ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/2947521

Generalized Low Rank Approximations of Matrices

Article *in* Machine Learning · September 2004

DOI: 10.1145/1015330.1015347 · Source: CiteSeer

CITATIONS
376

READS
518

1 author:

Jieping Ye
Arizona State University
344 PUBLICATIONS **16,612** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Application Driven Advanced AI Computing View project

National Institute on Aging (R21AG04376; ; RF1AG05171; R01AG031581; P30AG19610); the National Institute of Biomedical Imaging and Bioengineering (R01EB025032; U54EB020403);the Arizona Alzheimer's Consortium View project

All content following this page was uploaded by Jieping Ye on 05 June 2013.

The user has requested enhancement of the downloaded file.

Generalized Low Rank Approximations of Matrices *

Jieping Ye (jieping@cs.umn.edu)
Department of Computer Science & Engineering, University of Minnesota-Twin Cities, Minneapolis, MN 55455, USA

Abstract.

The problem of computing low rank approximations of matrices is considered. The novel aspect of our approach is that the low rank approximations are on a collection of matrices. We formulate this as an optimization problem, which aims to minimize the reconstruction (approximation) error. To the best of our knowledge, the optimization problem proposed in this paper does not admit a closed form solution in general. We thus derive an iterative algorithm, namely GLRAM, which stands for the **G**eneralized **L**ow **R**ank **A**pproximations of **M**atrices. GLRAM reduces the reconstruction error sequentially, and the resulting approximation is thus improved during successive iterations. Experimental results show that the algorithm converges rapidly.

We conduct extensive experiments on 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 (SVD) based method, in terms of the reconstruction error, misclassification error rate, and computational time. Results show that GLRAM is competitive with SVD for classification, while it has a much lower computational cost. However, GLRAM results in a higher reconstruction error than SVD. To further reduce the reconstruction error, we study the combination of GLRAM and SVD, namely GLRAM+SVD, where SVD is preceded by GLRAM. Experimental results show that when using the same number of reduced dimensions, GLRAM+SVD achieves significant reduction of reconstruction error as compared to GLRAM, while keeping the computational cost low.

Keywords: Singular Value Decomposition, matrix approximation, reconstruction error, classification.

Table of Contents

* A preliminary version of this paper appears in the Proceedings of the Twenty-First International Conference on Machine Learning (Outstanding Student Paper), Banff, Alberta, Canada, pp. 887-894, 2004.

© 2004 *Kluwer Academic Publishers. Printed in the Netherlands.*

1. Introduction

The problem of dimension reduction has recently received broad attention in areas such as machine learning, computer vision, and information retrieval (Berry et al., 1995; Castelli et al., 2003; Deerwester et al., 1990; Dhillon and Modha, 2001; Srebro and Jaakkola, 2003). The goal of dimension reduction is to obtain more compact representations of the data with limited loss of information. Traditional algorithms for dimension reduction are based on the so-called *vector space model*. 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 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 (Aggarwal, 2001; Castelli et al., 2003).

A well-known technique for dimension reduction is the low rank approximation by the Singular Value Decomposition (SVD), also called Latent Semantic Indexing (LSI) in information retrieval (Berry et al., 1995). An appealing property of this low rank approximation is that it achieves the smallest reconstruction error among all approximations with the same rank. Details can be found in Section 2. Some theoretical justification of the empirical success of LSI can be found in (Papadimitriou et al., 1998), where it shows that LSI works in the context of a simple probabilistic "Corpus-generating" model. 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 (Golub and Van Loan, 1996).

Several incremental algorithms have been proposed in the past (Brand, 2002; Gu and 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 the best of our knowledge, such algorithms come with no guarantees on the quality of the approximation produced. Random sampling can be applied to speed up the SVD computation. More details on this can be found in (Achlioptas and McSherry, 2001; Drineas et al., 1999; Frieze et al., 1998).

1.1. Contributions

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

Both GLRAM and SVD aim to minimize the reconstruction error. The essential difference is that GLRAM applies a bilinear transformation on the data. Such a bilinear transformation is particularly appropriate for data in matrix representation and often leads to lower computational cost compared to SVD. We apply GLRAM on image compression and retrieval, where each image is represented in its native matrix representation. To evaluate the proposed algorithm, we conduct extensive experiments on five well-known image datasets: PIX, ORL, AR, PIE, and USPS, where USPS consists of images of handwritten digits and the other four are face image datasets. GLRAM is compared with SVD, as well as 2DPCA, a recently proposed algorithm for dimension reduction. Details on 2DPCA can be found in Section 4.

Experimental results show that when using the same number of reduced dimensions, GLRAM is competitive with SVD for classification, while it has a much lower computational cost. However, GLRAM results in a higher reconstruction error than SVD. The underlying reason may be that GLRAM is able to utilize the locality information intrinsic in the image, which leads to good classification performance. In terms of compression ratio[1], GLRAM outperforms SVD, especially when the number of data points is relatively small compared to the number of dimensions. For large and high-dimensional datasets, the lack of available space becomes a critical issue. In this case, compression ratio will be an important factor in evaluating different dimension reduction algorithms.

To further reduce the reconstruction error of GLRAM, we study the combination of GLRAM and SVD, namely GLRAM+SVD, which applies SVD after the intermediate dimension reduction stage using GLRAM. The essence of this composite algorithm is a further dimension reduction stage by SVD following GLRAM. Since SVD is applied to a low-dimensional space transformed by GLRAM, the second stage by SVD can be implemented efficiently. We apply this algorithm to image datasets and compare it with SVD when using the same number of reduced dimensions. Experimental results show that when using the same number of reduced dimensions, GLRAM+SVD achieves a significant reduction of reconstruction error as

[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.

Table I. 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
M_i	reduced representation of A_i
ℓ_1	number of rows in M_i
ℓ_2	number of columns in M_i
d	common value for ℓ_1 and ℓ_2
k	rank of reduced matrix by SVD
A	data matrix of size N by n
n	number of training data points
N	dimension of training data ($N = rc$)

compared to GLRAM, while keeping the computational cost small. The reconstruction error of GLRAM+SVD is close to that of SVD, especially when the intermediate reduced dimension in the GLRAM stage is large, while it has a smaller computational cost than SVD.

In summary, GLRAM can be applied as a pre-processing step for SVD. The pre-processing by GLRAM reduces signifi cantly the computational cost of the SVD computation, while keeping the reconstruction error small (see Section 5.6).

1.2. ORGANIZATION

The rest of this paper is organized as follows. We give a brief overview of low rank approximations of matrices in Section 2. The problem of generalized low rank approximations of matrices is studied in Section 3. Some related work is presented in Section 4. A performance study is provided in Section 5. Conclusions and directions for future work can be found in Section 6.

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

2. Low Rank Approximations of Matrices

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 $A \in \mathbb{R}^{N \times n}$, 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 $B \in \mathbb{R}^{N \times n}$ 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 and Van Loan, 1996).

THEOREM 2.1. *Let the Singular Value Decomposition of $A \in \mathbb{R}^{N \times n}$ 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, $a_i \in \mathbb{R}^N$, of A can be approximated as $a_i \approx U_k a_i^L$, for some $a_i^L \in \mathbb{R}^k$. 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. The speed-up on a single distance computation using the reduced representations is $\frac{rc}{k}$. This forms the basis for Latent Semantics Indexing (Berry et al., 1995; Deerwester et al., 1990), used widely in informational retrieval.

Another potential application of the above rank-k approximation is for 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 $U_k \in \mathbb{R}^{N \times k}$ and $a_i^L \in \mathbb{R}^k$, for $i = 1, \cdots, n$, it requires $nk + Nk =$

$(n+N)k$ scalars 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 $A \in \mathbb{R}^{N\times n}$.

