem (2.2.5) can be rewritten as

$$\widetilde{\mathbf{v}} = \arg \min_{\|\mathbf{v}\|^2=1} \|\frac{Y^t\mathbf{u}_0}{d_0} - \mathbf{v}\|_F^2 + \frac{\lambda_1}{d_0^2}\|\mathbf{v}\|_1. \tag{2.2.6}$$

Without the L_2 norm one restriction on $\mathbf{v}$, it's a Lasso problem with a solution

$$\widetilde{\mathbf{v}} = S(\frac{Y^t\mathbf{u}_0}{d_0}, \frac{\lambda_1}{2d_0^2}). \tag{2.2.7}$$

By L_2 norm one restriction, we should consider two cases separately, i.e., the positive and negative sign of $\left(|Y^t\mathbf{u}_0| - \frac{\lambda_1}{d_0}\right)$. If all $|Y^t\mathbf{u}_0| - \frac{\lambda_2}{2d_0} < 0$, the lasso solution gives $S(\frac{Y^t\mathbf{u}_0}{d_0}, \frac{\lambda_1}{2d_0^2}) = \mathbf{0_m}$, but with $\|\mathbf{v}\|_2 = 1$, we should pick one direction (with largest $|Y^t\mathbf{u}_0|$) with estimation equal to 1, and the remaining coefficients all equal to 0. If there are some j such that $|Y^t\mathbf{u}_0| - \frac{\lambda_2}{2d_0} \geq 0$, we need to set all these directions non-zero but also normalize $\mathbf{v}$ to be L_2 norm one. This thresholding rule turns out to the Lasso solution (2.2.7) as well as a normalization process afterwards with L_2 norm one, i.e.,

$$\widetilde{\mathbf{v}} = \frac{S(\frac{Y^t\mathbf{u}_0}{d_0}, \frac{\lambda_1}{2d_0^2})}{\|S(\frac{Y^t\mathbf{u}_0}{d_0}, \frac{\lambda_1}{2d_0^2})\|_2} = \frac{S(Y^t\mathbf{u}_0, \frac{\lambda_1}{2d_0})}{\|S(Y^t\mathbf{u}_0, \frac{\lambda_1}{2d_0})\|_2}.$$

For the remaining optimization problem (2.2.4), if we are given $\mathbf{u} = \mathbf{u}_0$=$(u_{01}, ..., u_{0m})$ and $\mathbf{v} = \mathbf{v}_0$ = $(v_{01}, ..., v_{0n})$,

$$\begin{aligned}
\widetilde{d} &= \arg \min_{d>=0} \|Y - d\mathbf{u}_0\mathbf{v}_0^t\|_F^2 \\
&= \arg \min_d \sum_i \sum_j (y_{ij} - du_{0i}v_{0j})^2 \\
&= \arg \min_d \sum_i \sum_j y_{ij}^2 - 2d \sum_i \sum_j y_{ij}u_{0i}v_{0j} + d^2 \sum_i \sum_j u_{0i}^2 v_{0j}^2 \\
&= \arg \min_d -2d \sum_i \sum_j y_{ij}u_{0i}v_{0j} + d^2.
\end{aligned}$$

Taking the first derivative on term $-2d\sum_i \sum_j y_{ij}u_{0i}v_{0j} + d^2$ and setting it to 0, we have

$$2\widetilde{d} - 2\sum_i \sum_j y_{ij}u_{0i}v_{0j} = 0$$

and therefore,

$$\widetilde{d} = \sum_i \sum_j y_{ij} u_{0i} v_{0j}.$$

It's easy to find out that this solution can be considered as an OLS estimator on $\widetilde{d}$ of a linear model $Y_{mn} = dX_{mn}$, where Y_{mn} is a $\mathbb{R}^{mn}$ vector including all the elements of Y, and X_{mn} contains all the elements of $\mathbf{u}_0\mathbf{v}_0^t$.

2.2.3 Choice of Tuning

Since $\mathbf{u}_i$ and $\mathbf{v}_i$ are all sparse vectors, we can define the degree of sparsity of the vector $\mathbf{v}$ as the number of non-zero elements in $\mathbf{v}$, written as $\mathrm{df}_{\mathbf{v}}$. Therefore, the range of $\mathrm{df}_{\mathbf{v}}$ is $1, 2, .., n$. For the generic problem (2.2.2)

$$\widetilde{\mathbf{v}}_i = \arg \min_{\|\mathbf{v}_i\|_2=1} \|Y - d_i \mathbf{u}_i \mathbf{v}_i^t\|_F^2 + \lambda_1 \|\mathbf{v}_i\|_1,$$

the tuning parameter λ_1 determines the degree of sparsity of $\mathbf{v}_i$. Instead of tuning λ_1, we turn to tune $\mathrm{df}_{\mathbf{v}_i}$ due to a smaller pool of candidate values. However, given the sparsity degree $\mathrm{df}_{\mathbf{v}_i}$, we still need the value of λ_1 to solve $\mathbf{v}_i$ based on the solution in Lemma **??**. And for the same sparsity degree $\mathbf{v}_i$, it corresponds to a range of values of λ_1, that is even if the sparsity degree of $\mathbf{v}_i$ is the same, the coefficients estimation on non-zero directions of $\mathbf{v}_i$ can be different with respect to different λ_1. Therefore, we set up the rule that for the same sparsity degree of $\mathbf{v}_i$, we always pick the smallest λ_1 as the input to solve $\mathbf{v}_i$, and that will penalize less on the sparsity and make estimations more accurate.

Lee et al. (2010) embeded a BIC criteria into their SSVD algorithm to solve the model fitting and tune the penalty parameters simultaneously. We borrow their idea to our method, and tune sparsity degree of $\mathbf{u}_i$ and $\mathbf{v}_i$ when solving each generic problem. However, we change their BIC criteria into a traditional form. Taking (2.2.2) for instance, given d_i and $\mathbf{u}_i$, we define the BIC for the solution of $\widetilde{\mathbf{v}}_i$ with sparse degree $\mathrm{df}_{\widetilde{\mathbf{v}}_i}$ as

$$\begin{aligned}
\mathrm{BIC}(\mathrm{df}_{\widetilde{\mathbf{v}}_i}) &= -2\mathrm{loglike} + \mathrm{df}_{\widetilde{\mathbf{v}}_i} \log(mn) \\
&= -2\log(\frac{1}{\sqrt{2\pi\widehat{\sigma}^2}})^{mn} + 2\frac{\|Y - d_i \mathbf{u}_i \widetilde{\mathbf{v}}_i^t\|^2}{2\widehat{\sigma}^2} + \mathrm{df}_{\widetilde{\mathbf{v}}_i} \log(mn) \\
&= \mathrm{constant} + mn\log(\widehat{\sigma}^2) + \mathrm{df}_{\widetilde{\mathbf{v}}_i} \log(mn), \qquad (2.2.8)
\end{aligned}$$

where $\widehat{\sigma}^2 = \frac{\|Y - d_i \mathbf{u}_i \widetilde{\mathbf{v}}_i^t\|^2}{mn}$.

Therefore, the tuning flow for sparsity of $\mathbf{v}_i$ can be summarized as given a sparsity degree $\alpha \in \{1, 2, ..., n\}$, we choose the smallest λ_1 so that the solution $\widetilde{\mathbf{v}}_i$ in (2.2.2) satisfies $\mathrm{df}_{\widetilde{\mathbf{v}}_i} = \alpha$. We calculate different $\mathrm{BIC}(\mathrm{df}_{\widetilde{\mathbf{v}}_i})$ corresponding to different values of α and get the solution $\widetilde{\mathbf{v}}_i$ with the smallest BIC. Similarly, we can define the BIC criteria for $\mathrm{df}_{\widetilde{\mathbf{u}}_i}$ for generic problem (2.2.3) and apply the same tuning procedure to receive the estimation of $\mathbf{u}_i$.

Now, we introduce our algorithm for updating the i-th layer parameters $(d_i, \mathbf{u}_i, \mathbf{v}_i)$ by nesting a BIC tuning procedure for $\mathrm{df}_{\mathbf{v}_i}$ and $\mathrm{df}_{\mathbf{u}_i}$ (λ_1 and λ_2).

Algorithm 2.1 i-th Layer Updating with Tuning Embedding

1: **Initialization** Given $r-1$ layers estimations $(d_j, \mathbf{u}_j, \mathbf{v}_j)$, $j = 1, ..i-1, i+1, .., r$, let $Y = A - \sum_{j \neq i} d_j \mathbf{u}_j \mathbf{v}_j^t$, and the i-th layer estimations from previous step are $(d_i^{\mathrm{old}}, \mathbf{u}_i^{\mathrm{old}}, \mathbf{v}_i^{\mathrm{old}})$.

2: **Update i-th layer**

- **Step1:** Given $\mathbf{u}_i^{\mathrm{old}}$ and d_i^{old}, update

$$\mathbf{v}_i^{\mathrm{new}} = \arg \min_{\|\mathbf{v}\|_2 = 1} \|Y - d_i^{\mathrm{old}} \mathbf{u}_i^{\mathrm{old}} \mathbf{v}^t\|_F^2 + \lambda_{\mathbf{v}} \|\mathbf{v}\|_1,$$

  where $\lambda_{\mathbf{v}}$ is the smallest value of λ_1 corresponding to the sparsity degree $\mathrm{df}_{\mathbf{v}}$ such that $\mathrm{df}_{\mathbf{v}}$ minimizes the BIC criteria $\mathrm{BIC}(\mathrm{df}_{\mathbf{v}})$ defined as (2.2.8).

- **Step2:** Given $\mathbf{v}_i^{\mathrm{new}}$ and d_i^{old}, update

$$\mathbf{u}_i^{\mathrm{new}} = \arg \min_{\|\mathbf{u}\|_2 = 1} \|Y^t - d_i^{\mathrm{old}} \mathbf{v}_i^{\mathrm{new}} \mathbf{u}^t\|_F^2 + \lambda_{\mathbf{u}} \|\mathbf{u}\|_1,$$

  where $\lambda_{\mathbf{u}}$ is the smallest value of λ_2 corresponding to the sparsity degree $\mathrm{df}_{\mathbf{u}}$ such that $\mathrm{df}_{\mathbf{u}}$ minimizes the BIC criteria $\mathrm{BIC}(\mathrm{df}_{\mathbf{u}})$.

- **Step3:** Given $\mathbf{u}_i^{\mathrm{new}}$ and $\mathbf{v}_i^{\mathrm{new}}$, update d_i^{new} as

$$d_i^{\mathrm{new}} = \sum_i \sum_j y_{ij} u_i^{\mathrm{new}} v_j^{\mathrm{new}}.$$

2.2.4 Algorithm

With all preparations in previous sections, we can propose our final algorithm. Without loss of generality, let's consider the case when $r = 2$, and the data matrix is supposed to be decomposed as

$$A_{m\times n} \approx d_1\mathbf{u}_1\mathbf{v}_1^t + d_2\mathbf{u}_2\mathbf{v}_2^t,$$

where $\mathbf{u}_{1:2}$ and $\mathbf{v}_{1:2}$ are all sparse vectors in $\mathbb{R}^m$ and $\mathbb{R}^n$ respectively. The following algorithm solves this two-layer sparse decomposition estimation simultaneously, and embeds the parameters tuning in the iterative procedure.

Algorithm 2.2 Two-Layer Sparse Decomposition with Tuning Embedding

1: **Initialization**: Apply singular value decomposition on A, get the first two SVD triplets (d_1, $\mathbf{u}_1$, $\mathbf{v}_1$) and (d_2, $\mathbf{u}_2$, $\mathbf{v}_2$), set (d_1^{old}, $\mathbf{u}_1^{\text{old}}$, $\mathbf{v}_1^{\text{old}}$)=($d_1$, $\mathbf{u}_1$, $\mathbf{v}_1$) and (d_2^{old}, $\mathbf{u}_2^{\text{old}}$, $\mathbf{v}_2^{\text{old}}$)=($d_2$, $\mathbf{u}_2$, $\mathbf{v}_2$).

2: **Update**: Update each layer's parameter with tuning embedding.

- Given the 2nd layer estimation ($d_2^{\text{old}}, \mathbf{u}_2^{\text{old}}$, $\mathbf{v}_2^{\text{old}}$), calculate the residual $Y = A - d_2^{\text{old}}\mathbf{u}_2^{\text{old}}(\mathbf{v}_2^{\text{old}})^t$, apply algorithm 2.1 for the 1st layer and get an updated estimation ($d_1^{\text{new}}, \mathbf{u}_1^{\text{new}}$, $\mathbf{v}_1^{\text{new}}$).
- Given the 1st layer estimation ($d_1^{\text{new}}, \mathbf{u}_1^{\text{new}}$, $\mathbf{v}_1^{\text{new}}$), calculate the residual $Y = A - d_1^{\text{new}}\mathbf{u}_1^{\text{new}}(\mathbf{v}_1^{\text{new}})^t$, apply algorithm 2.1 for the 2nd layer and get an updated estimation ($d_2^{\text{new}}, \mathbf{u}_2^{\text{new}}$, $\mathbf{v}_2^{\text{new}}$).
- Refresh ($d_2^{\text{old}}, \mathbf{u}_2^{\text{old}}$, $\mathbf{v}_2^{\text{old}}$) = ($d_2^{\text{new}}, \mathbf{u}_2^{\text{new}}$, $\mathbf{v}_2^{\text{new}}$) for another round of update.

3: **Repetition**: Repeat the three procedures in **Update** until two layers estimations $d_1^{\text{new}}\mathbf{u}_1^{\text{new}}(\mathbf{v}_1^{\text{new}})^t$ and $d_2^{\text{new}}\mathbf{u}_2^{\text{new}}(\mathbf{v}_2^{\text{new}})^t$ converge.

2.3 Refitting to Control Bias

In our algorithm, these is a penalty term in the objective function, even if we could solve the problem with the true sparsity structure for each layer, the coefficient estimations may still not be accurate. This issue could affect the overall matrix estimation. Therefore, we propose a refitting procedure where we re-estimate each layer's nonzero coefficients given the sparsity structures to control the estimation bias.

Given a r-layer sparse decomposition of A solved by our algorithm,

$$A \approx \sum_{i=1}^{r} \widetilde{d}_i \widetilde{\mathbf{u}}_i \widetilde{\mathbf{v}}_i^t.$$

Let $\mathcal{A}_{\widetilde{\mathbf{u}}_i}$ be a set of active coefficient index of $\widetilde{\mathbf{u}}_i$ and $\mathcal{A}_{\widetilde{\mathbf{v}}_i}$ be a set of active coefficient index of $\widetilde{\mathbf{v}}_i$. We also define $\text{Card}(\mathcal{A}_{\widetilde{\mathbf{u}}_i}) = p$ and $\text{Card}(\mathcal{A}_{\widetilde{\mathbf{v}}_i}) = q$ be the number of index in $\mathcal{A}_{\widetilde{\mathbf{u}}_i}$ and $\mathcal{A}_{\widetilde{\mathbf{v}}_i}$ respectively. Then a refitting process for the i-th layer given other layers has the following objective function,

$$(\widehat{\mathbf{u}}_i, \widehat{\mathbf{v}}_i, \widehat{d}_i) = \arg \min_{\mathbf{u}_i, \mathbf{v}_i, d_i} \|Y - d_i \mathbf{u}_i \mathbf{v}_i^t\|_F^2 \tag{2.3.1}$$

$$\text{subject to } \|\mathbf{u}_i\|_2 = 1,\ \|\mathbf{v}_i\|_2 = 1, u_{ij} = 0 \text{ for } j \in \mathcal{A}_{\mathbf{u}_i}, v_{ij} = 0, \text{for } j \in \mathcal{A}_{\mathbf{v}_i},$$

where $Y = A - \sum_{j \neq i} \widetilde{d}_j \widetilde{\mathbf{u}}_j \widetilde{\mathbf{v}}_j^t$.

Without lost of generality, we can rearrange the elements order in $\widetilde{\mathbf{u}}_i$ and $\widetilde{\mathbf{v}}_i$ such that

$$\mathbf{u}_i = (\bar{\mathbf{u}}_i, \mathbf{0}),\ \mathbf{v}_i = (\bar{\mathbf{v}}_i, \mathbf{0})$$

where $\bar{\mathbf{u}}_i \in \mathbb{R}^p$ and $\bar{\mathbf{v}}_i \in \mathbb{R}^q$ contain all the nonzero elements in $\widetilde{\mathbf{u}}_i$ and $\widetilde{\mathbf{v}}_i$ respectively.

Then we can divide Y and $\widetilde{d}_i \widetilde{\mathbf{u}}_i \widetilde{\mathbf{v}}_i^t$ into four blocks,

$$\left[\begin{array}{c:c} Y_{11} & Y_{12} \\ \hdashline Y_{21} & Y_{22} \end{array}\right] \qquad \left[\begin{array}{c:c} d_i \bar{\mathbf{u}}_i \bar{\mathbf{v}}_i^t & 0 \\ \hdashline 0 & 0 \end{array}\right]$$

and we only need to deal with the upper left $p \times q$ sub-matrix to solve the nonzero elements of $\widehat{\mathbf{u}}_i$ and $\widehat{\mathbf{v}}_i$ in (2.3.1), i..e,

$$(\widehat{\bar{\mathbf{u}}}_i, \widehat{\bar{\mathbf{v}}}_i, \widehat{d}_i) = \arg \min_{\bar{\mathbf{u}}_i, \bar{\mathbf{v}}_i, d_i} \|Y_{11} - d_i \bar{\mathbf{u}}_i \bar{\mathbf{v}}_i^t\|_F^2 \tag{2.3.2}$$

$$\text{subject to } \|\bar{\mathbf{v}}_i\|_2 = 1,\ \|\bar{\mathbf{u}}_i\|_2 = 1$$

By the property of matrix SVD, the solution for (2.3.2) is the first component of SVD on the sub-matrix Y_{11}, where $\widehat{\bar{\mathbf{u}}}_i$ and $\widehat{\bar{\mathbf{v}}}_i$ equal to the left and right-singular vector, respectively.

Yet, we have not mentioned what values we should use for other layers when refitting the i-th layer. If $\bigcap_{i=1}^r \mathcal{A}_{\widetilde{\mathbf{u}}_i} = \emptyset$ and $\bigcap_{i=1}^r \mathcal{A}_{\widetilde{\mathbf{v}}_i} = \emptyset$, it does not matter what values we plug in for other

layers when we calculate $Y = A - \sum_{j \neq i} \widetilde{d}_j \widetilde{\mathbf{u}}_j \widetilde{\mathbf{v}}_j^t$, since all the layers are non-overlapping. Normally, this condition does not hold and other layers estimations will affect the refitting solution for the i-th layer. In this case, we need to iteratively refit each layer by plunging in other layers' refitted estimations and stop updating until every layer converges.

2.4 Choice of Total Layers Number

The last parameter we need to tune is the total number of layers r. A natural idea is to compare BIC which is a trade off between matrix estimation accuracy and parameter degree of complexity. We use the refitted estimation for each layer to calculate such a BIC value. For a r-layer decomposition,

$$\begin{aligned} \text{BIC}_r &= -2\text{loglike} + \left(\sum_{i=1}^{r} (\text{df}_{\widehat{\mathbf{u}}_i} + \text{df}_{\widehat{\mathbf{v}}_i} - 1) \right) \log(mn) \\ &= -2\log\left(\frac{1}{\sqrt{2\pi\widehat{\sigma}^2}}\right)^{mn} + 2\frac{\|A - \sum_{i=1}^{r} \widehat{d}_i \widehat{\mathbf{u}}_i \widetilde{\mathbf{v}}_i^t\|^2}{2\widehat{\sigma}^2} + \left(\sum_{i=1}^{r} (\text{df}_{\widehat{\mathbf{u}}_i} + \text{df}_{\widehat{\mathbf{v}}_i} - 1) \right) \log(mn) \\ &= \text{constant} + mn\log(\widehat{\sigma}^2) + \left(\sum_{i=1}^{r} (\text{df}_{\widehat{\mathbf{u}}_i} + \text{df}_{\widehat{\mathbf{v}}_i} - 1) \right) \log(mn), \end{aligned} \tag{2.4.1}$$

where $\widehat{\sigma}^2 = \frac{\|A - \sum_{i=1}^{r} \widehat{d}_i \widehat{\mathbf{u}}_i \widehat{\mathbf{v}}_i^t\|^2}{mn}$, and $(\widehat{\mathbf{u}}_i, \widehat{\mathbf{v}}_i, \widehat{d}_i)$ is a refitted estimation for i-th layer. To decide the total number of layers, we find the r with the smallest BIC value,

$$\widehat{r} = \arg\min_r \text{BIC}_r$$

2.5 Experiment Study

2.5.1 Toy Example

Let's revisit the toy example used in previous introduction section. The data contains a two-layer structure and it is generated as

$$A = d_1 \mathbf{u}_1 \mathbf{v}_1^t + d_2 \mathbf{u}_2 \mathbf{v}_2^t + E,$$

where $E = \{e_{ij}\}$ with $e_{ij} \overset{\text{iid}}{\sim} \mathsf{N}(0,1)$, $\mathbf{u}_{1:2}$ and $\mathbf{v}_{1:2}$ are sparse $\mathbb{R}^{10}$ vectors taking values as follows,

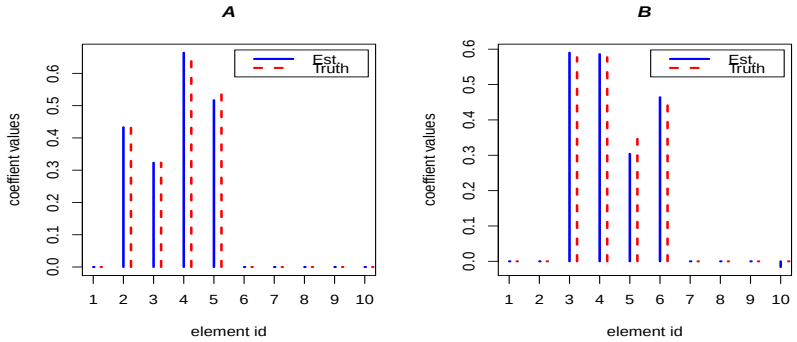

Figure 2.2: Comparison between estimation and truth on $\mathbf{u}_1$ and $\mathbf{v}_1$ for the first scenario: In figure A, blue lines represent each coefficient value for estimation of $\mathbf{u}_1$, red lines represent each coefficient value for the truth $\mathbf{u}_1$; In figure B, blue lines represent each coefficient value for estimation of $\mathbf{v}_1$, red lines represent each coefficient value for the truth $\mathbf{v}_1$. The estimations have the similar sparse structure for both $\mathbf{u}_1$ and $\mathbf{v}_1$.

$$\begin{aligned}
\mathbf{u}_1 &= (0, 2, 1.5, 3, 2.5, 0, 0, 0, 0, 0)^t, \\
\mathbf{u}_2 &= (0, 0, 0, 0, 2.5, 2.5, 1.5, 2, 0, 0)^t, \\
\mathbf{v}_1 &= (0, 0, 2.5, 2.5, 1.5, 2, 0, 0, 0, 0)^t, \\
\mathbf{v}_2 &= (0, 0, 0, 0, 2, 1.5, 3, 2.5, 0, 0)^t,
\end{aligned}$$

and scalars $d_1 = 4$ and $d_2 = 3$. We design three scenarios to test the performance of our method.

In the first scenario, we only apply the algorithm on the first layer data with an error matrix, i.e.,

$$A_1 = d_1 \mathbf{u}_1 \mathbf{v}_1^t + E.$$

Our estimations have $\mathrm{df}_{\widetilde{\mathbf{u}}_1} = 4$ and $\mathrm{df}_{\widetilde{\mathbf{v}}_1} = 5$ with

$$\begin{aligned}
\widetilde{\mathbf{u}}_1 &= (0, 0.433, 0.323, 0.664, 0.517, 0, 0, 0, 0, 0)^t, \\
\widetilde{\mathbf{v}}_1 &= (0, 0, 0.590, 0.586, 0.304, 0.464, 0, 0, 0, -0.016)^t.
\end{aligned}$$

We can see in Figure 2.2, the estimations pretty match the original first layer data sparsity structure. The Frobenius norm of the estimation residual $\|A - \widetilde{d}_1 \widetilde{\mathbf{u}}_1 \widetilde{\mathbf{v}}_1^t\|_F$ is 81.28484.

