Advances in Computer Science and its Applications (ACSA)
Vol. 2, No. 1, 2012, ISSN 2166-2924

www.worldsciencepublisher.org

Approximation theory of matrices based on its low ranking and stochastic computation

Behrouz Fathi Vajargah, Ateyeh Jahanbin

Department of Mathematics, University of Guilan, Iran

fathi@guilan.ac.ir, at.jahanbin@yahoo.com

Abstract: This work presents that the way to use the successful approximation of a desired matrix based on stochastic and SVD algorithms and compared their results together.

Key words. Low rank; Randomized algorithm; Monte Carlo method; Singular value decomposition; Matrix multiplication.

2000 Mathematical Subject: 65F30

1. Introduction. In computations, the data may consist of an $m \times n$ matrix A Then, it is often of interest to evaluate a low-rank approximation to$A$, i.e., an approximation D to the matrix A of rank not bigger than a specified rank k , where k is smaller than m and n Methods such as the singular value decomposition (SVD) can be employed to find an approximation to A which is the best in a well-defined sense [9].
Suppose A and B which are $m \times n$ and $n \times p$, respectively are the two input matrices. We perform c independent trials, where in each trial we randomly sample an element of $\{1, 2, . . . , n\}$ with an appropriate probability distribution P on {1, 2, . . . , n}. We form an $m \times c$ matrix C consisting of the sampled columns of A, each scaled appropriately, and we form a $c \times n$ matrix R using the corresponding rows of B, again scaled appropriately. The choice of P and the column and row scaling are crucial features of the algorithm. When these are chosen, we show that CR is a good approximation to AB. More precisely, we show that

$$\|AB - CR\|_F = O\big(\|A\|_F \|B\|_F / \sqrt{c}\big)$$

where $\|.\|_F$ denotes the Frobenius norm, i.e.,$\|A\|_F^2 = \sum_{i,j} A_{ij}^2$. This algorithm can be implemented without storing the matrices A and B in RAM, provided it can make two passes over the matrices stored in external memory and use $O\big(c(m + n + p)\big)$ additional RAM to construct C and R [2].

We are interested in developing and analyzing fast Monte Carlo algorithms for performing useful computations on large matrices. In this paper we consider the singular value decomposition (SVD) ; based on two related papers [5,6] we consider matrix multiplication and a new method for computing a compressed approximate decomposition of a large matrix. In this paper, we present a computational model for computing on massive data sets (the pass-efficient model) in which our algorithms may naturally be formulated; we also present algorithm for the approximation of the product of two matrices. Also, we present two algorithms for the computation of low-rank approximations to a matrix [4].

Recent interest in computing with massive data sets has led to the development of computational models in which the usual notions of time efficiency and space efficiency have been modified [12,11,1,9,8,3]. In the applications that motivate these data streaming models[11,3], e.g. the observational sciences and the monitoring and operation of large networked systems, the data sets are much too large to fit into main memory.

Definition: For a vector $x \epsilon \mathbb{R}^n$ we let x_i , i = 1, . . . , n, denote the *ith* element of x and we let $|x| = (\sum_{i=1}^{n} |x_i|^2)^{\frac{1}{2}}$. For a matrix $A \epsilon \mathbb{R}^{m\times n}$ we let, $A^{(j)}$, j = 1, . . . , n, denote the *j*th column of A as a column vector and, $A_{(i)}$ i = 1, . . . ,m, denote the *ith* row of A as a row vector; thus, if A_{ij} denotes the (i, j) *th* element of A. The range of an $A \epsilon \mathbb{R}^{m\times n}$ is

$$\text{range(A)} = \{y \epsilon \mathbb{R}^m \text{: y = Ax for some x } \epsilon \mathbb{R}^n\} = \text{span}(A^{(1)}, \cdots, A^{(n)}).$$

The rank of A, $rank(A)$, is the dimension of $rang(A)$ and is equal to the number of linearly independent columns of A ; since this is equal to rank(A^T) it also equals the number of linearly independent rows of A. The null space of A is

$$\text{null(A)} = \{x \in \mathbb{R}^n : Ax = 0\}.$$

For a matrix $A \in \mathbb{R}^{m\times n}$ we interest will be the Frobenius norm, which is defined by

$$\|A\|_F = \left(\sum_{i=1}^{m}\sum_{j=1}^{n} A_{ij}^2\right)^{\frac{1}{2}}$$

If $Tr(A)$ is the matrix trace which is the sum of the diagonal elements of A, then

$$\|A\|_F^2 = Tr(A^T A) = Tr(AA^T).$$

THE SINGULAR VALUE DECOMPOSITION(SVD)

If $A \in \mathbb{R}^{m\times n}$, then there exist orthogonal matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ and $U^T AV = \Sigma$ and Σ is a diagonal matrix $m\times n$ such that

$$U^T AV = \Sigma = diag(\sigma_1, \sigma_2, \cdots, \sigma_\rho) \quad , \quad \rho = min\{m, n\}$$