3. Generalized Low Rank Approximations of Matrices

In this section, we study the problem of generalized low rank approximations of matrices, which aims to approximate a collection 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 restrict its applicability to matrices with small size. Instead, we apply a different data representation model, under which, each datum is represented as a matrix and the collection of data is represented as a collection of matrices. The corresponding generalized low rank approximation problem becomes the problem of approximating a collection of matrices with lower rank. Details are given below.

3.1. Problem formulation

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

$$\min_{\substack{L\in\mathbb{R}^{r\times\ell_1}:\ L^TL=I_{\ell_1}\\ R\in\mathbb{R}^{c\times\ell_2}:\ R^TR=I_{\ell_2}\\ M_i\in\mathbb{R}^{\ell_1\times\ell_2}:\ i=1,\cdots,n}} \sum_{i=1}^{n} ||A_i - LM_iR^T||_F^2. \tag{1}$$

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 M_i's are not required to be diagonal.

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

- **Data compression:** The matrices L, R, and $\{M_i\}_{i=1}^n$ can be used to recover the original n matrices $\{A_i\}_{i=1}^n$, assuming LM_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, R, and $\{M_i\}_{i=1}^n$. Hence, the storage saved, or the *compression ratio* using the approximations is $\frac{nrc}{r\ell_1+c\ell_2+n\ell_1\ell_2}$.
- **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, we have $||A_i - A_j||_F \approx ||LM_iR^T - LM_jR^T||_F = ||M_i - M_j||_F$, since both L and R have orthonormal columns. It is clear that the computational cost for computing $||M_i - M_j||_F$ 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 5.

The formulation in Eq. (1) is general, in the sense that ℓ_1 and ℓ_2 can be different, i.e., M_i can have an arbitrary shape. We will study the effect of the shape of M_i on the algorithm in Section 5.2.

3.2. THE MAIN ALGORITHM

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

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

Proof. By the property of the trace of matrices,

$$\begin{aligned}\sum_{i=1}^{n} ||A_i - LM_iR^T||_F^2 &= \sum_{i=1}^{n} \text{trace}\left((A_i - LM_iR^T)(A_i - LM_iR^T)^T\right) \\ &= \sum_{i=1}^{n} \text{trace}(A_iA_i^T) + \sum_{i=1}^{n} \text{trace}(M_iM_i^T) \\ &\quad - 2\sum_{i=1}^{n} \text{trace}(LM_iR^TA_i^T), \end{aligned} \tag{2}$$

where the second term $\sum_{i=1}^n \text{trace}(M_iM_i^T)$ results from the fact that both L and R have orthonormal columns, and $\text{trace}(AB) = \text{trace}(BA)$, 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

$$\sum_{i=1}^{n} \text{trace}(M_i M_i^T) - 2 \sum_{i=1}^{n} \text{trace}(L M_i R^T A_i^T). \tag{3}$$

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

Theorem 3.1 implies that M_i is uniquely determined by L and R with $M_i = L^T A_i R$, 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 $\{M_i\}_{i=1}^n$ be the optimal solution to the minimization problem in Eq. (1). Then L and R solve the following optimization problem:*

$$\max_{\substack{L \in \mathbb{R}^{r \times \ell_1} : \ L^T L = I_{\ell_1} \\ R \in \mathbb{R}^{c \times \ell_2} : \ R^T R = I_{\ell_2}}} \sum_{i=1}^{n} ||L^T A_i R||_F^2. \tag{4}$$

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

$$\sum_{i=1}^{n} ||A_i - L M_i R^T||_F^2 = \sum_{i=1}^{n} ||A_i||_F^2 - \sum_{i=1}^{n} ||L^T A_i R||_F^2. \tag{5}$$

Hence the minimization in Eq. (1) is equivalent to the maximization of

$$\sum_{i=1}^{n} ||L^T A_i R||_F^2,$$

which completes the proof of the theorem.

To the best of 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, is stated in the following theorem:

THEOREM 3.3. *Let L, R and $\{M_i\}_{i=1}^n$ 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

$$M_L = \sum_{i=1}^{n} A_i R R^T A_i^T$$

corresponding to the largest ℓ_1 eigenvalues.

(2). For a given L, R consists of the ℓ_2 eigenvectors of the matrix

$$M_R = \sum_{i=1}^{n} A_i^T L L^T A_i$$

corresponding to the largest ℓ_2 eigenvalues.

Proof. By Theorem 3.2, L and R maximize

$$\sum_{i=1}^{n} ||L^T A_i R||_F^2,$$

which can be rewritten as

$$\begin{aligned} \sum_{i=1}^{n} \text{trace}(L^T A_i R R^T A_i^T L) &= \text{trace}\left(L^T \sum_{i=1}^{n} (A_i R R^T A_i^T) L \right) \\ &= \text{trace}\left(L^T M_L L \right), \end{aligned} \tag{6}$$

where $M_L = \sum_{i=1}^{n} A_i R R^T A_i^T$. Hence, for a given R, the maximum of

$$\sum_{i=1}^{n} ||L^T A_i R||_F^2 = \text{trace}\left(L^T M_L L \right)$$

is obtained, only if $L \in \mathbb{R}^{r \times \ell_1}$ consists of the ℓ_1 eigenvectors of the matrix M_L corresponding to the largest ℓ_1 eigenvalues. The maximization of trace $\left(L^T M_L L \right)$ can be considered as a special case of the more general optimization problem in (Edelman et al., 1998).

Similarly, by the property of the trace of matrices,

$$\sum_{i=1}^{n} ||L^T A_i R||_F^2$$

can also be rewritten as

$$\begin{aligned} \sum_{i=1}^{n} \text{trace}(R^T A_i^T L L^T A_i R) &= \text{trace}\left(R^T \sum_{i=1}^{n} (A_i^T L L^T A_i) R \right) \\ &= \text{trace}\left(R^T M_R R \right), \end{aligned} \tag{7}$$

where $M_R = \sum_{i=1}^{n} A_i^T L L^T A_i$. Hence, for a given L, the maximum of

$$\sum_{i=1}^{n} ||L^T A_i R||_F^2 = \text{trace}\left(R^T M_R R \right)$$

is obtained, only if $R \in \mathbb{R}^{c \times \ell_2}$ consists of the ℓ_2 eigenvectors of the matrix M_R corresponding to the largest ℓ_2 eigenvalues. This completes the proof of the theorem.

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 R following the above iterative procedure is given in **Algorithm GLRAM**.

Algorithm GLRAM

Input: matrices $\{A_i\}_{i=1}^n$, ℓ_1, and ℓ_2

Output: matrices L, R, and $\{M_i\}_{i=1}^n$

---

1. Obtain initial L_0 for L 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 ℓ_2 eigenvectors $\{\phi_j^R\}_{j=1}^{\ell_2}$ of M_R corresponding to the largest ℓ_2 eigenvalues
5. $R_i \leftarrow \left[\phi_1^R, \cdots, \phi_{\ell_2}^R\right]$
6. form the matrix $M_L = \sum_{j=1}^n A_j R_i R_i^T A_j^T$
7. compute the ℓ_1 eigenvectors $\{\phi_j^L\}_{j=1}^{\ell_1}$ of M_L corresponding to the largest ℓ_1 eigenvalues
8. $L_i \leftarrow \left[\phi_1^L, \cdots, \phi_{\ell_1}^L\right]$
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. $M_j \leftarrow L^T A_j R$
15. EndFor

Theoretically, the solution from **Algorithm GLRAM** is only locally optimal. The solution depends on the choice of the initial L_0 for L. We did extensive experiments using different choices of the initial L_0 and found that, for image datasets, GLRAM always converges to the same solution, regardless of the choice of the initial L_0. For simplicity, we set $L_0 = \begin{pmatrix} I_{\ell_1} \\ 0 \end{pmatrix}$ in the experiments, where I_{ℓ_1} is the $\ell_1 \times \ell_1$ identity matrix.

