Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 683053, 8 pages
http://dx.doi.org/10.1155/2013/683053

*Research Article*

# Split-and-Combine Singular Value Decomposition for Large-Scale Matrix

**Jengnan Tzeng**

*Department of Mathematical Sciences, National Chengchi University, No. 64, Section 2, ZhiNan Road, Wenshan District, Taipei City 11605, Taiwan*

Correspondence should be addressed to Jengnan Tzeng; jengnan@gmail.com

Received 16 November 2012; Revised 17 January 2013; Accepted 22 January 2013

Academic Editor: Nicola Mastronardi



The singular value decomposition (SVD) is a fundamental matrix decomposition in linear algebra. It is widely applied in many modern techniques, for example, high- dimensional data visualization, dimension reduction, data mining, latent semantic analysis, and so forth. Although the SVD plays an essential role in these fields, its apparent weakness is the order three computational cost. This order three computational cost makes many modern applications infeasible, especially when the scale of the data is huge and growing. Therefore, it is imperative to develop a fast SVD method in modern era. If the rank of matrix is much smaller than the matrix size, there are already some fast SVD approaches. In this paper, we focus on this case but with the additional condition that the data is considerably huge to be stored as a matrix form. We will demonstrate that this fast SVD result is sufficiently accurate, and most importantly it can be derived immediately. Using this fast method, many infeasible modern techniques based on the SVD will become viable.

## 1. Introduction

The singular value decomposition (SVD) and the principle component analysis (PCA) are fundamental 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], and feature extraction, high-dimensional data visualization. In recent years, digital information has been proliferating and many analytic methods based on the PCA and the SVD are facing the challenge of their significant computational cost. Thus, it is crucial to develop a fast approach to compute the PCA and the SVD.

Currently there are some well-known methods for computing the SVD. For example, the GR-SVD is a two-step method which performs Householder transformations to reduce the matrix to bidiagonal form then performs the $QR$ iteration to obtain the singular values [3, 4]. Since the off-diagonal regions are used to store the transform information, this approach is very efficient in saving the computational memory. If we only want to compute a few of the largest singular values and associated singular vectors of a large matrix, the Lanczos bidiagonalization is an important procedure for solving this problem [5–8]. However, the previously mentioned methods all require matrix multiplications for the SVD. One interesting problem is how do we compute the SVD for a matrix when the matrix size is huge and loading the whole matrix into the memory is not possible? The main purpose of this paper is to deal with this problem when the numerical rank of the huge matrix is small.

The second purpose of this paper is to update the SVD when the matrix size is extended by new data updating. If the rank of matrix is much smaller than the matrix size, Matthew proposed a fast SVD updating method for the low-rank matrix in 2006 [9]. A rank-$r$ thin SVD of an $m \times n$ matrix can be computed in $O(mnr)$ time for $r \leq \sqrt{\min(m, n)}$. Although the matrix size remains unchanged in the Matthew's method, we can still adopt the analysis of this paper for updating the SVD when the matrix size is changed.

Multidimensional scaling (MDS) is a method of representing the high-dimensional data into the low-dimensional configuration [10–12]. One of the key approaches of the MDS is simply the SVD, that is, if we can find a fast approach of

the MDS then it is possible to find a fast approach of the SVD. When the data configuration is Euclidean, the MDS is similar to the PCA, in that both can remove inherent noise with its compact representation of data. The order three computational complexity makes it infeasible to apply to huge data, for example, when the sample size is more than one million. In 2008, Tzeng et al. developed a fast multidimensional scaling method which turned the classical order three MDS method to be linear [13]. In this paper, we would like to implement SCMDS to the fast SVD approach, say SCSVD. The following subsections are reviews of the classical MDS and the SCSVD.

*1.1. Review of the Classical MDS.* Assume that $X$ is an $n$-by-$r$ matrix with rank $r$, where the $i$th row of $X$ represents the coordinates of the $i$th data point in an $r$-dimensional space. Let $D = XX^T$ be the product matrix of $X$. Torgerson [10] proposed the first classical MDS method to reconstruct a matrix $X$ of Cartesian coordinates of points from a given symmetric matrix $D$. The key of the classical MDS is to apply the double centering operator and the singular value decomposition.

Assume that $\dot{\imath}$ is an $n$-by-1 matrix whose elements are all 1's. We define a symmetric matrix $B$ by

$$\begin{aligned} B &= \left(X - \frac{1}{n}\dot{\imath}\dot{\imath}^T X\right)\left(X - \frac{1}{n}\dot{\imath}\dot{\imath}^T X\right)^T \\ &= XX^T - \frac{1}{n}XX^T\dot{\imath}\dot{\imath}^T - \frac{1}{n}\dot{\imath}\dot{\imath}^T XX^T \\ &\quad + \frac{1}{n^2}\dot{\imath}\dot{\imath}^T XX^T\dot{\imath}\dot{\imath}^T \\ &= D - \frac{1}{n}D\dot{\imath}\dot{\imath}^T - \frac{1}{n}\dot{\imath}\dot{\imath}^T D + \frac{1}{n^2}\dot{\imath}\dot{\imath}^T D\dot{\imath}\dot{\imath}^T \\ &= D - D_R - D_C + D_G, \end{aligned} \tag{1}$$

where $D_R = (1/n)D\dot{\imath}\dot{\imath}^T$ is the row mean matrix of $D$, $D_C = (1/n)\dot{\imath}\dot{\imath}^T D$ is the column mean matrix of $D$, and $D_G = (1/n^2)\dot{\imath}\dot{\imath}^T XX^T\dot{\imath}\dot{\imath}^T$ is the ground mean matrix of $D$. The operator from $D$ to $D - D_R - D_C + D_G$ is called double centering. If we define a matrix $H$ by

$$H = I - \frac{1}{n}\dot{\imath}\dot{\imath}^T, \tag{2}$$

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

$$\sqrt{B} = V\Sigma^{1/2}P^T = X - \frac{1}{n}\dot{\imath}\dot{\imath}^T X, \tag{3}$$

for some unitary matrix $P$. In practice, we set $P = I$ to obtain the MDS result, namely, $\sqrt{B}$.

