Non-Negative Sparse Regression and Column Subset Selection with L_1 Error

Aditya Bhaskara[*1] **and Silvio Lattanzi**[2]

1 **School of Computing, University of Utah, Salt Lake City, UT, USA**
bhaskara@cs.utah.edu
2 **Google Research, Zurich, Switzerland**
silviol@google.com

Abstract

We consider the problems of sparse regression and column subset selection under ℓ_1 error. For both problems, we show that in the non-negative setting it is possible to obtain tight and efficient approximations, without any additional structural assumptions (such as restricted isometry, incoherence, expansion, etc.). For sparse regression, given A, b with non-negative entries, we give an efficient algorithm to output a vector x of sparsity $O(k)$, for which $\|Ax - b\|_1$ is comparable to the smallest error possible using non-negative k-sparse x. We then use this technique to obtain our main result: an efficient algorithm for column subset selection under ℓ_1 error for non-negative matrices.

1998 ACM Subject Classification F.2.0 Analysis of Algorithms and Problem Complexity

Keywords and phrases Sparse regression, L1 error optimization, Column subset selection

Digital Object Identifier 10.4230/LIPIcs.ITCS.2018.7

1 Introduction

Sparsity plays a crucial role in learning and signal processing. Representing a signal as a sparse combination of "elementary" signals (sparse recovery) and finding bases in which a collection of signals have sparse representation (sparse coding) are fundamental problems, with applications ranging from genetics, to speech processing, to computer vision [12, 33, 31, 32, 38, 40, 41].

In most of these applications, we require recovery algorithms that are tolerant to noise. Different "types" of noise lead to different optimization formulations. For instance, if signals are corrupted under independent Gaussian noise, recovery algorithms with an ℓ_2 objective perform well. On the other hand, when there are a few yet large deviations, recovery algorithms with an ℓ_1 objective perform much better (cf. [13, 26, 42]). This is also the case for problems involving matrices, such as low rank approximation and column subset selection, in which we could either have uniform noise in all the columns, or have a few "outlier" columns with large error. Also, from a theoretical perspective, it is interesting to ask the approximation question for general ℓ_p norms. Interestingly, for the matrix problems above, minimizing the ℓ_1 error (and more generally ℓ_p error for $p \neq 2$) turns out to be significantly harder than minimizing the ℓ_2 error. The beautiful theory of singular value decompositions that lets us minimize ℓ_2 error does not have a counter-part for ℓ_1. Indeed, even finding approximate solutions has recently been shown to be NP hard [36]. However, given its effectiveness in applications ([31, 32, 38, 41]), many heuristics [27, 37, 43, 11, 20] and, more recently, approximation algorithms [15, 36] have been proposed.

* Partially supported by a Google Research Faculty Award.


© A. Bhaskara and S. Lattanzi;
licensed under Creative Commons License CC-BY

Our main focus in this paper is the well-studied problem of *column selection*, with the goal of minimizing the ℓ_1 reconstruction error. The column selection problem is the following: we are given a matrix A ($m \times n$), and the goal is to find a subset S of its columns of a prescribed size, so as to minimize the "reconstruction error" $\min_{X \in \mathbb{R}^{|S| \times n}} \|A - A_S X\|_1$, where A_S is the submatrix of A restricted to the columns S. Besides its direct applications [12, 31, 32, 38, 40, 41] column selection has found several applications as a tool for building efficient machine learning algorithms, including feature selection (as a pre-processing step for learning), coreset computation, and more broadly, as *interpretable* dimension reduction [1, 16, 21].

While column selection has been quite well understood with ℓ_2 error (in a sequence of works, including [10, 23, 25, 34]), it is computationally much harder under ℓ_1 error. Recently, Song et al. [36] provided the first CUR and low rank approximation algorithm for general matrices and for $p \in [1, 2]$. Their main result is an $(O(\log m)\,\mathrm{poly}(k))$-approximation to the error in $\mathrm{nnz}(A) + (n + m)\,\mathrm{poly}(k)$ time, for every k, where $\mathrm{nnz}(A)$ is the number of non-zero entries in A. They also prove that if one compares the error obtained by the column approximation with the best rank-k approximation, a factor of $\sqrt{k}$ is inevitable, even if one uses significantly more than k columns. Our result is incomparable in two respects: first, it gets past the lower bound by comparing the solution obtained with the error of the best k-column approximation. Second, our focus is on the non-negative setting (defined below), for which we obtain a significantly better approximation. A more detailed comparision with the works of Song et al. [36] and Chierichetti et al. [15] is provided in Section 1.2.

It is important to note that the non-negativity assumption is natural, in fact in many applications where column selection is used to "explain" a collection of points in terms of a smaller subset, we have that all the points are expressible as a *non-negative* (often even convex) combination of the selected points. Further, in applications such as recommender systems and image reconstruction, the points themselves have coordinates that are all non-negative. Motivated by such applications, we define the *non-negative column subset selection* problem, as follows.

▶ **Problem 1** (Non-negative CSS(A, B))**.** *Given two matrices A, B with non-negative entries and parameter k, find a subset S of the columns of A of size k, so as to minimize the best reconstruction error for B, i.e., the quantity* $\min_{X \geq 0} \|B - A_S X\|_1$. *($X \geq 0$ refers to entry-wise non-negativity.)*

Note that this is a slight generalization of column selection as explained earlier (which can be viewed as the case $B = A$). Our main result in this paper is an efficient algorithm for this problem; enroute to this we also design an algorithm for the ℓ_1 sparse regression problem.

Sparse regression

The key ingredient in our result is a new algorithm for another fundamental problem – sparse regression. Given a matrix A, and a target vector b, the goal here is to find a sparse x such that $\|Ax - b\|_1$ is as small as possible. The classic paper of Donoho [19], and the beautiful line of work [5, 14, 24, 30] have resulted in algorithms for approximate recovery, under special assumptions on A (for instance, the so-called restricted isometry property (RIP), or expansion properties of an appropriate graph associated with A). Here we solve the general version of the non-negative problem.

It is also known that sparse recovery is hard if we do not make assumptions on A [2, 22] (incidentally, even checking if a matrix satisfies the RIP condition is hard, in many interesting parameter ranges [6, 28, 39]). So our focus here is on a non-negative variant, defined as follows.

