Randomization or Condensation? Linear-Cost Matrix Sketching Via Cascaded Compression Sampling

Kai Zhang*
Department of Computer and Information Sciences
Temple University, Philadelphia PA
kzhang980@gmail.com

Chuanren Liu*
Decision Sciences and MIS
Drexel University
Philadelphia, PA
chuanren.liu@drexel.edu

Jie Zhang*
Institute of Science and Technology for Brain-Inspired Intellegence
Fudan University, Shanghai
jzhang080@gmail.com

Hui Xiong
Management Science and Information Systems Department, Rutgers
1 Washington Park, Newark NJ
hxiong@rutgers.edu

Eric Xing
Machine Learning Department
Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, PA
epxing@cs.cmu.edu

Jieping Ye
Department of Computational Medicine and Bioinformatics
University of Michigan, Ann Arbor
jpye@umich.edu

ABSTRACT

Matrix sketching is aimed at finding compact representations of a matrix while simultaneously preserving most of its properties, which is a fundamental building block in modern scientific computing. Randomized algorithms represent state-of-the-art and have attracted huge interest from the fields of machine learning, data mining, and theoretic computer science. However, it still requires the use of the entire input matrix in producing desired factorizations, which can be a major computational and memory bottleneck in truly large problems. In this paper, we uncover an interesting theoretic connection between matrix low-rank decomposition and *lossy signal compression*, based on which a cascaded compression sampling framework is devised to approximate an $m \times n$ matrix in only $O(m+n)$ time and space. Indeed, the proposed method accesses only a small number of matrix rows and columns, which significantly improves the memory footprint. Meanwhile, by sequentially teaming two rounds of approximation procedures and upgrading the sampling strategy from a uniform probability to more sophisticated, encoding-orientated sampling, significant algorithmic boosting is achieved to uncover more granular structures in the data. Empirical results on a wide spectrum of real-world, large-scale matrices show that by taking only linear time and space, the accuracy of our method rivals those state-of-the-art randomized algorithms consuming a quadratic, $O(mn)$, amount of resources.

KEYWORDS

Randomized algorithms, Matrix sketching, Low-rank decomposition, Cascaded compression sampling

* Equal contribution.

KDD '17, August 13-17, 2017, Halifax, NS, Canada

978-1-4503-4887-4/17/08…$15.00
DOI: 10.1145/3097983.3098050

1 INTRODUCTION

Matrix arises in many modern disciplines as a popular tool for data representation and modelling. Identifying structures of the matrix allows researchers to uncover important patterns in the data, which in turn translates into actionable decisions or scientific discoveries. Matrix low rank decomposition [15] is one such structure-finding technique. It is aimed at approximating an input matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ by $\mathbf{A} \approx \mathbf{P}\mathbf{Q}^\top$, where $\mathbf{P}$ and $\mathbf{Q}$ have a low column rank $k \ll m, n$. Such reduced representation can provide a universal computational platform to solve problems in various scientific computing and data analysis applications, with important applications in optimization, machine learning, information retrieval, computer vision, and bioinformatics [1, 9, 12, 13, 17, 19, 20, 22, 24, 25, 33].

Recent advances in randomized algorithms have made it state-of-the-art in solving low-rank approximation problems. These algorithms fully exploit stochastic properties of random sampling and projection to extract key structures from the data. Abundant literature exists, with examples including monte-carlo sampling [10, 14], adaptive sampling [3], CUR matrix decomposition [5, 25], random projection method [7, 11, 17], sparse embedding and fast transforms [8, 27]. These algorithms are computationally more efficient than singular value decomposition, the latter taking $O(n^2 m)$ time if $m \geq n$. In the meantime, important probabilistic error bounds are provided that fully characterize the behaviour of various sampling/randomization schemes. See reviews in [17, 24].

With unprecedented amount of electronic data acquired in modern scientific and engineering applications, however, existing randomized methods can still be computationally demanding on large data. In particular, these algorithms need to fully access and manipulate the entire input matrix, either in identifying an informative subspace for subsequent projection [17], or in computing high-quality sampling probabilities [10, 14, 25]. This necessitates $O(mn)$ time and space for an $m \times n$ dense matrix. Yet in case of exceptionally high dimensions, storing matrix alone already becomes onerous, not to mention complicated operations on top of it.

In this paper, we aim to combat the computational and memory bottlenecks of matrix low-rank decomposition on large data. Our main philosophy has a notable difference compared with existing randomized algorithms. In these algorithms, extensive theoretic

studies have been performed on probabilistic error bounds that various randomization schemes can achieve when sampling a desired volume of data, often taking $O(mn)$ time and space. In comparison, we consider the scenario that available computing resource (time and memory) is only linear for an $m \times n$ input matrix, i.e., $O(m+n)$. Under such condition, our goal is to fully optimize the sampling scheme subject to available computing resources, so as to make the best of what is given and maximally reduce approximation error.

We initiate this new problem setting because in big data applications, achieving a desired sampling rate and manipulating the entire input matrix (as is often needed in existing randomized algorithms) can be intractable. A linear-cost constraint, on the other hand, is much easier to accomplish. Of course, such a constraint poses significant challenges, because it only allows accessing a small, constant number of rows/columns of the data; also, a linear amount of resources hardly affords any advanced sampling scheme other than uniform; finally, approximation of general rectangular matrices with linear cost is much harder than for positive semi-definite (PSD) matrices that lie on a Riemannian manifold [12, 13, 30].

To solve these problems, we propose a new computational framework called cascaded compression sampling. Our key innovation is an interesting, theoretic connection between the quality of low-rank approximation and the encoding powers of sampling. This allows us to study low-rank approximation from a novel, lossy data compression point of view, where the choice of good sampling schemes can thus benefit naturally from the design of the code-books. In particular, we show that sampled columns/rows should correspond to representative code-vectors in the low-rank embeddings of the matrix. Motivated by this, we employ cascaded sampling procedures that upgrade from simple, random sampling to more sophisticated, compression-based sampling, so as to progressively uncover granular structures in the data. Theoretically, the rise of approximation quality is proven to be lower-bounded by the improvement of encoding powers through the compression sampling, thus guaranteeing the algorithmic boosting property. Computationally, both rounds of operations take only linear time and space, thus more efficient on large data. The performance of our approach rivals the quality of state-of-the-art randomized algorithms while being one to two orders of magnitude faster with only a small fraction of their memory costs. The performance gains are more significant on large, sparse matrices.

The rest of the paper is organized as follows. Section 2 reviews existing work. Section 3 presents our key theorem linking matrix low-rank approximation and lossy singal compression. Section 4 introduces the cascaded compression sampling framework, and proves its algorithmic boosting property. Section 5 reports experimental results, and the last section concludes the paper.

2 RELATED WORK

2.1 Quadratic-Cost Algorithms

We first review state-of-the-art randomized algorithms that take a quadratic, $O(mn)$ amount of time and space for an $m \times n$ matrix.

Monte-Carlo sampling method [10, 14] approximates an input matrix $\mathbf{A}$ by selecting a subset of columns with data-dependent probabilities p_j's. The rank-k basis of selected columns (often re-scaled by p_j's) $\mathbf{Q}_k$ is then used for final decomposition. If the probabilities satisfy $p_j \geq \beta \cdot \|\mathbf{A}_{[:,j]}\|^2 / (\sum_{i=1}^{n} \|\mathbf{A}_{[:,i]}\|^2)$, then with probability at least $1-\delta$, one has $\|\mathbf{A} - \mathbf{Q}_k\mathbf{Q}_k^\top\mathbf{A}\|_F^2 \leq \|\mathbf{A} - \mathbf{A}_k\|_F^2 + \epsilon\|\mathbf{A}\|_F^2$, where $\mathbf{A}_k$ is the best rank-k approximation, $\epsilon = 2\eta/\sqrt{k/\beta c}$, with $\beta \leq 1, \eta = 1 + \sqrt{(8/\beta)\log(1/\delta)}$.

Random projection methods [17] project $\mathbf{A}$ using $\mathbf{Q} \in \mathbf{R}^{m\times k}$ with orthonormal columns such that $\mathbf{A} \approx \mathbf{Q}\mathbf{Q}^\top\mathbf{A}$. Computing $\mathbf{Q}$ requires multiplying $\mathbf{A}$ with a Gaussian test matrix $\mathbf{\Omega}$ with q steps of power iterations, $\mathbf{Y} = (\mathbf{A}\mathbf{A}^\top)^q\mathbf{A}\mathbf{\Omega}$, and then a QR-decomposition $\mathbf{Y} = \mathbf{Q}\mathbf{R}$. Using a template $\mathbf{\Omega} \in \mathbb{R}^{m\times(k+p)}$ with over-sampling parameter p, $\mathbb{E}\left\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^\top\mathbf{A}\right\| = [1 + 4\sqrt{(k+p)\cdot\min(m,n)}/(p-1)]\Sigma_{k+1}$, where $\mathbb{E}$ is expectation with $\mathbf{\Omega}$, Σ_{k+1} is the $(k+1)$th singular value. New projection methods are explored in [7, 11].

