16811 Project report
Optimized Singular Value Decomposition Methods for Image compression

Akankshya Kar

December 2019

1 Introduction

The singular value decomposition (SVD) of a real matrix $A \in R^{m \times n}$ (of dimension $m \times n$) is represented as $A = U\Sigma V^T$ where U and V are the left and right singular vectors respectively and the diagonal elements of the matrix Σ are the singular values and describe the spectrum of the data. SVD method is used to solve systems of linear equations. In the field of Computer vision, lot of work has been done to turn the data around so that it can be solved using SVD. But as more data is available nowadays, traditional SVD techniques are not scaling well and are becoming computationally more expensive. In the field of Image compression, even as sparsity of the images are leveraged, SVD computation is still very slow and too expensive. Image compression in the field of Computer Vision is becoming an important area of research as more high-resolution images are being transferred. Despite increasing computational power, big data matrices in image processing is a big computational challenge for deterministic SVD algorithms. Therefore, an approximate SVD may be the only solution for many uses. Several innovative and useful methods to compute SVD of matrices useful for Computer Vision have been proposed in literature. Most of these methods utilize the ideas of (i) sparsification (is to replace the matrix by a surrogate that contains far fewer nonzero elements); (ii) column selection methods (assumes that a small set of columns describes most of the action of a numerically low-rank matrix); (iii) dimension reduction (Since the rows of a low-rank matrix are linearly dependent, they can be embedded into a low-dimensional space without substantially changing their geometric properties); (iv) or utilizing submatrix approximation (sub-collection of columns or rows or a square submatrix). However, one has to take note of the error penalty that the approximate low-rank SVD provides. The few methods of SVD among them that are discussed in this report are (i) Block Power SVD [4]; (iii) Tensor SVD [6]; (iv) Randomized SVD [3] and (v) Compressed SVD [1]. In this report, above methods are briefly described. The randomized SVD and compressed SVD methods have been implemented for a large data matrix and the results are compared.

2 SVD Methods

2.1 Block Power SVD

The Block Power method for SVD [2] utilizes the idea of classical power method to determine the k largest singular values and their corresponding singular vectors for a real rectangular matrix $A \in R^{m \times n}$. In the power method for a square matrix A, one picks a vector v and forms the sequence: $v; A_v; A_{2v}, \ldots\ldots$:In order to produce this sequence, explicit computation of the powers of A is not necessary and each vector in the sequence is obtained from the previous one by multiplying it by A. Block version of the power method uses the QR factorization at the normalization step. In QR factorization, if $A \in R^{m \times n}$ has linearly independent columns then it can be factored as $A = QR$ where Q is $m \times n$ with orthonormal columns ($Q^TQ = I$) and R is $n \times n$, upper triangular, with nonzero diagonal elements and R is nonsingular (diagonal elements are nonzero). SVD Power Method computes the dominant singular value $\Sigma 1 = \Sigma max$ of a rectangular real matrix and its associate right and left singular vectors U and V. The idea of Block SVD Power method is based on the above technique. From a block-vector $V_0 \in R^{m\times n}$,, two block-vector sequences V_k and U_k are constructed that converges respectively to the s first right and left singular vectors corresponding to singular values $\sigma 1$ $\sigma 2$ ...... σs,. In the QR Method for SVD, for a matrix $m \times n$., Initialization is done such that $T_0 = A$ and $S_0 = A^T$. For each k = 1; 2;...(until convergence) QR Factorization is used such that $T_k - 1 = U_k R_k$, $S_k - 1 = V_k Z_k T_k = R_k V_k and S_k = Z_k U$. This method is very simple and effective for computing all singular values.

