FAST MONTE-CARLO LOW RANK APPROXIMATIONS FOR MATRICES

Shmuel Friedland, Amir Niknejad

Department of Mathematics,
Statistics and Computer Science,
University of Illinois - Chicago,
Chicago, Illinois 60607-7045, USA
friedlan@uic.edu, niknejad@math.uic.edu.

Mostafa Kaveh, Hossein Zare

Department of Electrical
and Computer Engineering
University of Minnesota,
Minneapolis, MN 55455, USA
{mos, hossein}@ece.umn.edu

ABSTRACT

In many applications, it is of interest to approximate data, given by $m \times n$ matrix A, by a matrix B of at most rank k, which is much smaller than m and n. The best approximation is given by singular value decomposition, which is too time consuming for very large m and n.

We present here a Monte Carlo algorithm for iteratively computing a k-rank approximation to the data consisting of $m \times n$ matrix A. Each iteration involves the reading of $O(k)$ of columns or rows of A. The complexity of our algorithm is $O(kmn)$. Our algorithm, distinguished from other known algorithms, guarantees that each iteration is a better k-rank approximation than the previous iteration. We believe that this algorithm will have many applications in data mining, data storage and data analysis.

***Index Terms*— SVD decomposition, fast k-rank approximation, Monte-Carlo algorithm.**

I. INTRODUCTION

In many applied settings, dealing with a very large data set, it is important to reduce the size of data in order to make an inference about the features of data set, in a timely manner. Assume that the data set is represented by an $m \times n$ matrix $A \in \mathbb{R}^{m \times n}$. It is important to find an approximation $B \in \mathbb{R}^{m \times n}$ of a specified rank k to A, where k is much smaller than m and n.

Here are several motivation to obtain such B. First, the storage space needed for B is $k(m+n)$, which is much smaller than the storage space mn needed for A. Indeed, B can be represented as

$$B = \mathbf{x}_1\mathbf{y}_1^{\mathrm{T}} + \mathbf{x}_2\mathbf{y}_2^{\mathrm{T}} + \dots \mathbf{x}_k\mathbf{y}_k^{\mathrm{T}}, \tag{I.1}$$

where $\mathbf{x}_1, \dots, \mathbf{x}_k$ and $\mathbf{y}_1, \dots, \mathbf{y}_k$ are k column vectors with m and n coordinates respectively. We store the vectors $\mathbf{x}_1, \dots, \mathbf{x}_k, \mathbf{y}_1, \dots, \mathbf{y}_k$, which need the storage $k(m+n)$, and compute the entries of B, when needed, using the above expression of B.

The second most common application is clustering algorithms as in [1], [2], [3], [4] and [10]. Assume that our data represents n points and each point has m coordinates. That is, each point is represented by a column of the matrix A. We want to cluster the points in such a way that the distance between points in the same cluster is much smaller than the distance between any two points from different clusters. One way to do this is to project all the point on the k main orthonormal directions encoded by the first k left singular vectors of A. Then cluster using either k one dimensional subspaces or the whole k dimensional subspace. The k-rank approximation B gives the approximation to this k dimensional subspace and the approximation to the first k singular vectors. Another way to cluster is to use the projective clustering. It is known that a fast SVD, i.e. fast k-rank approximation, is a main tool in this area [2] and [3].

The third application is in DNA microarrays, in particular in data imputation [7]. Let A be the gene expression matrix. The m rows of the matrix A are indexed by the genes, while the n columns are indexed by the number of experiments. It is known that the effective rank of A is k, is usually much less then n. The SVD decomposition is the main ingredient of the FRAA, (*fixed rank approximation algorithm*), which successfully implemented in [7] to impute the corrupted entries of A. A remarkable improvement of FRAA, called IFRAA, (*improved fixed rank approximation algorithm*), is given in [6]. IFRAA is a combination of FRAA and any good clustering algorithm. One first apply FRAA, then cluster the genes to a relative small number of similar genes. Next one applies FRAA to each cluster of genes to impute the missing entries of genes from the data of the genes in that cluster only. The fast k-rank approximation algorithm suggested in this paper, can be used to cluster fast and efficiently similar genes. This clustering algorithm can be incorporated with the IFRAA algorithm.

The best approximation B, which minimizes the Frobenius norm $||A-B||_{\mathrm{F}}^2 := \mathrm{trace}\,(A-B)^{\mathrm{T}}(A-B)$, is given by the *singular value decomposition* (SVD) of A. Consult with [9] for detailed analysis of SVD. However SVD decomposition needs $O(mn \cdot \min(m,n))$ time computation, which is often too prohibitive.

One way to find a *fast* k-rank approximation is to choose at random $l \geq k$ columns or rows of A and obtain from them k-rank approximations of A. This is basically the spirit of the algorithm suggested in [8]. We call this algorithm the FKV algorithm. Assuming a statistical model for the distribution of the entries of A the authors give some error bounds on their k-rank approximation. The weak point of FKV algorithm is its inability to improve iteratively FKV approximation by incorporating additional parts of A.

The aim of this paper is to provide a sampling framework for iterative updates of k-rank approximations of A, by reading iteratively additional columns or rows of A, which improves *for sure* the approximation B each time it is updated. The quality of the approximation of B is given by the Frobenius norm $||B||_{\mathrm{F}}$, which is a nondecreasing sequence under these iterations. The rate of increase of the norms $||B||_{\mathrm{F}}$ can be used as a stopping rule for terminating the algorithm. Also the updating algorithm of k-rank approximation gives approximation to the first k singular values of A, and the approximations to the first k left and right singular vectors of A.

Assuming that the number of columns or rows which are read is $l = O(k)$, the complexity of our algorithm is $O(kmn)$. The intensive part of the computations is devoted to obtain the spectral decompositions of $(k+l) \times (k+l)$ real symmetric matrices, where the computation for each decomposition is of order $O(k^3)$. The $O(kmn)$ part of the algorithm is due to the multiplications of k column vectors, which approximate the k left or right singular eigenvectors of A, by A^{T} or A respectively. This part of the algorithm can be parallelized, which will speed significantly the

algorithm suggested here. The simulations that we performed show that we need a small number of iterations, (around 5), to get a very good approximation to the best k-rank approximation of A.

