Making Fisher Discriminant Analysis Scalable

Your Name EMAIL@YOURDOMAIN.EDU
Your Fantastic Institute, 314159 Pi St., Palo Alto, CA 94306 USA

Your CoAuthor's Name EMAIL@COAUTHORDOMAIN.EDU
Their Fantastic Institute, 27182 Exp St., Toronto, ON M6H 2T1 CANADA

Abstract

The Fisher linear discriminant analysis (LDA) is a classical method for classification and dimension reduction jointly. A major limitation of the conventional LDA is a so-called singularity issue. Many LDA variants, especially two-stage methods such as PCA+LDA and LDA/QR, were proposed to solve this issue. In the two-stage methods, an intermediate stage for dimension reduction is developed before the actual LDA method works. These two-stage methods are scalable because they are an approximate alternative of the LDA method. However, there is no theoretical analysis on how well they approximate the conventional LDA problem. In this paper we present theoretical analysis on the approximation error of a two-stage algorithm. Accordingly, we develop a new two-stage algorithm. Furthermore, we resort to a random projection approach, making our algorithm scalable. We also provide an implemention on distributed system to handle large scale problems. Our algorithm takes LDA/QR as its special case, and outperforms PCA+LDA while having a similar scalability. We also generalize our algorithm to kernel discriminant analysis, a nonlinear version of the classical LDA. Extensive experiments show that our algorithms outperform PCA+LDA and have a similar scalability with it.

1. Introduction

The Fisher linear discriminant analysis (LDA) has received wide applications in multivariate analysis and machine learning such as face recognition (Belhumeur et al., 1997; Martínez & Kak, 2001), text classification, microarray data classification, etc. The conventional LDA problem attempts to find an optimal linear transformation by minimizing the total class distance and maximizing the between-class distance simultaneously. It is well known that this optimization problem can be solved by applying eigenvalue decomposition to the scatter matrices. However, this requires the total scatter matrix to be nonsingular, which is usually not the case in real world applications. For example, we usually meet microarray datasets which have the "large p but small n" regime.

To address this singularity issue, the pseudoinverse and regularization methods have been widely employed (Hastie et al., 2009; Zhang et al., 2010). Recently, a two-stage approximate approach, such as the PCA+LDA algorithm (Belhumeur et al., 1997) and the LDA/QR algorithm (Ye & Li, 2005), has been also proposed. Typically, the two-stage approach applies a cheap intermediate dimension reduction method before the conventional LDA algorithm is performed. This class of algorithms is more scalable than the exact algorithm. To the best of our knowledge, however, there is no theoretical analysis on how well these algorithms approximate the exact LDA problem.

In this paper we first give theoretical analysis on how well a two-stage algorithm approximates the exact LDA in the sense of maximizing the LDA objective function. The theoretical analysis motivates us to devise a new two-stage LDA algorithm. Our algorithm outperforms the PCA+LDA (Belhumeur et al., 1997) while both have the similar scalability. To make our algorithm more scalable, we introduce a random-projection-based SVD (RSVD) method (Mahoney, 2011; Halko et al., 2011). Furthermore, we provide an implemention of our algorithm on distributed system to handle large scale problems. Additionally, we also show that our algorithm could naturally generalize to the kernel discriminant analysis (KDA) problem (Mika et al., 1999; Baudat & Anouar, 2000; Park & Park, 2005), which is a nonlinear generalization of the conventional LDA.

The remainder of the paper is organized as follows. In Section 2, we review the classical LDA and two two-stage al-

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

gorithms for efficiently solving LDA, the PCA+LDA algorithm and the LDA/QR algorithm. In Section 3, we present our algorithm and some theoretical analysis. We conduct empirical analysis and comparison in Section 4, and conclude our work in Section 5.

2. The Fisher Discriminant Analysis

In this section we briefly review the conventional Fisher LDA and KDA problems, as well as efficient algorithms for solving them.

2.1. Linear Discriminant Analysis

We are given a data matrix $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^T \in \mathbb{R}^{n\times p}$ where $\mathbf{x}_i$ is a p-dimensional input instance. Suppose the input instances are partitioned into m classes, such that $X^T = [X_1^T, X_2^T, \ldots, X_m^T]$ where $X_i \in \mathbb{R}^{n_i\times p}$ contains n_i instances from the i-th class and $\sum_{i=1}^m n_i = n$. The conventional LDA is to find the optimal linear transformation $G \in \mathbb{R}^{p\times q}$ that preserves the class structure in a low-dimensional space as well as in the original space. That is, G maps each $\mathbf{x}_i$ of X in the p-dimensional space to a vector $\mathbf{y}_i$ in the q-dimensional space.

The between-class, and total scatter matrices are defined by

$$S_w = \frac{1}{n}\sum_{i=1}^{m}\sum_{\mathbf{x}\in X_i}(\mathbf{x}-\mathbf{c}_i)(\mathbf{x}-\mathbf{c}_i)^T,$$
$$S_b = \frac{1}{n}\sum_{i=1}^{m} n_i(\mathbf{c}_i-\mathbf{c})(\mathbf{c}_i-\mathbf{c})^T,$$
$$S_t = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i-\mathbf{c})(\mathbf{x}_i-\mathbf{c})^T,$$

where $\mathbf{c}_i = \frac{1}{n_i}\sum_{\mathbf{x}\in X_i}\mathbf{x}$ is the mean of the j-th class and $\mathbf{c} = \frac{1}{n}\sum_{\mathbf{x}\in X}\mathbf{x}$ is the mean of the whole data set. In the low-dimensional space defined by the linear transformation G, the between-class, and total scatter matrices become $S_b^G = G^T S_b G$, and $S_t^G = G^T S_t G$, respectively.

We can simplify the formulation of the scatter matrices through precursors H_b, and H_t as

$$H_b = \frac{1}{\sqrt{n}}\left[\sqrt{n_1}(\mathbf{c}_1-\mathbf{c}), \ldots, \sqrt{n_m}(\mathbf{c}_m-\mathbf{c})\right], \quad (1)$$
$$H_w = \frac{1}{\sqrt{n}}\left[X_1^T - \mathbf{c}_1\mathbf{e}_1^T, \cdots, X_m^T - \mathbf{c}_m\mathbf{e}_m^T\right],$$
$$H_t = \frac{1}{\sqrt{n}}(X^T - \mathbf{c}\mathbf{e}^T), \quad (2)$$

where $\mathbf{e}_i = [1, \cdots, 1]^T \in \mathbb{R}^{n_i}$ and $\mathbf{e} = [1, \ldots, 1]^T \in \mathbb{R}^n$. Hence, it can be shown that we can rewrite the scatter matrices as $S_b = H_b H_b^T$, and $S_t = H_t H_t^T$.

