An Efficient, Sparsity-Preserving, Online Algorithm for Low-Rank Approximation

David Anderson [* 1] **Ming Gu** [* 1]

Abstract

Low-rank matrix approximation is a fundamental tool in data analysis for processing large datasets, reducing noise, and finding important signals. In this work, we present a novel truncated LU factorization called **Spectrum-Revealing LU** (SRLU) for effective low-rank matrix approximation, and develop a fast algorithm to compute an SRLU factorization. We provide both matrix and singular value approximation error bounds for the SRLU approximation computed by our algorithm. Our analysis suggests that SRLU is competitive with the best low-rank matrix approximation methods, deterministic or randomized, in both computational complexity and approximation quality. Numeric experiments illustrate that SRLU preserves sparsity, highlights important data features and variables, can be efficiently updated, and calculates data approximations nearly as accurately as possible. To the best of our knowledge this is the first practical variant of the LU factorization for effective and efficient low-rank matrix approximation.

1. Introduction

Low-rank approximation is an essential data processing technique for understanding large or noisy data in diverse areas including data compression, image and pattern recognition, signal processing, compressed sensing, latent semantic indexing, anomaly detection, and recommendation systems. Recent machine learning applications include training neural networks (Jaderberg et al., 2014; Kirkpatrick et al., 2017), second order online learning (Luo et al., 2016), representation learning (Wang et al., 2016), and reinforcement learning (Ghavamzadeh et al., 2010). Additionally, a recent trend in machine learning is to include an approximation of second order information for better accuracy and faster convergence (Krummenacher et al., 2016).

In this work, we introduce a novel low-rank approximation algorithm called Spectrum-Revealing LU (SRLU) that can be efficiently computed and updated. Furthermore, SRLU preserves sparsity and can identify important data variables and observations. Our algorithm works on any data matrix, and achieves an approximation accuracy that only differs from the accuracy of the best approximation possible for any given rank by a constant factor.[1]

The major innovation in SRLU is the efficient calculation of a truncated LU factorization of the form

$$
\begin{aligned}
\Pi_1 \mathbf{A} \Pi_2^T &= \begin{array}{c} \\ k \\ m-k \end{array} \begin{array}{cc} k & m-k \\ \multicolumn{2}{c}{\begin{pmatrix} \mathbf{L}_{11} & \\ \mathbf{L}_{21} & \mathbf{I}_{n-k} \end{pmatrix}} \end{array} \begin{array}{cc} k & n-k \\ \multicolumn{2}{c}{\begin{pmatrix} \mathbf{U}_{11} & \mathbf{U}_{12} \\ & \mathbf{S} \end{pmatrix}} \end{array} \\
&\approx \begin{pmatrix} \mathbf{L}_{11} \\ \mathbf{L}_{21} \end{pmatrix} \begin{pmatrix} \mathbf{U}_{11} & \mathbf{U}_{12} \end{pmatrix} \\
&\overset{\text{def}}{=} \widehat{\mathbf{L}}\widehat{\mathbf{U}},
\end{aligned}
$$

where Π_1 and Π_2 are judiciously chosen permutation matrices. The LU factorization is unstable, and in practice is implemented by pivoting (interchanging) rows during factorization, i.e. choosing permutation matrix Π_1. For the truncated LU factorization to have any significance, nevertheless, complete pivoting (interchanging rows and columns) is necessary to guarantee that the factors $\widehat{\mathbf{L}}$ and $\widehat{\mathbf{U}}$ are well-defined and that their product accurately represents the original data. Previously, complete pivoting was impractical as a matrix factorization technique because it requires accessing the entire data matrix at every iteration, but SRLU efficiently achieves complete pivoting through randomization and includes a deterministic follow-up procedure to ensure a hight quality low-rank matrix approximation, as supported by rigorous theory and numeric experiments.

[*] Equal contribution [1] University of California, Berkeley. Correspondence to: David Anderson <davidanderson@berkeley.edu>, Ming Gu <mgu@berkeley.edu>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

[1] The truncated SVD is known to provide the best low-rank matrix approximation, but it is rarely used for large scale practical data analysis. See a brief discussion of the SVD in supplemental material.

1.1. Background on the LU factorization

Algorithm 1 presents a basic implementation of the LU factorization, where the result is stored in place such that the upper triangular part of $\mathbf{A}$ becomes $\mathbf{U}$ and the strictly lower triangular part becomes the strictly lower part of $\mathbf{L}$, with the diagonal of $\mathbf{L}$ implicitly known to contain all ones. LU with partial pivoting finds the largest entry in the i^{th} column from row i to m and pivots the row with that entry to the i^{th} row. LU with complete pivoting finds the largest entry in the submatrix $\mathbf{A}_{i+1:m,i+1:n}$ and pivots that entry to $\mathbf{A}_{i,i}$. It is generally known and accepted that partial pivoting is sufficient for general, real-world data matrices in the context of linear equation solving.

Algorithm 1 The LU factorization

```
1: **Inputs:** Data matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$
2: **for** $i = 1, 2, \cdots, \min(m, n)$ **do**
3:     Perform row and/or column pivots
4:     **for** $k = i + 1, \cdots, m$ **do**
5:         $\mathbf{A}_{k,i} = \mathbf{A}_{k,i} / \mathbf{A}_{i,i}$
6:     **end for**
7:     $\mathbf{A}_{i+1:m,i+1:n} \;{-}{=}\; \mathbf{A}_{i+1:m,1:i} \cdot \mathbf{A}_{1:i,i+1:n}$
8: **end for**
```

Algorithm 2 Crout LU

```
**Inputs:** Data matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, block size $b$
**for** $j = 0, b, 2b, \cdots, \min(m, n)/b - 1$ **do**
    Perform column pivots
    $\mathbf{A}_{j+1:m,j+1:j+b} -=$
        $\mathbf{A}_{j+1:m,1:j} \cdot \mathbf{A}_{1:j,j+1:j+b}$.
    Apply Algorithm 1 on $\mathbf{A}_{j+1:m,j+1:j+b}$
    Apply the row pivots to other columns of $\mathbf{A}$
    $\mathbf{A}_{j+1:j+b,j+b+1:n} \;{-}{=}$
        $\mathbf{A}_{j+1:j+b,1:j} \cdot \mathbf{A}_{1:j,j+b+1:n}$
**end for**
```

Line 7 of Algorithm 1 is known as the Schur update. Given a sparse input, this is the only step of the LU factorization that causes fill. As the algorithm progresses, fill will compound and may become dense, but the LU factorization, and truncated LU in particular, generally preserves some, if not most, of the sparsity of a sparse input. A numeric illustration is presented below.