We believe that this algorithm will have many applications in data mining, data storage and data analysis.

II. SVD

In this section, we recall some basic facts about SVD, which are embedded in our algorithm, see [9]. Let $\mathbb{R}^m, \mathbb{R}^{m\times n}, \mathrm{O}_{md}(\mathbb{R}), \mathrm{S}_n(\mathbb{R})$, be the linear space of real column vectors with m coordinates, the linear space of real $m \times n$ matrices, the subset of $m \times d$ real valued matrices whose $d\ (\leq m)$ columns is an orthonormal system and the subspace of $n \times n$ real symmetric matrices respectively. For $S \in \mathrm{S}_n(\mathbb{R})$ we let $S \geq 0$ if S is nonnegative definite and $S > 0$ if S is positive definite. Denote by $\mathrm{diag}(d_1, \ldots, d_m) \in \mathbb{R}^{m\times m}$ the matrix with the diagonal entry d_i on the (i, i) position for $i = 1, ..., m$ and all other entries are equal to zero. Let $A \in \mathbb{R}^{m\times n}$ and denote by $r = \mathrm{rank}\, A$ the rank of A. Then the SVD decomposition of A is given as $A = U_r\Sigma_r V_r^{\mathrm{T}}$, where $U_r \in \mathrm{O}_{mr}(\mathbb{R}), V_r \in \mathrm{O}_{nr}(\mathbb{R})$ and $\Sigma_r \in \mathrm{S}_r(\mathbb{R})$ is a diagonal matrix. So $U_r^{\mathrm{T}}U_r = V_r^{\mathrm{T}}V_r = I_r$, and $\Sigma_r = \mathrm{diag}(\sigma_1, ..., \sigma_r)$, $\sigma_1 \geq ... \geq \sigma_r > 0$. Let $\mathbf{u}_1, \cdots, \mathbf{u}_r \in \mathbb{R}^m$ and $\mathbf{v}_1, \ldots, \mathbf{v}_r \in \mathbb{R}^n$ denote $1, \ldots, r$ orthonormal columns of U and V respectively. Then $A = U_r\Sigma_r V_r^{\mathrm{T}}$ can be written as $A = \sum_{q=1}^r \sigma_q \mathbf{u}_q\mathbf{v}_q^{\mathrm{T}}$. The vectors $\mathbf{u}_i, \mathbf{v}_i$ are called the *left* and *right singular vectors* of A respectively, which correspond to the singular value σ_i. The left and the right singular vectors can be computed from the right and the left singular vectors by the formulas:

$$\mathbf{u}_i = \frac{1}{\sigma_i}A\mathbf{v}_i, \quad \mathbf{v}_i = \frac{1}{\sigma_i}A^{\mathrm{T}}\mathbf{u}_i, \quad i = 1, \ldots, r. \tag{II.1}$$

Equivalently, $\sigma_1^2 \geq \ldots \geq \sigma_r^2 > 0$ are all the positive eigenvalues of the nonnegative definite symmetric matrices $AA^{\mathrm{T}}, A^{\mathrm{T}}A$, with the corresponding orthonormal eigenvectors $\mathbf{u}_1, \cdots, \mathbf{u}_r \in \mathbb{R}^m$ and $\mathbf{v}_1, \ldots, \mathbf{v}_r \in \mathbb{R}^n$. For the purposes of this paper, it will be convenient to assume that $\sigma_q = 0$ for any $q > \mathrm{rank}\, A$.

Denote by $||A||_{\mathrm{F}} := \sqrt{\mathrm{trace}\ A^{\mathrm{T}}A} = \sqrt{\mathrm{trace}\ AA^{\mathrm{T}}}$ the Frobenius (ℓ_2) norm of A. It is the Euclidean norm of A viewed as a vector with mn coordinates. Each term $\sigma_q\mathbf{u}_q\mathbf{v}_q^{\mathrm{T}}$ in SVD decomposition of A is a rank one matrix with $||\sigma_q\mathbf{u}_q\mathbf{v}_q^{\mathrm{T}}||_{\mathrm{F}} = \sigma_q$. For an integer $k \in [1, r]$ let $A_k := \sum_{i=1}^k \sigma_i\mathbf{u}_i\mathbf{v}_i^{\mathrm{T}}$. Then $\mathrm{rank}\, A_k = k$ for $k = 1, \ldots, r$. It is known that A_k is uniquely defined if and only if $\sigma_k > \sigma_{k+1}$. Note that $A_r = A$.

Let $\mathcal{R}(m, n, k)$ denote the set of $m \times n$ matrices of at most rank k, $(\min(m, n) \geq k)$. Then for each k, $k \leq r$, A_k gives the solution to the following approximation problem:

$$\min_{B\in\mathcal{R}(m,n,k)} ||A - B||_{\mathrm{F}} = ||A - A_k||_{\mathrm{F}} \sqrt{\sum_{q=k+1}^r \sigma_q^2}. \tag{II.2}$$

A_k is the unique solution to this minimal problem if and only if $\sigma_k > \sigma_{k+1}$.

