High-dimensional Covariance Matrix Estimation with an Application to Classification

Yingwei Zhai

January 16, 2017

1 Introduction

Covariance matrix estimation is one of the most popular subject in multivariate analysis, since it can be applied widely in social science, economics, biological study, engineering, and many other fields. The sample covariance matrix, which is the standard and most natural estimator, however, does not perform well and may result in invalid conclusions in high-dimensional settings. Thus many approaches for covariance matrix estimation have been developed in the past few years, especially in the field of the high-dimensional condition, where the dimension p is much larger than the sample size n. Here we only focus on the high-dimensional covariance matrix estimation and its application in classification.

Because of the popularity in this area, there have been enormous efforts to develop the methodologies to estimate the covariance matrix. For example, Banerjee, Ghaoui and d'Aspremont(2008), Bickel and Levina (2008), Fan, Fan and Lv(2008), Cai, Zhang and Zhou(2010), etc. In particular, Bickel and Levina (2008) used the banding estimator to regularize the covariance matrix. Using the banding matrix B_k, we can derive the banding estimator for Σ which is $\bar{\Sigma} \circ B_k$, where $\circ$ represents the Schur product. By choosing k, the banding estimator $\bar{\Sigma} \circ B_k$ attains a convergence rate which is easy to calculate.

Another useful way to smooth the covariance matrix is Tapering. Cai, Zhang and Zhou (2010). With a tapering matrix defined as T_k, the tapering estimator for Σ become $\bar{\Sigma} \circ T_k$. Similar to the banding process, by choosing a particular k, the tapering estimator $\bar{\Sigma} \circ T_k$ will achieve a rate of convergence.

Furthermore, Cai and Yuan(2012) proposed that using adaptive estimator through block thresholding is another good way to obtain our goal. We first construct blocks of a certain size, and then threshold the blocks by keeping the diagonal blocks and delete the large blocks. Again, we choose k by a specific rule and get the optimal rate of convergence, which is adaptive over the class of covariance matrices.

One of many important applications of the covariance matrix is classification. Hoffbeck and Landgrebe(1996), Shao et al.(2011), Dudoit et al.(2002), and Ledoit and Wolf (2013) are all very good examples in this area. In our paper we mainly focus on Shao et al.(2011). They used the thresholding approach to obtain a plausible estimator of the covariance matrix, and then applied the result to established a new way to classify a subject, the Sparse Linear Discriminant Analysis(SLDA), which improved the performance of traditional LDA. The thresholding method here is a development in one important technique of function estimation, wavelet shrinkage [Donoho and Johnstone (1994) and Donoho et al. (1995)], and the covariance matrix estimation from Bickel and Levina (2008). Shao et

al. also used a data-guided method via a leave-one-out cross-validation process to apply the SLDA and estimate the misclassification rate.

The rest of our paper is organized as follows. In Section 2, we first introduce the notation and terminology while setting up the model with assumption of bandable covariance matrices, which will reduce the difficulty from the high-dimensionality. Next, three distinct estimators of the covariance matrix are presented, which are banding, tapering and adaptive estimators. In Section 3, the well-known LDA is introduced with two classes, together with the conditional misclassification rate and classification rule. Then we implement the five estimation processes established in Section 2 to attain the covariance matrix, which are the original LDA, LDA with a thresholded estimator of the different between the mean of two classesδ, SLDA with banding estimator, SLDA with tapering estimator, and SLDA with adaptive block-thresholding estimator. Among each type we will explain how they are embed in to the LDA or Sparse LDA, basically following the proposal of Shao et al.(2011). In section 4, we conduct the numerical experiment. The dataset used is the Leukemia data from Golub et al., 1999. We apply the classification technique to distinguish the acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL). We compare the five methods by analyzing the corresponding misclassification rate. Last but not least, a proof of how the conditional misclassification rate works with tapering estimator can be found in the Appendix.

2 Model Setup and Three Types of Covariance Estimates

We assume that $X^{(1)},\ldots,X^{(n)}$ are n independent samples from the p dimensional Gaussian distribution $X=(X_1,\ldots,X_p)^T\sim N(\mu,\Sigma)$ with $p>>n$. As is known to all, the standard natural estimator, the sample covariance matrix,

$$\bar{\Sigma}=\frac{1}{n-1}\sum_{i=1}^{n}(X^{(i)}-\bar{X})(X^{(i)}-\bar{X})^T \tag{1}$$

is not a consistent estimator of Σ and performs poorly. To overcome this difficulty, we consider one of the most popular structured covariance matrics – the bandable covariance matrices, especially the class of covariance matrices proposed by Bickel and Levina (2008):

$$\mathcal{C}_\alpha=\mathcal{C}_\alpha(M_0,M):\quad=\quad\{\Sigma:\max_j\sum_i\{|\sigma_{ij}|:|i-j|\geq k\}\leq Mk^{-\alpha},\forall k,0<M_0^{-1}\leq\lambda_{min}(\Sigma),\lambda_{max}(\Sigma)\leq M_0\} \tag{2}$$

For any $\Sigma\in\mathcal{C}_\alpha$, σ_{ij} is close to 0 as $|i-j|$ is large, which indicates the "bandable" property of Σ.

Based on the sample covariance matrix, with $\log(p)=o(n)$, we will consider three following different estimators to estimate bandable $\Sigma\in\mathcal{C}_\alpha$:

The first is the banding estimator (Bickel and Levina, 2008). With a banding matrix defined as $B_k=I(|i-j|<k)_{1\leq i,j\leq p}$, the banding estimator for Σ is $\bar{\Sigma}\circ B_k$, and $\circ$ represents the Schur product. By choosing $k\asymp(\log p/n)^{1/(2(\alpha+1))}$, Bickel and Levina (2008) showed that the resulting banding estimator $\bar{\Sigma}\circ B_k$ attains the rate of convergence

$$||\bar{\Sigma}\circ B_k-\Sigma||=O_p((\frac{\log p}{n})^{\alpha/(2\alpha+2)})$$

uniformly over $\mathcal{C}_\alpha$, where $||\cdot||$ is the spectral norm.

The second is the tapering estimator (Cai, Zhang and Zhou, 2010). With a tapering matrix defined as $T_k = (\frac{2}{k}\{(k-|i-j|)_+ - (k/2-|i-j|)_+\})_{1\le i,j\le p}$, the tapering estimator for Σ is $\bar{\Sigma}\circ T_k$. By choosing $k \asymp n^{1/(2\alpha+1)}$, Cai, Zhang and Zhou (2010) showed that the tapering estimator $\bar{\Sigma}\circ T_k$ achieves the rate of convergence

$$||\bar{\Sigma}\circ T_k - \Sigma|| = O_p(n^{-\alpha/(2\alpha+1)} + (\frac{\log p}{n})^{1/2}) \tag{3}$$

uniformly over $\mathcal{C}_\alpha$. Let $\tilde{\Sigma} = \bar{\Sigma}\circ T_k$, we have the following property of the inverse of $\tilde{\Sigma}$:

$$||\tilde{\Sigma}^{-1} - \Sigma^{-1}|| = O_p(n^{-\alpha/(2\alpha+1)} + (\frac{\log p}{n})^{1/2}) \tag{4}$$

The third is the adaptive estimator through block thresholding (Cai and Yuan, 2012). Construction of this estimator consists of two steps, one is construction of blocks and the other is blocking thresholding. Here we introduce the construction of blocks briefly. We only consider the upper half of the $p\times p$ matrix because of the symmetric. First, blocks of size k_0 along the diagonal is constructed. Second, either one or two new blocks were created for each row successively towards the top right corner, which makes odd rows have 3 blocks of size k_0 and even rows have 2 blocks. Then we double the size of blocks to $2k_0$ and repeat the first two steps. The size of blocks will continue to be doubled until the upper half is covered. Note the indices of blocks are used to represent blocks. Once the blocks $\mathcal{B} = \{B_1,\ldots,B_N\}$ are constructed, for block $B = I\times J \in \mathcal{B}$, the following thresholding method is applied: keep the diagonal blocks, that is, $\hat{\Sigma}_B = \bar{\Sigma}_B$ if $I = J$; "kill" the large blocks, that is, $\hat{\Sigma}_B = 0$ if $d(B) > n/\log n$; threshold the intermediate blocks, that is,

$$\hat{\Sigma}_B = T_{\lambda_0}(\bar{\Sigma}_B) = \bar{\Sigma}\cdot I\left(||\bar{\Sigma}_B|| > \lambda_0\sqrt{||\bar{\Sigma}_{I\times I}||||\bar{\Sigma}_{J\times J}||}\sqrt{\frac{d(B)+\log p}{n}}\right) \tag{5}$$

where $d(B) = \max\{|I|,|J|\}$, and $\lambda_0 = 6$. By choosing $k_0 \asymp \log p$, Cai and Yuan (2012) shows that the block thresholding estimator obtains the optimal rate of covergence

$$||\hat{\Sigma} - \Sigma|| = O_p(n^{-\alpha/(2\alpha+1)} + (\frac{\log p}{n})^{1/2}) \tag{6}$$

adaptively over $\mathcal{C}_\alpha$, for $\forall\alpha > 0$.

3 Linear Discriminant Analysis (LDA)

In this paper we only consider the classification problem with two classes. Let $\mathbf{x}_k \sim N_p(\mu_k,\Sigma)$, $k = 1,2$, $\mu_1 \neq \mu_2$, and Σ is positive definite. The misclassification rule is the average of the probabilities of making two types of misclassification: classifying $\mathbf{x}$ to class 1 when $\mathbf{x} \sim N_p(\mu_2,\Sigma)$ and classifying $\mathbf{x}$ to class 2 when $\mathbf{x} \sim N_p(\mu_1,\Sigma)$.

