Hardness and Algorithms for Robust and Sparse Optimization

Eric Price* Sandeep Silwal† Samson Zhou‡

June 30, 2022

Abstract

We explore algorithms and limitations for sparse optimization problems such as sparse linear regression and robust linear regression. The goal of the sparse linear regression problem is to identify a small number of key features, while the goal of the robust linear regression problem is to identify a small number of erroneous measurements. Specifically, the sparse linear regression problem seeks a k-sparse vector $x \in \mathbb{R}^d$ to minimize $\|Ax - b\|_2$, given an input matrix $A \in \mathbb{R}^{n\times d}$ and a target vector $b \in \mathbb{R}^n$, while the robust linear regression problem seeks a set S that ignores at most k rows and a vector x to minimize $\|(Ax - b)_S\|_2$.

We first show bicriteria, NP-hardness of approximation for robust regression building on the work of [OWZ15] which implies a similar result for sparse regression. We further show fine-grained hardness of robust regression through a reduction from the minimum-weight k-clique conjecture. On the positive side, we give an algorithm for robust regression that achieves arbitrarily accurate additive error and uses runtime that closely matches the lower bound from the fine-grained hardness result, as well as an algorithm for sparse regression with similar runtime. Both our upper and lower bounds rely on a general reduction from robust linear regression to sparse regression that we introduce. Our algorithms, inspired by the 3SUM problem, use approximate nearest neighbor data structures and may be of independent interest for solving sparse optimization problems. For instance, we demonstrate that our techniques can also be used for the well-studied sparse PCA problem.

1 Introduction

Sparsity is often a key feature embedded within large datasets and fundamentals tasks in data science and machine learning. For example, in the sparse linear regression or variable selection problem, the goal is to find a k-sparse vector $x \in \mathbb{R}^d$ to minimize $\|Ax - b\|_2$, given an input matrix $A \in \mathbb{R}^{n\times d}$ and a target vector $b \in \mathbb{R}^n$. The intuition is that the target vector can be effectively summarized as the linear combination of a small number of columns of A that denote the important features of the data. Similarly, in the robust linear regression or constraint selection problem, the goal is to find a set S that ignores at most k rows and a vector $x \in \mathbb{R}^d$ to minimize $\|(Ax - b)_S\|_2$, given an input matrix $A \in \mathbb{R}^{n\times d}$ and a target vector $b \in \mathbb{R}^n$, where the notation $(\cdot)_S$ means we only measure the loss over coordinates in S. In other words, we suppose that up to k entries of the target vector can be arbitrarily corrupted and thus ignored in the computation of the resulting empirical risk minimizer. In both the sparse and robust linear regression problems, the L_2 loss function can be naturally generalized to other loss functions which align with specific goals, such as truncating or mitigating the penalty beyond a certain threshold.

Sparsity is a desirable attribute for model design in machine learning, statistics, estimation, and signal processing. Simpler models with a small number of variables not only provide more ease for interpetability,

*Department of Electrical and Computer Engineering, The University of Texas at Austin. ecprice@gmail.com
†Electrical Engineering and Computer Science Department, Massachusetts Institute of Technology. silwal@mit.edu
‡Computer Science Department, Carnegie Mellon University. samsonzhou@gmail.com

but also tend to have smaller generalization error [FKT15]. A common algorithmic approach for acquiring sparse vectors for the task of sparse linear regression is to use greedy algorithms that iteratively select features, e.g., stepwise selection, backward elimination, and least angle regression. Another common technique for inducing sparse solutions is to penalize the objective with a regularization function. For example, the well-known LASSO adds a penalty term to the regression objective that is proportional to the L_1 norm of the underlying minimizer x [Tib96] while ridge regression [HK70] often adds a penalty that is proportional to the L_2 norm of x.

1.1 Our Results

In this paper, we study algorithms and limitations for sparse regression and robust regression. In particular, we consider the following problems:

Problem 1.1 (Robust Regression)**.** *Given $A \in \mathbb{R}^{n\times d}, b \in \mathbb{R}^n$, a loss function $\mathcal{L} : \mathbb{R}^* \to \mathbb{R}$, and integer $0 < k \leq n$, find $T \subset [n]$ satisfying $|T| \leq k$ and $x \in \mathbb{R}^n$ to minimize $\mathcal{L}((Ax-b)_T)$, where $(Ax-b)_T$ denotes that we only measure the loss on the coordinates in T. The coordinates not in T are called* ignored*.*

