Greedy Column Subset Selection: New Bounds and Distributed Algorithms

Jason Altschuler JASONMA@PRINCETON.EDU
Princeton University, Princeton, NJ 08544

Aditya Bhaskara BHASKARA@CS.UTAH.EDU
School of Computing, 50 S. Central Campus Drive, Salt Lake City, UT 84112

Gang (Thomas) Fu THOMASFU@GOOGLE.COM
Vahab Mirrokni MIRROKNI@GOOGLE.COM
Afshin Rostamizadeh ROSTAMI@GOOGLE.COM
Morteza Zadimoghaddam ZADIM@GOOGLE.COM
Google, 76 9th Avenue, New York, NY 10011

Abstract

The problem of column subset selection has recently attracted a large body of research, with feature selection serving as one obvious and important application. Among the techniques that have been applied to solve this problem, the greedy algorithm has been shown to be quite effective in practice. However, theoretical guarantees on its performance have not been explored thoroughly, especially in a distributed setting. In this paper, we study the greedy algorithm for the column subset selection problem from a theoretical and empirical perspective and show its effectiveness in a distributed setting. In particular, we provide an improved approximation guarantee for the greedy algorithm which we show is tight up to a constant factor, and present the first distributed implementation with provable approximation factors. We use the idea of randomized composable core-sets, developed recently in the context of submodular maximization. Finally, we validate the effectiveness of this distributed algorithm via an empirical study.

1. Introduction

Recent technological advances have made it possible to collect unprecedented amounts of data. However, extracting patterns of information from these high-dimensional massive datasets is often challenging. How do we automatically determine, among millions of measured features (variables), which are informative, and which are irrelevant or redundant? The ability to select such features from high-dimensional data is crucial for computers to recognize patterns in complex data in ways that are fast, accurate, and even human-understandable (Guyon & Elisseeff, 2003).

An efficient method for feature selection receiving increasing attention is Column Subset Selection (CSS). CSS is a constrained low-rank-approximation problem that seeks to approximate a matrix (e.g. instances by features matrix) by projecting it onto a space spanned by only a few of its columns (features). Formally, given a matrix A with n columns, and a target rank $k < n$, we wish to find a size-k subset S of A's columns such that each column A_i of A ($i \in \{1, \ldots, n\}$) is contained as much as possible in the subspace span(S), in terms of the Frobenius norm:

$$\arg\max_{S \text{ contains } k \text{ of } A\text{'s columns}} \sum_{i=1}^{n} \|\text{proj}(A_i \mid \text{span}(S))\|_2^2$$

While similar in spirit to general low-rank approximation, some advantages with CSS include flexibility, interpretability and efficiency during inference. CSS is an unsupervised method and does not require labeled data, which is especially useful when labeled data is sparse. We note, on the other hand, unlabeled data is often very abundant and therefore scalable methods, like the one we present, are often needed. Furthermore, by subselecting features, as opposed to generating new features via an arbitrary function of the input features, we keep the semantic interpretation of the features intact. This is especially important in applications that require interpretable models. A third important advan-

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

tage is the efficiency of applying the solution CSS feature selection problem during inference. Compared to PCA or other methods that require a matrix-matrix multiplication to project input features into a reduced space during inference time, CSS only requires selecting a subset of feature values from a new instance vector. This is especially useful for latency sensitive applications and when the projection matrix itself may be prohibitively large, for example in restricted memory settings.

While there have been significant advances in CSS (Boutsidis et al., 2011; 2009; Guruswami & Sinop, 2012), most of the algorithms are either impractical and not applicable in a distributed setting for large datasets, or they do not have good (multiplicative $1-\varepsilon$) provable error bounds. Among efficient algorithms studied for the CSS problem is the simple *greedy algorithm*, which iteratively selects the best column and keeps it. Recent work shows that it does well in practice and even in a distributed setting (Farahat et al., 2011; 2013) and admits a performance guarantee (Çivril & Magdon-Ismail, 2012). However, the known guarantees depend on an arbitrarily large matrix-coherence parameter, which is unsatisfactory. Also, even though the algorithm is relatively fast, additional optimizations are needed to scale it to datasets with millions of features and instances.

1.1. Our contributions

Let $A \in \mathbb{R}^{m\times n}$ be the given matrix, and let k be the target number of columns. Let OPT_k denote the *optimal* set of columns, i.e., one that *covers* the maximum Frobenius mass of A. Our contributions are as follows.

Novel analysis of Greedy. For any $\varepsilon > 0$, we show that the natural greedy algorithm (Section 2), after $r = \frac{k}{\sigma_{\min}(OPT_k)\varepsilon}$ steps, gives an objective value that is within a $(1-\varepsilon)$ factor of the optimum. We also give a matching lower bound, showing that $\frac{k}{\sigma_{\min}(OPT_k)\varepsilon}$ is tight up to a constant factor. Here $\sigma_{\min}(OPT_k)$ is the smallest squared singular value of the *optimal* set of columns (after scaling to unit vectors).

Our result is similar in spirit to those of (Çivril & Magdon-Ismail, 2012; Boutsidis et al., 2015), but with an important difference. Their bound on r depends on the *least* $\sigma_{\min}(S)$ over *all* S of size k, while ours depends on $\sigma_{\min}(OPT_k)$. Note that these quantities can differ significantly. For instance, if the data has even a little bit of redundancy (e.g. few columns that are near duplicates), then there exist S for which $\sigma_{\min}$ is tiny, but the optimal set of columns could be reasonably well-conditioned (in fact, we would *expect* the optimal set of columns to be fairly well conditioned).

Distributed Greedy. We consider a natural distributed implementation of the greedy algorithm (Section 2). Here, we show that an interesting phenomenon occurs: even though partitioning the input does not work in general (as in coreset based algorithms), *randomly* partitioning works well. This is inspired by a similar result on submodular maximization (Mirrokni & Zadimoghaddam, 2015). Further, our result implies a 2-pass streaming algorithm for the CSS problem in the *random arrival* model for the columns.

We note that if the columns each have sparsity ϕ, (Boutsidis et al., 2016) gives an algorithm with total communication of $O(\frac{sk\phi}{\varepsilon} + \frac{sk^2}{\varepsilon^4})$. Their algorithm works for "worst case" partitioning of the columns into machines and is much more intricate than the greedy algorithm. In constrast, our algorithm is very simple, and for a random partitioning, the communication is just the first term above, along with an extra $\sigma_{\min}(OPT)$ term. Thus depending on $\sigma_{\min}$ and ε, each of the bounds could be better than the other.

