Block CUR: Decomposing Matrices using Groups of Columns

Urvashi Oswal[1], Swayambhoo Jain[2], Kevin S. Xu[3], and Brian Eriksson[4⋆]

[1] University of Wisconsin-Madison, Madison, WI
[2] Technicolor Research, Palo Alto, CA
[3] The University of Toledo, Toledo, OH
[4] Adobe, San Jose, CA

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. We present an algorithm for sampling useful column blocks and provide novel guarantees for the quality of the approximation. This algorithm has application in problems as diverse as biometric data analysis to distributed computing. We demonstrate the effectiveness of the proposed algorithms for computing the Block CUR decomposition of large matrices in a distributed setting with 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. In the biometric setting, the rows correspond to different users and columns correspond to users' biometric reaction to external stimuli, *e.g.,* watching video content, at a particular time instant. There is significant cost in acquiring each user's reaction to lengthy content so we sample a few important scenes to approximate the biometric response. An individual time sample in this use case cannot be queried in isolation due to the lack of context that caused that biometric reaction. Instead, collections of time segments (*i.e.,* blocks) must be presented to the user. The practical application of these algorithms is shown via experimental results using real-world user biometric data from a content testing environment.

1 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 [5].

⋆ This research was performed while U.O., K.S.X., and B.E. were at Technicolor.

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 often do not preserve the structure in original data. Preserving the original structure in the data may be desirable due to many reasons including interpretability 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 rows and columns of the matrix itself. This specific decomposition describes the matrix $\boldsymbol{A}$ as the product of a subset of matrix rows $\boldsymbol{R}$ and a subset of matrix columns $\boldsymbol{C}$ (along with a matrix $\boldsymbol{U}$ that fits $\boldsymbol{A} \approx \boldsymbol{CUR}$).

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.,* [11]). These methods have been applied successfully to many real-world problems including genetics [13], astronomy [19], and mass spectrometry imaging [18]. Unfortunately, a primary assumption of current CUR techniques, that individual rows and columns of the matrix can be queried, is either impossible or quite costly in many real world problems and instead require a block approach.

In this paper, we consider the following two applications which represent the two main motivating factors for considering block decompositions.

Biometric data analysis. In applications where the ordering of rows or columns is meaningful, such as images, video, or speech data matrices, sampling contiguous blocks of columns adds contextual information that is necessary for interpretability of the factorized representation. One emerging application is audience reaction analysis of video content using biometrics. We focus on the scenario where users watch video content while wearing sensors, and changes in biometric sensors indicate changes in reaction to the content. For example, increases in heart rate or a spike in electrodermal activity indicate an increase in content engagement. In this paper, a matrix of biometric data such as *Electrodermal Activity* (EDA) is collected from users reacting to external stimuli, e.g., watching video content. In prior work, EDA has shown to be useful for a variety of user analytics tasks to assess the reaction of viewers[15,8]. In this setting, m is the number of users and n corresponds to the number of time samples for which biometric reaction is collected. Unfortunately, there is significant cost in acquiring each user's reaction to lengthy content so instead we collect full responses (corresponding to some rows of the matrix) from only a limited number of users. For remaining users, we propose to collect responses for only a few important scenes of the video (corresponding to column blocks of the matrix) as shown in Figure 1a and then *approximate* their full response. An individual time sample in this use case cannot be queried in isolation due to the lack of context that caused that biometric reaction. Instead, collections of time segments (i.e., blocks) must be presented to the user. In this setting block sampling can be viewed as a **restriction** which leads to more interpretable solutions.

Distributed storage systems. Large-scale datasets often 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 1b, in order to minimize the overhead in terms of

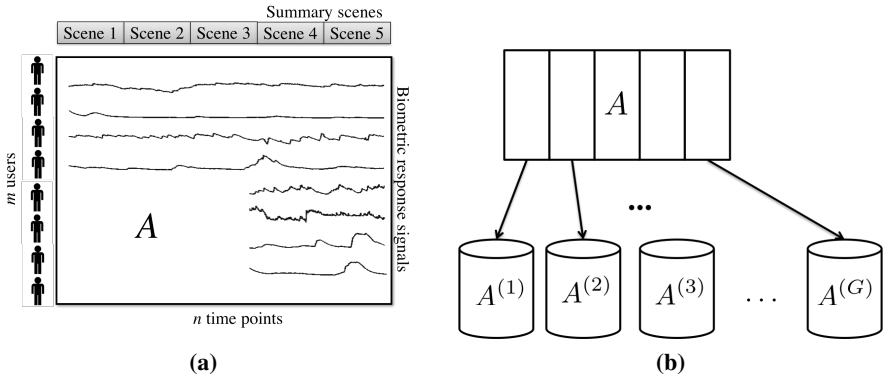


Fig. 1: Applications: (a) Biometric data analysis. Blocks of columns or time instances correspond to scenes in a video and provide context for biometric reaction. (b) Distributed storage of a large matrix across multiple nodes in a cluster. Blocks are allocated to each of the G nodes.

latency while keeping the throughput constant. In doing so, one forms a Block CUR decomposition, with more details provided in Section 4.2. Current CUR decomposition techniques do not take advantage of this predefined block structure.

