On Speeding up Jacobi Iterations for SVDs

Soumitra Pal   Sudipta Pathak   Sanguthevar Rajasekaran

Computer Science and Engineering, University of Connecticut

371 Fairfield Road, Storrs, CT 06269, USA

{mitra,sudipta.pathak}@uconn.edu,rajasek@engr.uconn.edu

Abstract—We live in an era of big data and the analysis of these data is becoming a bottleneck in many domains including biology and the internet. To make these analyses feasible in practice, we need efficient data reduction algorithms. The Singular Value Decomposition (SVD) is a data reduction technique that has been used in many different applications. For example, SVDs have been extensively used in text analysis. Several sequential algorithms have been developed for the computation of SVDs. The best known sequential algorithms take cubic time which may not be acceptable in practice. As a result, many parallel algorithms have been proposed in the literature. There are two kinds of algorithms for SVD, namely, QR decomposition and Jacobi iterations. Researchers have found out that even though QR is sequentially faster than Jacobi iterations, QR is difficult to parallelize. As a result, most of the parallel algorithms in the literature are based on Jacobi iterations. JRS is an algorithm that has been shown to be very effective in parallel. JRS is a relaxation of the classical Jacobi algorithm. In this paper we propose a novel variant of the classical Jacobi algorithm that is more efficient than the JRS algorithm. Our experimental results confirm this assertion. We also provide a convergence proof for our new algorithm. We show how to efficiently implement our algorithm on such parallel models as the PRAM and the mesh.

Keywords-SVD; Jacobi iterations; JRS; parallel algorithms

I. INTRODUCTION

Singular Value Decomposition (SVD) is considered a fundamental computation in linear algebra and it has application in almost every possible fields one can think of. Its widely used in areas such as statistical analysis techniques, independent component analysis, information retrieval, data mining, singnal processing etc. Applications involving analysis or BigData using SVD requires efficient and fast algorithms to process matrices with thousands of rows and columns in real time. Hence, there is a strong need for efficient SVD algorithms using sequential and parallel computers. In this paper we introduce novel sequential and parallel algorithms for SVD computation of $m \times n$ dense matrix A with $(m \geq n)$.

SVD takes as input a matrix $A \in \mathbb{R}^{m \times n}$ where $(m \geq n)$ and outputs three matrices U, S, V such that $A = USV^T$, where $U \in \mathbb{R}^{m \times m}$ is an orthogonal matrix (i.e. $U^T U = I$) and $V \in n \times n$ is an orthogonal matrix (i.e. $V^T V = I$) and $S \in \mathbb{R}^{m \times n}$ is a diagonal matrix. If $S = diag(\sigma_1, \sigma_2, ...\sigma_n)$, then these diagonal elements are called singular values of A. The columns vectors of U and V

are referred to as left and right singular vectors respectively.

A. Two-sided Jacobi Algorithm

Among all algorithms for finding SVD Jacobi iteration based algorithms are most widely used in parallel settings since they are easier to parallelize than others. There are two types of Jacobi algorithms for SVD, namely Two-sided Jacobi SVD algorithm and One-sided Jacobi SVD algorithm. Two-sided Jacobi SVD algorithm is applicable to symmetric matrices only. It applies a series of *Jacobi Rotations* on input matrix A to diagonalize it.

$$A_{k+1} = U_k A_k U_k^T, k = 1, 2, ...$$

where $U_k = U_k(i, j, \phi_{ij}^k)$ represents a rotation of (i, j)-plane.