Further optimizations. We also present techniques to speed up the implementation of the greedy algorithm. We show that the recent result of (Mirzasoleiman et al., 2015) (once again, on submodular optimization) can be extended to the case of CSS, improving the running time significantly.

We then compare our algorithms (in accuracy and running times) to various well-studied CSS algorithms. (Section 6.)

1.2. Related Work

The CSS problem is one of the central problems related to matrix approximation. Exact solution is known to be UG-hard (Çivril, 2014), and several approximation methods have been proposed over the years. Techniques such as importance sampling (Drineas et al., 2004; Frieze et al., 2004), adaptive sampling (Deshpande & Vempala, 2006), volume sampling (Deshpande et al., 2006; Deshpande & Rademacher, 2010), leverage scores (Drineas et al., 2008), and projection-cost preserving sketches (Cohen et al., 2015) have led to a much better understanding of the problem. (Guruswami & Sinop, 2012) gave the optimal dependence between column sampling and low-rank approximation. Due to the numerous applications, much work has been done on the implementation side, where adaptive sampling and leverage scores have been shown to perform well. A related, extremely simple algorithm is the greedy algorithm, which turns out to perform well and be scalable (Farahat et al., 2011; 2013). This was first analyzed by (Çivril & Magdon-Ismail, 2012), as we discussed.

There is also substantial literature about distributed algorithms for CSS (Pi et al., 2013; Feldman et al., 2015; Cohen et al., 2015; Farahat et al., 2015a;b; Boutsidis et al., 2016). In particular, (Farahat et al., 2015a;b) present distributed versions of the greedy algorithm based on MapReduce. Although they do not provide theoretical guarantees, their experimental results are very promising.

The idea of composable coresets has been applied explicitly or implicitly to several problems (Feldman et al., 2013; Balcan et al., 2013; Indyk et al., 2014). Quite recently,

for some problems in which coreset methods do not work in general, surprising results have shown that randomized variants of them give good approximations (da Ponte Barbosa et al., 2015; Mirrokni & Zadimoghaddam, 2015). We extend this framework to the CSS problem.

1.3. Background and Notation

We use the following notation throughout the paper. The set of integers $\{1, \ldots, n\}$ is denoted by $[n]$. For a matrix $A \in \mathbb{R}^{m \times n}$, A_j denotes the jth column ($A_j \in \mathbb{R}^m$). Given $S \subseteq [n]$, $A[S]$ denotes the submatrix of A containing columns indexed by S. The projection matrix Π_A projects onto the column span of A. Let $\|A\|_F$ denote the Frobenius norm, i.e., $\sqrt{\sum_{i,j} A_{i,j}^2}$. We write $\sigma_{\min}(A)$ to denote the minimum *squared* singular value, i.e., $\inf_{x:\|x\|_2=1} \frac{\|Ax\|_2^2}{\|x\|_2^2}$. We abuse notation slightly, and for a set of vectors V, we write $\sigma_{\min}(V)$ for the $\sigma_{\min}$ of the matrix with columns V.

Definition 1. *Given a matrix $A \in \mathbb{R}^{m \times n}$ and an integer $k \leq n$, the **Column Subset Selection (CSS) Problem** asks to find*

$$\arg\max_{S \subseteq [n], |S|=k} \|\Pi_{A[S]} A\|_F^2,$$

i.e., the set of columns that best explain *the full matrix A.*

We note that it is also common to cast this as a minimization problem, with the objective being $\|A - \Pi_{A[S]} A\|_F^2$. While the exact optimization problems are equivalent, obtaining multiplicative approximations for the minimization version could be harder when the matrix is low-rank.

For a set of vectors V and a matrix M, we denote

$$f_M(V) = \|\Pi_V M\|_F^2.$$

Also, the case when M is a single vector will be important. For any vector u, and a set of vectors V, we write

$$f_u(V) = \|\Pi_V u\|_2^2.$$

Remark 1. *Note that $f_M(V)$ can be viewed as the extent to which we can* cover *matrix M using vectors V. However, unlike combinatorial covering objectives, our definition is not submodular, or even subadditive. As an example, consider covering the following A using its own columns. Here, $f_A(\{A_1, A_2\}) = \|A\|_F^2 > f(\{A_1\}) + f(\{A_2\})$.*

$$A = \begin{pmatrix} 1 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

2. Greedy Algorithm for Column Selection

Let us state our algorithm and analysis in a slightly general form. Suppose we have two matrices A, B with the same number of rows and n_A, n_B columns respectively. The GCSS(A, B, k) problem is that of finding a subset S of columns of B, that maximizes $f_A(S)$ subject to $|S| = k$. Clearly, if $B = A$, we recover the CSS problem stated earlier. Also, note that scaling the columns of B will not affect the solution, so let us assume that the columns of B are all unit vectors. The greedy procedure iteratively picks columns of B as follows:

Algorithm 1 GREEDY($A \in \mathbb{R}^{m \times n_A}, B \in \mathbb{R}^{m \times n_B}, k \leq n_B$)

1: $S \leftarrow \emptyset$
2: **for** $i = 1 : k$ **do**
3: Pick column B_j that maximizes $f_A(S \cup B_j)$
4: $S \leftarrow S \cup \{B_j\}$
5: **end for**
6: Return S

Step (3) is the computationally intensive step in GREEDY – we need to find the column that gives the most *marginal gain*, i.e., $f_A(S \cup B_j) - f_A(S)$. In Section 5, we describe different techniques to speed up the calculation of marginal gain, while obtaining a $1-\varepsilon$ approximation to the optimum $f(\cdot)$ value. Let us briefly mention them here.

Projection to reduce the number of rows. We can left-multiply both A and B with an $r \times n$ Gaussian random matrix. For $r \geq \frac{k \log n}{\varepsilon^2}$, this process is well-known to preserve $f_A(\cdot)$, for any k-subset of the columns of B (see (Sarlos, 2006) or Appendix Section A.5 for details).

Projection-cost preserving sketches. Using recent results from (Cohen et al., 2015), we can project each *row* of A onto a random $O(\frac{k}{\varepsilon^2})$ dimensional space, and then work with the resulting matrix. Thus we may assume that the number of columns in A is $O(\frac{k}{\varepsilon^2})$. This allows us to efficiently compute $f_A(\cdot)$.

