

Randomized Algorithms for Singular Value Decomposition: Implementation and Application Perspective

Darko Janeković, Dario Bojanjac

University of Zagreb, Faculty of Electrical Engineering and Computing, Unska 3, 10000 Zagreb, Croatia.
darko.janekovic@gmail.com

Abstract—Singular value decomposition (SVD) is a key step in many algorithms in statistics, machine learning and numerical linear algebra. While classical singular value decomposition has been made efficient in terms of computational complexity, classical algorithms are not able to fully utilise modern computing environments. The goal of this work is to survey various implementations and applications of randomized algorithms for SVD. Algorithms are compared in terms of accuracy and time of execution. On example of robust principal component analysis (RPCA), it is shown that using randomized algorithms can yield a significant speedup for image processing and similar applications.

Keywords—SVD; PCA; Randomized algorithms; Low-rank approximation

I. INTRODUCTION

The singular value decomposition (SVD) is a fundamental matrix factorization used in many different algorithms. Even though, best known applications are connected to machine learning [1] [2], the SVD was applied to various disciplines [3] [4]. With large volumes of data available, applications of SVD became heavily data oriented. Matrices in question are often tall and skinny proving to be challenging for classical decompositions. Just as well, lack of out of core methods for matrix decompositions means algorithms are limited to central processing units (CPU). On the other hand, graphical processing units (GPU) represent novel compute architectures with limited memory size, high latency but immense arithmetic capacity [5].

Recently, randomized algorithms were proposed in order to bridge a gap between modern computer architectures and classical algorithms in numerical linear algebra. Randomized algorithms represent a class of algorithms based on random matrix theory where basic idea is to form an approximate subspace that captures action of a linear operator. Even though randomized algorithms are in essence stochastic, one can show strong upper bounds on variance of error [6].

From an applications perspective, randomized algorithms prove to be a stable and fast tool for performing scalable low-rank approximations or accurate and fast truncated SVD decomposition. This paper will give an overview of implementation techniques for matrices that fit in the main memory as well as out-of-core methods where matrices do not fully fit in memory. Former will be accompanied with analysis of

memory consumption and our implementation that follows one given in [6].

This paper is organized as follows. In Section II randomized algorithms are introduced. Section III describes implementation. Just as well, tests on synthetic matrices are presented. In Section IV the robust principal component analysis is defined and results on data matrices coming from a video stream are discussed.

II. RANDOMIZED ALGORITHMS

Randomized algorithms have been developed by Halko, Martisson, and Tropp by relying on randomized linear algebra and random matrix theory [6]. They are extremely stable and probability for large oscillations in solution are provably minuscule. Basic idea of randomized algorithms is to approximate the range of linear operator and compute factorization in the approximate lower dimensional basis. If approximate range fully captures action of linear operator, algorithm is exact. Approach is highly connected to low-rank approximation problem where goal is to approximate input matrix \mathbf{A} with low-rank matrix \mathbf{B} [7].

Let $\mathbf{A} \in \mathbb{C}^{m \times n}$ and $k \leq \min(m, n)$, find matrix \mathbf{B} such that

$$\min_{\text{rank}(\mathbf{B}) \leq k} \|\mathbf{A} - \mathbf{B}\|. \quad (1)$$

The SVD furnishes an optimal answer for the low-rank approximation problem given by (1). In order to solve low-rank approximation problem with SVD, one should compute full SVD and then truncate the result to the first k terms [8].

Randomized algorithms provide a framework for direct truncation to the first k components without computing the full factorization. The same can be accomplished with Krylov methods which are better suited for sparse matrices since they interact with linear operator \mathbf{A} through matrix-vector multiplication [9]. These operations are often bandwidth limited and not able to fully saturate the arithmetic unit. Randomized methods instead interact with linear operator \mathbf{A} through series of matrix-matrix multiplications that are able to saturate the arithmetic unit and achieve high performance. Moreover, these operations are easily parallelizable.