Using structured templates (sub-sampled randomized Hard-amard transform (SRHT) [4] or random Fourier transform [17]), the complexity of random projection reduces from $O(mnk)$ $O(mn\log k)$. More recently, sparse embedding is introduced for low-rank decomposition in input sparsity time [8], which allows approximation $\|\mathbf{A} - \hat{\mathbf{A}}\|_F = (1 \pm \epsilon)\|\mathbf{A} - \mathbf{A}_k\|_F$ with high probability in $O(nnz(\mathbf{A})) + O(nk^2/\epsilon + k^3/\epsilon^5)$ time. Here $nnz(\mathbf{A})$ is the number of non-zeros. It can be more efficient for sparse input matrices.

CUR matrix decomposition was proposed in [25] to improve the interpretability of low-rank matrix decomposition. As shown in Figure 1(a), it samples a subset of columns and rows from $\mathbf{A}$, as $\mathbf{C} \in \mathbb{R}^{m\times k_c}$ and $\mathbf{R} \in \mathbb{R}^{k_r\times n}$, and then solve

$$\min_{\mathbf{U}} \|\mathbf{A} - \mathbf{C}\mathbf{U}\mathbf{R}\|_F^2 \rightarrow \mathbf{U}^* = \mathbf{C}^\dagger\mathbf{A}\mathbf{R}^\dagger, \tag{1}$$

where † is pseudo-inverse. If leverage scores are used for sampling, then with high probability, $\|\mathbf{A} - \mathbf{C}\mathbf{U}\mathbf{R}\|_F \leq (2+\epsilon)\|\mathbf{A} - \mathbf{A}_k\|_F$ [25]. The leverage scores are computed by top-r singular vectors, taking $O(mn)$ space and $O(mnr)$ time. Computing $\mathbf{U}$ in (1) involves multiplying $\mathbf{A}$ with $\mathbf{C}^\dagger$ and $\mathbf{R}^\dagger$.

2.2 Linear-Cost Algorithms

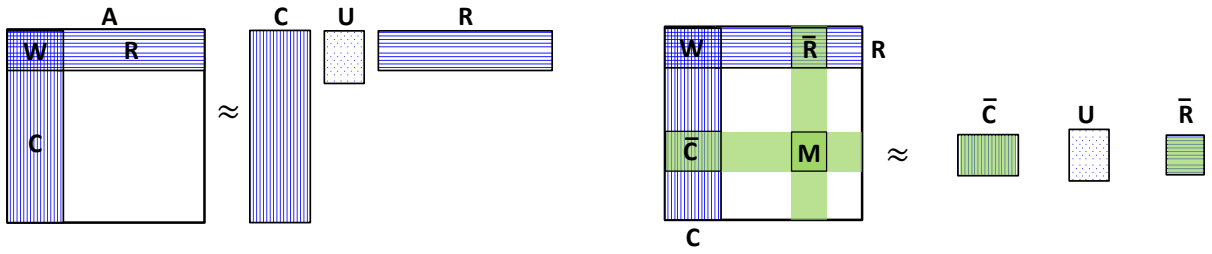
The randomized algorithms discussed in Section 2.1 take at least quadratic time and space. One way to reduce the cost is to restrict the calculations to only a small fraction of rows and columns. We use a general variant of CUR, called *Bilateral Re-sampled CUR* (BR-CUR), to initiate discussion.

Algorithm 1: Bilateral Resampled CUR (BR-CUR)

Input: $\mathbf{A}$, base sampling $\mathcal{I}_r^b, \mathcal{I}_c^b$, target sampling $\mathcal{I}_r^t, \mathcal{I}_c^t$;
Output: $\mathbf{C}$, $\mathbf{U}$, $\mathbf{R}$

1: Compute base rows and columns $\mathbf{C} = \mathbf{A}_{[:,\mathcal{I}_c^b]}$, $\mathbf{R} = \mathbf{A}_{[\mathcal{I}_r^b,:]}$.
2: Sample on base rows and columns $\bar{\mathbf{C}} = \mathbf{C}_{[\mathcal{I}_r^t,:]}$, $\bar{\mathbf{R}} = \mathbf{R}_{[:,\mathcal{I}_c^t]}$.
3: Compute target block matrix $\mathbf{M} = \mathbf{A}_{[\mathcal{I}_r^t,\mathcal{I}_c^t]}$.
4: Solve intersection matrix $\mathbf{U}^* = \arg\min_{\mathbf{U}} \|\mathbf{M} - \bar{\mathbf{C}}\mathbf{U}\bar{\mathbf{R}}\|_F^2$.
5: Reconstruct the input matrix by $\mathbf{A} \approx \mathbf{C}\mathbf{U}^*\mathbf{R}$.

As illustrated in Figure 1(b), BR-CUR has two rounds of samplings: blue "base"-sampling $\mathcal{I}_r^b$ (row) and $\mathcal{I}_c^b$ (column) to construct $\mathbf{C}$ and $\mathbf{R}$, and green "target"-sampling $\mathcal{I}_r^t$ (row) and $\mathcal{I}_c^t$ (column) to specify a sub-matrix from $\mathbf{A}$. In computing $\mathbf{U}$ (step 4), it only minimizes the error on the target sub-matrix, therefore computationally quite efficient. Let k_1 and k_2 be the number of samples



(a) Illustration of CUR method. (b) Illustration of Re-sampled CUR.

Figure 1: Graphical illustration of the CUR matrix decomposition and the bilateral re-sampled CUR method.

selected in the base and target sampling. Then BR-CUR only takes $O((m+n)(k_1+k_2))$ space and $O\left((m+n)\max(k_1,k_2)^2\right)$ time.

The BR-CUR procedure subsumes most linear-cost algorithms for matrix low-rank decomposition in the literature.

1. Nyström method [12, 13, 30]: for PSD matrix $\mathbf{A}$, upon using the same base and target sampling, and the same row and column sampling, i.e., $\mathcal{I}_c^b = \mathcal{I}_r^b = \mathcal{I}_c^t = \mathcal{I}_c^t$.

2. Pseudo-skeleton [16] or bilateral projection [35]: for rectangular matrix $\mathbf{A}$, and using the same base and target sampling, i.e., $\mathcal{I}_r^b = \mathcal{I}_r^t$ and $\mathcal{I}_c^b = \mathcal{I}_c^t$. It generalizes Nyström method from symmetric to rectangular matrices. Let $\mathbf{W}$ be the intersection of $\mathbf{C}$ and $\mathbf{R}$, then it has compact form,

$$\mathbf{A} \approx \mathbf{C}\mathbf{W}^{\dagger}\mathbf{R}. \tag{2}$$

3. Sketch-CUR [29]: in case of rectangular $\mathbf{A}$ and independent base/target sampling (with different rates).

These algorithms only access a small fraction of rows and columns to reconstruct the input matrix. However, the performance can be inferior, in particularly on general, rectangular matrices (Section 4.2 for detail). More recently, online streaming algorithms began to draw increasing attention for matrix decomposition [21, 26, 31]. Their memory cost is much smaller, but the whole matrix still needs to be fully accessed and the time complexity is at least $O(m \cdot n)$.

3 THEORETIC ANALYSIS OF MATRIX BILATERAL DECOMPOSITION

In this section, we present a novel theoretic analysis on bilateral decomposition of general, rectangular matrices. It links the quality of matrix approximation with the encoding powers of bilateral sampling, and inspires a novel, encoding-orientated sampling scheme.

3.1 A New Error Bound

As discussed in Section 2.1, for most randomized algorithms in the literature, the randomization scheme is pre-determined and then probabilistic theoretical guarantees are derived, specifying how many samples should be selected to achieve a desired accuracy [3, 10, 14, 17, 25]. In this work, our goal is different: *we want to maximally reduce the approximation error given a fixed rate of sampling*. Therefore, our error bound is expressed in terms of the numerical properties of sampling, instead of a pre-selected sampling scheme. Our error bound will shed important light on the design of effective sampling schemes that fully exploits a limited computing resource, thus quite useful to practitioners.