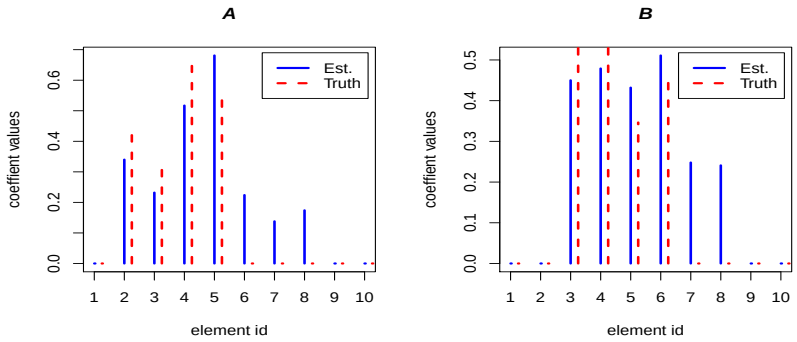

Figure 2.3: Comparison between estimation and truth on $\mathbf{u}_1$ and $\mathbf{v}_1$ for the second scenario: In figure A, blue lines represent each coefficient value for estimation of $\mathbf{u}_1$, red lines represent each coefficient value for the truth $\mathbf{u}_1$; In figure B, blue lines represent each coefficient value for estimation of $\mathbf{v}_1$, red lines represent each coefficient value for the truth $\mathbf{v}_1$. The estimations have a different sparse structure for both $\mathbf{u}_1$ and $\mathbf{v}_1$.

In the second scenario, we apply the algorithm on the whole data matrix A but only retrieve one layer estimation. This will give us the same estimation on the first layer when we want to sequentially receive a two-layer decomposition. The solution have $\mathrm{df}_{\widetilde{\mathbf{u}}_1} = 7$ and $\mathrm{df}_{\widetilde{\mathbf{v}}_1} = 6$, and

$$\begin{aligned} \widetilde{\mathbf{u}}_1 &= (0, 0.340, 0.232, 0.517, 0.681, 0.224, 0.138, 0.174, 0, 0)^t, \\ \widetilde{\mathbf{v}}_1 &= (0, 0, 0.450, 0.479, 0.432, 0.511, 0.248, 0.241, 0, 0)^t. \end{aligned}$$

As what we expect, Figure 2.3 shows that the estimation patterns for $\mathbf{u}_1$ and $\mathbf{v}_1$ do not match the truth. This is the drawback for sequential solving method, i.e., if we only use one layer to estimate the whole data, the estimation try to keep as much information as possible and it will possibly ignore the sparse structure. If we continue to solve the second layer estimation based on the residual $A - \widetilde{d}_1 \widetilde{\mathbf{u}}_1 \widetilde{\mathbf{v}}_i^t$, and combine the nonzero elements in $\widetilde{\mathbf{u}}_1$ and $\widetilde{\mathbf{u}}_2$, we find they are between 2nd to 8th. This matches the overall sparsity structure combining $\mathbf{u}_1$ and $\mathbf{u}_2$. It's similar for $\widetilde{\mathbf{v}}_1$ and $\widetilde{\mathbf{v}}_2$ together, where nonzero elements are reflect the truth from 3rd to 8th.

Lastly, we try to retrieve two-layer estimations simultaneously for A. We finally have $\mathrm{df}_{\widetilde{\mathbf{u}}_1} = 4$,

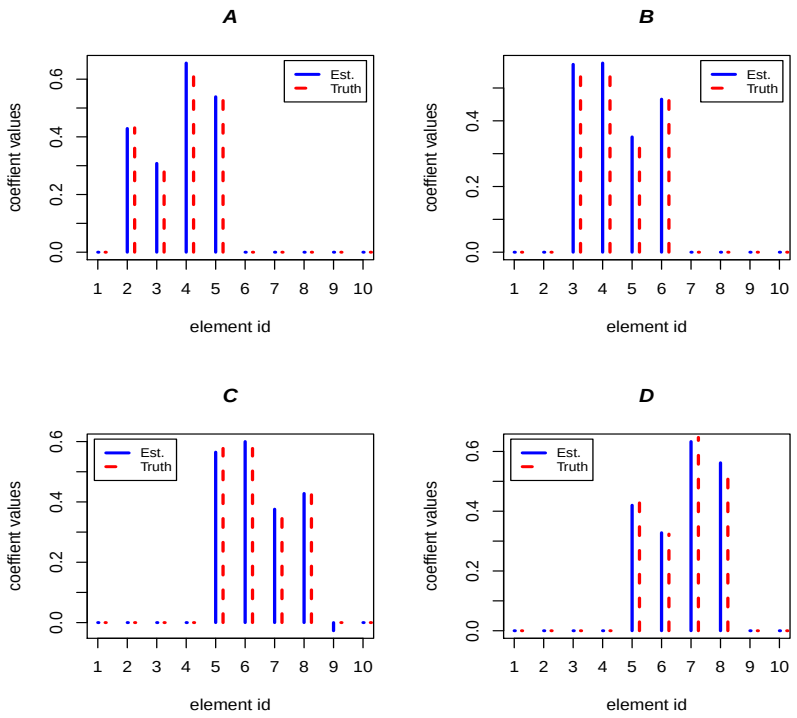

Figure 2.4: Comparison between estimation and truth on $\mathbf{u}_1$, $\mathbf{v}_1$, $\mathbf{u}_2$ and $\mathbf{v}_2$ for the third scenario: In figure A, blue lines represent each coefficient value for estimation of $\mathbf{u}_1$, red lines represent each coefficient value for the truth $\mathbf{u}_1$; In figure B, blue lines represent each coefficient value for estimation of $\mathbf{v}_1$, red lines represent each coefficient value for the truth $\mathbf{v}_1$; In figure C, blue lines represent each coefficient value for estimation of $\mathbf{u}_2$, red lines represent each coefficient value for the truth $\mathbf{u}_2$; In figure D, blue lines represent each coefficient value for estimation of $\mathbf{v}_2$, red lines represent each coefficient value for the truth $\mathbf{v}_2$; The estimations have the similar sparse structure for both $\mathbf{u}_{1:2}$ and $\mathbf{v}_{1:2}$.

$\mathrm{df}_{\widetilde{\mathbf{v}}_1} = 4$, $\mathrm{df}_{\widetilde{\mathbf{u}}_2} = 5$, $\mathrm{df}_{\widetilde{\mathbf{v}}_2} = 4$ with

$$\begin{aligned}
\widetilde{\mathbf{u}}_1 &= (0, 0.429, 0.308, 0.656, 0.539, 0, 0, 0, 0, 0)^t, \\
\widetilde{\mathbf{v}}_1 &= (0, 0, 0.572, 0.576, 0.351, 0.466, 0, 0, 0, 0)^t, \\
\widetilde{\mathbf{u}}_2 &= (0, 0, 0, 0, 0.565, 0.600, 0.376, 0.428, -0.027, 0)^t, \\
\widetilde{\mathbf{v}}_2 &= (0, 0, 0, 0, 0.420, 0.328, 0.633, 0.562, 0, 0)^t.
\end{aligned}$$

The results showed in Figure 2.4 give us a general idea of the estimations accuracy. The sparse patterns of the estimations are pretty close to the truth and only the sparsity degree for $\widetilde{\mathbf{u}}_2$ is different from that of $\mathbf{u}_2$. The only element not matching the true sparsity is in blue color in $\widetilde{\mathbf{u}}_2$,

and it's magnitude is very close to 0.

One thing to point of that if we repeat the simulation, the random error may effect the result. For instance, for a different A matrix, we can also have $\mathrm{df}_{\widetilde{\mathbf{u}}_1} = 4$, $\mathrm{df}_{\widetilde{\mathbf{v}}_1} = 4$, $\mathrm{df}_{\widetilde{\mathbf{u}}_2} = 6$ and $\mathrm{df}_{\widetilde{\mathbf{v}}_2} = 5$ for the third scenario, but those elements with different sparsity structure from the truth always have pretty small magnitude.

2.5.2 Food Example

In this section, we apply our muti-layer sparse decomposition algorithm to food nutritional data analyzed in Lazzeroni and Owen (2002). Since we can not open the link provided in their paper, we download the data from another website *http://www.invive.com/calorie.html*.

The data contains 7 variables for 961 foods, the first 6 are all nutritional measures including fat, food energy, carbohydrate, protein, cholesterol, and saturated fat. Since all the foods might have different packing and sizes, we divide all the nutritional values by the 7th variable weight. Moreover, we follow the similar procedure as Lazzeroni and Owen (2002) by removing first 6 column's mean.

To test the performance of our method, we first apply SSVD algorithm introduced in Lee et al. (2010). We use the same adaptive lasso penalty on both $\mathbf{u}$ (food aspect) and $\mathbf{v}$ (nutrition aspect) with a common weight parameter $\gamma = 2$. We sequentially extract the first three layers for the data matrix and the results are different from what are shown in the supplementary materials of Lee et al. (2010). For the first layer, it consists 754 foods and 5 nutritional facts: fat (0.0921), energy (0.9934), carbohydrate (0.0354), cholesterol (0.0553) and saturated fat (0.0188), where the number in parentheses are the values of estimated $\mathbf{v}_1$. It implies that the first layer mainly contains foods which are rich of energy. The second layer contains 159 foods and 4 nutritional measures with heavy emphasis on cholesterol: energy (-0.0690), carbohydrate (-0.0513), protein (0.0231), and cholesterol (0.9960). For the third layer, there are 220 foods with 5 nutritions involved. Energy (0.4430), carbohydrate (0.7028) and cholesterol (-0.5024) are the three main variables with largest magnitude of coefficients.

The results by SSVD are not easy to interpret since there are too many foods in each layer especially for the first one. This is not surprising since all the layers are extracted sequentially. When dealing with the first layer, the solution tends to cover as much information as possible which will include too many foods into this layer.

For our method, we extract the layers simultaneously. The first task is to determine how many

layers we need to use. Here, we compare the BIC for using one, two and three layers to estimate the data matrix. To prevent containing too many foods in each layers, we only allow the sparsity degree of $\mathbf{u}$ less than 400, which means at most 400 foods can be kept in each layer. The BIC values are 11596.32, 10836.9 and 10613.45, respectively. Therefore, we decide to use three layers.

After refitting each layer with our decomposition method, the first layer consists of 124 foods and 2 nutritional facts: fat (0.1195), energy (0.9928). The second layer consists of 8 foods with 1 nutritional measure cholesterol. The third layer consists of 20 foods with 1 nutritional measure energy. Comparing to the results from SSVD, our layers contain much less foods and each layer only focuses on 1 or 2 major nutritions.

Furthermore, we also want to understand the types of foods in each layers. The first layer still contains many foods, and we order these foods by magnitude of the values in $\mathbf{u}_1$. As Table (2.1) shown below, we only list the top 34 foods in the first layer with two groups. These two groups can be treated as liquid oils, and solid oils (butter and margarine). The second layer only contains 8 foods, and all of them are livers and eggs with full of cholesterol. For the third layer, although it still focuses on the energy measure similar as the first layer, it contains all different foods from first layer and they are mainly vegetables.

Foods	u_1	Foods	u_1
Layer 1, group 1			
LARD 1 CUP	0.1497	CORN OIL 1 TBSP	0.1476
OLIVE OIL 1 TBSP	0.1476	PEANUT OIL 1 TBSP	0.1476
SAFFLOWER OIL 1 TBSP	0.1476	SOYBEAN-COTTONSEED OIL, HYDRGN 1 TBSP	0.1476
SOYBEAN OIL, HYDROGENATED 1 TBSP	0.1476	SUNFLOWER OIL 1 TBSP	0.1476
FATS, COOKING/VEGETBL SHORTENG 1 TBSP	0.1458	LARD 1 TBSP	0.1458
OLIVE OIL 1 CUP	0.1457	PEANUT OIL 1 CUP	0.1457
CORN OIL 1 CUP	0.1454	SAFFLOWER OIL 1 CUP	0.1454
SOYBEAN-COTTONSEED OIL, HYDRGN 1 CUP	0.1454	SOYBEAN OIL, HYDROGENATED 1 CUP	0.1454
SUNFLOWER OIL 1 CUP	0.1454	FATS, COOKING/VEGETBL SHORTENG 1 CUP	0.1454
Layer 1, group 2			
MACADAMIA NUTS, OILRSTD, SALTED 1 OZ	0.1100	MACADAMIA NUTS, OILRSTD, UNSALT 1 OZ	0.1100
BUTTER, SALTED 1/2 CUP	0.1088	BUTTER, UNSALTED 1/2 CUP	0.1088
MARGARINE, REGULR, HARD, 80% FAT 1/2 CUP	0.1087	MACADAMIA NUTS, OILRSTD, SALTED 1 CUP	0.1086
MACADAMIA NUTS, OILRSTD, UNSALT 1 CUP	0.1086	MARGARINE, REGULR, SOFT, 80% FAT 8 OZ	0.1086
BUTTER, SALTED 1 TBSP	0.1082	BUTTER, UNSALTED 1 TBSP	0.1082
MARGARINE, REGULR,HARD, 80% FAT 1 TBSP	0.1082	MARGARINE, REGULR, SOFT, 80% FAT 1 TBSP	0.1082
MAYONNAISE, REGULAR 1 TBSP	0.1082	BUTTER, SALTED 1 PAT	0.1051
BUTTER, UNSALTED 1 PAT	0.1051	MARGARINE, REGULR, HARD, 80% FAT 1 PAT	0.1051
Layer 2			
EGGS, RAW, YOLK 1 YOLK	0.7265	CHICKEN LIVER, COOKED 1 LIVER	0.3578
BEEF LIVER, FRIED 3 OZ	0.2704	EGGS, COOKED, FRIED 1 EGG	0.2564
EGGS, COOKED, HARD-COOKED 1 EGG	0.2370	EGGS, RAW, WHOLE 1 EGG	0.2370
EGGS, COOKED, POACHED 1 EGG	0.2358	EGGS, COOKED, SCRAMBLED/OMELET 1 EGG	0.1935
Layer 3, top 10			
BEEF BROTH, BOULLN, CONSM,CNND 1 CUP	0.2301	PICKLES, CUCUMBER, DILL 1 PICKLE	0.2286
LETTUCE, CRISPHEAD, RAW,PIECES 1 CUP	0.2271	NATURE VALLEY GRANOLA CEREAL 1 OZ	0.2264
ONION SOUP, DEHYDRATD, PREPRED 1 PKT	0.2252	CABBAGE, CHINESE, PAK-CHOI, CKD 1 CUP	0.2243
LETTUCE, BUTTERHEAD, RAW, HEAD 1 HEAD	0.2237	ASPARAGUS, CANNED, SPEARS, NOSALT 4 SPEARS	0.2236
ASPARAGUS, CANNED, SPEARS, W/SALT 4 SPEARS	0.2236	CELERY, PASCAL TYPE, RAW, STALK 1 STALK	0.2236

Table 2.1: Food groups in first three layer estimations

Chapter 3

Sparse Matrix Decomposition with Missing Data

3.1 Introduction

Principal component analysis (PCA) is one of the most commonly used techniques in multivariate analysis for dimension reduction and feature extraction. It has a wide array of applications, ranging from image recognition to gene expression microarray analysis. Unfortunately, this method as well as its base tool singular value decomposition (SVD) require a complete matrix as an input for analysis. However, due to dramatic advances in science and technology, high-dimensional data are now routinely collected in most of the the fields, and for diverse reasons we frequently need to deal with values missing in the data. Witten (2007) mentioned that when some elements of the data matrix X are missing, those elements could simply be excluded from all computations. Therefore, they came up a"criterion" which only used the non missing data to perform their penalized matrix decomposition (PMD). The possibility of computing the PMD in the presence of missing data lead to a simple and automated method for the selection of the tuning parameters in their PMD. The basic idea is to tune the penalized parameters via cross validation among different folders of missing data, i.e., retrieve every estimation based on a incomplete data through the"criterion" and compare the estimation of the missing values to the truth. However, we find this"criterion" for missing data in Witten (2007) is only an approximate method without any detailed derivation. In this chapter, we will propose a concrete solution for PMD with missing data involved, and we also compare the difference between two solutions when dealing with sparse data matrices. Moreover, a simulation example is provided to help understand the differences.

3.2 Methodology

Consider a data matrix $A \in \mathbb{R}^{m\times n}$ with some missing elements. Let $C = \{(i,j) : A_{ij} \text{ is missing}\}$ and $C_j = \{i : (i,j) \notin C\}$. We want to find a one layer matrix decomposition on A focusing on non-missing elements. The problem can be considered as finding norm one vector $\mathbf{u} \in \mathbb{R}^m$, $\mathbf{v} \in \mathbb{R}^n$, and a scalar $d > 0$ to estimate A such that it minimizes the Frobenius norm between A and $d\mathbf{u}\mathbf{v}^t$ with respect to non-missing elements, i.e.,

$$\min_{\|\mathbf{u}\|_2^2=1,\|\mathbf{v}\|_2^2=1,d>0} \sum_{(i,j)\in C} (A_{ij} - du_iv_j)^2 \tag{3.2.1}$$

Moreover, we usually want $\mathbf{u}$ and $\mathbf{v}$ to be sparse and therefore we add L_1 penalty to the objective function (3.2.1) with tuning parameter λ_1 and λ_2,

$$\min_{\|\mathbf{u}\|_2^2=1,\|\mathbf{v}\|_2^2=1} \Bigg(\sum_{(i,j)\in C} (A_{ij} - du_iv_j)^2 + \lambda_1 \sum_{i=1}^{m} |u_i| + \lambda_2 \sum_{j=1}^{n} |v_j| \Bigg). \tag{3.2.2}$$

Directly minimize this objective function is not straightforward. We first consider to solve $\mathbf{v}$ when d and $\mathbf{u}$ are given. After that, we continue to solve $\mathbf{u}$ given d and updated $\mathbf{v}$, and solve d given updated $\mathbf{v}$ and updated $\mathbf{u}$. We keep doing these three steps until all the $\mathbf{u}$, $\mathbf{v}$ and d converge. Since the updating rule for $\mathbf{v}$ and $\mathbf{u}$ are symmetric, without loss of generality, in remaining part, we only focus on minimize the function with respect to $\mathbf{v}$ given $d > 0$ and $\mathbf{u}$, i.e.,

$$\mathbf{v} = \arg\min_{\|\mathbf{v}\|_2^2=1} \Bigg(\sum_{(i,j)\in C} (A_{ij} - du_iv_j)^2 + \lambda \sum_{j=1}^{n} |v_j| \Bigg). \tag{3.2.3}$$

Let $a_j = \sum_{i\in C_j} u_i^2$, $b_j = \sum_{i\in C_j} A_{ij}u_i$. Notice that $\mathbf{u}$ is a norm one vector, we have $a_j \in [0,1]$ for $j = 1 : m$. Without loss of generality, we assume there exists $s \le n$ such that $a_1, ..., a_s > 0$ and $a_{s+1} = a_{s+2} = ... = a_n = 0$, and we can rewrite the function in (3.2.1) is

$$\begin{aligned} \sum_{(i,j)\in C} (A_{ij} - du_iv_j)^2 &= d^2 \sum_{j=1}^{s} a_j v_j^2 - 2d \sum_{j=1}^{s} b_j v_j - 2d \sum_{j=s+1}^{n} \sum_{i\in C_j} A_{ij}u_iv_j + \sum_{(i,j)\in C} A_{ij}^2 \\ &= d^2 \sum_{j=1}^{s} a_j \Bigg(v_j - \frac{b_j}{da_j} \Bigg)^2 - 2d \sum_{j=s+1}^{n} \sum_{i\in C_j} A_{ij}u_iv_j + f(A,\mathbf{u}), \end{aligned}$$

where $f(A,\mathbf{u}) = \sum\limits_{(i,j)\in C} A_{ij}^2 - \sum\limits_{j=1}^{s} \frac{b_j^2}{a_j}$ is a function of A and $\mathbf{u}$. Since A and $\mathbf{u}$ are all given, we treat $f(A,\mathbf{u})$ as a constant.

Therefore, the minimum value of the objective function (3.2.3) as

$$
\begin{aligned}
& \min_{\|\mathbf{v}\|_2^2=1} \Big(\sum_{(i,j)\in C} (A_{ij} - du_i v_j)^2 + \lambda \sum_{j=1}^{n} |v_j| \Big) \\
= & \min_{\|\mathbf{v}\|_2^2=1} \Big[d^2 \sum_{j=1}^{s} a_j \Big(v_j - \frac{b_j}{da_j} \Big)^2 + \lambda \sum_{j=1}^{s} |v_j| \Big] + \Big[-2d \sum_{j=s+1}^{n} \sum_{i\in C_j} A_{ij} u_i v_j + \lambda \sum_{j=s+1}^{n} |v_j| \Big] + f(A,\mathbf{u}) \\
= & \min_{r\in[0,1]} \Big(\min_{\sum_{k=1}^{s} v_k^2 = r^2} \Big[d^2 \sum_{j=1}^{s} a_j \Big(v_j - \frac{b_j}{da_j} \Big)^2 + \lambda \sum_{j=1}^{s} |v_j| \Big] \\
& + \min_{\sum_{k=s+1}^{n} v_k^2 = 1-r^2} \Big[-2d \sum_{j=s+1}^{n} \sum_{i\in C_j} A_{ij} u_i v_j + \lambda \sum_{j=s+1}^{n} |v_j| \Big] \Big) + f(A,\mathbf{u}) \qquad (3.2.4) \\
= & \min_{r\in[0,1]} \Big(M_1(r) + M_2(r) \Big) + f(A,\mathbf{u}). \qquad (3.2.5)
\end{aligned}
$$

In (3.2.4), for given $r \in [0,1]$ we treat the first and second terms as two functions $M_1(r)$ and $M_2(r)$ with respect to r, and the final minimal value achieves as we minimize $M_1(r) + M_2(r)$ over $r \in [0,1]$.

In terms of finding closed form solution of (3.2.5), it's pretty complicate because solving $M_1(r)$ and $M_2(r)$ includes different situations as λ varies. To make things easier, now we assume $s = n$, i.e., $a_j > 0$ for all $j = 1, 2, ..., n$. It helps us get rid of $M_2(r)$ as well as r and only

$$
M_1(r=1) = \min_{\sum_{k=1}^{n} v_k^2 = 1} \Big[d^2 \sum_{j=1}^{n} a_j \Big(v_j - \frac{b_j}{da_j} \Big)^2 + \lambda \sum_{j=1}^{n} |v_j| \Big] \qquad (3.2.6)
$$

need to be considered. To keep our solution complete, we provide the method solving

$$
M_2(r) = \min_{\sum_{k=s+1}^{n} v_k^2 = 1-r^2} \Big(-2d \sum_{j=s+1}^{n} \sum_{i\in C_j} A_{ij} u_i v_j + \lambda \sum_{j=s+1}^{n} |v_j| \Big)
$$

in the Appendix C when $s < n$.

In many situations, it's common to add a weight matrix $W_{m\times n}$ with each element $w_{ij} \geq 0$ and

$\sum_{ij} w_{ij} = nm$ in (3.2.3). Then the new objective function becomes

$$\min_{\|\mathbf{v}\|_2^2=1} \Big(\sum_{i,j} w_{ij} \big(A_{ij} - d u_i v_j\big)^2 + \lambda \sum_{j=1}^{n} |v_j| \Big), \tag{3.2.7}$$

and it is a more general form of equation (3.2.3) since if we set $w_{ij} = 0$ for $(i,j) \in C$ and make other values of w_{ij} for $(i,j) \notin C$ the same, (3.2.3) and (3.2.7) become the same.

Define new a_j and b_j as $a_j = \sum_{i=1}^{m} u_i^2 w_{ij}$, $b_j = \sum_{i=1}^{m} A_{ij} u_i w_{ij}$, and we can rewrite (3.2.7) as

$$\min_{\|\mathbf{v}\|_2^2=1} \Big(\sum_{i,j} w_{ij} (A_{ij} - d u_i v_j)^2 + \lambda \sum_{j=1}^{n} |v_j| \Big) = \min_{\|\mathbf{v}\|_2^2=1} \Big[d^2 \sum_{j=1}^{n} a_j \Big(v_j - \frac{b_j}{d a_j} \Big)^2 + \lambda \sum_{j=1}^{n} |v_j| + f(A, \mathbf{u}) \Big],$$

if $a_j > 0$ for all $j = 1, 2, ..., n$. It becomes the same objective function as M_1 in (3.2.6). In the remaining part of this chapter, we focus on finding the closed form solution of M_1.