2.2 Zoom-SVD

In a time series data, patterns and tendencies are encoded. It is important to reveal these hidden factors for which SVD methods are effectively being used. However, standard SVD methods store all the data in a time sequence thus becomes very expensive while computing latent factors in an arbitrary time range. Zoom-SVD [4] overcomes this difficulty in two phases of computation namely storage phase and query phase. The methods incrementally compresses multiple time series data block by block to reduce the space cost in storage phase, and efficiently computes SVD for a given time range query in query phase by carefully stitching stored SVD results. Therefore, the methods avoids reconstructing the original data from the SVD results computed in the storage phase. The method uses Partial-SVD and Stitched-SVD to compute SVD without reconstructing the original data corresponding to the query time range. Incremental SVD dynamically calculates the SVD of a matrix with newly arrived data rows. If $A \in R^{t\times c}$ represents the multiple time series data with t is time length, and c is the number of time series. The matrix A is divided into length-b blocks represented by $A(i) \in R^{b\times c}$. The low-rank approximation result of each block matrix A(i) are then stored, to exploit an incremental SVD method in the process. Only the SVD results which occupy less space than the original data are stored. In the query phase of zoom SVD, two sub-modules, Partial-SVD and Stitched-SVD, are used. For the block matrix A(S) and its SVD U(S)(S)V(S), partial-SVD first eliminates rows of left singular vector matrix U(S) which are out of the query time range. After that, partial-SVD multiplies the remaining left singular vector matrix $X_s U(S)$ with the singular value matrix $\Sigma(S)$, and performs SVD of the resulting matrix to provide the output of Partial-SVD. The remaining right singular vector matrix output of Partial-SVD is computed by multiplying the right singular vector matrix of partial SVD with V^T. The Stitched-SVD module combines the Partial-SVD of A(S) and A(E), and the stored SVD results of blocks matrices A(S+1), A(S+1), $\cdots$,A(E1) in the query time range $[t_s, t_e\ ]$ to return the final SVD result corresponding to the query range.

2.3 Tensor-Singular Value Decomposition (t-SVD)

All of the approaches to handle multi-linear data extend the nearly optimal 2-D SVD based vector space approach to the higher order ($N > 2$) case. This results in loss of optimality in the representation Tensor-Singular Value Decomposition (t-SVD) [5, 6] is based on an operator theoretic interpretation of third-order tensors as linear operators on the space of oriented matrices can be extended recursively to higher order tensors. The t-product C = A * B of $A \in R^{n_1\times n_2\times n_3}$ and $B \in R^{n_2\times n_4\times n_3}$ is a tensor of size $n_1 \times n_4 \times n_3$ where the $(i; j)^{th}$ tube denoted by C(i; j; :) for i = 1; 2; ::::; n_1 and j = 1; 2; ::::; n4 of the tensor C is given by $\sum_{k=1}^{n_2} \mathcal{A}(i,k,:) * \mathcal{B}(k,j,:)$ Tensor transpose A^T of the tensor A of size $n_1 \times n_2 \times n_3$ is the $n_2 \times n_1 \times n_3$ tensor obtained by transposing each of the frontal slices and then reversing the order of transposed frontal slices 2 through n_3. The t-product of A and B can be computed efficiently by performing the fast Fourier transformation (FFT) along the tube fibers of A and B to get A' and B', multiplying the each pair of the frontal slices of A' and B' to obtain C', and then taking the inverse FFT along the third mode to get the result (Kolda and bader, 2009) The t-SVD of a tensor $M \in R^{n_1\times n_2\times n_3}$, is given by $M = USV^T$ where U and V are orthogonal tensors of size $n_1 \times n_2 \times n_3$ and $n_2 \times n_2 \times n_3$ respectively. S is a rectangular f-diagonal tensor of size n1 x n2 x n3, and * denotes the t-product. The t-svd method is very useful for the problem of video completion and de-noising from random sparse corruptions, and has significant performance gains.