There are many variations of the LU factorization. In Algorithm 2 the Crout version of LU is presented in block form. The column pivoting entails selecting the next b columns so that the in-place LU step is performed on a non-singular matrix (provided the remaining entries are not all zero). Note that the matrix multiplication steps are the bottleneck of this algorithm, requiring $O(mnb)$ operations each in general.

The LU factorization has been studied extensively since long before the invention of computers, with notable results from many mathematicians, including Gauss, Turing, and Wilkinson. Current research on LU factorizations includes communication-avoiding implementations, such as tournament pivoting (Khabou et al., 2013), sparse implementations (Grigori et al., 2007), and new computation of preconditioners (Chow & Patel, 2015). A randomized approach to efficiently compute the LU factorization with complete pivoting recently appeared in (Melgaard & Gu, 2015). These results are all in the context of linear equation solving, either directly or indirectly through an incomplete factorization used to precondition an iterative method. This work repurposes the LU factorization to create a novel efficient and effective low-rank approximation algorithm using modern randomization technology.

2. Previous Work

2.1. Low-Rank Matrix Approximation (LRMA)

Previous work on low-rank data approximation includes the Interpolative Decomposition (ID) (Cheng et al., 2005), the truncated QR with column pivoting factorization (Gu & Eisenstat, 1996), and other deterministic column selection algorithms, such as in (Batson et al., 2012).

Randomized algorithms have grown in popularity in recent years because of their ability to efficiently process large data matrices and because they can be supported with rigorous theory. Randomized low-rank approximation algorithms generally fall into one of two categories: sampling algorithms and black box algorithms. Sampling algorithms form data approximations from a random selection of rows and/or columns of the data. Examples include (Deshpande et al., 2006; Deshpande & Vempala, 2006; Frieze et al., 2004; Mahoney & Drineas, 2009). (Drineas et al., 2008) showed that for a given approximate rank k, a randomly drawn subset $\mathbf{C}$ of $c = O\left(k \log(k) \epsilon^{-2} \log\left(1/\delta\right)\right)$ columns of the data, a randomly drawn subset $\mathbf{R}$ of $r = O\left(c \log(c) \epsilon^{-2} \log\left(1/\delta\right)\right)$ rows of the data, and setting $\mathbf{U} = \mathbf{C}^{\dagger}\mathbf{A}\mathbf{R}^{\dagger}$, then the matrix approximation error $\|\mathbf{A} - \mathbf{CUR}\|_F$ is at most a factor of $1 + \epsilon$ from the optimal rank k approximation with probability at least $1 - \delta$. Black box algorithms typically approximate a data matrix in the form

$$\mathbf{A} \approx \mathbf{Q}^T \mathbf{Q} \mathbf{A},$$

where $\mathbf{Q}$ is an orthonormal basis of the random projection (usually using SVD, QR, or ID). The result of (Johnson & Lindenstrauss, 1984) provided the theoretical groundwork for these algorithms, which have been extensively studied (Clarkson & Woodruff, 2012; Halko et al., 2011; Martinsson et al., 2006; Papadimitriou et al., 2000; Sarlos, 2006; Woolfe et al., 2008; Liberty et al., 2007; Gu, 2015). Note that the projection of an m-by-n data matrix is of size m-by-ℓ, for some oversampling parameter $\ell \geq k$, and k is the target rank. Thus the computational challenge is the orthogonalization of the projection (the random projection can be applied quickly, as described in these works). A previous result on randomized LU factorizations for low-rank approximation was presented in (Aizenbud et al., 2016), but is uncompetitive in terms of theoretical results and computational performance with the work presented here.

For both sampling and black box algorithms the tuning parameter ϵ cannot be arbitrarily small, as the methods become meaningless if the number of rows and columns sampled (in the case of sampling algorithms) or the size of the random projection (in the case of black box algorithms) surpasses the size of the data. A common practice is $\epsilon \approx \frac{1}{2}$.

2.2. Guaranteeing Quality

Rank-revealing algorithms (Chan, 1987) are LRMA algorithms that guarantee the approximation is of high quality by also capturing the rank of the data within a tolerance (see supplementary materials for definitions). These methods, nevertheless, attempt to build an important submatrix of the data, and do not directly compute a low-rank approximation. Furthermore, they do not attempt to capture all positive singular values of the data. (Miranian & Gu, 2003) introduced a new type of high-quality LRMA algorithms that can capture all singular values of a data matrix within a tolerance, but requires extra computation to bound approximations of the left and right null spaces of the data matrix. Rank-revealing algorithms in general are designed around a definition that is not specifically appropriate for LRMA.

A key advancement of this work is a new definition of high quality low-rank approximation:

Definition 1 *A rank-k truncated LU factorization is **spectrum-revealing** if*

$$\left\| \mathbf{A} - \widehat{\mathbf{L}}\widehat{\mathbf{U}} \right\|_2 \quad \leq \quad q_1(k,m,n)\sigma_{k+1}\left(\mathbf{A}\right)$$

and

$$\sigma_j\left(\widehat{\mathbf{L}}\widehat{\mathbf{U}}\right) \quad \geq \quad \frac{\sigma_j\left(\mathbf{A}\right)}{q_2(k,m,n)}$$

for $1 \leq j \leq k$ and $q_1(k,m,n)$ and $q_2(k,m,n)$ are bounded by a low degree polynomial in k, m, and n.