Problem 1.2 (Sparse Regression)**.** *Given $A \in \mathbb{R}^{n\times d}, b \in \mathbb{R}^n$, a loss function $\mathcal{L} : \mathbb{R}^* \to \mathbb{R}$, and integer $0 < k \leq n$, find $x \in \mathbb{R}^n$ with $\|x\|_0 \leq k$ to minimize $\mathcal{L}(Ax - b)$.*

[OWZ15] showed the hardness of approximation for maximizing the number of satisfied linear equations; their result can be translated to show bicriteria robust regression is NP-hard, i.e., it is NP-hard to achieve a multiplicative factor approximation to the optimal loss while allowing a constant factor larger parameter for sparsity. We introduce a further reduction to show the NP-hardness of bicriteria sparse regression.

Theorem 1.1. *Let $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$ be any loss function such that $\mathcal{L}(0^n) = 0$ and $\mathcal{L}(x) > 0$ for $x \neq 0^n$. Given a matrix $X \in \mathbb{R}^{a\times b}$, vector $c \in \mathbb{R}^a$, and a sparsity parameter t, let $OPT = \min_w \mathcal{L}(Xw - c)$ where $\|w\|_0 = t$. Then for any $C_1 > 1$ and any constant $C_2 > 1$, it is NP-hard to find w' satisfying $\|w'\|_0 = C_2 t$ such that $\mathcal{L}(Xw' - c) \leq C_1 \cdot OPT$.*

We remark that the statement of Theorem 1.1 holds for any parameter $C_1 > 1$, showing that multiplicative approximation is NP-hard, even for arbitrarily large multiplicative factors that could depend on n and d. It also applies to a large number of commonly used loss functions, such as L_p loss or various M-estimators, e.g., [CW15, CWW19, TWZ+22]. See Table 1 for more details. For completeness, we also formalize the proof of [OWZ15] to show bicriteria hardness of robust regression.

To prove Theorem 1.1, we prove a reduction (see Corollary 2.6) which states that an algorithm for the sparse regression problem can be used to solve the robust regression problem using a polynomial time blow-up. However, it is known that the sparse regression problem requires runtime $O(n^k)$ under the minimum-weight k-clique conjecture [GV21]; could it be the case that robust regression is significantly easier? We give a fine-grained hardness result showing this is not the case:

Theorem 1.2. *For every $\varepsilon > 0$, there exists a sufficiently large n such that the robust regression problem requires $\Omega(n^{k/2+o(1)})$ randomized time, unless the minimum-weight k-clique conjecture is false.*

To complement our fine-grained hardness results, we develop nearly-matching upper bounds with *additive error* in the case there exists a diagonal S ignoring k rows achieving zero loss. Namely, we give an algorithm with time $O(nk(n/\varepsilon)^{Ck})$, where $C < 1$ can be any constant arbitrarily close to $\frac{1}{2}$, thereby nearly matching the lower bounds of Theorem 1.2.

Loss function	Formulation
L_p	$\lvert x\rvert^p$
Cauchy	$\frac{\lambda^2}{2}\log\left(1+(x/\lambda)^2\right)$
Fair	$\lambda\lvert x\rvert - \lambda^2\ln\left(1+\frac{\lvert x\rvert}{\lambda}\right)$
Geman-McClure	$\frac{x^2}{2+2x^2}$
Huber	$\begin{cases} x^2/2 & \text{if } \lvert x\rvert \le \lambda \\ \lambda\lvert x\rvert - \lambda^2/2 & \text{otherwise} \end{cases}$
$L_1 - L_2$	$2\left(\sqrt{1+\frac{x^2}{2}}-1\right)$
Tukey	$\begin{cases} \frac{\lambda^2}{6}\left(1-\left(1-\frac{x^2}{\lambda^2}\right)^3\right) & \text{if } \lvert x\rvert \le \lambda \\ \frac{\lambda^2}{6} & \text{otherwise} \end{cases}$
Welsch	$\frac{\lambda^2}{2}\left(1-e^{-\left(\frac{x}{\lambda}\right)^2}\right)$

Table 1: Theorem 1.1 shows the bicriteria hardness of approximation for sparse regression for common loss functions. The same bicriteria hardness also holds for robust regression.

Theorem 1.3. *Given $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$ such that*

$$\min_{S,x} \|S(Ax-b)\|_2 = 0$$

over all diagonal matrices S with $n-k$ one entries in the diagonal and k zero entries, there exists an algorithm that returns a diagonal matrix S' with $n-k$ one entries in the diagonal and k zero entries and a vector x' such that $\|S'(Ax'-b)\|_2 \le \varepsilon$ in time

$$\min_{c\ge 1} O\left(nk\cdot\left(\frac{12c\cdot n\cdot\|b\|_2}{\varepsilon}\right)^{\frac{k}{2}\cdot(1+1/(2c^2-1))}\right).$$

We obtain an algorithm with a similar guarantee for sparse regression.