Suppose μ_1, μ_2 and Σ are known, then we have the optimal classification rule, which means the rule with least rate of misclassfication, classifies $\mathbf{x}$ to class 1 if and only if

$$\delta'\Sigma^{-1}(\mathbf{x} - \bar{\mu}) \geqslant 0 \tag{7}$$

where $\bar{\mu} = \frac{\mu_1+\mu_2}{2}$ and $\delta = \mu_1 - \mu_2$. We assume the following regularity conditions:

$$M_0^{-1} \leq \lambda_{min}(\Sigma), \lambda_{max}(\Sigma) \leq M_0, \text{ and } M_0^{-1} \leq \max_{j \leq p} \delta_j^2 \leq M_0 \tag{8}$$

where δ_j is the jth component of δ. Denote the misclassification rate of this optimal rule as R_{OPT}. Using the normal distribtion, we can show that

$$R_{OPT} = \Phi(-\triangle_p/2), \triangle_p = \sqrt{\delta' \Sigma^{-1} \delta} \tag{9}$$

In practice, μ_1, μ_2 and Σ are unknown. We consider $n = n_1 + n_2 \to \infty$ when we apply the limit process, and assume that n_1/n converges to a constant $c \in (0,1)$. For a classfication rule T constructed using the training sample $\mathbf{X}$, we define the average of the conditional probabilities of making two types of misclassification with respect to x as $R_T(\mathbf{X})$, to evaluate the performance of the classification.

Here we hope to find a rule T such that $R_T(\mathbf{X})$ converges in probability to the same limit as R_{OPT}. Particularly, when $R_{OPT} \to_P 0$, we hope both $R_T(\mathbf{X}) \to_P 0$ and $R_T(\mathbf{X})$ and R_{OPT} obtain the same rate of convergence.
Therefore we apply the following definition in Shao et al. (2011) to assess the performance.

Definition 3.1. *Let T be a classification rule with conditional misclassification rate $R_T(\boldsymbol{X})$ given the training sample $\boldsymbol{X}$*

(i) T is asymptotically optimal if $R_T(\boldsymbol{X})/R_{OPT} \to_P 1$.
(ii) T is asymptotically sub-optimal if $R_T(\boldsymbol{X}) - R_{OPT} \to_P 0$.
(iii) T is asymptotically worst if $R_T(\boldsymbol{X}) \to_P 1/2$.

3.1 LDA

The original LDA uses the maximum likelihood estimators $\bar{\mathbf{x}}_1$, $\bar{\mathbf{x}}_2$ and $\bar{\Sigma}$, where

$$\bar{\mathbf{x}}_k = \frac{1}{n_k} \sum_{i=1}^{n_k} \mathbf{x}_{ki}, \quad k = 1,2 \quad \bar{\Sigma} = \frac{1}{n} \sum_{k=1}^{2} \sum_{i=1}^{n_k} (\mathbf{x}_{ki} - \bar{\mathbf{x}}_k)(\mathbf{x}_{ki} - \bar{\mathbf{x}}_k)'$$

The LDA applies the classification rule as follows:

$$\text{classifying } \mathbf{x} \text{ to class 1 if and only if } \hat{\delta}' \hat{\Sigma}^{-1} (\mathbf{x} - \hat{\bar{\mu}}) \geqslant 0$$

where $\hat{\delta}$, $\hat{\bar{\mu}}$, and $\hat{\Sigma}^{-1}$ are estimators of δ, $\bar{\mu}$ and Σ^{-1} respectively, constructed using the training sample $\mathbf{X}$. More specifically, $\hat{\delta} = \bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2$, $\hat{\bar{\mu}} = \bar{\mathbf{x}} = (\bar{\mathbf{x}}_1 + \bar{\mathbf{x}}_2)/2$ and $\hat{\Sigma}^{-1} = \bar{\Sigma}^{-1}$ when $\bar{\Sigma}^{-1}$ exists, $\hat{\Sigma}^{-1} =$ a generalized inverse $\bar{\Sigma}^{-}$ when $\bar{\Sigma}^{-1}$ does not exist. In Bickel and Levina (2004, Theorem 1), it was shown that if $p > n$ and $p/n \to \infty$, then the unconditional misclassfication rate $E[R_T(\mathbf{X})]$ converges to $1/2$, which implies that the LDA is asymptotically worst, and asymptotically sub-optimal is the best we can wish for. Hence we have plenteous room to boost the performance of LDA by changing the estimations of the parameters in this model.

3.2 LDA with Thresholding Estimator

To improve the performance of the initial LDA, we propose to apply the thresholding method to the estimator of δ (Bickel and Levina, 2008). There are two reasons for this proposal to be made:

1. when Δ_P is large, $N_p(\mu_1, \Sigma)$ and $N_p(\mu_2, \Sigma)$ have a large difference, which will result in a small misclassification rate (Δ_p is defined in (9)).

2. since δ needs to be estimated by the training sample X, which has a much smaller size than p, it will be difficult to construct a satisfactory classification rule with a large divergence rate of Δ_P.

Now we consider

$$D_{g,p} = \sum_{j=1}^{p} \delta_j^{2g}.$$

This is a measure of sparsity of δ, where δ_j is the jth component of δ, g is a constant with $g \in [0,1)$. δ is sparse when $D_{g,p} \ll p$. Shao et al. (2011) showed that $\Delta_p^2 \leqslant c_0||\delta||^2 \leqslant c_0^{1+2(1-g)} D_{g,p}$ under the conditions

$$c_0^{-1} \leqslant all\ eigenvalues\ of\ \Sigma \leqslant c_0, \tag{10}$$

and

$$c_0^{-1} \leqslant max_{j \leqslant p} \delta_j^2 \leqslant c_0. \tag{11}$$

Hence we have the procedure as let $\hat{\delta}$ to be thresholded at

$$a_n \asymp (\frac{\log p}{n})^{\alpha}$$

with $\alpha \in (0, 1/2)$, that is, the jth component of $\tilde{\delta}$ is $\hat{\delta}_j I(|\hat{\delta}_j| > a_n)$, where $\hat{\delta}_j$ is the jth component of $\hat{\delta}$, denoted by $\tilde{\delta}$. The result is derived by diminishing the boundedness of Δ_p and place the sparsity on δ. Shao et al. argues that since the δ is the difference of two normal distribution, imposing the sparsity condition on δ is weaker and more reasonable than imposing sparsity conditions on both $\mu_1$and μ_2.

Other than the sparse estimator $\tilde{\delta}$, the remaining parameters remains the same as Section 3.1, that is, $\hat{\bar{\mu}} = \bar{\mathbf{x}} = (\bar{\mathbf{x}}_1 + \bar{\mathbf{x}}_2)/2$ and $\hat{\Sigma}^{-1} = \bar{\Sigma}^{-1}$ when $\bar{\Sigma}^{-1}$ exists, $\hat{\Sigma}^{-1} =$ a generalized inverse $\bar{\Sigma}^{-}$ when $\bar{\Sigma}^{-1}$ does not exist.

3.3 Sparse LDA with Banding Estimator

Shao et al.(2011) introduced a sparse LDA based on the thresholding estimators of the mean effects, together with the banding covariance estimation proposed in Bickel and Levina (2008). The sparse LDA applies the classification rule as follows:

$$\text{classifying } \mathbf{x} \text{ to class 1 if and only if } \tilde{\delta}' \tilde{\Sigma}^{-1} (\mathbf{x} - \hat{\bar{\mu}}) \geqslant 0$$

where $\tilde{\delta}$, $\hat{\bar{\mu}}$, and $\tilde{\Sigma}^{-1}$ are estimators of δ, $\bar{\mu}$ and Σ^{-1} respectively, constructed using the training sample $\mathbf{X}$. Specifically, $\hat{\bar{\mu}} = \bar{\mathbf{x}} = (\bar{\mathbf{x}}_1 + \bar{\mathbf{x}}_2)/2$. Same as Section 3.2, $\tilde{\delta}$ is that $\hat{\delta}$ is thresholded at $a_n \asymp (\frac{\log p}{n})^{\alpha}$ with $\alpha \in (0, 1/2)$. $\tilde{\Sigma}^{-1}$ is the estimator of the inverse of Σ by using the banding covariance estimation proposed in Bickel and Levina (2008).

In addition, Shao et al. established the asymptotic optimality of SLDA: Define

$$C_{h,p} = max_{j \leqslant p} \sum_{l=1}^{p} |\sigma_{jl}|^h, \quad D_{g,p} = \sum_{j=1}^{p} \sigma_j^{2g}, \quad a_n = M_2 (\frac{logp}{n})^{\alpha},$$

$$q_n = the\ number\ of\ j's\ with\ |\delta|_j > a_n/r,\ and\ d_n = C_{h,p} (n^{-1} logp)^{(1-h)/2}.$$

Assume the conditions (10), (11) hold together with

$$\frac{logp}{n} \to 0,$$

and

$$b_n = max\Big\{ d_n, \frac{a_n^{1-g}\sqrt{D_{g,p}}}{\Delta_p}, \frac{\sqrt{C_{h,p}q_n}}{\Delta_p\sqrt{n}} \Big\} \to 0.$$

Then we have

1. The conditional misclassification rate of the SLDA as

$$R_{SLDA}(\mathbf{X}) = \Phi(-[1+O_p(b_n)]\Delta_p/2).$$

2. If Δ_p is bounded, then the SLDA is asymptotically optimal and

$$\frac{R_{SLDA}(\mathbf{X})}{R_{OPT}} - 1 = O_p(b_n)$$

.

3. If $\Delta_p \to \infty$, then the SLDA is asymptotically sub-optimal.
4. If $\Delta_p \to \infty$ and $b_n\Delta_p^2 \to 0$, then the SLDA is asymptotically optimal .

In this section we directly adopt the method that Shao et al. established, and the numerical improvement in misclassification rate will be presented in Section 4, by changing from LDA to SLDA .