Given an input matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, assume $\mathbf{A} = \mathbf{P}\mathbf{Q}^{\top}$, where $\mathbf{P} \in \mathbb{R}^{m\times r}$ and $\mathbf{Q} \in \mathbb{R}^{n\times r}$ are exact decomposition, say, from an SVD. Without loss of generality suppose we select k columns $\mathbf{C} = \mathbf{A}_{[:,\mathcal{Z}^c]}$ and k rows $\mathbf{R} = \mathbf{A}_{[\mathcal{Z}^r,:]}$, where $\mathcal{Z}^c$ and $\mathcal{Z}^r$ are sampling indices. These indices locate representative instances (rows) in $\mathbf{P}$ and $\mathbf{Q}$, denoted by $\mathbf{Z}^r = \mathbf{P}_{[\mathcal{Z}^r,:]}$ and $\mathbf{Z}^c = \mathbf{Q}_{[\mathcal{Z}^c,:]}$, respectively. In order to study how the bilateral sampling affects matrix approximation result, we adopt a data-encoding model. Let the m rows of $\mathbf{P}$ be segmented into k groups, where the group representatives are rows in $\mathbf{Z}^r$; similarly, let the n rows in $\mathbf{Q}$ fall into k groups, where the group representatives are rows in $\mathbf{Z}^c$. Let the $s^r(i)$ be the group assignment function that maps the i-th row in $\mathbf{P}$ to the $s^r(i)$-th row in $\mathbf{Z}^r$; similarly, $s^c(i)$ maps the i-th row in $\mathbf{Q}$ to the $s^c(i)$-th row in $\mathbf{Z}^c$. Then, the errors of reconstructing $\mathbf{P}$ and $\mathbf{Q}$ using representatives in $\mathcal{Z}^r$ and $\mathcal{Z}^c$ via respective mapping function $s^r(\cdot)$ and $s^c(\cdot)$ can be defined as

$$e^r = \sum_{l=1}^{m} \|\mathbf{P}_{[l,:]} - \mathbf{Z}^r_{[s^r(l),:]}\|^2, \quad e^c = \sum_{l=1}^{n} \|\mathbf{Q}_{[l,:]} - \mathbf{Z}^c_{[s^c(l),:]}\|^2. \tag{3}$$

We also define T^r and T^c as the maximum cluster sizes in $\mathbf{P}$ and $\mathbf{Q}$, respectively, as

$$T^r = \max_{1\le y\le k} |\{i : s^r(i) = y\}|, \; T^c = \max_{1\le y\le k} |\{i : s^c(i) = y\}|. \tag{4}$$

We can then analyze how matrix approximation error is associated with the encoding powers of bilateral sampling (in reconstructing the decompositions $\mathbf{P}$ and $\mathbf{Q}$) as follows.

Theorem 1. *Given an input matrix $\mathbf{A}$, and suppose one samples k columns $\mathbf{C}$ and k rows $\mathbf{R}$, with the intersection $\mathbf{W}$. Then we can bound the approximation error (2) as follows.*

$$\left\|\mathbf{A} - \mathbf{C}\mathbf{W}^{\dagger}\mathbf{R}\right\|_F \le \left(\sqrt{6k\theta}T^{\frac{3}{2}}\right)\cdot\sqrt{\mathbf{e^r}+\mathbf{e^c}} + \left(k\theta T\|\mathbf{W}^{\dagger}\|_F\right)\cdot\sqrt{\mathbf{e^c}\mathbf{e^r}}$$

Here $T = \max(T^r, T^c)$, T^r and T^c are defined in (4), e^r and e^c are defined in (3); θ is a data dependent constant (see Appendix). Proof can be found in the Appendix.

From Theorem 1, we can see that given a fixed sampling rate k the key quantities affecting the matrix approximation error are e^r and e^c (3), the encoding errors of reconstructing $\mathbf{P}$ and $\mathbf{Q}$ with their representative rows whose indices are specified in the bilateral sampling. In case both e^c and e^r approach zero, the approximation error will also approach zero. Namely, choosing a bilateral sampling that can reduce the encoding errors (3) is an effective way to

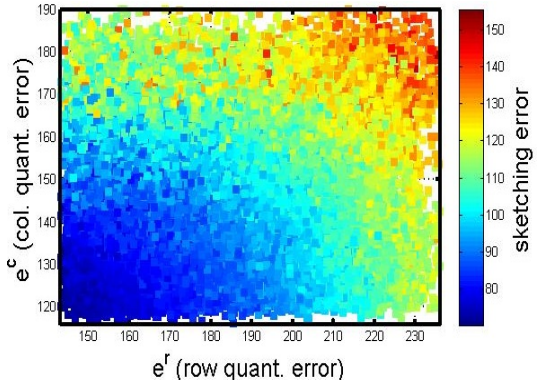


Figure 2: Matrix approximation error vs. row and column encoding errors.

bound matrix approximation error. We visualize their relations in Figure 2. Here, given $\mathbf{A}$ with exact decomposition $\mathbf{PQ}^\top$, we perform bilateral samplings many times on rows of $\mathbf{P}$ and $\mathbf{Q}$, each time using different choices such as uniform sampling, vector quantization with different number of iterations, and so on, such that resultant encoding errors vary a lot. Then we plot the encoding errors (e^r, e^c) and color-code it with the corresponding matrix approximation error $\mathcal{E}$. As can be seen, $\mathcal{E}$ shows a clear correlation with e^c and e^r. Only when both e^c and e^r are small (blue), $\mathcal{E}$ will be small; if either e^c or e^r is large, $\mathcal{E}$ will be large too.

3.2 Low-rank Compression Sampling

Theorem 1 provides an important criterion for sampling: the selected rows and columns of $\mathbf{A}$ should be representative code-vectors in the low-rank embeddings $\mathbf{P}$ and $\mathbf{Q}$, in order for the approximation error to be well bounded. To achieve this goal, we propose to use k-means to select representative rows in $\mathbf{P}$ and $\mathbf{Q}$. The k-means algorithm, a.k.a. vector quantization [23], is a lossy data compression technique that has been widely used in signal processing. The algorithm can iteratively compute a set of representative code-vectors by finding a local optimal of the encoding error (or distortion) as defined in (3). Of course, exact decompositions $\mathbf{P}$ and $\mathbf{Q}$ are impractical and possibly high dimensional. Therefore, we resort to an alternative low-dimensional embedding $\mathbf{P} \in \mathbb{R}^{m\times k}, \mathbf{Q} \in \mathbb{R}^{n\times k}$ that will be discussed in more detail in the cascaded compression sampling framework in Section 4.

We shall first examine behaviour of the k-means sampling on the low-rank embeddings of the input matrix. For dense matrices, their energy spreads across all the rows and columns, so the embedding has a fairly uniform distribution (Figure 3(a)). For sparse matrices, energy distribution tends to concentrate around a small number of rows/columns; and hence the embedding collapses towards the origin (Figure 3(b). The k-means algorithm assigns more code-vectors in densely distributed regions. Therefore the code-vectors are uniform for dense matrices (Figure 3(a)), but will be attracted to the origin for sparse matrices (Figure 3(b)).

In order to prevent k-means sampling from picking too many points close to the origin (e.g. low-energy rows/columns whose entries are mostly zeros), we propose a general, importance weighted k-means sampling as follows

$$e^r = \sum_{l=1}^{m} \|\mathbf{P}_{[l,:]} - \mathbf{Z}^r_{[s^r(l),:]}\|^2 \cdot \Upsilon(\|\mathbf{P}_{[l,:]}\|_2) \quad (5)$$

$$e^c = \sum_{l=1}^{n} \|\mathbf{Q}_{[l,:]} - \mathbf{Z}^c_{[s^c(l),:]}\|^2 \cdot \Upsilon(\|\mathbf{Q}_{[l,:]}\|_2). \quad (6)$$

Here we use the norm of $\mathbf{P}_{[l,:]}$ (or $\mathbf{Q}_{[l,:]}$) to re-weight the objective of k-means in (3), because it is an upper-bound of the energy of the ith row in $\mathbf{A}$ (up to a constant scaling), as $\|\mathbf{A}_{[i,:]}\| = \|\mathbf{P}_{[i,:]} \cdot \mathbf{Q}\| \leq \|\mathbf{P}_{[i,:]}\| \cdot \|\mathbf{Q}\|$. The $\Upsilon(\cdot)$ is a monotonous function adjusting the weights (e.g. power or step function). By doing this, the k-means code-vectors will be pushed away from the origin (Figure 3(c)). For dense matrices, the weighting function can be simply chosen as constant. In practice, the k-means code-vector is replaced with its closest in-sample point. Finally, the weighing can be deemed a prior knowledge (preference) on approximating rows and columns of the input matrix, which can be incorporated trivially into Theorem 1.

4 CASCADED COMPRESSION SAMPLING

Theorem 1 suggests that one perform weighted k-means sampling (5) and (6) on the bilateral embeddings of the input matrix to effectively control the approximation error. Since computing an exact embedding is impractical, we will resort to approximate embeddings discussed in the following framework.