To solve M_1, we claim that if $\mathbf{v}$ is the solution of M_1, then the sign of j-th element v_j in $\mathbf{v}$ is determined by the sign of $\frac{b_j}{d a_j}$, i.e., $\text{sgn}(v_j) = \text{sgn}(\frac{b_j}{d a_j})$. This is because if $\text{sgn}(v_j) = -\text{sgn}(\frac{b_j}{d a_j})$,

$$\begin{aligned} d^2 \sum_{j=1}^{n} a_j \Big(v_j - \frac{b_j}{d a_j} \Big)^2 &> d^2 \sum_{j=1}^{n} a_j \Big(v_j - (-\frac{b_j}{d a_j}) \Big)^2 \\ &= d^2 \sum_{j=1}^{n} a_j \Big((-v_j) - \frac{b_j}{d a_j} \Big)^2. \end{aligned}$$

It means v_j with $\text{sgn}(v_j) = -\text{sgn}(\frac{b_j}{d a_j})$ could not be the j-th element of $\mathbf{v}$ which minimizes M_1. Therefore, our claim is proved and we will focus on solving the magnitude of v_j and the objective function with respect to solving this magnitude is

$$\begin{aligned} & \min_{\|\mathbf{v}\|_2^2=1, v_j \geq 0} \Big(d^2 \sum_{j=1}^{n} a_j \big(v_j - \frac{|b_j|}{d a_j} \big)^2 + \lambda \sum_{j=1}^{n} |v_j| \Big) \\ = & \min_{\|\mathbf{v}\|_2^2=1, v_j \geq 0} d^2 \sum_{j=1}^{n} a_j \Big(v_j - \frac{|b_j| - \frac{\lambda}{2d}}{d a_j} \Big)^2. \end{aligned} \tag{3.2.8}$$

The solution for objective function (3.2.8) has three different cases as penalty parameter λ varies.

- Case I: $0 \leq \frac{\lambda}{2d} \leq \min_{j=1:n} |b_j|$.

Given Case I's condition, we have $\frac{|b_j| - \frac{\lambda}{2d}}{d a_j} \geq 0$, for $j = 1, 2, ..., n$. If we remove the $v_j \geq 0$ restriction,

the solution still guarantees $v_j \geq 0$, i.e.,

$$\min_{\|\mathbf{v}\|_2^2=1, v_j \geq 0} d^2 \sum_{j=1}^{n} a_j \left(v_j - \frac{|b_j| - \frac{\lambda}{2d}}{da_j} \right)^2 = \min_{\|\mathbf{v}\|_2^2=1} d^2 \sum_{j=1}^{n} a_j \left(v_j - \frac{|b_j| - \frac{\lambda}{2d}}{da_j} \right)^2. \tag{3.2.9}$$

This problem can be solved by Lagrange multiplier method. We have the Lagrange function with Lagrange multiplier α as

$$d^2 \sum_{j=1}^{n} a_j \left(v_j - \frac{|b_j| - \frac{\lambda}{2d}}{da_j} \right)^2 + \alpha \left(\sum_{j=1}^{n} v_j^2 - 1 \right).$$

Taking derivative with respect to v_j and α and setting all the equations equal to 0, we have the solution satisfies

$$\begin{cases} \sum_{j=1}^{n} v_j^2 = 1 \\ 2d^2 a_j \left(v_j - \frac{|b_j| - \frac{\lambda}{2d}}{da_j} \right) + 2\alpha v_j = 0 \Leftrightarrow v_j = \frac{d\left(|b_j| - \frac{\lambda}{2d}\right)}{d^2 a_j + \alpha}, \quad j = 1, 2, ..., n \end{cases}$$

where α is chosen to make $v_j > 0$ and $\sum_{j=1}^{n} v_j^2 = 1$. We would like to write v_j as $v_j(\alpha)$ since its value is related to α.

Notice that there are at most $2n$ different α making $\sum_{j=1}^{n} v_j^2(\alpha) = 1$. If $k = \arg\min_{j=1:n} d^2 a_j$, there exists an unique $\alpha > -d^2 a_k$, such that $\sum_{j=1}^{n} v_j^2(\alpha) = 1$ and $v_j(\alpha) > 0$ for all $j = 1, 2, ..., n$. This is the α chosen to solve v_j.

We provide a geometric interpretation of Case I in figure 3.1. This is a special two dimensional case ($n = 2$) for a easy visualization. When $n = 2$, equation (3.2.9) becomes

$$\min_{v_1^2 + v_2^2 = 1, v_{1,2} \geq 0} d^2 \Big[a_1 \left(v_1 - \frac{|b_1| - \frac{\lambda}{2d}}{da_1} \right)^2 + a_2 \left(v_2 - \frac{|b_2| - \frac{\lambda}{2d}}{da_2} \right)^2 \Big]. \tag{3.2.10}$$

With constrains a_1, $a_2 \in (0, 1]$ and $\frac{|b_i| - \frac{\lambda}{2d}}{da_i} > 0$ for $i = 1, 2$, equation (3.2.14) can be considered as an ellipse with its center $(\frac{|b_1| - \frac{\lambda}{2d}}{da_1}, \frac{|b_2| - \frac{\lambda}{2d}}{da_2})$ in the 1^{st} quadrant of the Cartesian coordinate. Finding the minimal value with conditions $v_1^2 + v_2^2 = 1$ and $v_{1,2} \geq 0$ is equal to magnify the ellipse until it touches the unit circle line in the 1^{st} quadrant. In Figure 3.1, The blue line is a quarter of unit circle in 1^{st} quadrant and it represents the the condition $v_1^2 + v_2^2 = 1$ and $v_{1,2} \geq 0$. There are other three concentric ovals in green, red and purple with different centers and major / minor axises. They

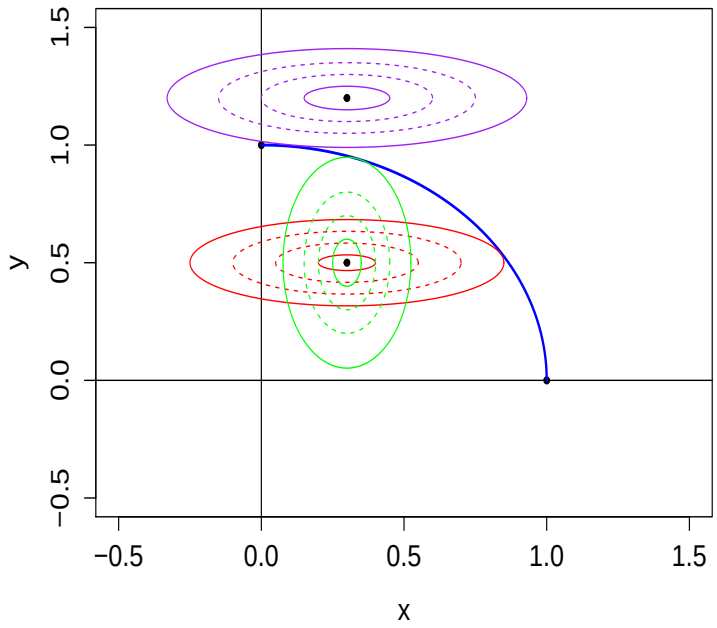

Figure 3.1: Geometric interpretation of solution under case I. The blue line is a 1/4 unit circle in 1^{st} quadrant. Green, red and purple lines are ellipses with different centers in 1^{st} quadrant and different major / minor axis.

stand for values with different combinations of $a_{1,2}$ and $\frac{|b_{1,2}|-\frac{\lambda}{2d}}{da_{1,2}}$. We try different centers as well as different shapes (major / minor axises) for these ellipses. No matter how they change, the ellipse is guaranteed to meet with the unit circle in 1^{st} quadrant. That is the reason why α should be chosen to make all $v_j(\alpha > 0)$ in the solution.

- Case II: $\frac{\lambda}{2d} \geq \max_{j=1:n} |b_j|$

Case II guarantees that for all $j = 1, 2, ..., n$, $\frac{|b_j|-\frac{\lambda}{2d}}{da_j} < 0$. We can no longer apply the similar method as Case I, and the following lemma is used to solve this case.

Lemma 3.1. *For* $a_1,\ a_2 \geq 0,\ c_1,\ c_2 \geq 0,$ *and* $0 < r \leq 1$

$$\min_{v_1^2+v_2^2=r^2, v_1, v_2 \geq 0} \left(a_1(v_1+c_1)^2 + a_2(v_2+c_2)^2 \right) = \min \left(a_1(r+c_1)^2 + a_2 c_2^2,\ a_1 c_1^2 + a_2(r+c_2)^2 \right).$$

In other words, the minimal value is chosen at the point $(v_1, v_2) = (r, 0)$ *or* $(v_1, v_2) = (0, r)$.

Proof: Without lost of generality, we assume $a_2 \geq a_1$, then

$$\begin{aligned}
&\min_{v_1^2+v_2^2=r^2, v_1, v_2 \geq 0} a_1(v_1+c_1)^2 + a_2(v_2+c_2)^2 \\
&= \min_{t\in[0,r]} a_1(t+c_1)^2 + a_2(\sqrt{r^2-t^2}+c_2)^2 \\
&\triangleq \min_{t\in[0,1]} F(t).
\end{aligned}$$

If we take derivative of $F(t)$,

$$\begin{aligned}
F^{'}(t) &= 2a_1(t+c_1) - 2a_2 t\frac{\sqrt{r^2-t^2}+c_2}{\sqrt{r^2-t^2}} \\
&= 2(a_1-a_2)t - 2a_2c_2\frac{t}{\sqrt{r^2-t^2}} + 2a_1c_1.
\end{aligned}$$

$F^{'}(t)$ is a decreasing function, $F^{'}(0) = 2a_1c_1 \geq 0$ and $F^{'}(1^{-}) \to -\infty$. Thus, there exists a positive number ϵ, such that $F(t)$ is non-decreasing between $[0,\epsilon]$, and $F(t)$ is decreasing between $[\epsilon, r]$. It indicates that the extreme point (with derivative value equal to 0) can not be a local minimal point and the overall minimal point between $[0,r]$ should be on the boundary, i.e.,

$$\min_{t\in[0,r]} F(t) = \min\Big(F(0), F(r)\Big).$$

□

Now we provide the solution for case II in the following theorem.

Theorem 3.1. *If* $\frac{\lambda}{2d} \geq \max_{j=1:n} |b_j|$, *the solution of* $\boldsymbol{v} = (v_1, v_2, ..., v_n)$ *to minimize*

$$\min_{\|\boldsymbol{v}\|^2=1, v_j\geq 0} d^2 \sum_{j=1}^{n} a_j \left(v_j - \frac{|b_j| - \frac{\lambda}{2d}}{da_j}\right)^2 \tag{3.2.11}$$

can only be chosen among n candidates: $v_k = 1$, $v_j = 0$ *for* $j \neq k$ *and* $k \in \{1, 2, ..., n\}$, *i.e., only one direction of* $\boldsymbol{v}$ *can take a nonzero value equaling to 1.*

Proof: Suppose the solution $\mathbf{v} = (\hat{v}_1, \hat{v}_2, ..., \hat{v}_n)$ for (3.2.11) has at least two nonzero directions, let's assume $\hat{v}_1$ and $\hat{v}_2 > 0$ with $\hat{v}_1^2 + \hat{v}_2^2 = r^2 \in (0,1]$, and $\sum_{i=3}^{n} \hat{v}_j^2 = 1 - r^2$. Let $c_j = \frac{-|b_j| + \frac{\lambda}{2d}}{da_j} > 0$,

then

$$\begin{aligned}
&\min_{\|\mathbf{v}\|_2^2=1, v_j\geq 0} d^2 \sum_{j=1}^{n} a_j \left(v_j - \frac{|b_j| - \frac{\lambda}{2d}}{da_j} \right)^2 \\
= \;& d^2 \left(a_1 \big(\hat{v}_1 + c_1\big)^2 + a_2 \big(\hat{v}_2 + c_2\big)^2 \right) + d^2 \sum_{j=3}^{n} a_j \big(\hat{v}_j + c_j\big)^2 \qquad (3.2.12) \\
> \;& d^2 \min \left(a_1 (r + c_1)^2 + a_2 c_2^2,\; a_1 c_1^2 + a_2 (r + c_2)^2 \right) + d^2 \sum_{j=3}^{n} a_j \big(\hat{v}_j + c_j\big)^2 \qquad (3.2.13)
\end{aligned}$$

where $>$ holds in (3.2.13) by applying lemma 3.1 on the term $a_1\big(\hat{v}_1 + c_1\big)^2 + a_2\big(\hat{v}_2 + c_2\big)^2$ in (3.2.12). This is a contradiction with the assumption $\hat{\mathbf{v}} = (\hat{v}_1, \hat{v}_2, ..., \hat{v}_n)$ is the solution for (3.2.11). Therefore, at most one v_j in $\mathbf{v}$ can be nonzero. Since $\|\mathbf{v}\|_2^2 = 1$, there is exactly one direction of $\mathbf{v}$ equals to 1 and the remaining coefficient are all equal to 0. □

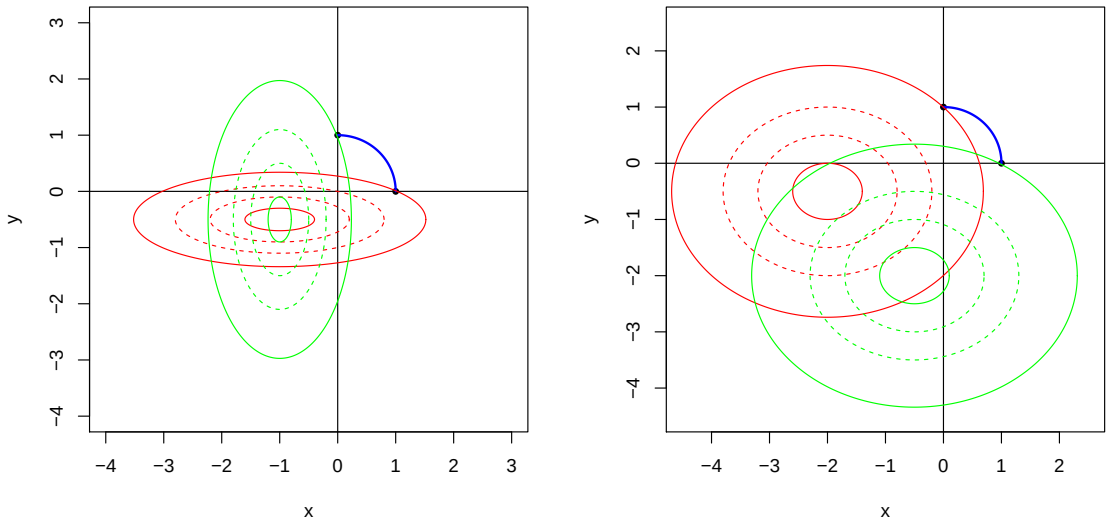

Figure 3.2: Geometric interpretation of Theorem 3.1. The blue line is 1/4 unit circle in 1^{st} quadrant. Green and red lines are ellipses with different centers in 3^{rd} quadrant and different major / minor axis.

Let's also consider the intuitive idea behind the Theorem 3.1 with a two dimensional case ($n = 2$). When $n = 2$, objective equation (3.2.11) becomes

$$\min_{v_1^2 + v_2^2 = 1, v_{1,2} \geq 0} d^2 \left[a_1 \left(v_1 - \frac{|b_1| - \frac{\lambda}{2d}}{da_1} \right)^2 + a_2 \left(v_2 - \frac{|b_2| - \frac{\lambda}{2d}}{da_2} \right)^2 \right]. \qquad (3.2.14)$$

With assumptions a_1, $a_2 \in (0,1]$ and $-\frac{|b_i|-\frac{\lambda}{2d}}{da_i} > 0$ for $i = 1$ and 2, minimizing equation (3.2.14) under conditions of $v_1^2 + v_2^2 = 1$ and $v_{1,2} \geq 0$ is equal to magnify an ellipse with a center in the 3^{rd} quadrant until it touches the quarter of unit circle line in the 1^{st} quadrant. In Figure 3.2, we show examples that with same center but different shapes and also ellipses with same shape (same values of major / minor axises) but different centers. Those ellipses can meet the unit circle in different places but the meeting points are guaranteed to be either $\mathbf{v} = (1,0)$ or $\mathbf{v}$=(0, 1). That corresponds the result in the theorem that only one direction of $\mathbf{v}$ can take a nonzero value equaling to 1.

In conclusion, if $\frac{\lambda}{2d} \geq \max_{j=1:n} |b_j|$, our solution is to check the object function's value by plugging in n different points and find the minimal, i.e.,

$$\begin{aligned} & \min_{\|\mathbf{v}\|_2^2=1, v_j \geq 0} d^2 \sum_{j=1}^{n} a_j \left(v_j - \frac{|b_j| - \frac{\lambda}{2d}}{da_j} \right)^2 \\ = \quad & \min \left(a_1 \big(1 - \frac{|b_1| - \frac{\lambda}{2d}}{da_1}\big)^2 + \sum_{j=2}^{n} a_j \big(\frac{|b_j| - \frac{\lambda}{2d}}{da_j}\big)^2, ..., a_n \big(1 - \frac{|b_n| - \frac{\lambda}{2d}}{da_n}\big)^2 + \sum_{j=1}^{n-1} a_j \big(\frac{|b_j| - \frac{\lambda}{2d}}{da_j}\big)^2 \right). \end{aligned}$$

- Case III: $\min_{j=1:n} |b_j| \leq \frac{\lambda}{2d} \leq \max_{j=1:n} |b_j|$

Without loss of generality, we assume $|b_j|$ is in a non-decreasing order, i.e., $b_1 \leq b_2 \leq ... \leq b_n$. In this case, there exists a positive integer s such that $\frac{\lambda}{2d} \geq |b_j|$ for $j = 1, 2, ..., s$ and $\frac{\lambda}{2d} < |b_j|$ for $j = s+1, ..., n$. Then, the objective function can be separated into two parts,

$$\begin{aligned} & \min_{\|\mathbf{v}\|_2^2=1, v_j \geq 0} d^2 \sum_{j=1}^{n} a_j \left(v_j - \frac{|b_j| - \frac{\lambda}{2d}}{da_j} \right)^2 \\ = \quad & \min_{\|\mathbf{v}\|_2^2=1, v_j \geq 0} \left[d^2 \sum_{j=1}^{s} a_j \left(v_j - \frac{|b_j| - \frac{\lambda}{2d}}{da_j} \right)^2 + d^2 \sum_{j=s+1}^{n} a_j \left(v_j - \frac{|b_j| - \frac{\lambda}{2d}}{da_j} \right)^2 \right] \\ = \quad & \min_{r \in [0,1]} \left(\min_{\sum_{k=1}^{s} v_k^2 = r^2, v_j \geq 0} d^2 \sum_{j=1}^{s} a_j \left[v_j - \frac{|b_j| - \frac{\lambda}{2d}}{da_j} \right]^2 + \min_{\sum_{k=s+1}^{n} v_k^2 = 1 - r^2, v_j \geq 0} d^2 \sum_{j=s+1}^{n} a_j \left[v_j - \frac{|b_j| - \frac{\lambda}{2d}}{da_j} \right]^2 \right) \\ \triangleq \quad & \min_{r \in [0,1]} \Big(m_2(r) + m_1(r) \Big). \end{aligned}$$

For any fixed $r \in [0,1]$, we can use the same method in Case I to solve $m_1(r)$, and use the method we deal with Case II to solve $m_2(r)$. The final solution is achieved by minimize a function with respect to r between 0 and 1. Notice that the solution from $m_2(r)$ which is Case II, can only have one non-zero v_k, $1 \leq k \leq s$, and the solution in $m_1(r)$ which is Case I, will have all $v_{s+1:n} > 0$. With a particular chosen of r in final solution, we can have either $n - s$ or $n - s + 1$ non-zero

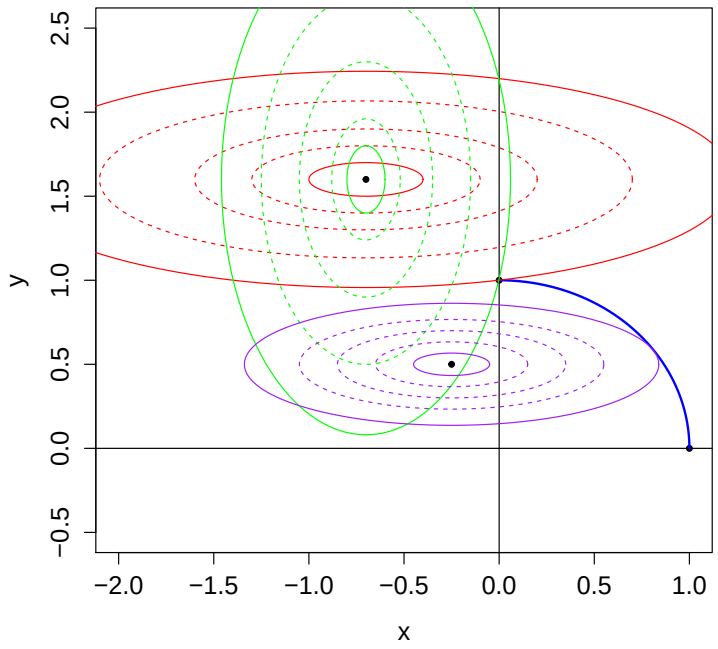

Figure 3.3: Geometric interpretation of solution under case III. The blue line is a 1/4 unit circle in 1^{st} quadrant. Green, red and purple lines are ellipses with different centers in 2^{nd} quadrant and different major / minor axis.

coefficients in $\hat{\mathbf{v}}$. If we use a two dimensional ($n = 2$) example in Figure 3.3 to interpret Case III, we will have a_1, $a_2 \in (0, 1]$, $-\frac{|b_1|-\frac{\lambda}{2d}}{da_1} > 0$ and $\frac{|b_2|-\frac{\lambda}{2d}}{da_2} > 0$. Obviously, the center of the ellipse is in the 2^{nd} quadrant. Different solutions are presented given different shapes and centers of the ellipses. The green and red lines represent the case when $v_1 = 0$ and $v_2 = 1$, i.e., $r = 0$ and all non-zero directions come from $m_1(r)$ part. The purple line is the solution that $v_{1,2} > 0$, i.e., $0 < r < 1$ and all non-zero directions come from $m_1(r)$ part except for one direction coming from $m_2(r)$. Based on the geometric plots in Figure 3.3, we find that no matter different locations and shapes of ellipses in Case III, we will never have a solution with $r = 1$, that is the case when the solution only has one non-zero coefficient and it comes from $m_2(r)$ part.

3.3 Comparison between two methods

In Witten (2007), they proposed an method dealing with missing data with the objective function defined as

$$\min_{\|\mathbf{u}\|_2^2=1,\|\mathbf{v}\|_2^2=1} \sum_{(i,j)\in C} (A_{ij} - du_i v_j)^2 := \max_{\|\mathbf{u}\|_2^2=1,\|\mathbf{v}\|_2^2=1} d\sum_{j=1}^{n} A_{ij}u_i v_j \tag{3.3.1}$$

In fact, this is only an approximate criterion which does not exactly hold for most of the time. Our method is a solid solution, and we provide a summary of a comparison between these two methods in the following.

- Case I: $0 \leq \frac{\lambda}{2d} \leq \min_{j=1:n} |b_j|$.

Let $\widehat{v}_j$ be the solution using our method and $\widetilde{v}_j$ be the solution using equation (3.3.1), then we have

$$\begin{cases} \widehat{v}_j = \frac{d\left(|b_j| - \frac{\lambda}{2d}\right)}{d^2 a_j + \alpha}, \alpha \text{ is chosen to make } \|\widehat{\mathbf{v}}\|^2 = 1 \\ \widetilde{v}_j = \frac{2d|b_j| - \lambda}{\alpha}, \alpha \text{ is chosen to make } \|\widetilde{v}_j\|^2 = 1 \end{cases}$$

Both methods have a dense solution, i.e., all the coefficients in $\mathbf{v}$ are non-zero. However, the magnitudes are different that we adjust $\widehat{v}_j$ according to the missing rate in each column of A. Two solutions are equal if and only if $a_j = 1$ for all $j = 1, 2, ..., n$ which means there is no missing values in the data.

- Case II: $\frac{\lambda}{2d} \geq \max_{j=1:n} |b_j|$

In this case, both of the solutions have only one non-zero direction: $\widehat{v}_j = 1$ if for all $k \neq j$

$$a_j\big(1 - \frac{|b_1| - \frac{\lambda}{2d}}{da_j}\big)^2 + \sum_{i\neq j, i=1:n} a_i\big(\frac{|b_i| - \frac{\lambda}{2d}}{da_i}\big)^2 \leq a_k\big(1 - \frac{|b_k| - \frac{\lambda}{2d}}{da_k}\big)^2 + \sum_{i\neq k, i=1:n} a_i\big(\frac{|b_i| - \frac{\lambda}{2d}}{da_i}\big)^2,$$

and $\widetilde{v}_j = 1$ if

$$j = \arg\min_k 2d|b_k|.$$

If $a_j = 1$ for all $j = 1, 2, .., n$, i.e., there is no missing in data matrix A, we find these two solutions are equivalent. Otherwise, It happens that two methods could have different choices as non-zero directions on the same data. We will show an example in the next simulation part.