Given $k \in [1, r]$ then $\sigma_1^2, \ldots, \sigma_k^2$, $\mathbf{u}_1, \ldots, \mathbf{u}_k$ and $\mathbf{v}_1, \ldots, \mathbf{v}_k$ are characterized by the following maximal characterization:

Theorem 2.1: Let $A \in \mathbb{R}^{m\times n}$ and $k \in [1, \min(m, n)]$. Let $\mathbf{x}_1, \ldots, \mathbf{x}_k$ and $\mathbf{y}_1, \ldots, \mathbf{y}_k$ be two sets of orthonormal vectors in $\mathbb{R}^m$ and $\mathbb{R}^n$ respectively. Then

$$\sum_{i=1}^k (A^{\mathrm{T}}\mathbf{x}_i)^{\mathrm{T}}(A^{\mathrm{T}}\mathbf{x}_i) \leq \sum_{i=1}^k \sigma_i^2, \ \sum_{i=1}^k (A\mathbf{y}_i)^{\mathrm{T}}(A\mathbf{y}_i) \leq \sum_{i=1}^k \sigma_i^2. \tag{II.3}$$

Equality in the first or second inequality occurs if $\mathrm{span}(\mathbf{x}_1, \ldots, \mathbf{x}_k)$ or $\mathrm{span}(\mathbf{y}_1, \ldots, \mathbf{y}_k)$, contains k linearly independent left or right singular vectors of A, corresponding the the k maximal singular values of A respectively.

The above characterization follows from the maximal, (Ky Fan characterization), of the sum of the first biggest eigenvalues of a real symmetric matrix:

Theorem 2.2: Let $S \in S_p(\mathbb{R})$ be a real $p \times p$ symmetric matrix. Let $\lambda_1 \geq \ldots \geq \lambda_p$ be the eigenvalues S arranged in a decreasing order and listed with their multiplicities. Let $\mathbf{w}_1, \ldots, \mathbf{w}_p \in \mathbb{R}^p$ be an orthonormal set of the corresponding eigenvectors of S: $S\mathbf{w}_i = \lambda_i\mathbf{w}_i, i = 1, \ldots, p$. Let $k \in [1, p]$ be an integer. Then for any orthonormal set $\mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathbb{R}^p$

$$\sum_{i=1}^k \mathbf{x}_i^{\mathrm{T}}S\mathbf{x}_i \leq \sum_{i=1}^k \lambda_i = \sum_{i=1}^k \mathbf{w}_i^{\mathrm{T}}S\mathbf{w}_i. \tag{II.4}$$

Equality holds if and only if the subspace $\mathrm{span}(\mathbf{x}_1, \ldots, \mathbf{x}_k)$ contains k linearly independent eigenvectors of S corresponding to the eigenvalues $\lambda_1, \ldots, \lambda_k$.

See for example [5] for proofs and the references. Note that $\mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathbb{R}^m$ is a system of orthonormal vectors in $\mathbb{R}^m$ if and only if the $m \times k$ matrix $[\mathbf{x}_1, \ldots, \mathbf{x}_k]$ is in $\mathrm{O}_{mk}(\mathbb{R})$.

To obtain Theorem 2.1 from Theorem 2.2 we let $p = m$, (or $p = n$), and S to be equal to AA^{T}, (or $A^{\mathrm{T}}A$). In (II.3) we emphasized the complexity of the computations of the left-hand side of the inequalities. See also [7] for applications of Theorem 2.2 to data imputation in DNA microarrays.

In the rest of the paper we give the version of our results to k-orthornormal systems $\mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathbb{R}^m$. Similar results holds for k-orthonormal systems $\mathbf{y}_1, \ldots, \mathbf{y}_k \in \mathbb{R}^n$.

The following proposition is used to show that the rate of the approximation of our iterates to A_k are improving with each iteration.

Proposition 2.3: Let $A \in \mathbb{R}^{m\times n}$ and $k \in [1, \min(m, n)]$ be an integer. Then for any k-orthonormal system $\mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathbb{R}^m$ the following equality holds:

$$||A - \sum_{i=1}^k \mathbf{x}_i(\mathbf{x}_i^{\mathrm{T}}A)||_{\mathrm{F}}^2 = ||A||_{\mathrm{F}}^2 - \sum_{i=1}^k (A^{\mathrm{T}}\mathbf{x}_i)^{\mathrm{T}}(A^{\mathrm{T}}\mathbf{x}_i). \tag{II.5}$$

In particular the best k-rank approximation of A is given by $\sum_{i=1}^k \mathbf{u}_i(\mathbf{u}_i^{\mathrm{T}}A)$, where $\mathbf{u}_1, \ldots, \mathbf{u}_k \in \mathbb{R}^m$ is an orthonormal sets of the left singular vectors of A corresponding to $\sigma_1, \ldots, \sigma_k$.

The proof of this formula follows straightforward, using the equalities $\mathrm{trace}\ \mathbf{x}\mathbf{y}^{\mathrm{T}} = \mathbf{x}^{\mathrm{T}}\mathbf{y} = \mathbf{y}^{\mathrm{T}}\mathbf{x}$ and the orthonormality of $\mathbf{x}_1, \ldots, \mathbf{x}_k$:

$$||A - \sum_{i=1}^k \mathbf{x}_i(\mathbf{x}_i^{\mathrm{T}}A)||_{\mathrm{F}}^2 = \mathrm{trace}\ (A - \sum_{i=1}^k \mathbf{x}_i(\mathbf{x}_i^{\mathrm{T}}A))^{\mathrm{T}}$$
$$(A - \sum_{i=j}^k \mathbf{x}_j(\mathbf{x}_j^{\mathrm{T}}A)) = \mathrm{trace}\ A^{\mathrm{T}}A -$$
$$2\sum_{i=1}^k (A^{\mathrm{T}}\mathbf{x}_i)^{\mathrm{T}}(A^{\mathrm{T}}\mathbf{x}_i) + \sum_{i=1}^k (A^{\mathrm{T}}\mathbf{x}_i)^{\mathrm{T}}(A^{\mathrm{T}}\mathbf{x}_i).$$

