Block CUR : Decomposing Large Distributed Matrices

Article · March 2017

CITATION
1

READS
45

4 authors, including:

Urvashi Oswal
University of Wisconsin–Madison
5 PUBLICATIONS **6** CITATIONS
SEE PROFILE

Swayambhoo Jain
University of Minnesota Twin Cities
21 PUBLICATIONS **83** CITATIONS
SEE PROFILE

Block CUR : Decomposing Large Distributed Matrices

Urvashi Oswal, Swayambhoo Jain, Kevin S. Xu, and Brian Eriksson

Abstract

A common problem in large-scale data analysis is to approximate a matrix using a combination of specifically sampled rows and columns, known as CUR decomposition. Unfortunately, in many real-world environments, the ability to sample specific individual rows or columns of the matrix is limited by either system constraints or cost. In this paper, we consider matrix approximation by sampling predefined *blocks* of columns (or rows) from the matrix. This regime is commonly found when data is distributed across multiple nodes in a compute cluster, where such blocks correspond to columns (or rows) of the matrix stored on the same node, which can be retrieved with much less overhead than retrieving individual columns stored across different nodes. We propose a novel algorithm for sampling useful column blocks and provide guarantees for the quality of the approximation. We demonstrate the practical utility of this algorithm for computing the block CUR decomposition of large matrices in a distributed setting using Apache Spark. Using our proposed block CUR algorithms, we can achieve a significant speed-up compared to a regular CUR decomposition with the same quality of approximation.

I. Introduction

The ability to perform large-scale data analysis is often limited by two opposing forces. The first force is the need to store data in a matrix format for the purpose of analysis techniques such as regression or classification. The second force is the inability to store the data matrix completely in memory due to the size of the matrix in many application settings. This conflict gives rise to storing factorized matrix forms, such as SVD or CUR decompositions [1].

We consider a matrix $\boldsymbol{A}$ with m rows and n columns, *i.e.,* $\boldsymbol{A} \in \mathbb{R}^{m \times n}$. Using a truncated k number of singular vectors (*e.g.,* where $k < \min\{m, n\}$), the singular value decomposition (SVD) provides the best rank-k approximation to the original matrix. The singular vectors of a SVD often do not preserve the structure in original data. Preserving the original structure in the data may be desirable due to many reasons including interpret-ability in case of biometric data or for storage efficiency in case of sparse matrices. This has led to the introduction of the CUR decomposition, where the factorization is performed with respect to a subset of the rows and columns of the matrix itself. This specific decomposition describes the matrix $\boldsymbol{A}$ as the product of a subset of the matrix rows $\boldsymbol{R}$ and a subset of the matrix columns $\boldsymbol{C}$ (along with a matrix $\boldsymbol{U}$ that fits to $\boldsymbol{A}$).

UO is with the Department of Electrical and Computer Engineering, University of Wisconsin – Madison, SJ is with the Department of Electrical and Computer Engineering, University of Minnesota – Twin Cities, KSX is with the Electrical Engineering and Computer Science Department, University of Toledo, BE is with Technicolor Research – Los Altos. Author emails: uoswal@wisc.edu, jainx174@umn.edu, kevin.xu@utoledo.edu, brian.eriksson@technicolor.com

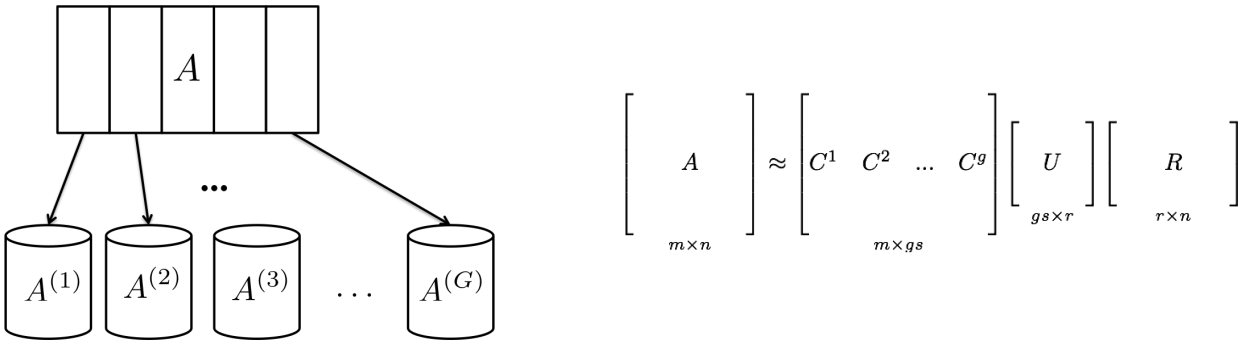

Fig. 1: (Left) Distributed storage of a large matrix across multiple nodes in a cluster. Blocks of columns are allocated to each of the G nodes. (Right) Example block CUR decomposition, where $\boldsymbol{C}^t \in \mathbb{R}^{m \times s}$ for $t \in [g]$ is sampled from $\{\boldsymbol{A}^{(j_t)} : j_t \in [G]\}$.

Significant prior work has examined how to efficiently choose the rows and columns in the CUR decomposition and has derived worst-case error bounds (*e.g.,* [2]). These methods have been applied successfully to many real-world problems including genetics [3], astronomy [4], and mass spectrometry imaging [5]. Unfortunately, these large-scale datasets require distributed storage, a regime where there can be substantial overhead involved in querying individual rows or columns of a matrix. In these regimes, it is more efficient to retrieve predefined *blocks* of rows or columns at one time corresponding to the rows or columns stored on the same node, as shown in Figure 1(a), in order to minimize the overhead. In doing so, one forms a block CUR decomposition, as shown in Figure 1(b). Current CUR decomposition techniques do not take advantage of this predefined block structure. Another setting where block CUR decompositions are useful involves processing data where the ordering of rows or columns is meaningful, such as images, video, or speech data matrices. In this setting, sampling contiguous blocks of columns is necessary for interpretability of the factorized representation.

Using these insights into real-world applications of CUR decomposition, this paper makes a series of contributions. We propose a randomized Block CUR algorithm for subset selection of rows and blocks of columns and derive worst-case error bounds for this randomized algorithm. This algorithm performs fast block sampling taking advantage of the natural storage of matrices in distributed environments. Additionally, this algorithm does not require a full SVD calculation, saving significant computation time in the large-scale setting. Finally, we present worst-case error bounds for this algorithm using theoretical results related to approximating matrix multiplication and generalized ℓ_2-regression in this sampling blocks setting.

Additionally, we demonstrate empirically that the proposed block CUR algorithms can achieve a significant speed-up when used to decompose large matrices in a distributed data setting. We conduct a series of CUR decomposition experiments in Amazon AWS using both synthetic and real-world data. In this distributed environment, we find that our Block CUR approach achieves a speed-up of 2x to 6x for matrices larger than 12000×12000. This is compared with previous CUR approaches that sample individual rows and columns and while achieving the same matrix approximation error rate.

The rest of the paper is organized as follows. The CUR decomposition model is introduced in Section II. We detail the specific steps of our block CUR algorithm and corresponding worst-case relative error bounds in Section III.

We present experiments using our proposed block CUR algorithms in a distributed setting using Apache Spark in Section IV. Finally, Section V introduces theoretical results used in the proofs of the relative error bounds.

II. Setup and Background

The need to factorize a matrix using a collection of rows and columns of that matrix has motivated the CUR decomposition literature. CUR decomposition is focused on sampling rows and columns of the matrix to provide a factorization that is close to the best rank-k approximation of the matrix. One of the most fundamental results for a CUR decomposition of a given matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ was obtained in [1]. This relative error bound result is summarized in the following theorem.

Theorem 1. *(Theorem 2 from [1] applied to $\boldsymbol{A}^T$) Given $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and an integer $k \leq \min\{m, n\}$, let $r = O(\frac{k^2}{\varepsilon^2} \ln(\frac{1}{\delta}))$ and $c = O(\frac{r^2}{\varepsilon^2} \ln(\frac{1}{\delta}))$. There exist randomized algorithms such that, if c columns are chosen to construct $\boldsymbol{C}$ and r rows are chosen to construct $\boldsymbol{R}$, then with probability $\geq 1 - \delta$, the following holds:*

$$\|\boldsymbol{A} - \boldsymbol{CUR}\|_F \leq (1 + \varepsilon)\|\boldsymbol{A} - \boldsymbol{A}_k\|_F$$

where $\varepsilon, \delta \in (0, 1)$, $\boldsymbol{U} = \boldsymbol{W}^\dagger$ and $\boldsymbol{W}$ is the scaled intersection of $\boldsymbol{C}$ and $\boldsymbol{R}$.

This theorem states that as long as enough rows and columns of the matrix are acquired (r and c, respectively), then the CUR decomposition will be within a constant factor of the error associated with the best rank-k approximation of that matrix.

Central to the proposed randomized algorithm was the concept of sampling columns of the matrix based on a *leverage score*. The leverage score measures the contribution of each column to the approximation of $\boldsymbol{A}$.

Definition 1. *The **leverage score** of a column is defined as the squared row norm of the top-k right singular vectors of $\boldsymbol{A}$ corresponding to the column:*

$$\ell_j = \|\boldsymbol{V}_{A,k}^T \boldsymbol{e}_j\|_2^2, \ j \in [n],$$

where $\boldsymbol{V}_{A,k}$ consists of the top-k right singular vectors of $\boldsymbol{A}$ as its rows, and $\boldsymbol{e}_j$ picks the j-th column of $\boldsymbol{V}_{A,k}^T$.