Although $\sqrt{B} \neq X$, $\sqrt{B} = V\Sigma^{1/2} = (X - (1/n)\dot{\imath}\dot{\imath}^T X)P$ for some unitary matrix $P$, it is the same as shifting all the rows of $X$ such that the row's mean is zero and then multiplying the result by $P$. Hence, the row of $\sqrt{B}$ preserves the pairwise relationship among the data points. $B$ is derived from $D$ by double centering, whose cost is lower than obtaining $\sqrt{B}$ by the SVD. When the data size is huge, the cost of the classical MDS is dominated by the cost of the SVD, whose $O(n^3)$ computational complexity makes it infeasible. Therefore, we need a fast MDS method to deal with the huge data problem.

The MDS method is useful in the application of dimension reduction. If the matrix $X$ is extended to the higher dimensional space and then shifted and/or rotated by an affine mapping, the double centering result of the corresponding product matrix stays the same. More generally, let $X_1 \in M_{n,r_1}$ be derived from $X \in M_{n,r}$ by an affine mapping such that $X_1 = XQ + \dot{\imath}b^T$, where $Q$ is orthogonal and $Q \in M_{r,r_1}$ and $b$ is a nonzero vector in $R^{r_1}$ and $r \ll \min(n, r_1)$. That is $X_1$ is a big matrix but with a rank less than $r$. Assume that $D_1 = X_1X_1^T$, the double centering of $D_1$ obtains the same $B$ as double centering of $D$.

**Theorem 1.** *Let $X_1$ be the dimensional extension from $X$ by an affine mapping $X_1 = XQ + \dot{\imath}b^T$. Then the double centering of $X_1X_1^T$ is the same as the double centering of $XX^T$.*

*Proof.* Let $D_1 = X_1X_1^T = (XQ + \dot{\imath}b^T)(XQ + \dot{\imath}b^T)^T$ and let $D = XX^T$, we have

$$\begin{aligned} D_1 &= XQQ^TX^T + XQb\dot{\imath}^T + \dot{\imath}b^TQ^TX^T + \|b\|^2\dot{\imath}\dot{\imath}^T \\ &= XX^T + a\dot{\imath}^T + \dot{\imath}a^T + \|b\|^2\dot{\imath}\dot{\imath}^T \\ &= D + a\dot{\imath}^T + \dot{\imath}a^T + \|b\|^2\dot{\imath}\dot{\imath}^T, \end{aligned} \tag{4}$$

where $a = XQb$ is a vector in $\Re^m$. By the definition of $D_r$, $D_c$ and $D_g$, we have

$$\begin{aligned} D_{1R} &= D_R + a\dot{\imath}^T + \frac{\sum a_i}{r_1}\dot{\imath}\dot{\imath}^T + \|b\|^2\dot{\imath}\dot{\imath}^T, \\ D_{1C} &= D_C + \dot{\imath}a^T + \frac{\sum a_i}{r_1}\dot{\imath}\dot{\imath}^T + \|b\|^2\dot{\imath}\dot{\imath}^T, \\ D_{1G} &= D_G + 2\frac{\sum a_i}{r_1}\dot{\imath}\dot{\imath}^T + \|b\|^2\dot{\imath}\dot{\imath}^T, \end{aligned} \tag{5}$$

where $a_i$ is the element of $a$. Then we have,

$$D_1 - D_{1R} - D_{1C} + D_{1G} = D - D_R - D_C + D_G. \tag{6}$$

□

Unlike the previous definition of $D$, if $D$ is a distance matrix with each element $d_{i,j} = \sqrt{(x_i - x_j)^T(x_i - x_j)}$, the double centering 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 $-1/2$, and then performs the SVD, which gives the configurations of the data set.

*1.2. Review of the SCMDS.* In 2008, we adapted the classical MDS so as to reduce the original $O(n^3)$ complexity to $O(n)$ [13], in which we have proved that when the data dimension

is significantly smaller than the number of data entries, there is a fast linear approach for the classical MDS.

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 the split-and-combine MDS (SCMDS).

Let $D$ be the $n$-by-$n$ product matrix for some $X$ and let $\mathcal{I}$ be a random permutation of $1, 2, \ldots, n$. Assume that $\mathcal{I}_k$ is a nonempty subset of $\mathcal{I}$ for $k = 1, \ldots, K$, where $\mathcal{I}_k \bigcap \mathcal{I}_{k+1} \neq \emptyset$ for $k = 1, \ldots, K-1$ and $\bigcup \mathcal{I}_k = \mathcal{I}$. $\mathcal{I}_k$ is considered as the index of overlapping subgroups. $D_k$ is the submatrix of $D$ that is extracted from the rows and columns of $D$ by the index $\mathcal{I}_k$. Then $D_k$ is the product matrix of the submatrix of $X$ by index $\mathcal{I}_k$. Since the size of $D_k$ is much smaller than $D$, we can perform the classical MDS easily on each $D_k$ to obtain the MDS result, say $Y_k$. After obtaining the representation coordinates of $Y_k$ for each subgroup, we have to combine these representations into one. Without loss of generality, we only demonstrate how to combine the coordinates of the first two subgroups.