Main contributions. Using these insights into real-world applications of CUR decomposition, this paper makes a series of contributions. We propose a simple randomized Block CUR algorithm for subset selection of rows and blocks of columns and derive novel worst-case error bounds for this randomized algorithm. On the theory side, we present new theoretical results related to approximating matrix multiplication and generalized ℓ_2 regression in the block setting. These results are the fundamental building blocks used to derive the error bounds for the presented randomized algorithms. The sample complexity bounds feature a non-trivial dependence on the matrix partition, $i.e.$, the distribution of information in the blocks of the matrix. This dependence is non-trivial in that it cannot be obtained by simply extending the analysis of the original individual column CUR setting to the Block CUR setting. As a result, our analysis finds a sample complexity improvement on the order of the *block stable rank* of a matrix (See Table 1 in Section 3).

On the practical side, this algorithm performs fast block sampling taking advantage of the natural storage of matrices in distributed environments (See Table 2 in Section 4.2). 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 using Apache Spark on Amazon Elastic Map-Reduce (Amazon EMR) 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. We also perform experiments with real-world user biometric data from a content testing environment and present interesting use cases where our algorithms can be applied to user analytics tasks.

2 Setup and background

2.1 Notation

Let $\boldsymbol{I}_k$ denote the $k \times k$ identity matrix and $\mathbf{0}$ denote a zero matrix of appropriate size. We denote vectors (matrices) with lowercase (uppercase) bold symbols like $\boldsymbol{a}$ ($\boldsymbol{A}$). The i-th row (column) of a matrix is denoted by $\boldsymbol{A}_i$ ($\boldsymbol{A}^i$). We represent the i-th block of rows of a matrix by $\boldsymbol{A}_{(i)}$ and the i-th block of columns of a matrix by $\boldsymbol{A}^{(i)}$.

Let $[n]$ denote the set $\{1, 2, \ldots, n\}$. Let $\rho = \text{rank}(\boldsymbol{A}) \leq \min\{m, n\}$ and $k \leq \rho$. The singular value decomposition (SVD) of $\boldsymbol{A}$ can be written as $\boldsymbol{A} = \boldsymbol{U}_{A,\rho}\boldsymbol{\Sigma}_{A,\rho}\boldsymbol{V}_{A,\rho}^T$ where $\boldsymbol{U}_{A,\rho} \in \mathbb{R}^{m\times\rho}$ contains the ρ left singular vectors; $\boldsymbol{\Sigma}_{A,\rho} \in \mathbb{R}^{\rho\times\rho}$ is the diagonal matrix of singular values, $\sigma_i(\boldsymbol{A})$ for $i = 1, \ldots, \rho$; and $\boldsymbol{V}_{A,\rho}^T \in \mathbb{R}^{\rho\times n}$ is an orthonormal matrix containing the ρ right singular vectors of $\boldsymbol{A}$. We denote $\boldsymbol{A}_k = \boldsymbol{U}_{A,k}\boldsymbol{\Sigma}_{A,k}\boldsymbol{V}_{A,k}^T$ as the best rank-k approximation to $\boldsymbol{A}$ in terms of Frobenius norm. The pseudoinverse of $\boldsymbol{A}$ is defined as $\boldsymbol{A}^\dagger = \boldsymbol{V}_{A,\rho}\boldsymbol{\Sigma}_{A,\rho}^{-1}\boldsymbol{U}_{A,\rho}^T$. Also, note that $\boldsymbol{C}\boldsymbol{C}^\dagger\boldsymbol{A} = \boldsymbol{U}_C\boldsymbol{U}_C^T\boldsymbol{A}$ is the projection of $\boldsymbol{A}$ onto the column space of $\boldsymbol{C}$, and $\boldsymbol{A}\boldsymbol{R}^\dagger\boldsymbol{R} = \boldsymbol{A}\boldsymbol{V}_{R,k}\boldsymbol{V}_{R,k}^T$ is the projection of $\boldsymbol{A}$ onto the row space of $\boldsymbol{R}$.

The Frobenius norm and spectral norm of a matrix are denoted by $\|\boldsymbol{A}\|_F$ and $\|\boldsymbol{A}\|_2$ respectively. The square of the Frobenius norm is given by $\|\boldsymbol{A}\|_F^2 = \sum_{i=1}^m \sum_{j=1}^n A_{i,j}^2$ $= \sum_{i=1}^k \sigma_i^2(\boldsymbol{A})$. The spectral norm is given by $\|\boldsymbol{A}\|_2 = \max_i \sigma_i(\boldsymbol{A})$.