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_\rho \geq 0$$

Then the SVD decomposition of A is given as $A = U\Sigma V^T$.
The vectors u^i, 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:

$$Av^i = \sigma_i u^i \qquad A^T u^i = \sigma_i v^i$$

Equivalently, $\sigma_1^2 \geq \sigma_2^2 \geq \cdots \geq \sigma_r^2 > 0$ are all the positive eigenvalues of the nonnegative definite symmetric matrices AA^T, $A^T A$, with the corresponding orthonormal eigenvectors $u_1, u_2, \cdots, u_r \in \mathbb{R}^m$ and $v_1, v_2, \cdots, v_r \in \mathbb{R}^n$.
we define r by $\sigma_1 \geq \sigma_2 \geq \cdot\ \cdot\ \cdot \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_\rho = 0$, then rank(A) = r,
$null(A) = span(v^{r+1}, \cdots, v^\rho)$ and $rang(A) = span(u^1, \cdots, u^r)$

If we let $U_r \in \mathbb{R}^{m\times r}$ denote the matrix consisting of the first r columns of U, $V_r \in \mathbb{R}^{r\times n}$ denote the matrix consisting of the first r columns of V , and $\Sigma_r \in \mathbb{R}^{r\times r}$ denote the principal $r \times r$ sub matrix of Σ, then

$$A = U_r \Sigma_r V_r^T = \sum_{t=1}^{r} \sigma_t u^t {v^t}^T$$

Note that this decomposition property provides a canonical description of a matrix as a sum of r rank-one matrices of decreasing importance. If $k \leq r$ and we define

$$A_k = U_k \Sigma_k V_k^T = \sum_{t=1}^{k} \sigma_t u^t {v^t}^T, \quad A_k = U_k U_k^T A = \left(\sum_{t=}^{k} u^t {u^t}^T\right) A \ ,$$
$$A_k = AV_k V_k^T = A\left(\sum_{t=1}^{k} v^t {v^t}^T\right)$$

A_k is the projection of A onto the space spanned by the top k singular vectors of A.
Furthermore, the distance between A and any rank-k approximation to A is minimized by, A_k i.e.,

$$\min_{D \in R^{m\times n}: rank(D) \leq k} \|A-D\|_F^2 = \|A-A_k\|_F^2 = \sum_{t=k+1}^{r} \sigma_t^2(A)$$

$$\|A_k\|_F^2 = \sum_{t=1}^{k} \sigma_t^2 \qquad , \qquad \|A\|_F^2 = \sum_{t=1}^{r} \sigma_t^2.$$

APPROXIMATING MATRIX MULTIPLICATION

Recall that for $A\epsilon\mathbb{R}^{m\times n}$and $B\epsilon\mathbb{R}^{n\times p}$, the product AB may be written as the sum of n rank-one matrices

$$AB = \sum_{t=1}^{n} A^{(t)}B_{(t)}$$

From this, a simple algorithm for approximate matrix multiplication suggests itself : pick a random subset of c columns of A to form an $m \times c$ matrix C; form an $c \times p$ matrix R out of the corresponding columns of B. Then, intuitively, that the product CR is an estimator (entry by entry) of the product AB.
We can sample the columns of A so that column i is chosen with probability P_i satisfying

$$P[i = k] = P_k \ , k = 1 \cdots n$$

$$P_k = \frac{|A^{(k)}||B_{(k')}|}{\sum_{k'=1}^{n}|A^{(k')}||B_{(k')}|}$$

$\boldsymbol{A.M.M.\ Algorithms}$:

Input: $A\epsilon\mathbb{R}^{m\times n}, B\epsilon\mathbb{R}^{n\times p}, c\epsilon\mathbb{Z}^{+}$ $\mathit{such\ that}\ 1 \le c \le n$, $\mathit{and}\ \{P_i\}_{i=1}^{n}$ $\mathit{such\ that}$
$P_i \ge 0$ $\mathit{and}\ \sum_{i=1}^{n} P_i = 1$
Output: $C\epsilon\mathbb{R}^{m\times c}$and $R\epsilon\mathbb{R}^{c\times p}$.
1. For t = 1 to c,
(a) Pick $i_t\epsilon\{1, \ldots, \mathrm{n}\}$ with **Pr** [i_t= k] =P_k, k = 1, . . . , n, independently
and with replacement.
(b) Set $C^{(t)}= A^{(i_t)}/\sqrt{cP_{i_t}}$ $\mathit{and}\ R_{(t)} = B_{(i_t)}/\sqrt{cP_{i_t}}$
2. Return C,R

When this algorithm is given as input two matrices A and B, a probability distribution $\{P_i\}_{i=1}^{n}$, and a number c of column-row pairs to choose, it returns as output matrices C and R such that the product CR is an approximation to AB. Observe that since

$$CR = \sum_{t=1}^{c} C^{(t)}R_{(t)} = \sum_{t=1}^{c} \frac{1}{cP_{(i_t)}} A^{(i_t)}B_{(i_t)}$$

Definition: The sequence of matrices$\{C^{(m)}R_{(m)}\}$ m=1,2,… where they are independent and identically distribution as CR we call the$C^{(1)}R_{(1)}, C^{(2)}R_{(2)}, \cdots$ perform a sequence of realized simulated matrices of CR.

Definition: For m=1,2,…,L $C^{(m)}R_{(m)}$ independent realized stochastic matrices with property $E\big(C^{(m)}R_{(m)}\big) = AB$ i.e. $C^{(m)}R_{(m)}$ with the same distribution as matrix CR where $E\big(C^{(m)}R_{(m)}\big) = AB$, then $G = \frac{1}{L}\sum_{m=1}^{L} C^{(m)}R_{(m)}$ is called the Monte Carlo simulated matrix of CR.