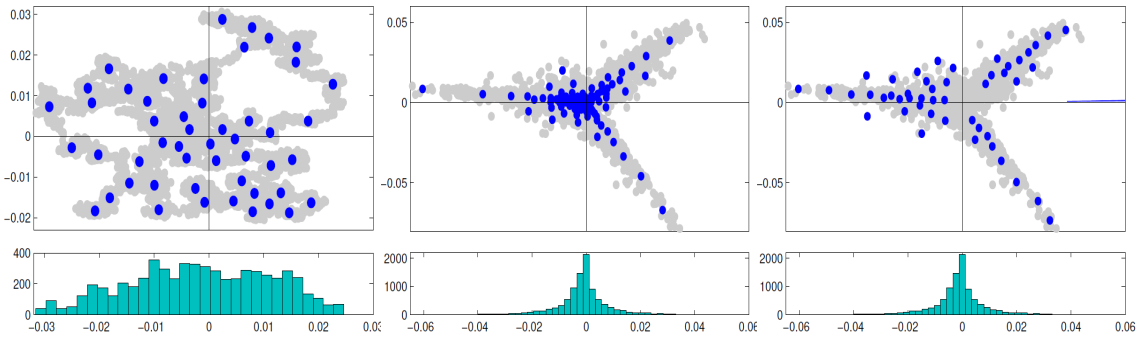
Algorithm 2: Cascaded Compression Sampling (CCS)

Input: $\mathbf{A}$; **Output**: $\mathbf{A} \approx \bar{\mathbf{U}}\bar{\mathbf{S}}\bar{\mathbf{V}}^\top$

1: *Pilot Sampling*: randomly select k columns and k rows $\mathbf{C} = \mathbf{A}_{[:,\mathcal{I}^c]}$, $\mathbf{R} = \mathbf{A}_{[\mathcal{I}^r,:]}$, $\mathbf{W} = \mathbf{A}_{[\mathcal{I}^r,\mathcal{I}^c]}$.
2: *Pilot approximation*: run $[\mathbf{U}, \mathbf{S}, \mathbf{V}]$ = `sketching`$(\mathbf{C}, \mathbf{R}, \mathbf{W})$, let $\mathbf{P} = \mathbf{U}\mathbf{S}^{\frac{1}{2}}$, and $\mathbf{Q} = \mathbf{V}\mathbf{S}^{\frac{1}{2}}$.
3: *Follow-up sampling*: perform weighted k-means on $\mathbf{P}$ and $\mathbf{Q}$, respectively, to obtain row index $\bar{\mathcal{I}}^r$ and column index $\bar{\mathcal{I}}^c$; let $\bar{\mathbf{C}} = \mathbf{A}_{[:,\bar{\mathcal{I}}^c]}$, $\bar{\mathbf{R}} = \mathbf{A}_{[\bar{\mathcal{I}}^r,:]}$, $\bar{\mathbf{W}} = \mathbf{A}_{[\bar{\mathcal{I}}^r,\bar{\mathcal{I}}^c]}$.
4: *Follow-up approximation*: $[\bar{\mathbf{U}}, \bar{\mathbf{S}}, \bar{\mathbf{V}}]$= `sketching`$(\bar{\mathbf{C}}, \bar{\mathbf{R}}, \bar{\mathbf{W}})$.

The CCS framework has two rounds of computations. First, we perform simple random sampling (step 1) and compute a pilot approximation of the input matrix (step 2). Although it can be less accurate, it provides a compact embedding of the input matrix ($\mathbf{P}$ and $\mathbf{Q}$). Then, as guided by Theorem 1, we apply weighted k-means on $\mathbf{P}$ and $\mathbf{Q}$ to identify representative samples (step 3); resultant sampling is used for final approximation (step 4). As will be shown both theoretically (Section 4.2) and empirically (Section 5), it is exactly the follow-up sampling that allows us to extract more informative samples to significantly improve approximation quality.

The CCS algorithm takes only $O((m+n)(k_1+k_2))$ space and $O((m+n)k_1k_2c)$ time, where k_1 and k_2 is the pilot and follow-up sampling rate, respectively, and c is the number of k-means iterations. In practice, $k_1 = k_2 \ll m, n$ and $c = 5$ in all our experiment, so the complexities are linear in $m+n$. In case of sparse input matrices, the complexity will be further reduced since only non-zero entries will be involved in the pilot sampling and approximation steps. As will be shown in Section 5, impressively, that our method is computationally more efficient than those algorithms taking only input-sparsity time [8, 27] on large sparse input matrices.



(a) dense matrix, k-means sampling (b) sparse matrix, k-means sampling (c) sparse matrix, weighted k-means

Figure 3: Top-2 dimensions of the low-rank embedding P, histogram on horizontal dimension, and (weighted) k-means sampling results.

4.1 The sketching Routine

This section discusses the `sketching` routine in Algorithm 2. Given a subset of rows $\mathbf{R}$, columns $\mathbf{C}$, and intersection $\mathbf{W}$ from $\mathbf{A}$, it returns $\mathbf{A} \approx \mathbf{USV}^\top$. Both Sketch-CUR and Pseudo-skeleton method are candidates. Empirically, their performance can be sensitive to the choice of the number of singular vectors used in computing the pseudo-inverse (1,2), as shown in Figure 4. In particular, the optimal rank can be quite small and varies from data to data.

In the following we propose a stabilized, parameter-free version of Pseudo-skeleton method (2). Assuming an SVD $\mathbf{W} = \mathbf{U}_w \Sigma_w \mathbf{V}_w^\top$, then $\mathbf{W}^\dagger = \mathbf{V}_w \Sigma_w^{-1} \Sigma_w \Sigma_w^{-1} \mathbf{U}_w^\top$. Plug this into $\mathbf{A} \approx \mathbf{C}\mathbf{W}^\dagger\mathbf{R}$, we have

$$\mathbf{A} \approx \left(\mathbf{C}\mathbf{V}_w \Sigma_w^{-1}\right) \Sigma_w \left(\Sigma_w^{-1} \mathbf{U}_w^\top \mathbf{R}\right).$$

Here, $\mathbf{U}_w$ and $\mathbf{V}_w$ are left and right singular vectors of $\mathbf{W}$, extrapolated via $\mathbf{U}_w^\top\mathbf{R}$ and $\mathbf{C}\mathbf{V}_w$, respectively, and then normalized by the singular values Σ_w. In case $\Sigma_w(i,i)$ approaches zero, the normalization becomes numerically unstable. To avoid this ambiguity, we use the norms of the extrapolated singular-vectors for normalization,

$$\mathbf{A} \approx \left(\mathbf{C}\mathbf{V}_w \mathbf{N}_c^{-1}\right) \sqrt{\frac{mn}{k^2}} \Sigma_w \left(\mathbf{N}_r^{-1} \mathbf{U}_w^\top \mathbf{R}\right), \tag{7}$$

$$s.t.\ \ \mathbf{N}_c = \mathrm{diag}(\|\mathbf{C}\mathbf{V}_w\|_\otimes),\ \mathbf{N}_r = \mathrm{diag}(\|\mathbf{R}^\top \mathbf{U}_w\|_\otimes). \tag{8}$$

Here $diag(\cdot)$ fills a diagonal matrix with given vector, $\|\cdot\|_\otimes$ returns column-wise norms, namely $\mathbf{N}_r$ and $\mathbf{N}_c$ are norms of extrapolated singular vectors. The constant $\sqrt{mn}/k$ adjusts scale of solution, which can also be computed adaptively using a small validation set. We can then define the `sketching()` routine with $\mathbf{U} = \mathbf{C}\mathbf{V}_w\mathbf{N}_c^{-1}$, $\mathbf{V} = \mathbf{N}_r^{-1}\mathbf{U}_w^\top\mathbf{R}$, and $\Sigma = \Sigma_w\sqrt{mn}/k$, see Algorithm 3. As can be seen from Figure 4, it gives stable result though using all singular vectors. In practice, this can be more convenient than choosing the best rank (or threshold) through validation on a hold-out data set.

In positive semi-definite (PSD) matrices, intriguingly, numerical sensitivity diminishes by sampling the same subset of rows and columns, reducing to the Nyström method [12, 13, 30]. We speculate that PSD matrix resides in a Riemannian manifold [2], so symmetric sampling better captures its structure. However, rectangular matrix is not endowed with any structural constraint. This makes the approximation of rectangular matrices particularly challenging and abundant results of the Nyström method may not be borrowed here directly [12, 13, 18, 30, 34].

Algorithm 3: Stabilized sketching

Input: Sampled columns ($\mathbf{C}$), rows ($\mathbf{R}$), and intersection ($\mathbf{W}$)
Output: Stable factorization

1: Compute singular value decomposition $\mathbf{W} = \mathbf{U}_w\Sigma_w\mathbf{V}_w^\top$.
2: Extrapolate left and right singular vectors as $\mathbf{C}\mathbf{V}_w$ and $\mathbf{U}_w^\top\mathbf{R}$.
3: Compute column norms of $\mathbf{C}\mathbf{V}_w$ and $\mathbf{R}^\top\mathbf{U}_w$, load them in diagonal matrices $\mathbf{N}_c$ and $\mathbf{N}_r$, respectively (8).
4: Normalize singular vectors as $\mathbf{C}\mathbf{V}_w\mathbf{N}_c^{-1}$ and $\mathbf{N}_r^{-1}\mathbf{U}_w^\top\mathbf{R}$.
5: Reconstruct by $\left(\mathbf{C}\mathbf{V}_w\mathbf{N}_c^{-1}\right)\Sigma_w\left(\mathbf{N}_r^{-1}\mathbf{U}_w^\top\mathbf{R}\right)$.
6: Re-scale by (7) or data-dependent factor through validation.