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

Theorem 3.2, the value of $\sum_{i=1}^{n} ||A_i - LM_iR^T||_F^2$ decreases, or

$$\text{RMSRE} \equiv \sqrt{\frac{1}{n}\sum_{i=1}^{n} ||A_i - LM_iR^T||_F^2} \tag{8}$$

decreases. Here RMSRE stands for the *Root Mean Square Reconstruction Error*. The convergence of **Algorithm GLRAM** follows, since RMSRE is bounded from below by 0, as stated in the following Theorem:

THEOREM 3.4. **Algorithm GLRAM** *monotonically decreases the RMSRE value as defined in Eq. (8), hence it converges.*

We thus use the relative reduction of the RMSRE value to check the convergence of **Algorithm GLRAM**. More specifically, let $\text{RMSRE}(i)$ and $\text{RMSRE}(i-1)$ be the RMSRE values at the i-th and $(i-1)$-th iterations of **Algorithm GLRAM**, then the convergence of the algorithm is determined by checking whether

$$\frac{\text{RMSRE}(i-1) - \text{RMSRE}(i)}{\text{RMSRE}(i-1)} < \eta,$$

for some small threshold $\eta > 0$. In the following experiments, we choose $\eta = 10^{-6}$. We show in Section 5 that the algorithm in general converges within two to three iterations.

REMARK 3.1. *Note that the transformation matrices L and R in GLRAM may not converge, even when the RMSRE value converges. To see why this is the case, consider two pairs of solutions (L,R) and (LP,RQ), for any orthogonal matrices* $P \in \mathbb{R}^{\ell_1 \times \ell_1}$ *and* $Q \in \mathbb{R}^{\ell_2 \times \ell_2}$. *Since*

$$RMSRE \equiv \sqrt{\frac{1}{n}\sum_{i=1}^{n} ||A_i - LM_iR^T||_F^2} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} ||A_i - LL^TA_iRR^T||_F^2},$$

it is easy to check that both (L,R) and (LP,RQ) result in the same RMSRE value. Thus, the solution is invariant under arbitrary orthogonal transformations. Two transformations L and $\hat{L}$ *can be compared by computing the largest principal angle (*Bjork and Golub, 1973*;* Golub and Van Loan, 1996*) between the column spaces of L and* $\hat{L}$. *If the angle is zero, L is essentially equivalent to* $\hat{L}$ *up to an orthogonal transformation. More details on this computation can be found in (*Bjork and Golub, 1973*;* Golub and Van Loan, 1996*).*

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 GLRAM** are the formation of the matrices M_R and M_L in Lines 3 and 6 respectively, and the formation of M_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 time of $M_j = (L^T(A_j R))$ 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 GLRAM**) is I. The total time complexity can be simplified as $O(I(r+c)^2 \max(\ell_1,\ell_2)n)$.

It is easy to verify that the space complexity of **Algorithm GLRAM** is $O(rc) = O(N)$. 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.

REMARK 3.2. *Note that GLRAM involves eigenvalue problems of size r^2 or c^2, as compared to size $rcn(= Nn)$ in SVD. This is the key reason why GLRAM has much lower costs in time and space than SVD.*

4. Related work

Wavelet transform is a commonly used scheme for image compression (Averbuch et al., 1996). Similar to the GLRAM algorithm in this paper, wavelets can be applied to images in matrix representation. A subtle but important difference between wavelet compression and GLRAM compression is that the former mainly aims to compress and reconstruct a single image with small cost of basis representations, which is extremely important for image transmission in computer networks, whereas, GLRAM compression aims to compress a set of images by making use of the correlation information between images.

The work that is most closely related to the current one is the two-dimensional Principal Component Analysis (2DPCA) algorithm recently proposed in (Yang et al., 2004). Like GLRAM, 2DPCA works with data in matrix representation. The key difference is that 2DPCA applies linear transformation on the right side of each data, while GLRAM applies two-sided linear transformation (Note that each data corresponds to a two-dimensional matrix). 2DPCA can be formulated as a trace optimization problem, from which a closed-form solution is obtained. However, a disadvantage of 2DPCA, as also mentioned in (Yang et al., 2004) is that the number of reduced dimensions of 2DPCA can be quite large. This can be a serious problem when using limited storage. More details are given below.

Table II. Comparison of SVD, 2DPCA, and GLRAM: n is the number of data points in the training dataset and $N = r \times c$ is the dimension of the data.

Methods	Time	Space
SVD	$O(nN\min(n,N))$	$O(nN)$
2DPCA	$O\left(nN^{3/2}\right)$	$O(N)$
GLRAM	$O\left(I(r+c)^2\max(\ell_1,\ell_2)n\right)$	$O(N)$

2DPCA aims to find a linear transformation $X \in \mathbb{R}^{c\times\ell}$ with $\ell < c$, such that each image $A_i \in \mathbb{R}^{r\times c}$ is transformed (projected) to $Y_i = A_i X \in \mathbb{R}^{r\times\ell}$. The variance of the n projections $\{Y_i\}_{i=1}^n$ can be computed as

$$\frac{1}{n-1}\sum_{i=1}^{n}||Y_i-\bar{Y}||_F^2 = \frac{1}{n-1}\sum_{i=1}^{n}X^T(A_i-\bar{A})^T(A_i-\bar{A})X,$$

where $\bar{Y} = \frac{1}{n}\sum_{i=1}^n Y_i = \bar{A}X$ is the mean and $\bar{A} = \frac{1}{n}\sum_{i=1}^n A_i$.

2DPCA aims to find the optimal transformation X such that the variance of the n data points in the transformed space is maximized. Specifically, the optimal transformation X can be computed by solving the following maximization problem:

$$X = \arg\max_{X^TX=I_\ell}\left(\frac{1}{n-1}\sum_{i=1}^{n}X^T(A_i-\bar{A})^T(A_i-\bar{A})X\right). \tag{9}$$

The optimal X can be obtained by computing the ℓ eigenvectors of the matrix $\frac{1}{n-1}\sum_{i=1}^n(A_i-\bar{A})^T(A_i-\bar{A})$ corresponding to the largest ℓ eigenvalues.

The algorithm requires $c\ell + nr\ell$ scalars to store the transformation $X \in \mathbb{R}^{c\times\ell}$ and the reduced representations $\{Y_i\}_{i=1}^n \in \mathbb{R}^{r\times\ell}$. Hence, the *compression ratio* by 2DPCA is $\frac{nrc}{c\ell+nr\ell} \approx \frac{c}{\ell}$.

Table II lists the time and space complexities of SVD, 2DPCA, and GLRAM. It is clear that GLRAM and 2DPCA have much smaller costs in time and space than SVD.

5. Experimental evaluations

In this section, we experimentally evaluate the GLRAM algorithm. All of our experiments are performed on a P4 2.785GHz Linux machine with 1GB

memory. A MATLAB version of the GLRAM algorithm can be accessed at http://www-users.cs.umn.edu/∼jieping/GLRAM/.

We present one synthetic dataset and five real-world image datasets used for our evaluation in Section 5.1. The effect of the ratio of ℓ_1 to ℓ_2 on reconstruction error is discussed in Section 5.2. Results show that for the datasets considered in the paper, choosing $\frac{\ell_1}{\ell_2} \approx 1$ achieves good performance in general. We thus set both ℓ_1 and ℓ_2 equal to a common value d in the following experiments. The sensitivity of GLRAM to the choice of the initial L_0 for L is studied in Section 5.3. In Sections 5.4–5.5, a detailed comparative study between the proposed algorithm and SVD is provided, where the comparison is made on the reconstruction error (measured by RMSRE), classification, and quality of compressed images. The results on 2DPCA (Yang et al., 2004) are also included. The effectiveness of SVD is critically dependent on the reduced dimension k. For a complete comparison, we use two different choices of k for SVD: (1) In SVD-I, k is chosen such that the compression ratio of SVD is approximately the same as that of GLRAM; (2) In SVD-II, k is chosen such that both SVD and GLRAM have the same number of reduced dimensions. Finally, we study the GLRAM+SVD algorithm in Section 5.6, where the dimension by GLRAM is further reduced by SVD.