An optimal transformation G can be obtained by solving the following optimization problem:

$$\underset{G}{\operatorname{argmax}}\left\{J(G) \triangleq \operatorname{trace}\left((S_t^G)^{-1}S_b^G\right)\right\}.$$

The solution to the optimization can be obtained through eigen-decomposition of the matrix $S_t^{-1}S_b$ whenever S_t is nonsingular (Fukunaga, 1990). However, when S_t is singular, we can use the eigen-decomposition of $S_t^\dagger S_b$, where $S_t^\dagger$ is the Moore-Penrose inverse of S_t.

Recently, an SVD-based algorithm was proposed to solve the eigen problem(Ye, 2005), which is described in Algorithm 1. Let G^* be obtained from Algorithm 1. Then it consists of the top q eigenvectors of $S_t^\dagger S_b$. Here $q = \operatorname{rank}(H_b)$ which is usually $m-1$ in most cases. We also have $(G^*)^T S_t G^* = I_q$ and $(G^*)^T S_b G^* = \Sigma_q^2$. Hence the objective function of LDA is actually $\operatorname{trace}(\Sigma_q^2)$.

Algorithm 1 The SVD based LDA algorithm

1: calculate H_t and H_b (2)
2: compute the reduced SVD of H_t as $H_t = U_t\Sigma_t V_t^T$
3: $B \leftarrow \Sigma_t^{-1}U_t^T H_b$
4: compute the reduced SVD of B as $B = P_q\Sigma_q Q_q^T$
5: **return** $G^* \leftarrow U_t\Sigma_t^{-1}P_q$

2.2. Kernel Discriminant Analysis

To apply LDA to nonlinear data, many KDA algorithms have been devised by using a so-called kernel trick (Mika et al., 1999; Baudat & Anouar, 2000; Park & Park, 2005). The kernel method first maps the original data into a high-dimensional space $\mathcal{H}$ by a nonlinear transformation $\Phi : \mathbb{R}^p \to \mathcal{H}$. Typically, Φ is implicitly available and we only know a kernel function $\kappa : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ such that $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \Phi(\mathbf{x}_1)^T\Phi(\mathbf{x}_2)$.

Let $K_b = [b_{ij}]_{1\le i\le n, 1\le j\le m}$ where

$$b_{ij} = \sqrt{n_j}\Big(\frac{1}{n_j}\sum_{\mathbf{x}_k\in X_j}\kappa(\mathbf{x}_i,\mathbf{x}_k) - \frac{1}{n}\sum_{k=1}^{n}\kappa(\mathbf{x}_i,\mathbf{x}_k)\Big), \quad (3)$$

and $K_t = [t_{ij}]_{1\le i\le n, 1\le j\le n}$ where

$$t_{ij} = \sqrt{n}\Big(\kappa(\mathbf{x}_i,\mathbf{x}_j) - \frac{1}{n}\sum_{k=1}^{n}\kappa(\mathbf{x}_i,\mathbf{x}_j)\Big). \quad (4)$$

Then the objective function of KDA becomes

$$\tilde{J}(\mathcal{G}) = \operatorname{trace}\left((A^T K_t K_t^T A)^{-1}(A^T K_b K_b^T A)\right),$$

where $A = [a_{ij}]_{1\le i\le n, 1\le j\le q}$ satisfies $[\mathcal{G}(\mathbf{x})]_j = \sum_{i=1}^n a_{ij}\kappa(\mathbf{x}, \mathbf{x}_i)$ for any $\mathbf{x} \in \mathbb{R}^p$ (Park & Park, 2005).

This objective function has a similar form with the LDA objective function. That is, it replaces H_b with K_b and H_t with K_t. Therefore, Algorithm 1 could be easily applied to KDA. Here we omit the details.

2.3. Two-stage methods

There exist two important two-stage methods for the conventional LDA, namely, PCA+LDA (Belhumeur et al., 1997; Yang & Yang, 2003) and LDA/QR (Ye & Li, 2005; Ye, 2005) or OCM+LDA (Park et al., 2003). These two algorithms first yield a column-orthogonal transformation matrix Z in the first stage and then implement the classical LDA algorithm on the reduced space obtained via Z.

The PCA+LDA algorithm first computes Z_{PCA} to maximize $\mathrm{trace}(Z^T S_t Z)$, which is equivalent to computing the top r left singular vectors of H_t. LDA maximizes the trace of ratio of between the class scatter matrix and the total scatter matrix. However, PCA maximizes the trace of the total scatter matrix, which is not compatible to the objective function of LDA. So some useful information which may be important for discrimination might be lost in the PCA stage.

The LDA/QR first computes a column orthogonal matrix Z_{OCM} by maximizing $\mathrm{trace}(Z^T S_b Z)$. This objective function is more compatible to the objective function of LDA. Ye & Li (2005) showed that if the QR decomposition with pivoting (Golub & Loan, 1996) of H_b is $H_b = QR\Pi$, then we can take Z_{OCM} as Q. The LDA/QR algorithm reduces the dimension to q, which is too small for many problems, thus a lot of information in the original data matrix might be lost.

3. Methodology

In this section we first give a theoretical bound on how well a two-stage LDA algorithm approximates the exact LDA algorithm when the transformation matrix of the first stage is given. We then propose a fast two-stage LDA algorithm based on the theoretical analysis. We also generalize our algorithm to the KDA problem.

3.1. Theoretical Analysis

In the first stage of a two-stage algorithm, we introduce a linear transformation $Z \in \mathbb{R}^{p\times r}$ to reduce the dimension to r. We assume Z to be column-orthonormal, that is, $Z^T Z = I_r$. In the second stage we apply the conventional LDA algorithm on the reduced total scatter matrix $\tilde{S}_t = Z^T S_t Z$ and the reduced between-class scatter matrix $\tilde{S}_b = Z^T S_b Z$ to obtain a linear transformation $\tilde{G}$. The final result then becomes $\hat{G}^Z = Z\tilde{G}$.

