Fast Singular Value Decomposition for Large-scale Growing Data

Jengnan Tzeng
Department of Mathematical Sciences, National Chengchi University
Taipei, Taiwan
jengnan@math.nccu.edu.tw

***Abstract*—Singular value decomposition (SVD) is a fundamental technique in linear algebra, and it is widely applied in many modern information technologies, for example, high dimensional data visualization, dimension reduction, data mining, latent semantic analysis, etc. However, when the matrix size of the data is huge and continuously growing, the matrix can not be loaded all at once into the computer memory and $O(n^3)$ computational cost of SVD becomes infeasible. To resolve this problem, we will adapt a fast multidimensional scaling method to obtain a fast SVD method, given that the significant rank of a huge matrix is small. This proposed fast SVD method can be easily implemented via parallel computing. We also propose a fast update method to be applied when the huge data is updated continuously. We will demonstrate that the approximated SVD result is sufficiently accurate, and most importantly it can be derived very efficiently. Using this fast update method, many modern techniques based on SVD which were infeasible will become viable.**

Keywords-Singular value decomposition; multidimensional scaling; parallel computing; huge matrix.

I. INTRODUCTION

Singular value decomposition (SVD) and Principle component analysis (PCA) are two fundamental techniques in linear algebra and statistics. There are many modern applications based on these two tools, such as linear discriminate analysis [1], multidimensional scaling analysis [2], feature extraction, high dimensional data visualization, etc. In recent years, digital information has been proliferating and many analytic methods based on PCA and SVD are facing the challenge of their significant computational cost. Thus, it is crucial to develop a fast method of PCA and SVD.

In 2008, Tzeng et al. [4] developed a fast multidimensional scaling (MDS) method which turned the classical $O(n^3)$ MDS method to be linear. MDS is a method to represent the high dimensional data into the low dimensional configuration. Because of the phenomenon of the curse of dimensionality, MDS is widely used in data mining, clustering, and many recommendation systems for web services. When the data configuration is Euclidean, MDS is similar to principle component analysis (PCA), in that both can remove inherent noise with its compact representation of data. The $O(n^3)$ computational complexity makes it infeasible to apply to huge data, for example, when the sample size is more than one million.

In the following section, we will show how to adapt classical MDS to be a fast Split-and-combine MDS (SCMDS). And using this SCMDS, we can modify the PCA and SVD method to become fast methods.

II. METHODOLOGY

In 2008, we adapted the classical MDS so as to reduce the original $O(n^3)$ complexity to $O(n)$ [4], in which we have proved that when the data dimension is significantly smaller than the number of data entries, there is a fast linear method for classical MDS. The following section begins with a review of SCMDS. Then we will demonstrate how to adapt SCMDS method to become the fast PCA, and with further modification, the fast PCA can become the fast SVD.

A. From MDS to SCMDS

The main idea of fast MDS is using statistical resampling to split data into overlapping subsets. We perform the classical MDS on each subset and get the compact Euclidean configuration. Then we use the overlapping information to combine each configuration of subsets to recover the configuration of the whole data. Hence, we named this fast MDS method by Split-and-combine MDS (SCMDS).

Assume X_1 and X_2 are matrices in which the columns are the two coordinates of the overlapping points obtained by applying MDS to two grouped data sets. Then there exists an affine mapping that maps X_1 to X_2. Let $\bar{X}_1$ and $\bar{X}_2$ be the means of columns of X_1 and X_2, respectively. In order to obtain the affine mapping, we apply QR factorization to both $X_1 - \bar{X}_1\dot{1}^T$ and $X_2 - \bar{X}_2\dot{1}^T$. Then we have $X_1 - \bar{X}_1\dot{1}^T = Q_1R_1$ and $X_2 - \bar{X}_2\dot{1}^T = Q_2R_2$. It is clear that the mean of the center of column vectors of $X_1 - \bar{X}_1\dot{1}^T$ has been shifted to zero. Because X_1 and X_2 come from the same data set, the difference between $X_1 - \bar{X}_1\dot{1}^T$ and $X_2 - \bar{X}_2\dot{1}^T$ is a rotation. Therefor, the triangular matrices R_1 and R_2 should be identical when there is no noise in X_1 and X_2. Due to randomness of the sign of columns of Q_i in QR factorization, the sign of columns of Q_i might need to be adjusted according to the corresponding diagonal elements of R_i, so that the signs of tridiagonal elements of R_1 and R_2 are the same.

