Randomized SVD Project Report

Member: Zixuan Shao(zixuans2), Josh Holder(jholder3)

Abstract

The purpose of the project is to compare the effectiveness of three low rank matrix approximation algorithms when dealing with large size matrix in practice. We did two numerical experiments to examine these algorithms. The first experiment uses artificial low rank matrices to evaluate the computational cost and approximation error of three algorithms when we have a fixed large input matrix. The second experiment focus on the application of these algorithms on a medium size face image dataset by applying principal component analysis and turns out randomized SVD is effective in image information storage and compression of large dataset.

Introduction

We used three SVD algorithms: Classical SVD, Basic randomized SVD and Single-pass randomized for general matrix.

Classical SVD:
Input: An m x n matrix A, a target rank k.
Output: Matrices U, D, V in an approximate rank-k of A.
(1) Take a standard SVD of A in reduced form and get u,d,v
(2) Take first k dominant entries as output

RSVD – Basic Randomized SVD (Martinson 9)
Input: An m x n matrix A, a target rank k, an oversampling parameter p.
Output: Matrices U, D, V in an approximate rank-k of A.
Stage A:
(1) Form an n x (k + p) Gaussian random matrix G.
(2) Form the sample matrix Y = A G
(3) Orthonormalize the columns of the sample matrix Q = orth(Y)
Stage B:
(4) Form the (k + p) x n matrix $B = Q^* A$
(5) Form the SVD of B: $B = U_0 DV$
(6) Form $U = QU_0$
(7) Drop the p element in U,D,V

Basically, stage A is used to approximate the range of A by using gaussian random matrix, which has almost linearly independent column vector so that the linear

combination of the vectors is not in null space of A and then Y is a linear independent set and span the range of A. The oversampling parameter prevents the gaussian matrix from failing to fully capture the action of A because of the perturbation of A, an increased sample size of random vector holds a better probability to span the range of A. In stage B, we access the input matrix again and compute the SVD, and since we just want rank-k matrix, we can choose the k dominant elements in the end. The following formula reveals the computation behind the algorithm.

$$A \approx QQ^*A = QB = QU_0DV^* = UDV^*$$

Single-pass Randomized SVD For A General Matrix (Martinsson 12)

Input: An m x n matrix A, a target rank k, an oversampling parameter p.
Output: Matrices U, D, V in an approximate rank-k of A.
Stage A:
(1) Form two Gaussian random matrices G_c and G_r of sizes n x (k + p) and m x (k + p), respectively.
(2) Form the sample matrices $Y_c = AG_c$ and $Y_r = A^* G_r$
(3) Form orthonormal matrices Q_c and Q_r consisting of the k dominant left singular vectors of Y_c and Yr.
Stage B:
(4) Let C denote the k x k least squares solution of the joint system of equations formed by $(G_r^*Q_c)\, C = Y_r^*Q_r$ and $C\,(Q_r^*G_c) = Q_c^*Y_c$
(5) Compute the SVD of C so that $C = \underline{U}D\underline{V}$
(6) Form $U = Q_c\underline{U}$ and $V = Q_r\underline{V}$.

Compared to the previous algorithm, one advantage of this algorithm is illustrated its name. Single-pass means that the algorithm only accesses the input matrix once in stage A while our RSVD access input matrix again in stage B. When the input matrix is too large to fit into the memory, transferring the matrix is extremely expensive. In such case, our single-pass randomized SVD comes in. Since the input matrix is not Hermitian, we have to capture both column space and row space in stage A and we have two equations right now.

$$G_r^*Q_cC = G_r^*Q_cQ_c^*AQ_r \approx G_r^*AQ_r = Y_r^*Q_r$$

$$CQ_r^*G_c = Q_c^*AQ_rQ_r^*G_c \approx Q_c^*AG_c = Q_c^*Y_c$$

$$\text{Where } C = Q_c^*AQ_r = Q_c^*UDVQ_r$$

Therefore, in stage B, we only need to solve a linear system of least square equations