3.4 Sparse LDA with Tapering Estimator

From the previous section, we showed the approach in Shao et al. (2011) in applying the banding estimator to LDA. In this section, we would like use tapering, another typical method based on smoothing the sample covariance matrices, to find the covariance matrix Σ and the corresponding Σ^{-1}. Cai, Ren and Zhou (2010) proposed to construct the tapering matrix as follows:

1. Define $M_1 = M \times n^{-\frac{\alpha}{2\alpha+1}}$, where M is a data-driven constant.
2. Define

$$T = \begin{cases} 1, \ if|\hat{\sigma}_{i,j}| > M_1 \\ 2 - \frac{2\hat{\sigma}_{i,j}}{M_1} \ if \frac{M_1}{2} \leqslant |\hat{\sigma}_{i,j}| \leqslant M_1 \\ 0, \ otherwise \end{cases}$$

3. The tapering estimator for Σ becomes $\tilde{\Sigma} = \bar{\Sigma} \circ B_k$, where $\circ$ represents the Schur product. By choosing the number $k \asymp n^{1/(2\alpha+1)}$, the tapering estimator achieves a convergence rate. Since we only have the range of α, the best value of α, which can lead to the smallest misclassification rate, is data-dependent.

After attaining the tapering estimator, we do the same manipulation as plugging the result into the conditional misclassification rate in SLDA , and show how it performs. Specifically, $\hat{\bar{\mu}} = \bar{\mathbf{x}} = (\bar{\mathbf{x}}_1 + \bar{\mathbf{x}}_2)/2$. $\tilde{\delta}$ is that $\hat{\delta}$ is thresholded at $a_n \asymp (\frac{\log p}{n})^{\alpha}$ with $\alpha \in (0, 1/2)$. $\tilde{\Sigma}^{-1}$ is the estimator of the inverse of Σ, which is estimated by the tapering estimator.

Notice that since this step is a development of the proposal in Shao et al.(2011), we proved that in SLDA, the properties of the conditional misclassification rate still hold when we change a banding estimator into a tapering estimator. The brief proof can be found in the Appendix.

3.5 Sparse LDA with Adaptive Block Thresholding Estimator

In this section, we proposed another estimator to use in SLDA, the adaptive block-thresholding estimator. Here we use the method proposed by Cai and Yuan(2012). As introduced in Section 2, we

first decompose the sample covariance matrix into blocks, and then apply the thresholding technique to each block, with respect to their dimensions and sizes.

Recall that we assume our covariance matrices are bandable in order to reduce the difficulty caused by the high-dimensionality. Therefore we try to make our covariance matrices as bandable as possible, which will be further explained in Section 4.

Our next step is to plug the adaptive block-thresholding estimator into the conditional misclassification rate in SLDA , and analyze it performance. In this section, we only change the estimator of Σ and Σ^{-1}, and keep the other parameters used in Shao et al.(2011), where $\hat{\mu} = \bar{\mathbf{x}} = (\bar{\mathbf{x}}_1 + \bar{\mathbf{x}}_2)/2$. $\tilde{\delta}$ denotes that $\hat{\delta}$ is thresholded at $a_n \asymp (\frac{\log p}{n})^{\alpha}$ with $\alpha \in (0, 1/2)$. $\tilde{\Sigma}^{-1}$ is the estimator of the inverse of Σ, which is estimated by the adaptive block-thresholding estimator. Note that blockwise thresholding estimation is very different from tapering estimation, as the former one does not depend on α.

This method is another development of the proposal in Shao et al.(2011), and the properties of the conditional misclassification rate preserves the same when we change a banding estimator into an adaptive block-thresholding estimator. The proof is very similar with the one in the case of tapering estimator, hence it is omitted.

3.6 Estimator of Σ^{-1}

For each of the above LDA method, we note that the estimator of Σ^{-1} is needed. However, the inverse of the estimator of Σ cannot be directly used as the estimator of Σ^{-1}. Although the estimator of Σ is positive definite with a high probability, it cannot be guaranteed that the estimator is positive definite with probability 1. To find the appropriate estimator of Σ^{-1}, we need to apply some special technique introduced in Cai and Zhou (2011). First of all, decompose $\hat{\Sigma}$ as

$$\hat{\Sigma} = \sum_{i=1}^{p} \hat{\lambda}_i v_i v_i^T,$$

where $\hat{\lambda}_i$'s and v_i's are the eigenvalues and eigenvectors of $\hat{\Sigma}$ respectively. The next step is to define $\hat{\lambda}_i{}^{+} = max(0, \hat{\lambda}_i)$ to be the positive part of $\hat{\lambda}_i$, and hence we have

$$\hat{\Sigma}^{+} = \sum_{i=1}^{p} \hat{\lambda}_i{}^{+} v_i v_i^T.$$

Cai and Zhou (2011) showed that $\hat{\Sigma}^{+}$ is a positive semi-definite estimator, and still achieves the optimal rate of convergence, just as $\hat{\Sigma}$. Moreover, we can set

$$\hat{\lambda}_i{}^{+} = max(\varepsilon_n, \hat{\lambda}_i)$$

where ε_n is some positive value if we prefer a positive definite estimator. Similarly, the corresponding $\hat{\Sigma}^{+}$ is now strictly positive definite and obtains the optimal convergence rate.

Now we use block-thresholding to construct an adaptive estimator for $\Omega := \Sigma^{-1}$, which is designed in Cai and Yuan(2012). Here we use eigen-decomposition $\hat{\Sigma}$ as

$$\hat{\Sigma} = \hat{U}\hat{D}\hat{U}^T,$$

where $\hat{U}$ is an orthogonal matrix and $\hat{D}$ is a diagonal matrix. Then we define

$$\hat{\Omega} = \hat{U} diag(min\{\hat{d}_{ii}^{-1}, n\})\hat{U}^T,$$

where $\hat{d}_{ii}^{-1}$ denotes the ith diagonal element of $\hat{D}$. Thus the optimal rate of convergence for Ω is adaptively achieved by $\hat{\Omega}$. Cai and Yua, however, argue that the truncation of $\hat{d}_{ii}^{-1}$ is needed to deal with the case where $\hat{\Sigma}$ is near singular.

4 Numerical Results

The dataset we used in this section is the Golub et al. (1999) data. This data basically is to classify between acute lymphoblastic leukemia (ALL) and acute myeloid leukemia (AML). There are 47 patients with ALL and 25 patients with AML. With Affymetrix Hgu6800 chips of these 72 patients, the dataset contains 7129 gene expressions (Affymetrix probes). Golub et al. (1999) pointed out that aLL arises from two different types of lymphocytes (T-cell and B-cell), and the data could be classified into three classes: AML, ALL-T and ALL-B. But in our paper, we only consider two-class classfication instead of three-class case. The chemotherapy regimens for ALL can be really harmful for AML patients, so distinguishing ALL from AML is what we are interested in.
In fact, this data has been analysed by many authors using different methods. Shao et al. (2011) considered the original data and applied the sparse LDA (SLDA) induced in their paper. By comparing with the result of LDA, they found that their SLDA method reduced the misclassification rate by 70% in the leave-one-out cross validation case. Liao and Qin (2007) also used the same data to distinguish AML from ALL through Logistic Regression method. Different from Shao et al. (2011), they applied the pre-processing procedures in Dudoit et al. (2002) to filter out genes that is not variable enough. In their data, p=3051 genes remained after pre-processing.
In this numerical section, we use the original data that the gene expression levels of p=7129 and n=72 patients. We use the R package *golubEsets* that directly gives the training samples and test samples. The training set has 27 patients with ALL and 11 patients with AML, while the test set has 20 patients with ALL and 14 patients with AML.

4.1 LDA

We first considered the LDA, that is, the original version of the LDA. Notice that, when we tried to find the inverse of sample covariance matrix, we used function eigs_sys() from the R package *rARPACK* and threshold the above-found eigenvalues with a parameter tol. This procedure was also applied to the other methods in our paper. Set $tol = 1$. 1 is enough, because we found that all eigenvalues that is larger than 1 actually are in the level of 10^7 or larger, and those that are smaller than 1 are in the level of 10^{-6} or smaller. The misclassification rate of the test data predicted by the classification rule derived from the training data is 14.70588%.

4.2 LDA with delta-thresholding

Then we tried the LDA with the delta-thresholding method. This edition of LDA only changes the estimator of δ to be a thresholded one. We fixed $\alpha = 0.3$ in the threshold of the sparse estimator $\hat{\delta}$. In this case, the parameter M_2 which helps change the threshold of $\hat{\delta}$ is considered to change, but interestingly, for $M2 = 50, 100, \ldots, 450, 500$, the misclassification rate keeps the same to be 14.70588%. Although the thresholding of the estimator of $delta$ might improve the performance of the LDA to some extent, obviously this is not the case here. Hence we considered the fixed $M2 = 300$ for the following three LDA methods with different covariance estimation.