The next theorem is the key theorem for updating the k-rank approximation $\sum_{i=1}^k \mathbf{x}_i(\mathbf{x}_i^{\mathrm{T}}A)$ of A, for some $[\mathbf{x}_1, \ldots, \mathbf{x}_k] \in \mathrm{O}_{mk}(\mathbb{R})$. One of the main ingredients of this theorem is the *modified Gram-Schmidt algorithm*, denoted by *MGSA*. MGSA obtains an orthonormal system of vectors $\mathbf{x}_1, \ldots, \mathbf{x}_p$ from any set of $l \geq p$ vectors $\mathbf{z}_1, \ldots, \mathbf{z}_l$. In each intermediate step of MGSA one has j vectors $\mathbf{x}_1, \ldots, \mathbf{x}_i, \mathbf{w}_{i+1}, \ldots, \mathbf{w}_j$, where $\mathbf{x}_s$ is a vector of length 1 which is orthogonal to all other vectors in this sequence

for $s = 1, \ldots, i$ and $\mathbf{w}_{i+1}, \ldots, \mathbf{w}_j$ are nonzero vectors. Then

$$\mathbf{x}_{i+1} := \frac{1}{\sqrt{\mathbf{w}_{i+1}^{\mathrm{T}}\mathbf{w}_{i+1}}}\mathbf{w}_{i+1},$$
$$\tilde{\mathbf{w}}_t := \mathbf{w}_t - (\mathbf{x}_{i+1}^{\mathrm{T}}\mathbf{w}_t)\mathbf{x}_{i+1} \text{ for } t = i+2, \ldots, j.$$

The new set of vectors $\mathbf{w}_{i+2}, \ldots, \mathbf{w}_{j'}$ are obtained from $\tilde{\mathbf{w}}_{i+2}, \ldots, \tilde{\mathbf{w}}_j$ by deleting all zero vectors in this sequence. The MGSA needs more iterations than the *Gram-Schmidt algorithm*, but is numerically stable [9].

Theorem 2.4: Let $\mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathbb{R}^m$, be an orthonormal system in $\mathbb{R}^m$. Let $\mathbf{w}_1, \ldots, \mathbf{w}_l \in \mathbb{R}^m$, be a given set in $\mathbb{R}^m$. Perform the MGSA on $\mathbf{x}_1, \ldots, \mathbf{x}_k, \mathbf{w}_1, \ldots, \mathbf{w}_l$, to obtain an orthonormal set $\mathbf{x}_1, \ldots, \mathbf{x}_p \in \mathbb{R}^m$, where $k \le p \le k + l$. Assume that $k < p$, i.e. $\mathrm{span}(\mathbf{w}_1, \ldots, \mathbf{w}_l) \nsubseteq \mathrm{span}(\mathbf{x}_1, \ldots, \mathbf{x}_k)$. Form $p \times p$ real symmetric matrix $S := [(A^{\mathrm{T}}\mathbf{x}_i)^{\mathrm{T}}(A^{\mathrm{T}}\mathbf{x}_j)]_{i,j=1}^p$, and assume that $\lambda_1 \ge \ldots \ge \lambda_k$ are the k-largest eigenvalues of S with the corresponding k-orthonormal vectors $\mathbf{o}_1, \ldots, \mathbf{o}_k \in \mathbb{R}^p$. Let $O := [\mathbf{o}_1, \ldots, \mathbf{o}_k] \in \mathrm{O}_{pk}(\mathbb{R})$ and define k-orthonomal vectors $\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_k \in \mathbb{R}^m$ as follows:

$$[\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_k] = [\mathbf{x}_1, \ldots, \mathbf{x}_p]O. \tag{II.6}$$

Then

$$\sum_{i=1}^k (A^{\mathrm{T}}\mathbf{x}_i)^{\mathrm{T}}(A^{\mathrm{T}}\mathbf{x}_i) \le \sum_{i=1}^k (A^{\mathrm{T}}\tilde{\mathbf{x}}_i)^{\mathrm{T}}(A^{\mathrm{T}}\tilde{\mathbf{x}}_i). \tag{II.7}$$

Furthermore

$$\lambda_i = (A^{\mathrm{T}}\tilde{\mathbf{x}}_i)^{\mathrm{T}}(A^{\mathrm{T}}\tilde{\mathbf{x}}_i), \quad i = 1, \ldots, k. \tag{II.8}$$

We now explain the essence of Theorem 2.4. View $\mathbf{x}_1, \ldots, \mathbf{x}_k$ as an approximation to the first k-left singular vectors of A, and $\sum_{i=1}^k \mathbf{x}_i(A^{\mathrm{T}}\mathbf{x}_i)^{\mathrm{T}}$ as a k-rank approximation to A. Hence $(A^{\mathrm{T}}\mathbf{x}_i)^{\mathrm{T}}(A^{\mathrm{T}}\mathbf{x}_i)$ is an approximation to σ_i^2 of A for $i = 1, \ldots, k$. Read additional vectors $\mathbf{w}_1, \ldots, \mathbf{w}_l \in \mathbb{R}^m$ such that at least one of this vectors is not in the subspace spanned by $\mathbf{x}_1, \ldots, \mathbf{x}_k$. Let $\mathbf{X}$ be the subspace spanned by $\mathbf{x}_1, \ldots, \mathbf{x}_k$ and $\mathbf{w}_1, \ldots, \mathbf{w}_l$. Hence $\mathbf{x}_1, \ldots, \mathbf{x}_k, \ldots, \mathbf{x}_p$ is the orthonormal basis of $\mathbf{X}$ obtained from the vectors $\mathbf{x}_1, \ldots, \mathbf{x}_k, \mathbf{w}_1, \ldots, \mathbf{w}_l$, using the MGSA. Note that $k < p \le k+l$, and in general one has that $p = k+l$. Consider the $p \times p$ nonnegative definite matrix $S = [(A^{\mathrm{T}}\mathbf{x}_i)^{\mathrm{T}}(A^{\mathrm{T}}\mathbf{x}_j)]_{i,j=1}^p$. Find its first k eigenvectors $\mathbf{o}_1, \ldots, \mathbf{o}_k \in \mathbb{R}^p$ and obtain $\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_k \in \mathbb{R}^m$ using (II.6). Then $C := \sum_{i=1}^k \tilde{\mathbf{x}}_i(A^{\mathrm{T}}\tilde{\mathbf{x}}_i)^{\mathrm{T}}$ is the best approximation of A by matrix B of rank k at most, whose columns are in the subspace $\mathbf{X}$. In particular, the approximation C is better than the previous approximation $\sum_{i=1}^k \mathbf{x}_i(A^{\mathrm{T}}\mathbf{x}_i)^{\mathrm{T}}$, which is equivalent to the (II.7).