▶ **Problem 2** (Non-negative sparse regression). *Given a non-negative matrix A, a non-negative* target *vector b, and a parameter k, find a vector $x \geq 0$ (entry-wise) of sparsity k that minimizes $\|Ax - b\|_1$.*

Finding sparse solutions to optimization problems is a classic theme in approximation theory. A classic approach to solve those problems for a large class of convex functions $f(x)$ over the domain $\Delta_n := \{(x_1, \ldots x_n) \ : \ x_i \geq 0, \sum_i x_i = 1\}$ is the Frank-Wolfe procedure that produces a solution that is $O(1/T)$ away from the optimum, after T iterations. This method can be used to obtain a trade-off between the approximation of the objective and the sparsity of the solution obtained. [17, 18, 35] are three excellent sources for this line of work. For ℓ_2 sparse regression, this implies that for *any* A, we can obtain a solution sparsity roughly $O(1/\epsilon^2)$, that has a loss ϵ away from the optimum (Maurey's lemma, see also [7]).

However, such a result is impossible with ℓ_1 error. It is easy to see that if A is the identity, and $b = (\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n})$, then the minimum over $x \in \Delta_n$ of $f(x) = \|Ax - b\|_1$ is zero, while for any k-sparse x, the error is $\geq 1 - \frac{k}{n}$. Thus, we ask the question: can we perform such an approximation, when the goal is simply to *compete* with the best k-sparse solution? Our contribution in Theorem 3 is to show that this is possible via a Frank-Wolfe type update using a novel potential function.

A setting very similar to ours was considered in the work [8], where it is shown that if there is a k-sparse vector x^* such that $Ax^* = b$ *exactly*, then an algorithm based on an exponential potential function finds an $O(k/\epsilon^3)$-sparse vector y such that $\|Ay - b\|_1 \leq \epsilon$.[1] The paper [8] uses sparse regression for learning low-dimensional Gaussian mixtures (i.e., express the p.d.f. of the mixture —obtained empirically— as a sparse convex combination of the p.d.f.'s of Gaussians). Our ability to handle error implies that our algorithm can learn mixtures even in the presence of noisy points.

Further related work. Problem 1 above is closely related to many well-studied questions. Blum et al. [9] consider the problem of finding a small subset Q of a given point set P, such that the convex hulls of P and Q are "close". This is equivalent to approximately representing the points in $P \setminus Q$ using points in Q – a goal similar to ours. However, they consider ℓ_2 error, and also require a good approximation for *every* point in P. Another closely related question is that of finding a non-negative matrix factorization (NMF), under the "anchor word" assumption [4, 3]. In NMF, we are given a non-negative matrix M, and the goal is to write $M = XY$, where X, Y are non-negative matrices. The anchor word assumption states that X can be chosen to be a subset of the columns of M, which reduces the problem to non-negative CSS. Our methods can thus be directly applied; however in [3] and related works, the measures of error are different from ours. Also in contrast to our result, much of the work in this area focuses on finding precisely k columns.

Notations. We review some of the notation we will use throughout the paper. For a matrix A, A_i refers to its i'th column, and $A^{(i)}$ refers to its ith row. By $\|A\|_1$, we refer to the sum of the absolute values of the entries in A. Also, $\text{nnz}(A)$ refers to the number of non-zero entries in the matrix. We will refer to Δ_n the "probability simplex" in n dimensions, namely $\{(x_1, \ldots, x_n) \ : \ x_i \geq 0, \sum_i x_i = 1\}$.

[1] Their *proof* can handle a small amount of noise, albeit under the restriction that in *every coordinate*, Ax^* and b are within a $(1 \pm \epsilon)$ factor. Note that is a lot more restrictive than $\|Ax^* - b\|_1$ being small.

1.1 Our results

Let us start by stating our result for sparse regression. The non-negative sparse regression problem (stated above) has as input a matrix A and a target vector b (both non-negative).

▶ **Theorem 3.** *Suppose there exists a non-negative k-sparse vector x^* such that $\|Ax^* - b\|_1 \leq \epsilon \|b\|_1$. Then, there is an efficient algorithm that, for any $\delta > 0$, outputs a y of sparsity $O(k \log(1/\delta)/\delta^2)$, with the guarantee that $\|Ay - b\|_1 \leq \left(4\sqrt{2(\epsilon + 2\delta)}\right) \|b\|_1$.*

The running time of the algorithm is $O(k \log(1/\delta)/\delta^2 \cdot \text{nnz}(A))$. Note also that if we set $\delta = \epsilon$, we obtain an error roughly $O(\sqrt{\epsilon})$ factor of the optimum. It is an interesting open problem to understand if this ϵ versus $\sqrt{\epsilon}$ guarantee is *necessary*. In our analysis, it arises due to a move from KL divergence to ℓ_1 error (via Pinsker's inequality). Indeed, our algorithm produces a much better approximation in KL divergence, as we will see. We then use the theorem above to show our main result, on the non-negative CSS problem.

▶ **Theorem 4.** *Let A, B be non-negative matrices, and suppose there exists a subset S of k columns of A, such that $\min_{X \geq 0} \|B - A_S X\|_1 \leq \epsilon \|B\|_1$. Then for any $\delta > 0$, there is an efficient algorithm that finds a set S' of $O(k \log(1/\delta)/\delta^2)$ columns of A that give an approximation error $O(\sqrt{\delta + \epsilon}) \|B\|_1$.*

Theorems 3 and 4 are proved in sections 2 and 3 respectively. Given the conceptual simplicity of the algorithms, we also implement them effectively, and show some preliminary experimental results in Appendix A.

1.2 Interpreting error bounds and comparisons to prior work

We now focus on the low-rank approximation result (Theorem 4) and discuss the tradeoff between error and the number of columns output. We then compare our result with prior work on ℓ_1 low rank approximation.

First, note that our error bound is *additive*, in a sense. Specifically, the approximation factor can be arbitrarily bad if ϵ is sufficiently small. Indeed, if $\epsilon = o(1)/k$, then prior work gives better approximations. While it is common in theory to assume that low-rank and k-column approximations have error that is *tiny* compared to the norm of the matrix, in practice it is quite common to have situations in which only (say) 90% of the "mass" can be "explained" via a low rank approximation, while the rest is noise. These are the settings in which our methods do significantly better (indeed, none of the earlier results we are aware of give any non-trivial guarantees in such settings). In the case of ℓ_2 error (when we can actually compute the error efficiently), we often observe a drop in the singular *values* (i.e., $\sigma_1, \ldots, \sigma_k$ are larger than the rest) in practice, but the total Frobenius mass on the *tail* is still non-trivial.