2.2 The CUR problem and other related work

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 [5]. We re-state it here for the sake of completion and setting the appropriate context for our results to be stated in the next section. This relative error bound result is summarized in the following theorem.

Theorem 1. *(Theorem 2 from [5] 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:*

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

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 $\boldsymbol{e}_j$ is the j-th column of identity matrix which 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 [5] 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 $\boldsymbol{A}$; therefore, this method of samping is also known as *subspace* sampling. By sampling columns that lie in this subspace more often, we get a relative-error low rank approximation of the matrix. 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 [4] and was refined in [5] to obtain performance error guarantees for CUR matrix decomposition.

These guarantees were subsequently improved in follow-up work [11]. Modified versions of this problem have been studied extensively for adaptive sampling [16], divide-and-conquer algorithms for parallel computations [10], and input-sparsity algorithms [2]. The authors of [16] propose an adaptive sampling-based algorithm which requires only $c = O(k/\varepsilon)$ columns to be sampled when the entire matrix is known and its SVD can be computed. The authors of [2] also proposed an optimal, deterministic CUR algorithm. In [1], 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.

These prior results require sampling of arbitrary rows and columns of the matrix $\boldsymbol{A}$ which may be either unrealistic or inefficient in many practical applications. 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 more natural environment of block sampling for biometric and distributed computation, explore the performance advantages of block sampling over individual column sampling, and provide the first non-trivial theoretical error guarantees for Block CUR decomposition. In the following section, we propose and analyze a randomized algorithm for sampling blocks of the matrix based on *block leverage scores*.

3 The Block CUR algorithm

A block may be defined as a collection of s columns or rows. For clarity of exposition, without loss of generality, we consider column blocks but the techniques and derivations also hold for row blocks by applying them to the transpose of the matrix. For ease of exposition, we also assume equal-sized blocks but one could easily extend the methods to blocks of varying sizes. Let $G = \lceil n/s \rceil$ be the number of 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.

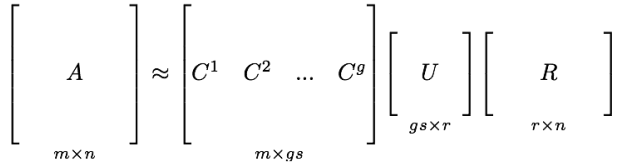


Fig. 2: Example Block CUR decomposition, where $\boldsymbol{C}^t \in \mathbb{R}^{m\times s}$ for $t \in [g]$ is sampled from $\{\boldsymbol{A}^{(j_t)} : j_t \in [G]\}$.

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 2. For example, in the biometric analysis setting each block could correspond to user reactions at a collection of time points corresponding to a scene in a movie. The goal is to approximate the users' reactions to the full movie using only their response to a summary of the movie (containing a subset of the scenes). Given the new regime of submatrix blocks, 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 $\boldsymbol{E}_g$ consists of the corresponding block of columns in the identity matrix which 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 [5], the block leverage scores measure how much a particular column block contributes to the approximation of the matrix $\boldsymbol{A}$.

3.1 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 $\boldsymbol{A}$.

In Algorithm 1, for $t \in [g]$, block $j_t \in [G]$ is sampled with some probability p_{j_t} and scaled using matrix $\boldsymbol{S} \in \mathbb{R}^{n\times gs}$. The (j_t, t)-th non-zero $s \times s$ block of $\boldsymbol{S}$ is defined as $\boldsymbol{S}_{j_t,t} = \boldsymbol{I}_s/\sqrt{gp_{j_t}}$ where $g = c/s$ is the number of blocks picked by the algorithm. This sampling matrix picks the blocks of columns and scales each block to compute $\boldsymbol{C} = \boldsymbol{A}\boldsymbol{S}$. A similar sampling and scaling matrix $\boldsymbol{S}_R$ is defined to pick the blocks of rows and scale each block to compute $\boldsymbol{R} = \boldsymbol{S}_R^T\boldsymbol{A}$. An example of sampling matrix $\boldsymbol{S}$ with blocks chosen in order $[1, 3, 2]$ is as follows:

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

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 $t \in [g]$, select a block of columns $j_t \in [G]$ independently with probability $p_{j_t} = \frac{\ell_i(\boldsymbol{R},r)}{r} = \frac{\|\boldsymbol{V}_{R,r}^T \boldsymbol{E}_i\|_F^2}{r}$ 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 addition to considering block sampling of columns, another advantage of this algorithm is not requiring the computation of a full SVD of $\boldsymbol{A}$. In many large-scale 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 (corresponding to users) 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 is not the focus of the experiments in this paper so it can also be replaced with other fast approximate calculations of leverage scores involving sketching or additional sampling [3,17]. 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. In terms of the biometric application, each row corresponds to a user's biometric reaction to a movie. Since collecting user reactions to lengthy content can be expensive, eliminating redundant sampling leads to huge savings in resources.