Assume that $X_1$ and $X_2$ are matrices whose columns are the two coordinate sets of the overlapping points with respect to $Y_1$ and $Y_2$, respectively. Then there exists an affine mapping that maps $X_1$ to $X_2$. Let $\overline{X}_1$ and $\overline{X}_2$ 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 $X_1 - \overline{X}_1 \mathrm{i}^T$ and $X_2 - \overline{X}_2 \mathrm{i}^T$. Then we have $X_1 - \overline{X}_1 \mathrm{i}^T = Q_1 R_1$ and $X_2 - \overline{X}_2 \mathrm{i}^T = Q_2 R_2$. It is clear that the mean of the center of column vectors of $X_1 - \overline{X}_1 \mathrm{i}^T$ has been shifted to zero. Because $X_1$ and $X_2$ come from the same data set, the difference between $X_1 - \overline{X}_1 \mathrm{i}^T$ and $X_2 - \overline{X}_2 \mathrm{i}^T$ is a rotation. Therefore, 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

$$Q_1^T \left( X_1 - \overline{X}_1 \mathrm{i}^T \right) = Q_2^T \left( X_2 - \overline{X}_2 \mathrm{i}^T \right). \tag{7}$$

Furthermore, we have

$$X_1 = Q_1 Q_2^T X_2 - Q_1 Q_2^T \left( \overline{X}_2 \mathrm{i}^T \right) + \overline{X}_1 \mathrm{i}^T. \tag{8}$$

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

Assume that there are $n$ points in a data set. We divide these $n$ samples into $K$ overlapping subgroups, and let $n_G$ be the number of points in each subgroup and let $n_I$ be the number of points in each intersection region. Then we have the relationship

$$K n_G - (K-1) n_I = n \tag{9}$$

or

$$K = \frac{(N - n_I)}{(n_G - n_I)}. \tag{10}$$

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

$$\frac{N - r}{(\alpha - 1) r} O\left(\alpha^3 r^3\right) + \frac{N - \alpha r}{(\alpha - 1) r} O\left(r^3\right) \approx O\left(r^2 N\right). \tag{11}$$

When $r \ll n$, the computational cost $O(r^2 n)$ is much smaller than $O(\sqrt{rn}n)$, which is the computation time of the fast MDS method proposed by Morrison et al., 2003 [14]. 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 therefore speed up MDS.

**Proposition 2.** *When the number of samples n is much greater than the dimension of data r, the computational complexity of SCMDS is $O(r^2 n)$, which is linear.*

## 2. Methodology

After reviewing the MDS and the SCMDS, We will now demonstrate how to adapt the SCMDS method to become the fast PCA and how the fast PCA can become the fast SVD with further modification.

*2.1. From the SCMDS to the SCPCA.* Because the MDS is similar to the PCA when the data configuration is Euclidean, we can adapt the SCMDS method to obtain the fast PCA with the similar constraint when the rank $r$ is much smaller than the size $m$ and $n$.

Equation (1) shows how to use double centering to convert the product matrix $D$ to $B$ and the $\sqrt{B} = X - \mathrm{i}\mathrm{i}^T X$. We note that the MDS result $\sqrt{B}$ is equal to the matrix obtained from shifting the coordinates of $X$ with the mean of data points to the original point followed by a transformation by some unitary matrix $P$.

Let $A \in M_{n,r}$; the purpose of the PCA is to find the eigenvectors of the covariance matrix of $A$ and to obtain the corresponding scores. Since the covariance matrix of $A$ defined by $(1/(r-1))(A - A(1/r)\mathrm{i}\mathrm{i}^T)(A - (1/r)\mathrm{i}\mathrm{i}^T)^T$ is symmetric and finding the eigenvectors of the covariance matrix of $A$ is similar to obtaining $V$ of $\sqrt{B}$ in (1), we set

$A - \mathrm{A}(1/r)\mathrm{i}\mathrm{i}^T = X - (1/n)\mathrm{i}\mathrm{i}^T X$ by omitting the constant $1/\sqrt{r-1}$. We note that such omission of the constant will not change the direction of the eigenvectors. When we want to carry out the PCA on $A$, the related matrix $X$ can be obtained by $X = A - A(1/r)\mathrm{i}\mathrm{i}^T + \mathrm{i}\mu$, where $\mu \in R^r$ is defined by

$$\mu = \frac{1}{2n}\mathrm{i}^T\left(A - A\frac{1}{r}\mathrm{i}\mathrm{i}^T\right). \tag{12}$$

In other words, carrying out the PCA on A is equivalent to carrying out the MDS on $D = XX^T$. That is

$$\begin{aligned}&\left(X - \frac{1}{n}\mathrm{i}\mathrm{i}^T X\right)\left(X - \frac{1}{n}\mathrm{i}\mathrm{i}^T X\right)^T \\ &\quad = \left(A - A\frac{1}{r}\mathrm{i}\mathrm{i}^T\right)\left(A - \frac{1}{r}\mathrm{i}\mathrm{i}^T\right)^T = V\Sigma V^T,\end{aligned} \tag{13}$$

and the result of PCA will be $V$ and $(1/\sqrt{r-1})\Sigma^{1/2}$. From the result of the MDS ($\sqrt{B} = V\Sigma^{1/2}$), we can easily obtain $V$ and $\Sigma$ by normalizing the column vectors of $\sqrt{B}$.