Definition 1 has precisely what we desire in an LRMA, and no additional requirements. The constants, $q_1(k,m,n)$ and $q_2(k,m,n)$ are at least 1 for any rank-k approximation by (Eckart & Young, 1936). This work shows theoretically and numerically that our algorithm, SRLU, is spectrum-revealing in that it always finds such q_1 and q_2, often with $q_1, q_2 = O(1)$ in practice.

Algorithm 3 TRLUCP

```
Inputs: Data matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, target rank $k$, block size $b$, oversampling parameter $p \geq b$, random Gaussian matrix $\Omega \in \mathbb{R}^{p \times m}$, $\widehat{\mathbf{L}}$ and $\widehat{\mathbf{U}}$ are initially 0 matrices
Calculate random projection $\mathbf{R} = \Omega\mathbf{A}$
for $j = 0, b, 2b, \cdots, k-b$ do
    Perform column selection algorithm on $\mathbf{R}$ and swap columns of $\mathbf{A}$
    Update block column of $\widehat{\mathbf{L}}$
    Perform block LU with partial row pivoting and swap rows of $\mathbf{A}$
    Update block row of $\widehat{\mathbf{U}}$
    Update $\mathbf{R}$
end for
```

2.3. Low-Rank and Other Approximations in Machine Learning

Low-rank and other approximation algorithms have appeared recently in a variety of machine learning applications. In (Krummenacher et al., 2016), randomized low-rank approximation is applied directly to the adaptive optimization algorithm AdaGrad to incorporate variable dependence during optimization to approximate the full matrix version of AdaGrad with a significantly reduced computational complexity. In (Kirkpatrick et al., 2017), a diagonal approximation of the posterior distribution of previous data is utilized to alleviate catastrophic forgetting.

3. Main Contribution: Spectrum-Revealing LU (SRLU)

Our algorithm for computing SRLU is composed of two subroutines: partially factoring the data matrix with randomized complete pivoting (TRLUCP) and performing swaps to improve the quality of the approximation (SRP). The first provides an efficient algorithm for computing a truncated LU factorization, whereas the second ensures the resulting approximation is provably reliable.

3.1. Truncated Randomized LU with Complete Pivoting (TRLUCP)

Intuitively, TRLUCP performs deterministic LU with partial row pivoting for some initial data with permuted columns. TRLUCP uses a random projection of the Schur complement to cheaply find and move forward columns that are more likely to be representative of the data. To accomplish this, Algorithm 3 performs an iteration of block LU factorization in a careful order that resembles Crout LU reduction. The ordering is reasoned as follows: LU with partial row pivoting cannot be performed until the needed columns are selected, and so column selection must first occur at each iteration. Once a block column is selected, a partial Schur update must be performed on that block column before proceeding. At this point, an iteration of block LU with partial row pivoting can be performed on the current block. Once the row pivoting is performed, a partial Schur update of the block of pivoted rows of $\mathbf{U}$ can be performed, which completes the factorization up to rank $j+b$. Finally, the projection matrix $\mathbf{R}$ can be cheaply updated to prepare for the next iteration. Note that any column selection method may be used when picking column pivots from $\mathbf{R}$, such as QR with column pivoting, LU with row pivoting, or even this algorithm can again be run on the subproblem of column selection of $\mathbf{R}$. The flop count of TRLUCP is dominated by the three matrix multiplication steps (lines 2, 5, and 7). The total number of flops is

$$F^{\text{TRLUCP}} = 2pmn + (m+n)k^2 + O\left(k(m+n)\right).$$

Note the transparent constants, and, because matrix multiplication is the bottleneck, this algorithm can be implemented efficiently in terms of both computation as well as memory usage. Because the output of TRLUCP is only written once, the total number of memory writes is $(m+n-k)k$. Minimizing the number of data writes by only writing data once significantly improves efficiency because writing data is typically one of the slowest computational operations. Also worth consideration is the simplicity of the LU decomposition, which only involves three types of operations: matrix multiply, scaling, and pivoting. By contrast, state-of-the-art calculation of both the full and truncated SVD requires a more complex process of bidiagonalization. The projection $\mathbf{R}$ can be updated efficiently to become a random projection of the Schur complement for the next iteration. This calculation involves the current progress of the LU factorization and the random matrix Ω, and is described in detail in the appendix.

3.2. Spectrum-Revealing Pivoting (SRP)

TRLUCP produces high-quality data approximations for almost all data matrices, despite the lack of theoretical guarantees, but can miss important rows or columns of the data. Next, we develop an efficient variant of the existing rank-revealing LU algorithms (Gu & Eisenstat, 1996; Miranian & Gu, 2003) to rapidly detect and, if necessary, correct any possible matrix approximation failures of TRLUCP.

Intuitively, the quality of the factorization can be tested by searching for the next choice of pivot in the Schur complement if the factorization continued and determining if the addition of that element would significantly improve the approximation quality. If so, then the row and column with this element should be included in the approximation and another row and column should be excluded to maintain rank. Because TRLUCP does not provide an updated Schur complement, the largest element in the Schur complement can be approximated by finding the column of $\mathbf{R}$ with largest norm, performing a Schur update of that column, and then picking the largest element in that column. Let α be this element, and, without loss of generality, assume it is the first entry of the Schur complement. Denote:

$$\Pi_1 \mathbf{A} \Pi_2^T = \begin{pmatrix} \mathbf{L}_{11} & & \\ \ell^T & 1 & \\ \mathbf{L}_{31} & & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{U}_{11} & u & \mathbf{U}_{13} \\ & \alpha & s_{12}^T \\ & s_{21} & \mathbf{S}_{22} \end{pmatrix}. \quad (1)$$