Second, we have a tradeoff between the number of columns output and the error we can obtain. We do not know if such a dependence is optimal, and this is an interesting open question. However, in [8], it was shown that even in the zero error case, a trade-off of this nature is essential, assuming that a random planted version of the set cover problem does not have polynomial time algorithms (which seems consistent with current algorithmic techniques).

Next, we compare with the previous work on ℓ_1 low rank approximation. Recently, Song et al. [36] and Chierichetti et al. [15] have made significant progress on the problem on both the algorithm and the hardness fronts. First, our methods are quite different in many respects, even when we restrict to non-negative matrices and we compare the error

of the algorithm with the best k-column approximation, as opposed to the best rank-k approximation. Second, our analysis also applies to the *generalized CSS* problem, where we have two matrices, and we approximate one using the columns of the other. Third, as mentioned above, our algorithm has weaker guarantees than the prior work when the matrix B has a *very low error* ($\ll (1/k)\|B\|_1$) ℓ_1 approximation.

2 Sparse recovery under noise

The aim of this section is to outline the proof of Theorem 3. We start with some simple observations about re-scaling. The first is that we may assume that $\|b\|_1 = 1$ without loss of generality, because otherwise, we can run the entire procedure with $b/\|b\|_1$, and re-scale the coefficients of the obtained y by a factor $\|b\|_1$. The second observation is that we may assume that all the columns of A (which we may assume to be non-zero, as zero columns can be ignored) have unit ℓ_1 norm, w.l.o.g. This is again simple: if not, we can solve the problem with a matrix whose columns are $A_i/\|A_i\|_1$, and then divide the obtained x_i by the corresponding $\|A_i\|_1$ to obtain a solution with the original matrix.

Thus, by way of simplifying notation, assume that the columns of A are denoted by the set V of vectors on the "probability simplex" Δ_n in $\mathbb{R}^n$, and we denote by $p \in \Delta_n$ be the target vector. Suppose there exist $v_1, \ldots, v_k \in V$, and non-negative α_i, such that $\|p - \sum_i \alpha_i v_i\|_1 \leq \epsilon$.

Our goal is to design an iterative algorithm that maintains a vector $q \in \Delta_n$ (the *current approximation* to p), and adds one vector from $v \in V$ to q (with appropriate step size) in each iteration so as to improve a potential. The most natural potential (which works for ℓ_p norms for all $p > 1$), is simply the ℓ_1 distance $\|p - q\|_1$. Unfortunately, for this potential, depending on the current q, there may not exist "local improvements". This can be observed in a simple example:

$$v_1 = (1,0,1,0),\ v_2 = (1,0,0,1),\ v_3 = (0,1,1,0),\ v_4 = (0,1,0,1).$$

Let $p = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$. Clearly, it is in the convex hull of the vectors. Now, consider $q = (1,1,0,0)$. It can easily be verified that there is no single vector that can be added in order to improve $\|p - q\|$. (Indeed, such examples are well-known [18].)

Another natural potential is the relative entropy $D_{\mathrm{KL}}(p \parallel q)$. The problem with this is that it can be extremely sensitive to changes in q when q is close to the *boundary* of the simplex. While we may hope to control the distance to the boundary via a "warm start", it turns out to be tricky to implement.

Instead, we maintain a potential that automatically controls the distance to the boundary, while at the same time, allows us to reason about proximity to the optimum at the end:

$$\Phi(q) := D_{\mathrm{KL}}\left(p \parallel \frac{p+q}{2}\right) = \sum_i p_i \log\left(\frac{2p_i}{p_i + q_i}\right). \tag{1}$$

The potential is, roughly speaking, "one part" of the Jensen-Shannon divergence. The updates we consider are analogous to Frank-Wolfe iterations. In particular, in every round of the algorithm we greedily select the column that minimizes the potential and using it we recompute the vector q as shown in Algorithm 1.

To prove our main results we analyze the drop of the potential in every round of our algorithm. Interestingly we can show that the potential decrease geometrically during the execution of the algorithm. More formally, we show the following lemma.

Algorithm 1 Warm-KL

Initialize $q^{(0)} = v$ for an arbitrary $v \in V$, $S = \emptyset$, $\eta = \delta^2/2k$, and $T = \lceil \log\left(\frac{1}{4\sqrt{\epsilon+2\delta}}\right) / \log\left(\frac{1}{1-\frac{\eta}{2}}\right) \rceil$
for $t = 1 \dots T$ **do**
  Find the column $u \in V$ that minimizes $\Phi((1-\eta)q^{(t-1)} + \eta u)$, and add it to S.
  Set $q^{(t)} \leftarrow (1-\eta)q^{(t-1)} + \eta u$.
end for

▶ **Lemma 5** (Potential drop). *Consider the execution of the algorithm, and suppose* $\Phi(q^{(t-1)}) \geq 4(\epsilon + 2\delta)$. *Then we have*

$$\Phi(q^{(t)}) \leq \left(1 - \frac{\eta}{2}\right) \Phi(q^{(t-1)}).$$

We first show how to prove Theorem 3 using Lemma 5, then in the next section we focus on the proof of the lemma.

Proof of Theorem 3. Note that $\Phi(q^{(t)}) = \sum_i p_i \log\left(\frac{2p_i}{p_i+q_i}\right) \leq \sum_i p_i \ln 2 \leq 1$. So after $T = \lceil \log\left(\frac{1}{4\sqrt{\epsilon+2\delta}}\right) / \log\left(\frac{1}{1-\frac{\eta}{2}}\right) \rceil$ steps, we have $\Phi(q^{(T)}) \leq 4(\epsilon + 2\delta)$. Now, using Pinsker's inequality, we have that

$$\left\| p - \frac{p + q^{(T)}}{2} \right\|_1 \leq \sqrt{2\Phi(q^{(T)})} \leq 2\sqrt{2(\epsilon + 2\delta)}.$$

This then implies that $\left\|p - q^{(T)}\right\|_1 \leq 4\sqrt{2(\epsilon + 2\delta)}$, completing the proof of the theorem. ◀

2.1 Analyzing the potential drop

As shown in the previous subsection the key is to analyze the drop in potential. Let us fix some t, and for convenience, write $q = q^{(t-1)}$, and $q' = q^{(t)}$. We have $q' = (1-\eta)q + \eta u$, for some $u \in V$. Then, we have

$$\Phi(q) - \Phi(q') = \sum_i p_i \log\left(\frac{p_i + q'_i}{p_i + q_i}\right) = \sum_i p_i \log\left(1 + \eta \cdot \frac{u_i - q_i}{p_i + q_i}\right). \tag{2}$$

Note that we wish to lower bound this potential drop. In other words, we need to prove that there exists a $u \in V$ such that the difference above is "large". Since we know that there is a linear combination $\sum_i \alpha_i v_i$ that is ϵ-close to p (in ℓ_1), the natural goal is to prove that one of the v_i achieves the necessary potential drop, by an averaging argument. This is typically done by approximating the change in Φ by a linear function of the u_i's. If $\eta \cdot \frac{u_i - q_i}{p_i + q_i} < 1$, this can be done using $\log(1+x) \approx x$. But unfortunately in our case, the term $\frac{u_i - q_i}{p_i + q_i}$ can be arbitrarily large, making such an approximation impossible.