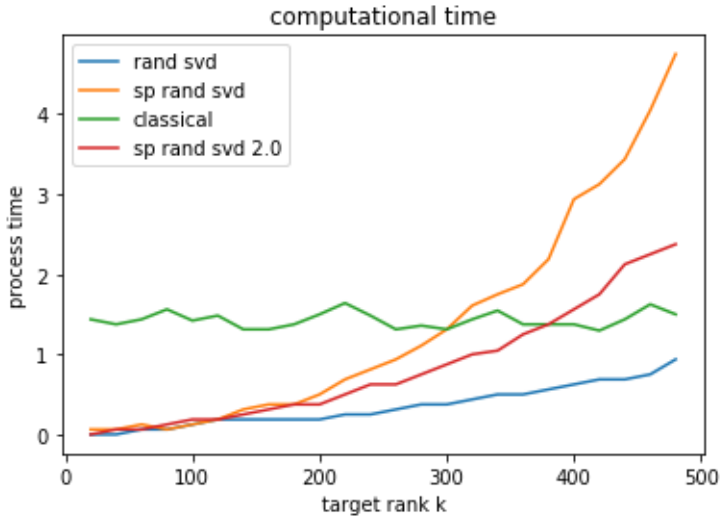
for the matrix C. We come up with two ways to solve the linear system. The first one is to construct the two equations into one equation and solving the merged version. The second one is to compare the residual of the two least square equations and choose the one with the least residual. It turns out the second implementation is better with regard to computational cost of the input matrix, which will be explained in detail in numerical experiment 1.

Numerical Experiment I

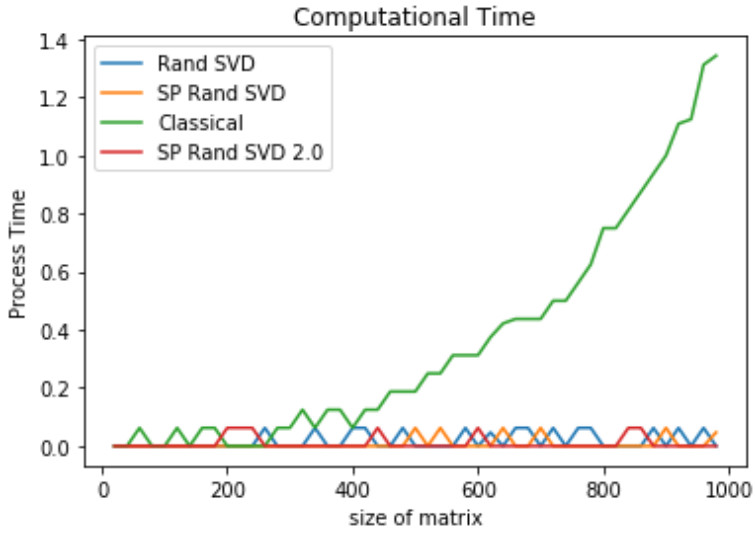
The first numerical experiment focuses on accuracy and execution time of these algorithms. In particular, we need to compare the computational cost and approximation error of input matrix.

Computational Cost

We compare the computational cost by using the CPU processing time in different target ranks with a random input matrix of shape 1000 x 1000 as shown on the figure below.



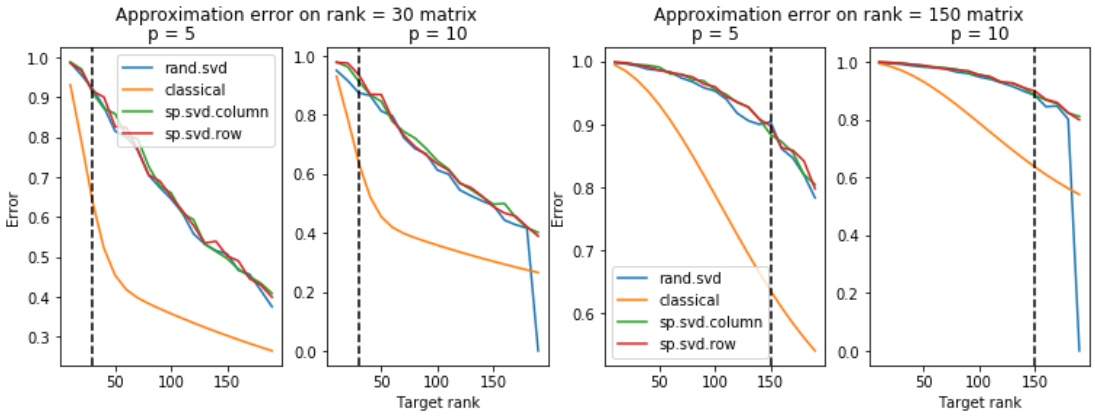
First looking at the two versions of single-pass randomized SVD, label “sp rand SVD 2.0” corresponds to the one comparing residual, which has a slower increasing rate as k increases. Although we don’t know what algorithm numpy.linalg.lstsq uses, we could somehow interpret the complexity is at least $O(k^2)$ from the gradient. Hence, a double-sized merged version of least square equation should have a larger cost than two equations with same size. The classical SVD has a complexity not depending on target rank and actually has $O(mn^2+n^3)$. The RSVD has a complexity O(mnk) which comes from Y = AG matrix-matrix multiplication, and the single-pass SVD performs a bit worse because it also includes a further least square solver with at least $O(k^2)$ besides the matrix-matrix multiplication in stage A. In general, randomized SVD is much more efficient than classical SVD although randomized SVD has an increasing target rank, but we can see from the graph that they intersect at a very large target rank, so in the case of low rank matrix approximation, this situation won’t happen.



