Evaluating and improving streaming methods for large scale SVD problems

Andreas Stathopoulos, Jeremy Myers, Toon Tran

Abstract

Large scale eigenvalue and singular value problems are typically solved using iterative methods [10, 12]. For extreme scale matrix sizes randomized projection methods can be much faster while still delivering sufficient accuracy, measured as the distance from the optimal low rank matrix. The accuracy can be improved by following the projection with subspace iteration [7, 8]. However, achieving high accuracy for individual singular vectors is generally more challenging.

In this work we address the problem where the matrix is so large that cannot be stored in its entirety and its re-computation is either too expensive or not possible; hence even iterative methods are infeasible. This situation is becoming increasingly common in the era of "bigger" data and is often referred to as streaming, i.e., when the data arrives in some order, is processed, and then forgotten. In terms of matrices, we assume that a matrix is streamed in m linear updates (see [14]), $A = \sum_{i=1}^{m} H_i$, but we focus our attention to streaming by rows, i.e., H_i is a set of rows of A.

Randomized methods are naturally suited for streaming. When a new H_i arrives, its randomized projection is recorded, and the method continues until the entire matrix has been streamed, at which point an approximate SVD can be computed from the projection. A series of improvements on this basic idea [18, 15, 16, 13] have resulted in an efficient randomized method called SketchySVD [14].

A different class of deterministic streaming SVD methods has been proposed but has not received as much attention despite its potential for more accurate approximations. We use Incremental SVD (ISVD) as a prototype such method. Inductively at step $i+1$, ISVD appends the new window H_{i+1} to an existing rank-k approximation $B^{(i)}$, computes the SVD of $[B^{(i)}; H_{i+1}]$, and then updates $B^{(i+1)}$ based on the rank-k truncation of the SVD. Earlier works [9, 2, 4, 3, 5, 20] compute only the left or right singular vectors, while the ISVD of Baker et al. [1] generalized these approaches to track both left and right singular vectors albeit at a higher computational cost.

A notable difference is that ISVD provides a running low-rank approximation at every window, while SketchySVD can wait till the end of streaming to compute it. Broadly speaking, randomized sketching methods have low time and space complexity whereas deterministic sketching methods have higher accuracy. However, the trade-offs have not been carefully studied in the literature. Empirical results with randomized sketching methods do not compare with streaming or use datasets that are typically small enough to be processed by batch methods [14]. In this work we explore these missing comparisons and we introduce some new ideas for improving ISVD. Our contributions can be summarized as follows:

- Traditional ISVD methods update the low rank approximation one row or a small number of rows at a time. Because ISVD accuracy improves with larger window size (number of rows in H_i), we instead make the window size as large as memory can hold. Because only a low rank approximation is needed, iterative methods can be used to solve the partial SVD of the large rectangular window.

- To evaluate the benefits of these streaming methods we need to address enormous problem sizes. For this reason, we provide a high-performance C++ implementation of both SketchySVD and ISVD called Skema (available at `https://github.com/jeremy-myers/`

`skema`). To perform dense and sparse matrix-vector multiplications (matvecs), Skema leverages the Kokkos Performance Portability Ecosystem [6] for hierarchical parallelism on heterogeneous architectures, including x86 and accelerators. To compute a few eigenpairs or singular triplets, Skema uses the PRIMME [11, 19] library. Dense matrix operations inside PRIMME utilize multithreaded BLAS on CPUs and MAGMA on accelerators.

- We provide a complexity analysis of the "large window" streaming method and compare it with a similar analysis of SketchySVD. We show that ISVD is more expensive than the same iterative method applied to the entire matrix for the same number of iterations. The overhead is proportional to the number of windows, especially for very sparse matrices. Therefore, choosing larger window size not only improves accuracy but also reduces the time overhead.

- We also perform extensive numerical results on problems with enormous dimensions that are much larger than those in the literature. Sources of problems include: stock price prediction (kernel learning), social networks/analysis graphs, and scientific simulation data. We observe that ISVD approximations are at least as accurate as SketchySVD ones and often several orders of magnitude more accurate. Comparing runtimes, SketchySVD is typically faster, but not as much as the complexity analysis suggests. This is because dense Gaussian embeddings involve too many operations while sparsemaps implementations present a challenging memory access pattern.

- Using iterative methods as the SVD solver at each window allows the use of initial guesses which are readily available from the previous window, i.e., $B^{(i)}$ transformed appropriately to correspond to the $[B^{(i)}H_i]$ matrix. Since the low rank approximations $B^{(i)}$ change relatively slowly between windows, initial guesses provide a substantial reduction in the number of iterations, around 30-50%.

- ISVD allows for further optimizations when solving for the largest eigenvalues of a symmetric positive definite (SPD) matrix. Such problems are common in large graph Laplacians, covariance matrices, and kernel methods in machine learning. Since the SVD and the eigenvalue problem are equivalent for SPD matrices, any right singular vector v of the rectangular window at any ISVD step is an approximation to an eigenvector of A. Therefore, for any row m of A we can compute the m-th value of the eigenvalue residual as $A(m,:)v - \lambda v(m)$. This motivates the following convergence criterion for each ISVD window. Notice that while the iterative method converges to the SVD of the rectangular window, the corresponding eigenvalue residuals for A stop making progress after some iterations. If we can estimate the A residuals we can check for this and stop early. We use reservoir sampling [17] to create and store a subset of rows of A, A_S, for which we can estimate the residual values and extrapolate the residual norm to the entire matrix. Reservoir sampling is a method to maintain a uniform sample of elements that have been streamed up to now. Preliminary results on this idea have been promising for further reducing the number of iterations.