After necessary modification to the sign of columns of Q_i, we conclude that

$$Q_1^T(X_1 - \bar{X}_1\dot{1}^T) = Q_2^T(X_2 - \bar{X}_2\dot{1}^T).$$

 ISBN: 978-1-61208-116-8

Furthermore, we have

$$X_1 = Q_1 Q_2^T X_2 - Q_1 Q_2^T (\bar{X}_2 \dot{1}^T) + \bar{X}_1 \dot{1}^T.$$

Here, the unitary operator is $U = Q_1 Q_2^T$ and the shifting is $b = -Q_1 Q_2^T \bar{X}_2 + \bar{X}_1$. Since the key processing of finding this affine mapping is QR decomposition, the computational cost is $O(k^3)$, where k is the number of columns of X_1 and X_2. Therefore, the cost $O(k^3)$ complexity is limited by the number of samples in each overlapping region. The proof of the computational cost of SCMDS is given as follows:

Assume that there are N points in a data set, we divide these N samples into K overlapping subgroups, where N_G is the number of points in each subgroup and N_I the number of points in each intersection region. Then we have the relationship

$$KN_G - (K-1)N_I = N$$

or

$$K = \frac{(N - N_I)}{(N_G - N_I)}.$$

For each subgroup, we apply classical MDS to compute the configuration of each group data, which cost $O(N_G^3)$. In each overlapping region, we apply QR factorization to compute the affine transformation, which cost $O(N_I^3)$. Assume that the true data dimension is p, and the lower bound of N_I is $p+1$. For the convenience, we take $N_G = \alpha p$ for some constant $\alpha > 2$. Then the total computational cost is about

$$\frac{N-p}{(\alpha-1)p} O(\alpha^3 p^3) + \frac{N - \alpha p}{(\alpha - 1)p} O(p^3) \approx O(p^2 N).$$

The first term of above equation is the complexity of MDS in K groups, and the second term is the complexity of QR in the $K-1$ overlapping regions.

When $p << N$, the computational cost $O(p^2 N)$ is much smaller than $O(\sqrt{N} N)$, which is the computation time of the fast MDS method proposed by Morrison et al., 2003 [3]. The key idea of our fast MDS method is to split data into subgroups, then combine the configurations to recover the whole one. Since all the order three complexities are restricted in the small number of data entries, we can therefor speed up MDS. The concept of split-and-combine is also similar to the concept of parallel computing. Thus, SCMDS method can be easily implemented via a parallel algorithm.

B. From SCMDS to SCPCA

Because MDS is similar to principle component analysis (PCA) when the data configuration is Euclidean, we can adapt SCMDS method to obtain the fast PCA in the same constrain $p << N$.

Assume that X is a p-by-N matrix, where there are N samples with p dimension. $D = X^T X$ indicates product matrix of X, and $\dot{1}$ is an N-by-1 vector whos elements are all 1's. We define a symmetric matrix B by

$$\begin{aligned} B &= \left(X - \frac{1}{N} X \dot{1}\dot{1}^T\right)^T \left(X - \frac{1}{N} X \dot{1}\dot{1}^T\right) \\ &= D - \frac{1}{N} D \dot{1}\dot{1}^T - \frac{1}{N} \dot{1}\dot{1}^T D + \frac{1}{N^2} \dot{1}\dot{1}^T D \dot{1}\dot{1}^T \\ &= D - \bar{D}_r - \bar{D}_c + \bar{D}_g, \end{aligned}$$

where $\bar{D}_r = \frac{1}{N} D \dot{1}\dot{1}^T$ is the row mean matrix of D, $\bar{D}_c = \frac{1}{N} \dot{1}\dot{1}^T D$ is the column mean matrix of D and $\bar{D}_g = \frac{1}{N^2} \dot{1}\dot{1}^T D \dot{1}\dot{1}^T$ is the ground mean matrix of D. The operator from D to $D - \bar{D}_r - \bar{D}_c + \bar{D}_g$ is called double centering. If we define a matrix H by

$$H = I - \frac{1}{N} \dot{1}\dot{1}^T,$$

B can be simplified to $B = HDH$. Since matrix B is symmetric, the SVD decomposes B into $B = Z \Sigma Z^T$. Then we have

$$\sqrt{B} = Z \Sigma^{\frac{1}{2}} P^T = (X - \frac{1}{N} X \dot{1}\dot{1}^T)^T,$$

for some unitary matrix P. In practice, we set $P = I$ to obtain the MDS result $\sqrt{B}$. Therefore, the row vector of $\sqrt{B}$ is the coordinates of X with the mean of data been shifted to the original point and rotated by some unitary matrix P.