- Case III: $\min_{j=1:n} |b_j| \leq \frac{\lambda}{2d} \leq \max_{j=1:n} |b_j|$

This is the most common observed case when λ is not too large or too small. $\widetilde{v}_j$ has non-zero coefficients if $\frac{\lambda}{2d} < |b_j|$, and $\widetilde{v}_j \propto 2d|b_j| - \lambda$. The non-zero coefficients in $\widehat{\mathbf{v}}$ contain two parts, for any j such that $\frac{\lambda}{2d} < |b_j|$, $\widehat{v}_j$ is guaranteed to be non-zero, and it is the same as $\widetilde{v}_j$. However, among all j that $\frac{\lambda}{2d} \geq |b_j|$, there could also be one more direction with non-zero coefficient while it's 0 for $\widetilde{v}_j$. Although the extra non-zero direction is data based which is not always guaranteed, it indicates that in some cases, our method could dig out additional signal information which is ignored by Witten's method.

3.4 Simulation

We design a toy data experiment to test the performance of our solution with respect to different patterns of data missing. The toy data is generated as

$$A = d\mathbf{u}_1\mathbf{v}_1^t + E,$$

where $E = \{e_{i,j}\} \overset{i.i.d.}{\sim} \mathsf{N}(0,1)$, $d = 4$ is a scalar, $\mathbf{u}$ and $\mathbf{v}$ are $\mathbb{R}^{10}$ sparse vectors with 4 non-zero coefficients,

$$\begin{aligned} \mathbf{u} &= \big(0, 2, 1.5, 3, 2.5, 0, 0, 0, 0, 0\big)^T \\ \mathbf{v} &= \big(0, 0, 2.5, 2.5, 1.5, 2, 0, 0, 0, 0\big)^T. \end{aligned}$$

Using this data generation process, if there is no noise term, A is a sparse matrix with signal information contained in the block formed by second to fifth row and third to sixth column. We present an example of a simulated data by adding an error term in the following matrix, and the signal area is in blue color. Meanwhile, we randomly remove 10% (10 points in red color) of the elements in A and we treat them as missing. Missing values outside the information block usually have less impact to the solution, and we mainly focus on testing the relationship between missing values in the information area and the sparse pattern in solution. Without of lost of generality, we choose the third and fourth column which corresponding to the solution $\hat{v}_3$ and $\hat{v}_4$ as the target. At the first stage, there is only one point $A_{5,3}$ missing in the signal area. If the algorithm works, since most of

the information is still complete, the decomposition solution $\widehat{\mathbf{v}} = (\widehat{v}_1, \widehat{v}_2, ..., \widehat{v}_{10})^T$ could have the same sparse structure as the actual, i.e., $\hat{v}_3, \hat{v}_4, \hat{v}_5, \hat{v}_6$ are non-zero and the other directions are zero.

$$\begin{pmatrix}
-1.11 & -0.05 & 0.45 & -0.41 & 2.71 & 0.20 & -1.19 & 1.18 & 2.15 & 1.99 \\
-0.39 & 1.0 & 18.85 & 20.55 & 12.52 & 14.57 & -0.13 & -2.72 & -0.74 & -1.64 \\
-1.22 & -0.35 & 14.89 & 14.86 & 11.01 & 11.37 & 0.48 & -0.98 & -0.69 & 0.06 \\
0.46 & 0.03 & 29.56 & 32.22 & 17.04 & 25.09 & 0.43 & -0.18 & 0.86 & -0.43 \\
-0.65 & -0.31 & 25.12 & 25.25 & 13.98 & 19.80 & -1.32 & -0.33 & 0.23 & -1.06 \\
0.23 & -0.36 & 0.00 & -0.70 & 0.59 & 1.05 & -0.39 & -0.07 & 1.81 & 0.87 \\
-1.26 & 0.10 & -1.01 & -0.23 & 0.55 & 1.36 & -1.06 & -0.03 & 0.19 & 1.31 \\
0.36 & -1.37 & 0.50 & -1.99 & -1.37 & -0.12 & -0.76 & 0.37 & 0.84 & -1.89 \\
-1.60 & 0.24 & 0.06 & 1.36 & 1.62 & 1.94 & -2.27 & 0.20 & 1.02 & 0.77 \\
2.17 & -0.25 & -0.90 & -0.80 & 0.01 & 0.99 & -0.01 & -0.80 & 0.45 & 1.10
\end{pmatrix}$$

To perform the experiment, we fix $\alpha = 3500$ which is the Case III in previous discussion. The algorithm needs an initial value $\mathbf{u}$ and d to solve $\mathbf{v}$, and we apply a SVD on the data matrix and set $\widehat{\mathbf{u}}$ to be the first column of U matrix, and set d to the largest singular value. Given $\widehat{\mathbf{u}}$ and d, we calculate the estimation $\widehat{\mathbf{v}}$ and compare its sparse structure to the truth. After that, we increase the number of missing values in third and fourth column of the informative area in A, i.e., we keep deleting values in A with an order of $A_{3,4}, A_{2,3}, A_{5,4}, A_{2,4}, A_{3,3}$, and each time we track the estimation on $\widehat{\mathbf{v}}$ to see how missing pattern will affect the decomposition solution. All the results are shown in the following Table 3.1.

From Table 3.1, it shows that we can successfully detect the sparse structure of $\mathbf{v}$ (non-zero coefficients v_3, v_4, v_5 and v_6) for the first four data missing patterns. The fourth missing pattern drops values in $A_{5,3}, A_{3,4}, A_{2,3}$ and $A_{5,4}$, and it already covers 50% of the information in third and fourth column. In the next missing case which 70% of signals ($A_{3,4}, A_{5,4}$ and $A_{2,4}$) in column 4 and 50% of signals ($A_{5,3}$ and $A_{2,3}$) in column 3 are also lost, we still retrieve the true sparse pattern with $r = 0.49$. This r value means that the sparse pattern is finally determined through two processes, i.e., the first part $m_1(r)$ digs out v_3, v_5 and v_6 as signal directions and $m_2(r)$ finds the remaining direction

Data Missing Pattern	True Signal Directions				r	Non Signal Directions
	$\widehat{v}_3$	$\widehat{v}_4$	$\widehat{v}_5$	$\widehat{v}_6$		$\widehat{v}_1, \widehat{v}_2, \widehat{v}_7, \widehat{v}_8, \widehat{v}_9, \widehat{v}_{10}$
$A_{5,3}$	0.63	0.66	0.15	0.38	1	all equal to 0
$A_{5,3}, A_{3,4}$	0.61	0.68	0.14	0.38	1	all equal to 0
$A_{5,3}, A_{3,4}, A_{2,3}$	0.64	0.66	0.14	0.37	1	all equal to 0
$A_{5,3}, A_{3,4}, A_{2,3}, A_{5,4}$	0.52	0.76	0.14	0.36	1	all equal to 0
$A_{5,3}, A_{3,4}, A_{2,3}, A_{5,4}, A_{2,4}$	0.33	0.87	0.13	0.34	0.49	all equal to 0
$A_{5,3}, A_{3,4}, A_{2,3}, A_{5,4}, A_{2,4}, A_{3,3}$	0.00	0.93	0.13	0.34	0.36	all equal to 0

Table 3.1: Sparse decomposition solutions using our method corresponding to different missing patterns on a toy data example. The true signal area is a block formed by column 3 to 6 and row 2 to 5, and data missing happens in column 3 and 4 of the signal block.

v_4 even that column 4 has only 25% of the information left. If we apply Witten's approximation method, it's the same that v_3, v_5 and v_6 could be recognized as signals, however the v_4 direction will be treated as an error. In the last case, data completeness becomes even worse, and both of the third and fourth signal columns have 75% missing rate. Witten's method treat both column 3 and 4 as errors due to little information involved, while our method still consider column 4 as informative through its solution in $m_2(r)$ part.

Chapter 4

A Bayesian Algorithm for Sparse Principle Components Analysis

4.1 Introduction

Principal Component Analysis (PCA) is an important visualization and dimension reduction tool with a wide range of applications in data visualization, data compression and system identification (Alter et al. (2001), Prasantha et al. (2007), Huang et al. (2012), Thomasian et al. (1998)). It projects the data onto the principal subspace spanned by k leading eigenvectors of the population covariance matrix, and most of the variance in the data is captured by these k modes. However, the linear combinations found by PCA typically involve all the variables, with non-zero loading, which can be difficulty to interpret. Moreover, in high dimensional setting where the dimension p can be much larger than n, classical PCA leads to very poor estimations and statistical properties (Paul (2007), Johnstone and Lu (2012), Shabalin and Nobel (2013))

To improve the performance of PCA, various proposals have been introduced. Jolliffe (1995) first proposed rotation techniques helping for interpretation on PCs. Jolliffe and Uddin (2000) claimed simplified component technique (SCoT) to find linear combinations of variables that maximize a criterion which contains a trade off between combination variance and simplicity. Vines (2000) restricted simple components with only integers such as 0, 1 and -1. Meanwhile, sparse PCA gained a lot of attention in recent years. It aims to find component loadings with many zero coefficients, thus increasing interpretability of PCs. Methodologies include regularized estimators based on penalizing, constraining the variance maximization formulation of PCA or thresholding. Cadima and Jolliffe (1995) first described a simple thresholding approach, which set regular PC loadings to zero if their absolute values are below a certain threshold. Jolliffe et al. (2003) continually proposed simplified component technique - LASSO (SCoTLASS), which applied LASSO penalty on the loadings to deal with the PCA optimization problem. Zou et al. (2006) reformulated PCA as a regression problem, and achieve PC sparseness by imposing the LASSO penalty on the regression

coefficients. d'Aspremont et al. (2007), d'Aspremont et al. (2008) and Vu et al. (2013) developed convex relaxation techniques that efficiently produce good approximate on the sparse PCA objective function. Johnstone and Lu (2012), Paul and Johnstone (2012)proposed a two stage procedure based on diagonal thresholding. Ma et al. (2013), Yuan and Zhang (2013) developed an iterative algorithm of thresholding.

On the theoretical part, in high dimensional setting, the sample eigenvector is no longer always a consistent estimator. Sometimes, they can even be nearly orthogonal to the true direction. This phenomenon has been studied and examined by many researchers when n, $p \to \infty$ with $p/n \to c \in (0, \infty)$ (Hoyle and Rattray (2004), Lu (2002), Nadler (2008), Onatski (2012), Paul (2007), Reimann et al. (1996)). Meanwhile, if there exists sparse leading eigenvectors in the data, it is also possible to consistently estimate them under high dimensional settings through new estimation conditions. For instance, under the single spiked covariance model assumption, Johnstone and Lu (2012) provided an algorithm for selecting a subset of coordinates with largest sample variances, and showed that if PCA is done on the selected subset, consistency is recovered when the leading eigenvalue is bounded and $(\log p \vee n)/n \to 0$. Under the same single spiked model, if the leading eigenvector has at most k nonzero loadings, Amini and Wainwright (2008) studied conditions for consistent recovering the support set of the maximal eigenvector using simple diagonal thresholding method in Johnstone and Lu (2012) and a semidefinite programming (SDP) relaxation for sparse PCA in d'Aspremont et al. (2007). Shen et al. (2013) established conditions for consistency of a sparse PCA method in High Dimension, Low Sample Size (HDLSS) setting, i.e., n is fixed and $p \to \infty$. Paul and Johnstone (2012) proposed an augmented sparse PCA (ASPCA) estimator of the leading eigenvectors based on a coordinate selection scheme combined with PCA and proved that their procedure obtains the optimal rate of convergence under a high-dimensional sparse setting. There are more theoretical works of estimating the leading eigenvectors from a high-dimensional population covariance, topics in consistency, rates of convergence and minimax risk bounds are covered in Lounici (2013), Ma et al. (2013), Berthet and Rigollet (2013), Cai et al. (2013), Vu and Lei (2012), Vu et al. (2013).

If we let the true leading PC to be ρ, identifying the non-zero loading set of ρ defined as $I = \{i : \rho_i \neq 0\}$ can be also considered as a variable selection problem. Variable selection, also know as feature selection is one of the important topics in statistics. It is used for simplifying models with easier interpretations and reducing overfitting to enhance generalization. The popular model

selection approaches include criteria based selections (Akaike (1973), Mallows (1973), Schwarz et al. (1978)), penalized regression (Tibshirani (1996), Fan and Li (2001), Fan et al. (2004)) and Bayesian approaches (Bottolo et al. (2010), Li and Zhang (2010), Stingo and Vannucci (2011)). The Bayesian variable selection makes inference on posterior, and its advantage is that we can easily incorporate prior knowledge as well as many sources of variation. A general model usually considers a joint density of latent variables $z = z_{1:p}$ and data $x = x_{1:n}$,

$$p(z, x) = p(z)p(x|z),$$

where the latent variables help to determine the distribution of the data. The latent variables are drawn from a prior $p(z)$ and connected to the observations through the likelihood $p(x|z)$. Inference in a Bayesian model refers to the posterior $p(z|x)$. If $p(z)$ is from Bernoulli distribution as well as the conjugated posterior, we can then study the inclusion probability of each variable, which provides more information comparing to a point estimator.

In complex Bayesian models, it is particularly useful to evaluate posterior distributions by Markov chain Monte Carlo (MCMC) sampling. For decades, there many different MCMC algorithms which use different techniques for generating the Markov Chain. The most popular ones include the Metropolis-Hastings (Metropolis et al. (1953), Hastings (1970)) and the Gibbs Sampler (Geman and Geman (1984)). In Metropolis-Hastings algorithm, items are selected from an arbitrary "proposal" distribution and are retained according to an acceptance rule. The Gibbs sampler is a special case in which the proposal distributions are conditional distributions of single components of a vector parameter. MCMC algorithms then are widely studied, and detailed reviews on related applications and extensions could be found in Robert (2004). However, Liang et al. (2008) pointed out that except for special cases such as linear models with carefully chosen priors, Bayesian inference via MCMC for large scale problems is inefficient. The known difficulties for MCMC in high dimension models include: 1) it's difficult to design a Markov chain that efficiently samples the state space of interest; 2) it's sensitive to prior choices and the resulting MCMC estimators could have high variance thus producing less reliable inference and poor forecasts; 3) there are no diagnostics to guarantee that the MCMC chain has converged.

On the other hand, Variational Bayes (VB) is an another efficient alternative to MCMC for Bayesian inference (Bishop (2006), Ormerod and Wand (2010)). It is based on approximating

the target posterior distribution by a variational distribution and solving it using optimization techniques. Specifically, suppose the posterior distribution of interest over a set of unobserved variables $Z = \{Z_1, ..., Z_M\}$ given some data X is $p(Z|X)$,but it is difficult to draw samples from it. We propose a variational distribution $q_v(Z)$ parameterized by v to approximate $p(Z|X)$. The distribution $q_v(Z)$ is usually restricted to belong to a family of simpler distributions than $P(Z|X)$, such as exponential family. Meanwhile, it's intended to make $q_v(Z)$ similar to the true posterior p(Z—X). The similarity between $q_v(Z)$ and the target posterior $p(Z|X)$ is measured by dissimilarity function $D(p, q)$, and the most common chosen function is the Kullback-Leibler divergence (Kullback and Leibler, 1951). The K-L divergence between two distributions $q_v(Z)$ and $p(Z|X)$ is defined as

$$D\big(q_v(Z) \parallel p(Z|X)\big) = \int_Z q_v(Z) \log \frac{q_v(Z)}{p(Z|X)} dZ = \mathbb{E}_q \log \frac{q_v(Z)}{p(Z|X)} = \mathbb{E}_q \log \frac{q_v(Z)}{p(Z, X)} + \log p(X)$$

To find a approximation as close as to the target p(Z—X), we want to minimize the above K-L divergence. The last term $\log p(X)$ is a constant given the data, thus minimizing $D\big(q_v(Z) \parallel p(Z|X)\big)$ is equivalent to maximize the equaltion:

$$\mathcal{L}(q_v) = \mathbb{E}_q \log \frac{q_v(Z)}{p(Z, X)} = \mathbb{E}_q \log q_v(Z) - \mathbb{E}_q \log p(Z, X).$$

In practice, with certain assumptions on $q_v(Z)$, the optimization in $\mathcal{L}(q_v)$ can be much more computationally available than the original posterior sampling. In details, the variational distribution $q_v(Z)$ is usually assumed to be factorized over all latent variables, i.e.,

$$q_v(Z) = \prod_{i=1}^{M} q_i(Z_i).$$

Therefore, it can be proved that the maximization solution for $\mathcal{L}(q_v)$ is to make each factor $q_i(Z_i)$ satisfy

$$q_i(Z_i) = \frac{\exp\big(\mathbb{E}_{i \neq j}\big[\log p(Z, X)\big]\big)}{\int \exp\big(\mathbb{E}_{i \neq j}\big[\log p(Z, X)\big]\big) dZ_i}. \tag{4.1.1}$$

We usually first initialize all of the $q_i(Z_i)$ appropriately, and then we cycle through the factors and replace each in turn with a revised estimate given by equation (4.1.1), using the current estimates for all of the other factors. The overall objective function is a convex optimization problem and therefore

it's guaranteed that this cycling update procedure converges in the end (Boyd and Vandenberghe, 2004).

These is always a comparison between MCMC and VB algorithms, and therefore it helps to learn when to use these two algorithm accordingly. The VB algorithm has several desired advantages compared to the MCMC sampling algorithm. In general, VB is suited to large data sets since it is typically a much faster, deterministic algorithms, and its convergence can be guaranteed through the evaluation of the objective function. However, there are also drawbacks of VB algorithm. Unlike MCMC, several empirical research have shown that variational inference does not necessarily achieve an arbitrary accuracy (Blei et al. (2006), Braun and McAuliffe (2010), Kucukelbir et al. (2016)). The reasons could be the gradient descent method used to solve the objective function only finds local maxima, and global solution is not guaranteed, and we usually put the assumption on the variational distribution that it can be factorized over parameter space, and this construction is not always be able well approximate the true posterior distribution. Nevertheless, there are already justifications on VB's effectiveness in many different practical fields, such as graphical models for information retrieval (Jordan, 2004), cluster analysis of gene-expression data (Teschendorff et al., 2005), and functional magnetic resonance imaging (fMRI) data (Flandin and Penny, 2007).

In this chapter, different from the popular PCA related approaches mentioned above which mainly focus on penalized likelihood and different kinds of thresholding, we develop a new Bayesian variable selection method to deal with Sparse PCA problem. The basic idea of our algorithm is applying Variational Bayes to find the best approximation of the true posterior distribution at each step through a hybrid of Expectation Maximization (EM) and Variational Bayes (VB) process. Empirical study shows that it converges very fast. We also study the asymptotic properties of our algorithm and prove that it achieves selection consistency by ruining algorithm once with certain assumptions and conditions.

4.2 Methodology

4.2.1 Model Setting and Priors

We consider a PCA setting when there is a single principal component. Suppose each data point is a p-dimensional column vector

$$x_i = c_i\rho + \xi w_i,\ i = 1, 2, ..., n, \tag{4.2.1}$$

where $\rho = (\rho_1, ..., \rho_p) \in \mathbb{R}^p$ is the single component to be estimated, $c_i \overset{\text{iid}}{\sim} \mathsf{N}(0, 1)$, and $w_i \sim \mathsf{N}_p(0, I)$ are independent p-dimensional noise vector. We can also write the data in a matrix form

$$X_{n\times p} = c_{n\times 1}\rho^T + \xi W^T,$$

where $c = (c_1, c_2, ..., c_n)^T$ and $W_{p\times n} = [w_1, w_2, ..., w_n]$ with w_i as the ith column.

Given data matrix X, the goal is to identify the non-zero loading set of ρ defined as $I = \{i : \rho_i \neq 0\}$. If the sample covariance matrix $A_{p\times p} = \frac{1}{n}X^TX$, we use a Gaussian approximation as a working likelihood and model A as

$$A_{p\times p} = \mathbf{u}\mathbf{v}^t + E, \tag{4.2.2}$$

where $E_{p\times p} = \{e_{ij}\}$ is the error matrix with $e_{ij} \overset{\text{iid}}{\sim} \mathsf{N}(0, \sigma^2)$, and $\mathbf{u}$, $\mathbf{v} \in \mathbb{R}^p$ with $\|\mathbf{v}\|^2 = 1$ for identifiability.

To derive our variational Bayesian algorithm to select non-zero coefficients in ρ, we put a prior on $\mathbf{v}$ which is uniformly distributed over all vectors with norm-square 1, i.e.,

$$\mathbf{v} \sim \text{Unif}(\mathbf{S}^{p-1}),\ \text{Unif}(\mathbf{S}^{p-1}) = \{x \in \mathbb{R}^p : \|x\| = 1\}.$$

For each u_i, $i = 1, 2, ..., p$, we define a binary vector $\mathbf{Z} = (Z_1, ..., Z_p)$, and each Z_i is a 0/1 Bernoulli variable indicating whether u_i is nonzero. $\mathbf{Z}_i$ itself also has a Bernoulli prior,

$$\pi(Z_i) \sim \mathsf{Bern}(\omega),\ j = 1, 2, ..., p,$$

$$\omega \sim \mathsf{Beta}(s_0, t_0),$$

where ω is a hyper-parameter. Given Z_i, the prior of u_i follows a mixture of two normal distributions,

$$\pi(u_i|Z_i = k) \sim \mathsf{N}(0, a_k\sigma^2),\ k = 0, 1$$

where a_0 and a_1 are tuning parameters and a_0 usually has a tiny value. In our model setting, we assume $a_0 = 0$ meaning that if $\rho_i = 0$, the prior of u_i, $\pi(u_i|Z_i = 0)$ is a point mass at 0, and we write it as $\delta(u_i)$. Notice that σ^2 is contained both in the variance of the prior of u_i and the variance of the error term E in the working likelihood, we put an Inverse Gamma distribution on it,

$$\varrho = \frac{1}{\sigma^2} \sim \mathsf{Gamma}(\nu_0, \lambda_0).$$

To summarize, we have elicited the model setting and all the prior distributions for the parameter space $\mathcal{H} = (u_{1:p}, Z_{1:p}, v_{1:p}, \sigma^2, \omega)$. Among of them, $(\mathbf{u}, \mathbf{Z})$ are variables of interest and $\Theta = (\omega, \sigma^2)$ are hyper-parameters. In next section, we will provide an updating rule with VB algorithm to get all parameters' posterior distributions.

4.2.2 A Variational Algorithm

It's pretty normal that in many real application examples, only the sample covariance matrix is observed instead of the whole data matrix. Therefore, in the retaining part of this chapter, we always assume only matrix A is available. Let $P(\mathbf{u}, \mathbf{Z}|A)$ be the posterior distribution of our interested variables $(\mathbf{u}, \mathbf{Z})$, and it does not have a nice analytical solution based on our model and prior setting. We propose a alternative method to approximate this posterior distribution by a new variational algorithm. Let $Q(\mathbf{u}, \mathbf{Z})$ be the variational distribution, to allow our approximation easy to solve, we assume that each dimension of $(\mathbf{u}, \mathbf{Z})$ is independent, and therefore $Q(\mathbf{u}, \mathbf{Z})$ can be factorized by the product of each dimension's distribution,

$$Q(\mathbf{u}, \mathbf{Z}) = \prod_{i=1}^{p} q_i(u_i, Z_i).$$

Furthermore, we also assume that each dimension's distribution has the same mixture of normal structure as the prior of u_i given Z_i, that is

$$q_i(u_i, Z_i) = \big[\alpha_i f_i(u_i)\big]^{Z_i} \big[(1-\alpha_i)\delta(u_i)\big]^{1-Z_i},$$

where $f_i(u_i)$ is a arbitrary probability density but could be solved through a optimization process on the objective function, and α_i is the parameter of the Binary distribution for Z_i. In order to make $Q(\mathbf{u}, \mathbf{Z})$ as close to the true posterior as possible, we use K-L divergence to define our objective function, i.e.,

$$\Omega(q_1, q_2, ..., q_p, \sigma^2, \omega, \mathbf{v}) = \mathbb{E}^{q_1, q_2, ..., q_p} \log \frac{Q(\mathbf{u}, \mathbf{Z})}{P(\mathbf{u}, \mathbf{Z}, \mathbf{v}, \sigma^2, \omega | A)}, \tag{4.2.3}$$

where $P(\mathbf{u}, \mathbf{Z}, \mathbf{v}, \sigma^2, \omega | A)$ is the whole posterior distribution including both variables of interest $(\mathbf{u}, \mathbf{Z})$ and hyper-parameters $\eta = (\mathbf{v}, \sigma^2, \omega)$. To optimize the objective function, our algorithm iteratively solves Q and η until the stopping criteria satisfies. It can be considered as a hybrid of Expectation-Maximization (EM) and Variational-Bayes (VB): for hyper-parameter η, we use a plug -in method by it's maximum a posteriori probability (MAP) estimator; for $(\mathbf{u}, \mathbf{Z})$, we approximate its posterior by $Q(\mathbf{u}, \mathbf{Z})$.