Theorem 1.4. *Given $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$ such that there exists a k-sparse vector x satisfying $Ax = b$, there exists an algorithm that returns a k-sparse vector $z \in \mathbb{R}^d$ satisfying $\|Az-b\|_2 \le \varepsilon$ in time*

$$\min_{c\ge 1} O\left(nk\cdot\left(\frac{12c\cdot d\cdot\|b\|_2}{\varepsilon}\right)^{\frac{k}{2}\cdot(1+1/(2c^2-1))}\right).$$

Our algorithms for Theorems 1.4 and 1.3 are inspired by techniques for the 3SUM problem and its generalization to k integers, in which the goal is to determine whether a set of n integers contain k integers that sum to zero. Rather than check all $O(n^k)$ possible k-sparse vectors, we first input all $\frac{k}{2}$-sparse vectors into an approximate nearest neighbor (ANN) data structure. Given a query q, the ANN data structure will output a point x such that $\|x-q\|_2 \le c\cdot\min_y\|y-q\|_2$, where the minimum is taken over all points that are input into the data structure. We can thus query the ANN data structure over all $\binom{n}{k/2}$ differences between the measurement vector b and all $\frac{k}{2}$-sparse vectors, reconstructing the k-sparse vector from the query that achieves the minimum value.

Surprisingly, our technique also works for the sparse principal component analysis (PCA) problem, in which the goal is to find a k-sparse unit vector $v \in \mathbb{R}^n$ to maximize v^TAv, given an input PSD matrix $A \in \mathbb{R}^{n\times n}$ of rank r.

Theorem 1.5. *There exists an algorithm that uses* $\widetilde{O}\left(\frac{k^2}{\varepsilon}\cdot\left(r\left(\frac{n\kappa}{\varepsilon}\right)^{k(1+\varepsilon)/2}+\left(\frac{1}{\varepsilon}\right)^{r/2+1}\right)\right)$ *time and with high probability, outputs a* k*-sparse unit vector* u *such that with high probability,*

$$u^TAu \geq (1-\varepsilon)\max_{\|v\|_2=1\,,\|v\|_0\leq k} v^TAv,$$

where κ *is the condition number of* A*.*

We also give a simple NP-hardness proof of the robust regression problem in the appendix using a reduction from the exact cover problem, in which the goal is to determine whether there exists a sub collection S' (of an input collection S of subsets of X) such that every member of X belongs to exactly one set in S'. Finally, we give explicit examples of why natural algorithms such as the greedy algorithm or alternating minimization fail to achieve multiplicative guarantees.

1.2 Prior Works

Robust regression. Robust regression has been well-studied in recent years. However, virtually all works rely on *distributional* assumptions on the input. The goal is to statistically recover the coefficient vector which minimizes the expected loss given a small number of corruptions to the distributional input. In contrast, we assume no distributional assumptions at all and our goal is to solve the computational problem given a corrupted input. Distributional assumptions have been well-studied in part because they are tractable; see [KKM18, KKK19, DKS19, CAT+20, ZJS20, BP21, JLST21] and the references within for a more comprehensive overview of distributional works.

Indeed, many such works such as [BJK15, BJKK17, SBRJ19] state their main motivation behind using distributional assumptions is that they believe the computational version, which we study, to be 'hard.' Since the focus of these works is statistical in nature, they do not rigorously justify why the robust regression problem in general is computationally hard. Our work aims to fill in this gap and initiate the study of hardness for robust regression. We remark that some works such as [BJK15, BJKK17, SBRJ19] assert that a proof of NP-hardness of robust regression is given in [SKPB12]. However, it seems [SKPB12] actually does not study robust regression at all; instead, it seems they study the problem of *sparse* regression, which, although related, does not imply anything about the hardness of robust regression. In fact, one of our results (Corollary 2.6) gives a reduction from robust regression to sparse regression, which implies that robust regression is a strictly easier problem than sparse regression. On the other hand, it is not clear whether hardness for sparse regression implies anything about hardness for robust regression.