For all the experiments, we use the K-Nearest-Neighbors (K-NN) method with $K = 1$ based on the Euclidean distance for classification (Duda et al., 2000; Fukunaga, 1990). We use *10-fold cross-validation* for estimating the misclassification error rate. 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 subset for testing. The misclassification error rate reported is the average from the ten runs.

5.1. DATASETS

We use the following six datasets (one synthetic dataset and five real-world image datasets) in our experiments:

- RAND is a synthetic dataset, consisting of 500 data points of size 100×100. All the entries are randomly generated between 0 and 225 (the same range as the four face image datasets).

- PIX[2] contains 300 face images of 30 persons. The image size of PIX image is 512×512. We subsample the images down to a size of $100 \times 100 = 10000$.

[2] http://peipa.essex.ac.uk/ipa/pix/faces/manchester/test-hard/

– ORL[3] is a well-known dataset for face recognition (Samaria and Harter, 1994). It contains the face images of 40 persons, for a total of 400 images. The image size is 92×112. The face images are perfectly centralized. The major challenge on this dataset is the variation of the face pose. We use the whole image as an instance (i.e., the dimension of an instance is $92 \times 112 = 10304$).

– AR[4] is a large face image dataset (Martinez and Benavente, 1998). The instance of each face may contain large areas of occlusion, due to the presence of sun glasses and scarves. The existence of occlusion dramatically increases the within-class variances of AR face image data. We use a subset of AR. This subset contains 1638 face images of 126 persons. Its image size is 768×576. We first crop the image from row 100 to 500, and column 200 to 550, and then subsample the cropped images down to a size of $101 \times 88 = 8888$.

– PIE[5] is a subset of the CMU–PIE face image dataset (Sim et al., 2004). PIE contains 6615 face instances of 63 persons. More specifically, each person has $21 \times 5 = 105$ instances taken under 21 different lighting conditions and 5 different poses. The image size of PIE is 640×480. We pre-process each image using a similar technique as above. The final dimension of each instance is $32 \times 24 = 768$.

– USPS[6] is an image dataset consisting of 9298 handwritten digits of “0" through “9". We use a subset of USPS. This subset contains 300 images for each digit, for a total of 3000 images. The image size is $16 \times 16 = 256$.

The statistics of all datasets are summarized in Table III.

5.2. Effect of the ratio of ℓ_1 to ℓ_2 on reconstruction error

In this experiment, we study the effect of the ratio of ℓ_1 to ℓ_2 on reconstruction error, where ℓ_1 and ℓ_2 are the row and column dimensions of the reduced representation M_i in GLRAM. To this end, we run GLRAM with different combinations of ℓ_1 and ℓ_2 with a constant product $\ell_1 \cdot \ell_2 = 400$. The results on PIX, ORL, and AR are shown in Table IV. It is clear from the table that the RMSRE value is small, when $\frac{\ell_1}{\ell_2} \approx 1$, and the minimum is achieved when $\frac{\ell_1}{\ell_2} = 1$ in all cases.

[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
[6] http://www-stat-class.stanford.edu/∼tibs/ElemStatLearn/data.html

Table III. Statistics of our test datasets.

Dataset	Size	Dimension	Number of classes
RAND	500	$100 \times 100 = 10000$	—
PIX	300	$100 \times 100 = 10000$	30
ORL	400	$92 \times 112 = 10304$	40
AR	1638	$101 \times 88 = 8888$	126
PIE	6615	$32 \times 24 = 768$	63
USPS	3000	$16 \times 16 = 256$	10

To examine whether this is related to the fact that for images, the number of rows (r) and the number of columns (c) are comparable, we subsample the images in PIX down to a size of $50 \times 100 = 5000$. The result on this dataset is included in Table IV. Interestingly, we observe the same trend in this dataset. That is, the RMSRE value is small, when $\frac{\ell_1}{\ell_2} \approx 1$. We have conducted similar experiments on other datasets and observed the same trend. This may be related to the effect of balancing between the left and right transformations involved in GLRAM. However, further theoretical analysis is needed for a more complete understanding.

Finally, we examine the effect of the ratio using the synthetic dataset. The result on RAND is included in the last column of Table IV. Not surprisingly, we observe the same trend as other datasets. That is, the RMSRE value is small, when $\frac{\ell_1}{\ell_2} \approx 1$.

The above experiment on both the synthetic and real-world datasets suggests that choosing $\frac{\ell_1}{\ell_2} \approx 1$ may be a good strategy in practice. In all the following experiments, we set both ℓ_1 and ℓ_2 equal to a common value d.

5.3. Sensitivity of GLRAM to the choice of the initial L_0

In this experiment, we examine the sensitivity of GLRAM to the choice of the initial L_0 for L (See Line 1 of Algorithm GLRAM). To this end, we run GLRAM with 10 different initial L_0's. The first one is chosen to be $L_0 = \begin{pmatrix} I_d \\ 0 \end{pmatrix}$, with the next nine being randomly generated.

First, we study the sensitivity of GLRAM using the image datasets and the results on ORL, AR, PIE with $d = 10$ are shown in Tables V–VII, respectively. We observe the same trend in other two image datasets (PIX and USPS) as well as different values of d and the results are omitted.

Table IV. Effect of the ratio of ℓ_1 to ℓ_2 on reconstruction error: Row shown in bold has minimum RMSRE (where $\ell_1 = \ell_2$)

Parameters		Datasets				
ℓ_1	ℓ_2	PIX	ORL	AR	PIX (50×100)	RAND
5	80	569.06	2128.8	3605.4	384.67	7189.9
8	50	441.72	1737.2	2822.0	290.97	7177.5
10	40	387.47	1580.1	2457.4	250.55	7174.1
16	25	294.90	1376.9	1978.3	180.88	7170.9
20	**20**	**278.01**	**1367.3**	**1902.8**	**169.28**	**7170.6**
25	16	279.81	1423.9	1965.4	172.43	7171.1
40	10	349.12	1697.6	2379.1	226.90	7174.2
50	8	406.06	1864.6	2629.4	269.79	7177.2
80	5	529.26	2366.1	3426.3	——	7190.0

Table V. Sensitivity of GLRAM to the choice of the initial L_0 on ORL

L_0	Iterations				Angle
	0	1	2	3	
1	12121.54874	1968.168612	1967.952294	1967.952278	0
2	4541.996022	1968.009617	1967.952283	1967.952278	1.195e-8
3	4307.024799	1968.074193	1967.952290	1967.952278	1.666e-8
4	4151.739975	1968.070217	1967.952288	1967.952278	1.382e-8
5	4433.835944	1968.065263	1967.952289	1967.952278	1.418e-8
6	4570.412072	1968.058954	1967.952288	1967.952278	1.043e-8
7	4353.377443	1968.020735	1967.952285	1967.952278	7.873e-7
8	4497.482467	1968.020989	1967.952284	1967.952278	1.275e-8
9	4401.673621	1968.039604	1967.952286	1967.952278	1.299e-8
10	4401.261703	1968.044285	1967.952286	1967.952278	1.209e-8

Tables V–VII show that GLRAM converges rapidly for all cases. It converges within two to three iterations with the specifi ed threshold ($\eta = 10^{-6}$). Surprisingly, with ten different initial L_0's, GLRAM converges to the same RMSRE value for all datasets. To check whether GLRAM converges to same solution, we compare the resulting left transformations L from the ten different runs. Two transformations can be compared by computing the largest principal angle between the column spaces of these two transformations, as discussed in Section 3.1. The angle between the left transformation (L) re-