Intuitively, a good Z should keep as much information as possible in both S_t and S_b. Fortunately, $\mathrm{rank}(S_b) = q \le m-1$ is small under the assumption that number of classes m is small. Therefore, if we let $\mathcal{R}(H_b) \subset \mathcal{R}(Z)$, all information will be kept after linear transformation Z. Here $\mathcal{R}(\cdot)$ is the range (or column-spanned space) of a matrix. The following theorem shows how well we can approximate the solution of exact LDA using such a Z.

Theorem 1 *If $Z \in \mathbb{R}^{p\times r}$ satisfies $\mathcal{R}(H_b) \subset \mathcal{R}(Z)$ and $Z^T Z = I_r$, then we have*

$$J(\hat{G}^Z) \ge \frac{1}{\|(G^*)^T Z Z^T H_t\|_2^2} J(G^*) \ge \frac{1}{\|H_t^\dagger Z Z^T H_t\|_2^2} J(G^*).$$

Here $\|\cdot\|_2$ is the spectral norm of a matrix, and G^* is the solution of Algorithm 1.

3.2. The SVD-QR-LDA Algorithm

To obtain a more accurate two-stage LDA algorithm, we can maximize the right-hand side of the inequality in Theorem 1. The optimization problem is

$$\min_Z \|H_t^\dagger Z Z^T H_t\|_2 \qquad \text{s.t} \quad Z^T Z = I_r, \quad \mathcal{R}(H_b) \subset \mathcal{R}(Z).$$

Unfortunately, it is hard to solve this minimization problem. Therefore, we use some heuristics to solve it approximately.

First, we observe that without the constraint $\mathcal{R}(H_b) \subset \mathcal{R}(Z)$, there is a closed-form solution to this maximization problem. Let $U \in \mathbb{R}^{p\times s}$ be the left singular vectors of H_t, where $s = \mathrm{rank}(H_t)$. For any index set $\mathcal{I} \subset \{1, \ldots, s\}$, let $U_\mathcal{I}$ be the columns of U indexed by $\mathcal{I}$. By direct calculation, we see that for any $\mathcal{I}$, $\|H_t^\dagger U_\mathcal{I} U_\mathcal{I}^T H_t\|_2 = 1$. Because we always have $\|H_t^\dagger Z Z^T H_t\|_2 \ge 1$ for any column-orthonormal matrix Z, $Z = U_\mathcal{I}$ is the desired solution.

Second, we simply add some columns to $U_\mathcal{I}$ to form $Z_\mathcal{I}$ such that all conditions in Theorem 1 are satisfied. We first do QR decomposition with pivoting such that $(I - U_\mathcal{I} U_\mathcal{I}^T) H_b = QR\Pi$. Then, $Z_\mathcal{I} = [U_\mathcal{I}, Q]$ is the required result. We can easily see that $Z_\mathcal{I}^T Z_\mathcal{I} = I$ and $\mathcal{R}(H_b) \subset \mathcal{R}(Z_\mathcal{I})$. Moreover, $Z_\mathcal{I}$ admits the following property:

Lemma 1 *The matrix $Z_\mathcal{I}$ described above satisfies*

$$\|H_t^\dagger Z_\mathcal{I} Z_\mathcal{I}^T H_t\|_2 \le \frac{\max_{1\le i\le s, i\notin\mathcal{I}} \sigma_i(H_t)}{\sigma_s(H_t)},$$

where $\sigma_j(H_t)$ is the j-th singular value of H_t in decreasing order.

According to Lemma 1, we choose $U_\mathcal{I}$ to be the left singular vectors associated with the largest $r-q$ singular values of H_t, to minimize $\max_{1\le i\le s, i\notin\mathcal{I}} \sigma_i(H_t)$. The pseudocode of our algorithm is given in Algorithm 2.

When $r = q$, we can easily see that Z_1 in our algorithm disappears and our algorithm becomes the same as LDA/QR.

Algorithm 2 SVD-QR-LDA algorithm

Input: give the target dimension r of the first stage

1: calculate H_t and H_b by (2), $q \leftarrow \mathrm{rank}(H_b)$
2: $Z_1 \leftarrow$ the top $r-q$ left singular vectors of H_t
3: compute QR decomposition with pivoting of $H_b - Z_1 Z_1^T H_b$ as $QR\Pi = H_b - Z_1 Z_1^T H_b$
4: $Z_2 \leftarrow$ first q columns of Q, $Z \leftarrow [Z_1, Z_2]$
5: $\tilde{H}_b \leftarrow Z^T H_b$, $\tilde{H}_t \leftarrow Z^T H_t$
6: $\tilde{G} \leftarrow$ result of the conventional LDA algorithm on $\tilde{H}_b$ and $\tilde{H}_t$
7: **return** $\hat{G} \leftarrow Z\tilde{G}$

Therefore, our algorithm can be regarded as a generalization of LDA/QR, with larger intermediate dimension r. But our algorithm gives a more accurate result than LDA/QR when r is much larger than q.

Note that Z_{PCA} in PCA+LDA does not hold the conditions in Theorem 1. Thus, we cannot compare our algorithm with PCA+LDA theoretically. However, we will see in Section 4 that the objective function of our algorithm is usually larger than the one of PCA+LDA when the intermediate dimension r is the same.

3.3. The Randomized SVD Algorithm

The time complexity of our algorithm is $T_{PCA}(r) + O(nrm)$, where $T_{PCA}(r)$ is the time complexity of calculating the top r singular vectors. If we use a classical algorithm for SVD, the time complexity is $O(np\min\{n,p\})$, which is too slow for large-scale matrices. To make our algorithm scalable, we use a random projection based algorithm (Halko et al., 2011) to calculate SVD. The time complexity is $O(npr)$, which is much smaller than $O(np\min\{n,p\})$ when r is small. Although the Krylov subspace method has the same time complexity $O(npr)$, Halko et al. (2011) showed that random projection based algorithms are more scalable. When the data matrix is not fitted in memory, randomized algorithms require only constant number of passes over the data, while the subspace method requires at least $O(r)$ passes. In our algorithm, we need as large r as possible to make our algorithm more accurate. Thus the number of passes that the Krylov subspace method needs is too large. Randomized algorithms are also more robust and are easily parallelized. Moreover, we will show empirically in Section 4 that the loss in accuracy incurred by using the randomized algorithm is very small.