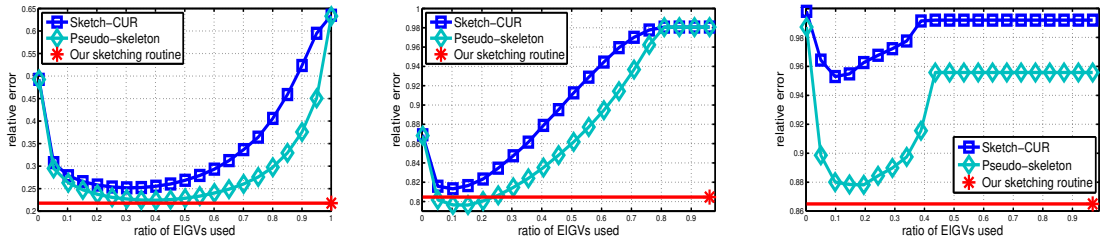
4.2 Algorithmic Boosting Property

In the literature, many two-step methods were designed for fast CUR decomposition. They start from an initial decomposition (fast JL-transform [11], random projection [3, 28, 29]), and then compute a sampling probability for subsequent CUR [3, 28]. Note that these two-step methods focus more on theoretic guarantees of the final approximation, but not the improvement of approximation quality between the two steps; in comparison, we target on algorithmically boosting the approximation accuracy from the first step to the second step, both of which are computationally quite cheap.

How to quantify the rise of approximation quality in a two-step method? How to choose the right pilot and follow-up sampling to save computations and maximize performance gains? By cascading random sampling with weighted k-means, we have derived a working example of "algorithmic boosting". Next, we show that, decrement of the error bound from the pilot to the follow-up approximation in CCS, is lower bounded by the drop of encoding errors achieved through the follow-up k-means sampling. Namely, the better the encoding based sampling, the larger the performance gain.

Theorem 2. *Let the error bound of the pilot and follow-up sketching in CABS be Ψ_p and Ψ_f, respectively. Then the error bound will drop by at least the following amount*

$$\begin{aligned}\Psi_p - \Psi_f \quad &\geq \quad \sqrt{\frac{3k\theta T_p^c T_p^r (T_p^r + T_p^c)}{2(e_p^r + e_p^c)}} \left|\Delta_{\mathbf{e}}^{\mathbf{r}} + \Delta_{\mathbf{e}}^{\mathbf{c}}\right| \\ &+ \quad k\theta \left\|\mathbf{W}_p^\dagger\right\|_F \sqrt{\frac{T^c T^r}{e_p^r e_p^c}} \left|\Delta_{\mathbf{e}}^{\mathbf{r}} e_f^c + \Delta_{\mathbf{e}}^{\mathbf{c}} e_f^r + \Delta_{\mathbf{e}}^{\mathbf{r}} \Delta_{\mathbf{e}}^{\mathbf{c}}\right|.\end{aligned}$$



(a) galaxy image (b) movie-lens (c) 20newsgroup

Figure 4: Performance of sketch-CUR and pseudo-skeleton method can be rank sensitive.

Parameters are defined in the same way as in Theorem 1, and sub-index $\{p, f\}$ denotes pilot or follow-up; $\Delta_{\mathbf{e}}^{\mathbf{r}}$ ($\Delta_{\mathbf{e}}^{\mathbf{c}}$) is the drop of row (column) encoding error in the follow-up k-means sampling. Proof of Theorem 2 can be found in the Appendix.

The boosting effect is not dependent on specific `sketching` routine. Empirically, using Pseudo-skeleton or Sketch-CUR can also achieve significant performance gains, showing the generality of our framework in achieving the goal of algorithmic boosting.

4.3 Relation with Matrix Coherence

Matrix coherence is a useful index to describe how easy it is to recover an input matrix through random sampling or matrix completion algorithms [6]. Let $\mathbf{U} \in \mathbb{R}^{m\times r}$ be the top r left singular vectors, then coherence can be defined as

$$\mu_0(\mathbf{U}) = \max_{ij} |\mathbf{U}_{ij}|, \quad or \quad \mu_1(\mathbf{U}) = \max_{i=1,2,\ldots,m} \|\mathbf{U}_{[i,:]}\|_2^2. \tag{9}$$

Our error bound is also related to matrix singular vectors: it depends on the quantization error induced in grouping/encoding the rows of the re-scaled singular vectors. According to [32], the quantization error can be written as

$$e = tr(\mathbf{U}\Delta\mathbf{U}^\top) - tr(\Pi\mathbf{U}\Delta\mathbf{U}^\top\Pi^\top) \tag{10}$$

where Δ is the singular value matrix, and $\Pi \in \mathbb{R}^{m\times k}$ is a partition matrix encoding the k-means sampling. For both the coherence (9) and quantization error (10), the lower value, the easier the approximation. An important difference is that the quantization error hinges on both the singular-vector structures and (unknown) sampling scheme, instead of a pre-defined, random sampling.

5 EXPERIMENTS

Our experiments run on a server with 2.6GHZ processor and 64G RAM. Benchmark data sets are described in Table 1. All codes are written in matlab and *fully optimized* by vectorized operations and matrix computations, to guarantee a fair comparison in case time consumption is considered.

First we compare the performance of linear-cost algorithms, including: (a) Sketch-CUR [29], where target sampling rate is three times as much as the base sampling rate; (b) Pseudo-skeleton [16], which is a generalized version of the Nyström method; (c) Ours (`pilot`), step 2 of Algorithm 2; (d) Ours (`follow-kmeans, wkmeans, hdthd`), step 4 of Algorithm 2, using constant weighting, power function weighting (power equals 5), and step function weighting, respectively[1]. In Figure 5, we gradually increase sampling rate from 1% to 15%, and report averaged error[2] $\|\mathbf{A} - \tilde{\mathbf{A}}\|_F/\|\mathbf{A}\|_F$ (for sparse matrices, only non-zero entries are considered) over 20 repeats. The number of selected rows and columns are both k for simplicity, with the sampling rate defined as $k/\sqrt{mn}$.

Table 1: Summary of benchmark data sets.

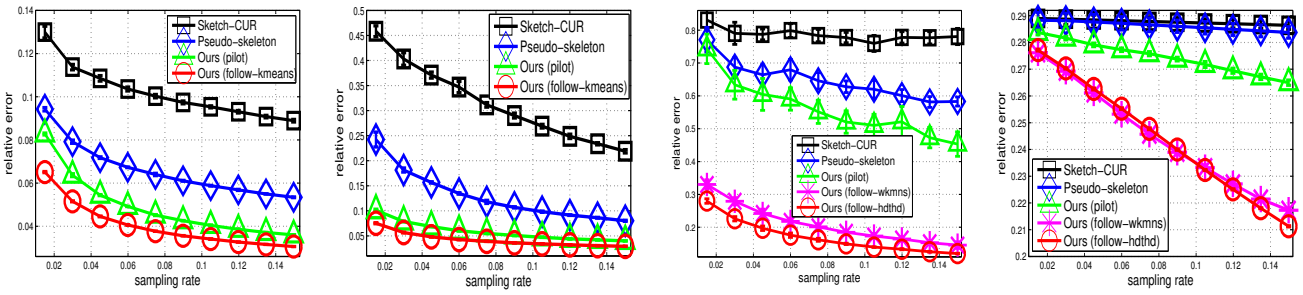
data	#row	#column	sparsity	content
heic0707a	18000	18000	dense	image (galaxy)
heic1206a	29566	14321	dense	image (galaxy)
mnist	60000	60000	dense	RBF kernel
a8a	22696	22696	dense	RBF kernel
movielens	69878	10677	0.56%	movie rating
20news	18774	61188	0.22%	word document
reuters	8293	18933	0.25%	word document
TDT2	10212	36771	0.35%	word document

We can see that our pilot approximation result is already more accurate than Pseudo-skeleton and Sketch-CUR, demonstrating the stability of our sketching routine proposed in Section 4.1. Our follow-up sampling is consistently better than pilot sampling, which demonstrates the "algorithmic boosting" effect; in particular, the boosting effect is more significant on sparse matrices (larger improvement of accuracy between the pilot and follow-up approximations). We also observe that on sparse matrices, the step function (`follow-hdthd`) is slightly better than the power function (`follow-wkmeans`) when used as weighting in k-means sampling.