Implementation of the sampling and running time. To implement the $A.M.M$ algorithm, it must be decided which elements of the input to sample and those elements must then be sampled. In the case of uniform sampling one can decide before the input is seen which column-row pairs to sample. Then, a single pass over the matrices is sufficient to sample the columns and rows of interest and to construct C and R; this requires $O(c(m+ p))$ additional time and space. We will see below that it is useful to sample according to a non uniform probability distribution that depends on column and row lengths, e.g.
In order to decide which column-row pairs to sample in such a case, one pass through the matrices and $O(n)$ additional time and space is sufficient; in the additional space running totals of $|A^{(k)}|^2$and $|B_{(k)}|^2$ are kept, so that after the first pass$|A^{(k)}|$, $|B_{(k)}|$, k = 1, . . . , n, and thus the probabilities, can be calculated in $O(n)$ additional time.
Then in a second pass the columns and rows of interest can be sampled and C and R can be constructed and stored; this requires $O(c(m + p))$ additional space and time. Thus, in addition to either one or two passes over the data, for both uniform and non uniform sampling, $O(c(m+n+p))$ additional space and time is sufficient to sample from the matrices A and B of the input and to construct the matrices C and R.

Lemma 1. Suppose $A\epsilon\mathbb{R}^{m\times n}$, $B\epsilon\mathbb{R}^{n\times p}$, $c\epsilon Z^{+}$such that $1 \leq c \leq n$,and $\{P_i\}_{i=1}^{n}$ are such that $P_i \geq 0$ and $\sum_{i=1}^{n} P_i = 1$. Construct C and R with the $A.M.M$ algorithm, and let CR be an approximation to AB.Then [5]

$$E\left[(CR)_{ij}\right] = (AB)_{ij}$$

$$Var\left[(CR)_{ij}\right] = \frac{1}{c}\sum_{k=1}^{n}\frac{A_{ik}^2 B_{ik}^2}{P_k} - \frac{1}{c}(AB)_{ij}^2 .$$

Lemma 2. Suppose $A\epsilon\mathbb{R}^{m\times n}$, $B\epsilon\mathbb{R}^{n\times p}$, $c\epsilon Z^{+}$such that $1 \leq c \leq n$,and $\{P_i\}_{i=1}^{n}$are such that $P_i \geq 0$ and $\sum_{i=1}^{n} P_i = 1$. Construct C and R with the $A.M.M$ algorithm, and let CR be an approximation to AB. Then [5]

$$E[\|AB - CR\|_F^2] = \sum_{k=1}^{n}\frac{\left|A^{(k)}\right|^2 \left|B_{(k)}\right|^2}{cP_k} - \frac{1}{\mathrm{c}}\|\mathrm{AB}\|_{\mathrm{F}}^2$$

Furthermore, if

$$P_k = \frac{|A^{(k)}||B_{(k)}|}{\sum_{k'=1}^{n}\left|A^{(k')}\right|\left|B_{(k')}\right|}$$

Then

$$E[\|AB - CR\|_F^2] = \frac{1}{c}\left(\sum_{k=1}^{n}\left|A^{(k)}\right|\left|B_{(k)}\right|\right)^2 - \frac{1}{c}\|AB\|_F^2 .$$

We will say that the sampling probabilities $P_k = \left|A^{(k)}\right|\left|B_{(k)}\right| / \sum_{k'=1}^{n}\left|A^{(k')}\right|\left|B_{(k')}\right|$ are the *optimal probabilities* since they minimize $E[\|AB - CR\|_F^2]$, which as Lemma 2 shows is one natural measure of the error. We will say that a set of sampling probabilities $\{P_i\}_{i=1}^{n}$ are *nearly optimal probabilities* if $P_k \geq \beta\left|A^{(k)}\right|\left|B_{(k)}\right| / \sum_{k'=1}^{n}\left|A^{(k')}\right|\left|B_{(k')}\right|$ for some positive constant $\beta \leq 1$.

Theorem 1. *Suppose* $\mathbb{R}^{m\times n}$, $B\epsilon\mathbb{R}^{n\times p}$, $c\epsilon Z^{+}$*such that* $1 \leq c \leq n$, *and* $\{P_i\}_{i=1}^{n}$*are such that* $\sum_{i=1}^{n} P_i = 1$*and such that for some positive constant* $\beta \leq 1$

$$P_k \geq \frac{\beta|A^{(k)}||B_{(k)}|}{\sum_{k'=1}^{n}\left|A^{(k')}\right|\left|B_{(k')}\right|}$$

Construct C and R with the $A.M.M$ algorithm, and let CR be an approximation toAB.
Then [5],

$$E[\|AB - CR\|_F^2] \leq \frac{1}{\beta c}\|A\|_F^2\|B\|_F^2$$

Furthermore, let $\delta \in (0, 1)$ and $\eta = 1 + \sqrt{8/\beta \log(1/\delta)}$ Then, with probability atleast $1 - \delta$,

$$\|AB - CR\|_F^2 \leq \frac{\eta^2}{\beta c}\|A\|_F^2\|B\|_F^2$$

In particular, if $c \geq 1/\beta\varepsilon^2$, then by using Jensen's inequality it follows that

$$E[\|AB - CR\|_F^2] \leq \varepsilon\|A\|_F^2\|B\|_F^2$$

and if, in addition, $c \geq \eta^2/\beta\varepsilon^2$, then with probability at least $1 - \delta$

$$\|AB - CR\|_F \leq \varepsilon \|A\|_F \|B\|_F \,.$$

By taking $B = A^T$ and applying Jensen's inequality, we have the following theorem as a corollary of Theorem 1.

