Generalized Low Rank Approximations of Matrices

Jieping Ye JIEPING@CS.UMN.EDU
Department of Computer Science, University of Minnesota, Minneapolis, MN 55455, USA

Abstract

We consider the problem of computing low rank approximations of matrices. The novelty of our approach is that the low rank approximations are on a sequence of matrices. Unlike the problem of low rank approximations of a single matrix, which was well studied in the past, the proposed algorithm in this paper does not admit a closed form solution in general. We did extensive experiments on face image data to evaluate the effectiveness of the proposed algorithm and compare the computed low rank approximations with those obtained from traditional Singular Value Decomposition based method.

keywords: Singular Value Decomposition (SVD), low rank approximation, classification.

1. Introduction

The language of linear algebra appeared quite early in information retrieval, and machine learning, through the use of *vector space model* (Kleinberg & Tomkins, 1999). Under this model, each datum is modeled as a vector and the collection of data is modeled as a single data matrix, where each column of the data matrix corresponds to a data point and each row of the data matrix corresponds to a feature dimension. The representation of data by vectors in Euclidean space allows one to compute the similarity between data points, based on the Euclidean distance or some other similarity metrics. The similarity metrics on data points naturally lead to similarity based indexing by representing queries as vectors and searching for their nearest neighbors. However, in many applications, such as images, the nearest neighbor searching is in a huge number of dimensions. Because of the *curse of the dimensionality*, that is, the query efficiency and accuracy degrades as the dimensionality increases, dimension reduction becomes an important pre-processing step (Aggarwal, 2001; Castelli et al., 2003).

A well-known technique for dimension reduction is the low rank approximation by the Singular Value Decomposition (SVD) (Golub & Van Loan, 1996). Let A be the data matrix, and let $A = U\Sigma V^T$ be the SVD of A, where U and V are orthogonal and Σ is diagonal. Then, for a given k, the optimal rank k approximation of A is given by $\text{best}_k(A) = U_k\Sigma_k V_k^T$, where U_k and V_k are the matrices formed by the first k columns of the matrices U and V, respectively, and Σ_k is the k-th principal submatrix of Σ. A key property of this rank k approximation is that it achieves the best possible approximation with respect to the Frobenius norm, among all matrices with rank k. More details can be found in Section 2.

The problem of low rank approximations of matrices has recently received broad attention in areas such as computer vision, information retrieval, and machine learning (Berry et al., 1995; Castelli et al., 2003; Deerwester et al., 1990; Dhillon & Modha, 2001; Srebro & Jaakkola, 2003). It becomes an important tool for extracting correlations and removing noise from data. However, applications of this technique to high-dimensional data, such as images and videos, quickly run up against practical computational limits, mainly due to the high time and space complexities of the SVD computation for large matrices. Several incremental algorithms have been proposed in the past (Gu & Eisenstat, 1993; Kanth et al., 1998) to deal with the high space complexity of SVD, where the data points are inserted incrementally to update the SVD. To our knowledge, such algorithms come with no guarantees on the quality of the approximation produced. Low rank approximation by random sampling was studied in (Achlioptas & McSherry, 2001; Drineas et al., 1999; Frieze et al., 1998) to speed up the SVD computation. However, the effectiveness of these approaches is dependent on the spectral structure of data matrix.

In this paper, we present a novel approach to alleviate the expensive SVD computation. The novelty lies in a new data representation model. Under this new model, each datum is represented as a matrix, instead of a vector, and the collection of data as a sequence of

This paper is a slightly revised version of the paper appearing in *Proceedings of the 21st International Conference on Machine Learning*, 2004. Copyright 2004 by the author.

matrices, instead of a single large matrix. The problem of low rank approximations becomes the problem of approximating a sequence of matrices with matrices of lower rank. We formulate this as a new optimization problem. Details will be given in Section 3. Unlike the traditional problem of approximating a single matrix, there is no closed form solution for the new optimization problem. We thus derive an iterative algorithm. Detailed mathematical justification for this iterative procedure is provided.

A natural application for this new approach is in image compression and retrieval, where each image is represented in its native matrix representation. To evaluate the proposed algorithm, we did extensive experiments on four well-known face image datasets: PIX, ORL, AR and PIE and compared the proposed algorithm with traditional SVD based method. Empirical results show that: (1) The proposed algorithm outperforms the traditional SVD based method in terms of classification accuracy, when using the same compression ratio[1]; (2) The proposed algorithm has distinctively less computational time than the traditional SVD based method.

The rest of this paper is organized as follows. We give a brief overview on low rank approximations in Section 2. The problem of generalized low rank approximations of matrices is studied in Section 3. A performance study is presented in Section 4. The conclusion is in Section 5.