Lazier-than-lazy greedy. (Mirzasoleiman et al., 2015) recently proposed the first algorithm that achieves a constant factor approximation for maximizing submodular functions with a *linear* number of marginal gain evaluations. We show that a similar analysis holds for GCSS, even though the cost function is not submodular.

We also use some simple yet useful ideas from (Farahat et al., 2013) to compute the marginal gains (see Section 5).

2.1. Distributed Implementation

We also study a distributed version of the greedy algorithm, shown below (Algorithm 2.1). ℓ is the number of machines.

Algorithm 2 DISTGREEDY(A, B, k, ℓ)

1: *Randomly* partition the columns of B into $T_1, \ldots, T_\ell$
2: (Parallel) compute $S_i \leftarrow$ GREEDY$(A, T_i, \frac{32k}{\sigma_{\min}(OPT)})$
3: (Single machine) aggregate the S_i, and compute $S \leftarrow$ GREEDY$(A, \cup_{i=1}^{\ell} S_i, \frac{12k}{\sigma_{\min}(OPT)})$
4: Return $\arg\max_{S' \in \{S, S_1, \ldots, S_\ell\}} f_A(S')$

As mentioned in the introduction, the key here is that the partitioning is done *randomly*, in contrast to most results on *composable summaries*. We also note that machine i only sees columns T_i of B, but requires evaluating $f_A(\cdot)$ on the full matrix A when running GREEDY.[1] The way to implement this is again by using projection-cost preserving sketches. (In practice, keeping a small sample of the columns of A works as well.) The sketch is first passed to all the machines, and they all use it to evaluate $f_A(\cdot)$.

We now turn to the analysis of the single-machine and the distributed versions of the greedy algorithm.

3. Peformance analysis of GREEDY

The main result we prove is the following, which shows that by taking only slightly more than k columns, we are within a $1-\varepsilon$ factor of the optimal solution of size k.

Theorem 1. *Let $A \in \mathbb{R}^{m\times n_A}$ and $B \in \mathbb{R}^{m\times n_B}$. Let OPT_k be a set of columns from B that maximizes $f_A(S)$ subject to $|S| = k$. Let $\varepsilon > 0$ be any constant, and let T_r be the set of columns output by* GREEDY(A, B, r)*, for $r = \frac{16k}{\varepsilon\sigma_{\min}(OPT_k)}$. Then we have*

$$f_A(T_r) \geq (1-\varepsilon) f_A(OPT_k).$$

We show in Appendix Section A.3 that this bound is tight up to a constant factor, with respect to ε and $\sigma_{\min}(OPT_k)$. Also, we note that GCSS is a harder problem than MAX-COVERAGE, implying that if we can choose only k columns, it is impossible to approximate to a ratio better than $(1-\frac{1}{e}) \approx 0.63$, unless P=NP. (In practice, GREEDY does much better, as we will see.)

The basic proof strategy for Theorem 1 is similar to that of maximizing submodular functions, namely showing that in every iteration, the value of $f(\cdot)$ increases significantly. The key lemma is the following.

Lemma 1. *Let S, T be two sets of columns, with $f_A(S) \geq f_A(T)$. Then there exists $v \in S$ such that*

$$f_A(T\cup v) - f_A(T) \geq \sigma_{\min}(S)\frac{\big(f_A(S)-f_A(T)\big)^2}{4|S| f_A(S)}.$$

Theorem 1 follows easily from Lemma 1, which we show at the end of the section. Thus let us first focus on proving the lemma. Note that for submodular f, the analogous lemma simply has $\frac{f(S)-f(T)}{|S|}$ on the right-hand side (RHS). The main ingredient in the proof of Lemma 1 is its *single vector* version:

Lemma 2. *Let S, T be two sets of columns, with $f_u(S) \geq f_u(T)$. Suppose $S = \{v_1, \ldots, v_k\}$. Then*

$$\sum_{i=1}^{k}\Big(f_u(T\cup v_i) - f_u(T)\Big) \geq \sigma_{\min}(S)\frac{\big(f_u(S)-f_u(T)\big)^2}{4 f_u(S)}.$$

Let us first see why Lemma 2 implies Lemma 1. Observe that for any set of columns T, $f_A(T) = \sum_j f_{A_j}(T)$ (sum over the columns), by definition. For a column j, let us define $\delta_j = \min\{1, \frac{f_{A_j}(T)}{f_{A_j}(S)}\}$. Now, using Lemma 2 and plugging in the definition of δ_j, we have

$$\begin{aligned}
&\frac{1}{\sigma_{\min}(S)}\sum_{i=1}^{k}\big(f_A(T\cup v_i) - f_A(T)\big) && (1)\\
&= \frac{1}{\sigma_{\min}(S)}\sum_{j=1}^{n}\sum_{i=1}^{k}\big(f_{A_j}(T\cup v_i) - f_{A_j}(T)\big)\\
&\geq \sum_{j=1}^{n}\frac{(1-\delta_j)^2 f_{A_j}(S)}{4} && (2)\\
&= \frac{f_A(S)}{4}\sum_{j=1}^{n}(1-\delta_j)^2\frac{f_{A_j}(S)}{f_A(S)} && (3)\\
&\geq \frac{f_A(S)}{4}\left(\sum_{j=1}^{n}(1-\delta_j)\frac{f_{A_j}(S)}{f_A(S)}\right)^2 && (4)\\
&= \frac{1}{4f_A(S)}\Big(\sum_{j=1}^{n}\max\{0, f_{A_j}(S) - f_{A_j}(T)\}\Big)^2 && (5)\\
&\geq \frac{1}{4f_A(S)}\Big(f_A(S) - f_A(T)\Big)^2 && (6)
\end{aligned}$$

To get (4), we used Jensen's inequality ($\mathbb{E}[X^2] \geq (\mathbb{E}[X])^2$) treating $\frac{f_{A_j}(S)}{f_A(S)}$ as a probability distribution over indices j. Thus it follows that there exists an index i for which the gain is at least a $\frac{1}{|S|}$ factor, proving Lemma 1.