We describe this randomized approach in Algorithm 3. We refer to Algorithm 2 based on the randomized approach as *RSVD-QR-LDA* for short. The time complexity of our RSVD-QR-LDA is $O(npr)$, which is similar to that of PCA+LDA using the same SVD algorithm. Since the data matrix in question possibly has singular values that decay slowly, we need some power iteration steps. Setting the number of power iterations q_{SVD} to 1 usually suffices in practice.

Algorithm 3 randomized SVD algorithm

Input: give an $p \times n$ matrix A, a target dimension k_{SVD}, an exponent q_{SVD}, and an oversampling parameter p_{SVD}

1: $\Omega \leftarrow$ an $n \times (k_{\mathrm{SVD}} + p_{\mathrm{SVD}})$ Gaussian test matrix
2: $Y \leftarrow (AA^T)^{q_{\mathrm{SVD}}} A\Omega$
3: compute QR decomposition of Y as $Y = QR$
4: $B \leftarrow Q^T A$
5: compute SVD of B as $B = \tilde{U}\Sigma V^T$
6: $U_{k_{\mathrm{SVD}}} \leftarrow$ first k_{SVD} columns of $Q\tilde{U}$, $\Sigma_{k_{\mathrm{SVD}}} \leftarrow$ first k_{SVD} columns and k_{SVD} rows of Σ, $V_{k_{\mathrm{SVD}}} \leftarrow$ first k_{SVD} columns of V
7: **return** $U_{k_{\mathrm{SVD}}}, \Sigma_{k_{\mathrm{SVD}}}, V_{k_{\mathrm{SVD}}}$

3.4. Implementation on Distributed systems

For large problems with millions of training data or features, it may be impractical to apply our algorithm on a single machine. Therefore, we implementat our algorithm on distributed systems. Our implementation requires that the scale of r should be moderate, such that a matrix of size $r \times r$ can fit in the memory of a single machine.

Our algorithm employs the randomized SVD algorithm described in section 3.3 for performing SVD. In section 3.3, we have discussed a lot of the scalability issues of the randomized SVD algorithm and the Krylov subspace method. Furthermore, compared with the randomized SVD algorithm, the Krylov subspace method requires much more iterations. Each iteration depends on the results of the previous iteration to proceed. Thus, it needs data transfer and synchronization work in each iteration. On distributed systems, the cost of data transfer and synchronization is large. Therefore, the Krylov subspace method can be much slower than the randomized method on distributed systems.

There are two operations in our algorithm that need to be implemented on distributed systems: matrix multiplication of large matrices and the QR decomposition of tall-and-skinny matrices. Here "tall-and-skinny matrix" means matrix which has a large row size and a small column size. Large matrix multiplication can be parallelized naturally. To implement the QR decomposition of thin matrix on distributed systems, many methods have been proposed (Cosnard et al., 1986; Halko, 2012). We use the Cholesky decomposition based algorithm for the QR decomposition to make our implementation simple. For a tall-and-skinny matrix Y, we compute the Cholesky decomposition of Y^TY as $Y^TY = R^TR$, where R is an upper triangular matrix. Then the QR decomposition of Y should be $Y = QR$ where $Q = YR^{-1}$.

3.5. Generalization to KDA

Algorithm 2 can be easily generalized to solve the KDA problem, because the objective function of KDA has a similar form as LDA. We present the detailed procedure in Algorithm 4.

Algorithm 4 RSVD-QR-KLDA algorithm

Input: give the target dimension r of the first stage

1: calculate K_t and K_b by (3) and (4), $q \leftarrow \mathrm{rank}(K_b)$
2: $Z_1 \leftarrow$ the top $r-q$ left singular vectors of K_t
3: compute QR decomposition with pivoting of $K_b - Z_1 Z_1^T K_b$ as $QR\Pi = K_b - Z_1 Z_1^T K_b$
4: $Z_2 \leftarrow$ first q columns of Q, $Z \leftarrow [Z_1, Z_2]$
5: $\tilde{K}_b \leftarrow Z^T K_b$, $\tilde{K}_t \leftarrow Z^T K_t$
6: $\tilde{A} \leftarrow$ result of the conventional KDA algorithm on $\tilde{K}_b$ and $\tilde{K}_t$
7: **return** $\hat{A} \leftarrow Z\tilde{A}$

We can also use the randomized SVD algorithm in Step 2 of Algorithm 4. The time complexity of the resulting algorithm is $T_k + O(n^2 r)$, where T_k is the time for calculating the kernel matrix K. Thus, our algorithm and the PCA+LDA algorithm also have the similar computational complexity.

4. Experiments

In this section we perform empirical analysis of our proposed algorithms on two face datasets, YaleB&E and CMU PIE, two middle-sized document datasets, News20 (Lang, 1995) and RCV1 (Lewis et al., 2004), and a large dataset, Amazon7 (Dredze et al., 2008; Blondel et al., 2013). The sizes of the five datasets are described in Table 1.

The YaleB&E dataset consists of Yale Face Database B (Georghiades et al., 2001) and the extended Yale Face Database B (Lee et al., 2005). We collect a subset of 2414 face images of 38 subjects, and crop and resize each image to 32×32. For the CMU PIE dataset, we use a subset containing 11554 images of 68 subjects, and the images are cropped and resized to 64×64. For these two datasets, we randomly pick 70% of data for training and the remaining for test. We repeat this procedure 5 times and report the averages of the objective function, classification accuracy and running time on these 5 repeats.

For News20, we use the first 80% of the original data for training and the left 20% for test. For RCV1, we follow the preprocessing process of Bekkerman & Scholz (2008). That is, we map all categories to the second level and delete all data with multiple labels. We also follow the setting in Bekkerman & Scholz (2008) for the training and test data.

Amazon7 dataset contains 1,362,109 reviews of Amazon product. We randomly pick 80% of the dataset for training, and the remaining is for test. We follow the preprocessing process of Blondel et al. (2013).

Table 1. The summary of datasets where n is the number of training data, p is the number of the input features, and m is the number of classes.