Next, we must find the row and column that should be replaced if the row and column containing α are important. Note that the smallest entry of $\mathbf{L}_{11}\mathbf{U}_{11}$ may still lie in an important row and column, and so the largest element of the inverse should be examined instead. Thus we propose defining

$$\overline{\mathbf{A}}_{11} \stackrel{\text{def}}{=} \begin{pmatrix} \mathbf{L}_{11} & \\ \ell^T & 1 \end{pmatrix} \begin{pmatrix} \mathbf{U}_{11} & u \\ & \alpha \end{pmatrix}$$

and testing

$$\|\overline{\mathbf{A}}_{11}^{-1}\|_{\max} \leq \frac{f}{|\alpha|} \quad (2)$$

for a tolerance parameter $f > 1$ that provides a control of accuracy versus the number of swaps needed. Should the test fail, the row and column containing α are swapped with the row and column containing the largest element in $\overline{\mathbf{A}}_{11}^{-1}$. Note that this element may occur in the last row or last column of $\overline{\mathbf{A}}_{11}^{-1}$, indicating only a column swap or row swap respectively is needed. When the swaps are performed, the factorization must be updated to maintain truncated LU form. We have developed a variant of the LU updating algorithm of (Gondzio, 2007) to efficiently update the SRLU factorization.

SRP can be implemented efficiently: each swap requires at most $O\left(k(m+n)\right)$ operations, and $\|\overline{\mathbf{A}}_{11}^{-1}\|_{\max}$ can be quickly and reliably estimated using (Higham & Relton, 2015). An argument similar to that used in (Miranian & Gu, 2003) shows that each swap will increase $\left|\mathbf{det}\left(\overline{\mathbf{A}}_{11}\right)\right|$ by a factor at least f, hence will never repeat. At termination, SRP will ensure a partial LU factorization of the form (1) that satisfies condition (2). We will discuss spectrum-revealing properties of this factorization in Section 4.2.

It is possible to derive theoretical upper bounds on the worst number of swaps necessary in SRP, but in practice, this number is zero for most matrices, and does not exceed $3-5$ in the most pathological data matrix of dimension at most 1000 we can contrive.

SRLU can be used effectively to approximate second order information in machine learning. SRLU can be used as a modification to AdaGrad in a manner similar to the

Algorithm 4 Spectrum-Revealing Pivoting (SRP)

```
Input: Truncated LU factorization $\mathbf{A} \approx \widehat{\mathbf{L}}\widehat{\mathbf{U}}$, tolerance $f > 1$
while $\|\overline{\mathbf{A}}_{11}^{-1}\|_{\max} > \frac{f}{|\alpha|}$ do
    Set $\alpha$ to be the largest element in $\mathbf{S}$ (or find an approximate $\alpha$ using $\mathbf{R}$)
    Swap row and column containing $\alpha$ with row and column of largest element in $\overline{\mathbf{A}}_{11}^{-1}$
    Update truncated LU factorization
end while
```

low-rank approximation method in (Krummenacher et al., 2016). Applying the initialization technique in this work, SRLU would likely provide an efficient and accurate adaptive stochastic optimization algorithm. SRLU can also become a full-rank approximation (low-rank plus diagonal) by adding a diagonal approximation of the Schur complement. Such an approximation could be appropriate for improving memory in artificial intelligence, such as in (Kirkpatrick et al., 2017). SRLU is also a freestanding compression algorithm.

3.3. The CUR Decomposition with LU

A natural extension of truncated LU factorizations is a CUR-type decomposition for increased accuracy (Mahoney & Drineas, 2009):

$$\Pi_1 \mathbf{A} \Pi_2^T \quad \approx \quad \widehat{\mathbf{L}} \left(\widehat{\mathbf{L}}^{\dagger} \mathbf{A} \widehat{\mathbf{U}}^{\dagger} \right) \widehat{\mathbf{U}} \quad \stackrel{\text{def}}{=} \quad \widehat{\mathbf{L}} \mathbf{M} \widehat{\mathbf{U}}.$$

As with standard CUR, the factors $\widehat{\mathbf{L}}$ and $\widehat{\mathbf{U}}$ retain (much of) the sparsity of the original data, while $\mathbf{M}$ is a small, k-by-k matrix. The CUR decomposition can improve the accuracy of an SRLU with minimal extra needed memory. Extra computational time, nevertheless, is needed to calculate $\mathbf{M}$. A more efficient, approximate CUR decomposition can be obtained by replacing $\mathbf{A}$ with a high quality approximation (such as an SRLU factorization of high rank) in the calculation of $\mathbf{M}$.

3.4. The Online SRLU Factorization

Given a factored data matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and new observations $\mathbf{B}\Pi_2^T = \begin{array}{c} \begin{array}{cc} \scriptstyle k & \scriptstyle m-k \end{array} \\ \begin{pmatrix} \mathbf{B}_1 & \mathbf{B}_2 \end{pmatrix} \end{array} \in \mathbb{R}^{s \times m}$, an augmented LU decomposition takes the form

$$\begin{pmatrix} \Pi_1 \mathbf{A} \Pi_2^T \\ \mathbf{B} \Pi_2^T \end{pmatrix} = \begin{pmatrix} \mathbf{L}_{11} & & \\ \mathbf{L}_{21} & \mathbf{I} & \\ \mathbf{L}_{31} & & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{U}_{11} & \mathbf{U}_{12} \\ & \mathbf{S} \\ & \mathbf{S}^{\text{new}} \end{pmatrix},$$

where $\mathbf{L}_{31} = \mathbf{B}_1 \mathbf{U}_{11}^{-1}$ and $\mathbf{S}^{\text{new}} = \mathbf{B}_2 - \mathbf{B}_1 \mathbf{U}_{11}^{-1} \mathbf{U}_{12}$. An SRLU factorization can then be obtained by simply performing correcting swaps. For a rank-1 update, at most 1 swap is expected (although examples can be constructed that require more than one swap), which requires at most $O\left(k\left(m+n\right)\right)$ flops. By contrast, the URV decomposition of (Stewart, 1992) is $O\left(n^2\right)$, while SVD updating requires $O\left(\left(m+n\right)\min^2\left(m,n\right)\right)$ operations in general, or $O\left(\left(m+n\right)\min\left(m,n\right)\log_2^2 \epsilon\right)$ for a numerical approximation with the fast multipole method.