The CUR algorithm involves randomly sampling r rows using probabilities generated by the calculated leverage scores to obtain the matrix $\boldsymbol{R}$, and thereafter sampling c columns of $\boldsymbol{A}$ based on leverage scores of the $\boldsymbol{R}$ matrix to obtain $\boldsymbol{C}$.

The key technical insight in [1] is that the leverage score of a column measures "how much" of the column lies in the subspace spanned by the top-k left singular vectors of $\boldsymbol{A}$. By sampling columns that lie in this subspace more often, we get a relative-error low rank approximation of the matrix.

Unfortunately, these prior results require both the calculation of SVD of $\boldsymbol{A}$ and the sampling of arbitrary rows and columns of the matrix $\boldsymbol{A}$. Both requirements may be either unrealistic or inefficient. In this paper, we focus on the problem of efficiently sampling pre-defined blocks of columns (or rows) of the matrix to provide a factorization

that is close to the best rank-k approximation of the matrix. In the following section, we propose and analyze a randomized algorithm for sampling blocks of the matrix based on *block leverage scores*.

III. The Block CUR Algorithm

A block is defined as a collection of s columns[1]. Let there be $G = \lceil n/s \rceil$ possible blocks in $\boldsymbol{A}$. We consider the blocks to be predefined due to natural constraints or cost, such as data partitioning in a distributed compute cluster. The goal of the block CUR algorithm is to approximate the underlying matrix $\boldsymbol{A}$ using g blocks of columns and r rows, as represented in Figure 1.

Given the new regime of submatrix blocks, we must first define new block-specific variables for leverage scores, matrix incoherence, and matrix stable rank. Informed by the prior CUR literature, we begin by defining a *block leverage score* for each block of columns.

Definition 2. *The **block leverage score** of a group of columns is defined as the sum of the squared row norms of the top-k right singular vectors of $\boldsymbol{A}$ corresponding to the columns in the block:*

$$\ell_g(\boldsymbol{A}, k) = \|\boldsymbol{V}_{A,k}^T \boldsymbol{E}_g\|_F^2, \ g \in [G],$$

where $\boldsymbol{V}_{A,k}$ consists of the top-k right singular vectors of $\boldsymbol{A}$, and $\boldsymbol{E}_g$ picks the columns of $\boldsymbol{V}_{A,k}^T$ corresponding to the elements in block g.

Much like the individual column leverage scores defined in [1], the block leverage scores measure how much a particular column block contributes to the approximation of the matrix $\boldsymbol{A}$.

We next define a property of matrix rank relative to the collection of matrix blocks. Specifically, we focus on the concept of *matrix stable rank* from [6]. With respect to our matrix blocks, we define the *block stable rank* as the minimum stable rank across all matrix blocks.

Definition 3. *Let $\boldsymbol{V}_{A,k}$ consist of the top-k right singular vectors of $\boldsymbol{A}$ then the **block stable rank** is defined as*

$$\alpha_A = \min_{g \in [G]} \|\boldsymbol{V}_{A,k}^T \boldsymbol{E}_g\|_F^2 / \|\boldsymbol{V}_{A,k}^T \boldsymbol{E}_g\|_2^2.$$

where $\boldsymbol{E}_g$ picks the columns of $\boldsymbol{V}_{A,k}^T$ corresponding to the elements in block g.

Intuitively, this gives a measure of how informative the worst matrix column block is.

Finally, when we sample rows uniformly at random, we can give relative error approximation guarantees when the matrix $\boldsymbol{A}$ satisfies a notion of *column space incoherence*. This avoids pathological constructions of rows of $\boldsymbol{A}$ that cannot be sampled at random.

Definition 4. *The top-k **column space incoherence** is defined as*

$$\mu := \mu(\boldsymbol{U}_{A,k}^T) = \frac{m}{k} \max_i \|\boldsymbol{U}_{A,k}^T \boldsymbol{e}_i\|_2^2$$

[1]To keep notation simple, we will consider just sampling column blocks, although the techniques and derivations also hold for row blocks by applying the same to the transpose.

where $\boldsymbol{e}_i$ picks the i-th column of $\boldsymbol{U}_{A,k}^T$.

The column space incoherence is used to provide a guarantee for approximation without computing the SVD of the entire matrix $\boldsymbol{A}$.

A. Algorithm Details

The Block CUR Algorithm, detailed in Algorithm 1, takes as input the matrix $\boldsymbol{A}$ and returns as output an $r \times n$ matrix $\boldsymbol{R}$ consisting of a small number of rows of $\boldsymbol{A}$ and an $m \times c$ matrix $\boldsymbol{C}$ consisting of a small number of column blocks from $\boldsymbol{A}$.

Algorithm 1: Block CUR

Input : $\boldsymbol{A}$, target rank k, size of each block s, error parameter ε, positive integers r, g
Output: $\boldsymbol{C}, \boldsymbol{R}, \widehat{\boldsymbol{A}} = \boldsymbol{CUR}$

1) *Row subset selection:* Sample r rows uniformly from $\boldsymbol{A}$ according to $p_i = 1/m$ for $i \in [m]$ and compute $\boldsymbol{R} = \boldsymbol{S}_R^T \boldsymbol{A}$.
2) *Column block subset selection:* For $t = \{1, 2, \ldots, g\}$, select a block of columns $j_t \in [G]$ independently with probability $p_{j_t} = Pr[j_t = i] = \frac{\ell_i(\boldsymbol{R}, r)}{r} = \frac{\|\boldsymbol{V}_{R,r}^T \boldsymbol{E}_i\|_F^2}{r}$ for $i \in [G]$ and update $\boldsymbol{S}$, where $\boldsymbol{V}_{R,r}$ consists of the top-r right singular vectors of $\boldsymbol{R}$, and $\boldsymbol{E}_i$ picks the columns $\boldsymbol{V}_{R,r}^T$ corresponding to the elements in block i. Compute $\boldsymbol{C} = \boldsymbol{AS}$.
3) *CUR approximation*: $\widehat{\boldsymbol{A}} = \boldsymbol{CUR}$ where $\boldsymbol{U} = \boldsymbol{W}^\dagger$, and $\boldsymbol{W} = \boldsymbol{RS}$ is the scaled intersection of $\boldsymbol{R}$ and $\boldsymbol{C}$.

In Algorithm 1, $\boldsymbol{C} = \boldsymbol{AS}$, where $\boldsymbol{S} \in \mathbb{R}^{n \times gs}$ is the block scaling and sampling matrix and the (j_t, t)-th non-zero $s \times s$ block of $\boldsymbol{S}$ is defined as

$$\boldsymbol{S}_{j_t,t} = \frac{1}{\sqrt{g p_{j_t}}} \boldsymbol{I}_s$$

where $g = c/s$ is the number of blocks picked by the algorithm. An example of the sampling matrix $\boldsymbol{S}$ with blocks chosen in the order $[1, 3, 2]$ is as follows:

$$\boldsymbol{S}_{n \times gs} = \begin{bmatrix} \frac{1}{\sqrt{gp_1}} \boldsymbol{I}_s & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \frac{1}{\sqrt{gp_2}} \boldsymbol{I}_s \\ \boldsymbol{0} & \frac{1}{\sqrt{gp_3}} \boldsymbol{I}_s & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}.$$

This sampling matrix picks the blocks of columns and scales each block to compute $\boldsymbol{C} = \boldsymbol{AS}$. A similar sampling and scaling matrix $\boldsymbol{S}_R$ is defined to pick the blocks of rows and scale each block to compute $\boldsymbol{R} = \boldsymbol{S}_R^T \boldsymbol{A}$.

In addition to considering block sampling of columns, another key advantage of this algorithm is not requiring the computation of a full SVD of $\boldsymbol{A}$. In many applications, it may not feasible to compute the SVD of the entire matrix $\boldsymbol{A}$. In these cases, algorithms requiring knowledge of the leverage scores cannot be used. Instead, we use an estimate of the block leverage scores called the *approximate block leverage scores*. A subset of the rows are chosen uniformly at random, and the block scores are calculated using the top-k right singular vectors of this row matrix instead of the entire $\boldsymbol{A}$ matrix. This step can also be replaced with other fast approximate calculations of leverage

Method	r	c	No. of sampling ops.
Traditional CUR	$\mathcal{O}\left(\frac{k^2}{\varepsilon^2}\log\left(\frac{1}{\delta}\right)\right)$	$\mathcal{O}\left(\frac{k^4}{\varepsilon^6}\log^3(\frac{1}{\delta})\right)$	$\mathcal{O}\left(\left(\frac{k^2}{\varepsilon^2}+\frac{k^4}{\varepsilon^6}\right)\log^3(\frac{1}{\delta})\right)$
Block CUR	$\mathcal{O}\left(\frac{k^2}{\varepsilon^2}\log(\frac{1}{\delta})\right)$	$\mathcal{O}\left(\frac{sk^4}{\alpha_R\varepsilon^6}\log^3(\frac{1}{\delta})\right)$	$\mathcal{O}\left(\left(\frac{k^2}{\varepsilon^2}+\frac{k^4}{\alpha_R\varepsilon^6}\right)\log^3(\frac{1}{\delta})\right)$

TABLE I: Table showing the number of rows r, columns c, and total number of sampling operation needed for given ϵ in traditional CUR and block CUR. For ease of comparison, we show the block CUR results from Corollary 1 with full SVD computation.

scores involving sketching or additional sampling [7], [8]. The advantage of using our approximate leverage scores is that the same set of rows is used to approximate the scores and also to compute the CUR approximation. Hence no additional sampling or sketching steps are required.