Theorem 2. Suppose $A\epsilon\mathbb{R}^{m\times n}$, $B\epsilon\mathbb{R}^{n\times p}$, $c\epsilon\mathbb{Z}^+$ such that $1 \leq c \leq n$, and $\{P_i\}_{i=1}^n$ are such that $\sum_{i=1}^n P_i = 1$ and such that $P_k \geq \beta |A^{(k)}||B_{(k)}| / \sum_{k'=1}^n |A^{(k')}| \, |B_{(k')}|$ for some positive constant $\beta \leq 1$. Furthermore, let $\delta \in (0, 1)$ and $\eta = 1 + \sqrt{8/\beta \log(1/\delta)}$. Construct C (and $R = C^T$)

with the $A.M.M$ algorithm, and let CC^T be an approximationto AA^T. Then [5],

$$E[\|AA^T - CC^T\|_F] \leq \frac{1}{\sqrt{\beta c}} \|A\|_F^2$$

and with probability at least $1 - \delta$,

$$\|AA^T - CC^T\|_F \leq \frac{\eta}{\sqrt{\beta c}} \|A\|_F^2 \,.$$

Linear Time SVD Algorithm (L.T.SVD)

Input: $A\epsilon\mathbb{R}^{m\times n}, B\epsilon\mathbb{R}^{n\times p}, c, k\epsilon\mathbb{Z}^+$ *such that* $1 \leq k \leq c \leq n$ *, and* $\{P_i\}_{i=1}^n$ *such that*
$P_i \geq 0$ *and* $\sum_{i=1}^n P_i = 1$

Output: $H_k\epsilon\mathbb{R}^{m\times n}$ *and* $\sigma_t(C), t = 1, \cdots, k.$
1. For t = 1 to c,
(a) Pick $i_t \epsilon 1, \ldots, n$ with $Pr\,[i_t = \alpha] = P_\alpha$, $\alpha = 1, \ldots, n$.
(b) Set $A^{(i_t)} / \sqrt{cP_{i_t}}$.
2. Compute $C^T C$ and its SVD; say $C^T C = \sum_{t=1}^c \sigma_t^2(C) y^t {y^t}^T$
3. Compute $h^t = Cy^t / \sigma_t(C)$ for t = 1, . . . , k.
4. Return H_k, where $H_k^{(t)} = h^t$ *, and* $\sigma_t(C)$, t = 1, . . . , k.

Linear time SVD approximation

Given a matrix $A\epsilon\mathbb{R}^{m\times n}$ we wish to approximate its top k singular values and the corresponding singular vectors. The strategy behind the Linear Time SVD algorithm is to pick c columns of the matrix A, rescale each by an appropriate factor to form a matrix $C\epsilon\mathbb{R}^{m\times n}$, and then compute the singular values and corresponding left singular vectors of the matrix C, which will be approximations to the singular values and left singular vectors of A, in a sense we make precise later. These are calculated by performing an SVD of the matrix CC^T to compute the right singular vectors of C and from them calculating the left singular vectors of C.
It will be shown that if the probabilities $\{P_i\}_{i=1}^n$ are chosen judiciously, then the left singular vectors of C are with high probability approximations to the left singular vectors of A.

We will show that in addition to this error the matrix $H_k H_k^T A$ has an error that depends on $\|AA^T - CC^T\|_F$. Then, using the results of Theorem 2, we will show that this additional error depends on $\|A\|_F^2$.

Theorem 3. Suppose $A\epsilon\mathbb{R}^{m\times n}$ and let H_k be constructed from the $L.T.SVD$ algorithm. Then [6],

$$\|A - H_k H_k^T A\|_F^2 \leq \|A - A_k\|_F^2 + 2\sqrt{k}\|AA^T - CC^T\|_F.$$

Theorem 4. Suppose $A\epsilon\mathbb{R}^{m\times n}$; let H_k be constructed from the $L.T.SVD$ algorithm by sampling c columns of A with probabilities $\{P_i\}_{i=1}^n$ such that $p_i \geq \beta |A^{(i)}|^2 / \|A\|_F^2$ for some positive $\beta \leq 1$, and le t. Let $\eta = 1 + \sqrt{8/\beta \log(1/\delta)}$. *let* $\varepsilon > 0$. If $c \geq 4k/\beta\varepsilon^2$, then [6],

$$E[\|A - H_k H_k^T A\|_F^2] \leq \|A - A_k\|_F^2 + \varepsilon \|A\|_F^2$$

and if $c \geq 4k\eta^2/\beta\varepsilon^2$, then with probability at least $1 - \delta$,