Sparse regression. The sparse regression problem has also recently received significant attention, e.g., [Nat95, DMA97, Mah15, FKT15, HIM18, CYW19, GV21]. [Nat95] showed the NP-harndess of sparse regression while [FKT15] showed that assuming SAT cannot be solved by a deterministic algorithm in $O(n^{\log\log n})$ time, then no polynomial-time algorithm can find a k'-sparse vector x with $\|Ax-b\|_2 \leq \text{poly}(n)$, for $k' = k\cdot 2^{\log^{1-\delta} n}$. Subsequently, [HIM18] showed the sparse regression required $\Omega(n^{k/2})$ time, assuming the k-SUM conjecture from fine-grained complexity. [GV21] strengthened this lower bound to $\Omega(n^{k-\varepsilon})$ time, for any constant $\varepsilon > 0$, using the minimum-weight k-clique conjecture.

L_1-relaxation based algorithms such as basis pursuit [CDS98], Lasso [Tib96], and the Dantzig selector [CT07] have been developed for practical usage that do not involve worst-case inputs. For example, they consider the setting $b = Ax + g$, where the noise vector g is drawn from a Gaussian distribution and the design matrix A is well-conditioned. There has also been an extensive study on other penalty classes, such as the smoothly clipped absolute deviation penalty [FL01], the L_p norm for bridge estimators [FF93], or as the regularization function for M-estimators [LW15]. [CYW19] showed the NP-hardness of $O(n^{C_1}d^{C_2})$ multiplicative approximation for these common regularizations of the sparse

regression problem and fixed constants $C_1, C_2 > 0$, when the loss function is convex and the penalty function is sparse, such as L_1-relaxation. By comparison, we show bicriteria NP-hardness of *any* multiplicative approximation of the actual sparse regression problem, even when the sparsity constraint can be relaxed up to a multiplicative factor.

Other sparse optimization problems. Sparsity has also been highly demanded in other optimization problems. In this paper, we show that our algorithmic ideas also extend to the sparse PCA problem, which was first introduced by [dGJL07] and subsequently shown to be NP-hard by [MWA06]. In fact, it is NP-hard to obtain any multiplicative approximation if the input matrix is not PSD [Mag17] and to obtain a $(1+\varepsilon)$-multiplicative approximation when the input matrix is PSD [CPR16a], though [APKD15] gave an *additive* polynomial time approximation scheme based on the bipartite maximum weight matching problem. In practice, techniques for the more general PCA problem based on rotating previously studied PCA approaches based on rotation [Jol95] or thresholding [CJ95] the top singular vector of the input matrix seemed to suffice for specific applications. L_1 relaxations [JTU03] and similar heuristics [ZH05, ZHT06, SH08] have also been considered for the sparse PCA problem. Another line of direction considered semidefinite programming relaxations [dBEG08, AW08, dKNS20, CDWZ20, CBZ$^+$22].

Our work also connects to the problem of recovering the sparsest non-zero element in a linear subspace problem (see Theorem C.1). This problem is known to be NP-hard in the worst case [CP86]. On the positive side, there exists work on planted settings of the problem where the subspace is generated by the span of random Gaussian vectors along with a planted sparse vector; see [DH14] and references within.

2 Bicriteria Hardness of Approximation

In this section, we show the bicriteria hardness of approximation for both robust regression and sparse regression. Our results also generalize to loss functions that have no penalty on the zero vector and positive penalty on any nonzero vector. The bicriteria hardness result for robust regression is immediately implied by the results in [OWZ15], but it is not phrased in terms of the robust regression problem. We formalize the details below for completeness. In addition, we extend the bicriteria hardness result for sparse regression in the following section.

2.1 Bicriteria Hardness of Approximation for Robust Regression

Definition 2.1 ($\mathsf{MaxKLin}$)**.** *Suppose there exist a list of n linear equations of the form $a_1 x_{i_1} + \ldots + a_k x_{i_k} = b$, where $a_1, \ldots, a_k, b$ are constants from a ring R and $x_{i_1}, \ldots, x_{i_k}$ are variables from the set $x_1, \ldots, x_d$. Then the goal of the $\mathsf{MaxKLin}(R)$ problem is to assign values in R to the variables $x_1, \ldots, x_d$ such to maximize the total number of satisfied linear equations.*

Definition 2.2 ($\mathsf{Bounded} - \mathsf{Max}\Gamma 3 - \mathsf{Lin}$)**.** *Suppose there exist a list of n linear equations of the form $x_{i_1} + x_{i_2} - x_{i_3} = b$, such that $|b| \leq B$ for some fixed $B \in R$ and $x_{i_1}, x_{i_2}, x_{i_3}$ are variables from the set $x_1, \ldots, x_d$. Then the goal of the $\mathsf{Bounded} - \mathsf{Max}\Gamma 3 - \mathsf{Lin}(R)$ problem is to assign values in R to the variables $x_1, \ldots, x_d$ such to maximize the total number of satisfied linear equations.*

We use the notation $OPT_R(I)$ to denote the maximum fraction of equations of an instance I that can be satisfied when the equations are evaluated over R.