If D is a distance matrix with each element $d_{i,j} = \sqrt{(x_i - x_j)^T (x_i - x_j)}$, the double center of D^2 is equivalent to $-2B$, provided that $\sum_{i=1}^N x_i = 0$. Hence, the MDS method performs double centering on D^2, multiplies by $-\frac{1}{2}$, and then performs SVD, which gives the configurations of the data set.

The constrain $\sum_{i=1}^N x_i = 0$ in MDS is the same as the constrain in computing PCA. The score matrix P of PCA is an unitary matrix, and it is derived by

$$(X - \frac{1}{N} X \dot{1}\dot{1}^T)(X - \frac{1}{N} X \dot{1}\dot{1}^T)^T = P \Sigma P^T.$$

If we have the unitary matrix P, we can use $(X - \frac{1}{N} X \dot{1}\dot{1}^T)^T P$ to obtain $\sqrt{B}$. The result of MDS ($\sqrt{B}$) simply uses the first r orthogonal columns of this unitary matrix to represent the data. We define an orthogonal matrix Z_r by the first r columns of Z and the sub-diagonal matrix Σ_r by the up-left block of Σ, then $\sqrt{B_r} = Z_r \Sigma_r^{\frac{1}{2}}$ is the r-dimensional configuration of data set. If we have r-dimensional MDS configuration, we can obtain the first r columns of the score matrix of PCA, denoted by P_r. That is

$$P_r = (X - \frac{1}{N} X \dot{1}\dot{1}^T) Z (\Sigma_r^{\frac{1}{2}})^{-1}.$$

Thus, the key scheme of PCA is embedded in the MDS method. When the number of samples is large and the data set is high dimensional, the complexity is costly. However, if there are many samples which are linearly dependent, the actual rank of the data matrix is much smaller than

 ISBN: 978-1-61208-116-8

the matrix size. In this case, SCMDS has advantage in computing speed. And the approach of obtaining P_r by SCMDS is called SCPCA.

C. From SCPCA to SCSVD

The concept of SVD and PCA are very similar. Since the PCA starts from decomposing the covariance matrix of data set, it can be considered as adjusting the center of mass of a row vector to zero. On the other hand, SVD operates directly on the product matrix without shifting. If the mean of the matrix rows is zero, the eigenvectors derived by SVD are equal to the eigenvectors derived by the PCA. We are looking for a method which will give a fast method to produce the SVD result without recomputing the eigenvectors of the whole data set, when the PCA result is given. The following is the mathematical analysis for this process.

Let X be a column matrix of data set. $\tilde{X} = X - \bar{X} \cdot \dot{1}^T$, where $\bar{X}$ is the mean of columns of X. Hence, the row mean of $\tilde{X}$ is zero. Assume that we have the PCA result of X, that is, $\tilde{X}\tilde{X}^T = P\Sigma^2 P^T$. Then we have $\tilde{X} = P\Sigma U^T$ for some orthogonal matrix U. Assume that the rank of $\tilde{X}$ is r and r is much smaller than the matrix size, we observe that $rank(X) = r$ or $rank(X) = r + 1$, depending on whether $\bar{X}$ is spanned by P. If $\bar{X}$ is spanned by P, then

$$X = \tilde{X} + \bar{X} \cdot \dot{1}^T = P\Sigma U^T + P \cdot c \cdot \dot{1}^T = P(\Sigma U^T + c \cdot \dot{1}^T),$$

where c is the coefficient vector of $\bar{X}$ when represented by P, *i.e.*, $\bar{X} = P \cdot c$.

If the singular value decomposition of $\Sigma U^T + c \cdot \dot{1}^T$ is $W\hat{\Sigma}V^T$, we have

$$X = P(W\hat{\Sigma}V^T) = (PW)\hat{\Sigma}V^T = Z\hat{\Sigma}V^T.$$

Because the matrix W is unitary, $Z = PW$ is automatically an orthogonal matrix as well. Then we have the SVD of X.