In practice, we do not want to produce the product matrix $D = XX^T$ when $n$ is huge. Rather, after we obtain matrix $X$ from $A$, we will randomly produce the index permutation $\mathcal{I}$ and use the index of overlapping subgroup $\mathcal{I}_k$ to compute $D_k$. And then we follow the step of the SCMDS to obtain the result, $\sqrt{B}$.

However, the result of the SCMDS is of the form $\sqrt{B}Q + \mathrm{i}b^T$, where $Q$ is an unitary matrix in $M_r$ and $b \in \mathfrak{R}^r$ is usually a nonzero vector. Removal of the factor $b$ can be easily done by computing the average of the row vectors of the SCMDS result. After removing the component $b$ from each row vector, we obtain $\sqrt{B}Q$.

Since $\sqrt{B}Q \in M_{m,r}$ and $r \ll m$, we have

$$\left(\sqrt{B}Q\right)^T\sqrt{B}Q = Q^T\Sigma^{1/2}V^TV\Sigma^{1/2}Q = Q^T\Sigma Q, \tag{14}$$

which is a $r \times r$ small matrix. It is easy to solve this small eigenvalue problem to obtain $Q$, $\Sigma$, and $V$. The computational cost from $\sqrt{B}Q + \mathrm{i}^b T$ to obtain $V$ and $\Sigma^{1/2}$ is $O(m)$. Hence, the SCMDS approach to computing the SVD is still linear.

Notice that there are some linear SVD methods for thin matrix ($m \ll n$ or $m \gg n$). Even in the case of big matrix with small rank, the traditional SVD method, for example the GR-SVD, can be implemented to be linear (because of the dependency, many components of the matrix become zero in the GR-SVD algorithm). However, if the rank of big matrix is almost full and the numerical rank is small or the matrix size is considerably huge that loading the whole matrix is impossible, the SCMDS has the advantage in computing speed. And we call this approach to obtaining $V$ and $\Sigma^{1/2}$ by the SCMDS to be SCPCA.

*2.2. From the SCPCA to the SCSVD.* The concepts of the SVD and the PCA are very similar. Since the PCA starts from decomposing the covariance matrix of a data set, it can be considered as adjusting the center of mass of a row vector to zero. On the other hand, the SVD operates directly on the product matrix without shifting. If the mean of the matrix rows is zero, the eigenvectors derived by the SVD are equal to the eigenvectors derived by the PCA. We are looking for a method which will give a fast approach 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: $\widetilde{X} = X - \overline{X}\cdot\mathrm{i}^T$, where $\overline{X}$ is the mean of columns of $X$. Hence, the row mean of $\widetilde{X}$ is zero. Assume that we have the PCA result of $X$, that is, $\widetilde{X}\widetilde{X}^T = U\Sigma^2U^T$. Then we have $\widetilde{X} = U\Sigma W^T$ for some orthogonal matrix $W$. Assume that the rank of $\widetilde{X}$ is $r$ and $r$ is much smaller than the matrix size. We observe that $\mathrm{rank}(X) = r$ or $\mathrm{rank}(X) = r + 1$, depending on whether $\overline{X}$ is spanned by columns of $U$. If $\overline{X}$ is spanned by $U$, then

$$X = \widetilde{X} + \overline{X}\cdot\mathrm{i}^T = U\Sigma W^T + U\cdot c\cdot\mathrm{i}^T = U\left(\Sigma W^T + c\cdot\mathrm{i}^T\right), \tag{15}$$

where $c$ is the coefficient vector of $\overline{X}$ when represented by $U$, that is, $\overline{X} = U\cdot c$.

If the singular value decomposition of $\Sigma W^T + c\cdot\mathrm{i}^T$ is $U_2\Sigma_1V_1^T$, we have

$$X = U\left(U_2\Sigma_1V_1^T\right) = \left(UU_2\right)\Sigma_1V_1^T = U_1\Sigma_1V_1^T. \tag{16}$$

Because the matrix $U_2$ is unitary, $U_1 = UU_2$ is automatically an orthogonal matrix as well. Then we have the SVD of $X$.

Checking the matrix size of $\Sigma W^T + c\cdot\mathrm{i}^T$, we can see that to compute the SVD of $\Sigma W^T + c\cdot\mathrm{i}^T$ is not a big task. This is because $\Sigma W^T + c\cdot\mathrm{i}^T$ 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 $\Sigma W^T + c\cdot\mathrm{i}^T$.