The notations used throughout the rest of this paper are listed in the Table 1.

Table 1. Notations

Notations	Descriptions
A_i	the i-th data point in matrix form
r	number of rows in A_i
c	number of columns in A_i
L	transformation on the left side
R	transformation on the right side
D_i	reduced representation of A_i
ℓ_1	number of rows in D_i
ℓ_2	number of columns in D_i
k	rank of reduced matrix by SVD
A	data matrix
n	number of training data points
N	dimension of training data

[1]Here the compression ratio means the percentage of space saved by the low rank approximations to store the data. Details can be found in Sections 2 and 3.

2. Low Rank Approximations

Traditional methods in information retrieval and machine learning deal with data in vectorized representation. A collection of data is then stored in a single matrix $A \in R^{N\times n}$, where each column of A corresponds to a vector in the N-dimensional space. A major benefit of this vector space model is that the algebraic structure of the vector space can be exploited (Berry et al., 1995).

For high-dimensional data, one would like to simplify the data, so that traditional machine learning and statistical techniques can be applied. However, crucial information intrinsic in the data should not be removed under this simplification. A widely used method for this purpose is to approximate the single data matrix, A, with a matrix of lower rank. Mathematically, the optimal rank k approximation of a matrix A, under the Frobenius norm can be formulated as follows:

> Find a matrix $B \in R^{N\times n}$ with $\text{rank}(B) = k$, such that $B = \arg\min_{\text{rank}(B)=k} ||A - B||_F$,

where the Frobenius norm, $||M||_F$, of a matrix $M = (M_{ij})$ is given by $||M||_F = \sqrt{\sum_{i,j} M_{ij}^2}$. The matrix B can be readily obtained by computing the Singular Value Decomposition (SVD) of A, as stated in the following theorem (Golub & Van Loan, 1996).

Theorem 2.1. *Let the Singular Value Decomposition of $A \in R^{N\times n}$ be $A = UDV^T$, where U and V are orthogonal, $D = diag(\sigma_1, \cdots, \sigma_r, 0, \cdots, 0)$, $\sigma_1 \geq \cdots \geq \sigma_r > 0$ and $r = rank(A)$. Then for $1 \leq k \leq r$, $\sum_{i=k+1}^{r} \sigma_i^2 = \min\{||A - B||_F^2 \mid rank(B) = k\}$. The minimum is achieved with $B = best_k(A)$, where $best_k(A) = U_k diag(\sigma_1, \cdots, \sigma_k)V_k^T$, and U_k and V_k are the matrices formed by the first k columns of U and V respectively.*

For any approximation, M, of A, we call $||A-M||_F$ the reconstruction error of the approximation. By Theorem 2.1, $B = U_k\text{diag}(\sigma_1, \cdots, \sigma_k)V_k^T$ has the smallest reconstruction error among all the rank k approximations of A. Under this approximation, each column, $a_i \in R^N$, of A can be approximated as $a_i \approx U_k a_i^L$, for some $a_i^L \in R^k$. Since U_k has orthonormal columns, $||U_k a_i^L - U_k a_j^L|| = ||a_i^L - a_j^L||$, i.e., the Euclidean distance between two vectors are preserved under the projection by U_k. It follows that $||a_i - a_j|| \approx ||U_k a_i^L - U_k a_j^L|| = ||a_i^L - a_j^L||$. Hence the proximity of a_i and a_j, in the original high-dimensional space, can be approximated by computing the proximity of their reduced representations a_i^L and a_j^L. This forms the basics for Latent Semantics Indexing (Berry et al., 1995; Deerwester et al., 1990), widely used in informational retrieval. Another potential application of the above rank k approximation is data compression. Since each a_i is approximated by $U_k a_i^L$, where U_k is common for every a_i, we need to keep U_k and $\{a_i^L\}_{i=1}^n$ only for all the approximations. Since $U_k \in R^{N\times k}$ and $a_i^L \in R^k$, for $i = 1, \cdots, n$, it requires $nk + Nk = (n+N)k$ scalars only to store the reduced representations. The storage saved, or *compression ratio*, using the rank k approximation is thus $\frac{nN}{(n+N)k}$, since the original data matrix $A \in R^{N\times n}$.