The running time of Algorithm 1 is essentially driven by the time required to compute the SVD of $\boldsymbol{R}$, *i.e.,* $\mathcal{O}(SVD(\boldsymbol{R}))$ time, and the time to construct $\boldsymbol{R}$, $\boldsymbol{C}$ and $\boldsymbol{U}$. 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 $\mathcal{O}(r^2 c)$ time.

3.2 Theoretical results and discussion

The main technical contribution of the paper is a novel relative-error bound on the quality of approximation using blocks of columns or rows to approximate a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$. Before stating the main result, we define two important quantities that measure important properties of the matrix $\boldsymbol{A}$ that are fundamental to the quality of approximation. We first define a property of matrix rank relative to the collection of matrix blocks. Specifically, we focus on the concept of *matrix stable rank* from [14] and 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*

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

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

Intuitively, the above definition gives a measure of how informative the worst matrix column block is. The second property is a notion of *column space incoherence*. When we sample rows uniformly at random, we can give relative error approximation guarantees when the matrix $\boldsymbol{A}$ satisfies an incoherence condition. 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*

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

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 fast approximation without computing the SVD of the entire matrix $\boldsymbol{A}$. Equipped with these definitions, we state the main result that provides a relative-error guarantee for the Block CUR approximation in Theorem 2.

Theorem 2. *Given $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ with incoherent top-k column space, i.e., $\mu \leq \mu_0$, let $r = O\left(\mu_0 \frac{k^2}{\varepsilon^2} \ln(\frac{1}{\delta})\right)$ and $g = O\left(\frac{r^2}{\alpha_R \varepsilon^2} \ln(\frac{1}{\delta})\right)$. 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:*

$$\|\boldsymbol{A} - \boldsymbol{C}\boldsymbol{U}\boldsymbol{R}\|_F \leq (1 + \varepsilon)\|\boldsymbol{A} - \boldsymbol{A}_k\|_F,$$

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

We provide a sketch of the proof and highlight the main technical challenges in proving the claim in Section 3.3 and defer the proof details to an extended version of the paper on arXiv [12]. In Section 3.3, we first provide a relative-error guarantee (Lemma 2) for the approximation provided by Algorithm 1. After applying standard boosting techniques (explained in Section 3.3) we get the main result stated above.

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

Block stable rank. Theorem 2 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

Table 1: Table comparing the sample complexity needed for given ε using our Block CUR result and a bound obtained by trivial extension of traditional CUR. For ease of comparison, we show the results with full SVD computation ignoring incoherence assumption stated in Corollary 1 in the [12]. The α_R term we introduce satisfies the bound $1 \leq \alpha_R \leq s$.

Results	r	g
Traditional CUR extended to block setting	$\mathcal{O}\left(\frac{k^2}{\varepsilon^2}\log\left(\frac{1}{\delta}\right)\right)$	$\mathcal{O}\left(\frac{k^4}{\varepsilon^6}\log^3(\frac{1}{\delta})\right)$
Our Block CUR	$\mathcal{O}\left(\frac{k^2}{\varepsilon^2}\log(\frac{1}{\delta})\right)$	$\mathcal{O}\left(\frac{k^4}{\boldsymbol{\alpha_R}\varepsilon^6}\log^3(\frac{1}{\delta})\right)$

stable relaxation of exact rank) of the blocks. The stable rank $\alpha = \|\boldsymbol{A}\|_F^2/\|\boldsymbol{A}\|_2^2$ is a relaxation of the rank of the matrix; in fact, it is stable under small perturbations of the matrix $\boldsymbol{A}$ [14]. 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 $1 \leq \alpha_R \leq s$. In the best case, when all the groups have full stable rank with equal singular values, α_R achieves its 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 total sampling complexity.

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, independent of incoherence, for the full SVD Block CUR approximation is stated as Corollary 1 in the extended version of the paper [12]. The corollary follows by a similar analysis as Theorem 2 so we defer the proof to the extended version [12]. Other than block sampling, the setup of this result is equivalent to the traditional column sampling result stated in Lemma 2. Next, we compare the block sampling result with extensions of traditional column sampling.