$$\|A - H_k H_k^T A\|_F^2 \leq \|A - A_k\|_F^2 + \varepsilon \|A\|_F^2.$$

Implementation details and running time

To measure the approximation error we defined the relative error of the approximation as,$\|A - H_k H_k^T A\|_F^2/\|A\|_F^2$ where A is the original data matrix and $H_k H_k^T A$ is k-rank approximation to A given by L.T.SVD algorithm. the optimal error $\|A - A_k\|_F^2/\|A\|_F^2$ where A_k is the optimal k-rank approximation to matrix A. The best approximation is given by singular value decomposition, which is too time consuming for very large m and n.
Given the elements to be sampled, the matrix C can then be constructed in one additional pass; this requires additional space and time that is $O(mc)$. Given $C\epsilon\mathbb{R}^{m\times c}$, computing $C^T C$ requires $O(mc)$additional space and $O(mc^2)$ additional time, and computing the SVD of $C^T C$ requires $O(c^3)$ additional time. Then computing H_k requires k matrix-vector multiplications for a total of $O(mck)$additional space and time. Thus, overall $O(cm+c^2)$ additional space and $O(c^2 m + c^3)$ additional time are required by the $L.T.SVD$ algorithm.

Constant Time SVD Algorithm (C.T.SVD)

Input: $A\epsilon\mathbb{R}^{m\times n}, c, w, k \in \mathbb{Z}^+$ *such that* $1 \leq w \leq m, 1 \leq c \leq n,$ *and* $1 \leq k \leq \min(w, c)$, $\{P_i\}_{i=1}^n$ *such that* $P_i \geq 0$ *and* $\sum_{i=1}^n P_i = 1$.
Output: $\sigma_t(W), t = 1, \cdots l$ *and* $\widetilde{H}_l\epsilon\mathbb{R}^{m\times l}$.

1. For t=1 to c,
   (a) *Pick* $i_t\epsilon 1, \cdots, n$ *with* $\Pr[i_t = \alpha] = P_\alpha, \alpha = 1, \cdots, n,$ *and save* $\{(i_t, P_{jt}) : t = 1, \cdots, c\}$.
   (b) *Set* $A^{(i_t)}/\sqrt{cP_{i_t}}$. (*note that* C *is not explicitly constructed in RAM.*)
2. *Choose* $\{q_j\}_{j=1}^m$ *such that* $q_j = |C_{(j)}|^2/\|C\|_F^2$.
3. For t=1 to w,
   (a) *Pick* $j_t \in 1, \cdots, m$ *with* $\Pr[j_t = \alpha] = q_\alpha, \alpha = 1, \cdots, m$.
   (b) *Set* $W_{(t)} = C_{(j_t)}/\sqrt{wq_{j_t}}$.
4. *Compute* $W^T W$ *and its SVD. Say* $W^T W = \sum_{t=1}^c \sigma_t^2(W) z^t {z^t}^T$.
5. *If a* $\|.\|_F$ *bound is desired, set* $\gamma = \varepsilon/100k$.
6. *Let* $l = \min\{k, \max\{t : \sigma_t^2(W) \geq \gamma \|W\|_F^2\}\}$.
7. *Return singular values* $\{\sigma_t(W)\}_{t=1}^l$ *and their corresponding singular vectors* $\{z^t\}_{t=1}^l$.

Constant time SVD approximation (C.T.SVD)

Given a matrix $A\epsilon\mathbb{R}^{m\times n}$ we now wish to approximate its top k singular values and the corresponding singular vectors the $C.T.SVD$ algorithm is to pick c columns
of the matrix A, rescale each by an appropriate factor to form a matrix $C \epsilon\mathbb{R}^{m\times c}$, and then compute approximations to the singular values and left singular vectors of the matrix C, which will then be approximations to the singular values and left singular vectors of A. In the $L.T.SVD$ algorithm, the left singular vectors of the matrix C are computed exactly. With the $C.T.SVD$ algorithm, sampling is performed again, drawing rows of C to construct a matrix $W\epsilon R^{w\times c}$. The SVD of $W^T W$ is then computed; let $W^T W = Z\sum_{W^T W} Z^T = Z\sum_W^2 Z^T$. The singular values and corresponding singular vectors so obtained are with high probability approximations to the singular values and singular vectors of $C^T C$ and thus to the singular values and right singular vectors of C. Note that this is simply using the $L.T.SVD$ algorithm to approximate the right singular vectors of C by randomly sampling rows of C.
In that case, $H_k^T H_k = I_k$, $H_k^T H_k$ was an orthonormal projection, and $H_k^T H_k A$ was our rank at most k approximation to A. In the constant time model, we do not have access to H_k but instead to$\widetilde{H}_l$, where the columns of $\widetilde{H}_l$, i.e., $\tilde{h}^t = CZ^T/\sigma_t W$ t = 1, . . . , ℓ, do not form an orthonormal set. However, if C and W are constructed by sampling with optimal probabilities, then with high probability the columns of $\widetilde{H}_l$are approximately orthonormal, $\widetilde{H}_l^T \widetilde{H}_l \approx I_l$, and $\widetilde{H}_l \widetilde{H}_l^T = \sum_{t=1}^l \tilde{h}^t {\tilde{h}^t}^T$s approximately an orthonormal projection. Applying this to A, we will get our low-rank approximation.