Outline of Proof of Theorem 2.4. Let $S = [s_{ij}]_{i,j=1}^p$. Let $\mathbf{e}_i = (\delta_{i1}, \ldots, \delta_{ip})^{\mathrm{T}}, i = 1, \ldots, p$ be the standard orthonormal basis in $\mathbb{R}^p$. Use the definition of A and Ky Fan characterization of the sum of the maximal k eigenvalues of symmetric S to deduce

$$\sum_{i=1}^k (A^{\mathrm{T}}\mathbf{x}_i)^{\mathrm{T}}(A^{\mathrm{T}}\mathbf{x}_i) = \sum_{i=1}^k \mathbf{e}_i^{\mathrm{T}}S\mathbf{e}_i \le \sum_{i=1}^k \lambda_i = \sum_{i=1}^k \mathbf{o}_i^{\mathrm{T}}S\mathbf{o}_i.$$

Let $C := A^{\mathrm{T}}[\mathbf{x}_1 \ldots \mathbf{x}_p]$. Then $S = C^{\mathrm{T}}C$. Hence the $\sqrt{\lambda_1} \ge \ldots \ge \sqrt{\lambda_p}$ are the singular values of C and $\mathbf{o}_1, \ldots, \mathbf{o}_p$ are the right singular vectors of C. Thus $C\mathbf{o}_i = \sqrt{\lambda_i}\mathbf{t}_i \in \mathbb{R}^n$, where $\mathbf{t}_i$ is the left singular vector of C corresponding to the singular value $\sqrt{\lambda_i}$ for $i = 1, \ldots, p$. A straightforward calculation shows that $\lambda_i = \mathbf{o}_i^{\mathrm{T}}S\mathbf{o}_i = (A^{\mathrm{T}}\tilde{\mathbf{x}}_i)^{\mathrm{T}}(A^{\mathrm{T}}\tilde{\mathbf{x}}_i)$ for $i = 1, \ldots, k$, which the equalities in (II.8). Hence (II.7) holds. □

III. ALGORITHM

One starts the algorithm by choosing the first k-rank approximation to A as follows. Let $\mathbf{c}_1, \ldots, \mathbf{c}_n \in \mathbb{R}^m$ be the n columns of A. Choose k integers $1 \le n_1 < \ldots < n_k \le n$. Let $\mathbf{x}_1, \ldots, \mathbf{x}_q \in \mathbb{R}^m$ be the orthonormal set obtained from $\mathbf{c}_{n_1}, \ldots, \mathbf{c}_{n_k}$ using the MGSA. Set

$$B_0 := \sum_{i=1}^q \mathbf{x}_i(A^{\mathrm{T}}\mathbf{x}_i)^{\mathrm{T}}. \tag{III.9}$$

In general $q = k$, but in some cases if $\mathbf{x}_1, \ldots, \mathbf{x}_k$ are linearly dependent, $q < k$. Assume for simplicity of the exposition that $q = k$. Then B_0 is of the form (I.1) where $\mathbf{y}_i = A^{\mathrm{T}}\mathbf{x}_i, i = 1, \ldots, k$.

Now update iteratively k-rank approximation of B_{t-1} of A to B_t, using Theorem 2.4, by letting $\mathbf{w}_1 := \mathbf{c}_{j_1}, \ldots, \mathbf{w}_l := \mathbf{c}_{j_l} \in \mathbb{R}^m$, for some l integers $1 \le j_1 < \ldots < j_l \le n$. That is, we choose another l sets of columns of A, preferably that were not chosen before, and update the k-rank approximation using the algorithm suggested by Theorem 2.4 to obtain an improved k-rank approximation B_t of A.

Furthermore one can use the k-rank approximation B_t from the above algorithm to approximate the first k-singular values $\sigma_1, \ldots, \sigma_k$, and the left and the right singular vectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$ and $\mathbf{v}_1, \ldots, \mathbf{v}_k$ as follows. First, the square roots $\sqrt{\lambda_1(S)}, \ldots, \sqrt{\lambda_k(S)}$ of the matrix S are approximations for $\sigma_1, \ldots, \sigma_k$. Second, the vectors $\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_k$ approximate $\mathbf{u}_1, \ldots, \mathbf{u}_k$. Let $\tilde{\mathbf{v}}_i := A^{\mathrm{T}}\tilde{\mathbf{x}}_i, i = 1, \ldots, k$. Then $||\tilde{\mathbf{v}}_i|| = \sqrt{\lambda_i(S)}$ for $i = 1, \ldots, k$. Third, the renormalized $\tilde{\mathbf{v}}_i$ which are given as $\frac{1}{||\tilde{\mathbf{v}}_i||}\tilde{\mathbf{v}}_i$ approximate the right singular eigenvectors $\mathbf{v}_i$ for $i = 1, \ldots, k$.

Fast k-rank approximation and SVD algorithm

Input: positive integers m, n, k, l, N, $m \times n$ matrix $\mathbf{A}$, $\epsilon > 0$.

Output: an $m \times n$ k-rank approximation B_f of A, with the ratios $\frac{||B_0||}{||B_t||}$ and $\frac{||B_{t-1}||}{||B_t||}$, approximations to k-singular values and k left and right singular vectors of A.

1. Choose k-rank approximation B_0 using k columns, (or rows), of A.

2. **for** $t = 1$ **to** N

- Select l columns, (or rows), from A at random and update B_{t-1} to B_t.

- Compute the approximations to k-singular values, and k left and right singular vectors of A.

- If $\frac{||B_{t-1}||}{||B_t||} > 1 - \epsilon$ let $f = t$ and finish.