Randomized algorithms solve low-rank approximation problem by finding an approximate subspace \mathbf{Q} such that

$$\min_{\mathbf{Q} \in \mathbb{C}^{m \times k}} \|\mathbf{A} - \mathbf{Q}\mathbf{Q}^* \mathbf{A}\|. \quad (2)$$

Range of \mathbf{A} is approximated by randomized projection onto the lower dimensional subspace \mathbf{Y} . Subspace \mathbf{Y} is further orthonormalized to produce a basis \mathbf{Q} . If \mathbf{A} is a rank- k linear operator, subspace \mathbf{Y} of dimension k will fully capture the range of \mathbf{A} . On the other hand, if rank of linear operator is greater than k , lower singular values will shift the basis vectors and span will not capture the best rank- k approximation. However, projecting to a slightly larger, $k + p$ dimensional subspace, decreases error super-exponentially with respect to oversampling parameter p . Complete upper bound is given by (3).

$$\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|^2 \leq \|\Sigma_2\|^2 + \|\Sigma_2\Omega_2\Omega_1^\dagger\|^2 \quad (3)$$

Former part of the error bound (3) is defined by Eckart-Young theorem [8], and the latter depends on oversampling parameter p . Matrix Ω_1 is a $k \times k+p$ Gaussian random matrix. Fig. 1 demonstrates norm of pseudoinverse with respect to the change of parameter p . It can be seen that expectation and variance are greatly reduced [6]. This is the fact that makes randomized algorithms usable in practice.

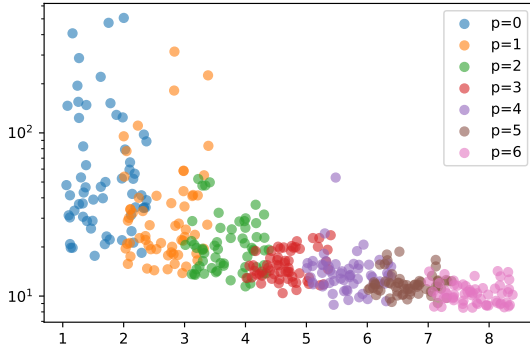


Figure 1. Norm of pseudoinverse $k \times k + p$ matrix with respect to the oversampling parameter p .

Finally, \mathbf{A} is restricted to $k + p$ dimensional subspace with basis \mathbf{Q}^T and the factorization is computed on a matrix with dimensions $k + p \times m$. Complete step is shown in Algorithm 1. Complete algorithm computes truncated SVD in $\mathcal{O}(mnk)$.

Algorithm 1 RSVD-Factorization

- 1: $\mathbf{B} \leftarrow \mathbf{Q}^*\mathbf{A}$
 - 2: $\hat{\mathbf{U}}\Sigma\mathbf{V}^* \leftarrow \text{SVD}(\mathbf{B})$
 - 3: $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$
-

A. Power method

The error in Algorithm 2 depends on how quickly the singular values decay. If singular spectrum is slowly decreasing, the shift of basis vectors will be more pronounced. Power method is able to mitigate these effects by computing

$$\mathbf{A}^{(q)} = (\mathbf{A}\mathbf{A}^*)^q\mathbf{A} = \mathbf{U}\Sigma^{2q+1}\mathbf{V}^* \quad (4)$$

Power method will attenuate lower singular values, making the singular spectrum suitable for randomized algorithms. However, high number of power method iterations will introduce subtle numerical errors due to rounding of lower singular values. Fortunately, this issue can be remedied by orthonormalizing the subspace after each iteration of power method and it is not encountered for $q \leq 4$ in double precision. In case lower precision is needed to reduce transfer costs or running time, mixed precision approach could be beneficial.

III. IMPLEMENTATION

General implementation given in this paper assumes that matrices fit in the main memory of either GPU or CPU. Finding a range of matrix \mathbf{A} is given in Algorithm 2.

Algorithm 2 RSVD-Range finder

- 1: $\mathbf{G} \sim \mathcal{N}(0, 1)^{n \times k+p}$
 - 2: $\mathbf{Y} \leftarrow \mathbf{G}\mathbf{A}$
 - 3: **for** $i \leftarrow 1, q$ **do**
 - 4: $\mathbf{Y} \leftarrow \mathbf{A}\mathbf{G}$
 - 5: $\mathbf{G} \leftarrow \mathbf{A}^*\mathbf{Y}$
 - 6: **end for**
 - 7: $\mathbf{Y} \leftarrow \mathbf{A}\mathbf{G}$
 - 8: $\mathbf{Q}, _ = \text{QR}(\mathbf{Y}, \text{inplace})$
-

Computational complexity of randomized range finder is $\mathcal{O}(mnk)$. More importantly, memory footprint of the algorithm are matrices \mathbf{Y} and \mathbf{Q} , where only \mathbf{Y} is a temporary matrix that is not needed in the factorization step. Second step is shown in Algorithm 1. Memory footprint of this algorithm are resulting matrices from the SVD. Matrix \mathbf{G} in Algorithm 2 can be reclaimed as matrix \mathbf{B} , and memory for matrix \mathbf{Y} can be used for storing resulting \mathbf{U} .