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

$$X = \widetilde{X} + \widetilde{X}\cdot\mathrm{i}^T = [U \mid u_{r+1}]\left(\begin{bmatrix}\Sigma & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}V^T\\ 0\end{bmatrix} + c\cdot\mathrm{i}^T\right), \tag{17}$$

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

$$u_{r+1} = \frac{\left(I - UU^T\right)\overline{X}}{\left\|\left(I - UU^T\right)\overline{X}\right\|}. \tag{18}$$

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

$$\left(\begin{bmatrix}\Sigma & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}V^T\\ 0\end{bmatrix} + c\cdot\mathrm{i}^T\right) = U_2\Sigma_1V_1^T. \tag{19}$$

Then $X = [U \mid u_{r+1}]U_2\Sigma_1V_1^T = U_1\Sigma_1V_1^T$, where $U_1 = [U \mid u_{r+1}]U_2$ is another orthogonal matrix, and hence the SVD of $X$ is completed.

**Proposition 3.** *Let $A \in M_{m,n}(\mathfrak{R})$ be a big size matrix and let its rank $r$ be much smaller than $\min\{m,n\}$. Then the SCSVD or the SCPCA has linear computational complexity.*

From the above analysis, we can have a fast PCA approach by computing the SCMDS first and then adapt the MDS result to obtain the PCA. We named this approach SCPCA. Similarly, the fast SVD approach, which computes the SCMDS first, then adapts the MDS result to obtain the PCA, and finally adapts the PCA result to the SVD, is called the SCSVD. These two new approaches 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 the SCPCA or SCMDS method, if $n_I \leq r$, we only get the approximated solutions of the PCA and the SVD. Under the necessary criterion, we can reduce the computational complexity from $\min\{O(rmn), O(mn^2), O(nm^2\}$ to $\min\{O(rm), O(rn)\}$. If the numerical rank is not small, for example, $r \approx \min(\sqrt{m}, \sqrt{n})$, the computational complexity becomes almost the same as the original PCA and SVD. Our approach has no advantage in the latter case.

## 3. SVD for Continuously Growing Data

In this section, we look for the solution when the data is updated constantly and we need to compute the SVD continuously. Instead of scanning all the data again, we try to use the previous result of the SCSVD together with the new updated data to compute the next SVD. Before introducing our updating method, we review the general updating methods first.

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 $E$. Assume that we have the singular value decomposition of $A$, that is,

$$A = U\Sigma V^T, \tag{20}$$

where $U \in M_{m,r}(\Re)$, $V \in M_{n,r}(\Re)$ are orthogonal and $\Sigma \in M_r(\Re)$ is a diagonal. Since the data gets updated, the data matrix becomes

$$A_1 = [A \mid E]. \tag{21}$$

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

If $E$ is spanned by the column space of $U$, we can represent the column matrix $E$ by $E = UC$, where $C$ is the coefficient matrix of $U$ with columns of $S$ as the basis. Since $U$ is orthogonal, the coefficient matrix $C$ can be computed easily by $C = U^T E$. Then we have

$$\begin{aligned} A_1 A_1^T &= [A \mid UC]\,[A \mid UC]^T \\ &= AA^T + UC(UC)^T \\ &= U\left(\Sigma^2 + CC^T\right)U^T \\ &= UQ\Sigma_1^2 Q^T U^T \\ &= U_1 \Sigma_1^2 U_1^T. \end{aligned} \tag{22}$$

Note that the matrix $\Sigma^2 + CC^T$ is positive symmetric and it is a small size matrix. Using the spectrum theorem, we can decompose this matrix into $Q\Sigma_1^2 Q^T$. Because the matrix $Q$ is unitary, $U_1$ is $U$ rotated by $Q$. In this case, the computational cost is dominated by the rotation from $U$ to $U_1$, and it is $O(mr^2)$.

In general, the extended matrix $E$ is not spanned by the column space of $U$ and the decomposition of $A_1 A_1^T$ should be modified by

$$\begin{aligned} A_1 A_1^T &= [A \mid UC + E_1]\,[A \mid UC + E_1]^T \\ &= AA^T + (UC + E_1)(UC + E_1)^T \\ &= [U \mid Q_1] \left[\begin{array}{c|c} \Sigma^2 + CC^T & CR_1^T \\ \hline R_1 C^T & R_1 R_1^T \end{array}\right] \left[\begin{array}{c} U^T \\ \hline Q_1^T \end{array}\right] \\ &= [U \mid Q_1]\, Q_2 \Sigma_1^2 Q_2^T \left[\begin{array}{c} U^T \\ \hline Q_1^T \end{array}\right] \\ &= U_1 \Sigma_1^2 U_1^T, \end{aligned} \tag{23}$$

where $E = UC + E_1$, $U_1 = [U \mid Q_1]Q_2$ and $E_1 = Q_1 R_1$ is the $QR$ decomposition of $E_1$. The cost to obtain $C$ and $E_1$ is $O(mkr)$, the computational cost of $QR$ decomposition of $E_1$ is $O(mk^2)$, the cost to obtain $\Sigma_1^2$ is $O((r+k)^3)$, and the cost of rotation $U$ to $U_1$ is $O(m(r+k)^2)$. When $m \gg r, k$ the cost of update process is linear $O(m)$. However, if the rank is not small ($r \approx m$ or $r \approx n$), the computational cost is dominated by $O((r+k)^3)$.