Now we provide the updating rule of all the parameters for our algorithm, and all the detailed derivations are provided in the Appendix D.

- Update $q_i(u_i, Z_i)$.

  It can be shown that given the MAP estimator $(\widehat{\mathbf{v}}, \widehat{\sigma}^2, \widehat{\omega})$, the optimal choice of $f_i(u_i)$ is a normal distribution $\mathbf{N}(\mu_{u_i}, \sigma^2_{u_i})$, where

$$\mu_{u_i} = \frac{\sum\limits_{j=1:p} A_{ij}\widehat{v}_j}{\frac{1}{a_1} + \sum\limits_{j=1:p} \hat{v}_j^2} = \frac{\sum\limits_{j=1:p} A_{ij}\widehat{v}_j}{\frac{1}{a_1} + 1}$$

$$\sigma^2_{u_i} = \frac{\widehat{\sigma}^2}{\frac{1}{a_1} + \sum\limits_{j=1:p} \hat{v}_j^2} = \frac{\widehat{\sigma}^2}{\frac{1}{a_1} + 1},$$

and the parameter of the posterior distribution of Z_i, $\alpha_i = \mathbb{P}(Z_i = 1)$ satisfies

$$\log \frac{\alpha_i}{1-\alpha_i} = \log \frac{\widehat{\omega}}{1-\widehat{\omega}} + \frac{\mu_{u_i}^2}{2\sigma_{u_i}^2} - \frac{1}{2}\log \frac{a_1\sigma^2}{\sigma_{u_i}^2}.$$

In fact, we can also update μ_{u_i} and $\sigma_{u_i}^2$ simultaneously for all the dimensions. Let $\boldsymbol{\mu}_u = (\mu_{u_1}, ..., \mu_{u_p})^t$ and $\boldsymbol{\sigma}_u^2 = (\sigma_{u_1}^2, ..., \sigma_{u_p}^2)^t$, then

$$\boldsymbol{\mu}_u = \frac{A\widehat{\mathbf{v}}}{\frac{1}{a_1}+1}$$

$$\boldsymbol{\sigma^2}_u = \frac{\sigma^2}{\frac{1}{a_1}+1}\mathbf{1}_{p\times 1},$$

where $\mathbf{1}_{p\times 1} = (1, 1, ..., 1)^t$.

- Update v_j.

  The point estimator $\widehat{\mathbf{v}}$ for square-norm 1 vector $\mathbf{v}$ satisfies

$$\begin{cases} \frac{\left(A^t(\mathbb{E}^Q\mathbf{u})\right)_k}{\widehat{v}_k} = c, \text{ with a constant c} \neq 0 \\ \sum\limits_{k=1:p} \widehat{v}_k^2 = 1 \end{cases}$$

  where $\mathbb{E}^Q\mathbf{u} = \left(\alpha_1\mu_{u_1}, ..., \alpha_p\mu_{u_p}\right)^t$, and $\left(A^t(\mathbb{E}^Q\mathbf{u})\right)_k = \sum\limits_{i=1:p} \alpha_i A_{ik}\mu_{u_i}$ represents the k-th element of the $\mathbb{R}^p$ vector $A^t(\mathbb{E}^Q\mathbf{u})$.

- Update σ^2

$$\widehat{\sigma}^2 = \frac{\mathbb{E}^Q \sum\limits_{i,j}(A_{ij} - u_i\widehat{v}_j)^2 + \frac{1}{a_1}\sum\limits_{i=1:p}\alpha_i(\mu_{u_i}^2 + \sigma_{u_i}^2) + 2}{p^2 + \sum\limits_{i=1:p}\alpha_i + 4},$$

  where

$$\begin{aligned} \mathbb{E}^Q \sum_{i,j}(A_{ij} - u_i\widehat{v}_j)^2 &= \mathbb{E}^Q \sum_{i,j} A_{ij}^2 - 2\mathbb{E}^Q \sum_{i,j} A_{ij}u_i\widehat{v}_j + \mathbb{E}^Q \sum_{i,j} u_i^2\widehat{v}_j^2 \\ &= \sum_{i,j} A_{ij}^2 - 2\sum_{i,j}\alpha_i\widehat{v}_j A_{ij}\mu_{u_i} + \sum_{i,j}\alpha_i\widehat{v}_j^2(\mu_{u_i}^2 + \sigma_{u_i}^2) \end{aligned}$$

- Update ω

$$\widehat{\omega} = \sum_{i=1}^{p} \alpha_i / p$$

We iteratively use the above rule to update all the parameters in a order of $q_i(u_i, Z_i)$, $\mathbf{v}$, σ^2 and ω, and after several loops, we need to decide when to stop. We usually refer to the total entropy of the vector α to check whether the feature selection result converges, i.e.,

$$H(\alpha) = \sum_{i=1}^{p} \Big(- \alpha_i \log \alpha_i - (1 - \alpha_i) \log(1 - \alpha_i) \Big).$$

If the changing rate of entropy $H(\alpha)$ between two iterations is less then a pre-specified value (5% in our simulation), we will stop. Usually, the updating converges pretty quickly in a few iterations.

4.2.3 Parameter tuning

When we apply the above updating rule to get the posterior non-zero probability for each element of the principal component, we use 0.5 as the cut off point. It means the i-th coefficient of ρ is estimated as non-zero if $\alpha_i > 0.5$. Define the non-zero coefficient selection set to be

$$\hat{I} = \{i : \ \alpha_i > 0.5\},$$

we apply PCA on a sub-matrix $S = A_{\hat{I} \times \hat{I}}$ to yield the eigenvector $\widehat{\rho}^S$ corresponding to the largest eigenvalue. Then the solution $\widehat{\mathbf{u}}$ is considered as the sparse principal component estimation with each element

$$\widehat{u}_i = \begin{cases} \widehat{\rho}_i^S, \text{ if } i \in \hat{I} \\ 0, \text{ if } i \notin \hat{I} \end{cases}.$$

Notice that α_i is solved based on the tuning parameter a_1 and therefore, $\hat{I}$ and $\widehat{\mathbf{u}}$ both depend on a_1, i.e., different chosen value of a_1 will lead to a different non-zero coefficient pool defined as $\{\widehat{\mathbf{u}}^{(1)}, ..., \widehat{\mathbf{u}}^{(k)}\}$, and we should provide a method to determine a suitable value of a_1.

We consider first provide a range of candidate values for a_1, and apply BIC as the criteria to determine the best choice among all the candidates, i.e., the best solution is chosen to minimize $\text{BIC}(\widehat{\mathbf{u}}^{(i)})$, for $i = 1, 2, ..., k$. In our simulation study, we set the tuning range for a_1 from 0 to 2 with 0.1 as a gap between two contiguous candidates, and we derive the BIC formula below by assuming

only the sample covariance matrix A is known,

$$\begin{aligned} \text{BIC}(\widehat{\mathbf{u}}^{(i)}) &= -2\text{loglike} + \text{df}_{\widehat{\mathbf{u}}^{(i)}} \log(np) \\ &= np \log(\widehat{\sigma}^2) + \text{df}_{\widehat{\mathbf{u}}^{(i)}} \log(np) + C, \end{aligned} \tag{4.2.4}$$

where $\widehat{\sigma}^2 = \frac{\text{Tr}\left(A(I - \widehat{\mathbf{u}}^{(i)}\widehat{\mathbf{u}}^{(i)t})\right)}{p}$ and C is a constant not affecting the choice of $\widehat{\mathbf{u}}^{(i)}$.

If we solve multiple sparse principal components problem, i.e., when there are p principal components $\rho_1, \rho_2, ..., \rho_p$, we try to solve $\widehat{\mathbf{u}}_1, ..., \widehat{\mathbf{u}}_p$ to estimate ρ_i respectively, it's easy to find that all the $\widehat{\mathbf{u}}_i$ are not orthogonal which is different from the ordinary principal component solution. Zou et al. (2006) pointed out a way to calculated cumulated explained variance for sparse PCA by projecting the new sparse direction onto the orthogonal complement space of the previous sparse PCs. We can apply the similar idea to define our generalized BIC criteria. If we denote the projection matrix for the orthogonal complement space of the previous $k-1$ directions by H_{k-1}, the new covariance matrix we are dealing with for k-th direction is $A_k = H_{k-1}AH_{k-1}^t$. If we yield a sparse solution $\widehat{\mathbf{u}}$ for covariance matrix A_k, we should also project it onto H_{k-1} space to get $\widehat{\mathbf{u}}_k = \frac{H_{k-1}\widehat{\mathbf{u}}_k}{\|H_{k-1}\widehat{\mathbf{u}}_k\|}$, and BIC value could be calculated afterwards through

$$\text{BIC}(\widehat{\mathbf{u}}_k) = np \log\Big(\frac{\text{Tr}[(A(I - \widehat{\mathbf{u}}_k\widehat{\mathbf{u}}_k^T)]}{p}\Big) + \text{df}_{\widehat{\mathbf{u}}_k} \log(np) + C. \tag{4.2.5}$$

Once we solve $\widehat{\mathbf{u}}_k$, we should update the new orthogonal complement space with the current k directions by $H_k = H_{k-1} - \widehat{\mathbf{u}}_k\widehat{\mathbf{u}}_k^T$, and H_k will be served for solving the $k+1$ sparse PC estimation $\widehat{u}_{k+1}$.

To sum up, we provide the pseudo code for our Variational Bayesian sparse PCA algorithm with BIC embedding for parameter tuning, and it is named sPCA-VB-BIC for later on usage.

Algorithm 4.1 sPCA-VB-BIC

Require: Initialize $\widehat{\mathbf{v}} = (\widehat{v}_1, ..., \widehat{v}_p)$, $\widehat{\sigma}^2$, $\widehat{\omega}$

1: **for** $a_1^{(k)}$ in tuning candidate pool $a_1^{(1)}$, ..., $a_1^{(m)}$ **do**

2: **Repeat**

3: **for** i in 1:p **do**

4: $\mu_{u_i} \leftarrow \frac{\sum_{j=1:p} A_{ij}\hat{v}_j}{\frac{1}{a_1^{(j)}}+1}$

5: $\sigma^2_{u_i} \leftarrow \frac{\hat{\sigma}^2}{\frac{1}{a_1^{(j)}}+1}$

6: $\alpha_i \leftarrow \text{Logit}^{-1}\Big(\log\frac{\hat{\omega}}{1-\hat{\omega}} + \frac{\mu^2_{u_i}}{2\sigma^2_{u_i}} - \frac{1}{2}\log\frac{a_1^{(j)}\sigma^2}{\sigma^2_{u_i}}\Big)$.

7: **end for**

8: $\widehat{\mathbf{v}} \leftarrow \frac{A^t(\mathbb{E}^q\mathbf{u})}{\|A^t(\mathbb{E}^q\mathbf{u})\|}$

9: $\widehat{\omega} \leftarrow \sum_{i=1}^{p} \alpha_i/p$

10: $\widehat{\sigma}^2 \leftarrow \frac{\mathbb{E}^q \sum_{i,j}(A_{ij}-u_i\hat{v}_j)^2 + \frac{1}{a_1}\sum_{i=1:p}\alpha_i(\mu^2_{u_i}+\sigma^2_{u_i})+2}{p^2+\sum_{i=1:p}\alpha_i+4}$

11: **Until** $H(\alpha) = \sum_{i=1}^{p}\Big(-\alpha_i\log\alpha_i - (1-\alpha_i)\log(1-\alpha_i)\Big)$ converges

12: **Calculate** $\widehat{\mathbf{u}}^{(k)}$ and BIC($\widehat{\mathbf{u}}^{(k)}$)

13: **end for**

14: $K \leftarrow \arg\min_{j=1:m}$ BIC($\widehat{\mathbf{u}}^{(j)}$)

Ensure: $\widehat{\mathbf{u}}^{(K)}$

4.2.4 Two-stages Method

When p is very large, directly applying sPCA-VB-BIC is time consuming. Johnstone and Lu (2009) proposed an algorithm including a thresholding step which can reduce the number of variables before embarking on PCA. Here we extend sPCA-VB-BIC algorithm to a two-stage sparse PCA method with a similar idea:

- First step: Define $\widehat{\sigma}^2_\nu$ as the diagonal value of the sample covariance matrix A, and we apply a simple thresholding rule on $\widehat{\sigma}^2_\nu$ to filter the elements large than a prefixed value,

$$S = \{\nu \ : \ \widehat{\sigma}^2_\nu \geq \widehat{\sigma}^2(1+\alpha_n)\},$$

where $\widehat{\sigma}^2 = \text{median}(\widehat{\sigma}^2_\nu)$, $\alpha_n = c\sqrt{\frac{\log\big(\max(p,n)\big)}{n}}$, and c is a positive constant usually with value

larger than $\sqrt{12}$.

- Second step: Let $A_S = (A_{i,j} : i, j \in S)$ be the sample covariance matrix of the selected variables from first step, we apply sPCA-VB-BIC algorithm on the matrix A_S to get a sparse PC estimation $\widehat{\rho}_\nu$, $\nu \in S$. The final estimation for each element of ρ yields to

$$\widehat{\rho}_{A,\nu} = \begin{cases} \widehat{\rho}_v, \text{ if } \nu \in S \\ 0, \text{ if } \nu \notin S \end{cases}$$

4.3 Selection and Consistency

We talk about the selection consistency of our algorithm in this section. For simplicity, we still use our single component data generation process (4.2.1) and working likelihood model (4.2.2), and we assume the principal component has a sparse representation. We establish a consistency property of our VB algorithm which could capture the sparse pattern under certain conditions. A low dimension case (p fixed) is first considered and we generate the result to a case which $p \to \infty$ and $p/n \to 0$. In the latter case setting, we also assume $\liminf \|\rho\| = \varrho > 0$.

4.3.1 Asymptotic consistency when p is fixed

Notice that I denotes the support set of the true principal component ρ, i.e., $I = \{i : \rho_i \neq 0\}$, then we define $I^c = \{i : \rho_i = 0\}$ and $k = \#\{i : \rho_i \neq 0\}$. The major rule we use to estimate this support set is calculating the posterior probability $\alpha_i = \mathbb{P}(Z_i = 1|A)$ through the Logit formula $\log \frac{\alpha_i}{1-\alpha_i}$ in updating rule of $q_i(u_i, Z_i)$. In algorithm implementation, we always add a truncation step in each iteration to avoid numerical computation issue, i.e., if $\log(\frac{\alpha_i}{1-\alpha_i}) > c$ or $\log(\frac{\alpha_i}{1-\alpha_i}) < -c$, we will not update α_i in this iteration since α_i is close to either 0 or 1 which are both easy to determine the value of Z_i. Here, c is a large enough positive number, for instance $c = 100$ in our numerical study. In the following part, we use a sample dependent value $c(n)$ to be this cutoff. Therefore, we could define our estimated support set $\widehat{I} = \{i : \log \frac{\alpha_i}{1-\alpha_i} \geq c(n)\}$. A algorithm has selection consistency property if it satisfies $\mathbb{P}(\widehat{I} = I) \to 1$.

Before providing the main theorem, we need to derive several properties. We rely on a single component model and the true sample covariance matrix A can be written as

$$A = \frac{1}{n}X^T X = \frac{||c||^2}{n}\rho\rho^T + \frac{\xi}{n}(\rho c^T W^T + Wc\rho^T) + \xi^2\frac{WW^T}{n}.$$

Its element-wise expectation is equal to

$$A_0 = \mathbb{E}A = \rho\rho^T + \xi^2 I_p,$$

where A_0 is the true covariance matrix. The difference of these two matrices is treated as an true error matrix $E = \{e_{ij}\}$, which can be decomposed into three parts,

$$\widetilde{E} = A - A_0 = \Big(\frac{||c||^2}{n} - 1\Big)\rho\rho^T + \frac{\xi}{n}(\rho c^T W^T + Wc\rho^T) + \xi^2\Big(\frac{WW^T}{n} - I_p\Big) := B + C + D.$$

Meanwhile, the true covariance matrix A is modeled by a working likelihood,

$$A_{p\times p} = \mathbf{u}\mathbf{v}^t + E,$$

where $E_{p\times p} = \{e_{ij}\}$ is a model based error matrix with $e_{ij} \overset{\text{iid}}{\sim} \mathsf{N}(0, \sigma^2)$. In order to perform a good estimation of the true model, we need to make an assumption on the working likelihood mode. We claim the following property for each element in E so that it's natural to assume the order of $\sigma^2 := \sigma^2(n) \sim \frac{1}{n}$ from now on.

Proposition 4.1. *Let $\widetilde{E}_{p\times p} = \{\widetilde{e}_{ij}\}$, then $Var(\widetilde{e}_{ij}) = O_p(\frac{1}{n}),\ \forall i, j$.*

Proof: Consider three parts in $\widetilde{E}$ separately. For B term, $\frac{\|c\|^2}{n} - 1 = \frac{1}{n}\sum_i (c_i^2 - 1)$, $(c_i^2 - 1)$'s are i.i.d. with mean 0 and variance 2. By central limit theorem (CLT),

$$\sqrt{n}(\frac{\|c\|^2}{n} - 1) \xrightarrow{\mathcal{D}} \mathsf{N}(0, 2).$$

The dimension p is fixed here, and the (i, j)-th element $B_{i,j}$ converges to a normal distribution,

$$\sqrt{n}B_{i,j} \xrightarrow{\mathcal{D}} \mathsf{N}(0, 2\rho_i^2\rho_j^2).$$

For C term, conditioned on c, the $\mathbb{R}^p$ vector Wc is distributed as $\mathsf{N}_p(0, \|c\|^2 I_p)$, and it leads to

$$\frac{\xi}{n}Wc \sim \mathsf{N}_p(0, \frac{\|c\|^2\xi^2}{n^2}I_p) \xrightarrow{\mathcal{D}} \mathsf{N}_p(0, \frac{\xi^2}{n}I_p).$$

Therefore, the (i,j)-th element of $C_{i,j}$ converges to a summation of two normal distributions,

$$\sqrt{n}C_{i,j} \xrightarrow{\mathcal{D}} \mathsf{N}(0, \xi^2\rho_i^2) + \mathsf{N}(0, \xi^2\rho_j^2).$$

For the third term D, the (i,j)-th element

$$D_{ij} = \begin{cases} \frac{\xi^2}{n}\sum_{k=1}^{n}(w_{ik}^2 - 1), \text{ if } i = j \\ \frac{\xi^2}{n}\sum_{k=1}^{n} w_{ik}w_{jk}, \text{ if } i \neq j \end{cases}$$

where w_{ik} is the k-th element of $w_i \overset{i.i.d}{\sim} \mathsf{N}_p(0, I)$. Using CLT,

$$\sqrt{n}D_{ij} \xrightarrow{\mathcal{D}} \begin{cases} \xi^2\mathsf{N}(0,2), \text{ if } i = j \\ \xi^2\mathsf{N}(0,1), \text{ if } i \neq j \end{cases}$$

Combining three terms, we claim that $\sqrt{n}\widetilde{e}_{ij} = \sqrt{n}\Big(B_{i,j} + C_{i,j} + D_{i,j}\Big)$ converges to a summation of serval normal distributions and each of them has a finite variance. Therefore, $\mathrm{Var}(\widetilde{e}_{ij}) = O_p(\frac{1}{n})$.

□

Let $\rho^* = \rho/\|\rho\|$ be the norm one true leading principle component of A_0. This is the eigenvector corresponding to the largest eigenvalue $\lambda = \|\rho\| + \xi^2$ of A_0. When implementing the algorithm, we usually choose the eigenvector $\widehat{\mathbf{v}}$ corresponding to the largest eigenvalue of the sample covariance matrix A as the initial value for $\mathbf{v}$. Thus, we are interested in how different between the ρ^* and $\widehat{\mathbf{v}}$ because we would like to make the starting point close to the truth. By the perturbation bounds theorem from Johnstone and Yu (2004), we have the following property.

Proposition 4.2. $\|\widehat{\mathbf{v}} - \rho^*\| = O_p(\frac{1}{\sqrt{n}})$.

Proof: The second largest eigenvalue of A_0 is ξ^2, and the gap between the first and second largest eigenvalue equals to $\delta = \|\rho\|$. Then if $\|\widetilde{E}\|_2 \leq \frac{\delta}{5} = \frac{\|\rho\|}{5}$, there exists $r \in \mathbb{R}^{p-1}$ such that

$$\angle(\rho^*, \widehat{\mathbf{v}}) \leq \frac{4}{\delta}\|\widetilde{E}\|_2$$

and

$$\limsup \|\widetilde{E}\|_2 \leq \xi\sqrt{\frac{p}{n}}\varrho + \xi^2(\frac{p}{n} + 2\sqrt{\frac{p}{n}}).$$

Since $\frac{1}{2}\|\widehat{\mathbf{v}} - \rho^*\| = \sin\left(\frac{1}{2}\angle(\rho^*, \widehat{\mathbf{v}})\right)$, when x goes to 0, $\sin(x) \simeq x$. Therefore, we have

$$\|\widehat{\mathbf{v}} - \rho^*\| = 2\sin\left(\frac{1}{2}\angle(\rho^*, \widehat{\mathbf{v}})\right) \simeq \angle(\rho^*, \widehat{\mathbf{v}}) \leq \frac{4}{\|\rho\|}\|\widetilde{E}\|_2 = O(\frac{1}{\sqrt{n}})$$

□

To show our algorithm works for selection consistency, we provide the following Theorem 4.1. It says if the initial values $\widehat{\omega}, \widehat{\sigma}^2(n) and \widehat{\mathbf{v}}$ as well as the tuning parameter a_1 are properly chosen, and we only update all the parameters once, we could have $\mathbb{P}(\widehat{I} = I) \to 1$, i.e.,

$$\mathbb{P}\Big(\min_{i\in I} \log\frac{\alpha_i}{1-\alpha_i} > c(n) \text{ and } \max_{i\in I^c} \log\frac{\alpha_i}{1-\alpha_i} < -c(n)\Big) \to 1$$

with a certain condition on $c(n)$.

Theorem 4.1. *Suppose p is fixed and $n \to \infty$, we set initial values of $\omega, \sigma^2(n)$ and $\mathbf{v}$ as $\omega^{(0)} \in (0,1)$, $\sigma^{(0)}(n) = O_p(\frac{1}{\sqrt{n}})$ and $\mathbf{v}^{(0)} = \widehat{\mathbf{v}}$. Let $c(n) = \theta\log(a_1)$ where $\theta \in (0, 1/2)$, if $a_1 \to \infty$ and $\log(a_1) = o(n)$, we have $\mathbb{P}(\widehat{I} = I) \to 1$.*

The proof relies on Proposition 4.2 and it is provided in the Appendix D.

4.3.2 Asymptotic consistency when $p \to \infty$ and $p/n \to 0$

When $p \to \infty$, we need to consider the order of variance σ^2 with respect to p as well. If we check the element-wise maximum of the true error matrix $\widetilde{E}$, we have a property mentioned in Lei and Vu (2015) that for a large enough constant C not related to n and p,

$$\mathbb{P}(\max_{i,j}|e_{ij}| \geq C\sqrt{\frac{\log p}{n}}) \leq 2p^{-2}.$$

Therefore, the upper bound order $\sqrt{\frac{\log p}{n}}$ controls the true error's magnitude which can be used to set the order of σ^2, i.e., $\sigma = o_p(\sqrt{\frac{\log p}{n}})$.

The next theorem provides all the initial values and other required conditions to achieve the selection consistency for $p \to \infty$ and $p/n \to 0$ case. Most of the conditions are similar to the fixed p

case except for the minimum signal condition and the order of σ^2. In the minimum signal condition, we also need to consider about a value k which is the total number of true signal elements in ρ.