Data set	n	test size	p	m
YaleB&E	1,689	725	1,024	38
PIE	8,087	3,467	4,096	68
News20	15,935	3,993	62,061	20
RCV1	15,564	518,571	47,236	53
Amazon7	1,089,687	272,422	262,144	7

4.1. Experiments with LDA Algorithms

We compare our RSVD-QR-LDA algorithm with the PCA-LDA algorithm. For the two small-size face datasets YaleB&E and CMU PIE, we also compare our algorithm with the conventional LDA solved by Algorithm 1, as well as DSVD-QR-LDA in Algorithm 2 where the SVD (Step 2) is computed by the classical SVD algorithm. After the LDA algorithm, we perform the k-NN classification algorithm on the reduced space. Particularly, k is selected via 10-fold cross-validation. The SVD step in both RSVD-QR-LDA and PCA-LDA is computed by Algorithm 3 with $q_{\mathrm{SVD}} = 1$ and $p_{\mathrm{SVD}} = 0.1 k_{\mathrm{SVD}}$. All the algorithms are implemented in python 2.7 on a PC with an Intel Xeon X5675 3.07 GHz CPU and 12GB memory.

In Figures 1, 2, 3 and 4, we report the average of the objective function of LDA, classification accuracy, and running time of the LDA algorithm, with respect to different r. For RSVD-QR-LDA and PCA-LDA, we run the algorithm 10 times for each training-test splitting, and report the standard deviation of both value of objective function and classification accuracy. The standard deviation is taken with respect to different running on the same splitting. For YaleB&E and CMU PIE, we let r from $m-1$ to about 1000, while for News20 and RCV1, we let r from $m-1$ to about 2000. When $r = m-1$, our algorithm is identical to LDA/QR. Thus, we also compare our algorithm with LDA/QR. We summarize the experimental results in Table 2, in which we report the best accuracy, the value of r for which the best accuracy is achieved, and the running time. Here the best r value is hand-picked best value on test data, and the standard deviation in Table 2 is taken with respect to different train/test splitting.

From these results, we can see that for both PCA-LDA and SVD-QR-LDA, the value of the objective function increases as r increases. For same r, the objective function of SVD-QR-LDA is larger than that of PCA-LDA. With regard to the classification accuracy, we find that in News20 and RCV1, SVD-QR-LDA outperforms PCA-LDA, and in YaleB&E and CMU PIE, both our algorithm and the

Table 2. The best mean classification accuracy (acc) and corresponding standard deviation (std), the best r value for which the best accuracy is achieved, and the running time of the algorithms (in second). Here "N/A" means that LDA is not applicable to this dataset.

Dataset	RSVD-QR-LDA			PCA-LDA			LDA	
	acc ($\pm$ std)	best r	time ($\pm$ std)	acc ($\pm$ std)	best r	time ($\pm$ std)	acc ($\pm$ std)	time ($\pm$ std)
YaleB&E	**95.5**($\pm$ 0.6)	429	0.85($\pm$ 0.03)	**95.4**($\pm$ 0.6)	723	1.38($\pm$ 0.04)	93.7($\pm$ 1.1)	0.74($\pm$ 0.03)
CMU PIE	**96.6**($\pm$ 0.4)	167	6.3($\pm$ 0.3)	96.3($\pm$ 0.4)	267	4.3($\pm$ 0.7)	93.0($\pm$ 0.4)	69.3($\pm$ 18.2)
News20	**85.8**	2052	184	83.1	2052	180	N/A	N/A
RCV1	**84.0**	302	14	82.3	2052	135	N/A	N/A

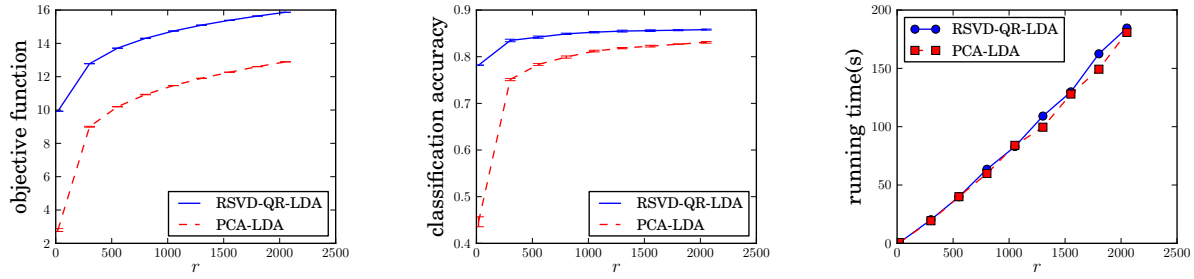

Figure 1. Objective function, classification accuracy and running time of two algorithms on the News20 dataset with respect to intermediate dimension r. We also draw the standard deviations of objective function and classification accuracy in the figure. The standard deviations are taken with respect to different running on the same training-test splitting.

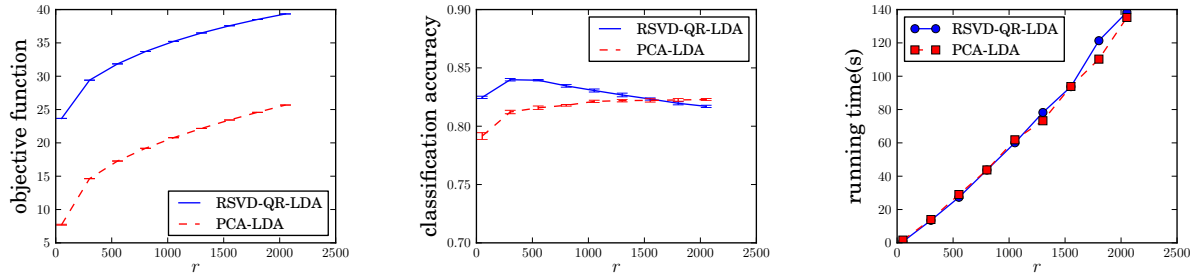

Figure 2. Objective function, classification accuracy and running time of two algorithms on the RCV1 dataset.