4. Theoretical Results for SRLU Factorizations

4.1. Analysis of General Truncated LU Decompositions

Theorem 1 *Let $\left(\cdot\right)_s$ denote the rank-s truncated SVD for $s \leq k \ll m, n$. Then for any truncated LU factorization with Schur complement $\mathbf{S}$:*

$$\|\Pi_1 \mathbf{A} \Pi_2^T - \widehat{\mathbf{L}}\widehat{\mathbf{U}}\| \quad = \quad \|\mathbf{S}\|$$

for any norm, and

$$\|\Pi_1 \mathbf{A} \Pi_2^T - \left(\widehat{\mathbf{L}}\widehat{\mathbf{U}}\right)_s \|_2 \quad \leq \quad 2\|\mathbf{S}\|_2 + \sigma_{s+1}\left(\mathbf{A}\right).$$

Theorem 2 *For a general rank-k truncated LU decomposition, we have for all $1 \leq j \leq k$,*

$$\sigma_j\left(\mathbf{A}\right) \leq \sigma_j\left(\widehat{\mathbf{L}}\widehat{\mathbf{U}}\right) \left(1 + \left(1 + \frac{\|\mathbf{S}\|_2}{\sigma_k\left(\widehat{\mathbf{L}}\widehat{\mathbf{U}}\right)}\right) \frac{\|\mathbf{S}\|_2}{\sigma_j\left(\mathbf{A}\right)}\right).$$

Theorem 3 *CUR Error Bounds.*

$$\|\Pi_1 \mathbf{A} \Pi_2^T - \widehat{\mathbf{L}}\mathbf{M}\widehat{\mathbf{U}}\|_2 \quad \leq \quad 2\|\mathbf{S}\|_2$$

and

$$\|\Pi_1 \mathbf{A} \Pi_2^T - \widehat{\mathbf{L}}\mathbf{M}\widehat{\mathbf{U}}\|_F \quad \leq \quad \|\mathbf{S}\|_F.$$

Theorem 1 simply concludes that the approximation is accurate if the Schur complement is small, but the singular value bounds of Theorem 2 are needed to guarantee that the approximation retains structural properties of the original data, such as an accurate approximation of the rank and the spectrum. Furthermore, singular values bounds can be significantly stronger than the more familiar norm error bounds that appear in Theorem 1. Theorem 2 provides a general framework for singular value bounds, and bounding the terms in this theorem provided guidance in the design and development of SRLU. Just as in the case of deterministic LU with complete pivoting, the sizes of $\frac{\|\mathbf{S}\|_2}{\sigma_k\left(\widehat{\mathbf{L}}\widehat{\mathbf{U}}\right)}$ and $\frac{\|\mathbf{S}\|_2}{\sigma_j\left(\widehat{\mathbf{L}}\widehat{\mathbf{U}}\right)}$ range from moderate to small for almost all data matrices of practical interest. They, nevertheless, cannot be effectively bounded for a general TRLUCP factorization, implying the need for Algorithm 4 to ensure that these terms are controlled. While the error bounds in Theorem 3 for the CUR decomposition do not improve upon the result in Theorem 1, CUR bounds for SRLU specifically will be considerably stronger. Next, we present our main theoretical contributions.

4.2. Analysis of the Spectrum-Revealing LU Decomposition

Theorem 4 *(SRLU Error Bounds.) For $j \le k$ and $\gamma = O\left(fk\sqrt{mn}\right)$, SRP produces a rank-$k$ SRLU factorization with*

$$\|\Pi_1 \mathbf{A}\Pi_2^T - \widehat{\mathbf{L}}\widehat{\mathbf{U}}\|_2 \le \gamma\sigma_{k+1}\left(\mathbf{A}\right),$$
$$\|\Pi_1 \mathbf{A}\Pi_2^T - \left(\widehat{\mathbf{L}}\widehat{\mathbf{U}}\right)_j\|_2 \le \sigma_{j+1}\left(\mathbf{A}\right)\left(1+2\gamma\frac{\sigma_{k+1}(\mathbf{A})}{\sigma_{j+1}(\mathbf{A})}\right)$$

Theorem 4 is a special case of Theorem 1 for SRLU factorizations. For a data matrix with a rapidly decaying spectrum, the right-hand side of the second inequality is close to $\sigma_{j+1}\left(\mathbf{A}\right)$, a substantial improvement over the sharpness of the bounds in (Drineas et al., 2008).

Theorem 5 *(SRLU Spectral Bound). For $1 \le j \le k$, SRP produces a rank-k SRLU factorization with*

$$\frac{\sigma_j\left(\mathbf{A}\right)}{1+\tau\frac{\sigma_{k+1}(\mathbf{A})}{\sigma_j(\mathbf{A})}} \le \sigma_j\left(\widehat{\mathbf{L}}\widehat{\mathbf{U}}\right) \le \sigma_j\left(\mathbf{A}\right)\left(1+\tau\frac{\sigma_{k+1}\left(\mathbf{A}\right)}{\sigma_j\left(\mathbf{A}\right)}\right)$$

for $\tau \le O\left(mnk^2f^3\right)$.

While the worst case upper bound on τ is large, it is dimension-dependent, and j and k may be chosen so that $\frac{\sigma_{k+1}(\mathbf{A})}{\sigma_j(\mathbf{A})}$ is arbitrarily small compared to τ. In particular, if k is the numeric rank of $\mathbf{A}$, then the singular values of the approximation are numerically equal to those of the data.