To deal with this, we take advantage of additional structure. The first observation is the following.

▶ **Lemma 6.** *Let* v_j, $1 \leq j \leq k$, *be vectors in* Δ_n *such that* $\left\|p - \sum_j \alpha_j v_j\right\|_1 \leq \epsilon$. *Then for all* $\delta > 0$, *there exists a subset* S^* *of* $[k]$, *such that* $\left\|p - \sum_{j \in S^*} \alpha_j v_j\right\|_1 \leq \delta + \epsilon$, *and additionally,* $\alpha_j \geq \delta/k$ *for all* $j \in S^*$. *(I.e., the coefficients used are all "non-negligible".)*

▶ Remark. Even though the lemma is straightforward, it is the only place in the proof we use the "promise" that there exists a k-sparse approximation to p.

Proof. The proof is simple: we consider $\sum_j \alpha_j v_j$, and remove all the terms whose coefficients are $< \delta/k$. As there are at most k terms, the total ℓ_1 norm of the removed terms is $\leq \delta$. This implies that considering only the terms that remain has an error at most $\epsilon + \delta$. ◀

The lemma shows that we can obtain a good approximation only using large coefficients. As a consequence, we now show that we can restrict our attention to a truncated version of the vectors V, which enables our analysis. Formally, define "truncated" vectors w_j by setting

$$w_{ji} = \min\left\{v_{ji}, \frac{kp_i}{\delta}\right\}.$$

I.e., the w_j are the vectors v_j, truncated so that no entry is more than k/δ times the corresponding entry in p. We start with a simple observation.

▶ **Lemma 7.** *For the vectors v_j, w_j as above, we have*

$$\left\|p - \sum_{j\in S^*} \alpha_j w_j\right\|_1 \leq \left\|p - \sum_{j\in S^*} \alpha_j v_j\right\|_1.$$

Proof. Let us look at the ith coordinate of the vectors on the LHS and RHS. The only way we can have $v_{ji} \neq w_{ji}$ (for some j) is when $v_{ji} > kp_i/\delta$, in which case $\alpha_j v_{ji} > p_i$. Thus in this coordinate, $\sum_{j\in S^*} \alpha_j v_j$ has a value larger than p_i. Now, by moving to $\sum_{j\in S^*} \alpha_j w_j$, we decrease the value, but remain larger or equal to p_i. Thus, the norm of the difference only improves. ◀

Let us now go back to the drop in potential we wished to analyze (eq. (2)). Our aim is to prove that setting $u = v_j$ for some $j \in S^*$ in the algorithm leads to a significant drop in the potential. We instead prove that setting $u = w_j$ (as opposed to v_j) leads to a significant drop in the potential. Then, we can use the fact that the RHS of (2) is monotone in u (noting that $v_{ji} \geq w_{ji}$) to complete the proof.

Let us analyze the potential drop when $u = w_j$ for some $j \in S^*$. Let $\eta = \delta^2/2k$. Write $\gamma_i := \frac{u_i - q_i}{p_i + q_i}$. Then, we have

$$\gamma_i = \frac{p_i + u_i}{p_i + q_i} - 1 \in [-1, k/\delta], \quad \text{as } w_{ji}/p_i \leq k/\delta \text{ for } j \in S^*.$$

Now, using the value of η, we have that $\eta\gamma_i \in (-\eta, \delta/2)$. This allows us to use a linear approximation for the logarithmic term in Φ. Once we move to a linear approximation, we can use an averaging argument to show that there exists a choice of u that improves the potential substantially.

More formally, let I_+ denote the set of indices with $u_i \geq q_i$ (i.e., $\gamma_i \geq 0$), and I_- denote the other indices. Then, using that, for $x > -1$, $\log(1+x) \geq x(1-x)$ we have that for any $i \in I_+$, we have $\log(1 + \eta\gamma_i) \geq \eta\gamma_i(1 - \frac{\delta}{2})$. Further, for $i \in I_-$, we have $\log(1 + \eta\gamma_i) \geq \eta\gamma_i(1 + \eta)$. Thus, in both the cases, we have that

$$\log(1 + \eta\gamma_i) \geq \eta\gamma_i - \frac{\delta\eta|\gamma_i|}{2}. \tag{3}$$

Now before showing how to use to the inequality to conclude the proof, we use an averaging argument to prove the existence of a good column j.

▶ **Lemma 8.** *There exists an index $j \in S^*$ such that setting $u = w_j$ in the algorithm gives*

$$\sum_i p_i \gamma_i \geq D_{KL}\left(p \parallel \frac{p+q}{2}\right) - (2\epsilon + \delta).$$

Proof. We start by recalling (via Lemmas 7 and 6), that

$$\left\| p - \sum_{j \in S^*} \alpha_j w_j \right\|_1 \leq \left\| p - \sum_{j \in S^*} \alpha_j v_j \right\|_1 \leq (\epsilon + \delta).$$

For convenience, let us write $r = \sum_{j \in S^*} \alpha_j w_j$, and so $\|p - r\|_1 \leq \epsilon + \delta$. Now, we wish to analyze the behavior of γ_i when we set $u = w_j$ for different j.