We now explain briefly the main steps of our algorithm. We read the dimensions m, n of the data matrix A. We set N as the maximal number of iterations that we are going to execute to find the k-rank approximation of A. We read the entries of the data matrix A, and finally the small parameter $\epsilon > 0$. We choose the k-rank approximation B_0 using (III.9). Assume that B_{t-1} is the current k-rank approximation to A. Next we choose additional l columns of A and update B_{t-1} to B_t using Theorem 2.4 as explained in the previous section. Recall that $||B_{t-1}|| \le ||B_t||$. If the relative improvement in $||B_t||$ is less than ϵ, i.e. $\frac{||B_{t-1}||}{||B_t||} > 1 - \epsilon$, we are satisfied with the approximation B_t and finish our algorithm. If this does not happen then our algorithm stops after the N iteration.

IV. COMPLEXITY OF THE ALGORITHM

We assume here that $m \geq n$ and we apply our algorithm to the randomly selected columns of A. (The second possibility is to apply our algorithm to the rows of A.) The first step of our algorithm is to find an orthonormal basis $\mathbf{x}_1, \ldots, \mathbf{x}_q$ of the subspace spanned by k randomly chosen columns of A. The MGSA needs k^2m flops assumed the worst, (generic), case $q = k$. See [9]. Let $\mathbf{y}_i := A^{\mathrm{T}}\mathbf{x}_i \in \mathbb{R}^n$ for $i = 1, \ldots, k$. The computations of $\mathbf{y}_1, \ldots, \mathbf{y}_k$ needs kmn operations and can be parallelized. $B_0 = \sum_{i=1}^{k} \mathbf{x}_i \mathbf{y}_i^{\mathrm{T}}$ and does not have to be computed.

To find B_1 we choose l columns $\mathbf{w}_1, \ldots, \mathbf{w}_l$ of A at random. Then we perform the MGSA on $\mathbf{x}_1, \ldots, \mathbf{x}_k, \mathbf{w}_1, \ldots, \mathbf{w}_l$ to find an orthonormal system $\mathbf{x}_1, \ldots, \mathbf{x}_p$, with $p \in [k, k+l]$. This step requires $m(k+l)^2$ flops. Let $\mathbf{y}_i = A^{\mathrm{T}}\mathbf{x}_i$ for $i = k+1, \ldots, p$. This step needs at most lmn flops and can be parallelized. Next we have the following two choices. First possibility is to find the SVD of $C_1 = [\mathbf{y}_1 \ldots \mathbf{y}_p]$. This needs at most $O((k+l)^2 n)$ flops. Second possibility is to compute the matrix $S_1 = C_1^{\mathrm{T}} C_1$ given in Theorem 2.4. This needs $nl(l+2k+1)/2$ flops. Then we compute the spectral decomposition of S_1 which needs $O((k+l)^3)$ flops. If $k+l$ is much smaller than n, it seems to us that the second method is more efficient.

The first k right singular vectors of C_1 are identical to the first k eigenvectors of S. Use these k vectors, as explained in Theorem 2.4, to update $\mathbf{x}_1, \ldots, \mathbf{x}_k$ and $\mathbf{y}_1, \ldots, \mathbf{y}_k$. This needs $kp(m+n)$ flops.

To update B_{t-1} to B_t for $t \geq 1$ require the same number of flops as to compute B_1. Hence, the most intensive part of the computation lies in the computation of the spectral decomposition S_t, which is $O((k+l)^3)$. Assuming that $l = O(k)$, we deduce that the intensive part of the computation is of order $O(k^3)$. The simulations that we performed show that we need a small number of iterations, (around 5), to get a very good approximation to the best k-rank approximation of A. Hence our algorithm is expected to converge in $O(kmn)$ steps.

V. SIMULATION RESULTS

To assess the performance of our k-rank approximation algorithm we conducted different simulation on synthetic data and images. We also implemented our algorithm for three different sampling methods:

1) Uniform sampling without replacement, abbreviated here *usnr*, i.e. we removed the sampled rows after sampling. Thus each row can be sampled at most once.
2) Uniform sampling with replacement, abbreviated here *usr*, i.e. we did not remove the sampled rows after sampling. Thus some rows were sampled more than once.
3) Weighted sampling, abbreviated here *ws*. Here the weight of each row was dependent on the the gradient image. Thus a row had a bigger weight if it differed more from its neighbor.

To measure the approximation error we defined the relative error of the approximation as $\| A - B_t \|_{\mathrm{F}}^2 \; / \; \| A \|_{\mathrm{F}}^2$, where A is the original data matrix and B_t is k-rank approximation to A given by our algorithm. In Figures 1-5 we show the convergence of the relative error $\|A - B_t\|_{\mathrm{F}}^2 / \|A\|_{\mathrm{F}}^2$ to the optimal error $\|A - A_k\|_{\mathrm{F}}^2 / \|A\|_{\mathrm{F}}^2$ where A_k is the optimal true k-rank approximation to matrix A, as a function of iteration parameter N.

Figure 1 shows the convergence results for the real images of Cameraman picture, (from Image Processing Toolbox of Matlab), which is given by $A \in \mathbb{R}^{256 \times 256}$. We chose here $k = 80$ and $l = 10$. Note here that *usnr* performs exceptionally well for the first 5 iterations. Actually, the third iteration of *usnr* gives a very good approximation to the optimal solution A_{80}.

Figure 2 shows the plots of the relative error versus total number of sampled rows in each method of sampling for the Cameraman. From these plots it is clear that *usnr* is the best, *ws* is the second and *usr* is the weakest.

Figure 3 shows the convergence results for the real images of Liftingbody picture, (from Image Processing Toolbox of Matlab), which is given by $A \in \mathbb{R}^{512 \times 512}$. We chose here $k = 100$ and $l = 15$. Again *usnr* performs the best, and *ws* performs almost as good as *usnr*. 5 iterations of *usnr* gives a reasonably good approximation to the optimal solution A_{100}.