B. Theoretical Results and Discussion

The main technical contribution of the paper is a relative-error bound on the quality of approximation using blocks of columns or rows to approximate a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$. The following result provides a relative-error guarantee for the Block CUR approximation using Algorithm 1,

Theorem 2. *Given $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ with incoherent top-k column space,* i.e. *$\mu \le \mu_0$, let $r = O(\mu_0 \frac{k^2}{\varepsilon^2}\ln(\frac{1}{\delta}))$ and $g = O(\frac{r^2}{\alpha_R\varepsilon^2}\ln(\frac{1}{\delta}))$. There exist randomized algorithms such that, if r rows and g column blocks are chosen to construct $\boldsymbol{R}$ and $\boldsymbol{C}$, respectively, then with probability $\ge 1-\delta$, the following holds:*

$$\|\boldsymbol{A} - \boldsymbol{CUR}\|_F \le (1+\varepsilon)\|\boldsymbol{A}-\boldsymbol{A}_k\|_F$$

where $\varepsilon, \delta \in (0,1)$ and $\boldsymbol{U} = \boldsymbol{W}^\dagger$ is the pseudoinverse of the scaled intersection of $\boldsymbol{C}$ and $\boldsymbol{R}$.

We provide a sketch of the proof in Section V and provide the details in the appendix.

We now detail the differences between our technique and prior CUR algorithms. This includes additional assumptions required, algorithmic trade-offs, and discussion of sampling and computational complexity.

1) Block Stable Rank: The theorem tells us that the number of blocks required to achieve an ε relative error depends on the structure of the blocks (through α_R). Intuitively, this is saying the groups that provide more information improve the approximation faster than less informative groups. The α_R term depends on the stable or numerical rank (a stable relaxation of exact rank) of the blocks. The stable rank $r = r(A) = \|A\|_F^2/\|A\|_2^2$ is a relaxation of the rank of the matrix, in fact it is stable under small perturbations of the matrix A [6]. For instance, the stable rank of an approximately low rank matrix tends to be low. The α_R term defined in Theorem 2 is the minimum stable rank of the column blocks. Thus, the α_R term gives a dependence of the block sampling complexity on the stable ranks of the blocks.

It is easy to check that $1 \le \alpha_R \le s$. In the best case, when all the groups have full stable rank with equal singular values, α_R achieves it's maximum. The worst case $\alpha_R = 1$ is achieved when a group or block is rank-1. That is, sampling groups of rank s gives us a lot more information than groups of rank 1, which leads to a reduction in the final group sampling complexity.

2) Incoherence: The column space incoherence (Definition 4) is used to provide a guarantee for approximation without computing the SVD of the entire matrix $\boldsymbol{A}$. However, if it is possible to compute the SVD of the entire matrix, then the rows can be sampled using row leverage scores, and the incoherence assumption can be dropped. The relative error guarantee for the full SVD Block CUR approximation is stated in the following corollary.

Corollary 1. *Given $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, let $r = O(\frac{k^2}{\varepsilon^2}\ln(\frac{1}{\delta}))$ and $g = O(\frac{r^2}{\alpha_R\varepsilon^2}\ln(\frac{1}{\delta}))$. There exist randomized algorithms such that, if r rows and g column blocks are chosen to construct $\boldsymbol{R}$ and $\boldsymbol{C}$, respectively, then with probability $\geq 1-\delta$, the following holds:*

$$\|\boldsymbol{A} - \boldsymbol{CUR}\|_F \leq (1+\varepsilon)\|\boldsymbol{A} - \boldsymbol{A}_k\|_F$$

where $\varepsilon, \delta \in (0,1)$, and $\boldsymbol{U} = \boldsymbol{W}^\dagger$ is the pseudoinverse of the scaled intersection of $\boldsymbol{C}$ and $\boldsymbol{R}$.

The corollary follows by similar analysis as Theorem 2 so we defer the proof to the appendix. Other than block sampling, the setup of this result is equivalent to the traditional column sampling result stated in Theorem 1. Next, we will use this result to compare block sampling with traditional column sampling.

3) Complexity: The running time of Algorithm 1 is essentially driven by the time required to compute the SVD of $\boldsymbol{R}$ and the construction of $\boldsymbol{R}$, $\boldsymbol{C}$ and $\boldsymbol{U}$. Algorithm 1 requires $\mathcal{O}(SVD(R))$ time, construction of $\boldsymbol{R}$ requires $\mathcal{O}(rn)$ time, construction of $\boldsymbol{C}$ takes $\mathcal{O}(mc)$ time, construction of $\boldsymbol{W}$ requires $\mathcal{O}(rc)$ time and construction of $\boldsymbol{U}$ takes $\mathcal{O}(r^2c)$ time.

The sampling complexity of the Block CUR derived in Theorem 2 and the full SVD Block CUR in Corollary 1 tells us the number of sampling operations or queries that need to be made to the memory in order to construct the $\boldsymbol{R}$ and $\boldsymbol{C}$ matrices. In order to compare complexity of block sampling with individual column sampling we focus our attention on the full SVD Block CUR result in Corollary 1. As shown in Table I the total number of individual columns queried by traditional CUR is always less than or equal to those required by block CUR because $1 \leq \alpha_R \leq s$. However, recall that in the distributed setting the key advantage of sampling column blocks is the efficieny of retrieving blocks of rows or columns at one time corresponding to the rows or columns stored on the same node. Thus, in Table I we see that the number of queries made by the Block CUR algorithm are much less than traditional CUR. In the best case scenario, we can get a speedup proportional to the block size (when $\alpha_R = s$).

In the corner case when each block contains only one important column, block sampling may sample s times more columns in total than column sampling to reach the same error (since $\alpha_R = 1$ and individual sampling has the flexibility to pick only the important columns). However, in this case (with $\alpha_R = 1$) the number of queries or sampling operations for Block CUR will be equal to those made by traditional CUR thus achieving the same time complexity.

In the next section, we demonstrate the speedup achieved by computing the block CUR decomposition of large matrices in a distributed setting using Apache Spark.

IV. Numerical results

In this section we provide empirical comparisons between the block CUR and traditional CUR algorithms on both synthetic and real-world data. We report the relative-error of the decomposition (*i.e.,* $\|\boldsymbol{A} - \boldsymbol{CUR}\|_F / \|\boldsymbol{A}\|_F$)

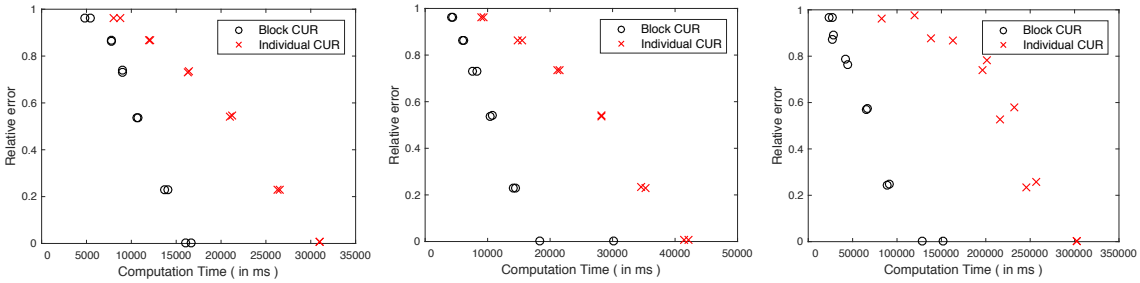

Fig. 2: (Left) Performance on 8000×8000 matrix with rank 800. (Center) Performance on 12000×12000 matrix with rank 1200, and (Right) Performance on 20000×20000 matrix with rank 2000.

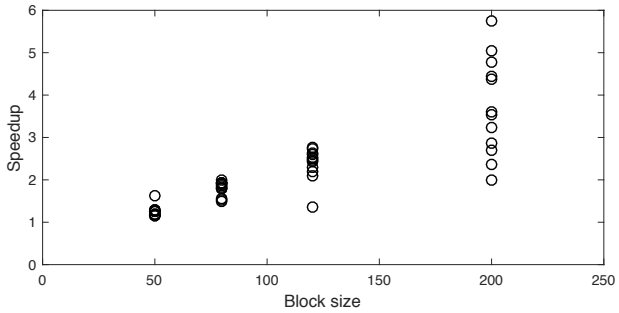

Fig. 3: Runtime speed-up from block sampling when compared to individual column sampling for varying block sizes.

and the running time of each algorithm on different data-sets.

We implemented the algorithms in Scala 2.10 and Apache Spark 2.11 on Amazon Elastic Map-Reduce (Amazon EMR). The compute cluster was constructed using four Amazon m4.4xlarge instances, with each compute node having a Intel Xeon E5-2676 v3 processor and 64 GB of RAM. Using Spark, we store the data sets as resilient distributed dataset (RDD), a collection of elements partitioned across the nodes of the cluster (see Figure 1(a)). In other words, Spark partitions the data into many blocks and distributes these blocks across multiple nodes in the cluster. Using block sampling, we can approximate the matrix by sampling only a subset of the important blocks. Meanwhile, individual column sampling would require looking up all the partitions containing specific columns of interest. Our experiments examine the runtime speed-up from our block sampling CUR that exploits the partitioning of data.