Checking the matrix size of $\Sigma U^T + c \cdot \dot{1}^T$, we can see that to compute the SVD of $\Sigma U^T + c \cdot \dot{1}^T$ is not a big task. This is because $\Sigma U^T + c \cdot \dot{1}^T$ is a r-by-n matrix, and under our assumption, r is much smaller than n, so we can apply the economic SVD to obtain the decomposition of $\Sigma U^T + c \cdot \dot{1}^T$.

On the other hand, if $\bar{X}$ is not spanned by P, the analysis becomes

$$X = \tilde{X} + \bar{X} \cdot \dot{1}^T = [P|p_{r+1}] \left(\begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U^T \\ 0 \end{bmatrix} + c \cdot \dot{1}^T \right),$$

where p_{r+1} is a unit vector defined by

$$p_{r+1} = \frac{(I - PP^T)\bar{X}}{\|(I - PP^T)\bar{X}\|}.$$

Using the same concept of diagonalization in the case when $\bar{X}$ is spanned by P, we find the SVD of

$$\left(\begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U^T \\ 0 \end{bmatrix} + c \cdot \dot{1}^T \right) = W\hat{\Sigma}V^T.$$

Then $X = [P|p_{r+1}]W\hat{\Sigma}V^T = Z\hat{\Sigma}V^T$, where $Z = [P|p_{r+1}]W$ is another orthogonal matrix and hence the SVD of X is completed.

From the above analysis, we can have a fast PCA method by computing SCMDS first, then adapt the MDS result to obtain PCA. We named this approach SCPCA. Similarly, the fast SVD method which computes SCMDS first, then adapts MDS result to obtain PCA, and finally adapts PCA result to SVD, is called the SCSVD. These two new methods work when the rank of X is much smaller than the number of samples and the number of variables. To obtain the exact solution, the parameter N_I must be greater than the rank of X. In SCPCA or SCMDS method, if $N_i \leq r$, we only get the approximated solution of PCA and SVD. Under the necessary criterion, we can reduce the computational complexity from $\min\{O(p^2n), O(pn^2)\}$ to $\min\{O(rp), O(rn)\}$. If the significant rank is not small, for example,$r \approx \sqrt{pn}$, the computational complexity becomes almost the same as the original PCA and SVD. Our method has no advantage in the latter case.

III. SVD FOR CONTINUOUSLY GROWING DATA

In this section, we look for the solution when the data is updated constantly and we need to compute SVD continuously. Instead of scanning all the data again, we try to use the previous SVD result together with the new updated data to compute the next SVD.

Let A be an m-by-n matrix, where m is the number of variables and n is the number of samples. And we assume that both m and n are huge. When new data comes in, we collect these new data to form a column matrix which is denoted by B. Assume that we have the singular value decomposition of A, that is

$$A = Z\Sigma V^T,$$

where $Z \in M_m(\Re)$, $V \in M_n(\Re)$ are orthogonal and Σ is a diagonal. Since the data gets updated, the data matrix becomes

$$A_1 = [A|B].$$

To compute the singular value decomposition of A_1, we need to compute the eigenvalue and eigenvector of $A_1 A_1^T$.

We can represent the column matrix B by $B = ZC$, where C is the coefficient matrix of B with columns of Z as the basis. Since Z is orthogonal, the coefficient matrix C can be computed easily by $C = Z^T B$. Then we have

$$\begin{aligned} A_1 A_1^T &= [A|ZC]\,[A|ZC]^T \\ &= AA^T + ZC(ZC)^T \\ &= Z(\Sigma^2 + CC^T)Z^T \\ &= ZU\hat{\Sigma}^2 U^T Z^T \\ &= Z_1 \hat{\Sigma}^2 Z_1^T. \end{aligned} \quad (1)$$

Note that the matrix $\Sigma^2 + CC^T$ is positive symmetric. Using the spectrum theorem, we can decompose this matrix

into $U\hat{\Sigma}^2U^T$. Because the matrix U is unitary, $Z1$ is Z rotated by U.

When the matrix size of A is huge, the computational cost of SVD is high. If the data is constantly growing, it is difficult to compute the singular value decomposition of A_1 in real time. Therefore, we look for an approximated solution with fast method.