Theorem5. Suppose $A\epsilon\mathbb{R}^{m\times n}$; let a description of $\widetilde{H}_l$ be constructed from the $C.T.SVD$ algorithm by sampling c columns of A with probabilities $\{P_i\}_{i=1}^{n}$ and w rows of C with probabilities $\{q_j\}_{j=1}^{m}$ where $p_i = |A^{(i)}|^2/\|A\|_F^2$ and $q_j = |C_{(j)}|^2/\|C\|_F^2$.

$let\ \eta = 1 + \sqrt{8\log(2/\delta)}$ and $\varepsilon > 0$. If a Frobenius norm bound is desired, and hence the $C.T.\ SVD$ algorithm is run with γ = ε /100k, then by choosing $c \geq k^2\eta^2/\varepsilon^4$ columns of A and $w \geq k^2\eta^2/\varepsilon^4$ rows of C we have that with probability at least 1 – δ [6],

$$\|A - \widetilde{H}_l^T\widetilde{H}_l A\|_F^2 \leq \|A - A_k\|_F^2 + \varepsilon\|A\|_F^2.$$

Example1: Let $A\epsilon\mathbb{R}^{m\times n}$ و $B\epsilon\mathbb{R}^{n\times p}$. We want to evaluate an approximation of AB by Monte Carlo method where C and R have the same structure as explained in $A.M.M$ method.

L / m,n,p	**10**	**50**	**100**	**150**	**200**
$m,n = 500, p = 300$	0.0402	0.0074	0.0033	0.0027	0.0021
$m,n = 1000, p = 900$	0.0394	0.0074	0.0038	0.0027	0.0019
$m,n = 1500, p = 1000$	0.0382	0.0076	0.0037	0.0024	0.0020
$m,n = 2000, p = 1000$	0.0373	0.0075	0.0038	0.0025	0.0020

Table 1: Relative error of results in example 1

It is well known that the relative error is $\frac{\|AB-CR\|_F^2}{\|A\|_F^2}$.

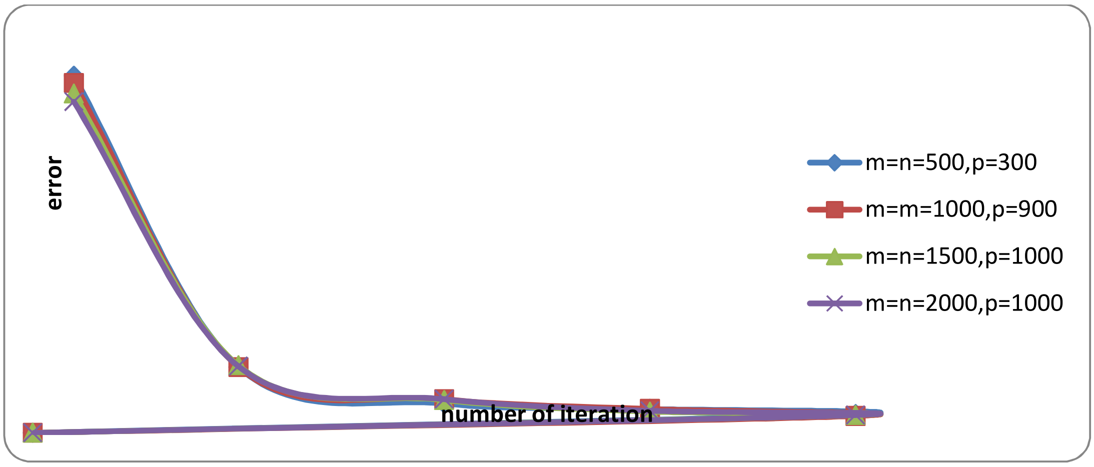

Fig 1: Error of results based on number of iterations based on $A.M.M$ Algo

In table 2, we compare the computational time needed for approximating AB by CR multiplication, using $c = 2$, $L = 200$ and the corresponding time needed for obtaining AB multiplication where we show them by $time1$ and $time2$, respectively.

m,n,p	$time1$	$time2$	$Relative\ Error$
$m,n = 500, p = 300$	1.6869	3.909653	0.0021
$m,n = 1000, p = 900$	4.812916	50.051374	0.0019

$m,n = 1500, p = 1000$	7.815693	125.800965	0.0020
$m,n = 2000, p = 1000$	10.503226	249.492549	0.0020

Table 2: Computational time for CR and AB

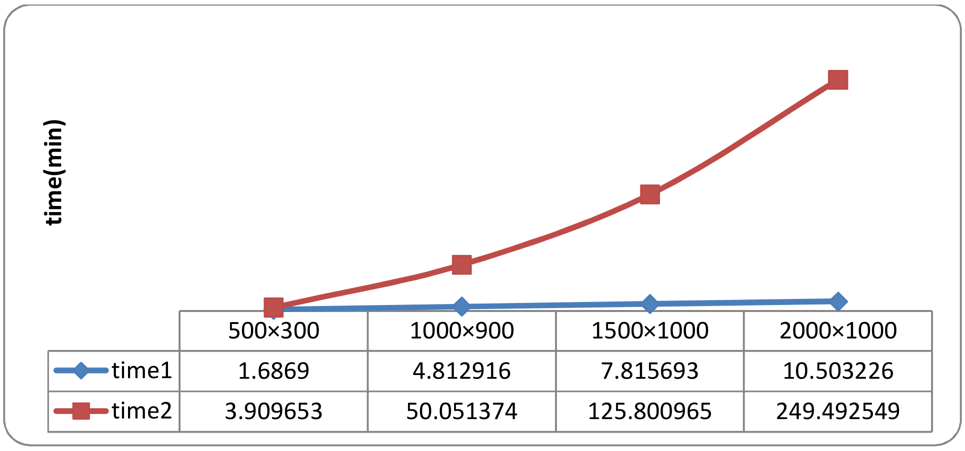

Fig 2: Comparison of computational time for obtaining CR and AB

We note that the computational time for obtaining CR based on algorithm $A.M.M$ is significantly less than the corresponding time for evaluating AB.