A. Synthetic Experiments

The synthetic data is generated by $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{V}$ where $\boldsymbol{U} \in \mathbb{R}^{m \times k}$ and $\boldsymbol{V} \in \mathbb{R}^{k \times n}$ are random matrices with i.i.d. Gaussian random entries, resulting in a low rank matrix $\boldsymbol{A}$. We perform CUR decomposition on matrices of size $m \times n$ with $m = n$, target rank k, and number of blocks G (set here across all experiments to be 100). The leverage scores are calculated by computing the SVD of the rows sampled uniformly with $\boldsymbol{R} \in \mathbb{R}^{r \times n}$. We sample one-sixth of the rows in all experiments.

Figure 2 shows the plots for the relative error achieved with respect to the runtime required to sample the $\boldsymbol{C}$ and $\boldsymbol{R}$ matrices for both the block CUR and traditional CUR algorithms. To focus on the speed-up achieved by

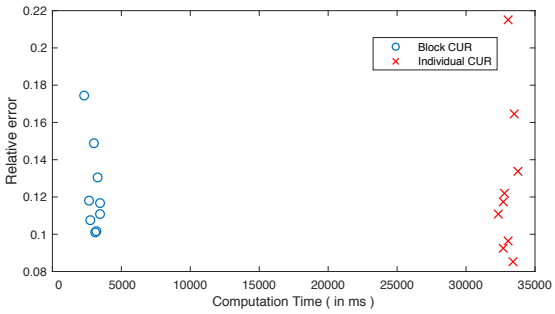

Fig. 4: Performance on 900×10000 Arcene dataset with block size 12.

taking into account the block storage of the data we compare the running time of only the sampling operations of the algorithms (which excludes the time required to compute the SVD). We note that other steps in both algorithms can be updated to include faster variants such as the approximation of the leverage scores by sketching or sampling [7].

We vary g, the number of blocks chosen, from 1 to 6. The number of columns chosen is thus $c = gs$, where s denotes the number of columns in a block and varies from 50 to 200. We repeat each algorithm (block CUR and traditional CUR) twice for the specified number of columns, with each realization as a point in the plot. The proposed block CUR algorithm samples the c columns in g blocks, while the traditional CUR algorithm samples the c columns one at a time.

Consistently, these results show that block sampling achieves the relative error much faster than the individual column sampling – with performance gains increasing as the size of the matrix grows, as shown in Figure 3. While the same amount of data is being transmitted regardless of whether block or individual column sampling is used, block sampling is much faster because it needs to contact fewer executors to retrieve blocks of columns rather the same number of columns individually. In the worst case, sampling individual columns may need to communicate with all of the executors, while block sampling only needs to communicate with g executors. Thus, by exploiting the partitioning of the data, the block CUR approach is able to achieve roughly the same quality of approximation as traditional column-based CUR, as measured by relative error, with significantly less computation time.

B. Real-World Experiment

We also conduct experiments on the Arcene dataset [9] which has 900 rows and 10000 columns. We compare the running time for both block and traditional CUR decomposition. We again find consistent improvements for the block-wise approach compared with individual column sampling. With block size s = 12, sampling up to 10 groups led to an average speed up of 11.22 over individual column sampling, as shown in Figure 4. The matrix is very low rank and sampling a few groups gave small relative errors.

V. Proof of main result

In this section we provide the proof sketch for the relative-error guarantee of approximation provided by Theorem 2. In proving this results, our theoretical results reveal the following contributions:

- As a consequence of the proof of Lemma 1, we show that using block leverage scores if enough blocks are sampled with high probability

$$\|\boldsymbol{A} - \boldsymbol{AS}(\boldsymbol{RS})^\dagger \boldsymbol{R}\|_F \leq (1+\varepsilon)\|\boldsymbol{A} - \boldsymbol{AR}^\dagger \boldsymbol{R}\|_F.$$

  This gives a guarantee on the approximate solution obtained by solving a block-sampled regression problem

$$\min_{\boldsymbol{X}\in\mathbb{R}^{m\times r}} \|(\boldsymbol{AS}) - \boldsymbol{X}(\boldsymbol{RS})\|_F$$

  instead of the entire least squares problem.
- In Lemma 2, we show that the multiplication of two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ can be approximated by the product of smaller sampled and scaled *blocks* of the matrices.
- As a special case of the above result, when $\boldsymbol{R} = \boldsymbol{A}$ we get a bound for the block column subset selection problem. If $g = \mathcal{O}(\frac{k^2}{\alpha_A \varepsilon^2}\log\left(\frac{1}{\delta}\right))$ blocks are chosen, then with probability at least $1-\delta$ we have

$$\|\boldsymbol{A} - \boldsymbol{CC}^\dagger \boldsymbol{A}\|_F \leq (1+\varepsilon)\|\boldsymbol{A} - \boldsymbol{A}_k\|_F.$$

A. Proof Sketch

We start with Lemma 1, which states a non-boosting approximation error result for Algorithm 1.

Lemma 1. *Given $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ with incoherent top-k column space,* i.e. *$\mu \leq \mu_0$, let $r = O(\mu_0 \frac{k^2}{\varepsilon^2})$ and $g = O(\frac{r^2}{\alpha_R \varepsilon^2})$. If rows and column blocks are chosen according to Algorithm 1, then with probability at least 0.7, the following holds:*

$$\|\boldsymbol{A} - \boldsymbol{CUR}\|_F \leq (1+\varepsilon)\|\boldsymbol{A} - \boldsymbol{A}_k\|_F$$

where $\varepsilon \in (0,1)$ and $\boldsymbol{U} = \boldsymbol{W}^\dagger$ is the pseudoinverse of the scaled intersection of $\boldsymbol{C}$ and $\boldsymbol{R}$.

Proof Sketch: First, note $\boldsymbol{U} = (\boldsymbol{RS})^\dagger$ and $\boldsymbol{C} = \boldsymbol{AS}$.

$$\|\boldsymbol{A} - \boldsymbol{CUR}\|_F = \|\boldsymbol{A} - \boldsymbol{AS}(\boldsymbol{RS})^\dagger \boldsymbol{R}\|_F$$

Recall that $\boldsymbol{R} \in \mathbb{R}^{r\times n}$ has rank no greater than r; $\boldsymbol{A} \in \mathbb{R}^{m\times n}$; $\varepsilon \in (0,1)$; and that the same column blocks from $\boldsymbol{R}$ and $\boldsymbol{A}$ are picked with the following probability distribution:

$$p_i = \frac{\ell_i(\boldsymbol{R}, r)}{r} = \frac{\|\boldsymbol{V}_{R,r}^T \boldsymbol{E}_i\|_F^2}{r},\ \forall i \in [G].$$

Here, we show that if $g = \mathcal{O}(\frac{r^2}{\alpha_R \varepsilon^2})$ blocks are chosen, then with probability at least 0.85 we have

$$\|\boldsymbol{A} - \boldsymbol{AS}(\boldsymbol{RS})^\dagger \boldsymbol{R}\|_F \leq (1+\varepsilon)\|\boldsymbol{A} - \boldsymbol{AR}^\dagger \boldsymbol{R}\|_F.$$

First step of the proof is to show that $\boldsymbol{V}_{R,r}^T \boldsymbol{S}$ is full rank. Suppose $\delta' = 0.05$.

Step 1: Using Lemma 2 (stated after this sketch) with $\beta = 1$, if $g = O(r^2 \ln(1/\delta')/\alpha_R \varepsilon_1^2)$ we get the following with probability $\geq 1-\delta'$ we have $\|\boldsymbol{V}_{R,r}^T \boldsymbol{V}_{R,r} - \boldsymbol{V}_{R,r}^T \boldsymbol{SS}^T \boldsymbol{V}_{R,r}\|_2 \leq \frac{\varepsilon_1}{2}$. Further, noting that $\boldsymbol{V}_{R,r}^T \boldsymbol{V}_{R,r} = \boldsymbol{I}_{R,r}$

gives us a bound on the singular values of $\boldsymbol{V}_{R,r}^T\boldsymbol{S}$, for all i,

$$|1-\sigma_i^2(\boldsymbol{V}_{R,r}^T\boldsymbol{S})| = |\sigma_i(\boldsymbol{V}_{R,r}^T\boldsymbol{V}_{R,r}) - \sigma_i(\boldsymbol{V}_{R,r}^T\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{V}_{R,r})|$$
$$\leq \|\boldsymbol{V}_{R,r}^T\boldsymbol{V}_{R,r} - \boldsymbol{V}_{R,r}^T\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{V}_{R,r}\|_2 \leq \frac{\varepsilon_1}{2}$$

Thus, it follows for all singular values of $\boldsymbol{V}_{R,r}^T\boldsymbol{S}$

$$\sqrt{1-\frac{\varepsilon_1}{2}} \leq \sigma_i(\boldsymbol{V}_{R,r}^T\boldsymbol{S}) \leq \sqrt{1+\frac{\varepsilon_1}{2}} \tag{1}$$

If $\varepsilon_1 < 1$, then we have, with probability $\geq 1-\delta'$, that $\boldsymbol{V}_{R,r}^T\boldsymbol{S}$ is full rank.