Let $Z = [z_1, z_2, \cdots, z_m]$. If the new updated data B has only the components in $\{z_1, z_2, \cdots, z_r\}$, where $r << m$, then only r-dimensional space will be perturbed by this new data. This is proved as follows.

Theorem Let $A = Z\Sigma V^T$. Assume that $A_1 = [A|B]$, where B has no component in i-th column of Z for $i > r$. Then the singular value decomposition of A_1 has the same spectrum σ_i and singular vector z_i, v_i for $i > r$.

Proof: Let

$$A = \begin{array}{c} \begin{array}{cc} r & p-r \end{array} \\ \begin{pmatrix} Z_1 & Z_2 \end{pmatrix} \end{array} \quad \begin{array}{c} \begin{array}{cc} r & p-r \end{array} \\ \begin{pmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{pmatrix} \end{array} \quad \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix},$$

Where Z_1 and V_1 are the first r columns of Z and V. Because B has no component in Z_2, $B = Z_1C$ for some C. Then $A_1A_1^T$ can be written as

$$\begin{aligned} A_1A_1^T &= [A|B][A|B]^T \\ &= AA^T + UU^T \\ &= \begin{pmatrix} Z_1 & Z_2 \end{pmatrix} \begin{pmatrix} \Sigma_1^2 + CC^T & 0 \\ 0 & \Sigma_2^2 \end{pmatrix} \begin{pmatrix} Z_1^T \\ Z_2^T \end{pmatrix} \\ &= \begin{pmatrix} Z_1 & Z_2 \end{pmatrix} \begin{pmatrix} U\hat{\Sigma}_1U^T & 0 \\ 0 & \Sigma_2^2 \end{pmatrix} \begin{pmatrix} Z_1^T \\ Z_2^T \end{pmatrix} \\ &= \begin{pmatrix} Z_1U & Z_2 \end{pmatrix} \begin{pmatrix} \hat{\Sigma}_1^2 & 0 \\ 0 & \Sigma_2^2 \end{pmatrix} \begin{pmatrix} U^TZ_1^T \\ Z_2^T \end{pmatrix} \\ &= \begin{pmatrix} \hat{Z}_1 & Z_2 \end{pmatrix} \begin{pmatrix} \hat{\Sigma}_1^2 & 0 \\ 0 & \Sigma_2^2 \end{pmatrix} \begin{pmatrix} \hat{Z}_1^T \\ Z_2^T \end{pmatrix} \end{aligned} \quad (2)$$

where U is unitary. We can see that the terms Σ_2 and Z_2 do not change if $Z_2^TB = 0$. Thus, the singular value decomposition of A_1 has the same spectrum σ_i and z_i for $i > r$. Moreover, the v_i for $i > r$ are unchanged too. ■

In many applications, we are only concerned with the first few spectrums and eigenvectors. For example, in high dimensional data visualization, we only consider the first two or three eigenvectors, that is $r = 2$ or 3. In this case, the new updated data should be have component in the space that spanned byZ_2. We are therefore interested in the performance of the approximated solution compared with the true solution when we only retained the first r components of B every time we updated the new data by the previous method. If the performance decays slowly, we can daringly compute only three or four components in many SVD-based methods and the result in low dimensional space is still quite reliable. So, we need to understand in what kind of conditions, the approximated solution is stable.

Now, we analyze the effect of the perturbation of a matrix in its eigenvalues and eigenvectors. Let matrix A be real symmetric and $A = S\Lambda S^T$, where S is unitary, such that $S^{-1} = S^T$. A matrix change ΔA produces changes in eigenvalues and eigenvectors, which are denoted by $\Delta\Lambda$ and ΔS respectively. Because S is orthogonal, $AS = S\Lambda$. Similarly, we have

$$(A + \Delta A)(S + \Delta S) = (S + \Delta S)(\Lambda + \Delta\Lambda).$$

The above equation can be represented by

$$A(\Delta S) + (\Delta A)S = S(\Delta\Lambda) + (\Delta S)\Lambda, \quad (3)$$

when ignoring the small terms $(\Delta A)(\Delta S)$ and $(\Delta S)(\Delta\Lambda)$.