$$u_{ii}^k = u_{jj}^k = c^k = \cos(\phi_{ij}^k) \tag{1}$$

$$u_{ij}^k = -u_{ji}^k = s^k = \sin(\phi_{ij}^k) \tag{2}$$

ϕ^k is chosen such that

$$b_{ij}^{k+1} = b_{ji}^{k+1} = 0, \tag{3}$$

$$\text{or} \tan(2\phi_{ij}^k) = \frac{2b_{ij}^k}{b_{ii}^k - b_{jj}^k} \tag{4}$$

$$\text{and where} |\phi_{ij}^k| \leq \frac{1}{4}\pi \tag{5}$$

c^k is computed by

$$c^k = \frac{1}{\sqrt{1 + t_k^2}} \text{ and} s^k = c^k t^k \tag{6}$$

where$t^k$is the smaller root (in magnitude) of the quadratic equation give

$$\tag{7}$$

$$t^{k^2} + 2\alpha^k t^k - 1 = 0, \alpha^k = \cot(2\phi_{ij}^k) \tag{8}$$

$$\text{hence,} t^k = \frac{sign(\alpha^k)}{|\alpha^k| + \sqrt{1 + \alpha^{k^2}}} \tag{9}$$

B_{k+1} is identical to B_k other than the rows and columns i and j and the modified elements are given by

$$b_{ii}^{k+1} = b_{ii}^k + t^k b_{ij}^k \tag{10}$$

$$b_{jj}^{k+1} = b_j j^k - t^k b_{ij}^k \tag{11}$$

$$b_{ix}^{k+1} = c^k b_{ix}^k + s_{jr}^k \tag{12}$$

$$b_{jx}^{k+1} = -s_k b_{ix}^k + c_k b_{jx}^k \tag{13}$$

$$x \neq i, j \tag{14}$$

The above algorithm ensures that with every rotation A_k approaches the diagonal matrix $S = \text{diag}(\sigma_1, \sigma_2, ...\sigma_n)$ and $(U_k......U_2 U_1)^T$ approaches a matrix whose $j-$th column is the eigenvector corresponding to σ_j. [1] shows that each sweepcauses the off-diagonal norm to decrase and evetually the algorithm converges. According to the original Jacobi scheme one should search for the largest off-diagonal element to eliminate at every rotation. However, this scheme is extremely time consuming in practice. Hence, a common scheme used is to choose elements (i, j) to be annihilated in cyclic order $(1, 2), (1, 3), (1, 4), ...(1, n), (2, 3), ...(2, n), ..., (n - 1, n)$. This is known as cyclic jacobi algorithm. citeforsythe1960cyclic, schonhage1961konvergenz, wilkinson1962note show that cyclic Jacobi algorithm has a quadratic convergence rate and argues that convergence ususally occurs within 6 to 10 sweeps or $3n^2$ to $5n^2$ Jacobi rotations.

B. One-sided Jacobi Algorithm

A more general and more widely used algorithm is One-Sided Jacobi SVD algorithm that deals with any matrix. One-Side Jacobi SVD algorithm finds an orthogonal matrix $V \in \mathbb{R}^{n \times n}$, such that $B = (b_1, b_2, ...,b_n) = AV$ where $B \in \mathbb{R}^{m \times n}$ is an orthogonal matrix. The above statement implies that:

$$b_i^T b_j = \sigma_i^2 \delta_{ij}$$

where

$$\delta_{ij} = \begin{cases} 0, & \text{for } i \neq j \\ 1, & \text{for } i = j \end{cases}$$

The quantities σ_i are called *Singular Values* of matrix A. B can be written as $B = US$ where $U = (u_1, u_2, ..., u_k)$such that $u_j = \frac{b_j}{\sigma_j}$ for $\sigma_j \neq 0$ and $U^T U = I \in \mathbb{R}^{n \times n}$. Hence we can now write $B = AV$ as $US = AV$ or using properties of orthogonality $A = USV^T$. If some of the σ_js are zeros singular values, the corresponding columns of U are set to null vectors. The first k singular values are always chosen to be non-zero causing the matrix $U^T U$ to be unit matrix of order k augmented to order n with zeros. This is written as follows :

$$S = \begin{pmatrix} \mathbf{1}_k & \\ & \mathbf{0}_{n-k} \end{pmatrix} \tag{15}$$

The matrix V used for orthogonalization of A is a product of matrices indexed by $V^{(k)}$.

$$V = \prod_{k=1}^{z} V^{(k)}$$

V is constructed in such a way that

$$(a_i, a_j) \begin{bmatrix} c & -s \\ s & c \end{bmatrix} = (a_i^{k+1}, a_j^{k+1}) \tag{16}$$