Theorem 2.1 (Theorem 1.2 in [OWZ15], Hardness of Approximation of $\mathsf{Bounded} - \mathsf{Max}\Gamma 3 - \mathsf{Lin}(R)$)**.** *For all constants $\varepsilon, \kappa \in (0,1)$ and $q \in \mathbb{N}$, given an instance of $\mathsf{Bounded} - \mathsf{Max}\Gamma 3 - \mathsf{Lin}(R)$, it is NP-hard to distinguish whether*

- *Completeness: There is a $(1-\varepsilon)$-good assignment over $\mathbb{Z}$, i.e., $OPT_{\mathbb{Z}}(I) \geq (1-\varepsilon)$.*

- *Soundness: There is no* $(1/q+\kappa)$*-good assignment over* $\mathbb{Z}_q$*, i.e.,* $OPT_{\mathbb{Z}_q}(I) \leq \frac{1}{q}+\kappa$.

Although there seems to be a typo in the statement of Lemma A.1 in [OWZ15], the corresponding proof provides the following guarantee:

Lemma 2.2 ([OWZ15])**.** *Given an instance* I *of* $\mathsf{Bounded}-\mathsf{Max}\Gamma 3-\mathsf{Lin}$*,* $OPT_{\mathbb{R}}(I) \geq \frac{1}{8} OPT_{\mathbb{Z}_q}(I)$.

By setting κ in the following statement to be $\frac{1}{8q}+\frac{1}{8}\kappa$ in the above formulations, we have that

Corollary 2.3. *For all constants* $\varepsilon \in (0,1), \kappa \in (0,1/8)$*, given an instance of* $\mathsf{Bounded}-\mathsf{Max}\Gamma 3-\mathsf{Lin}$*, it is NP-hard to distinguish whether*

- *Completeness: There is a* $(1-\varepsilon)$*-good assignment over* $\mathbb{Z}$*, i.e.,* $OPT_{\mathbb{Z}}(I) \geq (1-\varepsilon)$.
- *Soundness: There is no* κ*-good assignment over* $\mathbb{R}$*, i.e.,* $OPT_{\mathbb{R}}(I) \leq \kappa$.

Theorem 2.4. *Given a matrix* $A \in \mathbb{Z}^{n\times d}$*, a sparsity parameter* k*, and a vector* $b \in \mathbb{R}^n$*, let* $OPT = \min_{S,x} \|SAx - Sb\|_2$*, where the minimum is taken over all diagonal matrices* S *that have* k *entries that are zero and* $n-k$ *entries that are one and all* $x \in \mathbb{R}^d$*. Then for any* $C_1 > 1$ *(which can depend on the parameters* n, d*) and any constant* $C_2 > 1$*, it is NP-hard to find a matrix* S' *with* $C_2 k$ *entries that are zero and* $n - C_2 k$ *entries that are one and a vector* $x \in \mathbb{R}^d$ *such that* $\|S'Ax - S'b\|_2 \leq C_1 \cdot OPT$.

Proof. Let κ and ε be constants so that $\frac{1-\kappa}{\varepsilon} \geq C_2$. Given an instance I of $\mathsf{Bounded}-\mathsf{Max}\Gamma 3-\mathsf{Lin}$, set $k = \varepsilon n$, where n is the number of linear equations over the variables $x_1, \ldots, x_d$. We create the corresponding $n \times d$ matrix A by setting $A_{i,j} \pm 1$ if the coefficient of $x_j = \pm 1$ in the i-th linear equation. Otherwise, we set $A_{i,j} = 0$.

Observe that if there is a $(1-\varepsilon)$-good assignment over $\mathbb{Z}$, then there is a vector $x \in \mathbb{R}^d$ that satisfies $(1-\varepsilon)n$ linear equations. Thus by ignoring $k = \varepsilon n$ linear equations, the remaining coordinates of b are satisfied by the vector x. Hence, there exists a matrix S with k entries that are zero and $n-k$ entries that are one such that $SAx = Sb$ and in particular, $\|SAx - Sb\|_2 = 0$ and $OPT = 0$.

