Tighter Low-rank Approximation via Sampling the Leveraged Element*

Srinadh Bhojanapalli
The University of Texas at Austin
bsrinadh@utexas.edu

Prateek Jain
Microsoft Research, India
prajain@microsoft.com

Sujay Sanghavi
The University of Texas at Austin
sanghavi@mail.utexas.edu

Abstract

In this work, we propose a new randomized algorithm for computing a low-rank approximation to a given matrix. Taking an approach different from existing literature, our method first involves a specific biased sampling, with an element being chosen based on the leverage scores of its row and column, and then involves weighted alternating minimization over the factored form of the intended low-rank matrix, to minimize error only on these samples. Our method can leverage input sparsity, yet produce approximations in *spectral* (as opposed to the Frobenius) norm; this combines the best aspects of otherwise disparate current results, but with a dependence on the condition number $\kappa = \sigma_1/\sigma_r$. In particular we require $O(nnz(M) + \frac{n\kappa^2 r^5}{\epsilon^2})$ computations to generate a rank-r approximation to M in spectral norm. In contrast, the best existing method requires $O(nnz(M) + \frac{nr^2}{\epsilon^4})$ time to compute an approximation in Frobenius norm. Besides the tightness in spectral norm, we have a better dependence on the error ϵ. Our method is naturally and highly parallelizable.

This approach also leads to a new method to directly compute a low-rank approximation (in efficient factored form) to the product of two given matrices; it computes a small random set of entries of the product, and then executes weighted alternating minimization (as before) on these.

1 Introduction

Finding a low-rank approximation to a matrix is fundamental to a wide array of machine learning techniques. The large sizes of modern data matrices has driven much recent work into efficient (typically randomized) methods to find low-rank approximations that do not exactly minimize the residual, but run much faster / parallel, with fewer passes over the data. Existing approaches involve either intelligent sampling of a few rows / columns of the matrix, projections onto lower-dimensional spaces, or sampling of entries followed by a top-r SVD of the resulting matrix (with unsampled entries set to 0).

We pursue a different approach: we first sample entries in a specific biased random way, and then minimize the error on these samples over a search space that is the factored form of the low-rank matrix we are trying to find. We note that this is different from approximating a 0-filled matrix; it is instead reminiscent of matrix completion in the sense that it only looks at errors on the sampled entries [2]. Another crucial ingredient is how the sampling is done: we use a combination of ℓ_1 sampling, and of a distribution where the probability of an element is proportional to the sum of the leverage scores of its row and its column.

*Full version is available at arXiv:1410.3886

Reference	Frobenius norm	Spectral norm	Computation time
BJS14 (Our Algorithm)	$(1+\epsilon)\|\Delta\|_F$	$\|\Delta\| + \epsilon\|\Delta\|_F$	$O(nnz(M) + \frac{nr^5\kappa^2\log(n)}{\epsilon^2})$
CW13[3]	$(1+\epsilon)\|\Delta\|_F$	$(1+\epsilon)\|\Delta\|_F$	$O(nnz(M) + \frac{nr^2}{\epsilon^4} + \frac{r^3}{\epsilon^5})$
BG13 [1]	$(1+\epsilon)\|\Delta\|_F$	$c\|\Delta\| + \epsilon\|\Delta\|_F$	$O(n^2(\frac{r+\log(n)}{\epsilon^2}) + n\frac{r^2\log(n)^2}{\epsilon^4})$
NDT09[4]	$(1+\epsilon)\|\Delta\|_F$	$c\|\Delta\| + \epsilon\sqrt{n}\|\Delta\|$	$O(n^2\log(\frac{r\log(n)}{\epsilon}) + \frac{nr^2\log(n)^2}{\epsilon^4})$
WLRT08[6]	$(1+\epsilon)\|\Delta\|_F$	$\|\Delta\| + \epsilon\sqrt{n}\|\Delta\|$	$O(n^2\log(\frac{r}{\epsilon}) + \frac{nr^4}{\epsilon^4})$
Sar06[5]	$(1+\epsilon)\|\Delta\|_F$	$(1+\epsilon)\|\Delta\|_F$	$O(nnz(M)\frac{r}{\epsilon} + n\frac{r^2}{\epsilon^2})$

Table 1: Comparison of error rates and computation time of some low rank approximation algorithms. $\Delta = M - M_r$.

Both the sampling and the subsequent alternating minimization are naturally fast, parallelizable, and able to utilize sparsity in the input matrix. Existing literature has either focused on running in input sparsity time but approximation in (the weaker) Frobenius norm, or running in $O(n^2)$ time with approximation in spectral norm (see Table 1). Our method provides the best of both worlds: it runs in input sparsity time, with just two passes over the data matrix, and yields an approximation in spectral norm. It does however have a dependence on the ratio of the first to the r^{th} singular value of the matrix.