These bounds are **problem-specific bounds** because their quality depends on the spectrum of the original data, rather than universal constants that appear in previous results. The benefit of these problem-specific bounds is that an approximation of data with a rapidly decaying spectrum is guaranteed to be high-quality. Furthermore, if $\sigma_{k+1}\left(\mathbf{A}\right)$ is not small compared to $\sigma_j\left(\mathbf{A}\right)$, then no high-quality low-rank approximation is possible in the 2 and Frobenius norms. Thus, in this sense, the bounds presented in Theorems 4 and 5 are optimal.

Given a high-quality rank-k truncated LU factorization, Theorem 5 ensures that a low-rank approximation of rank ℓ with $\ell < k$ of the compressed data is an accurate rank-ℓ approximation of the full data. The proof of this theorem centers on bounding the terms in Theorems 1 and 2. Experiments will show that τ is small in almost all cases.

Stronger results are achieved with the CUR version of SRLU:

Theorem 6

$$\|\Pi_1 \mathbf{A}\Pi_2^T - \widehat{\mathbf{L}}\mathbf{M}\widehat{\mathbf{U}}\|_2 \quad \le \quad 2\gamma\sigma_{k+1}\left(\mathbf{A}\right)$$

and

$$\|\Pi_1 \mathbf{A}\Pi_2^T - \widehat{\mathbf{L}}\mathbf{M}\widehat{\mathbf{U}}\|_F \quad \le \quad \omega\sigma_{k+1}\left(\mathbf{A}\right),$$

where $\gamma = O\left(fk\sqrt{mn}\right)$ is the same as in Theorem 4, and $\omega = O\left(fkmn\right)$.

Theorem 7 *If $\sigma_j^2\left(\mathbf{A}\right) > 2\|\mathbf{S}\|_2^2$ then*

$$\sigma_j\left(\mathbf{A}\right) \ge \sigma_j\left(\widehat{\mathbf{L}}\mathbf{M}\widehat{\mathbf{U}}\right) \ge \sigma_j\left(\mathbf{A}\right)\sqrt{1-2\gamma\left(\frac{\sigma_{k+1}\left(\mathbf{A}\right)}{\sigma_j\left(\mathbf{A}\right)}\right)^2}$$

for $\gamma = O\left(mnk^2f^2\right)$ and f is an input parameter controlling a tradeoff of quality vs. speed as before.

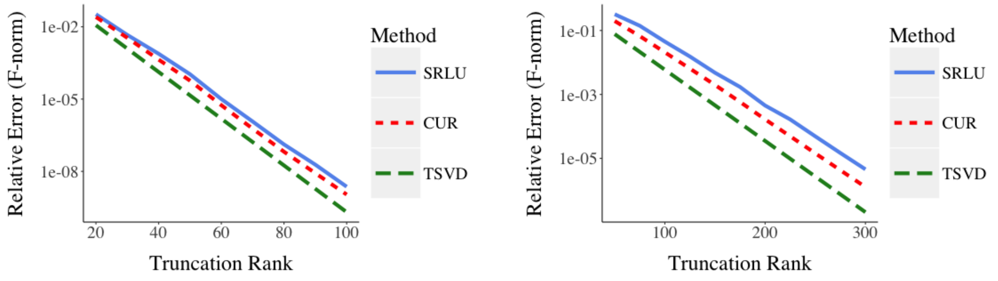
As before, the constants are small in practice. Observe that for most real data matrices, their singular values decay with increasing j. For such matrices this result is significantly stronger than Theorem 5.

5. Experiments

5.1. Speed and Accuracy Tests

In Figure 1, the accuracy of our method is compared to the accuracy of the truncated SVD. Note that SRLU did not perform any swaps in these experiments. "CUR" is the CUR version of the output of SRLU. Note that both methods exhibits a convergence rate similar to that of the truncated SVD (TSVD), and so only a constant amount of extra work is needed to achieve the same accuracy. When the singular values decay slowly, the CUR decomposition provides a greater accuracy boost. In Figure 2, the runtime of SRLU is compared to that of the truncated SVD, as well as Subspace Iteration (Gu, 2015). Note that for Subspace Iteration, we choose iteration parameter $q = 0$ and do not measure the time of applying the random projection, in acknowledgement that fast methods exist to apply a random projection to a data matrix. Also, the block size implemented in SRLU is significantly smaller than the block size used by the standard software LAPACK, as the size of the block size affects the size of the projection. See supplement for additional details. All numeric experiments were run on NERSC's Edison. For timing experiments, the truncated SVD is calculated with PROPACK.

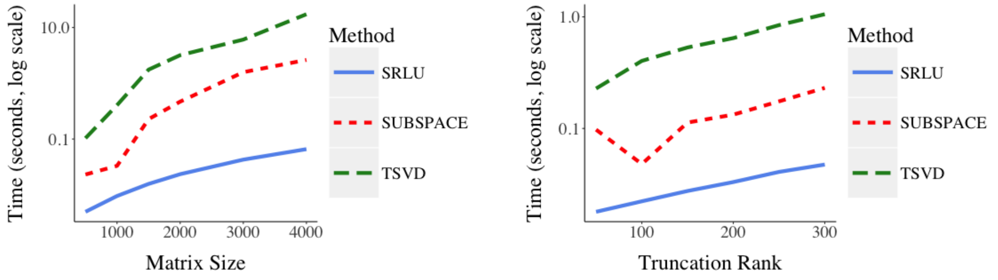
Even more impressive, the factorization stage of SRLU becomes arbitrarily faster than the standard implementation of the LU decomposition. Although the standard LU decomposition is not a low-rank approximation algorithm, it is known to be roughly 10 times faster than the SVD (Demmel, 1997). See appendix for details.



(a) Spectral Decay = 0.8

(b) Spectral Decay = 0.95

Figure 1. Accuracy experiment on random 1000x1000 matrices with different rates of spectral decay.