We start by defining $\tau_i^{(j)}$ as

$$\tau_i^{(j)} := \frac{w_{ji} - q_i}{p_i + q_i},$$

i.e., the value of γ_i obtained by setting $u = w_j$. For convenience, write $Z = \sum_{j \in S^*} \alpha_j$. Now by linearity, we have

$$\sum_{j \in S^*} \alpha_j \left(\sum_i p_i \tau_i^{(j)} \right) = \sum_i p_i \frac{(\sum_{j \in S^*} \alpha_j w_{ji}) - Z q_i}{p_i + q_i} = \sum_i p_i \frac{r_i - Z q_i}{p_i + q_i}.$$

Thus, by averaging (specifically, the inequality that if $\sum_j \alpha_j X_j \geq Y$ for $\alpha_j \geq 0$, then there exists a j such that $X_j \geq Y/(\sum_j \alpha_j)$), we have that there exists a $j \in S^*$ such that

$$\sum_i p_i \tau_i^{(j)} \geq \frac{1}{Z} \cdot \sum_i p_i \frac{r_i - Z q_i}{p_i + q_i} = \sum_i p_i \frac{r_i - q_i}{p_i + q_i} + \left(\frac{1}{Z} - 1\right) \sum_i p_i \frac{r_i}{p_i + q_i}. \tag{4}$$

The first term on the RHS can now be lower bounded as:

$$\sum_i p_i \frac{r_i - q_i}{p_i + q_i} = \sum_i p_i \left(\frac{2p_i}{p_i + q_i} - 1 - \frac{p_i - r_i}{p_i + q_i} \right) \geq \sum_i p_i \left(\frac{2p_i}{p_i + q_i} - 1 \right) - \sum_i |p_i - r_i|,$$

where we use the fact that $|p_i/(p_i + q_i)| \leq 1$. Thus, appealing to the inequality $(x - 1) \geq \log x$, if we write $D := D_{\mathrm{KL}}\left(p \parallel \frac{p+q}{2}\right)$, then we have

$$\sum_i p_i \frac{r_i - q_i}{p_i + q_i} \geq D - \epsilon - \delta, \quad \text{since } \|p - r\|_1 \leq \epsilon + \delta.$$

To conclude the proof, let us consider the second term in the RHS of (4). If $Z \leq 1$, the term is non-negative, and there is nothing to show. We next argue that $Z \leq 1 + \epsilon$, by showing that the sum of α_j over *all* $j \in S$ can be bounded by $1 + \epsilon$. In fact, note that $\left\| \sum_{j \in S} \alpha_j v_j \right\|_1 - 1 \leq \left\| p - \sum_{j \in S} \alpha_j v_j \right\|_1 \leq \epsilon$ (triangle inequality). Furthermore for $v_j \in \Delta_n$, we have $\left\| \sum_j \alpha_j v_j \right\|_1 = \sum_j \alpha_j$. Thus $\sum_{j \in S} \alpha_j \leq 1 + \epsilon$, and thus

$$\left(1 - \frac{1}{Z}\right) \sum_i \frac{p_i r_i}{p_i + q_i} \leq \frac{\epsilon}{1 + \epsilon} \sum_i r_i \leq \epsilon.$$

Plugging this into (4) we can conclude the proof of the lemma. ◀

We are now ready to complete the proof of the main lemma of this section – Lemma 5.

Proof of Lemma 5. Let $D := D_{\mathrm{KL}}\left(p \parallel \frac{p+q}{2}\right)$. We can use Lemma 8 in conjunction with Eq. (3) to obtain that there exists a j such that setting $u = w_j$ gives us

$$\sum_i p_i \log(1+\eta\gamma_i) \geq \sum_i \eta p_i \gamma_i - \frac{\delta\eta}{2}\sum_i p_i|\gamma_i| \geq \eta(D - 2\epsilon - \delta) - \frac{\delta\eta}{2}\sum_i p_i|\gamma_i|.$$

The last term on the RHS can be bounded, noting that

$$\sum_i p_i|\gamma_i| \leq \sum_i p_i \cdot \frac{|u_i - q_i|}{p_i + q_i} \leq \sum_i |u_i - q_i| \leq 2,$$

as u_i, p_i, q_i are all probability distributions, and $\frac{p_i}{p_i+q_i} \leq 1$. This implies that there exists a choice of $u = w_j$ (and thus setting $u = v_j$ also works, as discussed earlier), such that the potential drop is at least $\eta(D - 2\epsilon - \delta) - \eta\delta$. If $D \geq 4(\epsilon + \delta)$, this is at least $\eta D/2$. This completes the proof of the lemma. ◀

3 Low rank approximation

We now come to our main result – Theorem 4. Let the dimensions of B be $n \times m$ (thus A also has n rows for the problem to be well-defined). The main difference between this setting and the one in Section 2 is that we have a collection of m columns, and we wish to find a subset S that can "simultaneously" approximate all of them. Another technical difference is that the columns of B can all have different lengths, thus we can only re-scale all of them by the same amount.

Let us start with some simple assumptions we can make w.l.o.g. First, we may assume that $\|B\|_1 = 1$, as we can scale the entire matrix by $1/\|B\|_1$, solve the problem, and then multiply the obtained X (entry-wise) by $\|B\|_1$. Second, we may assume that every column of A has unit ℓ_1 norm, as otherwise, we can solve the problem using a matrix with columns $A_i/\|A_i\|_1$, and then re-scale the ith *row* of the obtained X by $1/\|A_i\|$ to find a solution to the original problem.

Under these assumptions (i.e., $\|B\|_1 = 1$ and $A_i \in \Delta_n$), we now show how Section 2 gives a "framework" that can be used here. The main idea is as follows.

Outline of the approach

Let us flatten the matrix B into an nm dimensional vector p (thus the (i,j)th entry of B now appears in the $[(j-1)n + i]$th position of p). Now, the CSS problem can be re-stated as expressing p as a linear combination of vectors of the form $A_i \otimes y_i$, for some non-negative vectors y_i (these will form the *rows* of X in the problem definition). Thus, let V denote the set of all vectors of the form

$$\{A_i \otimes z \ : \ A_i \text{ is a column of } A,\ z \in \Delta_m\}.$$

The goal is to use the framework of Section 2 to find a small subset of V. Unfortunately, the set V above is infinite! So we cannot apply Algorithm 1 directly. However, note that as long as we can find a $u \in V$ that satisfies the potential drop condition of Lemma 5, the analysis still applies. In the remainder of the section, we show how to design an efficient "oracle" to find such a u, thus proving the theorem.