Table VI. Sensitivity of GLRAM to the choice of the initial L_0 on AR

L_0	Iterations				Angle
	0	1	2	3	
1	15478.03754	2942.397518	2942.349355	2942.349354	0
2	6991.669530	2942.442488	2942.349364	2942.349354	4.151e-10
3	7251.491163	2942.465450	2942.349366	2942.349354	4.796e-10
4	7104.602018	2942.469570	2942.349367	2942.349354	4.958e-10
5	7191.534548	2942.467857	2942.349367	2942.349354	4.846e-10
6	6999.575066	2942.464490	2942.349366	2942.349354	4.728e-10
7	7222.620240	2942.453968	2942.349365	2942.349354	4.520e-10
8	7128.513114	2942.449357	2942.349364	2942.349354	4.335e-10
9	7150.567923	2942.445696	2942.349364	2942.349354	4.245e-10
10	7137.989273	2942.453681	2942.349365	2942.349354	4.423e-10

Table VII. Sensitivity of GLRAM to the choice of the initial L_0 on PIE

L_0	Iterations				Angle
	0	1	2	3	
1	2143.208547	432.4554351	432.4446340	432.4446324	0
2	881.7393872	432.4453732	432.4446324	432.4446324	1.077e-8
3	844.5166881	432.4459057	432.4446325	432.4446324	1.097e-8
4	818.2174414	432.4454948	432.4446325	432.4446324	7.922e-9
5	860.2096378	432.4469701	432.4446326	432.4446324	9.404e-9
6	857.9641181	432.4458738	432.4446325	432.4446324	9.862e-9
7	819.1340553	432.4465250	432.4446326	432.4446324	1.495e-8
8	821.9415560	432.4470740	432.4446329	432.4446324	1.311e-8
9	812.0204595	432.4471415	432.4446327	432.4446324	1.311e-8
10	873.4025024	432.4466218	432.4446325	432.4446324	1.050e-8

sulting from the first choice and the ones from other nine choices are shown in the last column of Tables V–VII. For all cases, the angles are around 10^{-10} to 10^{-7}. This implies that GLRAM essentially converges to the same solution (subject to an orthogonal transformation) for the ten different runs.

Next, we examine the sensitivity of GLRAM using RAND, the synthetic dataset. The result is shown in Table VIII. It is clear from the table that GLRAM converges much slower on RAND than on other datasets. We run

Table VIII. Sensitivity of GLRAM to the choice of the initial L_0 on RAND

L_0	Iterations				Angle
	0	1	2	3	
1	14134.49899	7397.343307	7308.715675	7308.283538	0
2	7672.847212	7309.866116	7308.719574	7308.190248	1.53
3	7561.946266	7309.863588	7308.702267	7308.200739	1.56
4	7599.646135	7309.997780	7308.836461	7308.333442	1.54
5	7677.215059	7309.932531	7308.882316	7308.315683	1.56
6	7642.565069	7309.951122	7309.004933	7308.541626	1.56
7	7644.935680	7309.744509	7308.854529	7308.382045	1.57
8	7609.209061	7309.737955	7308.546719	7308.079770	1.54
9	7689.611763	7309.901728	7308.790172	7308.354200	1.56
10	7612.699151	7309.977470	7308.937469	7308.470551	1.50

GLRAM with the threshold $\eta = 10^{-6}$, and it does not converge within ten iterations. Furthermore, GLRAM does not converge to the same solution, as shown in the last column of Table VIII (measured by the angle between two subspaces). Further experiments also show that the final RMSRE value may be different for different initial L_0's, even though the difference seems always small. This is likely due to the fact that there are some similarities among the images in the same image datasets, while the data in RAND is randomly generated.

The above experiment implies that for datasets with some hidden structures, such as faces and handwritten digits, GLRAM may converge to the global solution, regardless of the choice of the initial L_0. However, it is not true in general, as shown in the RAND dataset.

5.4. Comparison of reconstruction error and classification

In this experiment, we evaluate the effectiveness of the proposed algorithm in terms of the reconstruction error measured by RMSRE and classification measured by misclassification error rate, by comparing it with 2DPCA and SVD. For SVD, we apply two different versions: SVD-I and SVD-II as mentioned earlier. In SVD-I, k is chosen such that the compression ratio of SVD is approximately the same as that of GLRAM. In SVD-II, k is chosen such that both SVD and GLRAM have the same number of reduced dimensions. More specifically, in SVD-I, $k \approx \frac{d(r+c)+nd^2}{rc+n}$, and in SVD-II, $k = d^2$, where d is the common value for both ℓ_1 and ℓ_2. It is worthwhile to note that when

$n \gg rc$, $k \approx \frac{d(r+c)+nd^2}{rc+n} \approx d^2$. That is, for relatively large n (compared to rc), SVD-I approximates SVD-II.

Figures 1–5 show the results on the five image datasets: PIX, ORL, AR, PIE, and USPS respectively. The x-axis denotes the value of d, and the y-axis denotes the RMSRE value (left graph) and misclassification rate (right graph). Figure 6 shows the compression ratio of each algorithm on AR (left graph) and PIE (right graph), two representatives of all image datasets in Table III.

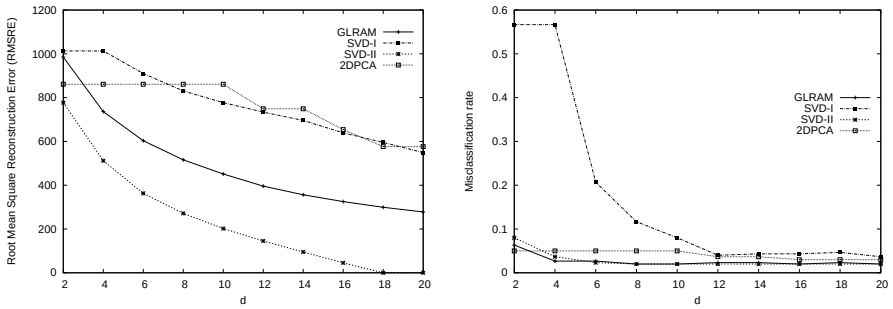

Figure 1. Comparison of reconstruction error (left) and misclassification rate (right) on PIX

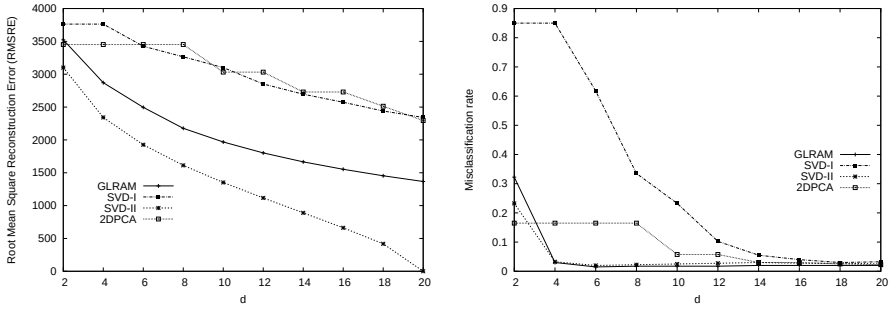

Figure 2. Comparison of reconstruction error (left) and misclassification rate (right) on ORL

The main observations include:

– As d increases, the reconstruction errors by GLRAM decrease monotonically for all cases, while the misclassification rate decrease monotonically in most cases. The same trend can be observed from other algorithms. Thus choosing a large d in general improves the performance of GLRAM in reconstruction and classification. However, the computational cost of GLRAM also increases as d increases, as shown

in Table II (Note that $d = \ell_1 = \ell_2$). There is a tradeoff between the performance and the computational cost, when choosing the best d in GLRAM.