If we fix the rank of matrix, we can see the advantage of randomized SVD even more clearly as the size of the matrices increases from the figure above.

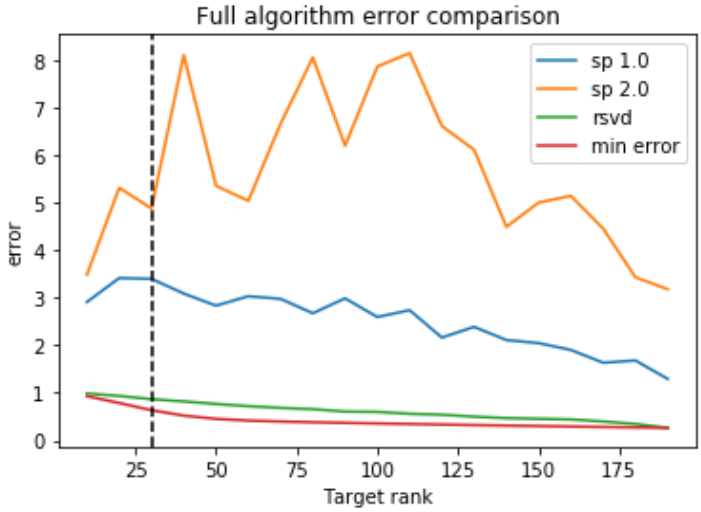
Approximation Error of input matrix

To compare accuracy, we firstly investigate on the performance of randomized SVD in stage A and extract k+1th singular value from classical SVD as a minimal error. Approximation error of input matrix A has an expression $||A-QQ^*A||$ where Q is the range approximation matrix we obtain from stage A and we use spectral norm (induced 2-norm) in practice.



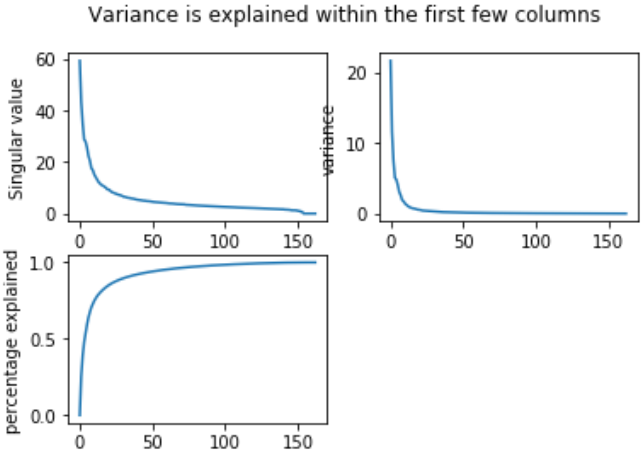
As the figure shown, randomized SVD using gaussian random matrix has a slightly greater error than minimal error from classical SVD. The figure below shows the error from the whole algorithm including stage B for a rank 30 matrix and turns out RSVD's error is still close to minimal error but two versions of single pass SVD have quite large error which could be introduced by the least square equation in stage B. If we just compare the error between two versions of single pass SVD, version 1 not only has a relatively smaller one but also a smoothly decaying pattern. Including the observation from computational, two versions of single pass SVD do have a tradeoff between computational cost and approximation error, and in general, RSVD has a

very good performance in both.