Sample complexity: comparison with extensions of traditional CUR results. In order to compare the sample complexity of our block sampling results with trivial block extensions of traditional column sampling results we focus our attention on the similar leverage score based CUR result in Theorem 1. A simple extension to block setting could be obtained by considering a larger row space in which blocks are expanded to vectors. This would lead to a sample complexity bound obtained by Theorem 1. The sampling complexity of the Block CUR derived in Theorem 2 tells us the number of sampling operations or queries that need to be made to memory in order to construct the $\boldsymbol{R}$ and $\boldsymbol{C}$ matrices. As shown in Table 1 the column block sample complexity obtained by traditional CUR extensions results is always greater than or equal to those required by our Block CUR result because $1 \leq \alpha_R \leq s$. This happens since traditional CUR-based results are obtained by completely ignoring the block structure of the matrix.

As a side note, the authors are aware that more recent adaptive column sampling-based algorithms such as [16,2] require only $c = O(k/\varepsilon)$ columns to be sampled. These results assume full computation of the SVD is possible, and they are byproducts

of heavy machinery using ideas like deterministic, Batson/Srivastava/Spielman (BSS) sampling and adaptive sampling on top of leverage scores. By extending these advanced techniques to block sampling, it may be possible to obtain tighter bounds but it does not bring new insight into the problem of sampling blocks and unnecessarily complicates the discussion. Therefore we defer this extension to future work.

3.3 Proof sketch of main result

In this section, we provide a sketch of the proof of Theorem 2 and defer the details to the extended version [12]. The proof of the main result rests on two important lemmas. These results are important in their own right and could be useful wherever the block sampling issue arises. The first result concerns approximate block multiplication.

Block multiplication lemma. The following lemma 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 1. *Let $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times p}$,$\varepsilon, \delta \in (0,1)$, and α_A be defined as $\alpha_A := \min_{i\in[G]} \frac{\|\boldsymbol{A}^{(i)}\|_F^2}{\|\boldsymbol{A}^{(i)}\|_2^2}$. Construct $\boldsymbol{C}_{m\times gs}$ and $\boldsymbol{R}_{gs\times n}$ using sampling probabilities p_i that satisfy*

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

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

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

The proof details are provided in the extended version of the paper [12]. The main difficulty in proving this claim is to account for the block structure. Even though one could trivially extend individual column sampling analysis to this setting by serializing the blocks, this would lead to trivial bounds as they do not leverage the block structure. Our results exploit this knowledge and hence introduce a dependence of the sample complexity on the block stable rank of the matrix.

Using the block multiplication lemma we prove Lemma 2, which states a non-boosting approximation error result for Algorithm 1.

Lemma 2. *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 $\varepsilon \in (0,1)$, $\boldsymbol{U} = \boldsymbol{W}^\dagger$ is the pseudoinverse of the scaled intersection of $\boldsymbol{C}$ and $\boldsymbol{R}$.

The proof of Lemma 2 follows standard techniques in [5] with modifications necessary for block sampling (see [12] for the proof details). Finally, the result in Theorem 2 follows by applying standard boosting methods to Lemma 2 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 $1 - e^{-t} = 1 - \delta$.

Remark. As a consequence of Lemma 2, we show that if enough blocks are sampled with high probability, then $\|\boldsymbol{A} - \boldsymbol{AS}(\boldsymbol{RS})^{\dagger}\boldsymbol{R}\|_F \leq (1+\varepsilon)\|\boldsymbol{A} - \boldsymbol{AR}^{\dagger}\boldsymbol{R}\|_F$. 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. 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 $\|\boldsymbol{A} - \boldsymbol{CC}^{\dagger}\boldsymbol{A}\|_F \leq (1 + \varepsilon)\|\boldsymbol{A} - \boldsymbol{A}_k\|_F$.

4 Experiments

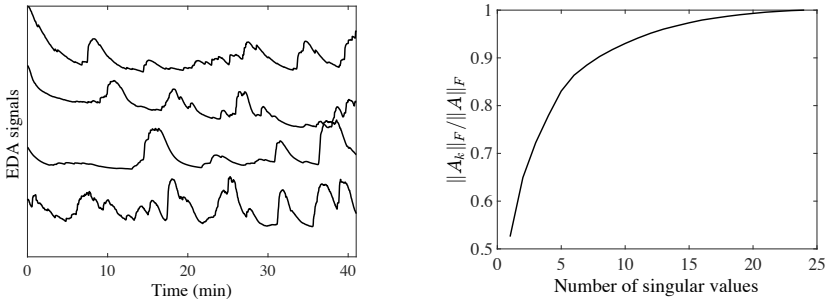
4.1 Experiments with biometric data

One emerging application is audience reaction analysis of video content using biometrics. Specifically, users watch video content while wearing sensors, with changes in biometric sensors indicating changes in reaction to the content. For example, increases in heart rate or a spike in electrodermal activity indicate an increase in content engagement. In prior work, biometric signal analysis techniques have been developed to determine valence [15] (*e.g.,* positive vs. negative reactions to films) and content segmentation [9]. Unfortunately these experiments require a large number of users to sit through the entire video content, which can be both costly and time-consuming.