Now, we discuss how to update the SVD in the SCSVD approach. If the updated column vectors $E$ are spanned by the column vectors of some subgroup of the original data, say $X_i$, the dimension will not increase by the extended matrix $E$. We just reset the distance matrix $D_i$ in this subgroup by

$$\begin{aligned} D_i = &\left([X_i \mid E_i] - \frac{1}{n_{G,i} + k}\,[X_i \mid E_i]\,\mathbf{i}\mathbf{i}^T\right)^T \\ &\times \left([X_i \mid E_i] - \frac{1}{n_i + k}\,[X_i \mid E_i]\,\mathbf{i}\mathbf{i}^T\right), \end{aligned} \tag{24}$$

where $n_{G,i}$ is the number of vectors in the original $i$-th subgroup. To compute the MDS result of this updated subgroup, it costs $O((n_{G,i}+k)^3)$. Then we will combine this $n_{G,i}+k$ MDS representation points with the remaining points. The cost of obtaining the combination affine mapping is still $O((n_I+k)^3)$ which is increased by the new update. Combining this new subgroup with the whole data set needs $O(r^2(n_{G,i}-n_I))$. Then transfering the MDS result to the SVD needs $O(mr^2)$, which is linear too.

If $E$ is not spanned by any subgroup of the previous SCMDS process, then we have to update $E$ to the largest group of the subgroups. Because the dimension of this updated group will increase, we have to make some modifications in the combined approach. If the column matrix of the MDS

configuration of the new updated group is $X_1 \in M_{r_1,k_1}$ and the column matrix of the configuration of the other elements is $X_2 \in M_{r,k_2}$, where $r_1 > r$, then we will extend the matrix $X_2$ to an $r_1$-by-$k_2$ matrix by filling zeros in the bottom of the original $X_2$. After adjusting $X_1$ and $X_2$ to the same dimension of rows, we perform the same combination step as before. Here, the width of matrix $X_2$ is almost the number of whole data, and this combination step takes $O(mr_1^2)$.

There is one important difference between the general approach and the SCSVD approach. To obtain the SVD from the MDS result, we need the column mean vector of the matrix. The column mean vector of the original matrix must be computed, and then the column mean vector is updated by the simple weighted average when the new data comes in.

Notice that no matter whether $E$ is spanned by $U$ or not, the update method in the SCSVD has no advantage over the general update method although the computational costs are of the same order. This is because the true computational cost of the SCSVD updating method is always more than the general updating method, even though they are of the same order. Thus, when we have to update the new data to the matrix on which the SVD will be performed, the general method is recommended. However, when the matrix size is so huge that to expand $\Sigma^2 + CC^T$ becomes impossible, it is better to recompute the SVD result. In this case and with the condition that the true rank is much smaller than the matrix size, the SCSVD approach is recommended.

## 4. Experimental Result

In this section, we show that our fast PCA and fast SVD methods work well for big-sized matrices with small ranks. The simulated matrix is created by the product of two slender matrices and one small diagonal matrix, that is, $A_{m,r} \times \Lambda_r \times B_{r,n}$, where $m, n \gg r$. The size of the first matrix is $m \times r$, the second matrix is $r \times r$, and the third matrix is $r \times n$. Then the product of these three matrixes is of size $m \times n$ and its rank is smaller than $r$. The absolute values of the diagonal elements of $\Lambda$ are decreasing to simulate the situation as data normally comes in. When $m$ and $n$ are large and $r$ is much smaller than $m$ and $n$, the simulated matrix satisfies our SCSVD condition. We pick $m = 4000$, $n = 4000$, and $r = 50$ as our first example. Each element of the simulated matrix is generated from the normal distribution $\mathcal{N}(0, 1)$ and then the diagonal terms of $\Lambda$ are adjusted. The average elapsed time of the SCSVD is 3.98 seconds, while the economical SVD takes 16.14 seconds. Here, each average value is derived from 100 repeats.

If we increase the matrix to $m = 20000$, $n = 20000$ and fix the same rank $r = 50$, the elapsed time of the economical SVD is 1209.92 seconds, but the SCSVD is only 195.85 seconds. We observe that our SCSVD method demonstrates significant improvement. Figure 1 shows the speed comparison between the economical SVD (solid line) and the SCSVD (dashed line) with the square matrix size from 500 to 4000 by fixed rank 50. 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 the SVD follows the order 3 increase, compared with linear increase of the SCSVD.

Note that when the estimated rank used in the SCSVD is greater than the real rank of data matrix, there is almost no error (except rounding error) between the economic SVD and the SCSVD. The error between the economical SVD and the SCSVD is computed by comparing the orthogonality. Assume that $A = U\Sigma V^T$ is derived by the SCSVD and $A = U_1\Sigma_1 V_1^T$ is derived from the economical SVD. Since the output of the column vectors of $U$ and $V$ might reverse, the error is defined by

$$\left\| U_{20}^T U_{1,20} - I_{20} \right\|, \tag{25}$$

where $U_{20}$ is the first 20th column of $U$, $U_{1,20}$ is the first 20th column of $U_1$, and $I_{20}$ is the identity matrix of size 20. The error is about $10^{-13}$ only. The average error in the same experiment in Figure 1 is $9.7430 \times 10^{-13}$ and the standard deviation is $5.2305 \times 10^{-13}$. Thus, when the estimated rank of the SCSVD is greater than the true rank, the accuracy of the SCSVD is pretty much the same as the SVD in the case of a small rank matrix.

We would like to explore what happens if the estimated rank is smaller than the true rank. According to the experiment of the SCMDS [13], if the estimated rank is smaller than the true rank, the performance of the SCMDS will deteriorate. In our experiment, we set the true rank of the simulated matrix to be 50 and then observe the estimated rank from 49 to 39. The relationship between the error and the estimated rank with different matrix size is shown in Figure 2.