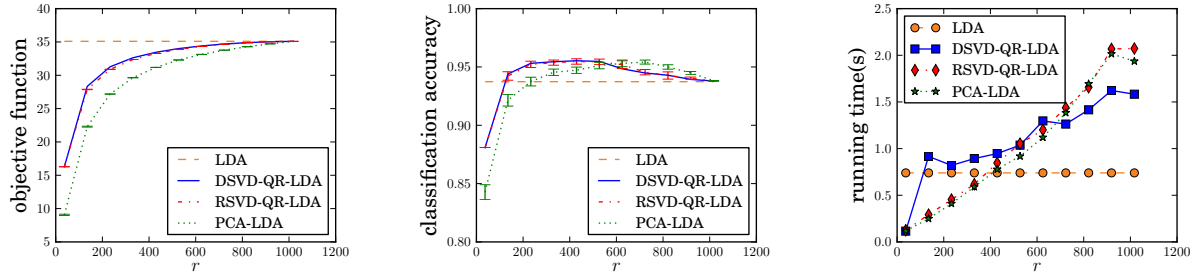

Figure 3. Objective function, classification accuracy and running time of four algorithms on the YaleB&E dataset.

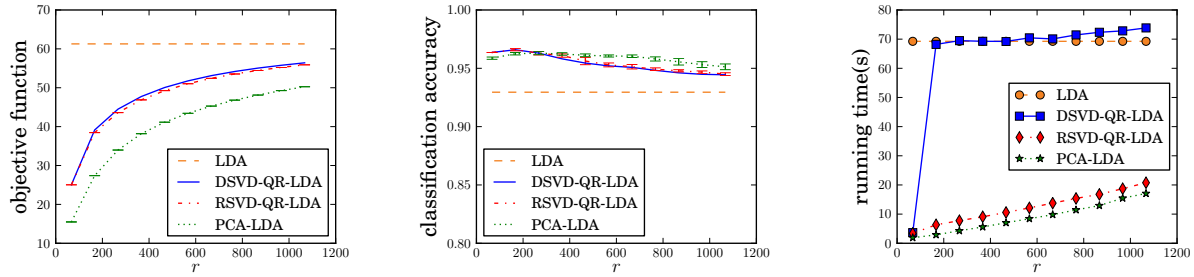

Figure 4. Objective function, classification accuracy and running time of four algorithms on the CMU PIE dataset.

PCA-LDA algorithm suffer from the problem of overfitting. However, our algorithm achieves the best accuracy when taking a smaller r, which makes our algorithm more efficient. Additionally, in YaleB&E and CMU PIE, both our algorithm and the PCA-LDA algorithm outperform the exact LDA algorithm. The running time of RSVD-QR-LDA is nearly the same with PCA-LDA on the same r, and both are much faster than the conventional LDA, especially on large scale datasets.

We can also see that RSVD-QR-LDA performs nearly the same with DSVD-QR-LDA and that the standard deviations are also small. This implies that the accuracy loss incurred by using the randomized algorithm is very small.

4.2. Experiments on the distributed system

We test our implementation on the distributed system on a Hadoop (Borthakur, 2007) cluster on Amazon Elastic MapReduce (EMR). We use an EMR cluster consisting of 8 m1.xlarge instance. Each m1.xlarge instance uses a Intel Xeon Family quad-core CPU and 15GB memory. The version of Hadoop we use is 1.0.3.

In Figure 5, we report the objective function of LDA, classification accuracy, and running time of the LDA algorithm, with respect to different r. In Table 3, we report the best accuracy, the value of r for which the best accuracy is achieved, and the running time.

From these results, we can see that our algorithm outperforms PCA-LDA in both the value of objective function and classification accuracy, and the running times of the two algorithms are nearly the same.

4.3. Experiments with KDA Algorithms

For KDA algorithms, we only conduct comparison on two face datasets: YaleB&E and CMU PIE. We use RBF kernel $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\frac{\|\mathbf{x}_1-\mathbf{x}_2\|^2}{\theta})$ in our experiment, where θ was set to the mean Euclidean distance among training data points. We report the objective function, classification accuracy and running time of the three algorithms in Figures 3 and 4. Here the running time does not include the time for computing the kernel matrix. We also summarize our results in Table 4, including the best accuracy, the value of r on which the best accuracy is achieved, and the running time.

It is seen from the results that our algorithm outperforms PCA+KDA in both the optimal value of objective function and classification accuracy. The running times of the two algorithms are nearly the same. But they are much faster than the conventional KDA on larger scale datasets.

5. Conclusion

In this paper we have proposed a novel two-stage algorithm for approximately solving the LDA problem, based on our theoretical analysis on the approximation error of two-stage methods. Our algorithm includes LDA/QR as a special case. We have made our algorithm scalable by using a randomized SVD algorithm and generalized our algorithm to the kernel LDA problem. Furthermore, we devise an implementation of our algorithm on distributed systems. We have conducted empirical analysis on several different datasets. The experimental results show that our algorithm outperforms PCA+LDA while they have the same scalability.

References

Baudat, G. and Anouar, F. Generalized discriminant analysis using a kernel approach. *Neural Computation*, 12 (10):2385–2404, 2000.

Bekkerman, R. and Scholz, M. Data weaving: scaling up the state-of-the-art in data clustering. In *Proceedings of the 17th ACM conference on Information and knowledge management (CIKM '08)*, pp. 1083–1092, New York, NY, USA, 2008. ACM.

Belhumeur, P. N., Hespanha, J., and Kriegman, D. J. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

Table 3. The best mean classification accuracy (acc) and the standard deviation (std), the best r value and the running time of algorithm (in minutes).

Dataset	RSVD-QR-LDA			PCA-LDA		
	acc	best r	time	acc	best r	time
Amazon7	**92.2**	500	82.4	90.6	500	73.3

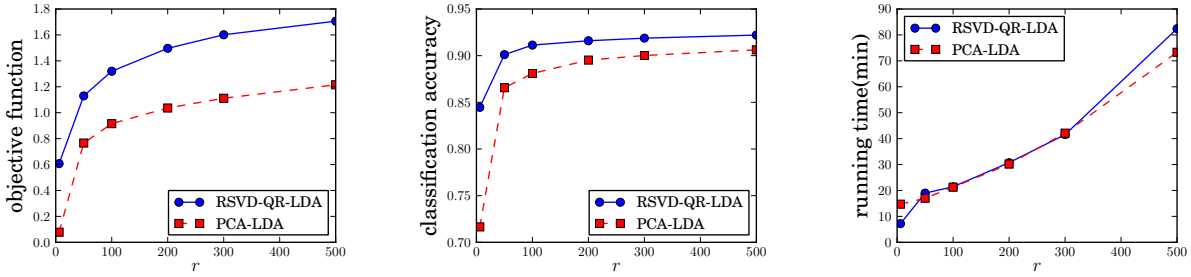

Figure 5. Objective function, classification accuracy and running time of the three algorithms on the Amazon7 dataset.