(a) Rank-100 Factorizations

(b) Time vs. Truncation Rank

Figure 2. Time experiment on various random matrices, and a time experiment on a 1000x1000 matrix with varying truncation ranks.

Table 1. Errors of low-rank approximations of the given target rank. SRLU is measured using the CUR version of the factorization.

DATA	k	Gaus.	SRFT	Dual BCH	SRLU
S80PI$_n$1	63	3.85	3.80	3.81	2.84
deter3	127	9.27	9.30	9.26	8.30
lc3d	63	18.39	16.36	15.49	16.94
lc3d	78				15.11

Next, we compare SRLU against competing algorithms. In (Ubaru et al., 2015), error-correcting codes are introduced to yield improved accuracy over existing random projection low-rank approximation algorithms. Their algorithm, denoted Dual BCH, is compared against SRLU as well as two other random projection methods: Gaus., which uses a Gaussian random projection, and SRFT, which uses a Fourier transform to apply a random projection. We test the spectral norm error of these algorithms on matrices from the sparse matrix collection in (Davis & Hu, 2011).

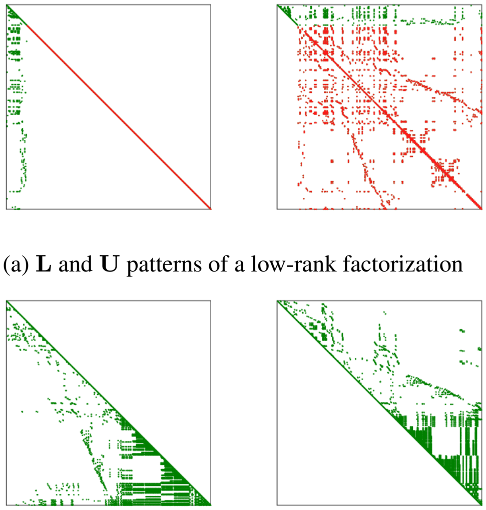
In Table 1, results for SRLU are averaged over 5 experiments. Using tuning parameter $f = 5$, no swaps were needed in all cases. The matrices being tested are sparse matrices from various engineering problems. S80PI$_n$1 is 4,028 by 4,028, deter3 is 7,647 by 21,777, and lp_ceria3d (abbreviated lc3d) is 3,576 by 4,400. Note that SRLU, a more efficient algorithm, provides a better approximation in two of the three experiments. With a little extra oversampling, a practical assumption due to the speed advantage, SRLU achieves a competitive quality approximation. The oversampling highlights an additional and unique advantage of SRLU over competing algorithms: if more accuracy is desired, then the factorization can simply continue as needed.

5.2. Sparsity Preservation Tests

The SRLU factorization is tested on sparse, unsymmetric matrices from (Davis & Hu, 2011). Figure 3 shows the sparsity patterns of the factors of an SRLU factorization of a sparse data matrix representing a circuit simulation (oscil_dcop), as well as a full LU decomposition of the data. Note that the LU decomposition preserves the sparsity of the data initially, but the full LU decomposition becomes dense. Several more experiments are shown in the supplement.

5.3. Towards Feature Selection

An image processing example is now presented to illustrate the benefit of highlighting important rows and columns selection. In Figure 4 an image is compressed to a rank-50 approximation using SRLU. Note that the rows and columns chosen overlap with the astronaut and the planet, implying that minimal storage is needed to capture the

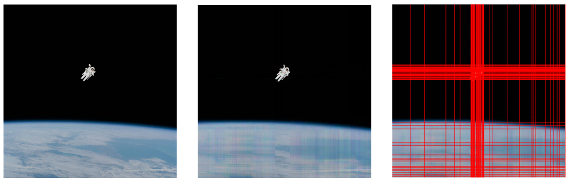


(a) **L** and **U** patterns of a low-rank factorization

(b) **L** and **U** patterns of the full factorization

Figure 3. The sparsity patterns of the L and U matrices of a rank 43 SRLU factorization, followed by the sparsity pattern of the L and U matrices of a full LU decomposition of the same data. For the SRLU factorization, the green entries compose the low-rank approximation of the data. The red entries are the additional data needed for an exact factorization.

black background, which composes approximately two thirds of the image. While this result cannot be called feature selection per se, the rows and columns selected highlight where to look for features: rows and/or columns are selected in a higher density around the astronaut, the curvature of the planet, and the storm front on the planet.



(a) Original (b) SRLU (c) Rows and Cols.

Figure 4. Image processing example. The original image (NASA), a rank-50 approximation with SRLU, and a highlight of the rows and columns selected by SRLU.

5.4. Online Data Processing

Online SRLU is tested here on the Enron email corpus (Lichman, 2013). The documents were initially reverse-sorted by the usage of the most common word, and then reverse-sorted by the second most, and this process was repeated for the five most common words (the top five words were used significantly more than any other), so that the most common words occurred most at the end of the corpus. The data contains 39,861 documents and 28,102 words/terms, and an initial SRLU factorization of rank 20 was performed on the first 30K documents. The initial factorization contained none of the top five words, but, after adding the remaining documents and updating, the top three were included in the approximation. The fourth and fifth words 'market' and 'california' have high covariance with at least two of the three top words, and so their inclusion may be redundant in a low-rank approximation.

6. Conclusion

We have presented SRLU, a low-rank approximation method with many desirable properties: efficiency, accuracy, sparsity-preservation, the ability to be updated, and the ability to highlight important data features and variables. Extensive theory and numeric experiments have illustrated the efficiency and effectiveness of this method.

Acknowledgements

This research was supported in part by NSF Award CCF-1319312.

References

Aizenbud, Y., Shabat, G., and Averbuch, A. Randomized lu decomposition using sparse projections. *CoRR*, abs/1601.04280, 2016.