Numerical Experiment II: Eigenface

In this section, we focus on real-life application of randomized SVD on large dense image matrix and perform principal component analysis using our algorithms on a medium size face image dataset from Yale face database, which are transformed to a single matrix so that each column vector is a flatten version of a face image. Before starting with PCA, we have to identify a suitable target rank and oversampling parameter so that target rank is both relatively small and accurate. We first did a classical SVD on the matrix and turns out 89.7% of the variance is stored in the first thirty eigenvectors, so we would only focus on the first thirty rank afterwards.

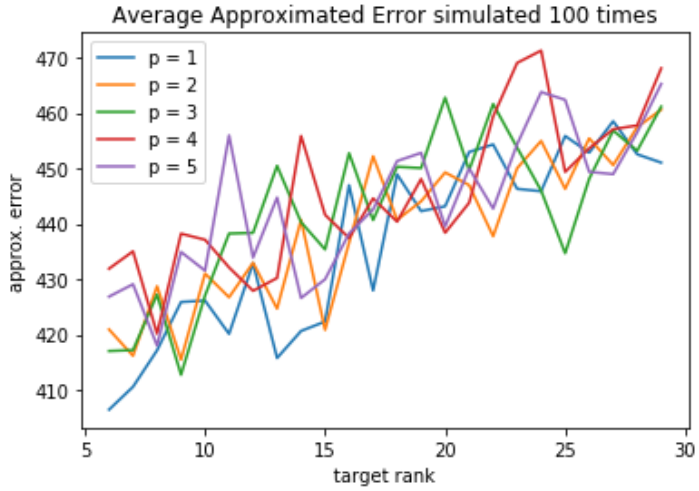


Approximation Error of Face Matrix

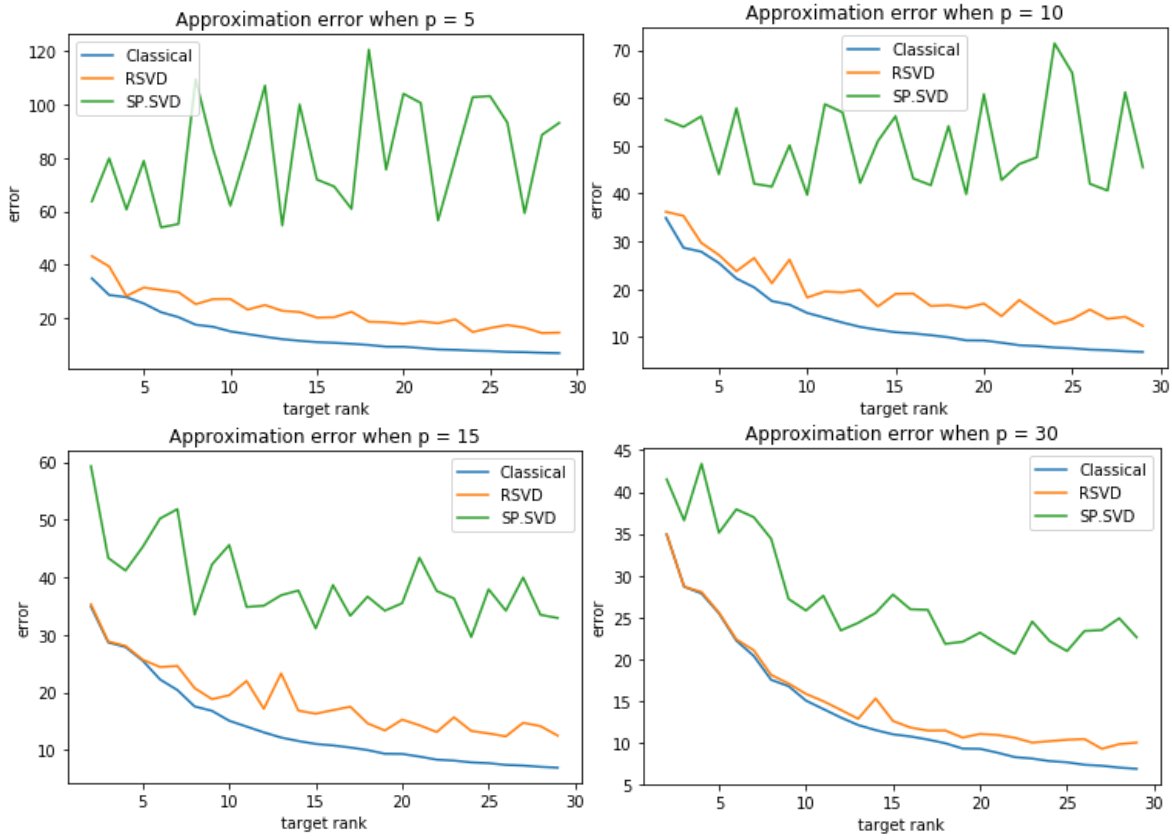
Our first approach is to find the appropriate range approximation matrix in stage A, but we couldn't implement the usual spectral norm due to the size of input matrix. Instead, we try to implement the posterior error estimation from the lemma 4.3. (Halko 241)