3. Generalized Low Rank Approximations

3.1. Problem formulation

In this section, we study the problem of generalized low rank approximations, which aims to approximate a sequence of matrices with lower rank. A key difference between this generalized problem and the low rank approximation problem, discussed in the last section, is the data representation model applied. Recall that the vector space model is applied for the traditional low rank approximations. The vector space model leads to a simple and closed form solution for low rank approximations by computing the SVD of the data matrix. However, the high time and space complexities of SVD restricts its applicability to matrices with large size. Instead, we apply a different data representation model, under which, each datum is represented by a matrix and the collection of data is represented by a sequence of matrices. The corresponding generalized low rank approximation problem becomes the problem of approximating a sequence of matrices with lower rank. Details are given below.

Let $A_i \in R^{r\times c}$, for $i = 1, \cdots, n$, be the n data points in the training set. We aim to compute two matrices $L \in R^{r\times \ell_1}$ and $R \in R^{c\times \ell_2}$ with orthonormal columns, and n matrices $D_i \in R^{\ell_1\times \ell_2}$, such that LD_iR^T is a good approximation of A_i, for all i. Mathematically, we can formulate this as the following minimization problem: Computing optimal L, R and $\{D_i\}_{i=1}^n$, which solve

$$\min_{\substack{L \in R^{r\times \ell_1}:\ L^TL = I_{\ell_1} \\ R \in R^{c\times \ell_2}:\ R^TR = I_{\ell_2} \\ D_i \in R^{\ell_1\times \ell_2}:\ i=1,\cdots,n}} \sum_{i=1}^n ||A_i - LD_iR^T||_F^2. \quad (1)$$

We can consider L and R in the above approximations as the two-sided linear transformations on the data in matrix form, with L and R as the transformations on the left and right sides, respectively. Recall that in the case of traditional low rank approximations, one-sided transformation is applied, which is U_k in our previous discussions. Note that the matrices $\{D_i\}_{i=1}^n$ are not required to be diagonal, which contrasts with the traditional low rank approximations by SVD.

The common transformations L and R with orthonormal columns, in the above approximations, naturally lead to several basic applications:

- **Data compression:** The matrices L, R, and $\{D_i\}_{i=1}^n$ can be used to recover the original n matrices $\{A_i\}_{i=1}^n$, assuming LD_iR^T is a good approximation of A_i, for each i. It requires $r\ell_1 + c\ell_2 + n\ell_1\ell_2$ scalars to store $L \in R^{r\times\ell_1}$, $R \in R^{c\times \ell_2}$, and $\{D_i\}_{i=1}^n \in R^{\ell_1\times\ell_2}$. Hence, the storage saved, or the *compression ratio* using the approximations is $\frac{nrc}{r\ell_1+c\ell_2+n\ell_1\ell_2}$, since $A_i \in R^{r\times c}$, for each i.

- **Similarity computation:** A common similarity metric between A_i and A_j is the Frobenius norm. Under this metric, the distance between A_i and A_j is $||A_i - A_j||_F$, whose computational complexity is $O(rc)$. Using the approximations, $||A_i - A_j||_F \approx ||LD_iR^T - LD_jR^T||_F = ||D_i - D_j||_F$, since both L and R have orthonormal columns. It is clear that the computational cost for computing $||D_i - D_j||_F$ is $O(\ell_1\ell_2)$. Hence, the speed-up on a single distance computation using the reduced representations, is $\frac{rc}{\ell_1\ell_2}$.

Note that as ℓ_1 and ℓ_2 decrease, the speed-up on the distance computation and the compression ratio increase. However, small values of ℓ_1 and ℓ_2 may lead to loss of information intrinsic in the original data. We discuss this trade-off in Section 4.

The formulation in Eq. (1) is general, in the sense that ℓ_1 and ℓ_2 can be different, which might be useful in the case when the number, r, of rows of A_i is much larger or smaller than the number, c, of columns of A_i.

3.2. The main algorithm

In the rest of this section, we show how to solve the minimization problem in Eq. (1). The result in the following theorem shows that the D_i's are dependent on the transformation matrices L and R, which significantly simplifies the minimization in Eq. (1).

Theorem 3.1. *Let L, R and $\{D_i\}_{i=1}^n$ be the optimal solution to the minimization problem in Eq. (1). Then $D_i = L^TA_iR$, for every i.*

Proof. By the property of the trace of matrices,

$$\begin{aligned}
& \textstyle\sum_{i=1}^n ||A_i - LD_iR^T||_F^2 \\
= & \textstyle\sum_{i=1}^n \mathrm{trace}\left((A_i - LD_iR^T)(A_i - LD_iR^T)^T\right) \\
= & \textstyle\sum_{i=1}^n \mathrm{trace}(A_iA_i^T) + \displaystyle\sum_{i=1}^n \mathrm{trace}(D_iD_i^T) \\
-2 & \textstyle\sum_{i=1}^n \mathrm{trace}(LD_iR^TA_i^T), \qquad (2)
\end{aligned}$$

where the second term $\sum_{i=1}^n \mathrm{trace}(D_iD_i^T)$ results from the fact that both L and R have orthonormal columns, and $\mathrm{trace}(AB) = \mathrm{trace}(BA)$, for any two matrices.