We multiply equation (3) by S^T, then we have

$$\begin{aligned} \Delta\Lambda &= S^T(\Delta A)S + S^TA(\Delta S) - S^T(\Delta S)\Lambda \\ &= S^T(\Delta A)S + \Lambda S^T(\Delta S) - S^T(\Delta S)\Lambda. \end{aligned} \quad (4)$$

Because the diagonal terms of $\Lambda S^T(\Delta S)$ and $S^T(\Delta S)\Lambda$ are the same, the diagonal part of $S^T(\Delta A)S$ is what we are looking for. Applying this concept to matrix $\Sigma^2 + CC^T$, CC^T can be considered as $S^T(\Delta A)S$. We can conclude that if the maximal element of the absolute value of CC^T is smaller than the difference between $\sigma_i - \sigma_{i+1}$ for $i = 1, \cdots, r$, then the order of columns of S will not change. The first r columns of $S + \Delta S$ can be approximated stably by the first r columns of S. If CC^T is too large such that the new spectrum $\hat{\sigma_{r+1}} > \hat{\sigma_r}$, the approximation solution that only use first r components to update the new spectrum and singular vectors will fault by using $\hat{z}_{r+1}$ to replace $\hat{z}_r$. This conclusion will be demonstrated in the experimental result.

IV. Experimental result

In this section, we show that our fast PCA and SVD method works well for big sized matrix with small rank. The simulated matrix is created by the product of two slender matrices. The size of the first matrix is p-by-r, and the second matrix is r-by-n. Then the product of these two matrixes is of size p-by-n and its rank is smaller than r. When p and n are large and r is much smaller than p and n, the simulated matrix satisfies our SCSVD condition. We pick $p = 4000$, $n = 4000$ and $r = 50$ as our fist example. The elements of the simulated matrix is generated from the normal distribution $\mathcal{N}(0, 1)$.

The average elapsed time of SCSVD is 3.98 seconds, while the economical SVD takes 16.14 seconds, If we increase the matrix to $p = 20000$, $n = 20000$ and the same rank $r = 50$, the elapsed time of economical SVD is 1209.92 seconds, but SCSVD is only 195.85 seconds. We observe that our SCSVD method demonstrates significant improvement.

Note that when the estimated rank used in SCSVD is greater than the real rank of data matrix, there is almost no error (except rounding error) between economic SVD and SCSVD. Figure 1 shows the speed comparison between economical SVD (solid line) and SCSVD (dashed line) with square matrix size from 500 to 4000 by fixed rank 50.

 ISBN: 978-1-61208-116-8

We also use fixed parameter $N_I = 51$ and $N_g = 2N_I$ in each simulation test. We can see that the computational cost of SVD follows the order 3 increase, compared with linear increase of SCSVD. The error between economical SVD and economical SVD, and that between SVD and SCSVD are shown in Figure 2. Because the results between economical SVD and SVD are very similar, we use solid line to represent the value of economical SVD and circle plot to represent SCSVD. The values in both Figure 1 and Figure 2 are the mean of the results from 100 repeated simulated matrices. The errors between SVD and economic SVD, and that between SVD and SCSVD are all under the 10^{-4} level. Thus, when the estimated rank of SCSVD is greater than the true rank, the accuracy of SCSVD is pretty much the same as SVD in the case of small rank matrix.

The purpose of the second simulation experiment is to observe the approximation performance of applying SCPCA to big full rank matrix. We generate random matrix with fixed number of columns and rows, say 1000. The square matrix is created by the form, $A_{p\times r} \cdot B_{r\times n} + \alpha E_{p\times n}$, where r is the essential rank, E is the perturbation and α is a small coefficient for adjusting the influence to the previous matrix. Such matrix can be considered as a big sized matrix with small rank added by a full rank perturbation matrix. We will show that our method works well for this type of matrices.

Figure 3 shows the error vs. estimated rank, where the error is computed by the difference between the original matrix and the composition of three matrices from SCSVD. All the elements of matrices A, B and E are randomly generated from the normal distribution $\mathcal{N}(0, 1)$, where $\alpha = 0.01$ and the essential rank $r = 50$. We can see that when the estimated rank increases, the composition error decreases. Especially when the estimated rank is greater than the essential rank r, the composition error decays rapidly. Thus, it is important to make sure that the estimated rank is greater than the essential rank. In other words, when the estimated rank of SCSVD is smaller than the essential rank, our SCSVD result can be used as the approximated solution of SVD.