Example 2: Suppose that $A\epsilon\mathbb{R}^{m\times n}$. Using a fixed $k = 1$ and increasing c (the number of columns that we select them randomly), we want to evaluate the relative error, the computational time of both $L.T.SVD$ and optimal SVD algorithms and compare them together. We note that for $k = 1$, $m = n = 1500$ the relative error sand speed of optimal are 0.2497 and 203**.** 215822 seconds**,** respectively.

c	***Relative Error***	***Time***
200	0.2509	3.720119
400	0.2502	8.071144
600	0.2499	31.635689
800	0.2498	55.630737
1000	0.2498	91.607197
1200	0.2497	144.304089

Table 3: Relative error and computational time of optimal *SVD* algorithm

Relative error of optimal k rank approximation is $\frac{\|A-A_k\|_F^2}{\|A\|_F^2}$

Relative error of the L.T. SVD algorithm is $\frac{\|A-H_kH_k^TA\|_F^2}{\|A\|_F^2}$

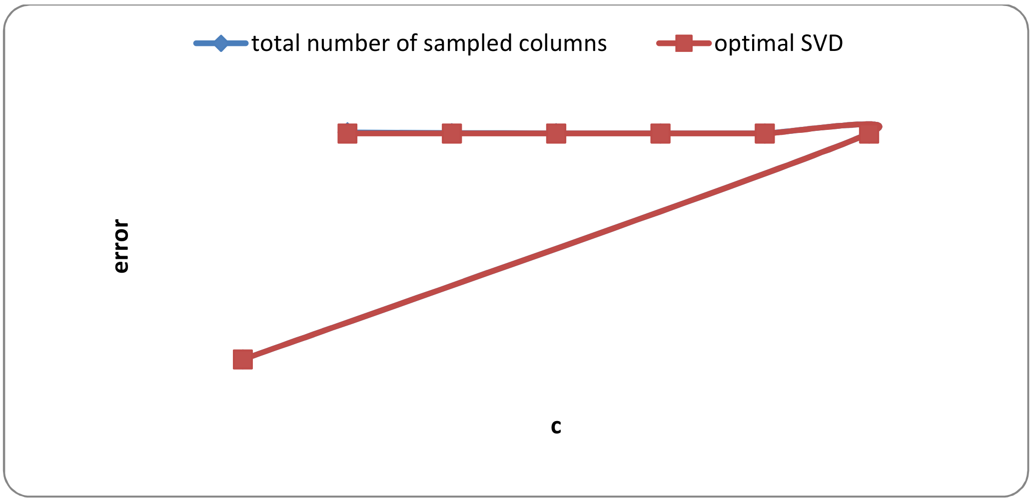

Fig 3: Error of results for optimal *SVD* algorithm based on ***c***

As we see with increasing ***c***, the relative error in $L.T.SVD$ algorithm is reduced. Now, if we set $c = 2$, $k = 1$, with increasing the number of iterations L in Monte Carlo method, we compare the relative error in $L.T.SVD$ and SVD algorithms in table 4.

L	Relative Error	Time
100	0.2509	3.041115
200	0.2504	4.121195
300	0.2501	5.228417
400	0.2500	6.309273
500	0.2499	7.386124
600	0.2499	8.472105
700	0.2499	9.565714
800	0.2498	10.678412

Table 4: Error of results for Monte Carlo algorithm based on *L*

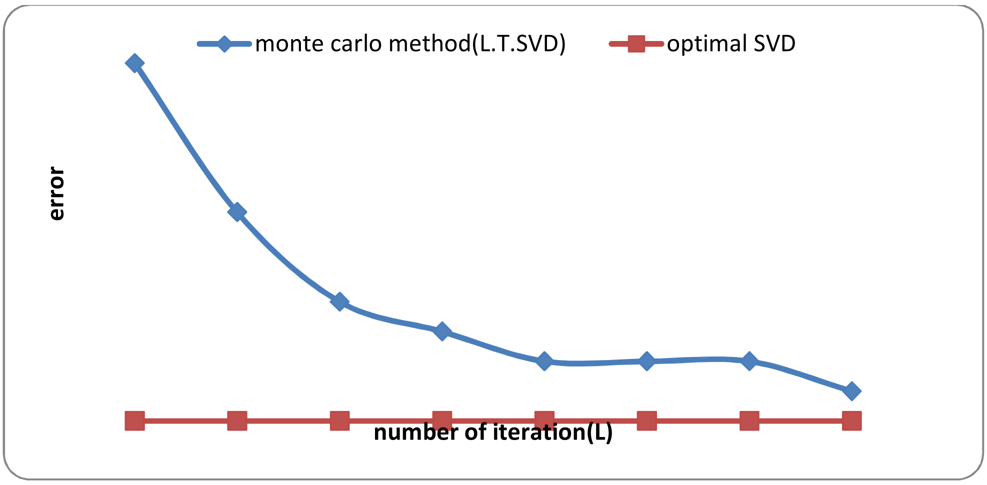

Fig 5: Comparison of errors in optimal SVD and Monte Carlo algorithms

From Fig 5, we conclude that with increasing the relative error will close to optimal SVD error.