Before we begin, let us observe that our normalizations indeed reduce our matrix problem to the problem analyzed in the previous section. For any $u \in V$, we have $\|u\|_1 = \|A_i\|_1 \|z\|_1 = 1$. We have also normalized so that $\|p\|_1 = 1$ (recall p is the flattened form of B). Now, suppose B has a good ℓ_1 approximation using a submatrix A_S, i.e., suppose $\|B - A_S X\| \leq \epsilon$,

for some matrix X. Then, denoting by $X^{(j)}$ the jth row of X, we can re-write the above as $\left\|B - \sum_{j\in S} A_j X^{(j)}\right\|_1 \leq \epsilon$. Now setting $y_j = X^{(j)} / \left\|X^{(j)}\right\|_1$ (so $y_j \in \Delta_m$), and $\alpha_j = \left\|X^{(j)}\right\|_1$, we can re-write the above as

$$\left\| p - \sum_{j\in S} \alpha_j (A_j \otimes y_j) \right\|_1 \leq \epsilon, \tag{5}$$

which is precisely the form we need. Let us thus see how to efficiently find a $u \in V$ that reduces the potential significantly.

3.1 Oracle for each iteration

Consider the t-th iteration, and let $q^{(t-1)}$ be the previous iterate (which we denote by q, for convenience). Our goal is to find a u such that $\Phi((1-\eta)q + \eta u)$ is small. The technical difficulty here is the logarithmic term in the definition of $\Phi()$.

As this was also the challenge in Section 2, we begin by recalling one key aspect of the analysis: the definition of the truncated vectors w. Following the notation earlier, and from Eq. (5), define $v_j = A_j \otimes y_j$, and let w_j be defined using $w_{ji} := \min\{v_{ji}, \frac{kp_i}{\delta}\}$ (i now ranges from 1 to nm). The main component of the argument was the existence of an index j such that $\sum_i p_i \frac{w_{ji}-q_i}{p_i+q_i}$ is large.

Thus, if we can find a $u \in V$ such that for its truncation h (defined by $h_i := \min\{u_i, \frac{kp_i}{\delta}\}$), the value $\sum_i p_i \frac{h_i-q_i}{p_i+q_i}$ is large, we would be done. When V is finite, we simply did this by iterating over all $u \in V$, constructing the h, and computing the values. In the current setting, we have infinitely many V. Fortunately we can handle this complication by analyzing the columns separately. First partition V into V_j, one for each column A_j. I.e., $V_j := \{A_j \otimes z \ : \ z \in \Delta_n\}$. We then design an efficient method to maximize the quantity described above over a single V_j. Since there are only finitely many columns in A, we can finally just take the maximum.

▶ **Lemma 9.** *Let us fix some j. There is an efficient algorithm to find $u \in V_j$ (defined above) to maximize $\sum_i p_i \frac{h_i-q_i}{p_i+q_i}$, where $h_i = \min\{u_i, \frac{kp_i}{\delta}\}$.*

Proof. Observe that as every u is of the form $A_j \otimes z$, we need to find a $z \in \Delta_m$ that maximizes the objective value. The key observation is that this can be written as a linear program! To define the LP, it helps to view i as being defined by (i_1, i_2), where $i = n(i_2 - 1) + i_1$, where $i_1 \in [n]$ and $i_2 \in [m]$. The variables are $h \in \mathbb{R}^{nm}$, $z \in \mathbb{R}^m$; p, q are given (p is the target and q is the current iterate). The LP is as follows:

$$\text{maximize} \sum_{i_1,i_2} \frac{p_{(i_1,i_2)}}{p_{(i_1,i_2)} + q_{(i_1,i_2)}} h_{(i_1,i_2)}, \quad \text{subject to} \tag{6}$$

$$\forall\ i_1 \in [n], i_2 \in [m],\ \ 0 \leq h_{(i_1,i_2)} \leq A_{j,i_1} \cdot z_{i_2} \tag{7}$$

$$\forall\ i_1 \in [n], i_2 \in [m],\ \ h_{(i_1,i_2)} \leq \frac{kp_{(i_1,i_2)}}{\delta} \tag{8}$$

$$\forall\ i_2 \in [m],\ \ z_{i_2} \geq 0 \tag{9}$$

$$\sum_{i_2 \in [m]} z_{i_2} = 1. \tag{10}$$

In the above, A_{j,i_1} is simply the i_1'th entry of the vector A_j. The optimum solution will satisfy $h_{(i_1,i_2)} = \min\{A_{j,i_1} \cdot z_{i_2}, \frac{kp_{(i_1,i_2)}}{\delta}\}$, as for any given z, this setting will maximize the objective (which is the goal). From the definition of V_j, it follows that the LP indeed captures the problem of maximizing over V_j. This proves the lemma. ◀

As the LP can be solved in polynomial time, we can follow this procedure for every column j of A. This completes the proof of Theorem 4 if we only require a polynomial time algorithm. As LP solvers can be inefficient, in the next subsection we developed a fast "greedy" solution to our specific LP.

4 Solving the linear program efficiently

While linear programs can be solved in polynomial time, they can often be a bottleneck in learning algorithms in practice. We thus give a simple "greedy" algorithm to solve the LP in (6)-(10)[2].

▶ **Theorem 10.** *Consider the LP* (6)-(10)*, written for a given column A_j of A. There is a greedy algorithm that find the optimal z in time $O(mn \log n)$. (For sparse matrices this can be further reduced to $O(\text{nnz}(A) \log n)$).*

Proof. Let us rewrite the objective function, splitting it according to i_2:

$$\sum_{i_2 \in [m]} \sum_{i_1 \in [n]} \frac{p_{(i_1,i_2)}}{p_{(i_1,i_2)} + q_{(i_1,i_2)}} h_{(i_1,i_2)}.$$

Let us now focus on one index i_2, and consider increasing z_{i_2} from 0 to 1. The value of $h_{(i_1,i_2)}$ is equal to $A_{j,i_1} \cdot z_{i_2}$ until z_{i_2} hits $\min\{1, kp_{(i_1,i_2)}/\delta A_{j,i_1}\}$, and then remains constant. (If $A_{j,i_1} = 0$, then it stays 0 throughout.) This happens for each i_1, and thus the inner summation, as a function of z_{i_2}, is piecewise linear and non-decreasing. Let us denote this function by $f_{i_2}(x)$.

Finding the optimal z can now be cast as the following problem: we have monotone, concave, piecewise-linear functions $f_1, \ldots, f_m$ defined on $[0, 1]$, and we wish to find $z_j \geq 0$, summing to 1, such that $\sum_j f_j(z_j)$ is maximized.

This can be greedily done as follows: suppose we have a sorted list of the "break-points" for each f_j (a break-point is the value of x at which a piece-wise linear function changes slope). Note that for every j, the number of break-points (in the interval $[0, 1]$) is no more than n. Now, we start with $z_j = 0$ for all j. In every iteration, we decide to increment one z_j. The choice will depend on the j that has the largest slope to the right of z_j. Once we pick a j, we increment the z_j until its next break-point (or until the sum of z_j's becomes 1).