$$\|(\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^*)\boldsymbol{A}\| \leq 10\sqrt{\frac{2}{\pi}} \max_{i=1,\ldots,r} \left\|(\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^*)\boldsymbol{A}\boldsymbol{\omega}^{(i)}\right\|$$

where w_i is drawn from a sequence of standard gaussian vectors of size r and such inequality holds with a probability at least $1\text{-}10^{-r}$. Since this estimate is a random variable, we ran a simulation 100 times with 5 random vectors then took the average for each target rank and some oversampling parameters p. The result contradicts with what we expected, in which the approximate error maintains a tendency of rising up although the pattern is somehow very random. However, intuitively, the more ranks we choose, the less error we could get from the estimation. Therefore, we switch to another general approach.



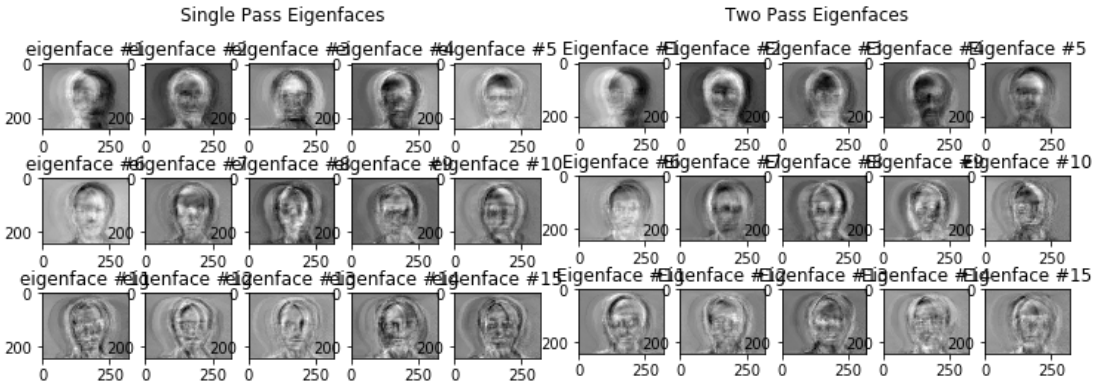
Our second approach is similar to the one in the first numerical experiment: we simply take the spectral norm of the difference between face matrix and output SVD of the whole algorithm, $\|A - UDV^*\|$.