Proof of Lemma 2. Let us first analyze the quantity $f_u(T\cup v_i) - f_u(T)$, for some $v_i \in S$. As mentioned earlier, we may assume the v_i are normalized. If $v_i \in \text{span}(T)$, this quantity is 0. Thus we can assume that such v_i have been removed from S. Now, adding v_i to T gives a gain because of the component of v_i orthogonal to T, i.e., $v_i - \Pi_T v_i$, where Π_T denotes the projector onto span(T). Define

$$v_i' = \frac{v_i - \Pi_T v_i}{\|v_i - \Pi_T v_i\|_2}.$$

By definition, span$(T\cup v_i) =$ span$(T\cup v_i')$. Thus the projection of a vector u onto span$(T\cup v_i')$ is $\Pi_T u + \langle u, v_i'\rangle v_i'$, which is a vector whose squared length is $\|\Pi_T u\|^2 + \langle u, v_i'\rangle^2 = f_u(T) + \langle u, v_i'\rangle^2$. This implies that

$$f_u(T\cup v_i) - f_u(T) = \langle u, v_i'\rangle^2. \quad (7)$$

[1] It is easy to construct examples in which splitting both A and B fails badly.

Thus, to show the lemma, we need a lower bound on $\sum_i \langle u, v_i' \rangle^2$. Let us start by observing that a more explicit definition of $f_u(S)$ is the squared-length of the projection of u onto span(S), i.e. $f_u(S) = \max_{x \in \text{span}(S), \|x\|_2=1} \langle u, x \rangle^2$. Let $x = \sum_{i=1}^{k} \alpha_i v_i$ be a maximizer. Since $\|x\|_2 = 1$, by the definition of the smallest squared singular value, we have $\sum_i \alpha_i^2 \leq \frac{1}{\sigma_{\min}(S)}$. Now, decomposing $x = \Pi_T x + x'$, we have

$$f_u(S) = \langle x, u \rangle^2 = \langle x' + \Pi_T x,\ u \rangle^2 = (\langle x', u \rangle + \langle \Pi_T x, u \rangle)^2.$$

Thus (since the worst case is when all signs align),

$$|\langle x', u \rangle| \geq \sqrt{f_u(S)} - |\langle \Pi_T x, u \rangle| \geq \sqrt{f_u(S)} - \sqrt{f_u(T)} = \frac{f_u(S) - f_u(T)}{\sqrt{f_u(S)} + \sqrt{f_u(T)}} \geq \frac{f_u(S) - f_u(T)}{2\sqrt{f_u(S)}}. \quad (8)$$

where we have used the fact that $|\langle \Pi_T x, u \rangle|^2 \leq f_u(T)$, which is true from the definition of $f_u(T)$ (and since $\Pi_T x$ is a vector of length ≤ 1 in span(T)).

Now, because $x = \sum_i \alpha_i v_i$, we have $x' = x - \Pi_T x = \sum_i \alpha_i (v_i - \Pi_T v_i) = \sum_i \alpha_i \|v_i - \Pi_T v_i\|_2 v_i'$. Thus,

$$\begin{aligned} \langle x', u \rangle^2 &= \big(\sum_i \alpha_i \|v_i - \Pi_T v_i\|_2 \langle v_i', u \rangle \big)^2 \\ &\leq \big(\sum_i \alpha_i^2 \|v_i - \Pi_T v_i\|_2^2 \big) \big(\sum_i \langle v_i', u \rangle^2 \big) \\ &\leq \big(\sum_i \alpha_i^2 \big) \big(\sum_i \langle v_i', u \rangle^2 \big). \end{aligned}$$

where we have used Cauchy-Schwartz, and then the fact that $\|v_i - \Pi_T v_i\|_2 \leq 1$ (because v_i are unit vectors). Finally, we know that $\sum_i \alpha_i^2 \leq \frac{1}{\sigma_{\min}(S)}$, which implies

$$\sum_i \langle v_i', u \rangle^2 \geq \sigma_{\min}(S) \langle x', u \rangle^2 \geq \sigma_{\min}(S) \frac{(f_u(S) - f_u(T))^2}{4 f_u(S)}$$

Combined with (7), this proves the lemma. □

Proof of Theorem 1. For notational convenience, let $\sigma = \sigma_{min}(OPT_k)$ and $F = f_A(OPT_k)$. Define $\Delta_0 = F$, $\Delta_1 = \frac{\Delta_0}{2}$, …, $\Delta_{i+1} = \frac{\Delta_i}{2}$ until $\Delta_N \leq \varepsilon F$. Note that the gap $f_A(OPT_k) - f_A(T_0) = \Delta_0$. We show that it takes at most $\frac{8kF}{\sigma \Delta_i}$ iterations (i.e. additional columns selected) to reduce the gap from Δ_i to $\frac{\Delta_i}{2} = \Delta_{i+1}$. To prove this, we invoke Lemma 1 to see that the gap filled by $\frac{8kF}{\sigma \Delta_i}$ iterations is at least $\frac{8kF}{\sigma \Delta_i} \cdot \sigma \frac{(\frac{\Delta_i}{2})^2}{4kF} = \frac{\Delta_i}{2} = \Delta_{i+1}$. Thus the total number of iterations r required to get a gap of at most $\Delta_N \leq \varepsilon F$ is:

$$r \leq \sum_{i=0}^{N-1} \frac{8kF}{\sigma \Delta_i} = \frac{8kF}{\sigma} \sum_{i=0}^{N-1} \frac{2^{i-N+1}}{\Delta_{N-1}} < \frac{16k}{\varepsilon \sigma}.$$

where the last step is due to $\Delta_{N-1} > \varepsilon F$ and $\sum_{i=0}^{N-1} 2^{i-N+1} < 2$. Therefore, after $r < \frac{16k}{\varepsilon\sigma}$ iterations, we have $f_A(OPT_k) - f_A(T_r) \leq \varepsilon f_A(OPT_k)$. Rearranging proves the lemma. □

4. Distributed Greedy Algorithm

We will now analyze the distributed version of the greedy algorithm that was discussed earlier. We show that in one *round*, we will find a set of size $O(k)$ as before, that has an objective value $\Omega(f(OPT_k)/\kappa)$, where κ is a condition number (defined below). We also combine this with our earlier ideas to say that if we perform $O(\kappa/\varepsilon)$ *rounds* of DISTGREEDY, we get a $(1-\varepsilon)$ approximation (Theorem 3).