Theorem 4.2. *Suppose p, $n \to \infty$ and $p/n \to 0$, we set initial values of $\omega, \sigma^2(n)$ and $\mathbf{v}$ as $\omega^{(0)} \in (0,1)$ and $\sigma^{(0)}(n) = O_p(\sqrt{\frac{\log p}{n}})$, $\mathbf{v}^{(0)} = \widehat{\mathbf{v}}$. Let $c(n) = \theta \log(a_1)$ where $\theta \in (0, 1/2)$, if $a_1 \to \infty$ and $\frac{\min_{i \in I}(\rho_i^*)^2}{\max(k, \log(a_1)) \log p/n} \to \infty$, we have $\mathbb{P}(\widehat{I} = I) \to 1$.*

The proof of Theorem 4.2 is different from Theorem 4.1. We need first consider a special case when $k = 1$, i.e., $\rho^* = (1, 0, ..., 0)^t$. The reason is that we can use a property from Paul (2007) that $\widehat{\mathbf{v}}_2/\|\widehat{\mathbf{v}}_2\|$ is distributed uniformly on the unit sphere $\mathbf{S}^{p-2}$, where $\widehat{\mathbf{v}} = (\widehat{v}_1, \widehat{\mathbf{v}}_2^t)^t$. This helps us to understand the performance of the error elements in the sample covariance matrix. Then, we could apply the consistency result to the common case by an orthogonal transformation.

4.4 Numerical Results

4.4.1 Three-peaks single principal component

Our first synthetic example was designed in Johnstone and Lu (2009). They also provided a sparse PCA method (AsPCA) including a pre-transformation on the data with a wavelet basis, and we only compare our method to AsPCA by using the same wavelet basis transformation. Meanwhile, we also check the result without a transformation by comparing the performance with Sparse PCA by Zou, Hastie and Tibshirani (2006).

The example contains a three-peak principal component ρ in $\mathbb{R}^p$ shown in Figure 4.1 with $p = 2048$. We set the i-th component of ρ by

$$\rho_i = f(i/p) \text{ for } i = 1, 2, ..., p$$

where

$$f(t) \propto 0.7\mathsf{B}\big(1500, 3000\big)(t) + 0.5\mathsf{B}\big(1200, 900\big)(t) + 0.5\mathsf{B}\big(600, 160\big)(t)$$

and $\mathsf{B}\big(a, b\big)(t)$ is a Beta density with parameter (a, b). We generate $n = 1024$ data points $x_i \in \mathbb{R}^p$ by

$$x_i = c_i\rho + w_i, \ i = 1, 2, ..., n$$

where $c_i \overset{\text{iid}}{\sim} \mathsf{N}(0,1)$, and $w_i \sim \mathsf{N}_p(0, I)$ are independent p-dimensional noise vector. We also scale ρ to satisfy $\|\rho\| = 10$.

For each element in ρ, we treat it 0 if its absolute value is less than 10^{-2} and we set the number of non-zero coefficients to be the true **sparse degree** for ρ. This sparse degree is used when applying the Sparse PCA. We also generate a subset of the sample covariance matrix which contains rows and columns with the true sparse degree id. Then we apply standard PCA on this subset covariance matrix and the solution is set to be the non-zero coefficients estimation for ρ. And we also set the estimations to be 0 for the zero loadings in ρ. This is an oracle method using the information not able to get based on the sample covariance matrix, and we name it sPCA-Oracle only for comparison purpose.

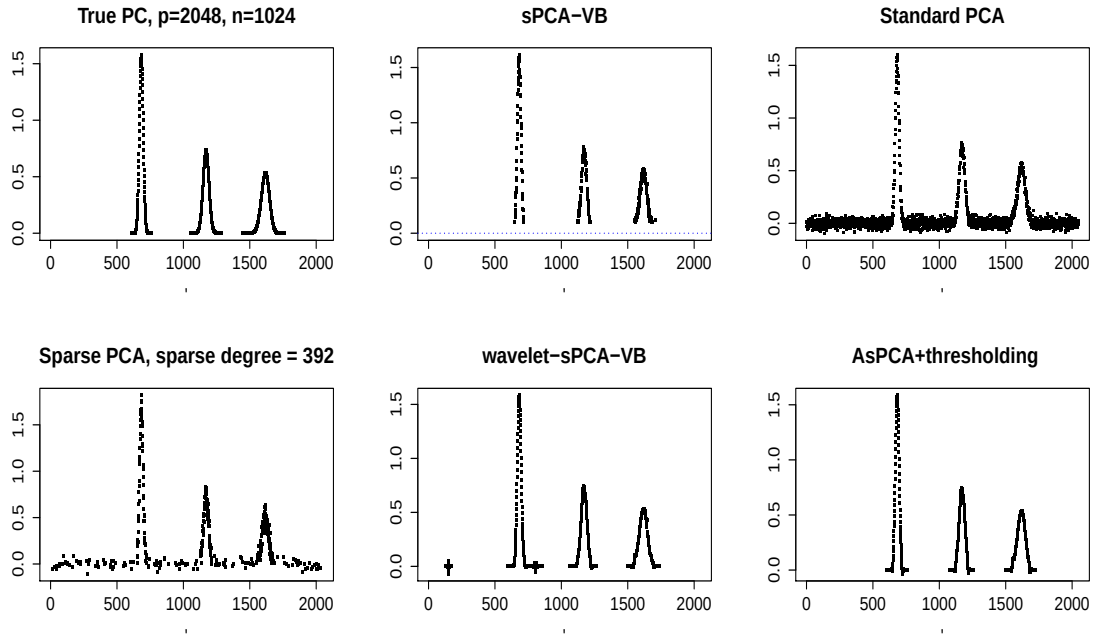

Figure 4.1: Single principal component for a step function example. (a) Single component ρ. (b) Sample principal component by sPCA-VB. (c) Sample principal component by Standard PCA. (d) Sample principal component by Sparse PCA using sparse degree 392. (e) Sample principal component by wavelet-sPCA-VB. (f) Sample principal component by AsPCA + thresholding.

Figure 4.1 provides the estimation $\widehat{\rho}$ by different methods. The sPCA-VB, standard PCA and Sparse PCA deal with the data without a wavelet transformation, and wavelet-sPCA and ASPCA involve finding a wavelet basis to make data more sparse before applying a particular algorithm. Comparing to the true ρ with 392 non-zero coefficients, all methods successfully obtain the three-

peaks pattern of the component. Standard PCA is not a sparse method with fluctuations on all the noise directions. Sparse PCA still can not detect some noise directions given the true degree of sparsity. sPCA-VB successfully filters out most of the noise terms, and even small values of ρ_i are also shrunk to 0. Among all these three methods, sPCA-VB seems to have the best estimation. On the other hand, for the other two transformation based methods, wavelet-sPCA and AsPCA could get a even better estimation to the truth. If we use average squared error (ASE)

$$\text{ASE} = \frac{\|\hat{\rho} - \rho\|}{p}$$

to measure the estimation accuracy, we provide some numerical results in Table 4.1 and Figure 4.2. These results are the ASE based on 50 iterations for different methods. Overall, sPCA-VB gives a better prediction accuracy (smaller ASE) on average than Sparse PCA, standard PCA, and it's also not far from the ASE benchmark provided by sPCA-Oracle. Meanwhile, if we apply a wavelet transformation before using sPCA-VB, our result is also better than that calculated by AsPCA with and without thresholding.

Non-wavelet	Standard PCA	Sparse PCA (Zou)	sPCA-Oracle	sPCA-VB
ASE	6.9e-4	8.7e-4	3.1e-4	3.9e-4
Wavelet	wavelet-sPCA-VB	AsPCA	AsPCA+Threshold	
ASE	1.7e-4	4.1e-4	2.3e-4	

Table 4.1: ASE with 50 iterations using different PCA methods with and without wavelet pre-transformation

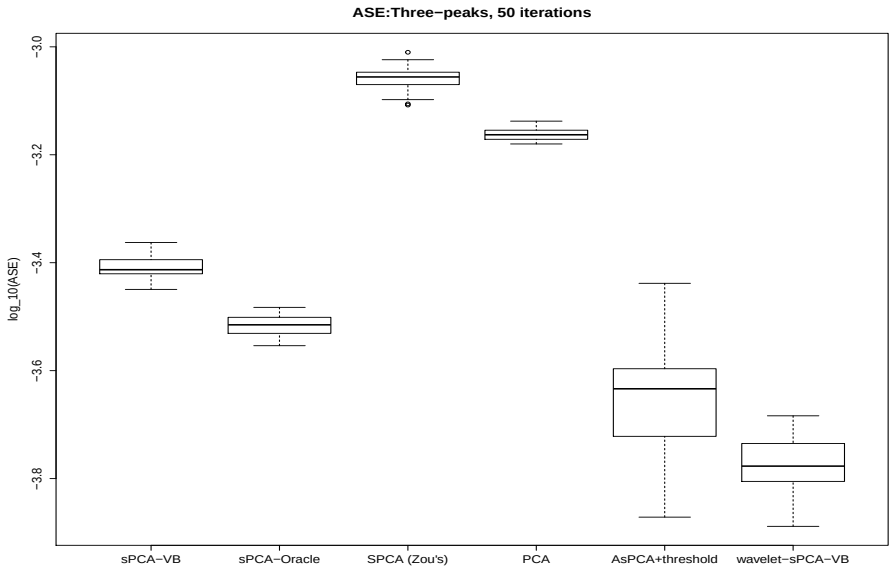

Figure 4.2: Boxplots of ASE with 50 iterations using different PCA methods for three-peak single principal component example

4.4.2 Two sparse principal components

Next we consider an example proposed in Shen and Huang (2008). There is an example of a covariance matrix Σ_1 with two sparse leading eigenvectors in $\mathbb{R}^{10}$,

$$\mathbf{v}_1 = (1,1,1,1,0,0,0,0,0.9,0.9)^T,\ \mathbf{v}_2 = (0,0,0,0,1,1,1,1,-0.3,0.3)^T.$$

We make Σ_1's 10 eigenvalues equal to 200, 100, 50, 50, 6, 5, 4, 3, 2, 1. Then each data point is a $\mathbb{R}^{10}$ vector generated from $\mathsf{N}(\mathbf{0}, \Sigma_1)$. We repeat our experiment 100 times with number of data points $n = 30$ and 300 respectively for each time. To determine the sparse degree (number of non-zero coefficients) for estimation $\widehat{\mathbf{v}}_1$ and $\widehat{\mathbf{v}}_2$ got through sPCA-VB-BIC, we try 20 different values (0.1, 0.2, ..., 2) as a candidate pool to tune parameter a_1. The reports of the performance by our method as well as the method introduced in Shen and Huang (2008) are shown in Table 4.2. Shen and Huang (2008) mainly proposed three different methods and their results were similar. We only consider one of the method sPCA-rSVD-SCAD with cross validation approach to select their tuning parameters.

In Table 4.2, estimation performance is measured by three aspects: the median angles between the estimation and true eigenvectors, the percentage of correctly / incorrectly identified zero loadings

Method	$\widehat{\mathbf{v}}_1$			$\widehat{\mathbf{v}}_2$		
	Median angle	Correct (%)	Incorrect (%)	Median angle	Correct (%)	Incorrect (%)
n=30						
sPCA-rSVD-SCAD-CV	10.68	45.25	2.50	22.40	43.25	12.83
sPCA-VB-BIC	10.14	88.75	0.67	15.15	68.75	17.33
n=300						
sPCA-rSVD-SCAD-CV	2.83	74.75	0.00	5.90	57.25	1.33
sPCA-VB-BIC	2.82	100.00	0.00	1.84	99.25	5.5

Table 4.2: Two sparse principal components example with $p = 10$ and $n = 30, 300$. Median angles between estimated $\widehat{\mathbf{v}}_1$ and $\mathbf{v}_1$, $\widehat{\mathbf{v}}_2$ and $\mathbf{v}_2$ in degree, percentage of correctly / incorrectly identified zero coefficients.

of the true eigenvectors. Overall, both methods have a better performance when n is large. Our method tends to have a more sparse solution for $\widehat{\mathbf{v}}_2$ and therefore it has slightly larger incorrect identified zeros percentage than sPCA-rSVD-SCAD-CV. Moreover, our method has much higher correctly identified zeros percentage and the median angle for $\widehat{\mathbf{v}}_1$ and $\widehat{\mathbf{v}}_2$ are also smaller, especially for the second direction.

4.4.3 High dimension low sample size setting

In this experiment. we consider a high dimension low sample size case which was also designed in Shen and Huang (2008). Data matrix $X_{n\times p}$ with $n = 50$ and $p = 500$ is generated from $\mathsf{N}(\mathbf{0}, \Sigma_2)$, where the covariance matrix Σ_2 has two leading sparse eigenvectors in $\mathbb{R}^{500}$

$$v_{1i} = \begin{cases} 1 & i = 1, ..., 10 \\ 0 & i = 11, ..., 500 \end{cases}, \quad v_{2i} = \begin{cases} 1 & i = 11, ..., 20 \\ 0 & i = 1, ..., 10, 21, ..., 500 \end{cases},$$

and both of them have 10 non-zero coefficients. We make Σ_2's first two eigenvalues $c_1 = 400$ and $c_2 = 300$, and the remaining eigenvalues are all set to 1, i.e., $c_k = 1$ for $k = 3, ..., 500$. Then we repeat the experiment 100 times and acquiring the estimation $\widehat{\mathbf{v}}_1$ and $\widehat{\mathbf{v}}_2$ through sPCA-VB-BIC and sPCA-rSVD-SCAD-CV in Shen and Huang (2008) for each time. When implementing our algorithm, for tuning parameter a_1, we still try 20 candidates from 0.1 to 2 with a gap 0.1 between each two contiguous values. Meanwhile, due to the large value of p, we apply the two-stages method for our algorithm as well.

In Table 4.3, we compare the results of sPCA-VB-BIC and sPCA-rSVD-SCAD-CV. The perfor-

mance metrics are the same as the previous section. Overall, both of the methods have very high percentages of correctly identified zero loadings for both $\widehat{\mathbf{v}}_1$ and $\widehat{\mathbf{v}}_2$. Notice that for the two-stages methods, the thresholding on the first stage always keeps the first 20 directions which are the true signals, and the second step will detect the number of non-zero coefficients either 10 or 20 when estimating $\mathbf{v}_1$ and $\mathbf{v}_2$ separately. Since c_1 and c_2 are pretty close in our setting, it could happen that $\widehat{\mathbf{v}}_1$ and $\widehat{\mathbf{v}}_2$ flip in such that $\widehat{\mathbf{v}}_1$ becomes an estimation of $\mathbf{v}_2$. That makes the incorrectly identified zeros percentage worse than the correctly identified zeros percentage. If we directly apply sPCA-VB-BIC without thresholding, it also could identify the first 20 directions as non-zero in the estimation of $\widehat{\mathbf{v}}_1$ and $\widehat{\mathbf{v}}_2$ for most of the time. In summary, our sPCA-VB-BIC approach has the lowest incorrectly identified zeros percentage and slightly highest median angle for the first principal component estimation. And the most valuable advantage by using two-stages sPCA is to reduce the computational time a lot without losing high estimation accuracy.

Method	$\widehat{\mathbf{v}}_1$			$\widehat{\mathbf{v}}_2$		
	Median angle	Correct (%)	Incorrect (%)	Median angle	Correct (%)	Incorrect (%)
sPCA-rSVD-SCAD-CV	2.05	98.85	10.30	1.85	98.88	11.90
sPCA-VB-BIC	4.90	99.03	4.90	1.84	99.19	5.10
sPCA-VB-2stage-BIC	1.44	99.17	7.80	1.74	99.17	7.90

Table 4.3: High dimension low sample size example with $p = 500$ and $n = 50$. Median angles in degree between estimated $\widehat{\mathbf{v}}_1$ and $\mathbf{v}_1$, $\widehat{\mathbf{v}}_2$ and $\mathbf{v}_2$, percentage of correctly / incorrectly identified zero coefficients are calculated for comparison between different methods.

Appendix A

Supplementary Material for Chapter 1

A.1 Proof of Preposition 1.1

Proof: Let $H_\alpha = \alpha\alpha^t$ be the projection matrix onto the α direction, then

$$\begin{aligned} \|S - \alpha\beta^t\|^2 &= \|\big[H_\alpha + (I - H_\alpha)\big](S - \alpha\beta^t)\|^2 \\ &= \|H_\alpha(S - \alpha\beta^t)\|^2 + \|(I - H_\alpha)(S - \alpha\beta^t)\|^2 \\ &= \|\alpha\alpha^t S - \alpha\beta^t\|^2 + \|(I - H_\alpha)S\|^2 \end{aligned} \tag{A.1.1}$$

Given α, the solution of β comes from minimizing the first term of equation (A.1.1) and that is $\alpha\alpha^t S - \alpha\beta^t = \mathbf{0}$. So $\beta = S\alpha$. Hence, with $\beta = S\alpha$, solving

$$\arg\min_\alpha \|S - \alpha\alpha^t S\|^2.$$

We first write $S = UDU^t$ where $U = (u_1, u_2, ..., u_p)$ is a $p \times p$ orthogonal matrix and $u_1, u_2, ..., u_p$ are not only a basis in $\mathbb{R}^p$ but also the eigen vectors of S. Meanwhile D is a $p \times p$ diagonal matrix with $\lambda_1, \lambda_2, ..., \lambda_p$ on its diagonal, without lost of generality, we just assume these terms are ordered and λ_1 is the largest one.

Then α can be written as a linear combination of $u_1, u_2, ..., u_p$, i.e., $\alpha = a_1u_1 + a_2u_2 + ... + a_pu_p =$

$U\gamma$, where $\gamma = (a_1, a_2, ..., a_p)^t$. Thus,

$$\begin{aligned}
\|S - \alpha\alpha^t S\|^2 &= \|UDU^t - U\gamma\gamma^t U^t UDU^t\|^2 \\
&= \|UDU^t - U\gamma\gamma^t DU^t\|^2 \\
&= \|U(I - \gamma\gamma^t)DU^t\|^2 \\
&= \|(I - \gamma\gamma^t)D\|^2 \qquad \text{(A.1.2)} \\
&= \|\big(\lambda_1(I - \gamma\gamma^t)_1, \lambda_2(I - \gamma\gamma^t)_2, ..., \lambda_p(I - \gamma\gamma^t)_p\big)|^2 \qquad \text{(A.1.3)} \\
&= \sum_{i=1}^{p} \|\lambda_i(I - \gamma\gamma^t)_i\|^2
\end{aligned}$$

Notice that we have equation (A.1.2) since L_2 norm of a matrix keeps the same by timing orthogonal matrix. In equation (A.1.3), $(I - \gamma\gamma^t)_i$ stands for the i-th column of matrix $I - \gamma\gamma^t$. Since

$$\begin{aligned}
\|\lambda_i(I - \gamma\gamma^t)_i\|^2 &= \|\lambda_i(e_i - a_i\gamma)\|^2 \qquad \text{(A.1.4)} \\
&= \lambda_i^2(e_i - a_i\gamma)^t(e_i - a_i\gamma) \\
&= \lambda_i^2(1 - a_i\gamma^t e_i - a_i e_i^t\gamma + a_i^2\gamma^t\gamma \qquad \text{(A.1.5)} \\
&= \lambda_i^2(1 - a_i^2 - a_i^2 + a_i^2) \\
&= \lambda_i^2(1 - a_i^2)
\end{aligned}$$

where in equation (A.1.4), $e_i = (0, 0, ..., 1, 0, ..., 0)^t$ is a vector in $\mathbb{R}^p$ with all elements equal 0 except for the i-th one. And in equation (A.1.5) we use the fact $\|\alpha\|^2 = 1$, and $\|a_1u_1 + a_2u_2 + ... + a_pu_p\|^2 = \sum_{i=1}^{p} a_i^2 = \gamma^t\gamma = 1$.

Therefore, $\arg\min_\alpha \|S - \alpha\alpha^t S\|^2 = \arg\min_\alpha \sum_{i=1}^{p} \lambda_i^2(1 - a_i^2)$, and it has the minimum value when all $a_i = 0$ except for $a_1 = 1$ since λ_1 is the largest eigen value. Finally, $\alpha = a_1u_1 + a_2u_2 + ... + a_pu_p = u_1$ which is the eigen vector corresponding to S's largest eigen value λ_1 and $\beta = S\alpha = UDU^t u_1 = \lambda_1 u_1$ so that $\frac{\beta}{\|\beta\|} = u_1$.

A.2 Proof of Preposition 1.2

Proof: For fixed α_0, if we can show $\|S - \alpha_0\beta^t\|^2 = \|S\alpha_0 - \beta\|^2 + \phi(\alpha_0)$, where $\phi(\alpha_0)$ is only a function of α_0, we finish the proof.

$$
\begin{aligned}
\|S - \alpha_0\beta^t\|^2 &= \|\big[H_{\alpha_0} + (I - H_{\alpha_0})\big](S - \alpha_0\beta^t)\|^2 \\
&= \|H_{\alpha_0}(S - \alpha_0\beta^t)\|^2 + \|(I - H_{\alpha_0})(S - \alpha_0\beta^t)\|^2 \\
&= \|H_{\alpha_0}S - \alpha_0\beta^t\|^2 + \|(I - H_{\alpha_0})S\|^2 \\
&= \sum_{j=1}^{p} \|H_{\alpha_0}S_j - \beta_j\alpha_0\|^2 + \|(I - H_{\alpha_0})S\|^2 \qquad \text{(A.2.1)} \\
&= \sum_{j=1}^{p} \big[\ \|H_{\alpha_0}S_j\|^2 - 2S_j^t H_{\alpha_0}^t \beta_j\alpha_0 + \beta_j^2\|\alpha_0\|^2\ \big] + \|(I - H_{\alpha_0})S\|^2 \\
&= \|H_{\alpha_0}S\|^2 + \sum_{j=1}^{p} \big[\ -2\beta_j\alpha_0^t S_j + \beta_j^2\ \big] + \|(I - H_{\alpha_0})S\|^2 \\
&= \sum_{j=1}^{p} \big[\ (\alpha_0^t S_j)^2 - 2\beta_j\alpha_0^t S_j + \beta_j^2\ \big] - \sum_{j=1}^{p}(\alpha_0^t S_j)^2 + \|S\|^2 \\
&= \sum_{j=1}^{p}(\alpha_0^t S_j - \beta_j)^2\ + \phi(\alpha_0) \\
&= \|\alpha_0^t S - \beta^t\|^2 + \phi(\alpha_0) \\
&= \|S\alpha_0 - \beta\|^2 + \phi(\alpha_0)
\end{aligned}
$$

where in equation (A.2.1), S_j is the j-th column of S and β_j is the j-th element of vector β. Therefore, $\|S - \alpha_0\beta^t\|^2 = \|S\alpha_0 - \beta\|^2 + \phi(\alpha_0)$, where $\phi(\alpha_0) = -\sum_{j=1}^{p}(\alpha_0^t S_j)^2 + \|S\|^2 = \|S\|^2 - \|\alpha_0^t S\|^2$ is just a function of α_0.

Appendix B

Supplementary Material for Chapter 2

B.1 Proof of Theorem 2.1

Proof: Let K be a $\mathbb{R}^{m\times m}$ orthogonal such that $KA = (1, 0..., 0)^t$. Notice that if K is an orthogonal matrix, $\|KA\|_{\mathrm{F}}^2 = \|A\|_{\mathrm{F}}^2$. Then given $\mathbf{u} = \mathbf{u}_0$ and $d = d_0 > 0$, the objective function for solution $\mathbf{v}$ is

$$\begin{aligned}
\hat{\mathbf{v}} &= \arg\min_{\|\mathbf{v}\|^2=1} \|A - d_0\mathbf{u}_0\mathbf{v}^t\|_F^2 + \lambda_2\|\mathbf{v}\|_1 \\
&= \arg\min_{\|\mathbf{v}\|^2=1} \|KA - d_0K\mathbf{u}_0\mathbf{v}^t\|_F^2 + \lambda_2\|\mathbf{v}\|_1 \\
&= \arg\min_{\|\mathbf{v}\|^2=1} \|B - d_0\mathbf{1}_1\mathbf{v}^t\|_F^2 + \lambda_2\sum_{j=1}^{n}|v_j| \\
&= \arg\min_{\|\mathbf{v}\|^2=1} \sum_{j=1}^{n}(b_{1j} - d_0v_j)^2 + \sum_{j=1}^{n}\sum_{i=2}^{m} b_{ij}^2 + \lambda_2\sum_{j=1}^{n}|v_j| \\
&= \arg\min_{\|\mathbf{v}\|^2=1} -2d_0\sum_{j=1}^{n} b_{1j}v_j + d_0^2\sum_{j} v_j^2 + \lambda_2\sum_{j=1}^{n}|v_j| \\
&= \arg\min_{\|\mathbf{v}\|^2=1} -2d_0\sum_{j=1}^{n} b_{1j}v_j + \lambda_2\sum_{j=1}^{n}|v_j|.
\end{aligned}$$

For the last equation above, we know $-2d_0\sum_{j=1}^{n} b_{1j}v_j + \lambda_2\sum_{j=1}^{n}|v_j| \geq \sum_{j=1}^{n}(\lambda_2 - 2d_0|b_{1j}|)|v_j|$ and the "=" holds iff $b_{1j}v_j = |b_{1j}||v_j|$, i.e., $\mathrm{sign}(v_j) = \mathrm{sign}(b_{1j})$, for all $j = 1, 2, ..., n$. We denote this is the universal condition 1.