Step 2: Using this result, we show that $\|\boldsymbol{A} - \boldsymbol{A}\boldsymbol{S}(\boldsymbol{R}\boldsymbol{S})^\dagger\boldsymbol{R}\|_F$ can be bounded by the sum of three terms to separate the random part and matrix perturbation part,

$$\|\boldsymbol{A} - \boldsymbol{A}\boldsymbol{S}(\boldsymbol{R}\boldsymbol{S})^\dagger\boldsymbol{R}\|_F = \|\boldsymbol{A} - \boldsymbol{A}\boldsymbol{S}(\boldsymbol{V}_{R,r}^T\boldsymbol{S})^\dagger\boldsymbol{V}_{R,r}^T\|_F$$
$$\leq \|\boldsymbol{A}\boldsymbol{V}_{R,r}^\perp\boldsymbol{V}_{R,r}^{\perp T}\|_F + \|\boldsymbol{A}\boldsymbol{V}_{R,r}^\perp\boldsymbol{V}_{R,r}^{\perp T}\boldsymbol{S}\|_F\|\Omega\|_2$$
$$+ \|\boldsymbol{A}\boldsymbol{V}_{R,r}^\perp\boldsymbol{V}_{R,r}^{\perp T}\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{V}_{R,r}\|_F$$

where $\Omega = (\boldsymbol{V}_{R,r}^T\boldsymbol{S})^\dagger - (\boldsymbol{V}_{R,r}^T\boldsymbol{S})^T$. From Lemma 2 and using the fact that $\boldsymbol{V}_{R,r}^T\boldsymbol{S}$ is full rank along with basic linear algebra, we bound these terms separately with probability $\geq 1-\delta'$ as follows:

$$\|\boldsymbol{Q}\boldsymbol{S}\|_F \leq (1/\delta')\|\boldsymbol{Q}\|_F \text{ for many } \boldsymbol{Q} \tag{2}$$

$$\|\Omega\|_2 \leq \varepsilon_1/\sqrt{2} \tag{3}$$

$$\|\boldsymbol{A}\boldsymbol{V}_{R,r}^\perp\boldsymbol{V}_{R,r}^{\perp T}\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{V}_{R,r}\|_F \leq \frac{\sqrt{r}}{\delta'\sqrt{\alpha_R g}}\|\boldsymbol{A}\boldsymbol{V}_{R,r}^\perp\boldsymbol{V}_{R,r}^{\perp T}\|_F \tag{4}$$

The proofs are provided in the appendix. Thus, we can conclude the following with probability $\geq 1-3\delta' = 0.85$:

$$\|\boldsymbol{A} - \boldsymbol{A}\boldsymbol{S}(\boldsymbol{R}\boldsymbol{S})^\dagger\boldsymbol{R}\|_F$$
$$\leq \left(1 + \left(\frac{1}{\delta'\sqrt{2}} + \frac{1}{\delta' 2\sqrt{2}}\right)\varepsilon_1\right)\|\boldsymbol{A}\boldsymbol{V}_{R,r}^\perp\boldsymbol{V}_{R,r}^{\perp T}\|_F$$
$$\leq (1+\varepsilon')\|\boldsymbol{A} - \boldsymbol{A}\boldsymbol{R}^\dagger\boldsymbol{R}\|_F$$

For $\delta' = 0.15/3 = 0.05$, we have $\varepsilon' = 2\varepsilon_1/0.05$.

So far we have shown the following

$$\|\boldsymbol{A} - \boldsymbol{C}\boldsymbol{U}\boldsymbol{R}\|_F = \|\boldsymbol{A} - \boldsymbol{A}\boldsymbol{S}(\boldsymbol{R}\boldsymbol{S})^\dagger\boldsymbol{R}\|_F$$
$$\leq (1+\varepsilon')\|\boldsymbol{A} - \boldsymbol{A}\boldsymbol{R}^\dagger\boldsymbol{R}\|_F$$

Next, we bound $\|\boldsymbol{A} - \boldsymbol{A}\boldsymbol{R}^\dagger\boldsymbol{R}\|_F$. Since $\boldsymbol{A}$ has incoherent column space, the uniform sampling distribution $p_j = 1/m$ satisfies eqn. (13) in [1] with $\beta = 1/\mu_0$. Consequently, we can apply modified version of Theorem 1 in [1] to get with probability at least 0.85, $\|\boldsymbol{A} - \boldsymbol{A}\boldsymbol{R}^\dagger\boldsymbol{R}\|_F \leq (1+\varepsilon)\|\boldsymbol{A} - \boldsymbol{A}_k\|_F$. Finally, we get with probability

at least 0.7

$$\begin{aligned}\|\boldsymbol{A}-\boldsymbol{CUR}\|_F &\leq (1+\varepsilon')^2\|\boldsymbol{A}-\boldsymbol{A}_k\|_F \\ &\leq (1+\varepsilon'')\|\boldsymbol{A}-\boldsymbol{A}_k\|_F \qquad \text{letting } \varepsilon''=3\varepsilon'\end{aligned}$$

This completes the proof of Lemma 1.

The result in Theorem 2 follows by applying standard boosting methods to Lemma 1 and running Algorithm 1 $t=\ln(\frac{1}{\delta})$ times. By choosing the solution with minimum error and observing that $0.3<1/e$, we have that the relative error bound holds with probability greater than $1-e^{-t}=1-\delta$. The details of all the proofs are provided in the appendix.

B. Block Multiplication Lemma

Finally, we state an intermediate result that was used in the proof of Lemma 1. It shows that the multiplication of two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ can be approximated by the product of the smaller sampled and scaled block matrices. This is the key lemma in proving the main result.

Lemma 2. *Let $\boldsymbol{A}\in\mathbb{R}^{m\times n}$, $\boldsymbol{B}\in\mathbb{R}^{n\times p}$, and $\varepsilon,\delta\in(0,1)$. Suppose $\alpha_A=\min_k\left(\frac{\|\boldsymbol{A}^{(k)}\|_F}{\|\boldsymbol{A}^{(k)}\|_2}\right)^2$. Construct $\boldsymbol{C}$ and $\boldsymbol{R}$ using the sampling probabilities p_i given by*

$$p_i\geq\beta\frac{\|\boldsymbol{A}^{(i)}\|_F^2}{\sum_{j=1}^{G}\|\boldsymbol{A}^{(j)}\|_F^2}$$

for all $i\in[G]$ and a constant $\beta\in(0,1]$. Then, with probability at least $1-\delta$,

$$\|\boldsymbol{AB}-\boldsymbol{CR}\|_F\leq\frac{1}{\delta\sqrt{\beta\alpha_A g}}\|\boldsymbol{A}\|_F\|\boldsymbol{B}\|_F.$$

The proof is provided in the appendix.

VI. Related work

The concept of sampling the important columns of a matrix based on the notion of *subspace* sampling first appeared in context of fast ℓ_2 regression in [10] and was refined in [1] to obtain performance error guarantees for CUR matrix decomposition. These guarantees were subsequently improved in follow-up work [2].

Modified versions of this problem have been studied extensively for adaptive sampling [11], divide-and-conquer algorithms for parallel computations [12], and input-sparsity algorithms [13]. The authors of [11] propose an adaptive sampling-based algorithm which requires only $c=O(k/\varepsilon)$ columns to be sampled. The authors of [13] also proposed an optimal, deterministic CUR algorithm. In [14], the authors prove the lower bound of the column selection problem; at least $c=k/\varepsilon$ columns are selected to achieve the $(1+\varepsilon)$ ratio. This rate is achieved by more complicated column sampling algorithms, which can be extended to block sampling to achieve tighter bounds on the sampling complexity. We stick to subspace sampling algorithms for the sake of simplicity and defer the extension of more complicated adaptive block sampling algorithms to future work.

Recently, there has been emphasis on settings with the matrix partially known. For example, a CUR algorithm and active column subset selection for partially observed matrices were proposed in [8] and [15], respectively. Our work differs from [8] due to their requirement of prior uniform sampling of the matrix to estimate the leverage scores, while our algorithm does not require these additional samples. With respect to [15], this prior work is focused on column subset selection and does not provide bounds for the CUR approximation.

In contrast to all known prior work, we focus on the block setting where a block of columns is sampled rather than sampling a single column, motivated by the distributed data setting. We analyze this problem by extending the notion of *subspace* sampling to the block setting and give relative error guarantees for block CUR decomposition. To best of our knowledge, this paper is the first to analyze the problem of column *block* subset selection.

Recent work in [16] considers a distributed implementation of CUR matrix decomposition on Apache Spark and demonstrate its capability on very large data sets. In contrast, in this paper we consider the more natural environment of block sampling for distributed computation, explore the performance advantages of block sampling over individual column sampling, and provide the first theoretical error guarantees for block CUR decomposition.

VII. Conclusions and Future Work

In this paper we extended the problem of CUR matrix decomposition to the block setting which is naturally relevant to distributed storage systems. We proposed a novel algorithm and derived its performance bounds. We demonstrated its pratical utility on real-life distributed storage systems. Some possible future directions for this work include calculating the leverage scores quickly or adaptively, and considering the algorithms and error bounds when the matrix has a pre-specified structure like sparsity.

References

[1] Petros Drineas, Michael W Mahoney, and S Muthukrishnan. Relative-error cur matrix decompositions. *SIAM Journal on Matrix Analysis and Applications*, 30(2):844–881, 2008.

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

[3] Peristera Paschou, Elad Ziv, Esteban G Burchard, Shweta Choudhry, William Rodriguez-Cintron, Michael W Mahoney, and Petros Drineas. Pca-correlated snps for structure identification in worldwide human populations. *PLoS Genet*, 3(9):e160, 2007.

[4] Ching-Wa Yip, Michael W Mahoney, Alexander S Szalay, István Csabai, Tamás Budavári, Rosemary FG Wyse, and Laszlo Dobos. Objective identification of informative wavelength regions in galaxy spectra. *The Astronomical Journal*, 147(5):110, 2014.