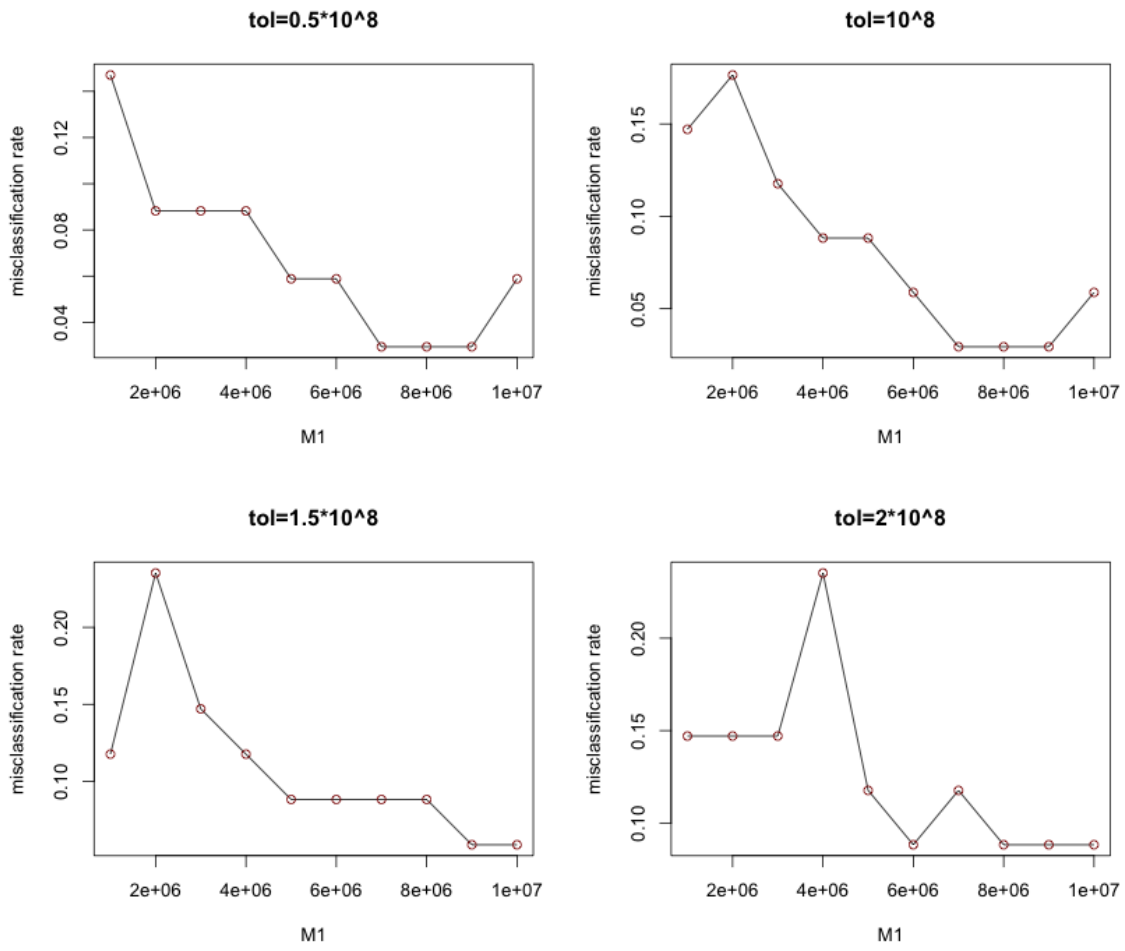

Figure 1: Misclassification rate with banding wrt M1 and tol

4.3 LDA with banding estimation

As mentioned before, $M2$ is fixed. We carried the LDA with banding covariance estimator. There are two parameters considered, $M1$ and tol. For $tol = 0.5 \times 10^8, 1 \times 10^8, 1.5 \times 10^8, 2 \times 10^8$, each panel from Figure 1 shows the performance of classification with respect to $M1 = 1 \times 10^6, 2 \times 10^6, \ldots, 10^7$. For $tol = 0.5 \times 10^8$, the LDA with banding estimation is best performed with the misclassification rate is 2.941176% when $M1 = 7 \times 10^6, 8 \times 10^6$ and 9×10^6. Also, for $tol = 1 \times 10^8$, the LDA with banding estimation is best performed with the misclassification rate is 2.941176% when $M1 = 7 \times 10^6, 8 \times 10^6$ and 9×10^6. Comparing the two subfigures in the first row from Figure 1, we found that the values of misclassification rates are the same starting from $M1 = 4 \times 10^6$. This is because we set the largest number of eigenvalues that can be used is 150. In both cases, when $M1 \geq 4 \times 10^6$, all 150 eigenvalues are used, which leads to the same misclassification rates. For $tol = 1.5 \times 10^8$, this method has the lowest misclassification rate 5.882353% when $M1 = 9 \times 10^6$ and 10^7. For $tol = 2 \times 10^8$, the lowest misclassification rate 8.823529% when $M1 = 6 \times 10^6, 8 \times 10^6, 9 \times 10^6$ and 10^7. We should see that when tol increases from 10^8, the best performances of the LDA becomes worse. In all, the smallest misclassification rate among all the above cases is 2.941176% when $tol = 10^8$ and $M1 = 8 \times 10^6$ and 9×10^6. We plot the image of banding matrix B when $M1 = 9 \times 10^6$ in Figure 2. The black color indicates that the value in B is 1, while the white one shows that the value in B is 0.

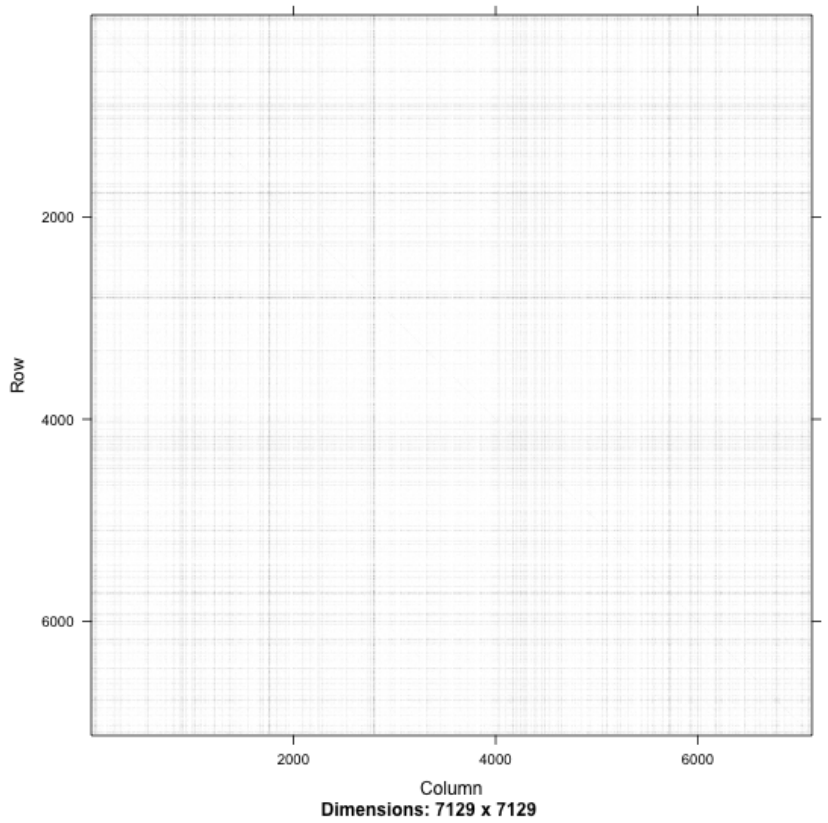

Figure 2: Banding Matrix B with $tol = 10^8$ and $M1 = 9 \times 10^6$

4.4 LDA with tapering estimation

Still, in this case, set $M2 = 300, \alpha = 0.3$ in the threshold a_n for the sparse estimator $\hat{\delta}$. We considered the LDA with tapering covariance estimator. There are two parameters, $tol = 0.5 \times 10^8, 1 \times 10^8, \ldots, 3 \times 10^8$ and $M1 = 1 \times 10^6, 2 \times 10^6, \ldots, 10^7$. Figure 3 shows how miscalssification rates changes with respect to $M1$ and tol. For $tol = 0.5 \times 10^8$, the smallest misclassification rate 8.823529% is first achieved at $M1 = 3 \times 10^6$, and then the rate keeps the same. Also, for $tol = 10^8$, we could have the smallest misclassification rate of 8.823529% at $M1 = 3 \times 10^6$, and then the rate keeps the same afterwards. For these two cases, the misclassification rate stays the same when M1 has been above some specific value. This is because that when tol is small, there would be more eigenvalues required to get the estimator of Σ^{-1}. However, in the setting of our function, we fixed the largest number of eigenvalues to be 150, which leads to the results in these two cases. For $tol = 1.5 \times 10^8$, the classification did not perform much better. But we noticed that when $tol = 2 \times 10^8$ and $M1 = 9 \times 10^6$, the tapering method achieves the smallest misclassification rate 2.2941176%. For $tol = 2.5 \times 10^8$ and above, their classification performation did not get much better, but in fact, it became worse and worse, with 11.764706% as the smallest misclassification rate of both $tol = 2.5 \times 10^8$ and $tol = 3 \times 10^8$.

We plot the image of tapering matrix T when $M1 = 9 \times 10^6$ in Figure 4. The black color indicates that the value in B is 1, the white one shows that the value in B is 0 and the grey area corresponds to the value between 0 and 1. When the color is darker, the value of the corresponding matrix element is closer to 1.