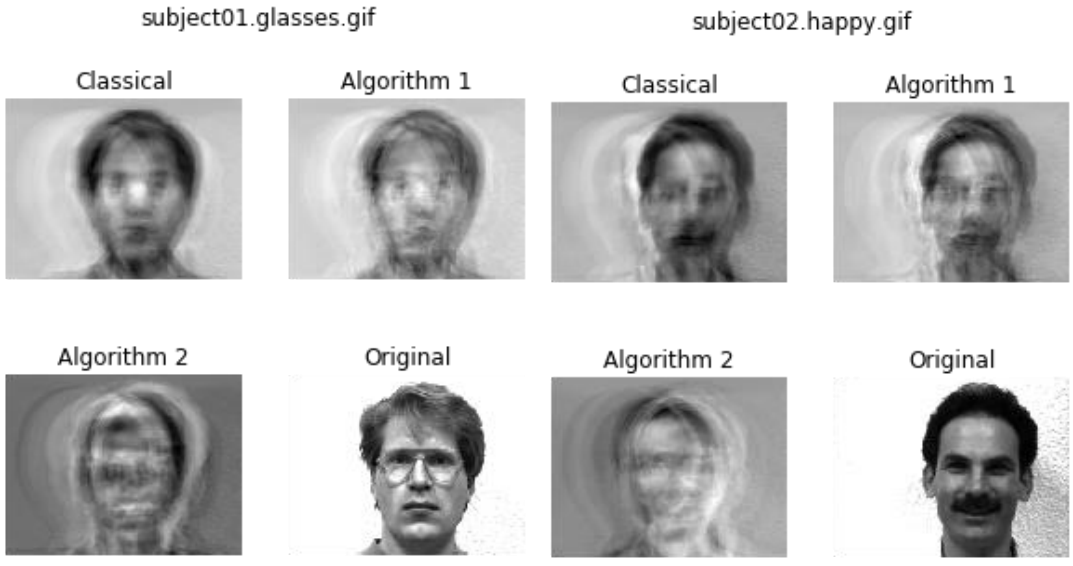
This result has more satisfying outcome which is similar to the first experiment, although the computational time is enormously long and in order to save more time, we choose version two of single-pass algorithm, the one with faster speed and we could expect the position of version one sitting in between version two and RSVD. From the figure above, we can see that RSVD is really close to minimal error while single-pass has relatively large and fluctuate error. Surprisingly, as we increase oversampling value, the error pattern improves a lot, especially for single-pass algorithm, which almost transforms to a steady decreasing pattern. For the RSVD, its low rank estimate almost coincides with the minimal error. However, as oversampling parameter increases to large value, its effect becomes less significant. In order to achieve a balance between approximation error and cost (time and space), we decide to choose target rank 15 and oversampling parameter 30.

Principal component analysis and image reconstruction

In this section, we start our principal component analysis and obtain eigenfaces from left dominant eigenvectors, and try to reconstruct a specific image from the eigenfaces.



As the figure shown, the dominant eigenfaces have clear face contour while the last few are much noisier.



For the image reconstruction, we construct a least square equation Uw=s where U is the eigenface matrix, w is the weight vector for a specific image vector and s is the image vector chosen from the database, solving for weight vector. Then we construct the matrix back from Uw and compare to the original image. The above figure shows two face image examples and we could see that classical algorithm restores the original image most precisely, following by RSVD and single-pass algorithm. RSVD and classical algorithm almost don't show much difference but single-pass algorithm performs so bad that the restored image shows no similarity with the original one. By applying the method to the 320 x 243 x 164 database, we could store most information in a (320 x 243) x 15 eigenface matrix and 15 x 164 weight matrix. Since the database intentionally includes nonstandard lighting and position which introduces error in our eigenfaces, the result could be further improved by using more standard face image.

Conclusion

Despite the problem of dealing with slow memory, RSVD algorithm performs much better than single-pass randomized SVD regarding both computational speed and accuracy in the numerical experiments. If we could properly choose the target rank and oversampling parameter, accuracy of RSVD can be improve to the same level as the classical SVD. Furthermore, RSVD has great advantage in terms of computational speed compared to classical one. Therefore, in general, RSVD maintains the best performance among the three algorithms. With regard to parameter, the oversampling parameters has a positive impact on the approximation accuracy but target rank holds a tradeoff between accuracy and speed in which case it is more difficult to choose an appropriate target rank when dealing with a real-life dataset.

References

Halko, N., et al. "Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions." *SIAM Review*, vol. 53, no. 2, 5 May 2011, pp. 217–288., doi:10.1137/090771806.

Martinsson, Per-Gunnar. "Randomized Methods for Matrix Computations." *IAS/Park City Mathematics Series The Mathematics of Data*, 7 Feb. 2019, pp. 187–229., doi:10.1090/pcms/025/04.

"UCSD Computer Vision." *Yale Face Database*, vision.ucsd.edu/content/yale-face-database.