Our alternative approach also yields new methods for the problem of directly finding the low-rank approximation of the product of two given matrices.

Notation: Capital letter M typically denotes a matrix. M^i denotes the i-th row of M, M_j denotes the j-th column of M, and M_{ij} denotes the (i,j)-th element of M. Unless specified otherwise, $M \in \mathbb{R}^{n\times d}$ and M_r is the best rank-r approximation of M. Also, $M_r = U^*\Sigma^*(V^*)^T$ denotes the SVD of M_r. $\kappa = \sigma_1^*/\sigma_r^*$ denotes the condition number of M_r, where σ_i^* is the i-th singular value of M. $\|u\|$ denotes the L_2 norm of vector u. Let $\|M\|, \|M\|_F$ denote the Spectral and Frobenius norm of M respectively. Also, $\|M\|_{1,1} = \sum_{ij}|M_{ij}|$. $dist(X,Y) = \|X_\perp^T Y\|$ denotes the principal angle based distance between subspaces spanned by X and Y orthonormal matrices.

Given a set $\Omega \subseteq [n]\times[d]$, $P_\Omega(M)$ is given by: $P_\Omega(M)(i,j) = M_{ij}$ if $(i,j)\in\Omega$ and 0 otherwise. $R_\Omega(M) = w.*P_\Omega(M)$ denotes the Hadamard product of w and $P_\Omega(M)$. That is, $R_\Omega(M)(i,j) = w_{ij}M_{ij}$ if $(i,j)\in\Omega$ and 0 otherwise. Similarly let $R_\Omega^{1/2}(M)(i,j) = \sqrt{w_{ij}}M_{ij}$ if $(i,j)\in\Omega$ and 0 otherwise.

2 Low-rank Approximation of Matrices

In this section we will present our main contribution: a new randomized algorithm for computing low-rank approximation of any given matrix. Our algorithm first samples a few elements from the given matrix $M \in \mathbb{R}^{n\times d}$, and then rank-$r$ approximation is computed using only those samples. Algorithm 1 provides a detailed pseudo-code of our algorithm.

The first step is sampling according to the distribution given in eq. (2). Computationally, our sampling procedure can be done in two passes and $O(m\log n + nnz(M))$ time. In our second step, we minimize the following non-convex problem by weighted alternating minimization 2.

$$\min_{U\in\mathbb{R}^{n\times r}, V\in\mathbb{R}^{d\times r}} \sum_{(i,j)\in\Omega} w_{ij}\left(M_{ij} - (UV^T)_{ij}\right)^2, \tag{1}$$

where $w_{ij} = 1/\hat{q}_{ij}$ when $\hat{q}_{ij} > 0$, 0 else.

We now provide our main result for low-rank approximation and show that Algorithm 1 can provide a tight approximation to M_r while using a small number of samples $m = \mathbb{E}[|\Omega|]$.

Theorem 2.1. *Let $M \in \mathbb{R}^{n\times d}$ be any given matrix ($n \geq d$) and let M_r be the best rank-r approximation to M. Set the number of samples $m = \frac{C}{\gamma}\frac{nr^3}{\epsilon^2}\kappa^2\log(n)\log^2(\frac{\|M\|}{\zeta})$, where $C > 0$ is any global constant, $\kappa = \sigma_1/\sigma_r$ where σ_i is the i-th singular value of M. Also, set the number of iterations of WAltMin procedure to be $T = \log(\frac{\|M\|}{\zeta})$. Then, with probability greater than $1-\gamma$*

Algorithm 1 LELA: Leveraged Element Low-rank Approximation

input matrix: $M \in \mathbb{R}^{n\times d}$, rank: r, number of samples: m, number of iterations: T

1: Sample $\Omega \subseteq [n] \times [d]$ where each element is sampled independently with probability: $\hat{q}_{ij} = \min\{q_{ij}, 1\}$

$$q_{ij} = m \cdot \left(\frac{\|M^i\|^2 + \|M_j\|^2}{2(n+d)\|M\|_F^2} + \frac{|M_{ij}|}{2\|M\|_{1,1}} \right). \tag{2}$$

2: Obtain $P_\Omega(M)$ using one pass over M

3: $\widehat{M}_r = \mathsf{WAltMin}(P_\Omega(M), \Omega, r, \hat{q}, T)$

output $\widehat{M}_r$

Sub-routine 2 WAltMin: Weighted Alternating Minimization

input $P_\Omega(M),\ \Omega,\ r,\ \hat{q},\ T$

1: $w_{ij} = 1/\hat{q}_{ij}$ when $\hat{q}_{ij} > 0$, 0 else, $\forall i, j$

2: Divide Ω in $2T+1$ equal uniformly random subsets, i.e., $\Omega = \{\Omega_0, \ldots, \Omega_{2T}\}$