A. Execution time and accuracy

Randomized algorithms for SVD with and without regularization of power method are compared with Lanczos SVD which are mathematically similar to the randomized algorithms. Accuracy and time of execution are given on Fig. 2, 3, 4. Execution time rises with rank k and accuracy decreases. Input matrices in all tests were random Gaussian matrices with dimensions 3072×512 . Experiments are ran on Intel Xeon Gold 5220S CPU. For every experiment involving randomized algorithms, $p = 10$ will be used.

Several effects can be noticed from the measurements:

- 1) regularization is shown to be costly without noticeable accuracy benefits,
- 2) RSVD without regularization is as fast as full SVD,
- 3) Lanczos methods are slow but very accurate.

First point is easily explained with the fact that regularization computes QR decomposition $2q + 1$ times instead of one time. Since the factorization is done in double precision, loss of precision does not justify increase in time of execution. The fact that RSVD without regularization is as fast as full SVD is just a proof that majority of time is spent in algorithm 1. For

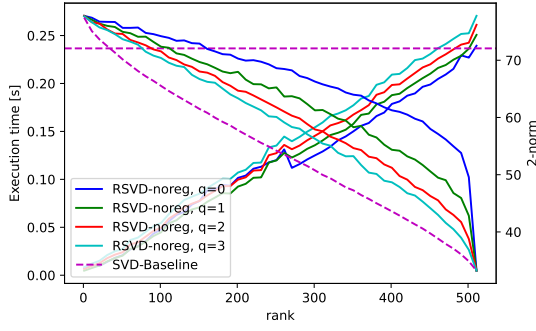


Figure 2. Randomized SVD without regularization of power method.

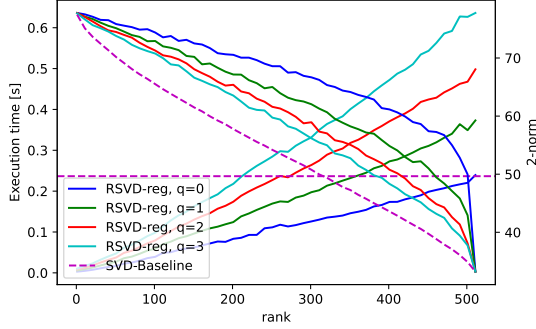


Figure 3. Randomized SVD with regularization of power method.

matrices that are tall and skinny, such as 10240×512 , majority of time is spent in QR decomposition since Householder QR does not scale well. This could be fixed by using Tall Skinny QR (TSQR) algorithm instead of Householder QR for matrices where TSQR is better suited [10].

TABLE I. TIME DISTRIBUTION FOR RSVD WITHOUT REGULARIZATION AND MATRICES WITH DIMENSIONS 10240×512 .

	q=0	q=1	q=2
GEMM	9.13 %	18.95 %	29.23 %
QR	65.58 %	49.54 %	49.94 %
SVD	25.14 %	31.51 %	20.83 %

Finally, as it was noted, Lanczos methods belong to the class of Krylov methods which interact with input matrix via series of matrix-vector operations. These operations are comparatively slow but on the other hand, these methods are much more precise than randomized algorithms.

B. Out of core implementation

Out-of-core is a divide and conquer technique for processing matrices that are bigger than the main memory. This approach was proposed in the context of GPU accelerated implementation [11]. Main advantage of this method is that at any given time, main memory stores just the fraction of original input matrix. Furthermore, as can be seen on (5), randomized projection holds no data dependencies and each block can compute resulting \mathbf{Y}_i without communication.

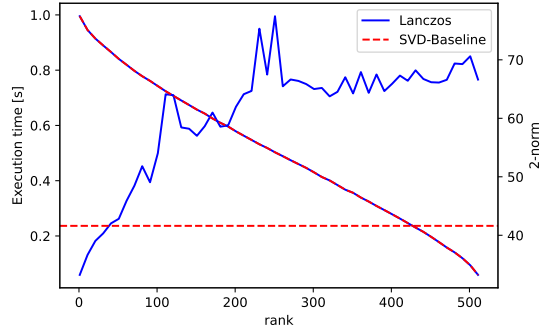


Figure 4. Lanczos method for SVD.