We consider the observed biometric signals as a matrix with m users (as rows) and n biometric time samples (as columns). Matrix approximation techniques, such as CUR decomposition, point to the ability to infer the complete matrix by showing the entire content to only a subset of users (*i.e.,* rows), while the remaining users see only selected scenes of the content (*i.e.,* column blocks). To replicate a user's true reaction to content, individual columns cannot be sampled (*e.g.,* showing the user 0.25 seconds of video content) given the lack of scene context. Instead, longer scenes must be shown to the user to gather a representative response. Therefore, the Block CUR decomposition proposed in this paper is directly applicable.

The biometric experiment setup is as follows. We attached 24 subjects with the Empatica E3 wearable sensor [6] that measures electro-dermal activity (EDA) at 4 Hz. The subjects were shown a 41-minute episode of the television series "NCIS", in the genres of action and crime. The resulting biometric data matrix was 24×9929. Our goal is to use Block CUR decomposition to show only a subset of users the entire content, and to then impute the biometric data for users that have viewed only a small number of selected scenes from the content.

Results. We refer to the biometric data matrix as $\boldsymbol{A}$ and plot the EDA traces (rows) corresponding to four users in Figure 3a. To demonstrate the low rank nature of the data, we plot the Frobenius norm of $\boldsymbol{A}$ covered by $\boldsymbol{A}_k$ as a function of k in Figure 3b. We find that for this data, only 5 singular vectors are needed to capture 80% of the total Frobenius norm of the complete matrix. Next, we segment the columns of this matrix into blocks such that $s = 120$ columns (or 30 seconds). In Figure 4, we show



(a) **(b)**

Fig. 3: Panel (a) shows EDA data for four users watching the NCIS video and (b) demonstrates the low rank nature of $\boldsymbol{A}$

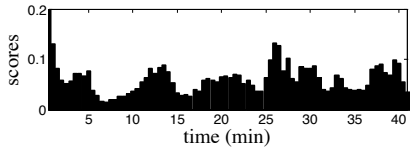
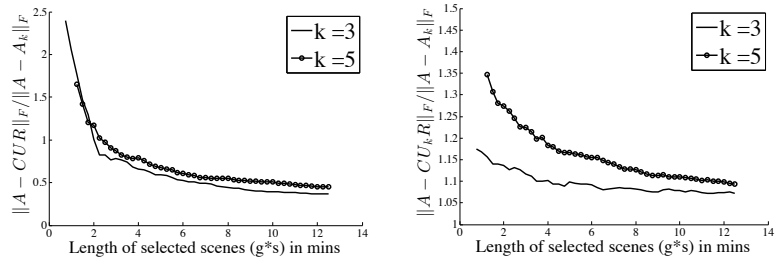
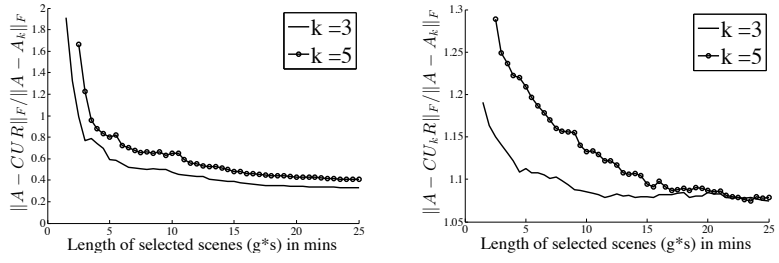


Fig. 4: Block leverage scores for EDA data with k = 5 and s = 120 columns (30 seconds).



(a) $s = 60$ using $\boldsymbol{U}$ **(b)** $s = 60$ using $\boldsymbol{U}_k$



(c) $s = 120$ using $\boldsymbol{U}$ **(d)** $s = 120$ using $\boldsymbol{U}_k$

Fig. 5: Error plots for two values of target rank, $k = 3, 5$.

the computed block leverage scores. The leverage scores seem to suggest that certain scenes are more important than others. For example, the highest leverage scores are around the 12, 26, and 38 minute marks. This corresponds to scenes of a dead body, unveiling of a clue to solving the mystery, and the final arrest, respectively.