Table 4. The best mean classification accuracy (acc) and the standard deviation (std), the best r value and the running time of algorithm (in second).

Dataset	RSVD-QR-KDA			PCA-KDA			KDA	
	acc ($\pm$ std)	best r	time ($\pm$ std)	acc ($\pm$ std)	best r	time ($\pm$ std)	acc ($\pm$ std)	time ($\pm$ std)
YaleB&E	**95.7**($\pm$ 0.6)	821	2.33($\pm$ 0.06)	**95.7**($\pm$ 0.6)	1017	2.75($\pm$ 0.09)	**95.7**($\pm$ 0.6)	1.89($\pm$ 0.30)
CMU PIE	98.2($\pm$ 0.3)	1067	28.9($\pm$ 0.2)	98.1($\pm$ 0.1)	1067	29.3($\pm$ 0.3)	**98.3**($\pm$ 0.3)	162.5($\pm$ 0.6)

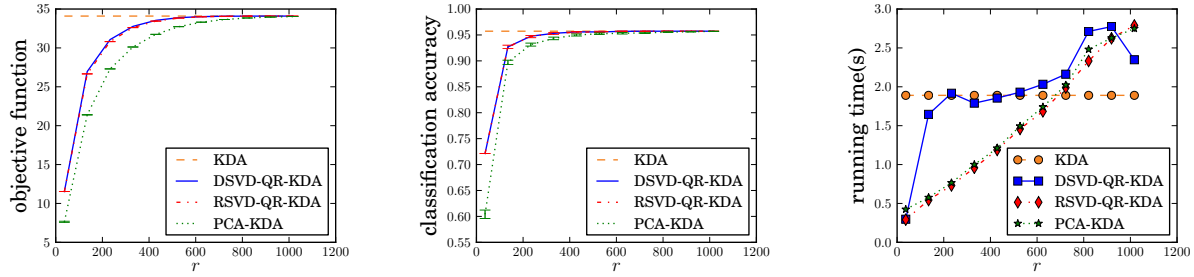

Figure 6. Objective function, classification accuracy and running time of the three algorithms on the YaleB&E dataset.

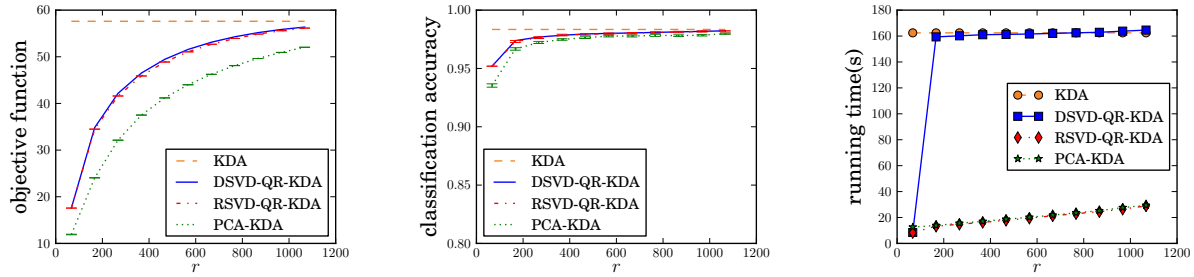

Figure 7. Objective function, classification accuracy and running time of the three algorithms on the CMU PIE dataset.

Blondel, Mathieu, Seki, Kazuhiro, and Uehara, Kuniaki. Block coordinate descent algorithms for large-scale sparse multiclass classification. *Machine Learning*, pp. 1–22, 2013.

Borthakur, Dhruba. The hadoop distributed file system: Architecture and design, 2007.

Cosnard, Michel, Muller, J-M, and Robert, Yves. Parallel qr decomposition of a rectangular matrix. *Numerische Mathematik*, 48(2):239–249, 1986.

Dredze, Mark, Crammer, Koby, and Pereira, Fernando. Confidence-weighted linear classification. In *Proceedings of the 25th international conference on Machine learning*, pp. 264–271. ACM, 2008.

Fukunaga, K. *Introduction to statistical pattern recognition*. Academic Pr, 1990.

Georghiades, A. S., Belhumeur, P. N., and Kriegman, D. J. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:643–660, 2001.

Golub, G. H. and Loan, C. F. Van. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.

Halko, N., Martinsson, P. G., and Tropp, J. A. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.

Halko, Nathan P. *Randomized methods for computing low-rank approximations of matrices*. PhD thesis, University of Colorado, 2012.

Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning, Second Edition: Data Mining, Inference, and Prediction*. Springer, 2009.

Lang, K. Newsweeder: Learning to filter netnews. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pp. 331–339. Morgan Kaufmann, 1995.

Lee, K.-C., Ho, J., and Kriegman, D. J. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5):684–698, 2005.

Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

Mahoney, M. W. Randomized algorithms for matrices and data. *Foundations and Trends in Machine Learning*, 3 (2):123–224, 2011.

Martínez, A. M. and Kak, A. C. Pca versus lda. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2):228–233, 2001.

Mika, S., Ratsch, G., Weston, J., Schölkopf, B., and Müller, K.-R. Fisher discriminant analysis with kernels. *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop*, pp. 41–48, 1999.

Park, C. H. and Park, H. Nonlinear discriminant analysis using kernel functions and the generalized singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 27(1):87–102, 2005.

Park, H., Jeon, L. M., and Rosen, Z. J. B. Lower dimensional representation of text data based on centroids and least squares. *BIT Numerical Mathematics*, 43:427–448, 2003.

Yang, J. and Yang, J.-Y. Why can LDA be performed in PCA transformed space? *Pattern Recognition*, 36(2): 563–566, 2003.

Ye, J. Characterization of a family of algorithms for generalized discriminant analysis on undersampled problems. *Journal of Machine Learning Research*, 6:483–502, 2005.

Ye, J. and Li, Q. A two-stage linear discriminant analysis via QR-decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):929–941, 2005.