4.1. Analyzing one round

We consider an instance of GCSS(A, B, k), and let OPT denote an optimum set of k columns. Let ℓ denote the number of machines available. The columns (of B) are partitioned across machines, such that machine i is given columns T_i. It runs GREEDY as explained earlier and outputs $S_i \subset T_i$ of size $k' = \frac{32k}{\sigma_{\min}(OPT)}$. Finally, all the S_i are moved to one machine and we run GREEDY on their union and output a set S of size $k'' = \frac{12k}{\sigma_{\min}(OPT)}$. Let us define $\kappa(OPT) = \frac{\sigma_{\max}(OPT)}{\sigma_{\min}(OPT)}$.

Theorem 2. *Consider running* DISTGREEDY *on an instance of* GCSS(A, B, k). *We have*

$$\mathbb{E}[\max\{f_A(S), \max_i\{f_A(S_i)\}\}] \geq \frac{f(OPT)}{8 \cdot \kappa(OPT)}.$$

The key to our proof are the following definitions:

$$OPT_i^S = \{x \in OPT \ :\ x \in \text{GREEDY}(A, T_i \cup x, k')\}$$
$$OPT_i^{NS} = \{x \in OPT \ :\ x \notin \text{GREEDY}(A, T_i \cup x, k')\}$$

In other words, OPT_i^S contains all the vectors in OPT that would have been selected by machine i if they had been added to the input set T_i. By definition, the sets (OPT_i^S, OPT_i^{NS}) form a partition of OPT for every i.

Proof outline. Consider any partitioning $T_1, \ldots, T_\ell$, and consider the sets OPT_i^{NS}. Suppose one of them (say the ith) had a large value of $f_A(OPT_i^{NS})$. Then, we claim that $f_A(S_i)$ is also large. The reason is that the greedy algorithm does *not* choose to pick the elements of OPT_i^{NS} (by definition) – this can only happen if it ended up picking vectors that are "at least as good". This is made formal in Lemma 3. Thus, we can restrict to the case when *none* of $f_A(OPT_i^{NS})$ is large. In this case, Lemma 4 shows that $f_A(OPT_i^S)$ needs to be large for each i. Intuitively, it means that most of the vectors in OPT will, in fact, be picked by GREEDY (on the corresponding machines), and will be considered when computing S. The caveat is that we might be unlucky, and for every $x \in OPT$, it might have happened that it was sent to machine j for which it was not part of OPT_j^S. We show that this happens with low probability, and this is where the random partitioning is crucial (Lemma 5). This implies that either S, or one of the S_i has a large value of $f_A(\cdot)$.

Let us now state two lemmas, and defer their proofs to Sections A.1 and A.2 respectively.

Lemma 3. *For S_i of size $k' = \frac{32k}{\sigma_{\min}(OPT)}$, we have*

$$f(S_i) \geq \frac{f_A(OPT_i^{NS})}{2} \text{ for all } i.$$

Lemma 4. *For any matrix A, and any partition (I, J) of OPT:*

$$f_A(I) + f_A(J) \geq \frac{f_A(OPT)}{2\kappa(OPT)}. \tag{9}$$

Our final lemma is relevant when none of $f_A(OPT_i^{NS})$ are large and, thus, $f_A(OPT_i^S)$ is large for *all* i (due to Lemma 4). In this case, Lemma 5 will imply that the expected value of $f(S)$ is large.

Note that T_i is a random partition, so the T_i, the OPT_i^S, OPT_i^{NS}, S_i, and S are all random variables. However, all of these value are fixed given a partition $\{T_i\}$. In what follows, we will write $f(\cdot)$ to mean $f_A(\cdot)$.

Lemma 5. *For a random partitioning $\{T_i\}$, and S of size $k'' = \frac{12k}{\sigma_{\min}(OPT)}$, we have*

$$\mathbb{E}[f(S)] \geq \frac{1}{2}\mathbb{E}\left[\frac{\sum_{i=1}^{\ell} f(OPT_i^S)}{\ell}\right]. \tag{10}$$

Proof. At a high level, the intuition behind the analysis is that many of the vectors in OPT are selected in the first phase, i.e., in $\cup_i S_i$. For an $x \in OPT$, let I_x denote the indicator for $x \in \cup_i S_i$.

Suppose we have a partition $\{T_i\}$. Then if x had gone to a machine i for which $x \in OPT_i^S$, then by definition, x will be in S_i. Now the key is to observe (see definitions) that the event $x \in OPT_i^S$ does not depend on where x is in the partition! In particular, we could think of partitioning all the elements except x (and at this point, we know if $x \in OPT_i^S$ for all i), and *then* randomly place x. Thus

$$\mathbb{E}[I_x] = \mathbb{E}\left[\frac{1}{\ell}\sum_{i=1}^{\ell}[[x \in OPT_i^S]]\right], \tag{11}$$

where $[[\ \cdot\ ]]$ denotes the indicator.

We now use this observation to analyze $f(S)$. Consider the execution of the greedy algorithm on $\cup_i S_i$, and suppose V^t denotes the set of vectors picked at the tth step (so V^t has t vectors). The main idea is to give a lower bound on

$$\mathbb{E}[f(V^{t+1}) - f(V^t)], \tag{12}$$

where the expectation is over the partitioning $\{T_i\}$. Let us denote by Q the RHS of (10), for convenience. Now, the trick is to show that for *any* V^t such that $f(V^t) \leq Q$, the expectation in (12) is large. One lower bound on $f(V^{t+1}) - f(V^t)$ is (where I_x is the indicator as above)

$$\frac{1}{k}\sum_{x \in OPT} I_x\big(f(V^t \cup x) - f(V^t)\big).$$

Now for every V, we can use (11) to obtain

$$\begin{aligned}
&\mathbb{E}[f(V^{t+1}) - f(V^t) | V^t = V] \\
&\quad \geq \frac{1}{k\ell}\sum_{x \in OPT}\mathbb{E}\left[\sum_{i=1}^{\ell}[[x \in OPT_i^S]]\right]\big(f(V \cup x) - f(V)\big) \\
&\quad = \frac{1}{k\ell}\mathbb{E}\left[\sum_{i=1}^{\ell}\sum_{x \in OPT_i^S}\big(f(V \cup x) - f(V)\big)\right].
\end{aligned}$$