The correctness of this procedure follows from the fact that we always increment the z_j with the largest slope so we maximize the value of $\sum_j f_j(z_j)$ (and because the functions are concave, we can never encounter a higher slope later). As for the run time, we note that the total number of break points is mn, and in the worst case, we could encounter each one. At each step, as the slopes are monotone, the choice of the j can be found using a priority queue, and this leads to a run time of $O(mn \log n)$. If the A is sparse, then the number of "slopes" we need to consider is at most the number of non-zero entries, which gives the desired bound. ◀

5 Conclusion

We have presented new algorithmic results for two fundamental problems, sparse regression and column subset selection, under ℓ_1 error. The key assumption necessary for us is the non-negativity of the associated matrices and the decomposition. Under this assumption,

[2] We note that our argument is similar to the fractional knapsack argument.

our algorithms provably achieve approximation guarantees with respect to the corresponding optimal solutions. Our sparse regression analysis gives a simple framework for obtaining ℓ_1 error guarantees. We have used it for our column selection result, but it may be applicable to other contexts as well, such as non-negative matrix factorization (for which our result applies under the anchor word assumption), matrix and tensor variants of sparse regression/recovery, and also in obtaining recovery algorithms for broader classes of mixture models under noise. We leave these as interesting avenues for future work.

References

1 Jason Altschuler, Aditya Bhaskara, Gang Fu, Vahab S. Mirrokni, Afshin Rostamizadeh, and Morteza Zadimoghaddam. Greedy column subset selection: New bounds and distributed algorithms. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2539–2548. JMLR.org, 2016. URL: `http://jmlr.org/proceedings/papers/v48/altschuler16.html`.

2 Edoardo Amaldi and Viggo Kann. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209(1Ð2):237–260, 1998.

3 Sanjeev Arora, Rong Ge, Yonatan Halpern, David M. Mimno, Ankur Moitra, David Sontag, Yichen Wu, and Michael Zhu. A practical algorithm for topic modeling with provable guarantees. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 280–288. JMLR.org, 2013. URL: `http://jmlr.org/proceedings/papers/v28/arora13.html`.

4 Sanjeev Arora, Rong Ge, Ravindran Kannan, and Ankur Moitra. Computing a nonnegative matrix factorization - provably. In Howard J. Karloff and Toniann Pitassi, editors, *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 145–162. ACM, 2012. `doi:10.1145/2213977.2213994`.

5 Arturs Backurs, Piotr Indyk, Ilya P. Razenshteyn, and David P. Woodruff. Nearly-optimal bounds for sparse recovery in generic norms, with applications to k-median sketching. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 318–337. SIAM, 2016. `doi:10.1137/1.9781611974331.ch24`.

6 Afonso S. Bandeira, Edgar Dobriban, Dustin G. Mixon, and William F. Sawin. Certifying the restricted isometry property is hard, 2012. `arXiv:arXiv:1204.1580`.

7 Siddharth Barman. Approximating nash equilibria and dense bipartite subgraphs via an approximate version of caratheodory's theorem. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 361–369. ACM, 2015. `doi:10.1145/2746539.2746566`.

8 Aditya Bhaskara, Ananda Theertha Suresh, and Morteza Zadimoghaddam. Sparse solutions to nonnegative linear systems and applications. In Guy Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015, San Diego, California, USA, May 9-12, 2015*, volume 38 of *JMLR Workshop and Conference Proceedings*. JMLR.org, 2015. URL: `http://jmlr.org/proceedings/papers/v38/bhaskara15.html`.

9 Avrim Blum, Sariel Har-Peled, and Benjamin Raichel. Sparse approximation via generating point sets. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual*

ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016, pages 548–557. SIAM, 2016. doi:10.1137/1.9781611974331.ch40.

10 Christos Boutsidis and David P. Woodruff. Optimal CUR matrix decompositions. *SIAM J. Comput.*, 46(2):543–589, 2017. doi:10.1137/140977898.

11 J.P. Brooks, J.H. Dulá, and E.L. Boone. A pure ℓ_1-norm principal component analysis. *Computational Statistics & Data Analysis*, 61:83–98, 2013.

12 Joe M Butler, D Timothy Bishop, and Jennifer H Barrett. Strategies for selecting subsets of single-nucleotide polymorphisms to genotype in association studies. *BMC genetics*, 6(1):S72, 2005.

13 Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *JACM*, 58(3):11:1–11:37, 2011.

14 Emmanuel J Candes, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on pure and applied mathematics*, 59(8):1207–1223, 2006.

15 Flavio Chierichetti, Sreenivas Gollapudi, Ravi Kumar, Silvio Lattanzi, Rina Panigrahy, and David P. Woodruff. Algorithms for ℓ_p low-rank approximation. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 806–814. PMLR, 2017. URL: http://proceedings.mlr.press/v70/chierichetti17a.html.

16 Hugh A Chipman and Hong Gu. Interpretable dimension reduction. *Journal of applied statistics*, 32(9):969–987, 2005.

17 Kenneth L. Clarkson. Coresets, sparse greedy approximation, and the frank-wolfe algorithm. *ACM Trans. Algorithms*, 6(4):63:1–63:30, 2010. doi:10.1145/1824777.1824783.

18 M. J. Donahue, C. Darken, L. Gurvits, and E. Sontag. Rates of convex approximation in non-hilbert spaces. *Constructive Approximation*, 13(2):187–220, 1997. doi:10.1007/BF02678464.

19 David L Donoho and Michael Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via ?1 minimization. *Proceedings of the National Academy of Sciences*, 100(5):2197–2202, 2003.

20 A. Eriksson and A. van den Hengel. Efficient computation of robust low-rank matrix approximations using the L_1 norm. *PAMI*, 34(9):1681–1690, 2012.

21 Dan Feldman, Mikhail Volkov, and Daniela Rus. Dimensionality reduction of massive sparse datasets using coresets. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2766–2774. Curran Associates, Inc., 2016. URL: http://papers.nips.cc/paper/6596-dimensionality-reduction-of-massive-sparse-datasets-using-coresets.pdf.

22 Dean P. Foster, Howard J. Karloff, and Justin Thaler. Variable selection is hard. In Peter Grünwald, Elad Hazan, and Satyen Kale, editors, *Proceedings of The 28th Conference on Learning Theory, COLT 2015, Paris, France, July 3-6, 2015*, volume 40 of *JMLR Workshop and Conference Proceedings*, pages 696–709. JMLR.org, 2015. URL: http://jmlr.org/proceedings/papers/v40/Foster15.html.