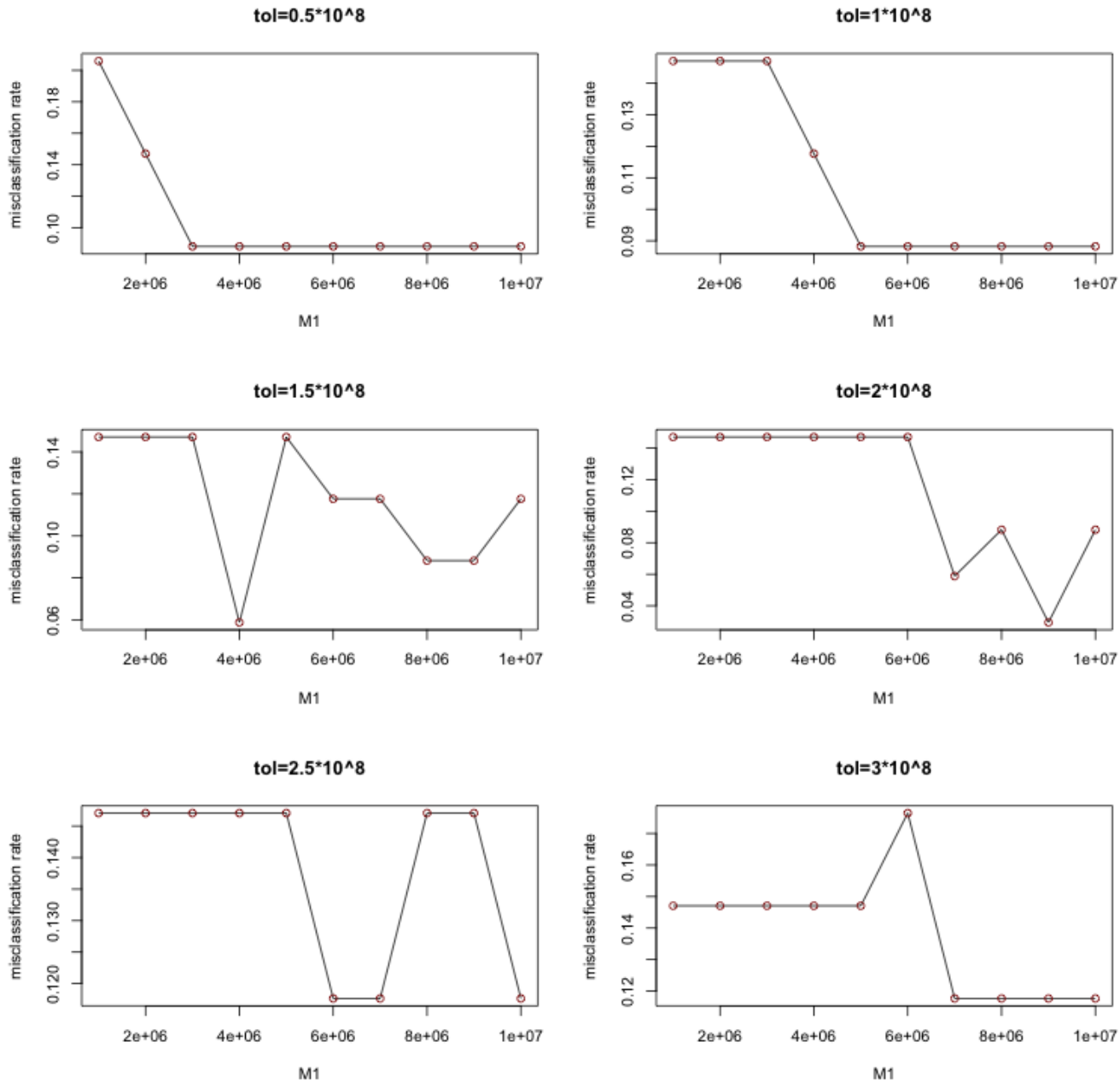

Figure 3: Misclassification Rate with tapering wrt M1 and tol

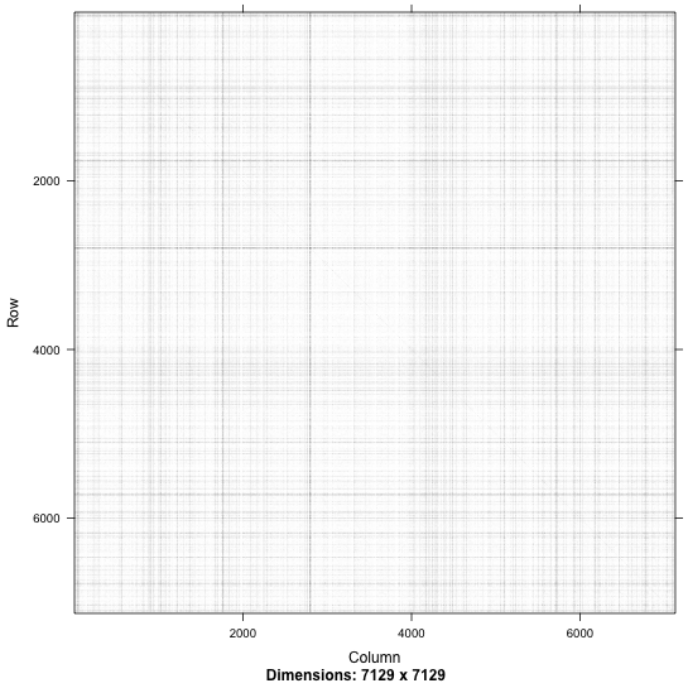

Figure 4: Tapering Matrix T with $M1 = 9 \times 10^6$

4.5 LDA with adaptive block thresholding estimation

For the LDA with the application of the adaptive block thresholding covariance estimation, we cannot directly apply this method like the above two methods. In order to use this adaptive covariance estimation method, we notice that there is an assumption of the existance of the "bandable" matrice. To deal with this assumption, we tried to resort the order of the variables to make the resorted sample covariance matrix closer to the "bandable" assumption.

We constructed a R function Index_order() to find the best order of the variables. We first find the largest value and its corresponding row number and column number from all the elements of the sample covarince matrix except the diagonal entries, and set the row number to be the first variable index and the column number to be the second. Then we locate the second variable and find the variable that has the largest covariance with the second variable, where we do not consider the first variable chosen. The variable found in this step is the third variable in the order. Next, we locate the third variable and find the varialbe that has the largest covariance with the second variable chosen, where we kick out the first and second variable chosen. This chosend variable is the forth variable in order. This procedure is continued until no variable is left.

In addition, in this case, we use the number of eigenvalues applied n_eig as the parameter instead of tol. From Figure 5, we could tell that the misclassification rate decreases roughly as the number of eigenvalues increases, although there might be some fluctuations from the decreasing curve. Also, we note that the smallest misclassification rate is 5.882353%. This value is first achieved when the number of eigenvalues is $n_{eig} = 230$.Even if the number of eigenvalues becomes larger, the classification performance still keeps fixed with the same misclassification rate 5.882353%. We would compare the numerical results of this method with the other methods mentioned above.

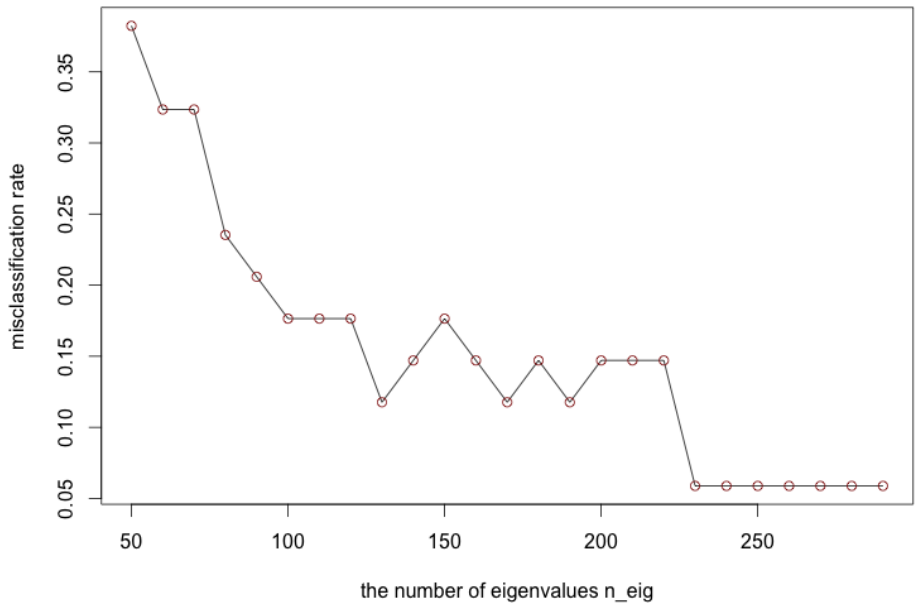

Figure 5: Misclassification rate with block thresholding wrt n_eig

4.6 Comparision of the Results

First, we compared all the four LDA derived methods with the orginal LDA method. We, at the very first, knew that the misclassification rate of the LDA is 14.70588%. We noticed that the best performance of the LDA with only the sparse thresholding esimator $\hat{\delta}$ keeps the same regardness of the change of parameter M_2 for the sparse-delta threshold, so we made an assumption that M_2 does not influence the LDA in our paper that much. So we just let $M_2 = 300$ be fixed. We found that the LDA with banding covariance estimation achievs the lowest misclassification rate as 2.941176%. Also, the LDA with tapering estimation has the same lowest misclassification rate. In addition the LDA with block thresholding estimation has the best classification rate with the lowest misclassification rate 5.882353%. We compared these values and noted that except the one that only applied the sparse thresholding $\hat{\delta}$, all the other three derived LDA methods performed much better. Compared with the LDA, both the banding and tapering LDA reduce the misclassification rate by nearly 80%, and the block-thresholding LDA reduces by 60%, which indicates that all three derived LDA's improved to a large content.

Second, we compared the banding LDA with the tapering LDA. From Figure 2 and Figure 4, two images of banding matrix B and tapering matrix T with the same value of M_1, we could almost find that these two figures are similar, but with careful oberservation, we could tell that these two images has some difference. Imagine that these two plots are two square pieces of cloth, and the T "cloth" is more dirty than the "B" cloth. To check specifically, we amplied the images of the three pairs of submatrixes, the first pair is $B[1:100, 1:100]$ and $T[1:100, 1:100]$, the second pair is $B[3000:3100, 3000:3100]$ and $T[3000:3100, 3000:3100]$, and the last pair is $B[6000:6100, 6000:6100]$ and $T[6000:6100, 6000:6100]$ (Figure 6). Now the relationship between these plots are much more clearly shown. The tapering matrix contains more non-zero entries, and thus contains more information. But still we should notice that the lowest misclassification rates are the same between these two methods.