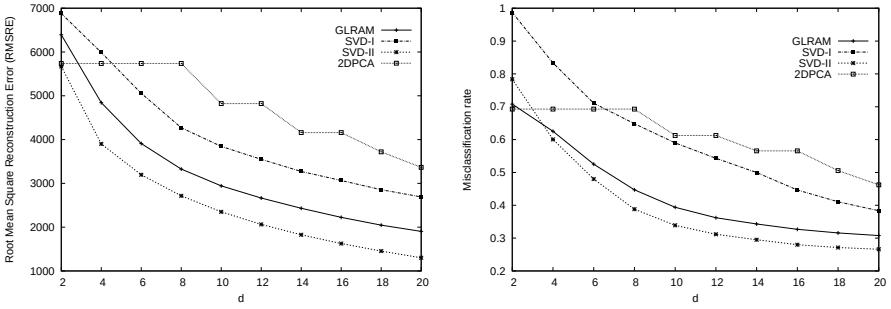

Figure 3. Comparison of reconstruction error (left) and misclassification rate (right) on AR

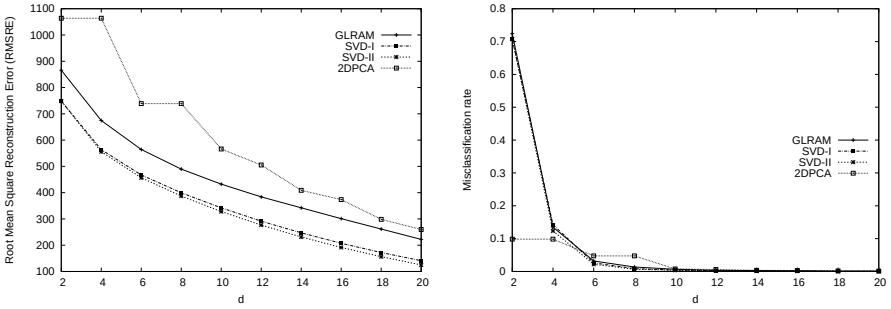

Figure 4. Comparison of reconstruction error (left) and misclassification rate (right) on PIE

– SVD-II has the smallest RMSRE value in all cases, while 2DPCA has the largest RMSRE value in most cases. In Section 2, SVD is shown to achieve the minimum reconstruction error among all rank-k approximations, which guarantees the best performance of SVD-II among all competing algorithms, when using the same number of reduced dimensions. The large reconstruction error of 2DPCA is due to its poor compression performance, when using only one-sided transformation, as compared to two-sided transformation in GLRAM.

– The performance of SVD-I is close to that of SVD-II for PIE and USPS. From Table III, $n \gg rc$ for both PIE and USPS. As discussed earlier, the reduced dimensions of SVD-I and SVD-II are close in this case. That is, SVD-I and SVD-II are essentially identical.

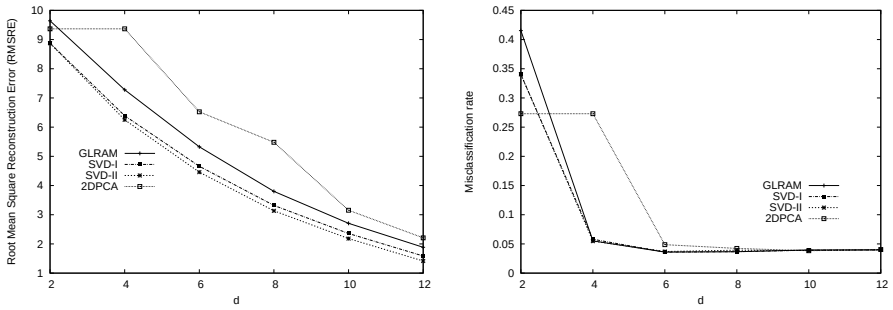

Figure 5. Comparison of reconstruction error (left) and misclassification rate (right) on USPS

– For PIX, ORL, and AR, GLRAM has smaller RMSRE value than SVD-I. This is mainly due to the fact that these three datasets have relatively large number of dimensions, compared to the number of data points. In this case, $k = \frac{d(r+c)+nd^2}{rc+n} < d^2$. For PIE and USPS, where the number of dimensions is relatively small, GLRAM has larger RMSRE value than SVD-I.

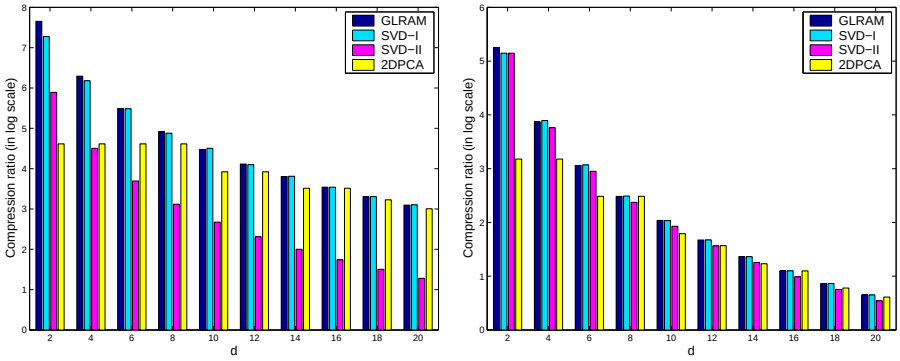

Figure 6. Comparison of compression ratio (on a log scale) on AR (left) and PIE (right)

– For datasets with relatively large number of dimensions compared to the number of data points, such as the AR datasets, the compression ratio of SVD-II is much smaller than others as shown in Figure 6 (left graph). As the number of data points get large, such as the PIE dataset, the compression ratio of SVD-II becomes close to GLRAM, as shown in Figure 6 (right graph).

– GLRAM is competitive with SVD-II for classifi cation in most cases, even though GLRAM has larger RMSRE values. This may be due to

the fact that GLRAM is able to utilize the locality information intrinsic in the image, which leads to good classification performance. We apply GLRAM to datasets without any locality property, such as text documents and gene expression data, by reshaping each vector as a matrix. GLRAM performs quite poorly in both the reconstruction error and classification, compared to SVD.

– The reconstruction error and misclassification rate of AR are much higher than those of other image datasets. This may be related to the large within-class variances of AR, due to the presence of sun glasses and scarves, as mentioned in Section 5.1.

5.5. COMPRESSION EFFECTIVENESS

In this experiment, we examine the quality of the images compressed by the proposed algorithm and compare it with SVD and 2DPCA. Image compression is commonly applied as a pre-processing step for storage and transmission of large image data. There exists a tradeoff between quality of compressed images and compression ratio, as a high compression ratio usually leads to compressed images with poor quality.

Figure 7 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 the GLRAM algorithm with $d = 10$. The compression ratio is about 98. The images compressed by SVD-I using the same compression ratio are shown in the third row of Figure 7. The images compressed by SVD-II and 2DPCA with approximately the same number of reduced dimensions as GLRAM are shown in the fourth and fifth rows of Figure 7 respectively. It is clear that the images compressed by our proposed algorithm have slightly better visual quality than those compressed by 2DPCA and SVD-I, while the ones by SVD-II have the best visual quality. However, the compression ratio of SVD-II (3.85) is much smaller than that of GLRAM (98.0).

Figure 8 shows images of 10 different digits from the USPS dataset. $d = 5$ is used in GLRAM. The compression ratio is about 10. Different from ORL dataset, SVD-I performs as well as SVD-II. GLRAM, SVD-I, and SVD-II perform slightly better than 2DPCA. Furthermore, the compression ratio of SVD-II (9.4) is close to that of GLRAM (10.2). The different behavior between ORL and USPS is related to the fact that USPS has a relatively large number of points compared to its dimension, i.e., $n \gg rc$, as discussed in the last section.

Figure 7. First row: raw images from ORL dataset. Second row: images compressed by GLRAM. Third row: images compressed by SVD-I. Fourth row: images compressed by SVD-II. Fifth row: images compressed by 2DPCA.

Figure 8. First row: raw images from USPS dataset. Second row: images compressed by GLRAM. Third row: images compressed by SVD-I. Fourth row: images compressed by SVD-II. Fifth row: images compressed by 2DPCA.