Using Algorithm 1, we uniformly sample EDA traces (rows) of 20 users and hold out the EDA traces of 4 users. We sample column blocks and plot the resulting error in Frobenius norm in Figure 5. The plots show the normalized Frobenius norm error of the CUR approximation as a function of the number of blocks, g, sampled. More precisely, the ratio $\|\boldsymbol{A} - \boldsymbol{CUR}\|_F / \|\boldsymbol{A} - \boldsymbol{A}_k\|_F$ and $\|\boldsymbol{A} - \boldsymbol{CU}_k\boldsymbol{R}\|_F / \|\boldsymbol{A} - \boldsymbol{A}_k\|_F$ are plotted for two values of the target rank, $k = 3$ and 5 and two values of block size, $s = 60$ and 120 columns per block (15 and 30 seconds), respectively. We also compare the error using $\boldsymbol{U}_k$, the rank-k approximation of $\boldsymbol{U}$, which leads to an exactly rank-k matrix approximation since this may be a restriction in some applications. We repeat Algorithm 1 ten times[5] and plot the mean normalized error over 10 trials.

The error drops sharply as we sample more blocks but quickly flattens demonstrating that a summary of the movie could suffice to approximate the full responses. The plots also show the interplay between the number of blocks sampled and the issue of context which is related to block size. To give the viewer some context we would want to make the scene as long as possible but we want to show them only a summary of the content to reduce the cost. These conflicting aims result in a trade-off of block size and the number of blocks sampled. For example, for $k = 5$, the normalized error is less than 1 when a 2.5 minute long clip is shown to the viewer, that is $g = 10$ with block size $s = 60$ columns (or 15 seconds), whereas the normalized error is less than 1 when a 3.5 minute long clip is shown to the viewer ($g = 7$) with block size $s = 120$ columns (or 30 seconds). These results demonstrate the practical use of the Block CUR algorithm.

4.2 Distributed experiments

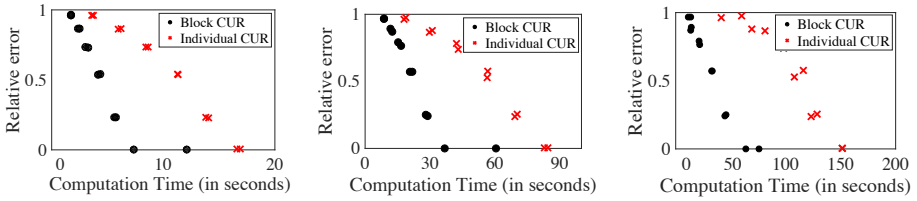
In this section we demonstrate empirically that the proposed block sampling based CUR algorithms can achieve a significant speed-up when used to decompose matrices in a distributed data setting by comparing their performance with individual column sampling based traditional CUR algorithms on both synthetic and real-world data. We report the relative-error of the decomposition (*i.e.,* $\|\boldsymbol{A} - \boldsymbol{CUR}\|_F / \|\boldsymbol{A}\|_F$) and the sampling 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 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 2). 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 as shown in Table

[5] These plots were generated using *sampling without replacement* even though our theory supports *sampling with replacement* since sampling the same blocks is inefficient in practice.

Table 2: Table comparing the number of sampling operations needed for given ε using our Block CUR result based on block sampling and traditional CUR based on individual column sampling (note this is not the same as the vectorized block columns in Table 1). This leads to speedup since it is more efficient to retrieve predefined blocks than querying individual rows or columns in these regimes. The α_R term we introduce satisfies the bound $1 \leq \alpha_R \leq s$.

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



(a) $n = 8000$ **(b)** $n = 12000$ **(c)** $n = 20000$

Fig. 6: Performance on synthetic $n \times n$ matrices with rank $n/10$.

2. Our experiments examine the runtime speed-up from our block sampling CUR that exploits the partitioning of data.

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.

Figure 6 shows the plots for relative error achieved with respect to the runtime required to sample $\boldsymbol{C}$ and $\boldsymbol{R}$ matrices for both Block CUR and traditional CUR algorithms. To focus on the speed-up achieved by taking into account the block storage of data we compare running times 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 leverage scores by sketching or sampling [3]. 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 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 6. While the same amount of data is being

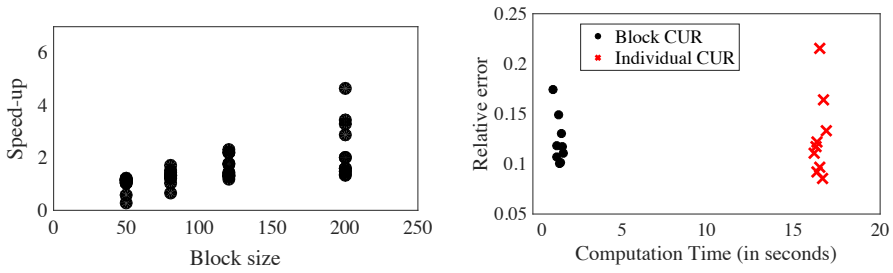


Fig. 7: Performance on $900 \times 10,000$ Arcene dataset with block size 12. (a) Runtime speed-up from block sampling compared to individual column sampling for varying block sizes. (b) Block CUR achieves similar relative errors as individual CUR with much lower computation time.