Figure 4 shows the plots of the relative error versus total number of sampled rows in each method of sampling for the Liftingbody. This time *usnr* and *ws* have comparable performance, while *usr* is slightly weaker.

Figure 5 shows the performance of our algorithm for a randomly generated data matrix $A \in \mathbb{R}^{3000 \times 500}$ with rank 500, the maximal possible rank. To obtain such A, we chose 500 vectors $\mathbf{x}_i \in \mathbb{R}^{3000}, \mathbf{y}_i \in \mathbb{R}^{500}$, where the entries of each column are drawn at random from the interval $[-1, 1]$ using the uniform distribution. Then $A = \sum_{i=1}^{500} \mathbf{x}_i \mathbf{y}_i^{\mathrm{T}}$. We were looking for rank $k = 100$ approximation of A. In each iteration we randomly picked $l = 100$ rows of the data matrix with and without replacement. For comparison the optimum relative error has been pointed out on the axis. (It is located at the level $\approx 0.95 \times 10^{-4}$). As we expected the convergence property is faster when the rows are sampled without replacement. We see that 5 iterations give a very good approximation to the optimal solution A_{100}.

Needless to say that the importance of this iterative algorithm can be observed when it is applied to the big matrices of data where the SVD of the data matrix is often impossible to compute. In this case the simple randomized SVD methods [8] may not be fast, since it needs to sample large enough number of rows to give the acceptable error bound on the approximation. To highlight this feature of our algorithm we applied our algorithm on a synthetic full rank data matrix of 8000×200. We chose the parameter l, N such that the relative error to be less than 2 times of optimal relative error. We observed that the speed up of our algorithm, which is the rate of the time needed by deterministic SVD to compute 100-rank approximation to the time needed by our algorithm to compute B_t, is 42 in our system. The results for three more data sets with different values of k has also been illustrated in Table I. Due to system limitation and the comparison issue we were not able to show the speed up for bigger matrices where deterministic SVD (implemented in Matlab) may fail to compute the SVD of the matrix. However, our algorithm can always compute the k-rank approximation of the matrix as long as $k+l$ is not too big since the algorithm in each iteration deals with small matrices.

VI. CONCLUSIONS

We have proposed a novel approach for fast computing of k-rank approximation of a given $m \times n$ data matrix, using Monte-Carlo method by choosing at random l columns or rows of A. The advantage of our algorithm is that we guarantee that every iteration improves the quality of our approximation. We applied our algorithm on images data and as well as on synthetic data matrices. Our results confirm the convergence of the relative errors of the approximation to the optimal relative error. To highlight the important feature of this algorithm we applied this method on a big matrix of randomly generated data. We observed for the reasonable level of relative error the algorithm is much faster than optimal k-rank approximation using deterministic SVD, which may also fail to compute the SVD for big matrices. We believe that this algorithm will have many applications in data mining, data storage and data analysis, where dealing with high dimensional data is a major problem.

VII. REFERENCES

[1] C.C. Aggrawal, C.M. Procopiuc, J.L. Wolf, P.S. Yu and J.S. Park, Fast algorithms for projected clustering, *Proc. of ACM SIGMOD Intl. Conf. Management of Data* 1999, 61-72.

[2] R. Agrawal, J.Gerhrke, D.Gunopulos, and P. Raghavan, Automatic subspace clustering of high dimensional data for data mining applications, *Proc. ACM SIGMOD Conf. on Management of Data*, 1998, 94-105.

[3] P. Drineas, A. Frieze, R. Kannan, S. Vempala and V. Vinay, Clustering large graphs via the singular value decomposition, *Journal of Machine Learning*, 56 (2004), 9-33.

[4] M. Ester, H.-P. Krieger, J. Sander and X.Xu, A density-based algortihm for discovering clusters in large spatial databases with nose, *Proc. 2nd Intl. Conf. Knowledge Discovery and Data Mining*, 1996, 226-231.

[5] S. Friedland, Inverse eigenvalue problems, *Linear Algebra Appl.* 17 (1977), 15-51.

[6] S. Friedland, M. Kaveh, A. Niknejad and H. Zare, An Algorithm for Missing Value Estimation for DNA Microarray Data, *Proceedings of ICASSP 2006*, Toulouse, France, 4 pp..

[7] S. Friedland, A. Niknejad and L. Chihara, A simultaneous reconstruction of missing data in DNA microarrays, *Linear Algebra Appl.*, to appear.

[8] A. Frieze, R. Kannan and S. Vempala, Fast Monte-Carlo algorithms for finding low rank approximations, *Proceedings of the 39th Annual Symposium on Foundation of Computer Science*, 1998.

[9] G.H. Golub and C.F. Van Loan, *Matrix Computation*, Johns Hopkins Univ. Press, 3rd Ed., 1996.

[10] C.M. Procopiuc, P.K. Agarwal, M. Jones and T.M. Murali, A Monte Carlo algorithm for fast projective clustering, *Proc. of ACM SIGMOD Intl. Conf. Management of Data* 2002.

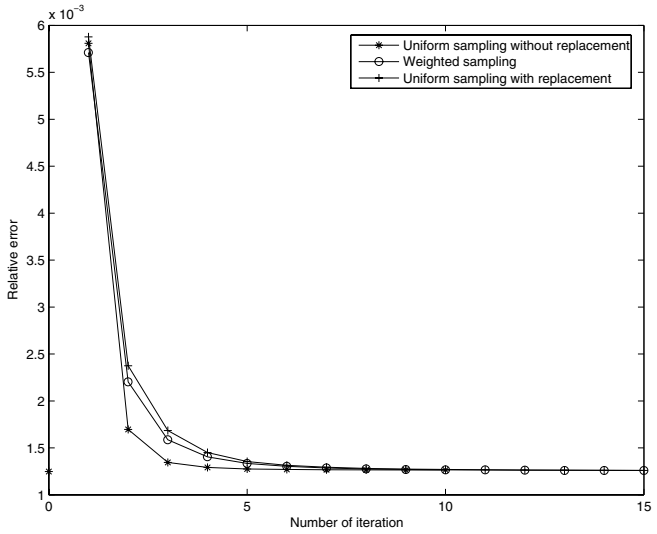

Fig. 1. Convergence property of the Monte-Carlo method for Cameraman image(256×256), $k = 80$.