In the last experimental result, we will show that we can set the estimated rank $r = 3$, starting from the SCSVD result and using the previous updating method to continuously update the new SVD. We will show that the performance of the first three components decays very slowly. Thus, many SVD-based modern techniques, for example, Fisher linear discrimination, Latent semantic analysis [5], eigentaste recommendation system [6], dimensional reduction, etc, become feasible even when dealing with huge data set.

We produce a series square random matrices A with size n-by-n for n between 1000 and 3000. Then we decompose A by SVD to obtain $A = Z\Sigma V^T$. We reset the diagonal terms of Σ to be exponential decay, so that the data can simulate the meaningful data in the real world. The maximal spectrum is set to be 10^4. Then we compose A by the new diagonal matrix V. We use SCSVD with estimated rank 3, and the parameter $N_I = \frac{n}{10}$, $N_g = 2N_I$.

We make 16 updates to the data, and each time we add 10% samples of original data. The new data is simulated from the normal distribution $\mathcal{N}(0, 1)$. We use our updating method to compute the first three new columns of Z and compare it with the true SVD result. Let $a^{(t)}$, $b^{(t)}$ be the maximal and minimal element of the absolute values of $\hat{Z}_3^{(t)T} Z_3^{(t)}$, respectively, where $\hat{Z}_3^{(t)}$ is the t-th updated Z by our updating method taking only the first three columns, and $Z^{(t)}$ is the t-th updated Z by normal SVD. If $a^{(t)}$ and $b^{(t)}$ are close to 1, the updated Z derived by our updating method is very close to the true Z. In Figure 4, we can see that both $a^{(t)}$ and $b^{t)}$ are close to 1, and they decay very slowly as the matrix size increases. In Figure 4, every point is the average value of 32 repeating simulations.

V. Conclusion and Future Work

We proposed fast PCA and SVD methods derived from the technique of SCMDS method. The new PCA and SVD have the same accuracy as the traditional PCA and SVD ones when the rank of a matrix is much smaller than its matrix size. The results of applying SCPCA and SCSVD to a full rank matrix are also quite reliable when the essential rank of the matrix is much smaller than its matrix size. In most information technology applications, the essential rank of a matrix is usually much smaller than its matrix size. In such cases, utilizing SCPCA or SCSVD in huge data applications will render good approximated results. Since the concept of split-and-combine is very similar to that of parallel computing, this SC-series methods (Split-and-combine series) can be easily implemented via parallel computing. Using our updating method for the growing data, we show that the approximated solution is very close to the actual solution, even when the estimated rank is as small as $r = 3$.

For the future work, we will focus on the cases when the data contains missing values. Our intuitive speculation is that the processing of splitting data should be somehow related to the locations where the missing values occur. We believe that it would be an interesting topic worth further exploration.

 ISBN: 978-1-61208-116-8

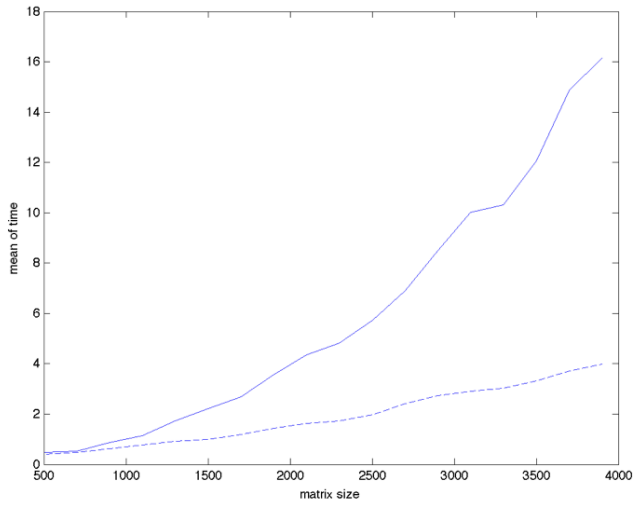

Figure 1. Comparison of the elapsed time between economical SVD (the solid line) and SCSVD (the dashed line).