On the other hand, if there is no κ-good assignment over $\mathbb{R}$ to I, then any vector $x \in \mathbb{R}^d$ will satisfy fewer than κn linear equations. In particular, even by ignoring $C_2 k \leq (1-\kappa)n$ linear equations, there still exists some coordinate of b that is not satisfied by the vector x. Thus for every matrix S' with $C_2 k$ entries that are zero and $n - C_2 k$ entries that are one, we have $S'Ax \neq Sb'$ and in particular, $\|S'Ax - S'b\|_2 > 0$ and therefore $\|S'Ax - S'b\|_2 > C_1 \cdot 0$ for any $C_1 > 1$.

Hence, any algorithm that finds a matrix S' with $C_2 k$ entries that are zero and $n - C_2 k$ entries that are one, as well as a vector $x \in \mathbb{R}^d$ such that $\|S'Ax - S'b\|_2 \leq C_1 \cdot OPT$ can differentiate whether (1) there is a $(1-\varepsilon)$-good assignment over $\mathbb{Z}$, i.e., $OPT_{\mathbb{Z}}(I) \geq (1-\varepsilon)$ or (2) there is no κ-good assignment over $\mathbb{R}$, i.e., $OPT_{\mathbb{R}}(I) \leq \kappa$. Therefore by Corollary 2.3, it is NP-hard to find a matrix S' with $C_2 k$ entries that are zero and $n - C_2 k$ entries that are one and a vector $x \in \mathbb{R}^d$ such that $\|S'Ax - S'b\|_2 \leq C_1 \cdot OPT$. □

Finally, we observe that in the proof of Theorem 2.4, in one case, the resulting loss is the all zeros vector while in the other case, there are at least $n - C_2 k$ nonzero entries. Thus the proof generalizes to any loss function $\mathcal{L}$ such that $\mathcal{L}(0^n) = 0$ and $\mathcal{L}(x) > 0$ for $x \neq 0^n$.

2.2 Bicriteria Hardness of Approximation for Sparse Regression

We now extend the bicriteria hardness results from the previous section for the problem of sparse regression. Note that in the sparse regression problem, including as many variables as possible only helps us. For example, including as many columns as possible in linear regression only enlarges the column space and hence finds a possibly closer vector to the target. In the linear regression case, we prove the following theorem which states that it is NP-hard to choose a set of columns to ignore, even if we relax the number of ignored columns by a multiplicative factor.

Theorem 2.5. *Given a matrix $X \in \mathbb{R}^{a\times b}$, vector $c \in \mathbb{R}^a$, and a sparsity parameter t, let $OPT = \min_w \|Xw - c\|$ where $\|w\|_0 = t$. Then for any $C_1 > 1$ and any constant $C_2 > 1$, it is NP-hard to find w' satisfying $\|w'\|_0 = C_2 t$ such that $\|Xw' - c\| \leq C_1 \cdot OPT$.*

Proof. We will show that solving the sparse regression problem allows us to solve the robust regression problem. Let $A \in \mathbb{Z}^{n\times d}, b \in \mathbb{R}^n$, and k be the parameters in Theorem 2.4. Note that the matrix A in the proof of Theorem 2.4 satisfies $n > d$. Given A, we construct the matrix $X \in \mathbb{R}^{(n-d)\times n}$ so that the rows of X will span the space that is orthogonal to the column subspace of A, i.e., $XA = 0$. We also let $c = -Xb$, and $t = k$. We claim that if we can solve the sparse regression problem with this derived instance, then we can solve the robust regression problem of Theorem 2.4. Parameters C_1 and C_2 are the same in both theorems.

First, note that the OPT value in Theorem 2.4 is equal to 0: we can ignore k linear constraints and get 0 loss. This means that there exists a diagonal matrix S and a vector x such that $SAx - Sb = 0$ which implies $Ax - b = w$ for $\|w\|_0 = k$ i.e., w has $n - k$ zero entries. Therefore, the value of OPT in the sparse regression problem is also 0 by multiplying the equation $Ax - b = w$ by the matrix X since $X(Ax - b) = Xw$ implies $Xw = c$. Now suppose we can solve the sparse regression problem with any factor $C_1' > 1$ approximation while satisfying $\|w'\|_0 \leq C_2' k$. Since we are assuming a multiplicative approximation factor, w' must also evaluate to 0 loss in our objective. Furthermore, w' has $n - C_2' k$ zero entries.