Since the first term in the right side of Eq. (2) is a constant, the minimization in Eq. (1) is equivalent to minimizing

$$\sum_{i=1}^n \mathrm{trace}(D_iD_i^T) - 2\sum_{i=1}^n \mathrm{trace}(LD_iR^TA_i^T). \qquad (3)$$

It is easy to check that the minimum of (3) is obtained, only if $D_i = L^TA_iR$, for every i. This completes the proof of the theorem. □

Theorem 3.1 implies that D_i is uniquely determined by L and R with $D_i = L^TA_iR$, for all i. Hence the key step for the minimization in Eq. (1) is the computation of the common transformations L and R. A key property on the optimal transformations L and R is stated in the following theorem:

Theorem 3.2. *Let L, R and $\{D_i\}_{i=1}^n$ be the optimal solution to the minimization problem in Eq. (1). Then L and R solve the following optimization problem:*

$$\max_{\substack{L \in R^{r\times\ell_1} : \quad L^TL = I_{\ell_1} \\ R \in R^{c\times\ell_2} : \quad R^TR = I_{\ell_2}}} \sum_{i=1}^n ||L^TA_iR||_F^2. \qquad (4)$$

Proof. From Theorem 3.1, $D_i = L^TA_iR$, for every i. Plugging $D_i = L^TA_iR$ into $\sum_{i=1}^n ||A_i - LD_iR^T||_F^2$, we obtain

$$\sum_{i=1}^n ||A_i - LD_iR^T||_F^2 = \sum_{i=1}^n ||A_i||_F^2 - \sum_{i=1}^n ||L^TA_iR||_F^2. \qquad (5)$$

Hence the minimization in Eq. (1) is equivalent to the maximization of $\sum_{i=1}^n ||L^TA_iR||_F^2$, which completes the proof of the theorem. □

To our knowledge, there is no closed form solution for the maximization in Eq. (4). A key observation, which leads to an iterative algorithm for the computation of L and R, which are locally optimal, is stated in the following theorem:

Theorem 3.3. *Let L, R and $\{D_i\}_{i=1}^n$ be the optimal solution to the minimization problem in Eq. (1). Then*

1. *For a given R, L consists of the ℓ_1 eigenvectors of the matrix $M_L = \sum_{i=1}^n A_iRR^TA_i^T$ corresponding to the largest ℓ_1 eigenvalues.*

2. *For a given L, R consists of the ℓ_2 eigenvectors of the matrix $M_R = \sum_{i=1}^n A_i^TLL^TA_i$ corresponding to the largest ℓ_2 eigenvalues.*

Proof. By Theorem 3.2, L and R maximize

$$\sum_{i=1}^n ||L^TA_iR||_F^2,$$

which can be rewritten as

$$\begin{aligned}
& \textstyle\sum_{i=1}^n \mathrm{trace}(L^TA_iRR^TA_i^TL) \\
= & \ \mathrm{trace}\left(L^T\sum_{i=1}^n (A_iRR^TA_i^T)L\right) \\
= & \ \mathrm{trace}\left(L^TM_LL\right). \qquad (6)
\end{aligned}$$

Hence, for a given R, the maximum of $\sum_{i=1}^n ||L^TA_iR||_F^2 = \mathrm{trace}\left(L^TM_LL\right)$ is obtained, only if $L \in R^{r\times\ell_1}$ consists of the ℓ_1 eigenvectors of the matrix M_L corresponding to the largest ℓ_1 eigenvalues. The maximization can be considered as a special case of the more general optimization problem in (Edelman et al., 1998).

Similarly, for a given L, the maximum of $\sum_{i=1}^n ||L^TA_iR||_F^2 = \mathrm{trace}\left(R^TM_RR\right)$ is obtained, only if $R \in R^{c\times\ell_2}$ consists of the ℓ_2 eigenvectors of the matrix M_R corresponding to the largest ℓ_2 eigenvalues, where $M_R = \sum_{i=1}^n A_i^TLL^TA_i$.

□

Theorem 3.3 provides us an iterative procedure for computing L and R. More specifically, for a given L, we can compute R by computing the eigenvectors of the matrix M_R. With the computed R, we can then update L by computing the eigenvectors of the matrix M_L. The procedure can be repeated until convergence. The pseudo-code for computing L and U following the above iterative procedure is given in **Algorithm 1**.