Zhang, Z., Dai, G., Xu, C., and Jordan, M. I. Regularized discriminant analysis, ridge regression and beyond. *Journal of Machine Learning Research*, 11:2199–2228, 2010.

A. Proof of Theorem 1

The following Lemma is a famous inequality for Hermitian matrices (See (Marshall & Olkin, 1979; Lasserre, 1995)).

Lemma 2 *For any two Hermetian matrix $A, B \in \mathbb{R}^{n\times n}$,*

$$\mathrm{trace}(AB) \le \sum_{i=1}^{n} \lambda_i(A)\lambda_i(B),$$

where $\lambda_i(A)$ is the i-th largest eigenvalue of A.

First of all, since $\mathcal{R}(H_b) \subset \mathcal{R}(Z)$, we have $H_b = ZZ^T H_b$, thus $S_b^{G^*} = (G^*)^T H_b H_b^T G^* = (G^*)^T ZZ^T H_b H_b^T ZZ^T G^* = S_b^{ZZ^T G^*}$. Then

$$\begin{aligned}
&J(G^*) - J(\hat{G}^Z) \\
&\le J(G^*) - J(ZZ^T G^*) \\
&= \mathrm{trace}\left(((S_t^{G^*})^{-1} - (S_t^{ZZ^T G^*})^{-1}) S_b^{G^*}\right) \\
&= \mathrm{trace}\left((I - (S_t^{ZZ^T G^*})^{-1})\Sigma_q^2\right) \\
&\overset{(a)}{\le} \sum_{i=1}^{q}\left(1 - \frac{1}{\lambda_i(S_t^{ZZ^T G^*})}\right)\alpha_i^2 \\
&\le \left(1 - \frac{1}{\lambda_1(S_t^{ZZ^T G^*})}\right)\sum_{i=1}^{q}\alpha_i^2 \\
&= \left(1 - \frac{1}{\|(G^*)^T ZZ^T H_t\|_2^2}\right) J(G^*),
\end{aligned}$$

where α_i is the i-th diagonal entry of Σ_q, and (a) follows from Lemma 2. Then

$$J(\hat{G}^Z) \ge \frac{1}{\|(G^*)^T ZZ^T H_t\|_2^2} J(G^*).$$

As for the second inequality, it then follows from

$$\begin{aligned}
\|(G^*)^T ZZ^T H_t\|_2 &= \|P_q^T \Sigma_t^{-1} U_t ZZ^T H_t\|_2 \\
&\le \|P_q^T\|_2 \|\Sigma_t^{-1} U_t^T ZZ^T H_t\|_2 \\
&= \|V_t \Sigma_t^{-1} U_t^T ZZ^T H_t\|_2 \\
&= \|H_t^\dagger ZZ^T H_t\|_2.
\end{aligned}$$

B. Proof of Lemma 1

Without loss of generality, let $Z = \begin{bmatrix} U_{\mathcal{I}} & Z_2 \end{bmatrix}$,

$$\Sigma_t = \begin{bmatrix} \Sigma_{\mathcal{I}} & 0 \\ 0 & \Sigma_{\mathcal{I}^c} \end{bmatrix} \quad \text{and} \quad U_t = \begin{bmatrix} U_{\mathcal{I}} & U_{\mathcal{I}^c} \end{bmatrix},$$

where $\mathcal{I}^c$ is the complement set of $\mathcal{I}$ in $\{1, \ldots, s\}$, $\Sigma_{\mathcal{I}} \in \mathbb{R}^{(r-q)\times(r-q)}$, $\Sigma_{\mathcal{I}^c} \in \mathbb{R}^{(s-r+q)\times(s-r+q)}$, $U_{\mathcal{I}} \in \mathbb{R}^{p\times(r-q)}$, $U_{\mathcal{I}^c} = \mathbb{R}^{p\times(s-r+q)}$ and $Z_2 = \mathbb{R}^{p\times q}$. Then

$$U_t^T ZZ^T U_t = \begin{bmatrix} I & 0 \\ 0 & U_{\mathcal{I}^c}^T Z_2 Z_2^T U_{\mathcal{I}^c} \end{bmatrix}.$$

Therefore,

$$\begin{aligned}
&\|H_t^\dagger ZZ^T H_t\|_2 \\
&= \|\Sigma_t^{-1} U_t^T ZZ^T U_t \Sigma_t\|_2 \\
&= \left\| \begin{bmatrix} \Sigma_{\mathcal{I}}^{-1} & 0 \\ 0 & \Sigma_{\mathcal{I}^c}^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & U_{\mathcal{I}^c}^T Z_2 Z_2^T U_{\mathcal{I}^c} \end{bmatrix} \begin{bmatrix} \Sigma_{\mathcal{I}} & 0 \\ 0 & \Sigma_{\mathcal{I}^c} \end{bmatrix} \right\|_2 \\
&= \left\| \begin{bmatrix} I & 0 \\ 0 & \Sigma_{\mathcal{I}^c}^{-1} U_{\mathcal{I}^c}^T Z_2 Z_2^T U_{\mathcal{I}^c} \Sigma_{\mathcal{I}^c} \end{bmatrix} \right\|_2 \\
&= \max\{1, \|\Sigma_{\mathcal{I}^c}^{-1} U_{\mathcal{I}^c}^T Z_2 Z_2^T U_{\mathcal{I}^c} \Sigma_{\mathcal{I}^c}\|_2\} \\
&\le \max\{1, \|\Sigma_{\mathcal{I}^c}^{-1}\|_2 \|U_{\mathcal{I}^c}^T Z_2 Z_2^T U_{\mathcal{I}^c}\|_2 \|\Sigma_{\mathcal{I}^c}\|_2\} \\
&= \max\{1, \|\Sigma_{\mathcal{I}^c}^{-1}\|_2 \|\Sigma_{\mathcal{I}^c}\|_2\} \\
&= \frac{\max_{1\le i\le s, i\notin\mathcal{I}} \sigma_i(H_t)}{\sigma_s(H_t)}.
\end{aligned}$$

References

Lasserre, Jean B. A trace inequality for matrix product. *Automatic Control, IEEE Transactions on*, 40(8):1500–1501, 1995.

Marshall, A. W. and Olkin, I. *Inequalities: Theory of Majorization and its Applications*. Academic Press, 1979.