Let's then talk about two different cases of the lower bound $\sum_{j=1}^{n}(\lambda_2 - 2d_0|b_{1j}|)|v_j|$. If $\lambda_2 - 2d_0|b_{1j}| \geq$

0 for all $j = 1, 2, ..., n$, let $k = \arg\max_k |b_{1k}|$,

$$\begin{aligned}\sum_{j=1}^{n}(\lambda_2 - 2d_0|b_{1j}|)|v_j| &\geq \sum_{j=1}^{n}(\lambda_2 - 2d_0|b_{1k}|)|v_j| \\ &= (\lambda_2 - 2d_0|b_{1k}|)\sum_{j=1}^{n}|v_j| \\ &\geq \lambda_2 - 2d_0|b_{1k}|,\end{aligned}$$

under the $\sum_{j=1}^{n} v_j^2 = 1$ condition, the two "=" both hold if we choose $v_k = 1$ and $v_j = 0$ for all $j \neq k$. Here, if $k = \arg\max_k |b_{1k}|$ is not unique, we can pick any of one and set its coefficient to be 1.

If at least $\exists\ j$ such that $\lambda_2 - 2d_0|b_{1j}| < 0$, let $\mathcal{H} = \{h : \lambda_2 - 2d_0|b_{1h}| < 0\}$.

$$\begin{aligned}\sum_{j=1}^{n}(\lambda_2 - 2d_0|b_{1j}|)|v_j| &= -\sum_{j\in\mathcal{H}}\big|\lambda_2 - 2d_0|b_{1j}|\big||v_j| + \sum_{j\notin\mathcal{H}}(\lambda_2 - 2d_0|b_{1j}|)|v_j| \\ &\geq -\sum_{j\in\mathcal{H}}\big|\lambda_2 - 2d_0|b_{1j}|\big||v_j| \qquad \text{(B.1.1)} \\ &\geq -\sqrt{\sum_{j\in\mathcal{H}}(\lambda_2 - 2d_0|b_{1j}|)^2}\cdot\sqrt{\sum_{j\in\mathcal{H}}|v_j|^2} \qquad \text{(B.1.2)} \\ &\geq -\sqrt{\sum_{j\in\mathcal{H}}(\lambda_2 - 2d_0|b_{1j}|)^2}. \qquad \text{(B.1.3)}\end{aligned}$$

The "=" holds in (2) iff $v_j = 0$ for $j \notin \mathcal{H}$, (3) is by the Caucky-Swachz inequality and "=" holds iff $\frac{\left|\lambda_2 - 2d_0|b_{1j}|\right|}{|v_j|} = c$, with a constant c $\neq 0$ for $j \in \mathcal{H}$, and "=" holds in (4) iff $\sum_{j\in\mathcal{H}} v_j^2 = 1$. Combining the universal condition, for the second case, the solution should be the intersection of all the "=" hold, and that is

$$\begin{cases} v_j = 0, \text{ for } j \notin \mathcal{H} \\ \frac{\left|\lambda_2 - 2d_0|b_{1j}|\right|}{|v_j|} = c, \text{ with a constant c} \neq 0, \text{ for } j \in \mathcal{H} \\ \sum_{j\in\mathcal{H}} v_j^2 = 1 \\ \text{sign}(v_j) = \text{sign}(b_{1j}) \end{cases}$$

■

Appendix C

Supplementary Material for Chapter 3

C.1 Method dealing with $s < n$ case for $M_2(r)$ in Chapter 3

For fixed $r \in [0, 1]$, we want to solve

$$M_2(r) = \min_{\sum_{k=s+1}^{n} v_k^2 = 1-r^2} -2d \sum_{j=s+1}^{n} \sum_{i \in C_j} A_{ij} u_i v_j + \lambda \sum_{j=s+1}^{n} |v_j|.$$

Let $b_j = \sum_{i \in C_j} A_{ij} u_i$, we write $M_2(r)$ as

$$\begin{aligned}
M_2(r) &= \min_{\sum_{k=s+1}^{n} v_k^2 = 1-r^2} -2d \sum_{j=s+1}^{n} b_j v_j + \lambda \sum_{j=s+1}^{n} |v_j| \\
&= \max_{\sum_{k=s+1}^{n} v_k^2 = 1-r^2} 2d \sum_{j=s+1}^{n} b_j v_j - \lambda \sum_{j=s+1}^{n} |v_j| \\
&\leq \max_{\sum_{k=s+1}^{n} v_k^2 = 1-r^2} \left[2d \sum_{j=s+1}^{n} |b_j||v_j| - \lambda \sum_{j=s+1}^{n} |v_j| \right] \qquad \text{(C.1.1)} \\
&= \max_{\sum_{k=s+1}^{n} v_k^2 = 1-r^2} \left[\sum_{j=s+1}^{n} \Big(2d|b_j| - \lambda \Big) |v_j| \right] \qquad \text{(C.1.2)}
\end{aligned}$$

where "=" holds in equation (A.1) if $\text{sgn}(v_j) = \text{sgn}(b_j)$.

Similar as dealing with $M_1(r)$, we consider three different situations as λ varies in equation (A.2).

- Case I: $0 \leq \lambda \leq \min_{j=s+1:n} 2d|b_j|$.

By Cauchy - Schwartz inequality,

$$\sum_{j=s+1}^{n}\Big(2d|b_j|-\lambda\Big)|v_j| \le \sqrt{\sum_{j=s+1}^{n}\Big(2d|b_j|-\lambda\Big)^2}\cdot\sqrt{\sum_{j=s+1}^{n}|v_j|^2},$$

and "=" holds iff $\frac{2d|b_j|-\lambda}{|v_j|}=\alpha$ for $j=s+1,....,n$. Combining other restrictions, the final solution of $v_{s+1},...,v_n$ satisfies

$$\begin{cases}\operatorname{sgn}(v_j)=\operatorname{sgn}(b_j) & j=s+1,...,n\\ \sum\limits_{j=s+1}^{n}v_j^2=1-r^2 & \\ \frac{2d|b_j|-\lambda}{|v_j|}=\alpha_1\in\mathbb{R} & j=s+1,...,n\end{cases}$$

- Case II: $\min\limits_{j=s+1:n}2d|b_j|<\lambda\le\max\limits_{j=s+1:n}2d|b_j|$.

Suppose $\exists\ T<n$, $j=s+1,...,T$, we have $2d|b_j|>\lambda$, then

$$\sum_{j=s+1}^{n}\Big(2d|b_j|-\lambda\Big)|v_j| \le \sum_{j=s+1}^{T}\Big(2d|b_j|-\lambda\Big)|v_j|,$$

and "=" holds if $v_{T+1},...,v_n=0$. Similar as case I, we then continue maximize

$$\sum_{j=s+1}^{T}\Big(2d|b_j|-\lambda\Big)|v_j|$$

with restriction $\sum\limits_{j=s+1}^{T}v_j^2=1-r^2$, and the final solution satisfies

$$\begin{cases}v_j=0 & j=T+1,...,n\\ \operatorname{sgn}(v_j)=\operatorname{sgn}(b_j) & j=s+1,...,T\\ \sum\limits_{j=s+1}^{T}v_j^2=1-r^2 & \\ \frac{2d|b_j|-\lambda}{|v_j|}=\alpha_2\in\mathbb{R} & j=s+1,...,T\end{cases}$$

- Case III: $\lambda>\max\limits_{j=s+1:n}2d|b_j|$.

Let $M=\max\limits_{j=s+1:n}2d|b_j|$, suppose $\exists\ T<n$, for $j=s+1,...,T$ we have $2d|b_j|=M$, then for

$j = T+1, ..., n$ we have $2d|b_j| < M < \lambda$.

$$\begin{aligned} \sum_{j=s+1}^{n} \Big(2d|b_j| - \lambda\Big)|v_j| &\leq (M-\lambda) \sum_{j=s+1}^{n} |v_j| \\ &\leq (M-\lambda)\sqrt{1-r^2}. \end{aligned}$$

There are $T - s$ different solutions making the above two "=" hold. Let k be any number in set {s+1, s+2,..., T},

$$v_j = \begin{cases} \sqrt{1-r^2} & j = k \\ 0 & j = 1, 2, ..., n \text{ and } j \neq k \end{cases}$$

Appendix D

Supplementary Material for Chapter 4

D.1 Derivation of parameter updating of sPCA-VB algorithm

The objective function is

$$\Omega(q_1, q_2, ..., q_p, \sigma^2, \omega, \mathbf{v}) = \mathbb{E}^{q_1,...,q_p} \log \frac{Q(\mathbf{u}, \mathbf{Z})}{P(\mathbf{u}, \mathbf{Z}, \mathbf{v}, \sigma^2, \omega | A)}, \tag{D.1.1}$$

and we optimize it by iteratively solving Q and $\eta = (\mathbf{v}, \sigma^2, \omega)$.

- Update $q(\mathbf{u}, \mathbf{Z})$

For each $i = 1, 2, ..., p$, given current solution of hyper-parameters $\widehat{\eta} = (\widehat{\mathbf{v}}, \widehat{\sigma}^2, \widehat{\omega})$ and the distributions of other q's, we solve $q_i(u_i, Z_i)$ by minimizing equation $(D.1.1)$. Due to the factorization form of $Q(\mathbf{u}, \mathbf{V}) = \prod_{i=1}^{p} q_i(u_i, Z_i)$, it is equivalent to minimize

$$\begin{aligned}
& \mathbb{E}^{Q} \log \frac{q_i(u_i, Z_i)}{p(A|\mathcal{H})\pi(\mathcal{H})} \\
= \; & \mathbb{E}^{q_i}_{(u_i,Z_i)} \mathbb{E}^{Q}_{\mathcal{H}[-u_i,-Z_i]} \log \frac{q_i(u_i, Z_i)}{p(A|\mathcal{H})\pi(\mathcal{H})} \\
= \; & \mathbb{E}^{q_i}_{(u_i,Z_i)} \mathbb{E}^{Q}_{\mathcal{H}[-u_i,-Z_i]} \log \frac{\left[\alpha_i f_i(u_i)\right]^{Z_i} \left[(1-\alpha_i)\delta(u_i)\right]^{1-Z_i}}{\left[\widehat{\omega} \frac{1}{\sqrt{2\pi a_1 \widehat{\sigma}^2}} \exp(-\frac{u_i^2}{2a_1\widehat{\sigma}^2})\right]^{Z_i} \left[(1-\widehat{\omega})\delta(u_i)\right]^{(1-Z_i)}} \\
& -\mathbb{E}^{q_i}_{(u_i,Z_i)} \mathbb{E}^{Q}_{\mathcal{H}[-u_i,-Z_i]} \log \prod_{l,j} \frac{1}{\sqrt{2\pi\widehat{\sigma}^2}} \exp\left(-\frac{(A_{l,j} - u_l \widehat{v}_j)^2}{2\widehat{\sigma}^2} \right) \\
= \; & \alpha_i \mathbb{E}_{u_i|Z_i=1} \log \frac{\alpha_i f_i(u_i)}{\widehat{\omega} \frac{1}{\sqrt{2\pi a_1 \widehat{\sigma}^2}} \exp(-\frac{u_i^2}{2a_1\widehat{\sigma}^2})} + (1-\alpha_i) \mathbb{E}_{u_i|Z_i=0} \log \frac{1-\alpha_i}{1-\widehat{\omega}} \\
& +\mathbb{E}^{q_i}_{(u_i,Z_i)} \mathbb{E}^{Q}_{\mathcal{H}[-u_i,-Z_i]} \sum_{l,j} \frac{(A_{l,j} - u_l \widehat{v}_j)^2}{2\widehat{\sigma}^2} + \text{cont.}
\end{aligned}$$

$$
\begin{aligned}
&= \alpha_i \mathbb{E}_{u_i|Z_i=1} \log \frac{\alpha_i f_i(u_i)}{\widehat{\omega} \frac{1}{\sqrt{2\pi a_1 \widehat{\sigma}^2}} \exp(-\frac{u_i^2}{2a_1\widehat{\sigma}^2})} + (1-\alpha_i)\mathbb{E}_{u_i|Z_i=0} \log \frac{1-\alpha_i}{1-\widehat{\omega}} \\
&\quad + \alpha_i \mathbb{E}^{q_i}_{u_i|Z_i=1} \log \exp\Big(\sum_{j=1:p} \frac{-2A_{i,j}u_i\widehat{v}_j + u_i^2\widehat{v}_j^2}{2\widehat{\sigma}^2} \Big) + \text{cont.} \qquad \text{(D.1.2)} \\
&= \mathbb{E}^{f_i}_{u_i|Z_i=1} \alpha_i \log \frac{f_i(u_i) \exp\Big(\sum_{j=1:p} \frac{-2A_{i,j}u_i\widehat{v}_j + u_i^2\widehat{v}_j^2}{2\widehat{\sigma}^2} \Big)}{\exp(-\frac{u_i^2}{2a_1\widehat{\sigma}^2})} + \text{cont.} \qquad \text{(D.1.3)}
\end{aligned}
$$

Minimizing (D.1.3) leads to $f_i(u_i) \propto \exp\Big(-\frac{u_i^2}{2a_1\widehat{\sigma}^2} - \sum_{j=1:p} \frac{-2A_{i,j}u_i\widehat{v}_j + u_i^2\widehat{v}_j^2}{2\widehat{\sigma}^2} \Big)$, and this is a normal distribution with mean and variance as

$$
\mu_{u_i} = \frac{\sum_{j=1:n} A_{ij}\widehat{v}_j}{\frac{1}{a_1} + \sum_{j=1:n} \widehat{v}_j^2} = \frac{\sum_{j=1:n} A_{ij}\widehat{v}_j}{\frac{1}{a_1} + 1} \qquad \text{(D.1.4)}
$$

$$
\sigma^2_{u_i} = \frac{\widehat{\sigma}^2}{\frac{1}{a_1} + \sum_{j=1:n} \widehat{v}_j^2} = \frac{\widehat{\sigma}^2}{\frac{1}{a_1} + 1} \qquad \text{(D.1.5)}
$$

Therefore, if we plug in the distribution $f_i(u_i)$ into (D.1.2), we have

$$
\begin{aligned}
&\mathbb{E}^Q \log \frac{q(u_i, Z_i)}{p(A|\mathcal{H})\pi(\mathcal{H})} \\
= \;& \alpha_i \mathbb{E}_{u_i|Z_i=1} \log \frac{\alpha_i \frac{1}{\sqrt{2\pi\widehat{\sigma}^2_{u_i}}} \exp\Big(-\frac{(u_i-\mu_{u_i})^2}{2\widehat{\sigma}^2_{u_i}} \Big)}{\widehat{\omega} \frac{1}{\sqrt{2\pi a_1\widehat{\sigma}^2}} \exp(-\frac{u_i^2}{2a_1\widehat{\sigma}^2})} + (1-\alpha_i) \log \frac{1-\alpha_i}{1-\widehat{\omega}} \\
& + \alpha_i \mathbb{E}^{q_i}_{u_i|Z_i=1} \Big(\sum_{j=1:p} \frac{-2A_{i,j}u_i\widehat{v}_j + u_i^2\widehat{v}_j^2}{2\widehat{\sigma}^2} \Big) + \text{cont.} \\
= \;& \alpha_i \log \frac{\alpha_i}{\widehat{\omega}} + \alpha_i \mathbb{E}_{u_i|Z_i=1} \log \Big[\sqrt{\frac{a_1\widehat{\sigma}^2}{\sigma^2_{u_i}}} \exp\Big(-\frac{(u_i-\mu_{u_i})^2}{2\widehat{\sigma}^2_{u_i}} + \frac{u_i^2}{2a_1\widehat{\sigma}^2} \Big) \Big] + (1-\alpha_i) \log \frac{1-\alpha_i}{1-\widehat{\omega}} \\
& + \alpha_i \mathbb{E}^{q_i}_{u_i|Z_i=1} \Big(\sum_{j=1:p} \frac{-2A_{i,j}u_i\widehat{v}_j + u_i^2\widehat{v}_j^2}{2\widehat{\sigma}^2} \Big) + \text{cont.} \qquad \text{(D.1.6)}
\end{aligned}
$$

To continue minimizing (D.1.6), we can take derivative with respect to α_i, and let it equal to 0, we

have

$$\Big(\log\frac{\alpha_i}{\widehat{\omega}}+1\Big)+\Big(\log\sqrt{\frac{a_1\widehat{\sigma}^2}{\sigma_{u_i}^2}}-\frac{\sigma_{u_i}^2}{2\sigma_{u_i}^2}+\frac{\mu_{u_i}^2+\sigma_{u_i}^2}{2a_1\widehat{\sigma}^2}\Big)-\Big(\log\frac{1-\alpha_i}{1-\widehat{\omega}}+1\Big)+\sum_{j=1:p}\frac{-2A_{i,j}\widehat{v}_j\mu_{u_i}+(\mu_{u_i}^2+\sigma_{u_i}^2)\widehat{v}_j^2}{2\widehat{\sigma}^2}=0,$$

where we use the fact that $\mathbb{E}^{q_i}_{u_i|Z_i=1}(u_i-\mu_{u_i})^2=\sigma_{u_i}^2$ and $\mathbb{E}^{q_i}_{u_i|Z_i=1}u_i^2=\mu_{u_i}^2+\sigma_{u_i}^2$. If we plug the form of μ_{u_i} and $\sigma_{u_i}^2$ in (D.1.4) and (D.1.5) into the last term above, we have

$$\log\frac{\alpha_i}{\widehat{\omega}}+\log\sqrt{\frac{a_1\widehat{\sigma}^2}{\sigma_{u_i}^2}}-\frac{1}{2}+\frac{\mu_{u_i}^2+\sigma_{u_i}^2}{2a_1\widehat{\sigma}^2}-\log\frac{1-\alpha_i}{1-\widehat{\omega}}+\frac{-2\widehat{\sigma}^2\frac{\mu_{u_i}^2}{\sigma_{u_i}^2}+(\mu_{u_i}^2+\sigma_{u_i}^2)(\frac{\widehat{\sigma}^2}{\sigma_{u_i}^2}-\frac{1}{a_1})}{2\widehat{\sigma}^2}=0,$$

and it turns out to be the logit updating formula for α_i, i.e.,

$$\log\frac{\alpha_i}{1-\alpha_i}=\log\frac{\widehat{\omega}}{1-\widehat{\omega}}+\frac{\mu_{u_i}^2}{2\sigma_{u_i}^2}-\frac{1}{2}\log\frac{a_1\widehat{\sigma}^2}{\sigma_{u_i}^2}.$$

- Update $\mathbf{v}$

Given all the distributions of $q_i(u_i,Z_i)$, $i=1,2,...,p$, part of the hyper-parameters $\widehat{\sigma}^2$ and $\widehat{\omega}$, we can solve $\widehat{\mathbf{v}}$ through

$$\widehat{\mathbf{v}}=\arg\min_{\mathbf{v}}\mathbb{E}^Q\log\frac{q(\mathbf{u},\mathbf{Z})}{p(A|\mathcal{H})\pi(\mathcal{H})}=\arg\max_{\mathbf{v}}\mathbb{E}^Q\log\frac{p(A|\mathcal{H})\pi(\mathcal{H})}{q(\mathbf{u},\mathbf{Z})}=\arg\max_{\mathbf{v}}\mathbb{E}^Q\log\Big[p(A|\mathcal{H})\pi(\mathbf{v})\Big].$$

With the L_2 norm one restriction $\sum_{k=1:p}v_k^2=1$, we write

$$\begin{aligned}
\mathbb{E}^Q\log\Big[p(A|\mathcal{H})\pi(\mathbf{v})\Big] &= \mathbb{E}^Q\log\prod_{i,j}\frac{1}{\sqrt{2\pi\widehat{\sigma}^2}}e^{-\frac{(A_{i,j}-u_iv_j)^2}{2\widehat{\sigma}^2}}\\
&= -\mathbb{E}^Q\sum_{i,j}\frac{(A_{i,j}-u_iv_j)^2}{2\widehat{\sigma}^2}+\text{cont.}\\
&= -\sum_{i,j}\frac{A_{i,j}^2+\alpha_i(\mu_{u_i}^2+\sigma_{u_i}^2)v_j^2-2A_{i,j}\alpha_i\mu_{u_i}v_j}{2\widehat{\sigma}^2}+\text{cont.}\\
&= -\sum_{i,j}\frac{(A_{i,j}-\mathbb{E}^{q_i}u_iv_j)^2}{2\widehat{\sigma}^2}-\sum_{i,j}\frac{\mathbb{E}^{q_i}u_i^2v_j^2}{2\widehat{\sigma}^2}+\sum_{i,j}\frac{(\mathbb{E}^{q_i}u_i)^2v_j^2}{2\widehat{\sigma}^2}+\text{cont.}\\
&= -\frac{\|A-\mathbb{E}^Q\mathbf{u}\mathbf{v}^t\|_{\mathrm{F}}^2}{2\widehat{\sigma}^2}-\frac{\sum_{i=1:p}\Big(\mathbb{E}^{q_i}u_i^2-(\mathbb{E}^{q_i}u_i)^2\Big)}{2\widehat{\sigma}^2}+\text{cont.}\\
&= -\frac{\|A^t\mathbb{E}^Q\mathbf{u}-\mathbf{v}\|^2}{2\widehat{\sigma}^2}+g(\mathbb{E}^Q\mathbf{u}),
\end{aligned}$$

where $g(\mathbb{E}^Q\mathbf{u})$ is a function of $\mathbb{E}^Q\mathbf{u}$ not involving $\mathbf{v}$. Therefore,

$$\widehat{\mathbf{v}} = \arg\max_{\|\mathbf{v}\|^2=1} \Big(-\frac{\|A^t\mathbb{E}^Q\mathbf{u}-\mathbf{v}\|^2}{2\widehat{\sigma}^2} + g(\mathbb{E}^Q\mathbf{u})\Big) = \arg\min_{\|\mathbf{v}\|^2=1} \frac{\|A^t\mathbb{E}^Q\mathbf{u}-\mathbf{v}\|^2}{2\widehat{\sigma}^2}$$

satisfies

$$\begin{cases} \frac{(A^t\mathbb{E}^Q\mathbf{u})_k}{v_k} = c, \text{ with a constant c} \neq 0 \\ \sum\limits_{k=1:p} v_k^2 = 1 \end{cases}$$

where $(A^t\mathbb{E}^Q\mathbf{u})_k = \sum\limits_{i=1:p} A_{i,k}\alpha_i\mu_{u_i}$ is the k-th element of $A^t\mathbb{E}^Q\mathbf{u}$.

- Update ω

Given all the distributions of $q_i(u_i, Z_i)$, $i = 1, 2, ..., p$, part of the hyper-parameters $\widehat{\sigma}^2$ and $\widehat{\mathbf{v}}$, we can solve $\widehat{\omega}$ through

$$\widehat{\omega} = \arg\min_{\omega} \mathbb{E}^Q \log\frac{q(\mathbf{u},\mathbf{Z})}{p(A|\mathcal{H})\pi(\mathcal{H})} = \arg\max_{\omega} \mathbb{E}^Q \log\frac{p(A|\mathcal{H})\pi(\mathcal{H})}{q(\mathbf{u},\mathbf{Z})} = \arg\max_{\omega} \mathbb{E}^Q \log\Big[p(A|\mathcal{H})\pi(\omega)\pi(\mathbf{u},\mathbf{Z})\Big].$$

If we continue simplifying the above equation and taking first derivative with respect to ω, we have

$$\begin{aligned}
& \frac{\partial}{\partial\omega}\mathbb{E}^Q \log\Big[p(A|\mathcal{H})\pi(\omega)\pi(\mathbf{u},\mathbf{Z})\Big] \\
= \; & \frac{\partial}{\partial\omega}\mathbb{E}^Q \log\Big[\prod_{i,j}\frac{1}{\sqrt{2\pi\widehat{\sigma}^2}}\exp(-\frac{(A_{i,j}-u_i\widehat{v}_j)^2}{2\widehat{\sigma}^2})\prod_{k=1:p}\Big(\omega\frac{1}{\sqrt{2\pi a_1\widehat{\sigma}^2}}\exp(-\frac{u_k^2}{2a_1\widehat{\sigma}^2})\Big)^{Z_k}\Big((1-\omega)\delta(u_k)\Big)^{1-Z_k}\Big] \\
= \; & \frac{\partial}{\partial\omega}\Big[\mathbb{E}^Q\sum_{k=1:p}\log\Big(\omega^{Z_k}(1-\omega)^{1-Z_k}\Big) + \text{cont}\Big] \\
= \; & \frac{\partial}{\partial\omega}\sum_{k=1:p}\Big(\alpha_k\mathbb{E}_{u_k|Z_k=1}\log\omega + (1-\omega_k)\mathbb{E}_{u_k|Z_k=0}\log(1-\omega)\Big) \\
= \; & \frac{\partial}{\partial\omega}\sum_{k=1:p}\Big(\alpha_k\log\omega + (1-\omega_k)\log(1-\omega)\Big) \\
= \; & \frac{1}{\omega}\sum_{k=1:p}\alpha_k - \frac{1}{1-\omega}\sum_{k=1:p}(1-\alpha_k) \qquad \text{(D.1.7)}
\end{aligned}$$

Set the last equation (D.1.7) equal to 0, we have the solution $\widehat{\omega} = \sum\limits_{k=1}^{p}\alpha_k/p$.