Now, we compare the computational time using Monte Carlo ($L.T.SVD$) and optimal SVD algorithms.

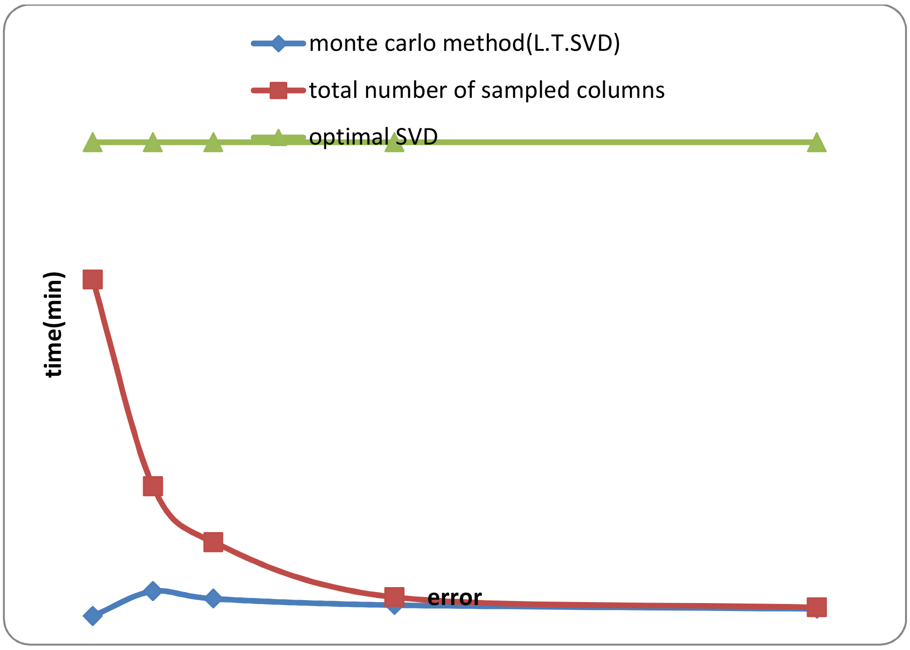

Fig 6: comparison of need time for approximation using MC and SVD methods.

References

[1] D. Achlioptas and F. McSherry, *Fast computation of low rank matrix approximations*, J. ACM, to appear.
[2] R. Agrawal, J.Gerhrke, D.Gunopulos, and P. Raghavan, Automaticsubspace clustering of high dimensional data for data mining applications, *Proc. ACM SIGMOD Conf. onManagement of Data*, 1998, 94-105.
[3] D. Barbara, C. Faloutsos, J. Hellerstein, Y. Ioannidis, H. V. Jagadish, T. Johnson, R. Ng, V. Poosala, K. Ross, and K. C. Sevcik, *The New Jersey data reduction report*, Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 1997.
[4] P. Drineas, R. Kannan, and M. W. Mahoney, *Fast Monte Carlo algorithms for matrices* II: *Computing a low-rank approximation to a matrix*, SIAM J. Comput., 36 (2006), pp.158–183.
[5] P. Drineas, R. Kannan, and M. W. Mahoney, Fast Monte Carlo algorithms for matrices I:Approximating matrix multiplication, SIAM J. Comput., 36 (2006), pp. 132–157.
[6] P. Drineas, R. Kannan, and M. W. Mahoney, *Fast Monte Carlo algorithms for matrices* II: *Computing a low-rank approximation to a matrix*, SIAM J. Comput., 36 (2006), pp. 158–183.
[7] P. Drineas, R. Kannan, and M. W. Mahoney, Fast Monte Carlo algorithms for matrices III: Computing a compressed approximate matrix decomposition, SIAM J. Comput., 36 (2006), pp. 184–206.
[8] P. Drineas and R. Kannan, *Pass efficient algorithms for approximating large matrices*, in Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms, 2003, pp. 223–232.
[9] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan, *An approximate L*1*-difference algorithm for massive data sets*, in Proceedings of the 40th Annual IEEE Symposium on the Foundations of Computer Science, 1999, pp. 501–511.
[10]G. H. Golub and C. F. Van Loan, Matrix Computations, Johns Hopkins University Press, London,1989.
[11] M. R. Henzinger, P. Raghavan, and S. Rajagopalan, *Computing on Data Streams*, Tech Report 1998-011, Digital Systems Research Center, Palo Alto, CA, 1998.

[12] J. I. Munro and M. S. Paterson, *Selection and sorting with limited storage*, in Proceedings of the 19th Annual IEEE Symposium on Foundations of Computer Science, 1978, pp. 253–258.