2.4 Randomised SVD

Computation of a low-rank approximation in Randomised SVD [3]to a given matrix involves (i) to construct a low-dimensional subspace that captures the action of the matrix and (ii) to restrict the matrix to the subspace and then compute a standard factorization (QR, SVD, etc.) of the reduced matrix. For the first step, we require a matrix Q for which Q has orthonormal columns and AQQ^TA. It is important that the basis matrix Q has as few columns as well as an accurate approximation of the input matrix. For this, random sampling methods are used very efficiently. The second step is implemented by using QR or SVD by forming B = QA, which yields the low-rank factorization A QB and then to compute an SVD of the small matrix: $B = U'\Sigma V^T$. And then to set U = QU' . When Q has few columns, this procedure is efficient because we can easily construct the reduced matrix B and rapidly compute its SVD. Given a matrix A, a target rank k, and an oversampling parameter p, a matrix Q with k + p orthonormal columns is constructed such that $\|A - QQ^T A\| \approx \min_{rank(X)\leq k} \|A - X\|$ Although there exists a minimizer Q that solves the fixed rank problem for p = 0, a small number of additional columns is crucial for the effectiveness of the computational methods. Given an m n matrix A, a target number k of singular vectors, and an exponent q (say q = 1 or q = 2), this procedure computes an approximate rank-2k factorization $U\Sigma V^T$, where U and V are orthonormal, and Σ is non-negative and diagonal. In the first step, an n 2k Gaussian test matrix Ω is generated where $Y = AA^TA\Omega$ by multiplying alternately with A and A. Then a matrix Q whose columns form an orthonormal basis for the range of Y is constructed. In the second step, B = QA is formed and SVD of the small matrix: B = U' V is computed. Computation is over by setting U = QU' . This method is inadequate in many applications because the singular spectrum of the input matrix may decay slowly. Therefore, q steps of a power iteration are introduced, where q = 1 or q = 2 usually suffices in practice. The Randomized SVD procedure requires only 2(q + 1) passes over the matrix, so it is efficient even for matrices stored out-of-core. The pairwise distances among a collection of N points in a Euclidean space are approximately maintained when the points are mapped randomly to a Euclidean space of dimension O(log N) (Lindenstrauss, 1974). This work has led to the concept of random embeddings which preserve Euclidean geometry and appropriate random low-dimensional embeddings preserve the geometry of point sets in finite-dimensional 1 spaces.

The algorithm for prototype for Randomized SVD involves he following steps for an m n matrix A, a target number k of singular vectors, and an exponent q (say q = 1 or q = 2) to compute rank-2k factorization in two stages. In Stage 1; (i) Generate an n 2k Gaussian test matrix; (ii) Form $Y = AA^TA\Omega$ by multiplying alternately with A and A^T; and (iii) Construct a matrix Q whose columns form an orthonormal basis for the range of Y. In Stage 2, the algorithm involves (i) Form B = QA; (ii) Compute an SVD of the small matrix: $B = U'\Sigma V^T$ and (iii) Set U = QU'

2.5 Compressed SVD

From an information retrieval perspective, it is necessary to understand the geometric relationships between the vectors, so that similarities and differences can be identified. The compressed matrix [1] must ensure that distances and angles between the data points are preserved. In order to obtain a low-rank SVD approximation of $X \in R^{m\times n}$ with target-rank k and m n, we require a random test matrix $\phi \in R^{l\times m}$ where l = k + p. Here p denotes a parameter for oversampling, and a small amount of oversampling is often sufficient. The compressed data matrix $Y \in R^{l\times n}$ as $Y := \phi X$. It may be noted that the singular values and vectors can be related to the eigen-decomposition of the inner and outer dot product of a matrix $A \in R^{m\times n}$ as $A^TA = (VSU^T)(USV^T) = VDV^T$, and $AA^T = (USV^T)(VSU^T) = UDU^T$where the eigenvalues are the squared singular values, i.e., D = S^2. Next, to approximate the right singular vectors by a smaller matrix $B \in R^{l\times l}$, $B := YY^T$is computed followed by computing the eigen-decomposition $B =: TDT^T$.