Now, using (1)-(5), we can bound the inner sum by

$$\sigma_{\min}(OPT_i^S)\frac{(\max\{0, f(OPT_i^S) - f(V)\})^2}{4f(OPT_i^S)}.$$

Now, we use $\sigma_{\min}(OPT_i^S) \geq \sigma_{\min}(OPT)$ and the identity that for any two nonnegative reals a, b: $(\max\{0, a - b\})^2/a \geq a/2 - 2b/3$. Together, these imply

$$\begin{aligned}
&\mathbb{E}[f(V^{t+1}) - f(V^t) | V^t = V] \\
&\quad \geq \frac{\sigma_{\min}(OPT)}{4k\ell}\mathbb{E}\left[\sum_{i=1}^{\ell}\frac{f(OPT_i^S)}{2} - \frac{2f(V)}{3}\right].
\end{aligned}$$

and consequently: $\mathbb{E}[f(V^{t+1}) - f(V^t)] \geq \alpha(Q - \frac{2}{3}\mathbb{E}[f(V^t)]$ for $\alpha = \sigma_{\min}(OPT)/4k$. If for some t, we have $\mathbb{E}[f(V^t)] \geq Q$, the proof is complete because f is monotone, and $V^t \subseteq S$. Otherwise, $\mathbb{E}[f(V^{t+1}) - f(V^t)]$ is at least $\alpha Q/3$ for each of the $k'' = 12k/\sigma_{\min}(OPT) = 3/\alpha$ values of t. We conclude that $\mathbb{E}[f(S)]$ should be at least $(\alpha Q/3) \times (3/\alpha) = Q$ which completes the proof. □

Proof of Theorem 2. If $f_A(OPT_i^{NS}) \geq \frac{f(OPT)}{4\kappa(OPT)}$ for some i, then we are done, because Lemma 3 implies that $f_A(S_i)$ is large enough. Otherwise, by Lemma 4, $f_A(OPT_i^S) \geq \frac{f(OPT)}{4\kappa(OPT)}$ for all i. Now we can use Lemma 5 to conclude that $\mathbb{E}[f_A(S)] \geq \frac{f(OPT)}{8\kappa(OPT)}$, completing the proof. □

4.2. Multi-round algorithm

We now show that repeating the above algorithm helps achieve a $(1 - \varepsilon)$-factor approximation.

We propose a framework with r epochs for some integer $r > 0$. In each epoch $t \in [r]$, we run the DISTGREEDY algorithm to select set S^t. The only thing that changes in different epochs is the objective function: in epoch t, the algorithm selects columns based on the function f^t which is defined to be: $f^t(V) = f_A(V \cup S^1 \cup S^2 \cdots \cup S^{t-1})$ for any t. We note that function f^1 is indeed the same as f_A. The final solution is the union of solutions: $\cup_{t=1}^{r} S^t$.

Theorem 3. *For any $\varepsilon < 1$, the expected value of the solution of the r-epoch* DISTGREEDY *algorithm, for $r = O(\kappa(OPT)/\epsilon)$, is at least $(1-\varepsilon)f(OPT)$.*

The proof is provided in Section A.7 of the appendix.

Necessity of Random Partitioning. We point out that the random partitioning step of our algorithm is crucial for the GCSS(A, B, k) problem. We adapt the instance from (Indyk et al., 2014) and show that even if each machine can compute $f_A(\cdot)$ exactly, and is allowed to output poly(k) columns, it cannot compete with the optimum. Intuitively, this is because the partition of the columns in B could ensure that in each partition i, the best way of covering A involve picking some vectors S_i, but the S_i's for different i could overlap heavily, while the global optimum should use different i to capture different *parts* of the space to be covered. (See Theorem 8 in Appendix A.8 for details.)

5. Further optimizations for GREEDY

We now elaborate on some of the techniques discussed in Section 2 for improving the running time of GREEDY. We first assume that we left-multiply both A and B by a random Gaussian matrix of dimension $r \times m$, for $r \approx k \log n/\varepsilon^2$. Working with the new instance suffices for the purposes of $(1-\varepsilon)$ approximation to CSS (for picking $O(k)$ columns). (Details in the Appendix, Section A.5)

5.1. Projection-Cost Preserving Sketches

Marginal gain evaluations of the form $f_A(S \cup v) - f_A(S)$ require summing the marginal gain of v onto each column of A. When A has a large number of columns, this can be very expensive. To deal with this, we use a *sketch* of A instead of A itself. This idea has been explored in several recent works; we use the following notation and result:

Definition 2 ((Cohen et al., 2015)). *For a matrix $A \in \mathbb{R}^{m\times n}$, $A' \in \mathbb{R}^{m\times n'}$ is a* rank-k Projection-Cost Preserving Sketch (PCPS) *with error $0 \le \varepsilon < 1$ if for any set of k vectors S, we have: $(1-\varepsilon)f_A(S) \le f_{A'}(S) + c \le (1+\varepsilon)f_A(S)$ where $c \ge 0$ is a constant that may depend on A and A' but is independent of S.*

Theorem 4. *[Theorem 12 of (Cohen et al., 2015)] Let R be a random matrix with n rows and $n' = O(\frac{k+\log\frac{1}{\delta}}{\varepsilon^2})$ columns, where each entry is set independently and uniformly to $\pm\sqrt{\frac{1}{n'}}$. Then for any matrix $A \in \mathbb{R}^{m\times n}$, with probability at least $1 - O(\delta)$, AR is a rank-k PCPS for A.*

Thus, we can use PCPS to sketch the matrix A to have roughly k/ε^2 columns, and use it to compute $f_A(S)$ to a $(1 \pm \varepsilon)$ accuracy for any S of size $\le k$. This is also used in our distributed algorithm, where we send the sketch to every machine.

5.2. Lazier-than-lazy Greedy

The natural implementation of GREEDY requires $O(nk)$ evaluations of $f(\cdot)$ since we compute the marginal gain of all n candidate columns in each of the k iterations. For submodular functions, one can do better: the recently proposed LAZIER-THAN-LAZY GREEDY algorithm obtains a $1-\frac{1}{e}-\delta$ approximation with only a linear number $O(n\log(1/\delta))$ of marginal gain evaluations (Mirzasoleiman et al., 2015). We show that a similar result holds for GCSS, even though our cost function $f(\cdot)$ is not submodular.

The idea is as follows. Let T be the current solution set. To find the next element to add to T, we draw a sized $\frac{n_B \log(1/\delta)}{k}$ subset uniformly at random from the columns in $B \setminus T$. We then take from this set the column with largest marginal gain, add it to T, and repeat. We show this gives the following guarantee (details in Appendix Section A.4.)