Now $Xw' = c$ which is the same as $X(w' + b) = 0$. Thus, $w' + b$ lies in the orthogonal complement of the rows of X by our construction of X. Hence, $w' + b$ lies in the column space of A. Therefore, there exists some x' such that $Ax' = w' + b$ or in other words, $Ax' - b = w'$. Letting S' have the diagonal which is the indicator for the sparsity of w', we get that S' has $C_2 k$ zero entries on the diagonal. This implies $\|S'Ax' - S'b\|_2 = 0 \leq C_1 \cdot OPT$. Since finding such a pair S', x' is NP-hard from Theorem 2.4, it must be the case that the sparse regression selection problem stated in the current theorem statement must also be NP-hard, as desired. □

Observe that in the proof of Theorem 2.5, in one case, we have $Xw = c$ so that the resulting loss is the all zeros vector while in the other case, the resulting vector is nonzero. Therefore, the proof of Theorem 2.5 generalizes to any loss function $\mathcal{L}$ such that $\mathcal{L}(0^n) = 0$ and $\mathcal{L}(x) > 0$ for $x \neq 0^n$, giving Theorem 1.1.

Note that the reduction given in the above proof implies the following general statement: robust regression is 'easier' than sparse regression selection. That is, if there exists an algorithm for sparse regression, we can use it to solve robust regression as well.

Corollary 2.6. *Let $\mathcal{A}$ be an algorithm which solves the following sparse regression problem:*

$$\min_w \|Xw - c\|_2 \;\; s.t. \;\; \|w\|_0 = t$$

in time $f(X, c, t)$. Consider the robust regression problem of

$$\min_{S,x} \|S(Ax - b)\|_2$$

where S is constrained to be a diagonal matrix with $n - k$ one entries in the diagonal and k zero entries. $\mathcal{A}$ solves this problem in time $f(X', c', k)$ where X' is any matrix satisfying $X'A = 0$ and $c' = -X'b$.

Proof. The proof follows from the proof of Theorem 2.5. □

3 Fine-Grained Hardness

In this section, we prove a fine-grained hardness result for robust regression. We first need the following definition and theorem from [GV21].

Definition 3.1 (Definition 1 in [GV21], k-SLR$_p$)**.** *For any integer $k \geq 2$ and $1 \leq p \leq \infty$, the k-sparse regression problem with respect to the ℓ_p norm is defined as follows. Given a matrix $A \in \mathbb{R}^{M \times N}$, a target vector $b \in \mathbb{R}^M$, and a number $\delta > 0$, distinguish between:*

- *a **YES** instance: there is some k-sparse $x \in \mathbb{R}^N$ such that $\|Ax - b\|_p \leq \delta$, and*
- *a **NO** instance: for all k-sparse $x \in \mathbb{R}^N$, $\|Ax - b\|_p > \delta$.*

Theorem 3.1 (Theorem 3 in [GV21])**.** *For any integer $k \geq 4$, the k-SLR$_2$ problem requires time $\Omega(N^{k-o(1)})$ (randomized) time, unless the min-weight-k-clique conjecture is false.*

Using this theorem (or rather its proof), we can prove the hardness of the following decision version of sparse regression:

Problem 3.1. *Given a matrix $A \in \mathbb{R}^{M \times N}$, a target vector $b \in \mathbb{R}^M$, integer $0 < k \leq M$, and a number $\delta > 0$, distinguish between:*

- *a **YES** instance: there is some diagonal matrix S with k zeros on the diagonal and $n - k$ ones on the diagonal and some $x \in \mathbb{R}^N$ such that $\|S(Ax - b)\|_2 \leq \delta$, and*
- *a **NO** instance: for all diagonal matrices S with k zeros and $n - k$ ones on the diagonal and all $x \in \mathbb{R}^N$, $\|S(Ax - b)\|_2 > \delta$.*

We first recall the following theorem of [GV21] and the key underlying details of the proof, which we encapsulate in the following theorem.

Theorem 3.2. *[GV21] Let $G = (V, E)$ be a graph with N vertices with w_e denoting the integer weight of edge e. Let W, k be integer parameters and $Z = |\{(u, v) \notin E | u, v \in V\}|$ to be the number of non-edges in the graph G. Suppose that we know a partition of the N vertices into k blocks of size N/k such that if a k-clique of weight at most W exists then there is such a k-clique with exactly one vertex in each block [1]. Set*

$$\alpha = \sqrt{\max\left(1, \sum_{e \in E} w_e + 8W\right)}, \beta = \sqrt{\sum_{e \in E} w_e + 8W + \alpha^2 Z} \cdot \max\left(8Z, 50\left(\alpha^2 Z, + \sum_{e \in E} w_e\right)\right).$$