$$\begin{bmatrix} \mathbf{Y}_1 \\ \mathbf{Y}_2 \\ \mathbf{Y}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \mathbf{A}_3 \end{bmatrix} \cdot \mathbf{G} \quad (5)$$

Block decomposition for power method shown with (6) is fairly similar to the scheme (5) for randomized sampling with one caveat. In order to compute $\mathbf{G} \leftarrow \mathbf{A}^* \mathbf{Y}$, extensive communication is needed since i -th processing unit is holding \mathbf{G}_i and sum reduction is needed.

$$\begin{bmatrix} \mathbf{Y}_1 \\ \mathbf{Y}_2 \\ \mathbf{Y}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \mathbf{A}_3 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{A}_1^T & \mathbf{A}_2^T & \mathbf{A}_3^T \end{bmatrix} \begin{bmatrix} \mathbf{Y}_1 \\ \mathbf{Y}_2 \\ \mathbf{Y}_3 \end{bmatrix}. \quad (6)$$

Algorithm is well-suited for execution with multiple nodes or multiple GPUs. However, some care should be taken in order to efficiently compute sum reduction for power method.

IV. NUMERICAL EXPERIMENTS

The most popular application of randomized SVD is the robust principal components analysis (RPCA). Main reason for that fact is that it is possible to predict ranks and accurately truncate computation of SVD. When time of execution is compared to the classical SVD, significant speedup can be observed.

A. RPCA

Robust principal component analysis is a statistical method for decomposing data matrix \mathbf{M} on a sum of low-rank matrix \mathbf{L} and sparse matrix \mathbf{S} [12]. Method finds a solution to the convex optimization problem

$$\begin{aligned} \min_{\mathbf{L}, \mathbf{S}} \quad & \|\mathbf{L}\|_* + \lambda \|\mathbf{S}\|_1 \\ \text{subject to} \quad & \mathbf{L} + \mathbf{S} = \mathbf{M} \end{aligned} \quad (7)$$

where $\|\cdot\|_*$ and $\|\cdot\|_1$ denote the nuclear norm and ℓ_1 norm of a matrix. Problem can be solved with various solvers, in continuation Inexact Augmented Lagrange Multiplier (IALM) is used. IALM is a fast and stable method that first computes optimal \mathbf{L} with an assumption of constant \mathbf{S} and then vice versa [13] [12]. Solving RPCA with IALM is well suited for randomized SVD since it is observed that rank of \mathbf{L} is

monotonically rising with each iteration making it easy to predict rank and accurately truncate SVD.

One of canonical applications for RPCA is the removal of static background in surveillance videos [13]. Static background is a low rank matrix \mathbf{L} and dynamic content in a scene is sparse matrix \mathbf{S} . Data matrix is assembled by vertical stacking of flattened 1280×720 frames. Implementation using randomized SVD showed significant speedup on large data matrices where SVD truncation is able to cut down execution of SVD. Running time for randomized and classical SVD is shown in table II.

TABLE II. RPCA RUNNING TIME WITH RESPECT TO THE SVD ALGORITHM

Matrix size	921600×141	921600×211	921600×281
RSVD	227 s	323 s	431 s
SVD	560 s	876 s	1206 s

Removal of static background is demonstrated on Fig. 5 from VIRAT dataset [14]. On Fig. 6 and 7 low-rank static background and dynamic sparse content is shown.

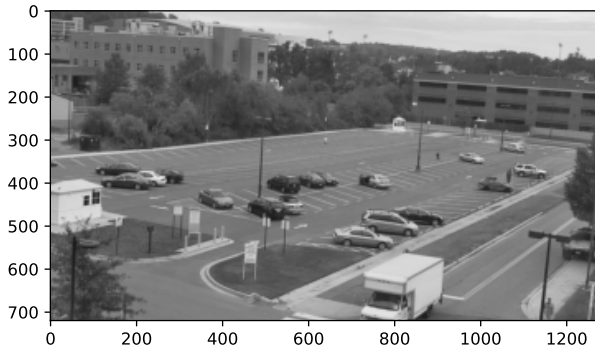


Figure 5. Original frame from VIRAT database.