We can see that when the estimated rank decreases, the error arises rapidly. Lines in Figure 2 from the bottom to the top are the matrix size with $500 \times 500$ to $4000 \times 4000$, respectively. The error increases slowly when the matrix size increases. We observe that making the estimated rank greater than the true dimension is essential for our method.

The purpose of the second simulation experiment is to observe the approximation performance of applying the SCPCA to a big full rank matrix. We generate a random matrix with a fixed number of columns and rows, say 1000. The square matrix is created by the form, $A_{1000,r} \times \Lambda \times B_{r,1000} + \alpha E_{1000,1000}$, where $r = 50$ is the essential rank, $E$ is the perturbation, and $\alpha$ is a small coefficient for adjusting the influence to the previous matrix. Such a matrix can be considered as a big-sized matrix with a 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 versus estimated rank, where the error is defined as (25), which is a comparison of the orthogonality between $U$ and $U_1$. All the elements of matrices $A$, $B$, $E$ are randomly generated from the normal distribution $\mathcal{N}(0, 1)$ and $\Lambda$ is a diagonal matrix in which the diagonal terms decay by the order, 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 $r$, there is almost no error. 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 the SCSVD is smaller than the essential rank, our SCSVD result can be used as the approximated solution of the SVD.

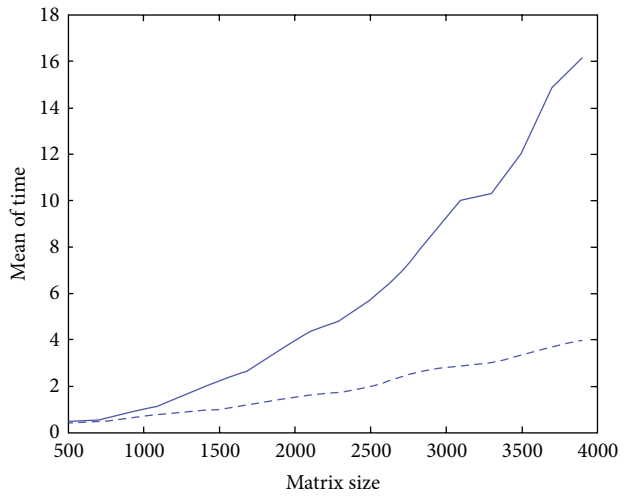


FIGURE 1: Comparison of the elapsed time between economical SVD (the solid line) and SCSVD (the dashed line).

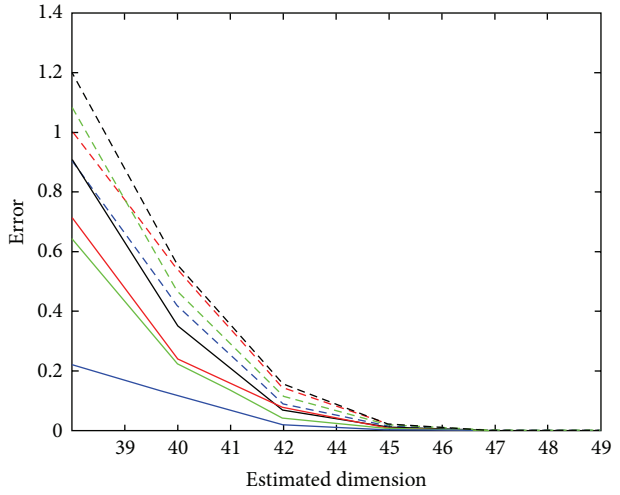


FIGURE 2: The relationship between errors and estimated dimension of matrix size from 500 to 4000 with step size 500. The true rank of these matrices is 50. From the bottom to top is matrix of sizes 500, 1000, 1500,. . ., respectively.

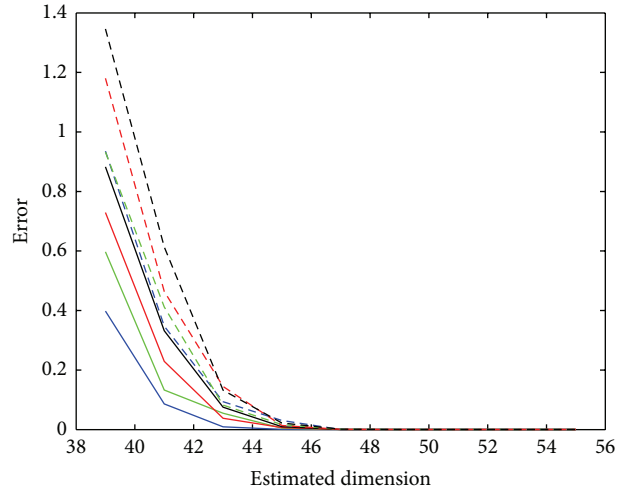


FIGURE 3: The effect of estimated rank to the error. The matrix size is from 500-by-500 to 4000-by-4000 (from the bottom to the top, resp.) and its essential rank is 50 ($\alpha = 0.01$). When the estimated rank is greater than 50, there is almost no composition error.