Batson, J., Spielman, D. A., and Srivastava, N. Twice-ramanujan sparsifiers. *SIAM Journal on Computing*, 41 (6):1704–1721, 2012.

Chan, T. F. Rank revealing qr factorizations. *Linear algebra and its applications*, (88/89):67–82, 1987.

Cheng, H., Gimbutas, Z., Martinsson, P.-G., and Rokhlin, V. On the compression of low rank matrices. *SIAM J. Scientific Computing*, 26(4):1389–1404, 2005.

Chow, E. and Patel, A. Fine-grained parallel incomplete lu factorization. *SIAM J. Scientific Computing*, 37(2), 2015.

Clarkson, K. L. and Woodruff, D. P. Low rank approximation and regression in input sparsity time. *CoRR*, abs/1207.6365, 2012.

Davis, T. A. and Hu, Y. The university of florida sparse matrix collection. *ACM Transactions on Mathematical Software*, 38:1:1–1:25, 2011. URL `http://www.cise.ufl.edu/research/sparse/matrices`.

Demmel, J. *Applied Numerical Linear Algebra*. SIAM, 1997.

Deshpande, A. and Vempala, S. Adaptive sampling and fast low-rank matrix approximation. In *APPROX-RANDOM*, volume 4110 of *Lecture Notes in Computer Science*, pp. 292–303. Springer, 2006.

Deshpande, A., Rademacher, L., Vempala, S., and Wang, G. Matrix approximation and projective clustering via volume sampling. *Theory of Computing*, 2(12):225–247, 2006.

Drineas, P., Mahoney, M. W., and Muthukrishnan, S. Relative-error cur matrix decompositions. *SIAM J. Matrix Analysis Applications*, 30(2):844–881, 2008.

Eckart, C. and Young, G. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

Frieze, A. M., Kannan, R., and Vempala, S. Fast monte-carlo algorithms for finding low-rank approximations. *J. ACM*, 51(6):1025–1041, 2004.

Ghavamzadeh, M., Lazaric, A., Maillard, O.-A., and Munos, R. Lstd with random projections. In *NIPS*, pp. 721–729, 2010.

Gondzio, J. Stable algorithm for updating dense lu factorization after row or column exchange and row and column addition or deletion. *Optimization: A journal of Mathematical Programming and Operations Research*, pp. 7–26, 2007.

Grigori, L., Demmel, J., and Li, X. S. Parallel symbolic factorization for sparse lu with static pivoting. *SIAM J. Scientific Computing*, 29(3):1289–1314, 2007.

Gu, M. Subspace iteration randomization and singular value problems. *SIAM J. Scientific Computing*, 37(3), 2015.

Gu, M. and Eisenstat, S. C. Efficient algorithms for computing a strong rank-revealing qr factorization. *SIAM J. Sci. Comput.*, 17(4):848–869, 1996.

Halko, N., Martinsson, P.-G., and Tropp, J. A. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.

Higham, N. J. and Relton, S. D. Estimating the largest entries of a matrix. 2015. URL http://eprints.ma.man.ac.uk/.

Jaderberg, M., Vedaldi, A., and Zisserman, A. Speeding up convolutional neural networks with low rank expansions. *CoRR*, abs/1405.3866, 2014.

Johnson, W. B. and Lindenstrauss, J. Extensions of lipschitz mappings into a hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

Khabou, A., Demmel, J., Grigori, L., and Gu, M. Lu factorization with panel rank revealing pivoting and its communication avoiding version. *SIAM J. Matrix Analysis Applications*, 34(3):1401–1429, 2013.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N. C., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13): 3521–3526, 2017.

Krummenacher, G., McWilliams, B., Kilcher, Y., Buhmann, J. M., and Meinshausen, N. Scalable adaptive stochastic optimization using random projections. In *NIPS*, pp. 1750–1758, 2016.

Liberty, E., Woolfe, F., Martinsson, P.G., Rokhlin, V., and Tygert, M. Randomized algorithms for the low-rank approximation of matrices. *Proceedings of the National Academy of Sciences*, 104(51):20167, 2007.

Lichman, M. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

Luo, H., Agarwal, A., Cesa-Bianchi, N., and Langford, J. Efficient second order online learning by sketching. In *NIPS*, pp. 902–910, 2016.

Mahoney, M. W. and Drineas, P. Cur matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 106(3):697–702, 2009.

Martinsson, P.-G., Rokhlin, V., and Tygert, M. A randomized algorithm for the approximation of matrices. *Tech. Rep., Yale University, Department of Computer Science*, (1361), 2006.

Melgaard, C. and Gu, M. Gaussian elimination with randomized complete pivoting. *CoRR*, abs/1511.08528, 2015.

Miranian, L. and Gu, M. Stong rank revealing lu factorizations. *Linear Algebra and its Applications*, 367:1–16, 2003.

NASA. Nasa celebrates 50 years of spacewalking. URL https://www.nasa.gov/image-feature/nasa-celebrates-50-years-of-spacewalking. Accessed on August 22, 2015. Published on June 3, 2015. Original photograph from February 7, 1984.

Papadimitriou, C. H., Raghavan, P., Tamaki, H., and Vempala, S. Latent semantic indexing: A probabilistic analysis. *J. Comput. Syst. Sci.*, 61(2):217–235, 2000.

Sarlos, T. Improved approximation algorithms for large matrices via random projections. In *FOCS*, pp. 143–152. IEEE Computer Society, 2006.

Stewart, G. W. An updating algorithm for subspace tracking. *IEEE Trans. Signal Processing*, 40(6):1535–1541, 1992.

Ubaru, S., Mazumdar, A., and Saad, Y. Low rank approximation using error correcting coding matrices. In *ICML*, volume 37, pp. 702–710, 2015.

Wang, W. Y., Mehdad, Y., Radev, D. R., and Stent, A. A low-rank approximation approach to learning joint embeddings of news stories and images for timeline summarization. pp. 58–68, 2016.

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