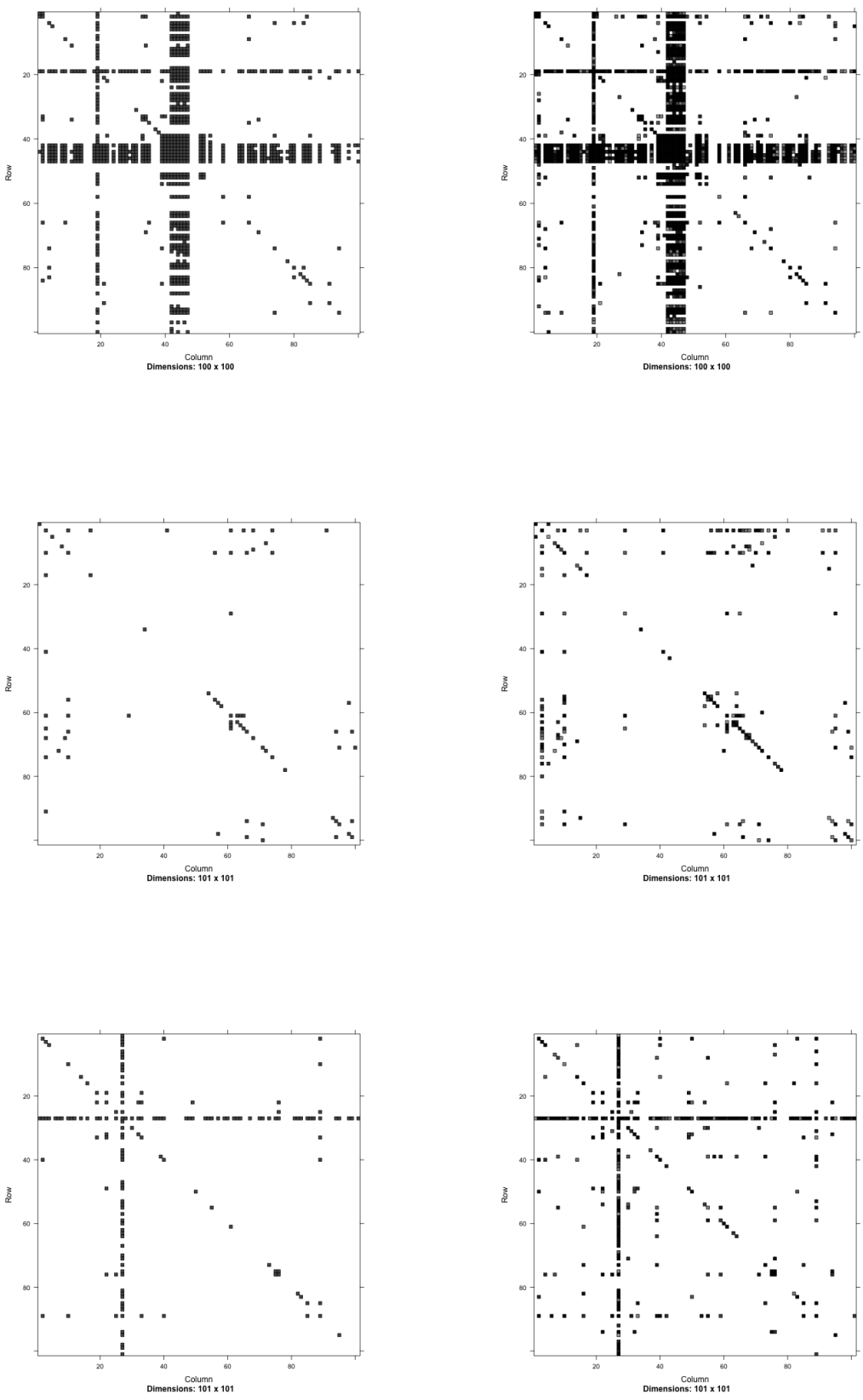

Figure 6: Comparison of submatrixes between Banding matrix B and Tapering matrix T

Last but not least, we needed to pay attention to our block thresholding LDA method here. We found that the misclassification rate of this LDA method 5.882353% is not lowest among all these method. Apparently, we knew that the lowest misclassification rate of the banding and tapering LDA's is 2.941176%, which is 50% lower. This result is not surprising, because we use the rearranged sample covariance matrix to try to be close to the assumption of "bandable" matrices. Thus, as seen from the results, our rearrangement method cannot be not optimal. But still this method performs well as it efficiently reduced the misclassification rate by nearly 60%.

5 Conclusion

In this paper we considered the high-dimensional setting with large dimension p and small sample size n, where $p >> n$, and tried to apply the covariance estimation of this high-dimensional setting to two-class classification problem. We made some review of several covariance estimation methods based on the assumption of "bandable" matrices, combined the essential techniques from some previous papers together, derived some new LDA methods and applied all LDA's to process with our leukemias dataset.

We first introduced the well-known LDA, and implemented five different ways to estimate the covariance matrix, which are the original LDA (with a regular sample covariance matrix), LDA with a thresholded estimator of the different between the mean of two classes,δ, Sparse LDA with banding estimator (following the method of Bickel and Levina, 2008 and Shao et al., 2011), Sparse LDA with tapering estimator (following the method of Cai, Zhang and Zhou, 2010), and Sparse LDA with adaptive block-thresholding estimator(following the method of Cai and Yuan, 2012). In this process, we needed to find the estimator of the inverse of covariance matrix $\hat{\Sigma}$. Hence we carefully illustrated how to make $\hat{\Sigma}$ positive semi-definite and positive definite.

We applied all these techinques to deal with our two-class classification problem. By analyzing the misclassification rate of the test data. We checked the performance of all sparse LDA's, and found the both banding and tapering LDA's worked best among these LDA's, and the block thresholding LDA empirically performed well but not best. But in theoretical aspects, we found that when changing the banding estimator with tapering estimator or adaptive block-thresholding estimator, the properties of conditional misclassification rate still remain.

Appendix I

Proof-Misclassification Rate Using the Tapering Estimator

According to the definition of LDA, given $\mathbf{X}$, the conditional misclassification rate $R_{SLDA}(\mathbf{X})$ is

$$\frac{1}{2}\sum_{k=1}^{2}\phi\left\{\frac{(-1)^k\tilde{\delta}'(\mu_k-\bar{x}_k)-\hat{\delta}'\tilde{\Sigma}^{-1}\tilde{\delta}/2)}{\sqrt{\tilde{\delta}'\tilde{\Sigma}^{-1}\Sigma\tilde{\Sigma}^{-1}\tilde{\delta}}}\right\}$$

According to (3) and (4), the tapering estimator $\tilde{\Sigma}$ we proposed satisfies

$$||\tilde{\Sigma}-\Sigma|| = O_p(d_n), \quad and \;\; ||\tilde{\Sigma}^{-1}-\Sigma^{-1}|| = O_p(d_n)$$

Hence we can derive

$$\tilde{\delta}'\tilde{\Sigma}^{-1}\Sigma\tilde{\Sigma}^{-1}\tilde{\delta} = \tilde{\delta}'\tilde{\Sigma}^{-1}\tilde{\delta}[O_p(d_n)] = \tilde{\delta}'\Sigma^{-1}\tilde{\delta}[O_p(d_n)]$$

Our next step is to show that

$$\frac{\tilde{\delta}'\tilde{\Sigma}^{-1}(\hat{x}_1-\mu_1)}{\sqrt{\tilde{\delta}'\tilde{\Sigma}^{-1}\Sigma\tilde{\Sigma}^{-1}\tilde{\delta}}} = \frac{O_p(\sqrt{k_n})+O_p(\sqrt{C_{h,p}q_n/n})}{\sqrt{1+O_p(d_n)}} \tag{12}$$

We partition δ into two part. Let $\delta=(\delta_1',\delta_0')'$, and $\tilde{\delta}=(\tilde{\delta}_1',\mathbf{0}')'$ where $\tilde{\delta}_1'$ is the $\hat{q}$-vector containing nonzero components of $\tilde{\delta}$ and δ_1 has $\hat{q}$ dimensions, where $\hat{q}$ is the number of j's with $|\hat{\delta}_j|>a_n$.
It is easy to show that $||\tilde{\delta}_1-\delta_1||^2 = O_P(q_n/n)$ and $||\delta_0||^2 = O(a_n^{2(1-g)}D_{g,p})$. We define
$k_n = max\{a_n^{2(1-g)}D_{g,p}, q_n/n\}$, then $||\tilde{\delta}-\delta||^2 = ||\tilde{\delta}_1-\delta_1||^2+||\delta_0||^2 = O_P(k_n)$. Hence we get

$$\tilde{\delta}_\prime\Sigma^{-1}\tilde{\delta} = \Delta_p^2+2\delta'\Sigma(\tilde{\delta}-\delta)+(\tilde{\delta}-\delta)'\Sigma^{-1}(\tilde{\delta}-\delta) = \Delta_p^2[1+O_P(\sqrt{k_n}/\Delta_p)]$$

According to the partition of δ, we write

$$\Sigma=\begin{pmatrix}\Sigma_1 & \Sigma_{12}\\ \Sigma_{12}' & \Sigma_2\end{pmatrix}, \quad \Sigma^{-1}=\begin{pmatrix}C_1 & C_{12}\\ C_{12}' & C_2\end{pmatrix}, \quad C_{12}=-\Sigma_1^{-1}\Sigma_{12}C_2$$

Similarly,

$$\tilde{\Sigma}=\begin{pmatrix}\tilde{\Sigma}_1 & \tilde{\Sigma}_{12}\\ \tilde{\Sigma}_{12}' & \tilde{\Sigma}_2\end{pmatrix}, \quad \Sigma^{-1}=\begin{pmatrix}\tilde{C}_1 & \tilde{C}_{12}\\ \tilde{C}_{12}' & \tilde{C}_2\end{pmatrix}, \quad \tilde{C}_{12}=-\tilde{\Sigma}_1^{-1}\tilde{\Sigma}_{12}\tilde{C}_2$$