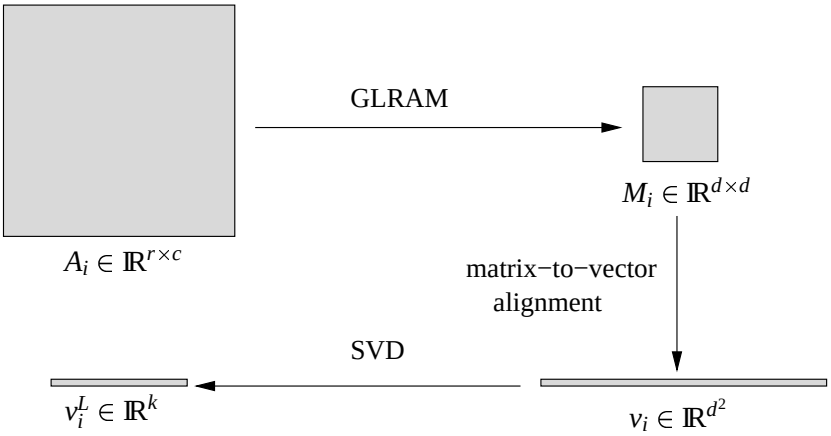

Figure 9. Flowchart of the GLRAM+SVD algorithm. It has two stages: In the first stage, each data point $A_i \in \mathbb{R}^{r\times c}$ is transformed to $M_i \in \mathbb{R}^{d\times d}$; In the second stage, M_i is first transformed to a vector $v_i \in \mathbb{R}^{d^2}$ by the so-called *matrix-to-vector alignment*, which is further reduced to a vector $v_i^L \in \mathbb{R}^k$ by SVD. Note that $d < \min(r,c)$ and $k < d^2$.

5.6. GLRAM+SVD

In this experiment, we study the combination of GLRAM and SVD, namely GLRAM+SVD, where the dimension by GLRAM is further reduced by SVD. More specifically, in the first stage, each data point $A_i \in \mathbb{R}^{r\times c}$ is reduced to $M_i \in \mathbb{R}^{d\times d}$ by GLRAM, with $d < \min(r,c)$. In the second stage, each M_i is first transformed to a vector $v_i \in \mathbb{R}^{d^2}$ by the *matrix-to-vector alignment*, where a matrix is transformed to a vector by concatenating all its rows together consecutively. Then v_i is further reduced to $v_i^L \in \mathbb{R}^k$ by SVD with $k < d^2$. The flowchart of the GLRAM+SVD algorithm is shown in Figure 9 graphically.

The complexity of the first (GLRAM) stage is $O(I(r+c)^2dn)$, where the number of iterations I is usually small. The second stage applies SVD to a n by d^2 matrix, hence takes $O(nd^2\min(n,d^2))$. Therefore, the total time complexity of GLRAM+SVD is $O(nd((r+c)^2+\min(nd,d^3)))$. Assuming $r \approx c \approx \sqrt{N}$, the time complexity can be simplified as $O(nd(N+\min(nd,d^3)))$. It is clear that both the GLRAM and SVD stages in GLRAM+SVD have much smaller computational costs than SVD, especially when d is small.

We apply GLRAM+SVD to the image datasets and compare it with SVD and GLRAM in terms of the reconstruction error and the computational time, when using the same number of reduced dimensions. For simplicity, we use $k = 100$ in SVD for PIX, ORL and AR and $k = 25$ for PIE and USPS. Hence, the reduced dimension by GLRAM and GLRAM+SVD is 100 (or 25). The value of d determines the intermediate dimension of the GLRAM stage in GLRAM+SVD. We examine the effect of d on the performance of

GLRAM+SVD, and the results are summarized in Figures 10–14, where the *x*-axis denotes the value of *d* (between 15 and 40 for ORL and AR, between 6 and 16 for PIE, and between 6 and 12 for USPS) in GLRAM+SVD and the *y*-axis denotes the reconstruction error, measured by the RMSRE value (left graph) and the computational time, measured in seconds (right graph). It is worthwhile to note that the reduced dimension of GLRAM is fixed in the comparison, while the reduced dimension of the intermediate (GLRAM) stage in GLRAM+SVD varies.

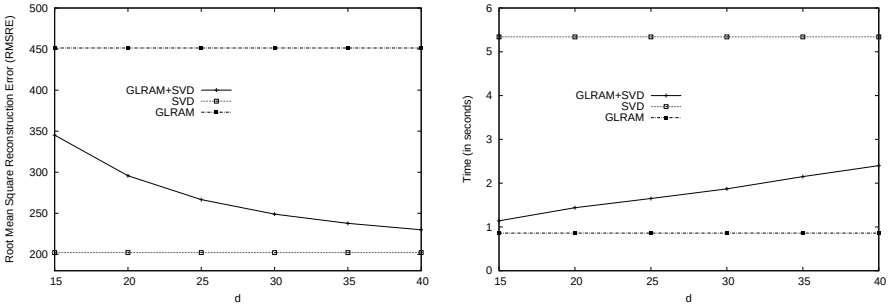

Figure 10. Comparison of reconstruction error (left) and computational time (right) on PIX

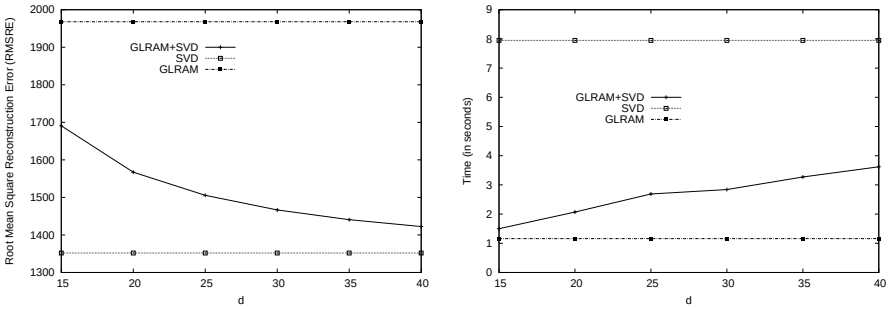

Figure 11. Comparison of reconstruction error (left) and computational time (right) on ORL

The main observations include:

- As *d* increases, the RMSRE value of GLRAM+SVD decreases. By combining GLRAM and SVD, GLRAM+SVD achieves a dramatic reduction of the RMSRE value as compared to GLRAM.
- The computational time of GLRAM+SVD increases, as *d* increases. There is a tradeoff between the reconstruction error and the computational time, when choosing the best *d*.

The above experiment shows that it may be beneficial to combine GLRAM with SVD, since it has much lower reconstruction error than GLRAM, while keeping the computational cost low.

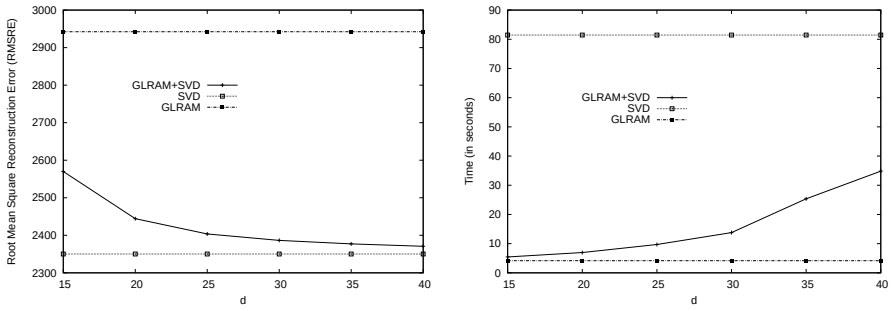

Figure 12. Comparison of reconstruction error (left) and computational time (right) on AR