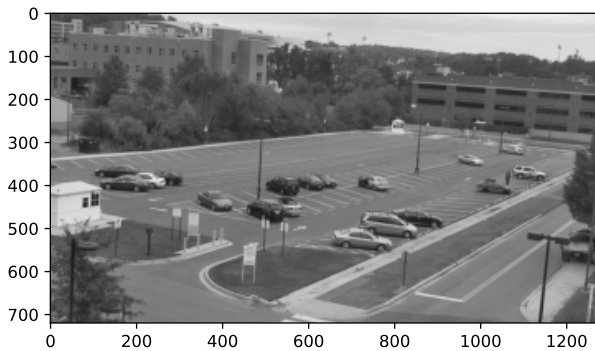


Figure 6. Low-rank matrix \mathbf{L} .

V. FUTURE WORK

Future work includes optimizing the QR factorization with call to TSQR factorization and robust GPU out-of-core implementation. We would also be interested in a fast preprocessing step that could predict optimal parameter q depending on the singular spectrum and matrix dimensions.

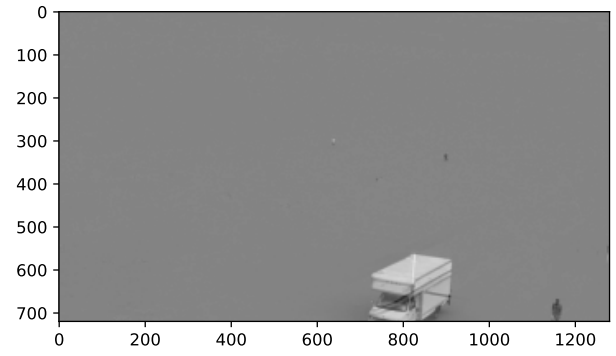


Figure 7. Sparse matrix \mathbf{S} .

ACKNOWLEDGMENT

The authors would like to thank prof. dr. sc. Nenad Antić and Laboratory for Advanced Computing (LNR) for lending computer resources needed to generate results for this paper. This work has been supported in part by Croatian Science Foundation under the project IP-2019-04-1064.

REFERENCES

- [1] T. Bouwmans, S. Javed, H. Zhang, Z. Lin, and R. Otazo, "On the applications of robust pca in image and video processing," *Proceedings of the IEEE*, vol. 106, no. 8, pp. 1427–1457, 2018.
- [2] M. Aharon, M. Elad, and A. Bruckstein, "K-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on signal processing*, vol. 54, no. 11, pp. 4311–4322, 2006.
- [3] Y. Cao, L. Wen, and J. Rong, "A svd accelerated kernel-independent fast multipole method and its application to bem," *WIT Transactions on Modelling and Simulation*, vol. 56, pp. 431–443, 2013.
- [4] S. Chakraborty, S. Chatterjee, N. Dey, A. S. Ashour, and A. E. Hasanien, "Comparative approach between singular value decomposition and randomized singular value decomposition-based watermarking," in *Intelligent techniques in signal processing for multimedia security*, pp. 133–149, Springer, 2017.
- [5] J. L. Hennessy and D. A. Patterson, *Computer architecture: a quantitative approach*. Elsevier, 2011.
- [6] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM review*, vol. 53, no. 2, pp. 217–288, 2011.
- [7] N. Kishore Kumar and J. Schneider, "Literature survey on low rank approximation of matrices," *Linear and Multilinear Algebra*, vol. 65, no. 11, pp. 2212–2244, 2017.
- [8] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936.
- [9] D. S. Watkins, *The matrix eigenvalue problem: GR and Krylov subspace methods*. SIAM, 2007.
- [10] J. Demmel, L. Grigori, M. Hoemmen, and J. Langou, "Communication-optimal parallel and sequential qr and lu factorizations," *SIAM Journal on Scientific Computing*, vol. 34, no. 1, pp. A206–A239, 2012.
- [11] Y. Lu, I. Yamazaki, F. Ino, Y. Matsushita, S. Tomov, and J. Dongarra, "Reducing the amount of out-of-core data access for gpu-accelerated randomized svd," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 19, p. e5754, 2020.
- [12] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?," *Journal of the ACM (JACM)*, vol. 58, no. 3, pp. 1–37, 2011.
- [13] T. Bouwmans and E. H. Zahzah, "Robust pca via principal component pursuit: A review for a comparative evaluation in video surveillance," *Computer Vision and Image Understanding*, vol. 122, pp. 22–34, 2014.
- [14] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C.-C. Chen, J. T. Lee, S. Mukherjee, J. Aggarwal, H. Lee, L. Davis, *et al.*, "A large-scale benchmark dataset for event recognition in surveillance video," in *CVPR 2011*, pp. 3153–3160, IEEE, 2011.