23 Alan M. Frieze, Ravi Kannan, and Santosh Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *J. ACM*, 51(6):1025–1041, 2004. doi:10.1145/1039488.1039494.

24 Rahul Garg and Rohit Khandekar. Gradient descent with sparsification: an iterative algorithm for sparse recovery with restricted isometry property. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 337–344. ACM, 2009.

25 Venkatesan Guruswami and Ali Kemal Sinop. Optimal column-based low-rank matrix reconstruction. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1207–1214. SIAM, 2012. URL: `http://portal.acm.org/citation.cfm?id=2095211&CFID=63838676&CFTOKEN=79617016`.

26 Peter J. Huber. *Robust Statistics*. John Wiley & Sons, New York„ 1981.

27 Qifa Ke and Takeo Kanade. Robust L_1 norm factorization in the presence of outliers and missing data by alternative convex programming. In *CVPR*, pages 739–746, 2005.

28 Pascal Koiran and Anastasios Zouzias. Hidden cliques and the certification of the restricted isometry property. *IEEE Trans. Information Theory*, 60(8):4999–5006, 2014. `doi:10.1109/TIT.2014.2331341`.

29 Robert Krauthgamer, editor. *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*. SIAM, 2016. `doi:10.1137/1.9781611974331`.

30 Anastasios Kyrillidis, Stephen Becker, Volkan Cevher, and Christoph Koch. Sparse projections onto the simplex. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 235–243. JMLR.org, 2013. URL: `http://jmlr.org/proceedings/papers/v28/kyrillidis13.html`.

31 Cewu Lu, Jiaping Shi, and Jiaya Jia. Scalable adaptive robust dictionary learning. *TIP*, 23(2):837–847, 2014.

32 Deyu Meng and Fernando. D. L. Torre. Robust matrix factorization with unknown noise. In *ICCV*, pages 1337–1344, 2013.

33 Bruno A. Olshausen and David J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37(23):3311–3325, 1997. `doi:10.1016/S0042-6989(97)00169-7`.

34 Saurabh Paul, Malik Magdon-Ismail, and Petros Drineas. Column selection via adaptive sampling. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 406–414, 2015. URL: `http://papers.nips.cc/paper/6011-column-selection-via-adaptive-sampling`.

35 Shai Shalev-Shwartz, Nathan Srebro, and Tong Zhang. Trading accuracy for sparsity in optimization problems with sparsity constraints. *SIAM J. on Optimization*, 20(6):2807–2832, 2010.

36 Zhao Song, David P. Woodruff, and Pelin Zhong. Low rank approximation with entrywise ℓ_1-norm error. In *STOC*, 2017.

37 Naiyan Wang, Tiansheng Yao, Jingdong Wang, and Dit-Yan Yeung. A probabilistic approach to robust matrix factorization. In *ECCV*, pages 126–139, 2012.

38 Naiyan Wang and Dit-Yan Yeung. Bayesian robust matrix factorization for image and video processing. In *ICCV*, pages 1785–1792, 2013.

39 Tengyao Wang, Quentin Berthet, and Yaniv Plan. Average-case hardness of RIP certification. *CoRR*, abs/1605.09646, 2016. `arXiv:1605.09646`.

40 Kai Wei, Yuzong Liu, Katrin Kirchhoff, Chris Bartels, and Jeff Bilmes. Submodular subset selection for large-scale speech training data. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 3311–3315. IEEE, 2014.

41 L. Xiong, X. Chen, and J. Schneider. Direct robust matrix factorization for anomaly detection. In *ICDM*, pages 844–853, 2011.

42 L. Xu and A. L. Yuille. Robust principal component analysis by self-organizing rules based on statistical physics approach. *IEEE Transactions on Neural Networks*, 6(1):131–143, 1995.

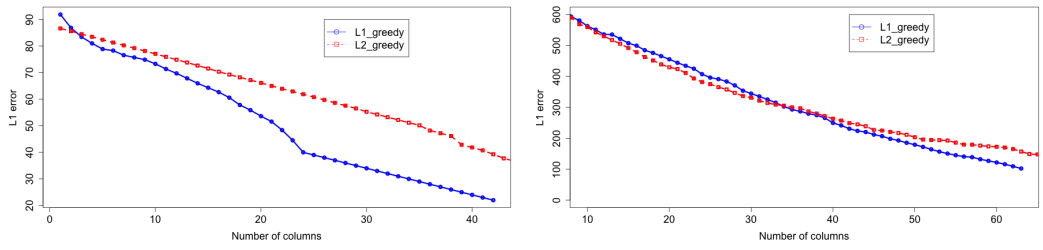
43 Y. Zheng, G. Liu, S. Sugimoto, S. Yan, and M. Okutomi. Practical low-rank matrix approximation under robust L_1-norm. In *CVPR*, pages 1410–1417, 2012.

A Experiments

We now outline some preliminary experimental results that show the effectiveness of our ℓ_1 column subset selection algorithm. In particular we compare our algorithm with the greedy algorithm of [1], which optimizes a potential function similar to ours, but tailored towards ℓ_2 approximation. We compare on synthetic as well as a real-life dataset.

The synthetic dataset is constructed as follows. The matrix A has all its columns on the unit simplex Δ_n, where $n = 150$. The points are divided into two categories – inliers and outliers. The inliers are all in the convex hull of a special set of $k = 15$ points, and there are 115 of them (including the k special points). Along with these, there are $M = 40$ outliers, which are chosen to be sparser than the average point in the special set. (Note that the special points are the "anchor points" in the non-negative factorization connection pointed to in Section 1.) Thus the dataset has a large number of outliers (25%). The figure shows the decay of ℓ_1 error as we pick more columns. Note that even in the first few iterations, our potential allows quickly zeroing in on the right columns.

We then consider a real world matrix obtained from a region economic model, known as the WM1 matrix. The matrix is asymmetric and it is 207 x 277 and contains 2909 real-valued entries. The results on these matrices (showing rank vs the approximation error) for the two algorithms we consider can be found in the figure below. It is interesting to observe that the our ℓ_1 based algorithms has better performances when the error becomes smaller and it converges earlier to a better solution.



(a) (b)

Figure 1 L1-recovery error of the two column selection algorithms on (a) a synthetic dataset, (b) WM1 dataset.