Theorem 5. *Let $A \in \mathbb{R}^{m\times n_A}$ and $B \in \mathbb{R}^{m\times n_B}$. Let OPT_k be the set of columns from B that maximizes $f_A(S)$ subject to $|S| = k$. Let $\varepsilon, \delta > 0$ be any constants such that $\epsilon + \delta \le 1$. Let T_r be the set of columns output by* LAZIER-THAN-LAZY GREEDY(A, B, r)*, for $r = \frac{16k}{\varepsilon\sigma_{\min}(OPT_k)}$. Then we have:*

$$\mathbb{E}[f_A(T_r)] \ge (1-\varepsilon-\delta)f_A(OPT_k)$$

Further, this algorithm evaluates marginal gain only a linear number $\frac{16n_B\log(1/\delta)}{\varepsilon\sigma_{\min}(OPT_k)}$ of times.

Note that this guarantee is nearly identical to our analysis of GREEDY in Theorem 1, except that it is in expectation. The proof strategy is very similar to that of Theorem 1, namely showing that the value of $f(\cdot)$ increases significantly in every iteration (see Appendix Section A.4).

Calculating marginal gain faster. We defer the discussion to Appendix Section A.6.

6. Experimental results

In this section we present an empirical investigation of the GREEDY, GREEDY++ and DISTGREEDY algorithms. Additionally, we will compare with several baselines:

Random: The simplest imaginable baseline, this method selects columns randomly.

2-Phase: The two-phased algorithm of (Boutsidis et al., 2009), which operates by first sampling $\Theta(k\log k)$ columns based on properties of the top-k right singular space of the input matrix (this requires computing a top-k SVD), then finally selects exactly k columns via a deterministic procedure. The overall complexity is dominated by the top-k SVD, which is $O(\min\{mn^2, m^2n\})$.

PCA: The columns of the rank-k PCA projection matrix will be used to serve as an upper bound on performance, as they explicitly minimize the Forbenius reconstruction criteria. Note this method only serves as an upper bound and

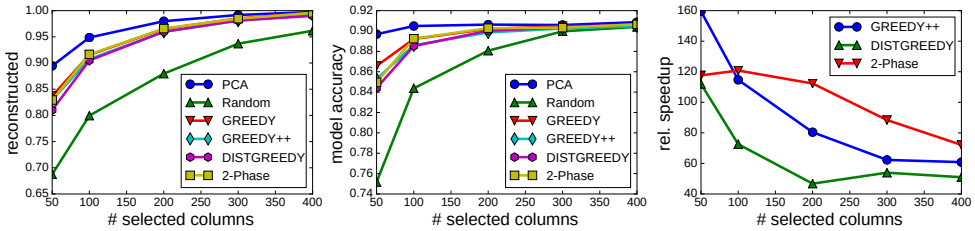

Figure 1. A comparison of reconstruction accuracy, model classification accuracy and runtime of various column selection methods (with PCA proved as an upper bound). The runtime is shown plot shows the relative speedup over the naive GREEDY algorithm.

does not fall into the framework of column subset selection.

We investigate using these algorithms using two datasets, one with a small set of columns (mnist) that is used to compare both scalable and non-scalable methods, as well as a sparse dataset with a large number of columns (news20.binary) that is meant to demonstrate the scalability of the GREEDY core-set algorithm.[2]

Finally, we are also interested in the effect of column selection as a preprocessing step for supervised learning methods. To that end, we will train a linear SVM model, using the LIBLINEAR library (Fan et al., 2008), with the subselected columns (features) and measure the effectiveness of the model on a held out test set. For both datasets we report test error for the best choice of regularization parameter $c \in \{10^{-3}, \ldots, 10^{4}\}$. We run GREEDY++ and DISTGREEDY with $\frac{n}{k}\log(10)$ marginal gain evaluations per iteration and the distributed algorithm uses $s = \sqrt{\frac{n}{k}}$ machines with each machine recieving $\frac{n}{s}$ columns.

6.1. Small scale dataset (mnist)

We first consider the MNIST digit recognition task, which is a ten-class classification problem. There are n = 784 input features (columns) that represent pixel values from the 28×28-pixel images. We use m = 60,000 instances to train with and 10,000 instances for our test set.

From Figure 1 we see that all column sampling methods, apart from Random, select columns that approximately provide the same amount of reconstruction and are able to reach within 1% of the performance of PCA after sampling 300 columns. We also see a very similar trend with respect to classification accuracy. It is notable that, in practice, the core-set version of GREEDY incurs almost no additional error (apart from at the smallest values of k) when compared to the standard GREEDY algorithm.

Finally, we also show the relative speed up of the competitive methods over the standard GREEDY algorithm. In this small dataset regime, we see that the core-set algorithm does not offer an improvement over the single machine GREEDY++ and in fact the 2-Phase algorithm is the fastest. This is primarily due to the overhead of the distributed core-set algorithm and the fact that it requires two greedy selection stages (e.g. map and reduce). Next, we will consider a dataset that is large enough that a distributed model is in fact necessary.

[2]Both datasets can be found at: www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/multiclass.html.

n	Rand	2-Phase	DISTGREEDY	PCA
500	54.9	81.8 (1.0)	80.2 (72.3)	85.8 (1.3)
1000	59.2	84.4 (1.0)	82.9 (16.4)	88.6 (1.4)
2500	67.6	87.9 (1.0)	85.5 (2.4)	90.6 (1.7)

Table 1. A comparison of the classification accuracy of selected features. Also, the relative speedup over the 2-Phase algorithm for selecting features is shown in parentheses.

6.2. Large scale dataset (news20.binary)

In this section, we show that the DISTGREEDY algorithm can indeed scale to a dataset with a large number of columns. The news20.binary dataset is a binary class text classification problem, where we start with n = 100,000 sparse features (0.033% non-zero entries) that represent text trigrams, use m = 14,996 examples to train with and hold-out 5,000 examples to test with.

We compare the classification accuracy and column selection runtime of the naive random method, 2-Phase algorithm as well as PCA (that serves as an upper bound on performance) to the DISTGREEDY algorithm. The results are presented in Table 1, which shows that DISTGREEDY and 2-Phase both perform significantly better than random sampling and come relatively close to the PCA upper bound in terms of accuracy. However, we also find that DISTGREEDY can be magnitudes of order faster than the 2-Phase algorithm. This is in a large part because the 2-Phase algorithm suffers from the bottleneck of computing a top-k SVD. We note that an approximate SVD method could be used instead, however, it was outside the scope of this preliminary empirical investigation.