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 than 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.

Real-world experiments. We also conduct experiments on the Arcene dataset [7] which has 900 rows and 10,000 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 11.2 over individual column sampling, as shown in Figure 7. The matrix is very low rank, and sampling a few groups gave small relative errors.

5 Conclusion

In this paper we extended the problem of CUR matrix decomposition to the block setting which is naturally relevant to distributed storage systems and biometric data analysis. We proposed a novel algorithm and derived its performance bounds. We demonstrated its practical utility on real-world distributed storage systems and audience analytics. 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. Boutsidis, C., Drineas, P., Magdon-Ismail, M.: Near-optimal column-based matrix reconstruction. SIAM Journal on Computing **43**(2), 687–717 (2014)

2. Boutsidis, C., Woodruff, D.P.: Optimal CUR matrix decompositions. In: Proceedings of the 46th Annual ACM Symposium on Theory of Computing. pp. 353–362. ACM (2014)
3. Drineas, P., Magdon-Ismail, M., Mahoney, M.W., Woodruff, D.P.: Fast approximation of matrix coherence and statistical leverage. Journal of Machine Learning Research **13**(Dec), 3475–3506 (2012)
4. Drineas, P., Mahoney, M.W., Muthukrishnan, S.: Sampling algorithms for l_2 regression and applications. In: Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 1127–1136. Society for Industrial and Applied Mathematics (2006)
5. Drineas, P., Mahoney, M.W., Muthukrishnan, S.: Relative-error CUR matrix decompositions. SIAM Journal on Matrix Analysis and Applications **30**(2), 844–881 (2008)
6. Garbarino, M., Lai, M., Bender, D., Picard, R., Tognetti, S.: Empatica E3—A wearable wireless multi-sensor device for real-time computerized biofeedback and data acquisition. In: Proceedings of the 4th International Conference on Wireless Mobile Communication and Healthcare. pp. 39–42 (November 2014)
7. Guyon, I., Gunn, S.R., Ben-Hur, A., Dror, G.: Result analysis of the NIPS 2003 feature selection challenge. In: Advances in Neural Information Processing Systems. pp. 545–552 (2004)
8. Jain, S., Oswal, U., Xu, K.S., Eriksson, B., Haupt, J.: A compressed sensing based decomposition of electrodermal activity signals. IEEE Transactions on Biomedical Engineering **64**(9), 2142–2151 (2017)
9. Lian, W., Rao, V., Eriksson, B., Carin, L.: Modeling correlated arrival events with latent semi-markov processes. In: Proceedings of the International Conference on Machine Learning (2014)
10. Mackey, L.W., Jordan, M.I., Talwalkar, A.: Divide-and-conquer matrix factorization. In: Advances in Neural Information Processing Systems. pp. 1134–1142 (2011)
11. Mahoney, M.W., Drineas, P.: CUR matrix decompositions for improved data analysis. Proceedings of the National Academy of Sciences **106**(3), 697–702 (2009)
12. Oswal, U., Jain, S., Xu, K.S., Eriksson, B.: Block CUR: Decomposing large distributed matrices. arXiv preprint arXiv:1703.06065 (2017)
13. Paschou, P., Ziv, E., Burchard, E.G., Choudhry, S., Rodriguez-Cintron, W., Mahoney, M.W., Drineas, P.: PCA-correlated SNPs for structure identification in worldwide human populations. PLoS Genetics **3**(9), e160 (2007)
14. Rudelson, M., Vershynin, R.: Sampling from large matrices: An approach through geometric functional analysis. Journal of the ACM **54**(4), 21 (2007)
15. Silveira, F., Eriksson, B., Sheth, A., Sheppard, A.: Predicting audience responses to movie content from electro-dermal activity signals. In: Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing. pp. 707–716. ACM (2013)
16. Wang, S., Zhang, Z.: A scalable CUR matrix decomposition algorithm: lower time complexity and tighter bound. In: Advances in Neural Information Processing Systems. pp. 647–655 (2012)
17. Xu, M., Jin, R., Zhou, Z.H.: CUR algorithm for partially observed matrices. In: Proceedings of the International Conference on Machine Learning. pp. 1412–1421 (2015)
18. Yang, J., Rübel, O., Mahoney, M.W., Bowen, B.P.: Identifying important ions and positions in mass spectrometry imaging data using cur matrix decompositions. Analytical Chemistry **87**(9), 4658–4666 (2015)
19. Yip, C.W., Mahoney, M.W., Szalay, A.S., Csabai, I., Budavári, T., Wyse, R.F., Dobos, L.: Objective identification of informative wavelength regions in galaxy spectra. The Astronomical Journal **147**(5), 110 (2014)