Then we compare our approach with state-of-the-art randomized algorithms (with quadratic costs), including (a) Adaptive-CUR [28], which uses error distribution of an initial approximation to guide subsequent sampling; (b) CUR Method, a two-step method using fast JL-transform to compute leverage scores and then perform CUR [11]; (c) Random projection [17], with $q = 1$ step of power iteration; (d) Input-sps-time: sparse embedding method by [8]; (e) Ours (`k-means`) for dense matrices with constant weighting function; and (f) Ours (`hd-thd`) for sparse matrices with step weighting function. Averaged performance over 20 repeats is shown in Fig 6. The

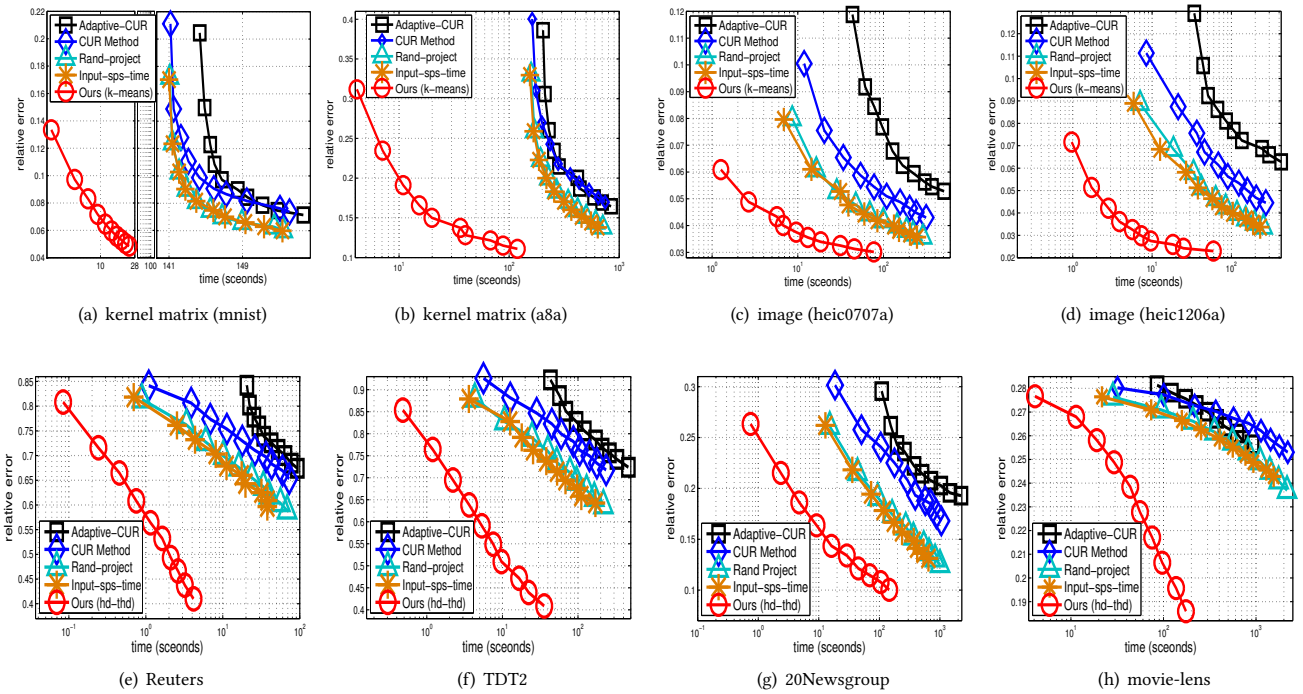
[1]For step function, $\Upsilon_i = 1$ if $\|P_{[i,:]}\|_2$ is among the top-k largest for $i = 1, 2, \ldots, m$, and $\Upsilon_i = 0$ otherwise; so we also call this "hard-threshold" (`hdthd`). In other words, only top-k points are sampled under such weighting scheme.

[2]For all methods under comparison, when computing the error, the approximation $\tilde{\mathbf{A}}$ is first fit to $\mathbf{A}$ by linear transform to remove global scaling and translation factors.



(a) image (b) image (c) 20newsgroup (d) movie-lens

Figure 5: Relative error of our method and other randomized algorithms (with linear costs).



(a) kernel matrix (mnist) (b) kernel matrix (a8a) (c) image (heic0707a) (d) image (heic1206a)

(e) Reuters (f) TDT2 (g) 20Newsgroup (h) movie-lens

Figure 6: Relative approximation error of our method and randomized algorithms (with quadratic costs).

sampling rate for our method is chosen higher than quadratic-cost methods so that their time consumptions are more comparable.

We can see that the Adaptive-CUR can be slower, which we speculate is due to the computation of the residue of approximation on the whole input matrix. The sparse embedding method (input-sparsity-time) is faster and more accurate than the random projection with a (dense) Gaussian template matrix, but the difference can be insignificant on both dense and sparse matrices. The fast CUR method has a similar time cost compared with random projection, but is less accurate. Our approach has significant performance gains on both dense and sparse matrices; its advantage is most obvious on kernel matrices by only computing a small subset of its rows and columns; in comparison, all other methods need to compute the entire kernel matrix. Overall, our approach can be one to two orders of magnitude faster than existing algorithms in order to attain similar accuracies. In the meantime, our memory cost is only a small fraction of theirs. The larger the data, the higher the performance gains that can be expected.

6 CONCLUSIONS

In this work, by unravelling the interesting underlying connection between matrix decomposition and lossy data compression, we proposed cascaded compression sampling for linear-cost low-rank decomposition of large matrices. Our method outperfroms existing randomized algorithms using either dense or sparse projection schemes, demonstrating the power of data encoding in solving large-scale matrix sketching problems. We are currently investigating how to characterize the behaviour of compression-based sampling more precisely using sharp probabilistic error bounds.

APPENDIX

Proof of Theorem 1

We partition $\mathbf{A}$ into equal-sized blocks, by the clusters defined in Section 3.1. Remind that rows in $\mathbf{P}$ are grouped into k clusters with cluster size $T^r_{(i)}$. We add virtual instances to $\mathbf{P}$ such that all clusters have the same size, i.e., $T^r = \max T^r_{(i)}$. Virtual instances added to the qth cluster are chosen as the qth representative in $\mathbf{Z}^r$, so they induce no extra quantization errors. Then, we can re-partition $\mathbf{P}$ into T^r partitions each containing exactly k instances. We use $\mathcal{I}^r_i$ to denote the indexes of these partitions for $i = 1, 2, ..., T^r$; similarly rows in $\mathbf{Q}$ are completed by virtual rows, falling in partitions $\mathcal{I}^c_j$ for $j = 1, 2, ..., T^c$. Finally $\mathbf{A}$ is augmented to kT^r-by-kT^c matrix, forming a number of T^cT^r blocks each is of size k-by-k. The approximation error on each of these blocks is bounded as

$$\frac{1}{\sqrt{k\theta}}\left\|\mathbf{A}_{[\mathcal{I}^r_i,\mathcal{I}^c_j]} - \mathbf{C}_{[\mathcal{I}^r_i,:]}\mathbf{W}^\dagger\mathbf{R}^\top_{[\mathcal{I}^c_j,:]}\right\|_F \le \sqrt{e^r_j + e^c_j} + \sqrt{e^r_i} + \sqrt{e^c_i} + \sqrt{k\theta e^r_i e^c_j}\left\|W^\dagger\right\|_F. \quad (11)$$

Here $e^r_i = \sum_{l\in\mathcal{I}^r_i}\|\mathbf{P}_{[l,:]} - \mathbf{Z}^r_{[s(l),:]}\|^2$ is the error of encoding rows of $\mathbf{P}$ (via $\mathcal{I}^r_i$) with representative $\mathbf{Z}^r$ via the mapping s^r. Similarly, $e^c_i = \sum_{l\in\mathcal{I}^c_i}\|\mathbf{P}_{[l,:]} - \mathbf{Z}^c_{[s(l),:]}\|^2$ is the error of encoding rows in $\mathbf{Q}$ (specified by $\mathcal{I}^c_j$) with representative $\mathbf{Z}^c$.

To prove this inequality, define $\langle\mathbf{X},\mathbf{Y}\rangle = \mathbf{X}\mathbf{Y}^\top$ for matrices $\mathbf{X},\mathbf{Y}$ with proper dimensions

$$\mathbf{C}_{[\mathcal{I}^r_i,:]} = \left\langle\mathbf{P}_{[\mathcal{I}^r_i,:]},\mathbf{Q}_{[\mathcal{Z}^c,:]}\right\rangle, \mathbf{R}_{[\mathcal{I}^c_j,:]} = \left\langle\mathbf{P}_{[\mathcal{I}^c_j,:]},\mathbf{Q}_{[\mathcal{Z}^r,:]}\right\rangle, \quad (12)$$

$$\mathbf{A}_{[\mathcal{I}^r_i,\mathcal{I}^r_j]} = \left\langle\mathbf{P}_{[\mathcal{I}^r_i,:]},\mathbf{Q}_{[\mathcal{I}^c_j,:]}\right\rangle, \mathbf{W} = \left\langle\mathbf{P}_{[\mathcal{Z}^r,:]},\mathbf{Q}_{[\mathcal{Z}^c,:]}\right\rangle. \quad (13)$$

Here we use transposed version of $\mathbf{R}$ for convenience of proof. The change of representation won't affect the correctness of our proofs. We also define the following difference matrices

$$\Delta_C = \mathbf{C}_{[\mathcal{I}^r_i,:]} - \mathbf{W}, \Delta_R = \mathbf{R}_{[\mathcal{I}^c_j,:]} - \mathbf{W}, \Delta_A = \mathbf{A}_{[\mathcal{I}^r_i,\mathcal{I}^c_j]} - \mathbf{W}, \quad (14)$$