In conclusion, we have demonstrated that DISTGREEDY is able to scale to larger sized datasets while still selecting effective features.

References

Balcan, Maria-Florina, Ehrlich, Steven, and Liang, Yingyu. Distributed k-means and k-median clustering on general communication topologies. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pp. 1995–2003, 2013.

Boutsidis, C., Mahoney, M. W., and Drineas, P. An improved approximation algorithm for the column subset selection problem. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pp. 968–977, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

Boutsidis, C., Drineas, P., and Magdon-Ismail, M. Near optimal column-based matrix reconstruction. In *Proceedings of the 2011 IEEE 52Nd Annual Symposium on Foundations of Computer Science*, FOCS '11, pp. 305–314, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-0-7695-4571-4.

Boutsidis, C., Liberty, E., and Sviridenko, M. Greedy minimization of weakly supermodular set functions. *CoRR*, abs/1502.06528, 2015.

Boutsidis, C., Woodruff, D. P., and Zhong, P. Communication-optimal distributed principal component analysis in the column-partition model. STOC '16 (to appear). ACM, 2016.

Çivril, A. Column subset selection problem is ug-hard. *J. Comput. Syst. Sci.*, 80(4):849–859, June 2014. ISSN 0022-0000.

Çivril, A. and Magdon-Ismail, M. Column subset selection via sparse approximation of svd. *Theor. Comput. Sci.*, 421:1–14, March 2012. ISSN 0304-3975.

Cohen, M. B., Elder, S., Musco, C., Musco, C., and Persu, M. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, STOC '15, pp. 163–172, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3536-2.

da Ponte Barbosa, Rafael, Ene, Alina, Nguyen, Huy L., and Ward, Justin. The power of randomization: Distributed submodular maximization on massive datasets. *CoRR*, abs/1502.02606, 2015.

Deshpande, A. and Rademacher, L. Efficient volume sampling for row/column subset selection. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS '10, pp. 329–338, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4244-7.

Deshpande, A. and Vempala, S. Adaptive sampling and fast low-rank matrix approximation. In *Proceedings of the 9th International Conference on Approximation Algorithms for Combinatorial Optimization Problems, and 10th International Conference on Randomization and Computation*, APPROX'06/RANDOM'06, pp. 292–303, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-38044-2, 978-3-540-38044-3.

Deshpande, A., Rademacher, L., Vempala, S., and Wang, G. Matrix approximation and projective clustering via volume sampling. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '06, pp. 1117–1126, Philadelphia, PA, USA, 2006. Society for Industrial and Applied Mathematics. ISBN 0-89871-605-5.

Drineas, P., Frieze, A., Kannan, R., Vempala, S., and Vinay, V. Clustering large graphs via the singular value decomposition. *Mach. Learn.*, 56(1-3):9–33, June 2004. ISSN 0885-6125.

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

Fan, Rong-En, Chang, Kai-Wei, Hsieh, Cho-Jui, Wang, Xiang-Rui, and Lin, Chih-Jen. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

Farahat, A. K., Ghodsi, A., and Kamel, M. S. An efficient greedy method for unsupervised feature selection. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining*, ICDM '11, pp. 161–170. IEEE Computer Society, 2011. ISBN 978-0-7695-4408-3.

Farahat, A. K., Ghodsi, A., and Kamel, M. S. A fast greedy algorithm for generalized column subset selection. In *Advances in Neural Information Processing Systems 27*, 2013.

Farahat, A. K., Elgohary, A., Ghodsi, A., and Kamel, M. S. Greedy column subset selection for large-scale data sets. *Knowl. Inf. Syst.*, 45(1):1–34, October 2015a. ISSN 0219-1377.

Farahat, A. K., Elgohary, A., Ghodsi, A., and Kamel, M. S. Greedy column subset selection for large-scale data sets. *Knowl. Inf. Syst.*, 45(1):1–34, October 2015b. ISSN 0219-1377.

Feldman, D., Schmidt, M., and Sohler, C. Turning big data into tiny data: Constant-size coresets for *k*-means, PCA and projective clustering. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pp. 1434–1453, 2013.

Feldman, D., Volkov, M., and Rus, D. Dimensionality reduction of massive sparse datasets using coresets. *CoRR*, abs/1503.01663, 2015.

Frieze, A., Kannan, R., and Vempala, S. Fast monte-carlo algorithms for finding low-rank approximations. *J. ACM*, 51(6):1025–1041, November 2004. ISSN 0004-5411.

Guruswami, V. and Sinop, A. K. Optimal column-based low-rank matrix reconstruction. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pp. 1207–1214. SIAM, 2012.

Guyon, I. and Elisseeff, A. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, March 2003. ISSN 1532-4435.

Indyk, Piotr, Mahabadi, Sepideh, Mahdian, Mohammad, and Mirrokni, Vahab S. Composable core-sets for diversity and coverage maximization. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014*, pp. 100–108, 2014.

Johnson, W. and Lindenstrauss, J. Extensions of Lipschitz mappings into a Hilbert space. In *Conference in modern analysis and probability (New Haven, Conn., 1982)*, volume 26 of *Contemporary Mathematics*, pp. 189–206. American Mathematical Society, 1984.

Mirrokni, V. and Zadimoghaddam, M. Randomized composable core-sets for distributed submodular maximization. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, STOC '15, pp. 153–162, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3536-2.

Mirzasoleiman, B., Badanidiyuru, A., Karbasi, A., Vondrák, J., and Krause, A. Lazier than lazy greedy. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pp. 1812–1818, 2015.

Pi, Y., Peng, H., Zhou, S., and Zhang, Z. A scalable approach to column-based low-rank matrix approximation. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pp. 1600–1606. AAAI Press, 2013. ISBN 978-1-57735-633-2.

Rudelson, M. and Vershynin, R. Non-asymptotic theory of random matrices: extreme singular values. *Proceedings of the International Congress of Mathematicians*, III:1576–1602, 2010.

Sarlos, T. Improved approximation algorithms for large matrices via random projections. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '06, pp. 143–152, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2720-5.

Vershynin, R. Introduction to the non-asymptotic analysis of random matrices, 2010.