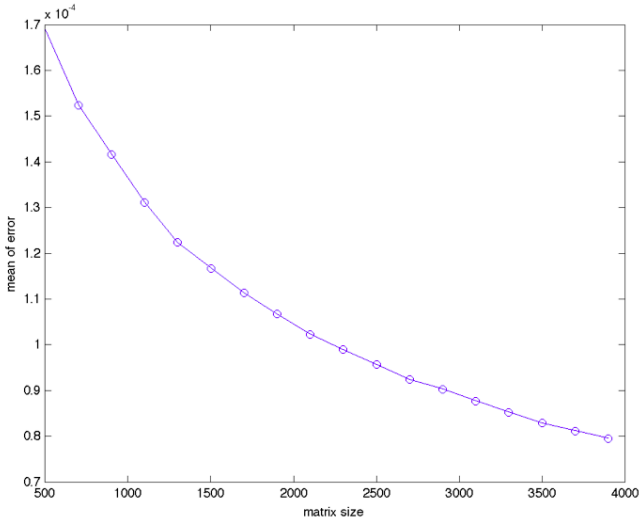

Figure 2. Comparison of the composition errors between economical SVD (the solid line) and SCSVD (the circle plot).

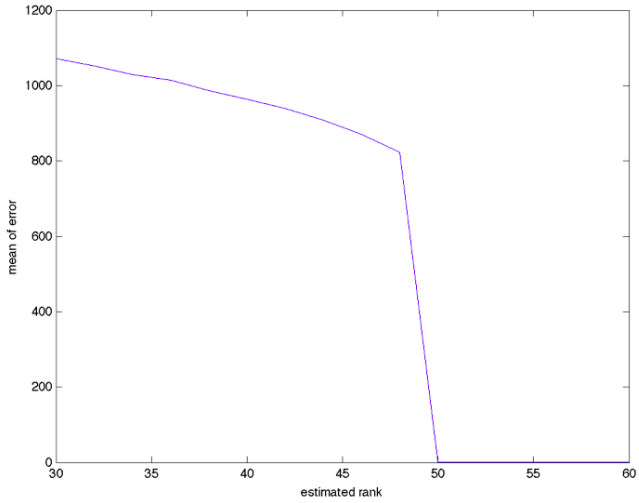

Figure 3. The effect of estimated rank to the composition error. The matrix size is 1000-by-1000 and its essential rank is 50 ($\alpha = 0.01$). When the estimated rank is greater than 50, there is almost no composition error

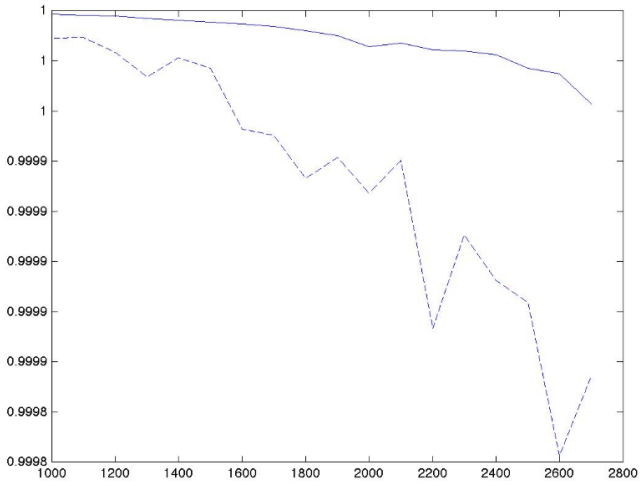

Figure 4. The orthogonality between approximated SVD and true SVD. The solid line is $a^{(t)}$ and the dashed line is $b^{(t)}$

REFERENCES

[1] D. J. Hand, "Discrimination and classification", Wiley Series in Probability and Mathematical Statistics, Chichester: Wiley, 1981

[2] M. Cox, T. Cox "Multidimensional scaling", Handbook of data visualization, Springer, 2008

[3] A. Morrison, G. Ross and M. Chalmers, "Fast multidimensional scaling through sampling, springs and interpolation",Information Visualization, Vol. 2 , Issue 1, pp. 68 - 77, 2003

[4] J. Tzeng, H. Lu and W. Li, "Multidimensional scaling for large genomic data sets", BMC Bioinformatics, 9:179, 2008

[5] G. W. Furnas, S. Deerwester, S. T. Dumais, T. K. Landauer, R.A. Harshman, L. A. Streeter and K. E. Lochbaum, "Information retrieval using a singular value decomposition model of latent semantic structure", Annual ACM conference on Research and Development in Information Retrieval, pp. 465-480, 1988

[6] K. Goldberg, T. Roeder, D. Gupta and C. Perkins, "Eigentaste: A constant time collaborative filtering method", Information Retrieval, Springer, 2001

 ISBN: 978-1-61208-116-8