We will also need the following inequality,

$$(\phi(\mathbf{x}_1,\mathbf{y}_1) - \phi(\mathbf{x}_2,\mathbf{y}_2))^2 \le \theta\cdot(\|\mathbf{x}_1-\mathbf{x}_2\|^2 + \|\mathbf{y}_1-\mathbf{y}_2\|^2), \quad (15)$$

where $\mathbf{x}_1,\mathbf{x}_2,\mathbf{y}_1,\mathbf{y}_2$ are $1\times d$ vectors, $\phi(\cdot,\cdot)$ be the inner product between two such vectors, $\theta = 2f'(\xi)^2$ (using Lagrangian mean-value theorem). Using Proposition 2, we can bound norms of the difference matrices

$$\begin{aligned}
&\|\Delta_A\|_F^2 = \left\|\mathbf{A}_{[\mathcal{I}^r_i,\mathcal{I}^c_j]} - \mathbf{W}\right\|_F^2\\
=\ &\left\|\left\langle\mathbf{P}_{[\mathcal{I}^r_i,:]},\mathbf{Q}_{[\mathcal{I}^c_j,:]}\right\rangle - \left\langle\mathbf{P}_{[\mathcal{Z}^r,:]},\mathbf{Q}_{[\mathcal{Z}^c,:]}\right\rangle\right\|_F^2\\
=\ &\sum_{p,q=1}^{k}\left[\phi\left(\mathbf{P}_{[\mathcal{I}^r_i(p),:]},\mathbf{Q}_{[\mathcal{I}^c_j(q),:]}\right) - \phi\left(\mathbf{P}_{[\mathcal{Z}^r(p),:]},\mathbf{Q}_{[\mathcal{Z}^c(q),:]}\right)\right]^2\\
\le\ &\theta\sum_{p,q=1}^{k}\left(\left\|\mathbf{P}_{[\mathcal{I}^r_i(p),:]} - \mathbf{P}_{[\mathcal{Z}^r(p),:]}\right\|^2 + \left\|\mathbf{Q}_{[\mathcal{I}^c_j(q),:]} - \mathbf{Q}_{[\mathcal{Z}^c(q),:]}\right\|^2\right)\\
=\ &k\theta\left(e^r_i + e^c_j\right).
\end{aligned}$$

Here we have used the pre-defined relation $s^r(\mathcal{I}^r_i(p)) = p$, and $s^c(\mathcal{I}^c_j(q)) = q$, since the partition index $\mathcal{I}^c_i$ and $\mathcal{I}^c_j$ has the corresponding representative set $\mathcal{Z}$. Similarly,

$$\begin{aligned}
&\|\Delta_C\|_F^2 = \left\|\mathbf{C}_{[\mathcal{I}^r_i,:]} - \mathbf{W}\right\|_F^2\left\|\left\langle\mathbf{P}_{[\mathcal{I}^r_i,:]},\mathbf{Q}_{[\mathcal{Z}^c,:]}\right\rangle - \left\langle\mathbf{P}_{[\mathcal{Z}^r,:]},\mathbf{Q}_{[\mathcal{Z}^c,:]}\right\rangle\right\|_F^2\\
&\le\theta\sum_{p,q=1}^{k}\left\|\mathbf{P}_{[\mathcal{I}^r_i(p),:]} - \mathbf{P}_{[\mathcal{Z}^r(p),:]}\right\|^2 = \theta k e^r_i,
\end{aligned}$$

and $\|\Delta_R\|_F^2 \le \theta k e^c_j$. By using above three inequalities, and combining them with (13), we can prove (11). With this proposition, and by using the inequality $\sum_i^n\sqrt{x_i} \le \sqrt{n\sum_i x_i}$, the overall approximation error can be bounded as follows

$$\begin{aligned}
&\frac{1}{\sqrt{k\theta}}\left\|\mathbf{A} - \mathbf{C}\mathbf{W}^\dagger\mathbf{R}^\top\right\|_F \le \frac{1}{\sqrt{k\theta}}\sum_{i=1}^{T^r}\sum_{j=1}^{T^c}\left\|\mathbf{A}_{[\mathcal{I}^r_i,\mathcal{I}^c_j]} - \mathbf{C}_{[\mathcal{I}^r_i,:]}\mathbf{W}^\dagger\mathbf{R}_p[\mathcal{I}^c_j,:]^\top\right\|_F\\
&\le\sqrt{T^c}\sum_i\sqrt{\sum_j(e^c_j + e^r_j)} + \sqrt{T^r}\sum_j\sqrt{\sum_i e^r_i} + \sqrt{T^c}\sum_i\sqrt{\sum_j e^c_j}\\
&+\sqrt{k\theta}\left\|\mathbf{W}^\dagger\right\|_F\sqrt{T^c}\sum_i\sqrt{\sum_j e^r_i e^c_j}\\
&\le\sqrt{T^cT^r(T^re^c + T^ce^r)} + T^c\sqrt{T^re^r} + T^r\sqrt{T^ce^c} + \sqrt{k\theta}\sqrt{T^cT^re^ce^r}\left\|\mathbf{W}^\dagger\right\|_F\\
&\le\sqrt{3(T^c+T^r)(e^c+e^r)} + \sqrt{k\theta}\sqrt{T^cT^re^ce^r}\|\mathbf{W}^\dagger\|_F.
\end{aligned}$$

By using $T = \max(T^r, T^c)$, we can easily prove Theorem 1.

Proof of Theorem 2

The CCS algorithm uses only the k-dimensional embedding instead of the exact embedding as stated in Theorem 1. The consequence is that the resultant error bound will be loosened by the trailing singular values of the input matrix, as follows

$$\begin{aligned}
&\left\|\mathbf{A} - \mathbf{C}\mathbf{W}^\dagger\mathbf{R}^\top\right\|_F \le \left\|\mathbf{A}_k - \mathbf{C}\mathbf{W}^\dagger\mathbf{R}^\top\right\|_F + \left\|\mathbf{A}_{\overline{k}}\right\|_F\\
\le\ &\sqrt{T^cT^r}\left(\sqrt{3k\theta(e^r+e^c)(T^r+T^c)} + k\theta\sqrt{e^ce^r}\left\|\mathbf{W}^\dagger\right\|_F\right) + \|\mathbf{A}_{\overline{k}}\|_F\\
=\ &\mu\sqrt{e^r+e^c} + \nu\sqrt{e^r\cdot e^c} + \|\mathbf{A}_{\overline{k}}\|_F,
\end{aligned}$$

where we have $\mu = \sqrt{3k\theta T^cT^r(T^r+T^c)}, \nu = k\theta\sqrt{T^cT^r}\|\mathbf{W}^\dagger\|_F$, and $\|\mathbf{A}_{\overline{k}}\|_F = \sqrt{\sum_{i=k+1}^{\min(m,n)}\sigma_i^2}$ is a constant which is the l_2-norm of the $\min(m,n) - k$ singular values. In case singular-value spectrum decays rapidly, this constant can be quite small. In other words the error bound is only slightly loosened if rank-k embeddings are used for the follow-up sampling.

In the following we will use the updated error bound for the pilot and follow-up sketching, as

$$\begin{aligned}
\Psi_p &= \mu_p\sqrt{e^r_p + e^c_p} + \nu_p\sqrt{e^r_p\cdot e^c_p} + \|\mathbf{A}_{\overline{k}}\|\\
\Psi_f &= \mu_p\sqrt{e^r_f + e^c_f} + \nu_f\sqrt{e^r_f\cdot e^c_f} + \|\mathbf{A}_{\overline{k}}\|,
\end{aligned}$$

where

$$\begin{aligned}
\mu_p &= \sqrt{3k\theta T^c_pT^r_p(T^r_p + T^c_p)}, \nu_p = k\theta\sqrt{T^c_pT^r_p}\left\|\mathbf{W}^\dagger_p\right\|_F\\
\mu_f &= \sqrt{3k\theta T^c_fT^r_f(T^r_f + T^c_f)}, \nu_f = k\theta\sqrt{T^c_fT^r_f}\left\|\mathbf{W}^\dagger_f\right\|_F.
\end{aligned}$$

Here the sub-index $\{p, f\}$ denotes the pilot and the follow-up step, and all parameters are defined in the same way as in Theorem 1. For example, $T^c_{p,f}$ and $T^r_{p,f}$ are the maximum cluster sizes in the column and row embeddings; $e^c_{p,f}$ and $e^r_{p,f}$ are the encoding errors for the column and row embeddings. The above relation holds because the random sampling in the pilot step can be deemed as equivalently running on the the rank-k embeddings of the input matrix. This instantly gives the following guarantee

$$\Delta^r_e = e^r_p - e^r_f \geq 0, \Delta^c_e = e^c_p - e^c_f \geq 0.$$

Here Δ^r_e and Δ^c_e are exactly the drop of the encoding errors achieved by the k-means sampling algorithm in the follow-up sampling step. Next, we will show that, the drop of the error bounds from the pilot sketching step to the follow-up sketching step in CCS, can be exactly quantified by the drop of the encoding errors Δ^r_e and Δ^c_e.