, $i < j$ and $||a_i^{k+1}||_2 > ||a_j^{k+1}||_2$, here a_i is the $i-$th column of A. This is achieved by choosing

$c = [\frac{\beta + \gamma}{2\gamma}]^{\frac{1}{2}}$ and $s = [\frac{\alpha}{2\gamma c}]$, $s = [\frac{\gamma - \beta}{2\gamma}]^{\frac{1}{2}}$ and $c = [\frac{\alpha}{2\gamma s}]$

where $\alpha = 2a_i^T a_j$, $\beta = ||a_j||_2^2$ and $\gamma = (\alpha^2 + \beta^2)^{\frac{1}{2}}$

The matrices given by $V^{(k)}$ are called rotation matrices and z is not dependent of dimension of A.

Describe one sided Jacobi with math notations from Dr Rajs paper:

C. Block Householder Jacobi

For $A \in \mathbb{R}^{m \times n}$ where $m >> n$, a common practice is to reduce the problem complexity by applying initial orthogonal factorization on A followed by one-sided Jacobi method. Given A we apply the following orthogonal facotirzation given by $A = QR$, where $Q \in \mathbb{R}^{m \times n}$ is an orthogonal matrix and $R \in \mathbb{R}^{n \times n}$ is an upper triangular matrix. Next, One-sided Jacobi SVD algorithm is applied on R. The Block Householder- Jacobi SVD algorithm is presented below:
Step 1: Factorize A by $A = QR$ where $Q \in \mathbb{R}^{m \times n}$ with orthonormal columns and $R \in \mathbb{R}^{n \times n}$ is upper triangular.
Step 2: Determine SVD of the upper triangular matrix using One-sided Jacobi algorithm.

$$R = \hat{U} \begin{bmatrix} \hat{S} \\ 0 \end{bmatrix} V^T \tag{17}$$

where $\hat{U} \in \mathbb{R}n \times p$ and $V \in \mathbb{R}n \times p$ orthogonal matrices and $\hat{S} = \text{diag}(\sigma_i)$ is a diagonal matrix containing p non-zero singular values of A.
Step 3: Compute $U = Q\hat{U}$. Left singular vectors of A are given by u_i where u_i are columns of U.

D. Two-sided JRS Algorithm

E. One-sided JRS Algorithm

F. Group JRS Algorithm

[2]
This paper has done same parallel implementation: [3].
This paper seems to do some kind of sorting for [4].

G. Contributions

Subsection text here.

II. QUICK PIVOTING

Wherever Times is specified, Times Roman or Times New Roman may be used. If neither is available on your system, please use the font closest in appearance to Times. Avoid using bit-mapped fonts if possible. True-Type 1 or Open Type fonts are preferred. Please embed symbol fonts, as well, for math, etc.

III. Parallel Implementation

Wherever Times is specified, Times Roman or Times New Roman may be used. If neither is available on your system, please use the font closest in appearance to Times. Avoid using bit-mapped fonts if possible. True-Type 1 or Open Type fonts are preferred. Please embed symbol fonts, as well, for math, etc.

IV. Experimental Results

V. Conclusion

The conclusion goes here. this is more of the conclusion

Acknowledgment

The authors would like to thank... more thanks here

References

[1] F. T. Luk and H. Park, "A proof of convergence for two parallel jacobi svd algorithms," *Computers, IEEE Transactions on*, vol. 38, no. 6, pp. 806–811, 1989.

[2] S. Rajasekaran and M. Song, *A Novel Scheme for the Parallel Computation of SVDs*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 129–137.

[3] M. I. Soliman, "Memory Hierarchy Exploration for Accelerating the Parallel Computation of SVDs," *Neural, Parallel Sci. Comput.*, vol. 16, no. 4, pp. 543–561, Dec. 2008.

[4] B. B. Zhou and R. P. Brent, "On parallel implementation of the one-sided jacobi algorithm for singular value decompositions," in *Parallel and Distributed Processing, 1995. Proceedings. Euromicro Workshop on*. IEEE, 1995, pp. 401–408.