[5] Jiyan Yang, Oliver Rbel, Michael W Mahoney, and Benjamin P Bowen. Identifying important ions and positions in mass spectrometry imaging data using cur matrix decompositions. *Analytical chemistry*, 87(9):4658–4666, 2015.

[6] Mark Rudelson and Roman Vershynin. Sampling from large matrices: An approach through geometric functional analysis. *Journal of the ACM (JACM)*, 54(4):21, 2007.

[7] Petros Drineas, Malik Magdon-Ismail, Michael W Mahoney, and David P Woodruff. Fast approximation of matrix coherence and statistical leverage. *Journal of Machine Learning Research*, 13(Dec):3475–3506, 2012.

[8] Miao Xu, Rong Jin, and Zhi-Hua Zhou. Cur algorithm for partially observed matrices. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1412–1421, 2015.

[9] Isabelle Guyon, Steve R Gunn, Asa Ben-Hur, and Gideon Dror. Result analysis of the nips 2003 feature selection challenge. In *NIPS*, volume 4, pages 545–552, 2004.

[10] Petros Drineas, Michael W Mahoney, and S Muthukrishnan. Sampling algorithms for l 2 regression and applications. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1127–1136. Society for Industrial and Applied Mathematics, 2006.

[11] Shusen Wang and Zhihua Zhang. A scalable cur matrix decomposition algorithm: lower time complexity and tighter bound. In *Advances in Neural Information Processing Systems*, pages 647–655, 2012.

[12] Lester W Mackey, Michael I Jordan, and Ameet Talwalkar. Divide-and-conquer matrix factorization. In *Advances in Neural Information Processing Systems*, pages 1134–1142, 2011.

[13] Christos Boutsidis and David P Woodruff. Optimal cur matrix decompositions. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 353–362. ACM, 2014.

[14] Christos Boutsidis, Petros Drineas, and Malik Magdon-Ismail. Near-optimal column-based matrix reconstruction. *SIAM Journal on Computing*, 43(2):687–717, 2014.

[15] Yining Wang and Aarti Singh. Provably correct active sampling algorithms for matrix column subset selection with missing data. *arXiv preprint arXiv:1505.04343*, 2015.

[16] Alex Gittens, Jey Kottalam, Jiyan Yang, Michael F. Ringenburg, Jatin Chhugani, Evan Racahk, Mohitdeep Singh, Yushu Yao, Curt Fischer, Oliver Ruebel, Benjamin Bowen, Norman G. Lewis, Michael W. Mahoney, Venkat Krishnamurthy, and Prabhat. A multi-platform evaluation of the randomized CX low-rank matrix factorization in Spark. In *International Workshop on Parallel and Distributed Computing for Large Scale Machine Learning and Big Data Analytics (ParLearning)*, 2016.

APPENDIX

A. *Proof of Lemma 2*

Proof. Note that,

$$\boldsymbol{E}\left[\left(\frac{\boldsymbol{A}^{(j_t)}\boldsymbol{B}_{(j_t)}}{gp_{j_t}}\right)_{i_1i_2}\right] = \sum_{k=1}^{G} p_k \left(\frac{\boldsymbol{A}^{(k)}\boldsymbol{B}_{(k)}}{gp_k}\right)_{i_1i_2} = \frac{1}{g}(\boldsymbol{AB})_{i_1i_2}$$

Since each block is picked independently we have,

$$\begin{aligned}
var[(\boldsymbol{CR})_{i_1i_2}] &= var\left[\sum_{t=1}^{g}\left(\frac{\boldsymbol{A}^{(j_t)}\boldsymbol{B}_{(j_t)}}{gp_{j_t}}\right)_{i_1i_2}\right] \\
&= \sum_{t=1}^{g} var\left[\left(\frac{\boldsymbol{A}^{(j_t)}\boldsymbol{B}_{(j_t)}}{gp_{j_t}}\right)_{i_1i_2}\right] \\
&= \sum_{t=1}^{g}\left(\boldsymbol{E}\left[\left(\frac{\boldsymbol{A}^{(j_t)}\boldsymbol{B}_{(j_t)}}{gp_{j_t}}\right)^2_{i_1i_2}\right] - \boldsymbol{E}\left[\left(\frac{\boldsymbol{A}^{(j_t)}\boldsymbol{B}_{(j_t)}}{gp_{j_t}}\right)_{i_1i_2}\right]^2\right) \\
&= g\left(\sum_{k=1}^{G} p_k\left(\frac{\boldsymbol{A}^{(k)}\boldsymbol{B}_{(k)}}{gp_k}\right)^2_{i_1i_2} - \frac{(\boldsymbol{AB})^2_{i_1i_2}}{g^2}\right) \\
&= \frac{1}{g}\left(\sum_{k=1}^{G}\frac{\left(\boldsymbol{A}^{(k)}\boldsymbol{B}_{(k)}\right)^2_{i_1i_2}}{p_k} - (\boldsymbol{AB})^2_{i_1i_2}\right)
\end{aligned}$$

$$
\begin{aligned}
\mathbb{E}[\|\boldsymbol{AB}-\boldsymbol{CR}\|_F^2] &= \sum_{i_1=1}^{m}\sum_{i_2=1}^{p} var[(\boldsymbol{CR})_{i_1 i_2}] \\
&= \sum_{i_1=1}^{m}\sum_{i_2=1}^{p}\frac{1}{g}\left(\sum_{k=1}^{G}\frac{\left(\boldsymbol{A}^{(k)}\boldsymbol{B}_{(k)}\right)^2_{i_1 i_2}}{p_k} - (\boldsymbol{AB})^2_{i_1 i_2}\right) \\
&= \left(\sum_{k=1}^{G}\frac{1}{gp_k}\sum_{i_1=1}^{m}\sum_{i_2=1}^{p}\left(\boldsymbol{A}^{(k)}\boldsymbol{B}_{(k)}\right)^2_{i_1 i_2}\right) - \frac{\|\boldsymbol{AB}\|_F^2}{g} \\
&= \sum_{k=1}^{G}\frac{\|\boldsymbol{A}^{(k)}\boldsymbol{B}_{(k)}\|_F^2}{gp_k} - \frac{\|\boldsymbol{AB}\|_F^2}{g} \\
&\leq \sum_{k=1}^{G}\frac{\|\boldsymbol{A}^{(k)}\|_2^2\|\boldsymbol{B}_{(k)}\|_F^2}{gp_k} \\
&\leq \sum_{k=1}^{G}\left(\frac{\|\boldsymbol{A}^{(k)}\|_2}{\|\boldsymbol{A}^{(k)}\|_F}\right)^2\frac{\|\boldsymbol{A}^{(k)}\|_F^2\|\boldsymbol{B}_{(k)}\|_F^2}{gp_k} \\
&\leq \frac{1}{\beta\alpha_A g}\|\boldsymbol{A}\|_F^2\|\boldsymbol{B}\|_F^2
\end{aligned}
$$

where $\alpha_A = \min_k\left(\frac{\|\boldsymbol{A}^{(k)}\|_F}{\|\boldsymbol{A}^{(k)}\|_2}\right)^2$. Also, note $1 \leq \alpha_A \leq s$.

By Jensen's inequality,

$$
\begin{aligned}
\mathbb{E}[\|\boldsymbol{AB}-\boldsymbol{CR}\|_F]^2 &\leq \mathbb{E}[\|\boldsymbol{AB}-\boldsymbol{CR}\|_F^2] \\
&\leq \frac{1}{\beta\alpha_A g}\|\boldsymbol{A}\|_F^2\|\boldsymbol{B}\|_F^2
\end{aligned}
$$

And by Markov's inequality, with probability $\geq 1-\delta$, we have

$$
\|\boldsymbol{AB}-\boldsymbol{CR}\|_F \leq \frac{1}{\delta}\mathbb{E}[\|\boldsymbol{AB}-\boldsymbol{CR}\|_F] \leq \frac{1}{\delta\sqrt{\beta\alpha_A g}}\|\boldsymbol{A}\|_F\|\boldsymbol{B}\|_F
$$

□

B. Proof of Lemma 1

First, note $\boldsymbol{U} = (\boldsymbol{RS})^\dagger$ and $\boldsymbol{C} = \boldsymbol{AS}$.

$$
\|\boldsymbol{A}-\boldsymbol{CUR}\|_F = \|\boldsymbol{A}-\boldsymbol{AS}(\boldsymbol{RS})^\dagger\boldsymbol{R}\|_F
$$

Recall that $\boldsymbol{R} \in \mathbb{R}^{r\times n}$ has rank no greater than r; $\boldsymbol{A} \in \mathbb{R}^{m\times n}$; $\varepsilon \in (0,1)$; and that the same column blocks from

$\boldsymbol{R}$ and $\boldsymbol{A}$ are picked with the following probability distribution:

$$p_i = \frac{\|\boldsymbol{V}_{R,r}^T \boldsymbol{E}_i\|_F^2}{r},\ \forall i \in [G].$$

We can use Lemma 1.1 (stated and proved in next section) with probability at least 0.85 we have

$$\|\boldsymbol{A} - \boldsymbol{AS}(\boldsymbol{RS})^\dagger \boldsymbol{R}\|_F \leq (1+\varepsilon)\|\boldsymbol{A} - \boldsymbol{AR}^\dagger \boldsymbol{R}\|_F.$$

In the paper we collapse Lemma 1.1 into the proof of Lemma 1, but here we state and prove the Lemma separately since it may be of general interest.