Theoretically, the solution from **Algorithm 1** is only locally optimal. The solution depends on the choice of the initial L_0 for L. We did extensive experiments and found that choosing $L_0 = \begin{pmatrix} I_{\ell_1} \\ 0 \end{pmatrix}$, where I_{ℓ_1} is an identity matrix, produces excellent results. We thus use this initial L_0 in all the experiments.

```
Algorithm 1: Generalized Low Rank Approximations
Input:  matrices $\{A_i\}_{i=1}^n$
Output: matrices $L$, $R$, and $\{D_i\}_{i=1}^n$
1. Obtain initial $L_0$ and set $i \leftarrow 1$
2. While not convergent
3.     form the matrix $M_R = \sum_{j=1}^n A_j^T L_{i-1} L_{i-1}^T A_j$
4.     compute the $\ell_2$ eigenvectors $\{\phi_j^R\}_{j=1}^{\ell_2}$
       of $M_R$ corresponding to the largest
       $\ell_2$ eigenvalues
5.     $R_i \leftarrow [\phi_1^R, \cdots, \phi_{\ell_2}^R]$
6.     form the matrix $M_L = \sum_{j=1}^n A_j R_i R_i^T A_j^T$
7.     compute the $\ell_1$ eigenvectors $\{\phi_j^L\}_{j=1}^{\ell_1}$
       of $M_L$ corresponding to the largest
       $\ell_1$ eigenvalues
8.     $L_i \leftarrow [\phi_1^L, \cdots, \phi_{\ell_1}^L]$
9.     $i \leftarrow i + 1$
10. EndWhile
11. $L \leftarrow L_{i-1}$
12. $R \leftarrow R_{i-1}$
13. For j from 1 to $n$
14.     $D_j \leftarrow L^T A_j R$
15. EndFor
```

Theorem 3.3 implies that the updates of the matrices in Lines 5 and 8 of the **Algorithm 1** increase the value of $\sum_{i=1}^n ||L^T A_i R||_F^2$. Hence by Theorem 3.2, the value of $\sum_{i=1}^n ||A_i - LD_iR^T||_F^2$ decreases, or

$$\text{RMSRE} \equiv \sqrt{\frac{1}{n}\sum_{i=1}^n ||A_i - LD_iR^T||_F^2} \qquad (7)$$

decreases. Here RMSRE stands for the *root mean square reconstruction error*. A similar measure was used in (Ye et al., 2004). The convergence of the **Algorithm 1** follows, since RMSRE is bounded from below by 0, as stated in the following Theorem:

Theorem 3.4. Algorithm 1 *monotonically decreases the the RMSRE value as defined in Eq. (7), hence it converges.*

We thus use the RMSRE value to check the convergence of **Algorithm 1**. More specifically, let RMSRE(i) and RMSRE($i-1$) be the RMSRE value at the i-th and $(i-1)$-th iterations from **Algorithm 1**, then the convergence of the algorithm is determined by checking whether RMSRE$(i-1)$ − RMSRE$(i) < \eta$, for some small threshold $\eta > 0$. In the following experiments, we choose $\eta = 0.05$.

We investigate the convergence property of the proposed algorithm, using the four datasets described in Section 4. For simplicity, we set $\ell_1 = 20$ and $\ell_2 = 20$ for all cases. The results on all four datasets show that the RMSRE value drops dramatically during the first and second iterations and it stabilizes after two iterations. Thus, the proposed algorithm converges within two iterations for all four datasets.

3.3. Time and space complexities

We close this section by analyzing the time and space complexities of the proposed algorithm.

The most expensive steps in **Algorithm 1** are the formation of the matrices M_R and M_L in Lines 3 and 6 respectively, and the formation of D_j in Lines 13–15.

It takes $O(\ell_1 c(r+c)n)$ time for computing M_R and $O(\ell_2 r(r+c)n)$ time for computing M_L. The computation of $D_j = (L^T(A_jR))$ using the given order is $O(rc\ell_2 + r\ell_2\ell_1) = O(r\ell_2(c+\ell_1))$. Assume the number of iterations in the while loop (from Line 2 to Line 10 in **Algorithm 1**) is I. The total time complexity can be simplified as $O(I(r+c)^2 \max(\ell_1, \ell_2)n)$.

The key to the low space complexity of the algorithm is that the formation of the matrices M_R and M_L can be processed by reading the matrices A_i incrementally. It is easy to verify that the space complexity for **Algorithm 1** is $O(rc)$.

4. Experimental evaluations

In this section, we experimentally evaluate our proposed algorithm on face image data. All of our experiments are performed on a P4 1.80GHz linux machine with 1GB memory. For face images, the number, r, of rows and the number, c, of columns are comparable, we thus set both ℓ_1 and ℓ_2 equal to a common value, d, in all the following experiments, for simplicity. However, the algorithm works in the general case.