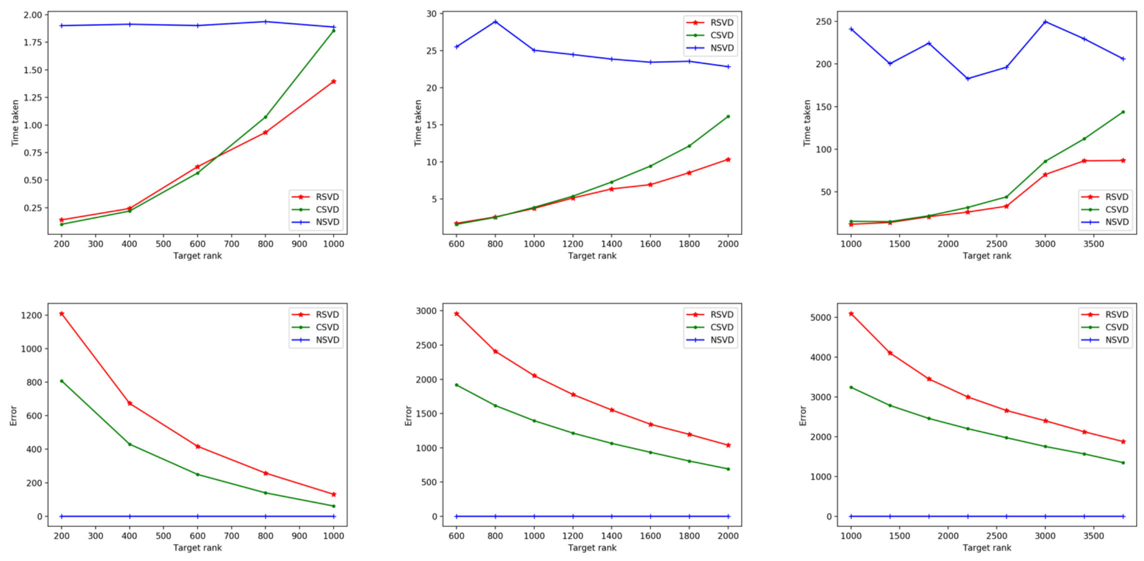
The matrix $T \in R^{l\times l}$ and $D \in R^{l\times l}$ are the approximate eigenvectors and eigenvalues, respectively. Hence, the approximate singular values are $S = \sqrt{D}$. If, $l > k$ the decomposition is truncated. The left or right singular vectors can be approximated from each other as $U = XVS^1$ or $V = X^TUS^{-1}$. Thus, the approximate right singular vectors $V' \in R^{n\times k}$ are recovered as $V' := Y^TTS^1$. Similarly, the approximate left singular vectors $U \in R^{m\times k}$ are obtained as $U` := XVS^1$. As the columns of U' are only approximately orthogonal due to the approximation errors in the eigenvalues, an additional step is introduced to compute the SVD of the scaled right singular vectors. Then, the last step is updates the right singular vectors V := V'Q which completes the approximate computation of the singular value decomposition

3 Results

The algorithms for randomized SVD (RSVD) and compressed SVD (CSVD) as well as the regular SVD (NSVD) were implemented in Python on the McBook Pro laptop (2.7 GHz Dual-Core Intel Core i5 with 8GB RAM). The images were obtained from google Arts and culture [1] which were used in [1].

[1]https://www.google.com/culturalinstitute/beta/asset/mwEV7sO9uSFCpw.

1079x1553 (k=600) **2848x4096 (k=1000)** **5696x8191 (k=1500)**



The first 3 rows are the results for image size and k rank. The 4th row is the time taken to compute one picture vs Rank and 5th row the Frobenius norm error vs Rank

The image with the largest dimension has 30,000 20,857 pixels which could not fit the memory of the computer. However, experimental test runs for SVD computation were carried out using images of 1079x1553, 2848x4096 and 5696x8191. Computational time requirement and errors due to dimension reduction were computed varying the target rank (k) of the reduced matrix. The images were reconstructed using the reduced dimension SVDs. The results are shown in Fig 2. It is found that the reconstructed images using CSVD have realistically reproduced the images. The image using higher number of pixels and with target k of 1500 is closest to the original image. Examination of computational time taken reveals that CSVD consumes more processing time than RSVD after some target k is exceeded e.g. both the procedures take same time till k=2500 for 5696x8191 size matric. However, once the k value increases, the CSVD need more time. The NCVD procedure always takes more time for all k for the images of all dimension size. As expected, the NSVD method always has the least errors. For a given image, the CSVD has better skill in terms accuracy than the RSVD. The difference in accuracy is especially large with lower threshold value of k. Therefore, CSVD method can be safely used for image compression.

4 Conclusion

There are several innovative algorithms for SVD computation useful for image compression. Few of them have been discussed here briefly. The randomized SVD and compressed SVD methods have been implemented and tested. It is found that for a given image and target threshold value of k, the CSVD method produces better results in terms of accuracy of reconstructed image and computer time requirement.

References

[1] N Benjamin Erichson, Steven L Brunton, and J Nathan Kutz. Compressed singular value decomposition for image and video processing. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1880–1888, 2017.

[2] AH Bentbib and A Kanber. Block power method for svd decomposition. *Analele Universitatii" Ovidius" Constanta-Seria Matematica*, 23(2):45–58, 2015.

[3] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[4] Jun-Gi Jang, Dongjin Choi, Jinhong Jung, and U Kang. Zoom-svd: Fast and memory efficient method for extracting key patterns in an arbitrary time range. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1083–1092. ACM, 2018.

[5] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[6] Zemin Zhang, Gregory Ely, Shuchin Aeron, Ning Hao, and Misha Kilmer. Novel methods for multilinear data completion and de-noising based on tensor-svd. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3842–3849, 2014.