Next, we bound $\|\boldsymbol{A} - \boldsymbol{AR}^\dagger \boldsymbol{R}\|_F$. Since $\boldsymbol{A}$ has incoherent column space, the uniform sampling distribution $p_j = 1/m$ satisfies eqn. (13) in [1] with $\beta = 1/\mu_0$. Consequently, we can apply modified version of Theorem 1 in [1] we get with probability at least 0.85, $\|\boldsymbol{A} - \boldsymbol{AR}^\dagger \boldsymbol{R}\|_F \leq (1+\varepsilon)\|\boldsymbol{A} - \boldsymbol{A}_k\|_F$. Finally, we get with probability 0.7

$$\begin{aligned}\|\boldsymbol{A} - \boldsymbol{CUR}\|_F &\leq (1+\varepsilon')^2 \|\boldsymbol{A} - \boldsymbol{A}_k\|_F \\ &\leq (1+\varepsilon'')\|\boldsymbol{A} - \boldsymbol{A}_k\|_F \qquad \text{letting } \varepsilon'' = 3\varepsilon'\end{aligned}$$

This completes the proof of Lemma 1.

1) Approximating generalized ℓ_2 regression in the block setting:

In this section, we give theory for generalized least squares using block subset selection that is used to prove the main results for the algorithms but applies to arbitrary matrices $\boldsymbol{A}$ and $\boldsymbol{B}$. Given matrices $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{r\times n}$, the generalized least squares problem is

$$\min_{\boldsymbol{X}\in\mathbb{R}^{m\times r}} \|\boldsymbol{A} - \boldsymbol{XB}\|_F.$$

It is well-known that the solution to this optimization problem is given by $\widehat{\boldsymbol{X}} = \boldsymbol{AB}^\dagger$. To approximate this problem by a subsampled problem, we sample some blocks of columns from $\boldsymbol{A}$ and $\boldsymbol{B}$ to approximate the standard ℓ_2 regression by the following optimization:

$$\min_{\boldsymbol{X}\in\mathbb{R}^{m\times r}} \|(\boldsymbol{AS}) - \boldsymbol{X}(\boldsymbol{BS})\|_F.$$

The solution of this problem is given by $\tilde{\boldsymbol{X}} = \boldsymbol{AS}(\boldsymbol{BS})^\dagger$. In the following lemma, we give a guarantee stating that, when enough blocks are sampled with the specified probability, the approximate solution is close to the actual solution to the ℓ_2 regression.

Lemma 3. *Suppose $\boldsymbol{B} \in \mathbb{R}^{r\times n}$ has rank no greater than k; $\boldsymbol{A} \in \mathbb{R}^{m\times n}; \varepsilon, \delta \in (0,1)$; and let the same column blocks from $\boldsymbol{B}$ and $\boldsymbol{A}$ be picked with the following probability distribution:*

$$p_i = \frac{\|(\boldsymbol{V}_{B,k})_{(i)}\|_F^2}{k},\ \forall i \in [G].$$

If $g = \mathcal{O}(\frac{k^2}{\alpha_B \delta^4 \varepsilon^2})$ blocks are chosen, then with probability at least $1-\delta$ we have

$$\|\boldsymbol{A} - \boldsymbol{AS}(\boldsymbol{BS})^\dagger \boldsymbol{B}\|_F \leq (1+\varepsilon)\|\boldsymbol{A} - \boldsymbol{AB}^\dagger \boldsymbol{B}\|_F.$$

Proof. Let $\boldsymbol{B} = \boldsymbol{U}_k \boldsymbol{\Sigma}_k \boldsymbol{V}_k^T$ and $\alpha = \max_i \left(\frac{\|\boldsymbol{V}_k^T E_i\|_2}{\|\boldsymbol{V}_k^T E_i\|_F}\right)^2$

We start by showing $\boldsymbol{V}_k^T \boldsymbol{S}$ is full rank. Using Lemma 2, if $g \geq 8\alpha_R^{-1} k^2 \delta^{-2} \varepsilon_1^{-2}$ and $0 < \varepsilon_1 < 1$, we get the following with probability $\geq 1 - \delta_1$

$$\|\boldsymbol{V}_k^T \boldsymbol{V}_k - \boldsymbol{V}_k^T \boldsymbol{S}\boldsymbol{S}^T \boldsymbol{V}_k\|_2 = \|\boldsymbol{I}_k - \boldsymbol{V}_k^T \boldsymbol{S}\boldsymbol{S}^T \boldsymbol{V}_k\|_2 \leq 4\frac{k}{\delta\sqrt{\alpha_R g}} \leq \frac{\varepsilon_1}{2}$$

This further gives us a bound on the singular values of $\boldsymbol{V}_k^T \boldsymbol{S}$, for all i,

$$|1 - \sigma_i^2(\boldsymbol{V}_k^T \boldsymbol{S})| = |\sigma_i(\boldsymbol{V}_k^T \boldsymbol{V}_k) - \sigma_i(\boldsymbol{V}_k^T \boldsymbol{S}\boldsymbol{S}^T \boldsymbol{V}_k)| \leq \|\boldsymbol{I}_k - \boldsymbol{V}_k^T \boldsymbol{S}\boldsymbol{S}^T \boldsymbol{V}_k\|_2 \leq \varepsilon_1 \tag{5}$$

Thus, it follows for all singular values of $\boldsymbol{V}_k^T \boldsymbol{S}$

$$\sqrt{1-\varepsilon_1} \leq \sigma_i(\boldsymbol{V}_k^T \boldsymbol{S}) \leq \sqrt{1+\varepsilon_1} \tag{6}$$

Now, consider

$$\begin{aligned}
\|\Omega\|_2 &= \|(\boldsymbol{V}_k^T \boldsymbol{S})^\dagger - (\boldsymbol{V}_k^T \boldsymbol{S})^T\|_2 && \text{by definition} \\
&= \|\boldsymbol{\Sigma}^{-1}_{\boldsymbol{V}_k^T \boldsymbol{S}} - \boldsymbol{\Sigma}_{\boldsymbol{V}_k^T \boldsymbol{S}}\|_2 && \text{from SVD} \\
&= \max_i \left|\sigma_i(\boldsymbol{V}_k^T \boldsymbol{S}) - \frac{1}{\sigma_i(\boldsymbol{V}_k^T \boldsymbol{S})}\right| && \text{by definition} \\
&= \max_i \frac{|\sigma_i^2(\boldsymbol{V}_k^T \boldsymbol{S}) - 1|}{|\sigma_i(\boldsymbol{V}_k^T \boldsymbol{S})|} && \text{by simple manipulation} \\
&\leq \frac{\|\boldsymbol{V}_k^T \boldsymbol{V}_k - \boldsymbol{V}_k^T \boldsymbol{S}\boldsymbol{S}^T \boldsymbol{V}_k\|_2}{\sqrt{1 - \|\boldsymbol{V}_k^T \boldsymbol{V}_k - \boldsymbol{V}_k^T \boldsymbol{S}\boldsymbol{S}^T \boldsymbol{V}_k\|_2}} && \text{from equation 5} \\
&\leq \frac{\varepsilon_1/2}{\sqrt{1-\varepsilon_1/2}} && \text{from lemma 2 w.h.p.} \\
&\leq \varepsilon_1/\sqrt{2} && \text{if } \varepsilon_1 < 1 \Rightarrow \sqrt{1-\varepsilon_1/2} > 1/\sqrt{2}
\end{aligned}$$

Also, for any $\boldsymbol{Q}$ we have ,

$$\mathbb{E}[\|\boldsymbol{QS}\|_F^2] = \mathbb{E}\left[\sum_{t=1}^{g} \left\| \frac{1}{\sqrt{g p_{j_t}}} \boldsymbol{Q}^{(j_t)} \right\|_F^2\right] = \sum_{t=1}^{g} \mathbb{E}\left[\frac{1}{g p_{j_t}} \left\|\boldsymbol{Q}^{(j_t)}\right\|_F^2\right] = \sum_{t=1}^{g}\sum_{i=1}^{G} p_i \frac{1}{g p_i} \left\|\boldsymbol{Q}^{(i)}\right\|_F^2 = \|\boldsymbol{Q}\|_F^2$$

By Jensen's inequality,

$$\mathbb{E}[\|\boldsymbol{QS}\|_F]^2 \leq \mathbb{E}[\|\boldsymbol{QS}\|_F^2] = \|\boldsymbol{Q}\|_F^2$$

By applying Markov's inequality, we get with probability $\geq 1 - \delta'$,

$$\|\boldsymbol{QS}\|_F \leq \frac{1}{\delta'}\mathbb{E}[\|\boldsymbol{QS}\|_F] \leq \frac{1}{\delta'}\|\boldsymbol{Q}\|_F \tag{7}$$

The following will be useful later,

$$\boldsymbol{AS}(\boldsymbol{BS})^\dagger \boldsymbol{B} = \boldsymbol{AS}(\boldsymbol{U}_k\boldsymbol{\Sigma}_k\boldsymbol{V}_k^T\boldsymbol{S})^\dagger\boldsymbol{U}_k\boldsymbol{\Sigma}_k\boldsymbol{V}_k^T = \boldsymbol{AS}(\boldsymbol{V}_k^T\boldsymbol{S})^\dagger\boldsymbol{\Sigma}_k^{-1}\boldsymbol{U}_k^T\boldsymbol{U}_k\boldsymbol{\Sigma}_k\boldsymbol{V}_k^T = \boldsymbol{AS}(\boldsymbol{V}_k^T\boldsymbol{S})^\dagger\boldsymbol{V}_k^T$$