3: $R_{\Omega_0}(M) \leftarrow w. * P_{\Omega_0}(M)$

4: $U^{(0)}\Sigma^{(0)}(V^{(0)})^T = SVD(R_{\Omega_0}(M), r)$ //Best rank-r approximation of $R_{\Omega_0}(M)$

5: Trim $U^{(0)}$ and let $\widehat{U}^{(0)}$ be the output (see Section 2)

6: **for** $t = 0$ to $T-1$ **do**

7: $\quad \widehat{V}^{(t+1)} = \mathrm{argmin}_V \|R^{1/2}_{\Omega_{2t+1}}(M - \widehat{U}^{(t)}V^T)\|_F^2$, for $V \in \mathbb{R}^{d\times r}$.

8: $\quad \widehat{U}^{(t+1)} = \mathrm{argmin}_U \|R^{1/2}_{\Omega_{2t+2}}(M - U(\widehat{V}^{(t+1)})^T)\|_F^2$, for $U \in \mathbb{R}^{n\times r}$.

9: **end for**

output Completed matrix $\widehat{M}_r = \widehat{U}^{(T)}(\widehat{V}^{(T)})^T$.

for any constant $\gamma > 0$, the output $\widehat{M}_r$ of Algorithm 1 with the above specified parameters m, T, satisfies:

$$\|M - \widehat{M}_r\| \le \|M - M_r\| + \epsilon \|M - M_r\|_F + \zeta.$$

That is, if $T = \log(\frac{\|M\|}{\epsilon\|M - M_r\|_F})$, we have: $\|M - \widehat{M}_r\| \le \|M - M_r\| + 2\epsilon \|M - M_r\|_F$.

2.1 Direct Low-rank Approximation of Matrix Product

In this section we present a new algorithm for the following problem: suppose we are given two matrices, and desire a low-rank approximation of their product AB; in particular, we are *not* interested in the actual full matrix product itself (as this may be unwieldy to store and use, and thus wasteful to produce in its entirety). One example setting where this arises is when one wants to calculate the joint counts between two very large sets of entities; for example, web companies routinely come across settings where they need to understand (for example) how many users both searched for a particular query and clicked on a particular advertisement. The number of possible queries and ads is huge, and finding this co-occurrence matrix from user logs involves multiplying two matrices – query-by-user and user-by-ad respectively – each of which is itself large.

Algorithm: Suppose we are given an $n_1 \times d$ matrix A and another $d \times n_2$ matrix B, and we wish to calculate a rank-r approximation of the product $A \cdot B$. Our algorithm proceeds in two stages:

1. Choose a biased random set $\Omega \subset [n_1] \times [n_2]$ of elements as follows: choose an intended number m (according to Theorem 2.2 below) of sampled elements, and then independently include each $(i, j) \in [n_1] \times [n_2]$ in Ω with probability given by $\hat{q}_{ij} = \min\{1, q_{ij}\}$ where

$$q_{ij} \ := \ m \cdot \left(\frac{\|A^i\|^2}{n_2\|A\|_F^2} + \frac{\|B_j\|^2}{n_1\|B\|_F^2} \right), \tag{3}$$

   Then, find $P_\Omega(A \cdot B)$, i.e. only the elements of the product AB that are in this set Ω.

2. Run the alternating minimization procedure WAltMin$(P_\Omega(A \cdot B), \Omega, r, \hat{q}, T)$, where T is the number of iterations (again chosen according to Theorem 2.2 below). This produces the low-rank approximation in factored form.

The computation complexity of the algorithm is $O(|\Omega| \cdot (d + r^2)) = O(m(d + r^2)) = O(\frac{nr^3\kappa^2}{\epsilon^2} \cdot (d + r^2))$ (suppressing terms dependent on norms of A and B), where $n = \max\{n_1, n_2\}$.

Theorem 2.2. *Consider matrices $A \in \mathbb{R}^{n_1 \times d}$ and $B \in \mathbb{R}^{d \times n_2}$ and let $m = \frac{C}{\gamma} \cdot \frac{(\|A\|_F^2 + \|B\|_F^2)^2}{\|AB\|_F^2} \cdot \frac{nr^3}{(\epsilon)^2}\kappa^2 \log(n) \log^2(\frac{\|A\|_F + \|B\|_F}{\zeta})$, where $\kappa = \sigma_1^*/\sigma_r^*$, σ_i^* is the i-th singular value of $A \cdot B$ and $T = \log(\frac{\|A\|_F + \|B\|_F}{\zeta})$. Let Ω be sampled using probability distribution* (3)*. Then, the output $\widehat{AB}_r = WAltMin(P_\Omega(A \cdot B), \Omega, r, \hat{q}, T)$ of Sub-routine 2 satisfies (w.p. $\geq 1 - \gamma$):* $\|A \cdot B - \widehat{AB}_r\| \leq \|A \cdot B - (A \cdot B)_r\| + \epsilon\|A \cdot B - (A \cdot B)_r\|_F + \zeta$.