We present the four face image datasets used for our evaluation in the first part. The effect of the common value, d, for both ℓ_1 and ℓ_2, used in the proposed algorithm is discussed in the second part. Finally, a detailed comparative study between the proposed algorithm and SVD is provided, where the comparison is made on classification accuracy, and efficiency.

For all the experiments, we use the K-Nearest neighbors (K-NN) method based on the Euclidean distance for classification. We use *10-fold cross-validation* for estimating the classification accuracy. In 10-fold cross-validation, we divide the data into ten subsets of (approximately) equal size. Then we do the training and testing ten times, each time leaving out one of the subsets for training, and using only the omitted sub-

Table 2. Statistics of our test datasets

Dataset	Size	Dim	# of classes
PIX	300	10000	30
ORL	400	10304	40
AR	1638	8888	126
PIE	6615	38500	63

set for testing. The classification accuracy reported is the average from the ten runs.

4.1. Datasets

We use the following four well known face datasets, which are publicly available, in our experiments: PIX[2], ORL[3], AR[4] (Martinez & Benavente, 1998), and PIE[5](Sim et al., 2002). The statistics of the four datasets are summarized in Table 2. Note that for AR and PIE, we only use a subset of the whole datasets.

4.2. The effect of the value of d

Recall that we choose a common value, d, for both ℓ_1 and ℓ_2. Thus, the value of d determines the dimensionality in the transformed space by the proposed algorithm. A large d leads to a small compression ratio, which may not be effective for dimension reduction, while a small d may lose some information intrinsic in the dataset. It is difficult to determine the optimal value of d theoretically. We did extensive experiments using different values of d on the four datasets. The results are summarized in Figure 1, where the x-axis denotes the the value of d (between 2 and 20) and the y-axis denotes the classification accuracy with 1-Nearest-Neighbor as the classifier. As shown in Figure 1, the accuracy curves on the PIX, ORL and PIE datasets stabilize around $d = 4$ to 6, while the accuracy curve on the AR dataset stabilizes around $d = 18$. Note that the low accuracy on the AR dataset may be related to the large within-class variance of each class/individual in the AR dataset.

4.3. Classification effectiveness

In this experiment, we evaluate the effectiveness of the proposed algorithm in terms of classification accuracy and compare with SVD. Figures 2–4 show the accuracy curves of these two algorithms on the three face image datasets: PIX, ORL, and AR, respectively. The x-

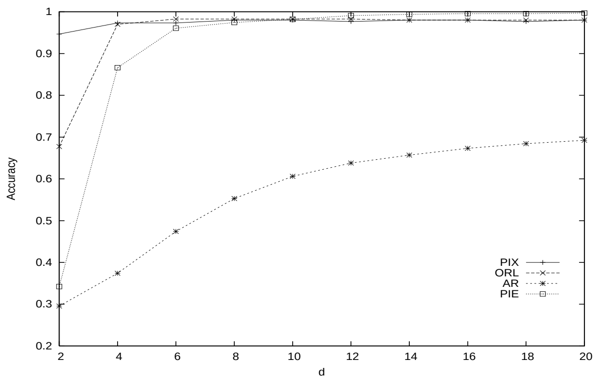

Figure 1. The effect of the value of d

axis denotes the number of nearest neighbors in K-NN, and the y-axis denotes the classification accuracy. Note that SVD is not applicable for PIE, due to its large size. However, the proposed algorithm has the minimum space requirement and is applicable for PIE. For all datasets, $d = 20$ is applied for the proposed algorithm. The reduced rank, k, in SVD is chosen such that both SVD and the proposed algorithm have the same compression ratio. Thus, $k = 12, 15, 73$ are chosen for datasets PIX, ORL, and AR, respectively.

The main observations include:

- For all datasets, both the proposed algorithm and SVD have the best performance when $K = 1$ nearest neighbor is used in K-NN.
- Our proposed algorithm outperforms SVD consistently for all datasets.
- Interestingly, SVD is not applicable for PIE, since SVD requires the whole data matrix to reside in main memory, which is not the case for the PIE dataset, due to its large size. However, our proposed algorithm has the minimum memory requirement and is thus applicable for PIE.

To give a concrete idea of the difference, Figure 5 shows images for 10 different persons from the ORL dataset. The 10 images in the first row are the original images from the dataset. The 10 images in the second row are the ones compressed by our proposed algorithm with $d = 20$. The compression ratio is about 25. The same compression ratio can be obtained, if the reduced rank in SVD is set to be 15. The resulting images are shown in the third row of Figure 5. It is clear that the images compressed by our proposed algorithm have better visual quality than those compressed by SVD, when using the same compression ratio.