We break down the left hand term into 3 manageable components:

$$\begin{aligned}
&\|\boldsymbol{A} - \boldsymbol{AS}(\boldsymbol{BS})^\dagger\boldsymbol{B}\|_F \\
&= \|\boldsymbol{A} - \boldsymbol{AS}(\boldsymbol{V}_k^T\boldsymbol{S})^\dagger\boldsymbol{V}_k^T\|_F && \text{by Lemma (shown above)} \\
&= \|\boldsymbol{A} - \boldsymbol{A}\boldsymbol{V}_k\boldsymbol{V}_k^T\boldsymbol{S}(\boldsymbol{V}_k^T\boldsymbol{S})^\dagger\boldsymbol{V}_k^T + \boldsymbol{A}\boldsymbol{V}_k^\perp\boldsymbol{V}_k^{\perp T}\boldsymbol{S}(\boldsymbol{V}_k^T\boldsymbol{S})^\dagger\boldsymbol{V}_k^T\|_F && \text{since } (\boldsymbol{V}_k\boldsymbol{V}_k^T + \boldsymbol{V}_k^\perp\boldsymbol{V}_k^{\perp T}) = \boldsymbol{I} \\
&= \|\boldsymbol{A} - \boldsymbol{A}\boldsymbol{V}_k\boldsymbol{V}_k^T + \boldsymbol{A}\boldsymbol{V}_k^\perp\boldsymbol{V}_k^{\perp T}\boldsymbol{S}(\boldsymbol{V}_k^T\boldsymbol{S})^\dagger\boldsymbol{V}_k^T\|_F && \text{since } \boldsymbol{V}_k^T\boldsymbol{S} \text{ is full rank} \\
&\leq \|\boldsymbol{A} - \boldsymbol{A}\boldsymbol{V}_k\boldsymbol{V}_k^T\|_F + \|\boldsymbol{A}\boldsymbol{V}_k^\perp\boldsymbol{V}_k^{\perp T}\boldsymbol{S}(\boldsymbol{V}_k^T\boldsymbol{S})^\dagger\boldsymbol{V}_k^T\|_F && \text{triangle inequality} \\
&= \|\boldsymbol{A}\boldsymbol{V}_k^\perp\boldsymbol{V}_k^{\perp T}\|_F + \|\boldsymbol{A}\boldsymbol{V}_k^\perp\boldsymbol{V}_k^{\perp T}\boldsymbol{S}(\Omega + (\boldsymbol{V}_k^T\boldsymbol{S})^T)\|_F && \text{define } \Omega = (\boldsymbol{V}_k^T\boldsymbol{S})^\dagger - (\boldsymbol{V}_k^T\boldsymbol{S})^T \\
&\leq \|\boldsymbol{A}\boldsymbol{V}_k^\perp\boldsymbol{V}_k^{\perp T}\|_F + \|\boldsymbol{A}\boldsymbol{V}_k^\perp\boldsymbol{V}_k^{\perp T}\boldsymbol{S}\|_F\|\Omega\|_2 + \|\boldsymbol{A}\boldsymbol{V}_k^\perp\boldsymbol{V}_k^{\perp T}\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{V}_k\|_F && \text{submultiplicativity} \\
&\leq \left(1 + \frac{1}{\delta'}\|\Omega\|_2\right)\|\boldsymbol{A}\boldsymbol{V}_k^\perp\boldsymbol{V}_k^{\perp T}\|_F + \|\boldsymbol{A}\boldsymbol{V}_k^\perp\boldsymbol{V}_k^{\perp T}\boldsymbol{V}_k - \boldsymbol{A}\boldsymbol{V}_k^\perp\boldsymbol{V}_k^{\perp T}\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{V}_k\|_F && \text{by (7) and since } \boldsymbol{V}_k^{\perp T}\boldsymbol{V}_k = 0 \\
&\leq \left(1 + \frac{1}{\delta'}\|\Omega\|_2\right)\|\boldsymbol{A}\boldsymbol{V}_k^\perp\boldsymbol{V}_k^{\perp T}\|_F + \frac{1}{\delta_2\sqrt{\alpha_R g}}\|\boldsymbol{A}\boldsymbol{V}_k^\perp\boldsymbol{V}_k^{\perp T}\|_F\|\boldsymbol{V}_k\|_F && \text{Lemma 2 and} \|\boldsymbol{V}_k\|_F = \sqrt{k} \\
&\leq \left(1 + \frac{1}{\delta'}\|\Omega\|_2\right)\|\boldsymbol{A}\boldsymbol{V}_k^\perp\boldsymbol{V}_k^{\perp T}\|_F + \frac{\sqrt{k}}{\delta_2\sqrt{\alpha_R g}}\|\boldsymbol{A}\boldsymbol{V}_k^\perp\boldsymbol{V}_k^{\perp T}\|_F && \|\boldsymbol{A} - \boldsymbol{A}\boldsymbol{B}^\dagger\boldsymbol{B}\|_F = \|\boldsymbol{A}\boldsymbol{V}_k^\perp\boldsymbol{V}_k^{\perp T}\|_F \\
&\leq \left(1 + \frac{1}{\delta'}\|\Omega\|_2 + \frac{\varepsilon_1}{\sqrt{8}\delta_2}\right)\|\boldsymbol{A}\boldsymbol{V}_k^\perp\boldsymbol{V}_k^{\perp T}\|_F && \text{see (*) below} \\
&\leq \left(1 + \frac{1}{\delta'}\|\Omega\|_2 + \frac{\varepsilon_1}{\sqrt{8}\delta_2}\right)\|\boldsymbol{A} - \boldsymbol{A}\boldsymbol{B}^\dagger\boldsymbol{B}\|_F && \text{since } \boldsymbol{A}\boldsymbol{V}_k\boldsymbol{V}_k^T = \boldsymbol{A}\boldsymbol{B}^\dagger\boldsymbol{B}
\end{aligned}$$

where (*) follows since $\frac{\sqrt{k}}{\sqrt{\alpha_R g}} \leq \frac{k}{\delta_1\sqrt{\alpha_R g}} \leq \frac{\varepsilon_1}{\sqrt{8}}$

Thus, we can conclude the following with probability $\geq 1 - (\delta' + \delta_1 + \delta_2) = 1 - \delta$

$$\begin{aligned}
\|\boldsymbol{A} - \boldsymbol{AS}(\boldsymbol{BS})^\dagger\boldsymbol{B}\|_F &\leq \left(1 + \left(\frac{1}{\sqrt{2}\delta'} + \frac{1}{2\delta_2}\right)\varepsilon_1\right)\|\boldsymbol{A} - \boldsymbol{A}\boldsymbol{B}^\dagger\boldsymbol{B}\|_F \\
&\leq (1+\varepsilon)\|\boldsymbol{A} - \boldsymbol{A}\boldsymbol{B}^\dagger\boldsymbol{B}\|_F
\end{aligned}$$

by setting $\delta' = \delta_1 = \delta_2 = \delta/3$ and $\varepsilon = \frac{6\varepsilon_1}{\delta}$.

Lemma 2 is used, then $g \geq 36 * 8\alpha_R\frac{k^2}{\varepsilon^2\delta^4}$. Finally, note that $\varepsilon_1 \leq \varepsilon < 1$ by assumption.

□

C. Proof of Corollary 1

First, note $\boldsymbol{U} = (\boldsymbol{RS})^\dagger$ and $\boldsymbol{C} = \boldsymbol{AS}$.

$$\|\boldsymbol{A} - \boldsymbol{CUR}\|_F = \|\boldsymbol{A} - \boldsymbol{AS}(\boldsymbol{RS})^\dagger\boldsymbol{R}\|_F$$

Similar to the proof of Lemma 1, we can use Lemma 1.1 with probability at least 0.85 we have

$$\|\boldsymbol{A} - \boldsymbol{AS}(\boldsymbol{RS})^{\dagger}\boldsymbol{R}\|_F \leq (1+\varepsilon)\|\boldsymbol{A} - \boldsymbol{AR}^{\dagger}\boldsymbol{R}\|_F.$$

Recall that $\boldsymbol{A} \in \mathbb{R}^{m\times n}; \varepsilon \in (0,1)$; and that the rows $\boldsymbol{R}$ are picked from $\boldsymbol{A}$ with the following probability distribution:

$$p_i = \frac{\|e_i^T \boldsymbol{U}_{A,k}\|_2^2}{k}, \ \forall i \in [m].$$

We bound $\|\boldsymbol{A} - \boldsymbol{AR}^{\dagger}\boldsymbol{R}\|_F$ using Theorem 1 in [1] we get with probability at least 0.85, $\|\boldsymbol{A} - \boldsymbol{AR}^{\dagger}\boldsymbol{R}\|_F \leq (1+\varepsilon)\|\boldsymbol{A} - \boldsymbol{A}_k\|_F$. Finally, we get with probability 0.7

$$\begin{aligned}
\|\boldsymbol{A} - \boldsymbol{CUR}\|_F &\leq (1+\varepsilon')^2\|\boldsymbol{A} - \boldsymbol{A}_k\|_F \\
&\leq (1+\varepsilon'')\|\boldsymbol{A} - \boldsymbol{A}_k\|_F \qquad \text{letting } \varepsilon'' = 3\varepsilon'
\end{aligned}$$

This completes the proof of Corollary 1.