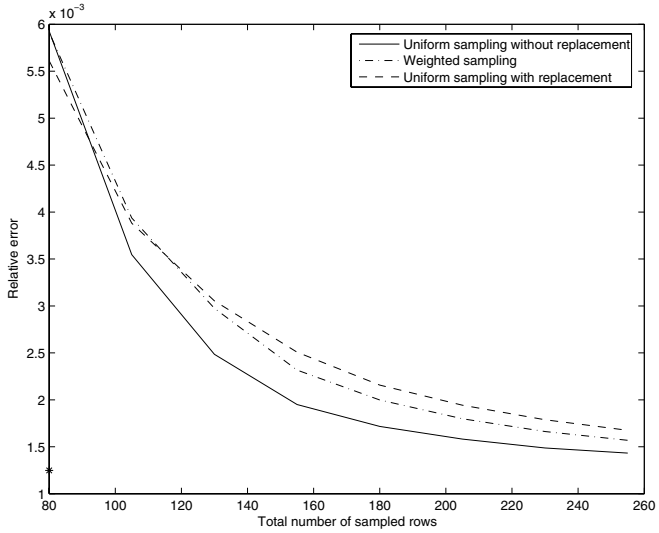

Fig. 2. Cameraman: Relative error versus total number of sampled rows, $k = 80$.

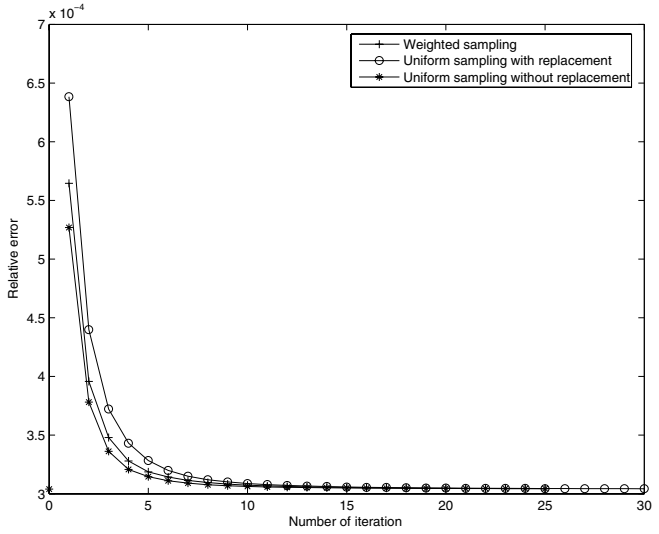

Fig. 3. Convergence property of the Monte-Carlo method for Liftingbody image(512×512), $k = 100$.

Table I. Comparison of relative error and speed up of our algorithm with optimum k-rank approximation algorithm

Data sets	Speed up	Re. ratio
Cameraman(256×256), $k = 80$	1.145	1.083
Liftingbody (512×512), $k = 100$	8	1.08
Map image(627×865) $k = 200$	3.33	1.067
Random matrix(8000×200) $k = 100$	42	1.1

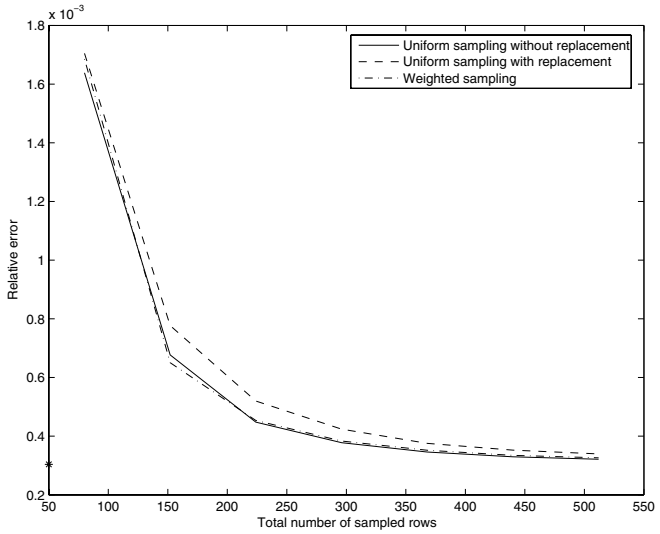

Fig. 4. Liftingbody: relative errors versus total number of sampled rows, $k = 100$

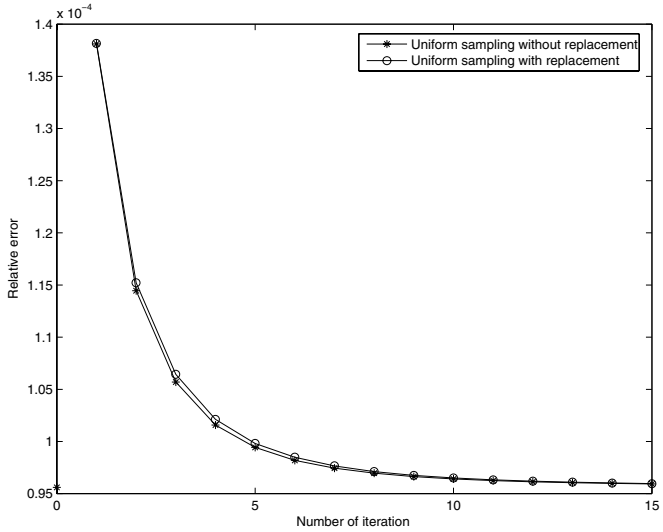

Fig. 5. Convergence property of the Monte-Carlo method for random data matrix(3000×500).