[2]http://peipa.essex.ac.uk/ipa/pix/faces/manchester/
[3]http://www.uk.research.att.com/facedatabase.html
[4]http://rvl1.ecn.purdue.edu/~aleix/aleix_face_DB.html
[5]http://www.ri.cmu.edu/projects/project_418.html

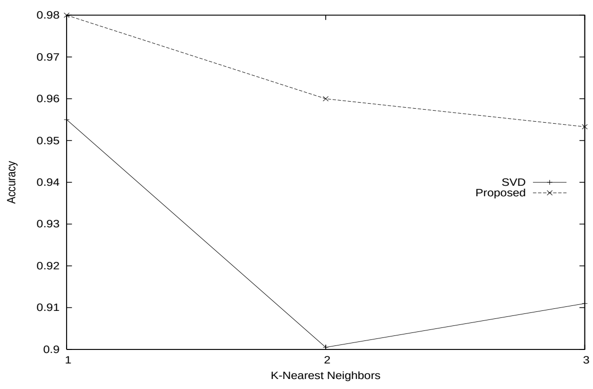

Figure 2. Comparison of classification accuracy using PIX dataset

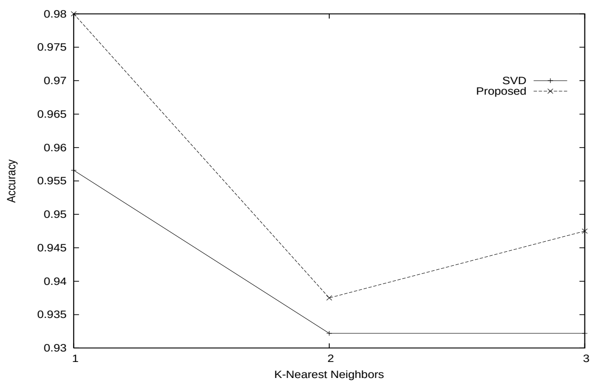

Figure 3. Comparison of classification accuracy using ORL dataset

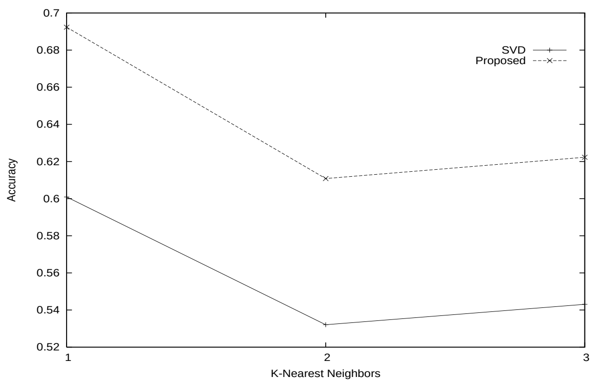

Figure 4. Comparison of classification accuracy using AR dataset

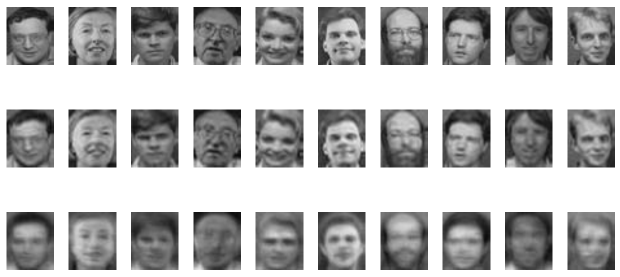

Figure 5. First row: original images. Second row: images compressed by the proposed algorithm. Third row: images compressed by SVD.

4.4. Comparative study on efficiency

In this experiment, we examine the efficiency of the proposed algorithm and compare with SVD. Figure 6 shows the CPU time of the proposed algorithm and SVD on four datasets. The CPU time of SVD on the PIE dataset is not shown, due to the same reason as mentioned above. The main observation is:

- The proposed algorithm has distinctly less computational time than SVD. As the size of the dataset gets larger from PIX to ORL to AR, the speedup of our proposed algorithm over SVD increases. For the AR dataset, our proposed algorithm is almost two orders of magnitude faster compared to SVD.

Note that the time complexity of the proposed algorithm is $O(I(r+c)^2dn)$. Assuming $r \approx c \approx \sqrt{N}$ (note $N = rc$), the time complexity of the proposed algorithm can be simplified as $O(IdnN)$, while the time complexity of SVD is $O(n^2N)$, assuming $n < N$. Hence, as n gets larger, the speed-up of the proposed algorithm over SVD is also larger.