- Update σ^2

Given all the distributions of $q_i(u_i, Z_i)$, $i = 1, 2, ..., p$, part of the hyper-parameters $\widehat{\omega}$ and $\widehat{\mathbf{v}}$, we can solve $\widehat{\sigma}^2$ through

$$\widehat{\sigma}^2 = \arg\min_{\sigma^2} \mathbb{E}^Q \log \frac{q(\mathbf{u}, \mathbf{Z})}{p(A|\mathcal{H})\pi(\mathcal{H})} = \arg\max_{\sigma^2} \mathbb{E}^Q \log \frac{p(A|\mathcal{H})\pi(\mathcal{H})}{q(\mathbf{u}, \mathbf{Z})} = \arg\max_{\sigma^2} \mathbb{E}^Q \log \Big[p(A|\mathcal{H})\pi(\sigma^2)\pi(\mathbf{u}, \mathbf{Z})\Big].$$

Further calculating the this equation, we have

$$\begin{aligned}
& \mathbb{E}^Q \log \Big[p(A|\mathcal{H})\pi(\sigma^2)\pi(\mathbf{u}, \mathbf{Z})\Big] \\
= \; & \mathbb{E}^Q \log \prod_{i,j} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(A_{i,j} - u_i\widehat{v}_j)^2}{2\sigma^2}} + \mathbb{E}^Q \log \frac{1}{\Gamma(1)} (\sigma^2)^{-2} e^{-\frac{1}{\sigma^2}} \\
& + \mathbb{E}^Q \log \prod_{k=1:p} \Big(\widehat{\omega} \frac{1}{\sqrt{2\pi a_1 \sigma^2}} e^{-\frac{u_k^2}{2a_1\sigma^2}}\Big)^{Z_k} \Big((1 - \widehat{\omega})\delta(u_k)\Big)^{1 - Z_k} + \text{cont.} \\
= \; & -p^2 \log \sqrt{2\pi\sigma^2} - \mathbb{E}^Q \sum_{i,j} \frac{(A_{i,j} - u_i\widehat{v}_j)^2}{2\sigma^2} - 2\log(\sigma^2) - \frac{1}{\sigma^2} \\
& + \sum_{k=1:p} \alpha_k \mathbb{E}_{u_k|Z_k=1} \log \Big(\frac{1}{\sqrt{\sigma^2}} e^{-\frac{u_k^2}{2a_1\sigma^2}}\Big) + \sum_{k=1:p} (1 - \alpha_k) \mathbb{E}_{u_k|Z_k=0} \log \Big((1 - \widehat{\omega})\delta(u_k)\Big) + \text{cont.} \\
= \; & -p^2 \log \sqrt{2\pi\sigma^2} - \mathbb{E}^Q \sum_{i,j} \frac{(A_{i,j} - u_i\widehat{v}_j)^2}{2\sigma^2} - 2\log(\sigma^2) - \frac{1}{\sigma^2} \\
& + \sum_{k=1:p} \alpha_k \Big(-\frac{1}{2}\log\sigma^2 - \frac{\mu_{u_k}^2 + \sigma_{u_k}^2}{2a_1\sigma^2}\Big) + \text{cont.} \qquad \text{(D.1.8)}
\end{aligned}$$

Take derivative with respect to σ^2 in above equation (D.1.8) and set it equal to 0, we solve $\widehat{\sigma}^2$ as

$$-\frac{p^2}{2\sigma^2} + \frac{\mathbb{E}^Q \sum_{i,j}(A_{i,j} - u_i\widehat{v}_j)^2}{2(\sigma^2)^2} - \frac{2}{\sigma^2} + \frac{1}{(\sigma^2)^2} - \frac{\sum_{k=1:p} \alpha_k}{2\sigma^2} + \frac{\sum_{k=1:p} \alpha_k(\mu_{u_k}^2 + \sigma_{u_k}^2)}{2a_1(\sigma^2)^2} = 0$$

$$\Leftrightarrow \frac{p^2 + \sum_{k=1:p} \alpha_k + 4}{2\sigma^2} = \frac{\mathbb{E}^Q \sum_{i,j}(A_{i,j} - u_i\widehat{v}_j)^2 + \sum_{k=1:p} \alpha_k(\mu_{u_k}^2 + \sigma_{u_k}^2)/a_1 + 2}{2\sigma^4}$$

$$\Leftrightarrow \widehat{\sigma}^2 = \frac{\mathbb{E}^Q \sum_{i,j}(A_{ij} - u_i\widehat{v}_j)^2 + \frac{1}{a_1} \sum_{i=1:p} \alpha_i(\mu_{u_i}^2 + \sigma_{u_i}^2) + 2}{p^2 + \sum_{i=1:p} \alpha_i + 4}$$

where $\mathbb{E}^Q \sum_{i,j}(A_{ij} - u_i\widehat{v}_j)^2$ can be further calculated as

$$\begin{aligned}\mathbb{E}^Q \sum_{i,j}(A_{ij} - u_i\widehat{v}_j)^2 &= \mathbb{E}^Q \sum_{i,j} A_{ij}^2 - 2\mathbb{E}^Q \sum_{i,j} A_{ij}u_i\widehat{v}_j + \mathbb{E}^Q \sum_{i,j} u_i^2\widehat{v}_j^2 \\ &= \sum_{i,j} A_{ij}^2 - 2\sum_{i,j} \alpha_i\widehat{v}_j A_{ij}\mu_{u_i} + \sum_{i,j} \alpha_i\widehat{v}_j^2(\mu_{u_i}^2 + \sigma_{u_i}^2)\end{aligned}$$

D.2 Proof of Theorem 4.1

Let $\widehat{\lambda}$ be the largest eigenvalue of sample covariance matrix A, and $\|\rho\|^2+\xi^2$ is the largest eigenvalue of true covariance matrix A_0, by Weyl's inequality, λ satisfies

$$|\widehat{\lambda} - (\|\rho\|^2 + \xi^2)| \leq \|E\|_2 \leq \xi\|\rho\|\sqrt{\frac{p}{n}} + \xi^2(\frac{p}{n} + 2\sqrt{\frac{p}{n}}).$$

Since p is fixed, and $\|\rho\|^2$ and ξ are finite, we have $\widehat{\lambda} = O_p(1)$.

Now, let's take a look at the updating rule for parameter α_i. If we set initial value of $\mathbf{v}^{(0)} = \widehat{\mathbf{v}}$, $\omega^{(0)} \in (0,1)$ and $(\sigma^{(0)})^2 \sim \frac{1}{n}$, one step updating for α_i gives us

$$\log\frac{\alpha_i}{1-\alpha_i} = \log\frac{\omega^{(0)}}{1-\omega^{(0)}} + \frac{\mu_{u_i}^2}{2\sigma_{u_i}^2} - \frac{1}{2}\log\frac{a_1(\sigma^{(0)})^2}{\sigma_{u_i}^2} = \log\frac{\omega^{(0)}}{1-\omega^{(0)}} + \frac{(A\mathbf{v}^{(0)})_i^2}{2(\frac{1}{a_1}+1)(\sigma^{(0)})^2} - \frac{1}{2}\log\Big(1+a_1\Big)$$

The key term in the above equation is the order of $A\mathbf{v}^{(0)} = A\widehat{\mathbf{v}} = \widehat{\lambda}\widehat{\mathbf{v}}$. Using proposition 2, $\|\widehat{\mathbf{v}} - \rho^*\| = O(\frac{1}{\sqrt{n}})$, thus there exists a constant C not dependent to n, for any $i = 1, 2, ..., p$

$$|\rho_i^*| - C\frac{1}{\sqrt{n}} < |\widehat{v}_i| < |\rho_i^*| + C\frac{1}{\sqrt{n}}.$$

Therefore, given $\min_{i\in I}(\rho_i^*) \neq 0$ which is fixed,

$$\begin{aligned}\min_{i\in I}\log\frac{\alpha_i}{1-\alpha_i} &= \log\frac{\omega^{(0)}}{1-\omega^{(0)}} + \frac{(\min_{i\in I}\widehat{\lambda}\widehat{v}_i)^2}{2(\frac{1}{a_1}+1)(\sigma^{(0)})^2} - \frac{1}{2}\log\Big(1+a_1\Big) \\ &\sim \log\frac{\omega^{(0)}}{1-\omega^{(0)}} + \frac{\widehat{\lambda}^2\Big(\min_{i\in I}|\rho_i^*| - C\frac{1}{\sqrt{n}}\Big)^2}{2(\frac{1}{a_1}+1)\frac{1}{n}} - \frac{1}{2}\log\Big(1+a_1\Big) \\ &\sim \frac{n\min_{i\in I}(\rho_i^*)^2}{\frac{1}{a_1}+1} - \log(a_1) \\ &\sim n\end{aligned}$$

To make $\min_{i\in I}\log\frac{\alpha_i}{1-\alpha_i} < c(n)$, one sufficient condition is that $c(n)\to\infty$ and $c(n)=o(n)$, and $c(n)=\theta\log(a_1)$ with $\theta\in(0,1/2)$ also satisfies this condition.

On the other side,

$$\begin{aligned}
\max_{i\in I^c}\log\frac{\alpha_i}{1-\alpha_i} &= \log\frac{\omega^{(0)}}{1-\omega^{(0)}}+\frac{(\max_{i\in I^c}\widehat{\lambda}\widehat{v}_i)^2}{2(\frac{1}{a_1}+1)(\sigma^{(0)})^2}-\frac{1}{2}\log(1+a_1)\\
&< \log\frac{\omega^{(0)}}{1-\omega^{(0)}}+\frac{\widehat{\lambda}^2(|\rho_k^*|+C\frac{1}{\sqrt{n}})^2}{2(\frac{1}{a_1}+1)(\sigma^{(0)})^2}-\frac{1}{2}\log(1+a_1)\\
&\sim \frac{\frac{1}{n}}{(\frac{1}{a_1}+1)\frac{1}{n}}-\frac{1}{2}\log(a_1)\\
&\sim -\frac{1}{2}\log(a_1)
\end{aligned}$$

where we assume $\max_{i\in I^c}\widehat{v}_i = v_k$, and $\rho_k^*=0$ due to $\rho_i^*=0$ for any $i\in I^c$. Thus, it is guaranteed that if $c(n)=\theta\log(a_1)$ with $\theta\in(0,1/2)$, we could also have $\max_{i\in I^c}\log\frac{\alpha_i}{1-\alpha_i} < -c(n)$.

Combining the above two statements, we show that under given initial values of ω, $\sigma^2(n)$ and $\mathbf{v}$, as well as conditions on $c(n)$, a_1 and $\log(a_1)$,

$$\mathbb{P}\Big(\min_{i\in I}\log\frac{\alpha_i}{1-\alpha_i} > c(n) \text{ and } \max_{i\in I^c}\log\frac{\alpha_i}{1-\alpha_i} < -c(n)\Big)\to 1,$$

which is equivalent to $\mathbb{P}(\widehat{I}=I)\to 1$.

References

Abdullah, A. and Hussain, A. (2006). A new biclustering technique based on crossing minimization. *Neurocomputing*, 69(16):1882–1896.

Akaike, H. (1973). Maximum likelihood identification of gaussian autoregressive moving average models. *Biometrika*, pages 255–265.

Allen, G. I., Grosenick, L., and Taylor, J. (2014). A generalized least-square matrix decomposition. *Journal of the American Statistical Association*, 109(505):145–159.

Alter, O., Brown, P. O., and Botstein, D. (2001). Processing and modeling genome-wide expression data using singular value decomposition. In *BiOS 2001 The International Symposium on Biomedical Optics*, pages 171–186. International Society for Optics and Photonics.

Amini, A. A. and Wainwright, M. J. (2008). High-dimensional analysis of semidefinite relaxations for sparse principal components. In *2008 IEEE International Symposium on Information Theory*, pages 2454–2458. IEEE.

Ben-Dor, A., Shamir, R., and Yakhini, Z. (1999). Clustering gene expression patterns. *Journal of computational biology*, 6(3-4):281–297.

Berthet, Q. and Rigollet, P. (2013). Complexity theoretic lower bounds for sparse principal component detection. In *COLT*, pages 1046–1066.

Bishop, C. M. (2006). Pattern recognition. *Machine Learning*, 128.

Blei, D. M., Jordan, M. I., et al. (2006). Variational inference for dirichlet process mixtures. *Bayesian analysis*, 1(1):121–144.

Bottolo, L., Richardson, S., et al. (2010). Evolutionary stochastic search for bayesian model exploration. *Bayesian Analysis*, 5(3):583–618.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Braun, M. and McAuliffe, J. (2010). Variational inference for large-scale models of discrete choice. *Journal of the American Statistical Association*, 105(489):324–335.

Cadima, J. and Jolliffe, I. T. (1995). Loading and correlations in the interpretation of principle compenents. *Journal of Applied Statistics*, 22(2):203–214.

Cai, T. T., Ma, Z., Wu, Y., et al. (2013). Sparse pca: Optimal rates and adaptive estimation. *The Annals of Statistics*, 41(6):3074–3110.

Candès, E. J., Li, X., Ma, Y., and Wright, J. (2011). Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11.

Chandrasekaran, V., Sanghavi, S., Parrilo, P. A., and Willsky, A. S. (2009). Sparse and low-rank matrix decompositions. In *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*, pages 962–967. IEEE.

Chandrasekaran, V., Sanghavi, S., Parrilo, P. A., and Willsky, A. S. (2011). Rank-sparsity incoherence for matrix decomposition. *SIAM Journal on Optimization*, 21(2):572–596.

Cheng, Y. and Church, G. M. (2000). Biclustering of expression data. In *Ismb*, volume 8, pages 93–103.

Clemmensen, L., Hastie, T., Witten, D., and Ersboll, B. (2011). Sparse discriminant analysis. *Technometrics*, 53(4):406–413.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (1990). *Introduction to Algorithms*. MIT Press and McGraw-Hill, 1 edition.

Cui, N. (2012). *Contributions to modeling parasite dynamics and dimension reduction*. PhD thesis, University of Illinois at Urbana-Champaign.

d'Aspremont, A., Bach, F., and Ghaoui, L. E. (2008). Optimal solutions for sparse principal component analysis. *Journal of Machine Learning Research*, 9(Jul):1269–1294.

d'Aspremont, A., El Ghaoui, L., Jordan, M. I., and Lanckriet, G. R. (2007). A direct formulation for sparse pca using semidefinite programming. *SIAM review*, 49(3):434–448.

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96(456):1348–1360.

Fan, J., Peng, H., et al. (2004). Nonconcave penalized likelihood with a diverging number of parameters. *The Annals of Statistics*, 32(3):928–961.

Fisher, R. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7:179–188.

Flandin, G. and Penny, W. D. (2007). Bayesian fmri data analysis with sparse spatial basis function priors. *NeuroImage*, 34(3):1108–1125.

Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741.

Gemulla, R., Nijkamp, E., Haas, P. J., and Sismanis, Y. (2011). Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 69–77. ACM.

Guan, N., Tao, D., Luo, Z., and Yuan, B. (2012). Nenmf: an optimal gradient method for nonnegative matrix factorization. *IEEE Transactions on Signal Processing*, 60(6):2882–2898.

Hartigan, J. A. (1972). Direct clustering of a data matrix. *Journal of the american statistical association*, 67(337):123–129.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109.

He, X. and Niyogi, P. (2003). Locality preserving projections. *Advances in Neural Information Processing Systems*, 16:153–160.

Horn, R. A. and Johnson, C. R. (1985). *Matrix Analysis*. Cambridge University Press, Cambridge, 2 edition.

Hoyle, D. C. and Rattray, M. (2004). Principal-component-analysis eigenvalue spectra from data with symmetry-breaking structure. *Physical Review E*, 69(2):026124.

Huang, J. Z., Shen, H., and Buja, A. (2012). The analysis of two-way functional data using two-way regularized singular value decompositions. *Journal of the American Statistical Association*.

Johnstone, I. M. and Lu, A. Y. (2012). On consistency and sparsity for principal components analysis in high dimensions. *Journal of the American Statistical Association*.

Jolliffe, I. T. (1995). Rotation of principal components: choice of normalization constraints. *Journal of Applied Statistics*, 22(1):29–35.

Jolliffe, I. T. (2002). *Principal Component Analysis*. Springer-Verlag, New York, 2 edition.

Jolliffe, I. T., Trendafilov, N. T., and Uddin, M. (2003). A modified principal component technique based on the lasso. *Journal of computational and Graphical Statistics*, 12(3):531–547.

Jolliffe, I. T. and Uddin, M. (2000). The simplified component technique: an alternative to rotated principal components. *Journal of Computational and Graphical Statistics*, 9(4):689–710.

Jordan, M. I. (2004). Graphical models. *Statistical Science*, pages 140–155.

Kim, H. and Park, H. (2008). Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. *SIAM journal on matrix analysis and applications*, 30(2):713–730.

Kluger, Y., Basri, R., Chang, J. T., and Gerstein, M. (2003). Spectral biclustering of microarray data: coclustering genes and conditions. *Genome research*, 13(4):703–716.

Kucukelbir, A., Tran, D., Ranganath, R., Gelman, A., and Blei, D. M. (2016). Automatic differentiation variational inference. *arXiv preprint arXiv:1603.00788*.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Lazzeroni, L. and Owen, A. (2002). Plaid models for gene expression data. *Statistica sinica*, pages 61–86.

Lee, D. D. and Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791.

Lee, D. D. and Seung, H. S. (2001). Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562.

Lee, M., Shen, H., Huang, J. Z., and Marron, J. S. (2010). Biclustering via sparse singular value decomposition. *Biometrics*, 66(4):1087–1095.

Li, F. and Zhang, N. R. (2010). Bayesian variable selection in structured high-dimensional covariate spaces with applications in genomics. *Journal of the American statistical association*, 105(491):1202–1214.

Li, K.-C. (1991). Sliced inverse regression for dimension reduction. *Journal of American Statistical Association*, 86:316–327.

Liang, F., Paulo, R., Molina, G., Clyde, M. A., and Berger, J. O. (2008). Mixtures of g priors for bayesian variable selection. *Journal of the American Statistical Association*, 103(481):410–423.

Lin, C.-J. (2007a). On the convergence of multiplicative update algorithms for nonnegative matrix factorization. *IEEE Transactions on Neural Networks*, 18(6):1589–1596.

Lin, C.-J. (2007b). Projected gradient methods for nonnegative matrix factorization. *Neural computation*, 19(10):2756–2779.

Liu, J., Yang, J., and Wang, W. (2004). Biclustering in gene expression data by tendency. In *Computational Systems Bioinformatics Conference, 2004. CSB 2004. Proceedings. 2004 IEEE*, pages 182–193. IEEE.

Loog, M. and Haeb-Umbach, R. (2001). Multiclass linear dimension reduction by weighted pairwise fisher criteria. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(7):762–766.

Lounici, K. (2013). Sparse principal component analysis with missing observations. In *High dimensional probability VI*, pages 327–356. Springer.

Lu, A. Y. (2002). *Sparse principal component analysis for functional data*. PhD thesis, Stanford University.

Ma, Z. et al. (2013). Sparse principal component analysis and iterative thresholding. *The Annals of Statistics*, 41(2):772–801.

Mallows, C. L. (1973). Some comments on c p. *Technometrics*, 15(4):661–675.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092.

Nadler, B. (2008). Finite sample approximation results for principal component analysis: A matrix perturbation approach. *The Annals of Statistics*, pages 2791–2817.

Onatski, A. (2012). Asymptotics of the principal components estimator of large factor models with weakly influential factors. *Journal of Econometrics*, 168(2):244–258.

Ormerod, J. T. and Wand, M. P. (2010). Explaining variational approximations. *The American Statistician*, 64(2):140–153.

Paatero, P. and Tapper, U. (1994). Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126.

Parkhomenko, E., Tritchler, D., and Beyene, J. (2009). Sparse canonical correlation analysis with application to genomic data integration. *Statistical Applications in Genetics and Molecular Biology*, 8(1):1–34.

Paul, D. (2007). Asymptotics of sample eigenstructure for a large dimensional spiked covariance model. *Statistica Sinica*, pages 1617–1642.

Paul, D. and Johnstone, I. M. (2012). Augmented sparse principal component analysis for high dimensional data. *arXiv preprint arXiv:1202.1242*.

Prasantha, H., Shashidhara, H., and Murthy, K. B. (2007). Image compression using svd. In *Conference on Computational Intelligence and Multimedia Applications, 2007. International Conference on*, volume 3, pages 143–145. IEEE.

Reimann, P., Van den Broeck, C., and Bex, G. J. (1996). A gaussian scenario for unsupervised learning. *Journal of Physics A: Mathematical and General*, 29(13):3521.

Robert, C. P. (2004). *Monte carlo methods*. Wiley Online Library.

Schwarz, G. et al. (1978). Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464.

Shabalin, A. A. and Nobel, A. B. (2013). Reconstruction of a low-rank matrix in the presence of gaussian noise. *Journal of Multivariate Analysis*, 118:67–76.

Shen, D., Shen, H., and Marron, J. S. (2013). Consistency of sparse pca in high dimension, low sample size contexts. *Journal of Multivariate Analysis*, 115:317–333.

Shen, H. and Huang, J. Z. (2008). Sparse principal component analysis via regularized low rank matrix approximation. *Journal of multivariate analysis*, 99(6):1015–1034.

Sill, M., Kaiser, S., Benner, A., and Kopp-Schneider, A. (2011). Robust biclustering by sparse singular value decomposition incorporating stability selection. *Bioinformatics*, 27(15):2089–2097.

Stingo, F. C. and Vannucci, M. (2011). Variable selection for discriminant analysis with markov random field priors for the analysis of microarray data. *Bioinformatics*, 27(4):495–501.

Sugiyama, M. (2006). Local fisher discriminant analysis for supervised dimensionality reduction. 23rd International Conference on Machine Learning, Pittsburgh, PA.

Sugiyama, M. (2007). Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis. *The Journal of Machine Learning Research*, 8:1027–1061.

Tanay, A., Sharan, R., and Shamir, R. (2002). Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18(suppl 1):S136–S144.

Teschendorff, A. E., Wang, Y., Barbosa-Morais, N. L., Brenton, J. D., and Caldas, C. (2005). A variational bayesian mixture modelling framework for cluster analysis of gene-expression data. *Bioinformatics*, 21(13):3025–3033.

Thomasian, A., Castelli, V., and Li, C.-S. (1998). Clustering and singular value decomposition for approximate indexing in high dimensional spaces. In *Proceedings of the seventh international conference on Information and knowledge management*, pages 201–207. ACM.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of Royal Statistical Society, Series B*, 58:267–288.

Trendafilov, N. T. and Jolliffe, I. T. (2006). Projected gradient approach to the numerical solution of the scotlass. *Computational Statistics & Data Analysis*, 50(1):242–253.

Vines, S. (2000). Simple principal components. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 49(4):441–451.

Vu, V. Q., Cho, J., Lei, J., and Rohe, K. (2013). Fantope projection and selection: A near-optimal convex relaxation of sparse pca. In *Advances in neural information processing systems*, pages 2670–2678.

Vu, V. Q. and Lei, J. (2012). Minimax rates of estimation for sparse pca in high dimensions. In *AISTATS*, volume 15, pages 1278–1286.

Witten, D. M., Tibshirani, R., and Hastie, T. (2009). A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, 10-3:515–534.

Yuan, X.-T. and Zhang, T. (2013). Truncated power method for sparse eigenvalue problems. *Journal of Machine Learning Research*, 14(Apr):899–925.

Zou, H., Hastie, T., and Tibshirani, R. (2006). Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15-2:265–286.