- In some cases, the streaming order of rows can be chosen by the user. An example is when a row of a covariance or a kernel matrix is computed on demand from the data (data requires $O(n)$ storage vs $O(n^2)$ storage for the entire matrix). Therefore, the question arises of what is the effect of streaming order in the final accuracy of the low rank space, and whether this is achieved early or late during streaming. Based on the convergence analysis of [1] we show that, on average, each window provides a similar additive improvement on accuracy. This means that ISVD does not see the best accuracy until the last window. We have observed that if rows are streamed in the order of decreasing row norms of A, the final accuracy is

achieved very early in streaming. Thus, we explore the idea of using the row norms of the low rank approximation $B^{(i)}$ as leverage scores to stream first the remaining rows with the largest norms. This heuristic also achieves a similar behavior where most of the accuracy is achieved earlier. If combined with our residual norm estimation using reservoir sampling, this heuristic may suggest stopping the streaming before all windows have been streamed. This approach is still under investigation.

References

[1] Chris G. Baker, Kyle A. Gallivan, and Paul Van Dooren. Low-rank incremental methods for computing dominant singular subspaces. *Linear Algebra and its Applications*, 436(8):2866–2888, 2012.

[2] Matthew Brand. Incremental Singular Value Decomposition of Uncertain Data with Missing Values. In Anders Heyden, Gunnar Sparr, Mads Nielsen, and Peter Johansen, editors, *Computer Vision — ECCV 2002*, volume 2350 of *Lecture Notes in Computer Science*, pages 707–720, Berlin, 2002. Springer.

[3] Matthew Brand. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications*, 415(1):20–30, 2006.

[4] Y. Chahlaoui, K. Gallivan, and P. Van Dooren. Recursive calculation of dominant singular subspaces. *SIAM Journal on Matrix Analysis and Applications*, 25(2):445–463, 2003.

[5] Yongsheng Cheng, Jiang Zhu, and Xiaokang Lin. An enhanced incremental SVD algorithm for change point detection in dynamic networks. *IEEE access : practical innovations, open solutions*, 6:75442–75451, 2018.

[6] H. Carter Edwards, Christian R. Trott, and Daniel Sunderland. Kokkos: Enabling manycore performance portability through polymorphic memory access patterns. *Journal of Parallel and Distributed Computing*, 74(12):3202–3216, December 2014.

[7] M. Gu. Subspace Iteration Randomization and Singular Value Problems. *SIAM Journal on Scientific Computing*, 37(3):A1139–A1173, 2015.

[8] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. *Siam Review*, 53(2):217–288, May 2011.

[9] A. Levey and M. Lindenbaum. Sequential Karhunen-Loeve basis extraction and its application to images. *IEEE Transactions on Image Processing*, 9(8):1371–1374, 2000.

[10] Y. Saad. *Numerical methods for large eigenvalue problems.* Manchester University Press, 1992.

[11] Andreas Stathopoulos and James R. McCombs. PRIMME: PReconditioned Iterative Multi-Method Eigensolver – Methods and software description. *ACM Transactions on Mathematical Software*, 37(2):21:1–21:30, 2010.

[12] G. W. Stewart. *Matrix Algorithms Vol.II: Eigensystems.* SIAM, Philadelphia, PA, 2001.

[13] Joel A. Tropp, Alp Yurtsever, Madeleine Udell, and Volkan Cevher. Fixed-Rank Approximation of a Positive-Semidefinite Matrix from Streaming Data. *CoRR*, abs/1706.05736, 2017.

[14] Joel A. Tropp, Alp Yurtsever, Madeleine Udell, and Volkan Cevher. Streaming Low-Rank Matrix Approximation with an Application to Scientific Simulation. *SIAM Journal on Scientific Computing*, 41(4):A2430–A2463, 2019.

[15] Jalaj Upadhyay. Fast and space-optimal low-rank factorization in the streaming model with application in differential privacy, April 2016.

[16] Jalaj Upadhyay. The price of privacy for low-rank factorization. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 4180–4191, Red Hook, NY, USA, 2018. Curran Associates Inc.

[17] Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11(1):37–57, March 1985.

[18] Franco Woolfe, Edo Liberty, Vladimir Rokhlin, and Mark Tygert. A fast randomized algorithm for the approximation of matrices. *Applied and Computational Harmonic Analysis*, 25(3):335–366, November 2008.

[19] Lingfei Wu, Eloy Romero, and Andreas Stathopoulos. PRIMME_SVDS: A High-Performance Preconditioned SVD Solver for Accurate Large-Scale Computations. *Siam Journal On Scientific Computing*, 39(5):S248–S271, 2017.

[20] Yangwen Zhang. An answer to an open question in the incremental SVD, 2022.