5. Conclusions

A novel algorithm for low rank approximations of a sequence of matrices is presented. The algorithm is iterative, in the sense that the approximation is improved during iterations. Empirical results show that the algorithm converges within few iterations.

A natural application of this approach is in image compression and retrieval, where each image is represented in its native matrix form. We evaluate the proposed algorithm in terms of classification accuracy, and efficiency, and compare with traditional SVD based method. A key observation is that the proposed algorithm has minimum space requirement, and lower

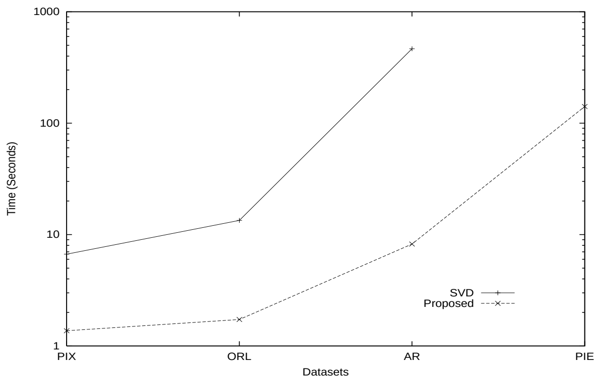

Figure 6. Comparison of efficiency using the four datasets. Note that SVD is not applicable for the PIE dataset, due to its large size.

time complexity than SVD, which is desirable for large datasets (such as PIE), while experiments show superior performance of the proposed algorithm over SVD, in terms of classification accuracy.

Acknowledgment

Research is sponsored, in part, by the Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory cooperative agreement number DAAD19-01-2-0014, the content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred.

References

Achlioptas, D., & McSherry, F. (2001). Fast computation of low rank matrix approximations. *ACM STOC Conference Proceedings* (pp. 611–618).

Aggarwal, C. C. (2001). On the effects of dimensionality reduction on high dimensional similarity search. *ACM PODS Conference Proceedings*. Santa Barbara, California, USA.

Berry, M., Dumais, S., & O'Brie, G. (1995). Using linear algebra for intelligent information retrieval. *SIAM Review*, *37*, 573–595.

Castelli, V., Thomasian, A., & Li, C.-S. (2003). Csvd: Clustering and singular value decomposition for approximate similarity searches in high dimensional space. *IEEE Transactions on Knowledge and Data Engineering*, *15*, 671–685.

Deerwester, S., Dumais, S., Furnas, G., Landauer, T., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the Society for Information Scienc*, *41*, 391–407.

Dhillon, I., & Modha, D. (2001). Concept decompositions for large sparse text data using clustering. *Machine Learning*, *42*, 143–175.

Drineas, P., Frieze, A., Kannan, R., Vempala, S., & Vinay, V. (1999). Clustering in large graphs and matrices. *ACM SODA Conference Proceedings* (pp. 291–299).

Edelman, A., Arias, T. A., & Smith, S. T. (1998). The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, *20*, 303–353.

Frieze, A., Kannan, R., & Vempala, S. (1998). Fast monte-carlo algorithms for finding low-rank approximations. *ACM FOCS Conference Proceedings* (pp. 370–378).

Golub, G. H., & Van Loan, C. F. (1996). *Matrix computations*. Baltimore, MD, USA: The Johns Hopkins University Press. Third edition.

Gu, M., & Eisenstat, S. C. (1993). *A fast and stable algorithm for updating the singular value decomposition* (Technical Report Technical Report YALEU/DCS/RR-966, Department of Computer Science, Yale University).

Kanth, K. V. R., Agrawal, D., Abbadi, A. E., & Singh, A. (1998). Dimensionality reduction for similarity searching in dynamic databases. *ACM SIGMOD Conference Proceedings* (pp. 166–176).

Kleinberg, J., & Tomkins, A. (1999). Applications of linear algebra in information retrieval and hypertext analysis. *ACM PODS Conference Proceedings*.

Martinez, A., & Benavente, R. (1998). *The ar face database* (Technical Report CVC Tech. Report No. 24).

Sim, T., Baker, S., & Bsat, M. (2002). The cmu pose, illumination, and expression (pie) database. *Proc. 4th Intl. Conf. on FG'02*.

Srebro, N., & Jaakkola, T. (2003). Weighted low-rank approximations. *ICML Conference Proceedings* (pp. 720–727).

Ye, J., Janardan, R., & Li, Q. (2004). GPCA: An efficient dimension reduction scheme for image compression and retrieval. *ACM SIGKDD Conference Proceedings*.