3 Simulations

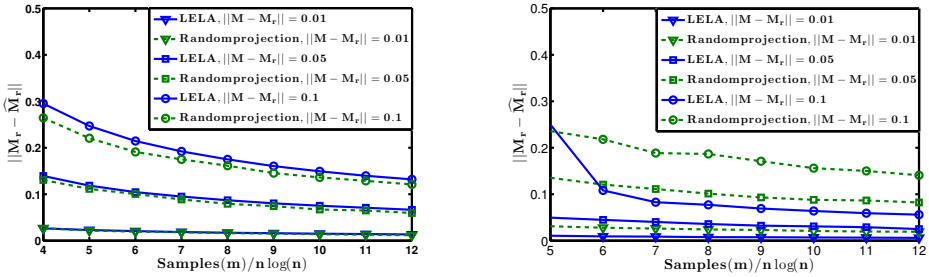

Figure 1: Figure plots how the error $||M_r - \widehat{M}_r||$ decreases with increasing number of samples m for different values of noise $||M - M_r||$, for incoherent and coherent matrices respectively. Algorithm LELA 1 is run with m samples and Gaussian projections algorithms is run with corresponding dimension of the projection $l = m/n$. Computationally LELA algorithms takes $O(nnz(M) + m\log(n))$ time for computing samples and Gaussian projections algorithm takes $O(nm)$ time. **(a)**:For same number of samples both algorithms have almost the same error for incoherent matrices. **(b):** For coherent matrices clearly the error of LELA algorithm (solid lines) is much smaller than that of random projections (dotted lines).

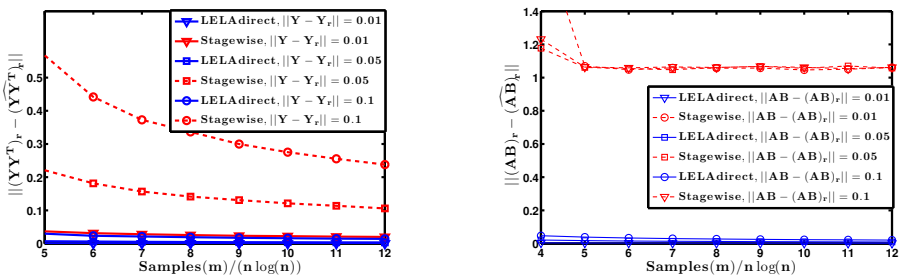

Figure 2: **(a)**:Figure plots the error $||(YY^T)_r - \widehat{(YY^T)}_r||$ for LELA direct 2.1 and Stagewise algorithm for incoherent matrices. Stagewise algorithm is first computing rank-r approximation $\widehat{Y}_r$ of Y using algorithm 1 and setting the low rank approximation $\widehat{(YY^T)}_r = \widehat{Y}_r\widehat{Y}_r^T$. Clearly directly computing low rank approximation of YY^T has smaller error. **(b):**Error $||(AB)_r - \widehat{(AB)}_r||$ for LELA direct 2.1 and Stagewise algorithm.

In this section we present some simulation results on synthetic data to show the error performance of the algorithm 1. We consider random matrices of size 1000 by 1000 and rank-5. M_r is a rank 5 matrix with all singular values 1. We consider two cases one in which M_r is incoherent and other in which M_r is coherent. The input to algorithms is the matrix $M = M_r + Z$, where Z is a Gaussian noise matrix with $||Z|| = 0.01, 0.05$ and 0.1.

References

[1] C. Boutsidis and A. Gittens. Improved matrix algorithms via the subsampled randomized hadamard transform. *SIAM Journal on Matrix Analysis and Applications*, 34(3):1301–1340, 2013.

[2] Y. Chen, S. Bhojanapalli, S. Sanghavi, and R. Ward. Coherent matrix completion. In *Proceedings of The 31st International Conference on Machine Learning*, pages 674–682, 2014.

[3] K. L. Clarkson and D. P. Woodruff. Low rank approximation and regression in input sparsity time. In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*, pages 81–90. ACM, 2013.

[4] N. H. Nguyen, T. T. Do, and T. D. Tran. A fast and efficient algorithm for low-rank approximation of a matrix. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 215–224. ACM, 2009.

[5] T. Sarlos. Improved approximation algorithms for large matrices via random projections. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 143–152. IEEE, 2006.

[6] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert. A fast randomized algorithm for the approximation of matrices. *Applied and Computational Harmonic Analysis*, 25(3):335–366, 2008.