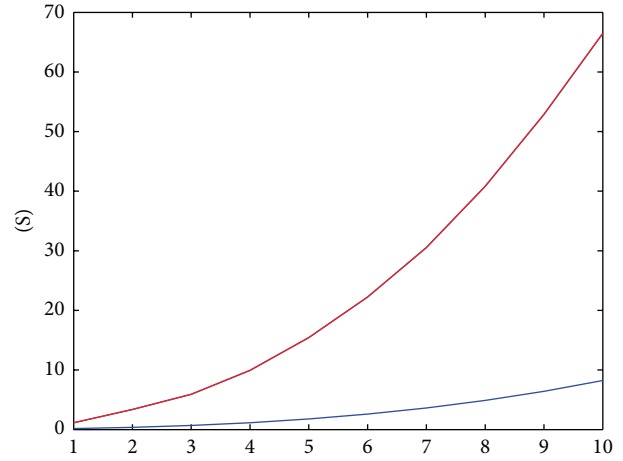


FIGURE 4: Compare the update time for SCMDS approach (red line) and the general approach (blue line).

In the last experimental result, we let the matrix be growing and observe the performance between the general and the SCSVD update approaches. We start from a matrix that is formed by $A_{4000,50} \times \Lambda_{50} \times B_{50,4000} + 0.01 \cdot E_{4000,4000}$, where $\Lambda$ is a diagonal matrix such that the diagonal terms are positive and decayed rapidly like natural data and each element in $A$ and $B$ is a random number from the normal distribution $\mathcal{N}(0, 1)$. Then we update a $100 \times 4000$ random matrix into the matrix 10 times of the size. The element of the updated matrix is also from the normal distribution $\mathcal{N}(0, 1)$. When the matrix gets updated, we compute the SVD decomposition to obtain the first 20 columns of $U$. The updated result of the general approach is $U_1 \times \Sigma_1 \times V_1$, and the updated result of the SCMDS approach is $U_2 \times \Sigma_2 \times V_2$. We compute the error as (25) for $U_1$ and $U_2$. The reason that we concern ourselves only with the first 20 bases of the column space is that it is rare to use such a high dimension in the dimension reduction applications.

The computational time of the SCMDS is more than the general approach; however, the difference is within one second. The error of the SCMDS approach to update 100 rows compared with recomputing the SVD decomposition is $4.0151 \times 10^{-5}$ with standard deviation $3.5775 \times 10^{-5}$. The error of general approach is $1.4371 \times 10^{-12}$ with standard deviation $1.5215 \times 10^{-12}$. The update time of the SCMDS and general approach is shown in Figure 4. The red solid line is the time for the SCMDS and the blue line is that for general approach. We can see that the SCMDS approach is about 8 times of general approach. Hence the SCSVD update approach is not recommended for either saving time or controlling error.

## 5. Conclusion

We proposed the fast PCA and fast SVD methods derived from the technique of the SCMDS method. The new PCA and the SVD have the same accuracy as the traditional PCA and the SVD method when the rank of a matrix is much smaller than the matrix size. The results of applying the SCPCA and the SCSVD to a full rank matrix are also quite reliable when the essential rank of matrix is much smaller than the matrix size. Thus, we can use the SCSVD in a huge data application to obtain a good approximated initial value. The updating algorithm of the SCMDS approach is discussed and compared with the general update approach. The performances of the SCMDS approach both in the computational time and error are worse than the general approach. Hence, the SCSVD method is only recommended for computing the SVD for a large matrix but is not recommended for the update approach.

## Acknowledgment

This work was supported by the National Science Council.

## References

[1] D. J. Hand, *Discrimination and Classification*, Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons, Chichester, UK, 1981.

[2] M. Cox and T. Cox, *Multidimensional Scaling, Handbook of Data Visualization*, Springer, Berlin, Germany, 2008.

[3] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," *Numerische Mathematik*, vol. 14, no. 5, pp. 403–420, 1970.

[4] T. F. Chan, "An improved algorithm for computing the singular value decomposition," *ACM Transactions on Mathematical Software*, vol. 8, no. 1, pp. 72–83, 1982.

[5] P. Deift, J. Demmel, L. C. Li, and C. Tomei, "The bidiagonal singular value decomposition and Hamiltonian mechanics," *SIAM Journal on Numerical Analysis*, vol. 28, no. 5, pp. 1463–1516, 1991.

[6] L. Eldén, "Partial least-squares vs. Lanczos bidiagonalization. I. Analysis of a projection method for multiple regression," *Computational Statistics & Data Analysis*, vol. 46, no. 1, pp. 11–31, 2004.

[7] J. Baglama and L. Reichel, "Augmented implicitly restarted Lanczos bidiagonalization methods," *SIAM Journal on Scientific Computing*, vol. 27, no. 1, pp. 19–42, 2005.

[8] J. Baglama and L. Reichel, "Restarted block Lanczos bidiagonalization methods," *Numerical Algorithms*, vol. 43, no. 3, pp. 251–272, 2006.

[9] M. Brand, "Fast low-rank modifications of the thin singular value decomposition," *Linear Algebra and Its Applications*, vol. 415, no. 1, pp. 20–30, 2006.

[10] W. S. Torgerson, "Multidimensional scaling. I. Theory and method," *Psychometrika*, vol. 17, pp. 401–419, 1952.

[11] M. Chalmers, "Linear iteration time layout algorithm for visualizing high-dimensional data," in *Proceedings of the 7th Conference on Visualization*, pp. 127–132, November 1996.

[12] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

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

[14] A. Morrison, G. Ross, and M. Chalmers, "Fast multidimensional scaling through sampling, springs and interpolation," *Information Visualization*, vol. 2, no. 1, pp. 68–77, 2003.