We also use the inequality $g(x) - g(y) \geq (x - y) \cdot g'(x)$ for the function $g(x) = \sqrt{x}$ and any pair of numbers $x \geq y \geq 0$. Namely $\sqrt{x} - \sqrt{y} \geq (x - y)\frac{1}{2\sqrt{x}}$. We assume that $T^r_p = T^r_f = T^r$, and $T^c_p = T^c_f = T^c$. This is justified because the pilot random sampling and the follow-up k-means sampling can be deemed as running on the same, rank-k embedding of the input matrix. Namely the maximum cluster sizes in the two rounds of samplings are the same. So we can write $\mu_p = \mu_f = \mu$. On other hand, we assume that $\|\mathbf{W}^\dagger_f\|_F \leq \|\mathbf{W}^\dagger_p\|_F$. This is because the follow-up sampling selects highly non-redundant rows and columns as representatives, so the norm of $\|\mathbf{W}^\dagger\|_F$ typically drops. In other words,

$$\nu_p\sqrt{e^r_p e^c_p} - \nu_f\sqrt{e^r_f e^c_f} \geq k\theta\sqrt{e^r_p e^c_p}\sqrt{T^c T^r}\left\|\mathbf{W}^\dagger_p\right\|_F - k\theta\sqrt{e^r_f e^c_f}\sqrt{T^c T^r}\left\|\mathbf{W}^\dagger_p\right\|_F$$
$$= k\theta\sqrt{T^c T^r}\left\|\mathbf{W}^\dagger_p\right\|_F\left(\sqrt{e^r_p e^c_p} - \sqrt{e^r_f \cdot e^c_f}\right) \geq 0.$$

So we further bound the difference as follows

$$\Psi_p - \Psi_f = \left(\mu_p\sqrt{e^r_p + e^c_p} - \mu_p\sqrt{e^r_f + e^c_f}\right) + \left(\nu_p\sqrt{e^r_p \cdot e^c_p} - \nu_f\sqrt{e^r_f \cdot e^c_f}\right)$$
$$\geq \mu\left(\sqrt{e^r_p + e^c_p} - \sqrt{e^r_f + e^c_f}\right) + k\theta\sqrt{T^c T^r}\left\|\mathbf{W}^\dagger_p\right\|_F\left(\sqrt{e^r_p \cdot e^c_p} - \sqrt{e^r_f \cdot e^c_f}\right)$$
$$\geq \mu\frac{1}{2\sqrt{e^r_p + e^c_p}}\left(e^r_p - e^r_f + e^c_p - e^c_f\right) + \left\|\mathbf{W}^\dagger_p\right\|_F\frac{k\theta\sqrt{T^c T^r}}{\sqrt{e^r_p \cdot e^c_p}}\left(e^r_p e^c_p - e^r_f \cdot e^c_f\right)$$
$$= \frac{\sqrt{3k\theta T^c T^r(T^r + T^c)}}{2\sqrt{e^r_p + e^c_p}}(\Delta^r_e + \Delta^c_e) + \left\|\mathbf{W}^\dagger_p\right\|_F\frac{k\theta\sqrt{T^c T^r}}{\sqrt{e^r_p \cdot e^c_p}}\left(\Delta^r_e e^c_f + \Delta^c_e e^r_f + \Delta^r_e\Delta^c_e\right).$$

This completes the proof of Theorem 2.

REFERENCES

[1] O. Alter, P.O. Brown, and D. Botstein. 2000. Singular value decomposition for genome-wide expression data processing and modeling. *Proceedings of the National Academy of Sciences* 97, 18 (2000), 10101–10106.

[2] R. Bhatia. 2015. *Positive Definite Matrices.* Princeton University Press.

[3] C. Boutsidis, P. Drineas, and M. Magdon-Ismail. 2004. Near-Optimal Column-Based Matrix Reconstruction. *SIAM J. Comput.* 43, 2 (2004), 687–717.

[4] C. Boutsidis and A. Gittens. 2013. Improved matrix algorithms via the Subsampled Randomized Hadamard Transform. *SIAM Journal of MAtrix Analysis and Applications* 34, 3 (2013), 1301–1340.

[5] C. Boutsidis and D.P. Woodruff. 2014. Optimal CUR matrix decompositions. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing.* 353–362.

[6] E.J. Candes and B. Recht. 2009. Exact Matrix Completion via Matrix Completion. *Foundations of Computational Mathematics* 9 (2009), 717–772.

[7] K.L. Clarkson, P. Drineas, M. Magdon-Ismail, M.W. Mahoney, X. Meng, and D.P. Woodruff. 2013. The Fast Cauchy Transform and Faster Robust Linear Regression. In *Annual ACM-SIAM Symposium on Discrete Algorithms.* 466–477.

[8] K.L. Clarkson and D.P. Woodruff. 2013. Low rank approximation and regression in input sparsity time. In *Annual ACM Symposium on theory of computing.*

[9] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 41, 6 (1990), 391–407.

[10] P. Drineas, R. Kannan, and M.W. Mahoney. 2006. Fast Monte Carlo algorithms for matrices II: computing a low-rank approximation to a matrix. *SIAM Journal of Computing* 36, 1 (2006), 158–183.

[11] P. Drineas, M. Magdon-Ismail, M.W. Mahoney, and D.P. Woodruff. 2012. Fast approximation of matrix coherence and statistical leverage. *Journal of Machine Learning Research* 13, 1 (2012), 3475–3506.

[12] P. Drineas and M.W. Mahoney. 2005. On the Nyström Method for Approximating a Gram Matrix for Improved Kernel-Based Learning. *Journal of Machine Learning Research* 6 (2005), 2153–2175.

[13] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. 2004. Spectral grouping using the Nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26, 2 (2004), 214–225.

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

[15] G. H. Golub and C. F. Van Loan. 1996. *Matrix Computation.* Johns Hopkins University Press.

[16] S.A. Goreinov, E.E. Tyrtyshnikov, and N.L. Zamarashki. 1997. A theory of pseudoskeleton approximations. *Linear Algebra Appl.* 261, 1 (1997), 1–21.

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

[18] S. Kumar, M. Mohri, and A. Talwalkar. 2012. Sampling methods for the Nyström method. *Journal of Machine Learning Research* 13, 1 (2012), 981–1006.

[19] L. Lan, K. Zhang, H. Ge, W. Cheng, J. Liu, A. Rauber, X. Li, J. Wang, and H. Zha. 2017. Low-rank Decomposition Meets Kernel Learning: A Generalized Nyström Method. *Artificial Intelligence* 250 (2017), 1–15.

[20] D.D. Lee and H.S. Seung. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature* 401, 6755 (1999), 788 – 791.

[21] E. Liberty. 2013. Simple and deterministic matrix sketching. In *ACM SIGKDD international conference on Knowledge discovery and data mining.* 581–588.

[22] E. Liberty, F. Woolfe, P. Martinsson, V. Rokhlin, and M. Tygert. 2007. Randomized algorithms for the low-rank approximation of matrices. *Proceedings of National Academy of Sciences* 104, 51 (2007), 20167–20172.

[23] Y. Linde, A. Buzo, and R.M. Gray. 1980. An Algorithm for Vector Quantizer Design. *IEEE Transactions on Communications* 28, 1 (1980), 84–95.

[24] M.W. Mahoney. 2011. Randomized Algorithms for Matrices and Data. *Foundations and Trends in Machine Learning* 3, 2 (2011), 123 – 224.

[25] M.W. Mahoney and P. Drineas. 2009. CUR matrix decompositions for improved data analysis. *Proceedings of National Academy of Sciences* 106 (2009), 697–702.

[26] I. Mitliagkas, C. Caramanis, and P. Jain. 2013. Memory limited, streaming PCA. In *Advances in Neural Information Processing Systems.* 2886–2894.

[27] J. Nelson and H.L. Nguyên. 2015. OSNAP: Faster Numerical Linear Algebra Algorithms via Sparser Subspace Embeddings. In *IEEE Annual Symposium on Foundations of Computer Science (FOCS).* 135–143.

[28] S. Wang and Z. Zhang. 2012. A Scalable CUR Matrix Decomposition Algorithm: Lower Time Complexity and Tighter Bound. In *Advances in Neural Information Processing Systems 25.* 647–655.

[29] S. Wang, Z. Zhang, and T. Zhang. 2016. Towards More Efficient Nystrom Approximation and CUR Matrix Decomposition. *Journal of Machine Learning Research* 17 (2016), 1–49.

[30] C.K. I. Williams and M. Seeger. 2001. Using the Nystrom Method to Speed Up Kernel Machines. In *Advances in Neural Information Processing Systems 13.* 682–688.

[31] S. Yun, M. lelarge, and A. Proutiere. 2015. Fast and Memory Optimal Low-Rank Matrix Approximation. In *Advances in Neural Information Processing Systems 28.* 3177–3185.

[32] H. Zha, C. Ding, M. Gu, X. He, and H. Simon. 2002. Spectral relaxation for K-means clustering.. In *Neural Information Processing Systems 14.*

[33] K. Zhang, L. Lan, Z. Wang, and F. Moerchen. 2012. Scaling up Kernel SVM on Limited Resources: A Low-rank Linearization Approach. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research),* Vol. 22. 1425–1434.

[34] K. Zhang, I. Tsang, and J. Kwok. 2008. Improved Nyström low-rank approximation and error analysis. In *International conference on Machine learning.* 1232–1239.

[35] T. Zhou and D. Tao. 2011. Godec: Randomized low-rank & sparse matrix decomposition in noisy case (2011). In *International Conference on Machine Learning.*