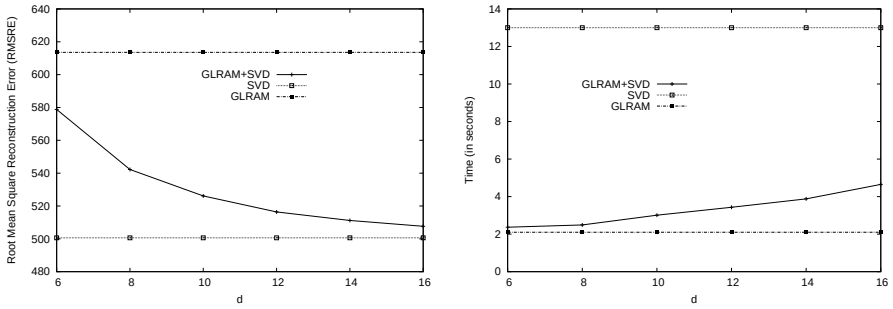

Figure 13. Comparison of reconstruction error (left) and computational time (right) on PIE

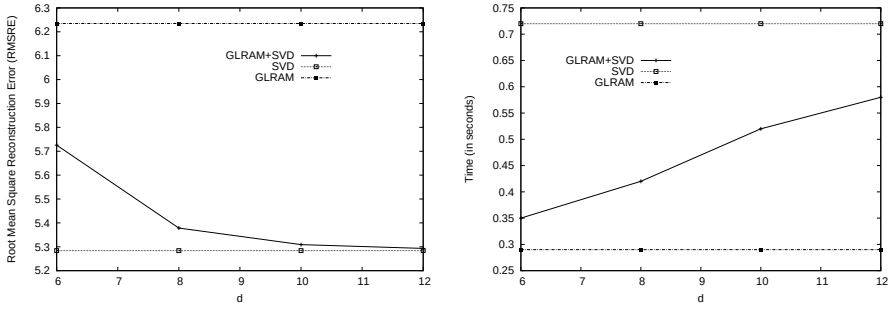

Figure 14. Comparison of reconstruction error (left) and computational time (right) on USPS

6. Conclusions and future work

A novel algorithm, namely GLRAM, for low rank approximations of a collection of matrices is presented. The algorithm works iteratively and successively improves the approximation during the iterations. Experimental results show that the algorithm converges within few iterations. Detailed analysis shows that the proposed algorithm has asymptotically minimum space requirement and lower time complexity than SVD, which is desirable for large and high-dimensional datasets.

A natural application for GLRAM is in image compression and retrieval, where each image is represented in its native matrix form. We evaluate the proposed algorithm in terms of the reconstruction error and classification, and compare it with 2DPCA and SVD. Experimental results show that when using the same number of reduced dimensions, the proposed algorithm is competitive with 2DPCA and SVD for classification, while GLRAM results in a higher reconstruction error than SVD. In terms of compression ratio, GLRAM outperforms SVD, especially when the number of dimensions is relatively large compared to the number of data points.

To further reduce the reconstruction error of GLRAM, we study the GLRAM+SVD algorithm, where SVD is preceded by GLRAM. In this composite algorithm, GLRAM can be considered as a pre-processing step for SVD. Extensive experiments show that, when using the same number of reduced dimensions, GLRAM+SVD achieves a significant reduction of reconstruction error as compared to GLRAM, while keeping the computational cost small. The reconstruction error of GLRAM+SVD is close to that of SVD, especially when the intermediate reduced dimension in the GLRAM stage is large, while it has a smaller computational cost than SVD.

There are several crucial questions that still remain to be answered:

- In Section 5.2, we study the effect of the ratio of ℓ_1 to ℓ_2 on reconstruction error. Experimental results show that choosing $\frac{\ell_1}{\ell_2} \approx 1$ works well in practice, even when the original row (r) and column (c) dimensions are quite different. It may be related to the effect of balancing between the left and right transformations involved in GLRAM. However, a rigorous theoretical justification behind this is still not available.

- In Section 5.3, we study the convergence property of GLRAM and the sensitivity of GLRAM to the choice of the initial L_0. Extensive experiments show that for image datasets, GLRAM may converge to the global solution, regardless of the choice of the initial L_0. However, it is not true in general, as shown in the RAND dataset. The remaining question is whether there exist certain conditions on the A_i's, under which GLRAM has the global convergence property.

Acknowledgment

We thank the Associate Editor and the reviewers for helpful comments that greatly improved the paper. We also thank Prof. Ravi Janardan and Dr. Chris Ding for helpful discussions.

This 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. Support Fellowships from Guidant Corporation and from the Department of Computer Science & Engineering, at the University of Minnesota, Twin Cities is gratefully acknowledged.

References

Achlioptas, D. and F. McSherry: 2001, 'Fast computation of low rank matrix approximations'. In: *ACM STOC Conference Proceedings*. pp. 611–618.

Aggarwal, C. C.: 2001, 'On the effects of dimensionality reduction on high dimensional similarity search'. In: *ACM PODS Conference Proceedings*. pp. 256 – 266.

Averbuch, A., D. Lazar, and M. Israeli: 1996, 'Image compression using wavelet transform and multiresolution decomposition'. *IEEE Transactions on Image Processing* **5**(1), 4–15.

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

Bjork, A. and G. Golub: 1973, 'Numerical methods for computing angles between linear subspaces'. *Mathematics of Computation* **27**(123), 579–594.

Brand, M.: 2002, 'Incremental Singular Value Decomposition of Uncertain Data with Missing Values'. In: *Proceedings of the Seventh European Conference on Computer Vision*. pp. 707–720.

Castelli, V., A. Thomasian, and C.-S. Li: 2003, 'CSVD: Clustering and Singular Value Decomposition for Approximate Similarity Searches in High Dimensional Space'. *IEEE Transactions on Knowledge and Data Engineering* **15**(3), 671–685.

Deerwester, S., S. Dumais, G. Furnas, T. Landauer, and R. Harshman: 1990, 'Indexing by Latent Semantic Analysis'. *Journal of the Society for Information Science* **41**, 391–407.

Dhillon, I. and D. Modha: 2001, 'Concept Decompositions for Large Sparse Text Data using Clustering'. *Machine Learning* **42**, 143–175.

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

Duda, R., P. Hart, and D. Stork: 2000, *Pattern Classification*. Wiley.

Edelman, A., T. A. Arias, and S. T. Smith: 1998, 'The Geometry of Algorithms with Orthogonality Constraints'. *SIAM Journal on Matrix Analysis and Applications* **20**(2), 303–353.

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

Fukunaga, K.: 1990, *Introduction to Statistical Pattern Classification*. San Diego, California, USA: Academic Press.

Golub, G. H. and C. F. Van Loan: 1996, *Matrix Computations*. Baltimore, MD, USA: The Johns Hopkins University Press, third edition.

Gu, M. and S. C. Eisenstat: 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., D. Agrawal, A. E. Abbadi, and A. Singh: 1998, 'Dimensionality reduction for similarity searching in dynamic databases'. In: *ACM SIGMOD Conference Proceedings*. pp. 166–176.

Martinez, A. and R. Benavente: 1998, 'The AR face database'. Technical Report CVC Tech. Report No. 24.

Papadimitriou, C. H., H. Tamaki, P. Raghavan, and S. Vempala: 1998, 'Latent Semantic Indexing: A Probabilistic Analysis'. In: *Proceedings of the Seventeenth ACM Conference on Principles of Database Systems*. pp. 159–168.

Samaria, F. and A. Harter: 1994, 'Parameterisation of a Stochastic Model for Human Face Identification'. In: *Proceedings of 2nd IEEE Workshop on Applications of Computer Vision, Sarasota FL*. pp. 138–142.

Sim, T., S. Baker, and M. Bsat: 2004, 'The CMU pose, illumination, and expression (PIE) database'. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **25**(12), 1615 – 1618.

Srebro, N. and T. Jaakkola: 2003, 'Weighted low-rank approximations'. In: *ICML Conference Proceedings*. pp. 720–727.

Yang, J., D. Zhang, A. Frangi, and J. Yang: 2004, 'Two-dimensional PCA: a new approach to appearance-based face representation and recognition'. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **26**(1), 131–137.

View publication stats