Here Σ_1, $\tilde{\Sigma_1}$, C_1 and $\tilde{C}_1$ are $q_n\times q_n$ matrices with q_n is the number of j's with $|\delta_j|>a_n/r$.
Then we define $\check{\delta}_1=(\tilde{\delta}_1',\mathbf{0}')'$ and $\bar{x}_1-\mu_1=(\xi_1',\xi_0')'$ where $\check{\delta}_1$ and ξ_1 are also of dimension q_n. We have

$$\tilde{\delta}'\tilde{\Sigma}^{-1}(\hat{x}_1-\mu_1) = \check{\delta}_1{}'\tilde{C}_1\xi_1+\check{\delta}_1{}'\tilde{C}_{12}\xi_0 = \check{\delta}_1{}'\tilde{C}_1\xi_1-\check{\delta}_1{}'\tilde{\Sigma}_1^{-1}\tilde{\Sigma}_{12}\tilde{C}_2\xi_0,$$

where

$$(\check{\delta}_1{}'\tilde{C}_1\xi_1)^2 \leqslant (\check{\delta}_1{}'\tilde{C}_1\xi_1)(\check{\delta}_1{}'\tilde{C}_1\check{\delta}_1) = O_P(q_n/n)(\tilde{\delta}'\tilde{\Sigma}^{-1}\tilde{\delta})$$

, and hence

$$\check{\delta}_1{}'\tilde{C}_1\xi_1 = O_P(\sqrt{k_n})(\sqrt{\tilde{\delta}'\tilde{\Sigma}^{-1}\tilde{\delta}}).$$

Furthermore,

$$(\check{\delta}_1{}'\tilde{\Sigma}_1^{-1}\tilde{\Sigma}_{12}\tilde{C}_2\xi_0)^2 \leqslant (\check{\delta}_1{}'\tilde{\Sigma}_1^{-1}\check{\delta}_1)(\xi_0'\tilde{C}_2\tilde{\Sigma}_{12}'\tilde{\Sigma}_1^{-1}\tilde{\Sigma}_{12}\tilde{C}_2\xi_0) \leqslant (\tilde{\delta}'\tilde{\Sigma}\tilde{\delta})(\xi_0'\tilde{C}_2\tilde{\Sigma}_{12}'\tilde{\Sigma}_1^{-1}\tilde{\Sigma}_{12}\tilde{C}_2\xi_0),$$

and hence under the condition $c_0^{-1} \leqslant all\ eigenvalues\ of\ \Sigma \leqslant c_0$,

$$\begin{aligned}
E[\xi_0'\tilde{C}_2\tilde{\Sigma}_{12}'\tilde{\Sigma}_1^{-1}\tilde{\Sigma}_{12}\tilde{C}_2\xi_0] &\leqslant c_0 E[\xi_0'\tilde{C}_2\tilde{\Sigma}_{12}'\tilde{\Sigma}_{12}\tilde{C}_2\xi_0] \\
&= c_0 n^{-1} trace(\Sigma_{12}C_2\Sigma_2C_2\Sigma_{12}') \\
&\leqslant c_0^4 n^{-1} trace(\Sigma_{12}\Sigma_{12}') \\
&= \frac{c_0^4}{n}\sum_{j=1}^{q_n}\sum_{l=q_n+1}^{p}\sigma_{jl}^2 \\
&\leqslant \frac{c_0^{6-h}q_n}{n} max_{l\leqslant p}\sum_{j=1}^{p}|\sigma_{jl}|^h \\
&= O(C_{h,p}q_n/n)
\end{aligned}$$

Therefore, (1) has been proved. Similarly, if we replace $\bar{x}_1 - \mu_1$ by $\bar{x}_2 - \mu_2$ or $\hat{\delta} - \delta$, the result will still hold. Hence we can derive

$$\begin{aligned}
\frac{(-1)^k\tilde{\delta}'(\mu_k - \bar{x}_k) - \hat{\delta}'\tilde{\Sigma}^{-1}\tilde{\delta}/2)}{\sqrt{\tilde{\delta}'\tilde{\Sigma}^{-1}\Sigma\tilde{\Sigma}^{-1}\tilde{\delta}}} &= \frac{O_P(\sqrt{k_n}) + O_P(\sqrt{C_{h,p}q_n/n})}{\sqrt{1+O_P(d_n)}} - \frac{\Delta_P\sqrt{1+O_P(\sqrt{k_n/\Delta_p})}}{2\sqrt{1+O_P(d_n)}} \\
&= O_P(\sqrt{k_n}) + O_P(\sqrt{C_{h,p}q_n/n}) - \frac{\Delta_p}{2}[1+O_P(\sqrt{k_n/\Delta_p}) + O_P(d_n)] \\
&= (-1)\frac{\Delta_p}{2} - \frac{\Delta_p}{2}O_P(b_n),
\end{aligned}$$

which proves the conditional misclassification rate with tapering estimator $\tilde{\Sigma}$. Furthermore, the properties of this rate also remains, following the proposal of Shao el al.

Appendix II

```
# Step1: Get the data
# Download the Leukaemia data: (Golub et al., 1999)
# Save the training and test data as well as the labels into one dataset fil
## try http:// if https:// URLs are not supported
source("https://bioconductor.org/biocLite.R")
library("golubEsets")
data(Golub_Train)
golub_train <- data.frame(Golub_Train)[1:7129]
data(Golub_Test)
golub_test <- data.frame(Golub_Test)[1:7129]
golub_train.pheno=pData(Golub_Train)
attach(golub_train.pheno)
golub_train.type = as.numeric(ALL.AML=="AML")
table(golub_train.type)
detach(golub_train.pheno)
golub_test.pheno=pData(Golub_Test)
attach(golub_test.pheno)
golub_test.type = as.numeric(ALL.AML=="AML")
table(golub_test.type)
detach(golub_test.pheno)
save(golub_train,golub_test,golub_train.type,golub_test.type, file = "saved


#=====================================================================
# Step2: Model 1:Use the basic LDA
p=7129
n=38
library(matrixcalc)
# construct a new function to get miu, delta and Sigma from a dataset X
# and its corresponding type vector y
Get_MDS<-function(X,y,n){
golub_train1=X[y==0,]
golub_train2=X[y==1,]
mean1=colMeans(golub_train1)
mean2=colMeans(golub_train2)
# The estimator for miu
miu0=(mean1+mean2)/2
# The estimator for delta
delta0=mean1-mean2
# The sample covariance matrix
X1=golub_train1-mean1%*%t(c(rep(1,7129)))
X2=golub_train2-mean2%*%t(c(rep(1,7129)))
# n is the sample size
Sigma=(t(X1)%*%as.matrix(X1)+t(X2)%*%as.matrix(X2))/n
list0=list("Sigma"=Sigma,"delta"=delta0,"miu"=miu0)
```

```
return(list0)
}
a=Get_MDS(golub_train,golub_train.type,38)
Sigma=a$Sigma
delta0=a$delta
miu0=a$miu
#-------------------------------------------------------------------
# Next should find the inverse of Sigma
# we use the function eigs_sym() in package rARPACK to find the first 50 eig
library(rARPACK)
# Here we construct a function Solve_sigma() to find the estimator of Sigma^
Solve_sigma<-function(Sigma,tol){
svd0<-eigs_sym(Sigma,150,which="LA")
a=svd0$values>tol
if (sum(as.numeric(a))==150)
{print("150 is not enough")}
U=svd0$vectors[,a]
DIAG=diag(svd0$values[a]^(-1))
Sigma_plus_inverse=U%*%DIAG%*%t(U)
return(Sigma_plus_inverse)
}
Sigma_inverse=Solve_sigma(Sigma,tol)
#-------------------------------------------------------------------
#Now we have Sigma_inverse, delta0 and miu0, and then we have the classifica
# we use the classification rule to find the misclassfication rate
Rate_mis<-function(Sigma_inverse,delta,miu,type){
type_predict=matrix(0,1,34)
for (i in 1:34){
X=golub_test[i,]
if (t(delta0)%*%Sigma_inverse%*%(t(X)-miu0)>=0)
type_predict[i]=0
else type_predict[i]=1
}
rate=sum(as.numeric(type_predict!=type))/34
return(rate)
}
# Apply the above function to find the misclassification rate
rate_LDA0=Rate_mis(Sigma_inverse,delta0,miu0,golub_test.type)
rate_LDA0 # 0.1470588


#============================================================
# Each following method apply the same thresholding to the estimate of delta
alpha=0.3
M2=300
delta_thr<-function(delta0,alpha,M2){
delta1=delta0*as.numeric(abs(delta0)>M2*(log(p)/n)^alpha)
```

```
return(delta1)
}

#=====================================================================
# Step 3: Model 2:a slightly different LDA, only threshold delta
Rate_LDA1=matrix(0,1,10)
for (i in 1:10){
M2=50*i
delta1<-delta_thr(a$delta,alpha,M2)
Rate_LDA1[i]=Rate_mis(Sigma_inverse,delta1,a$miu,golub_test.type)
}
Rate_LDA1 #all are 0.1470588,keep the same

#=====================================================================
# Step 4: Model 3: banding estimation
Sigma_B<-function(Sigma,M1){
SigmaB=Sigma
SigmaB[abs(SigmaB)<M1]=0
return(SigmaB)
}
Rate_LDA2=matrix(0,4,10)
M2=300
for(j in 1:4){
print("j is",j)
tol=j*5*10^7
for (i in 1:10){
print(i)
M1=i*10^6
SigmaB=Sigma_B(a$Sigma,M1)
SigmaB_inverse=Solve_sigma(SigmaB,tol)
delta1<-delta_thr(a$delta,alpha,M2)
Rate_LDA2[j,i]=Rate_mis(SigmaB_inverse,delta1,a$miu,golub_test.type)
}
}
attach(mtcars)
par(mfrow=c(2,2))
M1=10^6*c(1:10)
plot(M1,Rate_LDA2[1,],type="l",xlab="M1",ylab="misclassification rate", mai
points(M1,Rate_LDA2[1,],cex=1,col="dark red")
plot(M1,Rate_LDA2[2,],type="l",xlab="M1",ylab="misclassification rate", mai
points(M1,Rate_LDA2[2,],cex=1,col="dark red")
plot(M1,Rate_LDA2[3,],type="l",xlab="M1",ylab="misclassification rate", mai
points(M1,Rate_LDA2[3,],cex=1,col="dark red")
plot(M1,Rate_LDA2[4,],type="l",xlab="M1",ylab="misclassification rate", mai
points(M1,Rate_LDA2[4,],cex=1,col="dark red")
# Check the image of banding matrix B when tol=10^8 and M1=9*10^6
library(SparseM)
```

```
# tol=10^8
M1=9*10^6
B=matrix(1,p,p)
index=abs(a$Sigma)<M1
B[index]=0
B_sparse=Matrix(B,sparse=TRUE)
image(B_sparse)
image1=image(B_sparse[1:100,1:100])
image2=image(B_sparse[6000:6100,6000:6100])
image1
image2
#=============================================================================
# Step 5: Model 4: tapering estimation
Sigma_T<-function(Sigma,M1){
T=matrix(1,p,p)
T[abs(Sigma)<M1/2]=0
index=abs(Sigma)<M1&abs(Sigma)>=M1/2
T[index]=(2-2*abs(Sigma)/M1)[index]
SigmaT=Sigma*T
return(SigmaT)
}
Rate_LDA3=matrix(0,4,10)
for (j in 1:6){
tol=j*0.5*10^8
for (i in 1:10){
print(i)
M1=i*10^6
SigmaT=Sigma_T(a$Sigma,M1)
SigmaT_inverse=Solve_sigma(SigmaT,tol)
delta1<-delta_thr(a$delta,alpha,M2)
Rate_LDA3[j,i]=Rate_mis(SigmaT_inverse,delta1,a$miu,golub_test.type)
}
}
attach(mtcars)
par(mfrow=c(3,2))
M1=10^6*c(1:10)
j=1
plot(M1,Rate_LDA3[j,],type="l",xlab="M1",ylab="misclassification rate", mai
points(M1,Rate_LDA3[j,],cex=1,col="dark red")
j=2
plot(M1,Rate_LDA3[j,],type="l",xlab="M1",ylab="misclassification rate", mai
points(M1,Rate_LDA3[j,],cex=1,col="dark red")
j=3
plot(M1,Rate_LDA3[j,],type="l",xlab="M1",ylab="misclassification rate", mai
points(M1,Rate_LDA3[j,],cex=1,col="dark red")
j=4
plot(M1,Rate_LDA3[j,],type="l",xlab="M1",ylab="misclassification rate", mai
```

```
points(M1,Rate_LDA3[j,],cex=1,col="dark red")
j=5
plot(M1,Rate_LDA3[j,],type="l",xlab="M1",ylab="misclassification rate", main
points(M1,Rate_LDA3[j,],cex=1,col="dark red")
j=6
plot(M1,Rate_LDA3[j,],type="l",xlab="M1",ylab="misclassification rate", main
points(M1,Rate_LDA3[j,],cex=1,col="dark red")
# Check the image of tapering matrix T when tol=2*10^8, M1=9*10^6
# tol=2*10^8
M1=9*10^6
T=matrix(1,p,p)
T[abs(Sigma)<M1/2]=0
index=abs(Sigma)<M1&abs(Sigma)>=M1/2
T[index]=(2-2*abs(Sigma)/M1)[index]
T_sparse=Matrix(T,sparse=TRUE)
image(T_sparse)
image3=image(T_sparse[1:100,1:100])
image4=image(T_sparse[6000:6100,6000:6100])
image3
image4
#=============================================================================
# Step 6: Model5: Block Thresholding
# We should notice that the assuption is bandable matrice
# We construct the following function to find the best order
# This is to make the matrix close to the "bandable" definition
p=7129
Index_order<-function(Sigma){
index_order=matrix(0,1,p)
Sigma0=abs(Sigma)
diag(Sigma0)=0
index_order[1]=which(Sigma0==max(Sigma0),arr.ind=TRUE)[1]
index_order[2]=which(Sigma0==max(Sigma0),arr.ind=TRUE)[2]
l=index_order[2]
for (i in 2:p-1){
index_a=index_order[1:i]
vector=Sigma0[l,]
vector[index_a]=0
l=which.max(vector)
index_order[i+1]=l
}
return(index_order)
}

# We construct a new Solve_inverse1() wrt the number of eigenvalues n_eig c
# here, we don't consider tol any more
# This is for the case when Solve_inverse() is used, "not enough" is always
Solve_sigma1<-function(Sigma,n_eig){
```

```
svd0<-eigs_sym(Sigma,n_eig,which="LA")
U=svd0$vectors
DIAG=diag(svd0$values^(-1))
Sigma_plus_inverse=U%*%DIAG%*%t(U)
return(Sigma_plus_inverse)
}

# adaptive estimator
# lambda0 is the parameter that can be changed
# We construct a function to get the block thresholding covariance estimator
Sigma_BT<-function(Sigma){
# construct the block thresholding estimator
# k0=log(p)=8
# n/log(n)=10.45
# So in this case, need to keep the diagonal blocks
Sigma_BT=matrix(0,p,p)
# we find that 7129=891*8+1, so the last block on the diagonal is of size 1
# spectral norms of the diagonal blocks are stored in a vector
norm_diag=matrix(0,892)
for (i in 1:891){
a=8*(i-1)
Sigma_BT[(a+1):(a+8),(a+1):(a+8)]=Sigma[(a+1):(a+8),(a+1):(a+8)]
norm_diag[i]=spectral.norm(Sigma[(a+1):(a+8),(a+1):(a+8)])
}
Sigma_BT[7129,7129]=Sigma[7129,7129]
norm_diag[892]=spectral.norm(Sigma[7129,7129])
lambda0=6
# threshold blocks that d(B)=k0 except the diagonal ones
# Also need to threshold one block lies in the last row and the last column
add_matrix=matrix(0,p,p)
# 1+445*2=891
for (j in 1:445){
b=16*(j-1)
a1=(b+1):(b+8)
a2=(b+9):(b+16)
a3=(b+17):(b+24)
b1=sqrt((8+log(p))/n)
add_matrix[a1,a2]=Sigma[a1,a2]*(spectral.norm(Sigma[a1,a2])>lambda0*b1*sqrt(
add_matrix[a1,a3]=Sigma[a1,a3]*(spectral.norm(Sigma[a1,a3])>lambda0*b1*sqrt(
add_matrix[a2,a3]=Sigma[a2,a3]*(spectral.norm(Sigma[a2,a3])>lambda0*b1*sqrt(
}
add_matrix[7121:7128,7129]=Sigma[7121:7128,7129]*(spectral.norm(Sigma[7121:7
# kill other blocks, no need to write code, as Sigma_BT is initiated with a
# The final covariance estimator is as follows:
Sigma_BT=Sigma_BT+add_matrix+t(add_matrix)
return(Sigma_BT)
}
```

```
n=38
a=Get_MDS(golub_train,golub_train.type,n)
IO=Index_order(a$Sigma)
SigmaBT=Sigma_BT(a$Sigma[IO,IO])
alpha=0.3
M2=300
Rate_LDA4=matrix(0,1,25)
for (i in c(21:25)){
print(i)
n_eig=10*i+40
SigmaBT_inverse=Solve_sigma1(SigmaBT,n_eig)
delta1<-delta_thr(a$delta,alpha,M2)
Rate_LDA4[i]=Rate_mis(SigmaBT_inverse,delta1[IO],a$miu[IO],golub_test.type)
}

plot.new()
par(mfrow=c(1,1))
n_eig=seq(50,290,by=10)
plot(n_eig,Rate_LDA3,type="l",xlab="the number of eigenvalues n_eig", ylab="
points(n_eig,Rate_LDA3,cex=1,col="dark red")
```

References

Banerjee, Onureena, Laurent El Ghaoui, and Alexandre d'Aspremont. "Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data." The Journal of Machine Learning Research 9 (2008): 485-516.

Bickel, Peter J., and Elizaveta Levina. "Some theory for Fisher's linear discriminant function,'naive Bayes', and some alternatives when there are many more variables than observations." Bernoulli (2004): 989-1010.

Bickel, Peter J., and Elizaveta Levina. "Covariance regularization by thresholding." The Annals of Statistics (2008): 2577-2604.

Cai, T. Tony, Cun-Hui Zhang, and Harrison H. Zhou. "Optimal rates of convergence for covariance matrix estimation." The Annals of Statistics 38.4 (2010): 2118-2144.

Cai, Tony, and Weidong Liu. "Adaptive thresholding for sparse covariance matrix estimation." Journal of the American Statistical Association 106.494 (2011): 672-684.

Cai, T. Tony, and Ming Yuan. "Adaptive Covariance Matrix Estimation Through Block Thresholding". Ann. Statist. 40.4 (2012): 2014-2042. Web.

Cai, T. Tony, Zhao Ren, and Harrison H. Zhou. "Estimating Structured High-Dimensional Covariance And Precision Matrices: Optimal Rates And Adaptive Estimation". Electronic Journal of Statistics 10 (2016): 1-59. Print.

Dudoit, Sandrine, Jane Fridlyand, and Terence P Speed. "Comparison Of Discrimination Methods For The Classification Of Tumors Using Gene Expression Data". Journal of the American Statistical Association 97.457 (2002): 77-87. Web.

Fan, Jianqing, Yingying Fan, and Jinchi Lv. "High dimensional covariance matrix estimation using a factor model." Journal of Econometrics 147.1 (2008): 186-197.

Golub, T. R. "Molecular Classification Of Cancer: Class Discovery And Class Prediction By Gene Expression Monitoring". Science 286.5439 (1999): 531-537. Web.

Hoffbeck, Joseph P., and David A. Landgrebe. "Covariance matrix estimation and classification with limited training data." IEEE Transactions on Pattern Analysis & Machine Intelligence 7 (1996): 763-767.

Shao, Jun et al. "Sparse Linear Discriminant Analysis By Thresholding For High Dimensional Data". The Annals of Statistics 39.2 (2011). Print.