Define the matrix $A = \begin{pmatrix} C \\ D \end{pmatrix}$ and the vector $b = \begin{pmatrix} c \\ d \end{pmatrix}$ as follows: $C \in \mathbb{R}^{\binom{N}{2} \times N}$ with rows indexed by all unordered pairs of possible edges and columns indexed by vertices. For a possible edge $e = (u, v)$ not in G, the row of C corresponding to e has 2α in the columns of u and v and the corresponding entry of c has α. If $e = (u, v) \in E$ then the columns of u and v have the entry $2\sqrt{w_e}$ and the corresponding entry of c has $\sqrt{w_e}$. All other entries of C are 0. We also have $D \in \mathbb{R}^{k \in N}$

$$D = \begin{pmatrix} \beta 1 & 0 & \cdots & 0 \\ 0 & \beta 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \beta 1 \end{pmatrix}, \qquad d = \begin{pmatrix} \beta \\ \beta \\ \vdots \\ \beta \end{pmatrix}$$

[1] This is a standard assumption and is without loss of generality; see Section 2.2 of [GV21]. We can assume the k blocks are N/k consecutive integers of $\{1, \ldots, N\}$ in order.

where 1 *denotes a row vector of length* N/k *of all ones. Finally, set* $\delta = \sqrt{\sum_{e \in E} w_e + 8W + \alpha^2 Z} > 0$. *The following statements hold about* A *and* b*:*

- *(Completeness) If* G *contains a* k*-clique of weight at most* W *then we can let* x *be the indicator vector of the clique and we have* $\|Ax - b\|_2 \leq \delta$.
- *(Soundness) If there exists* x *such that* $\|Ax - b\|_2 \leq \delta$ *then* x *must be the indicator vector of a* k*-clique in* G *of weight at most* W *and furthermore, each block of* N/k *vertices must have exactly one vertex in the clique.*

We now formalize the conditions and provide the proof of Theorem 1.2.

Theorem 3.3. *For every* $\varepsilon > 0$*, there exists a sufficiently large* M *such that Problem 3.1 requires* $\Omega(M^{k/2+o(1)})$ *randomized time, unless the min-weight-*k*-clique conjecture is false.*

Proof. Consider the hard instance of Theorem 3.1 and 3.2, given by $A \in \mathbb{R}^{M \times N}, b \in \mathbb{R}^M, \delta > 0$, and parameter k where we set $M = \Theta(\binom{N}{2})$, specified fully later. Note that A in the proof in [GV21] is constructed from a min-weight-k-clique graph instance G and is composed of matrices C, D in their notation (see the statement of Theorem 3.2). We will augment A in two ways. First, note that D there is a $k \times N$ matrix. We will copy each row of D so that each row is copied $k+1$ times. The corresponding part of the b vector is also copied. We will also add N additional rows to A: one additional row $\tilde{C} \cdot x_i = 0$ for each $i \in [N]$ for some polynomially large factor $\tilde{C}$. The parameter k in Problem 3.1 will be equal to the same k in Theorem 3.1.

We now mimick the proof of Theorem 3.1 in [GV21]. We first handle the easier direction. We claim that if there is a k-clique of weight at most W for the k-clique instance corresponding to A, then we are in the **YES** case of Problem 3.1. Indeed, the proof of Theorem 3.1 in [GV21], summarized in Theorem 3.2, shows that we can let x be the indicator vector for the k-clique and the loss is at most δ by ignoring k of the $\tilde{C} \cdot x_i = 0$ constraints, setting the value of the ignored variables to be equal to 1, and letting the rest of the variables (which are not ignored) be equal to 0. The matrix D will contribute 0 loss, even with the augmentation.

We now handle the remaining case. We claim that if there exists S, x as in the **YES** case of Problem 3.1, then a k-clique of weight at most W exists in G. (Soundness case of Theorem 3.2). Let's focus on the D matrix and point out useful facts. Each row of D was copied sufficiently many times such that we can't ignore all the copies of any row of D. Furthermore, the x vector is partitioned into k blocks of size N/k and each row of D represents the constraint that one of these N/k blocks sums to 1. Lastly, the sum of each block must be in the range $[1 - 2\delta/\beta, 1 + 2\delta/\beta]$ (for the parameter β defined in Theorem 3.2) since otherwise, some row of D will already give loss at least δ, which cannot happen since we are assuming we are in the **YES** case of Problem 3.1. This implies the following two statements.

First, each of the N/k (consecutive) blocks of x must have at least 1 ignored variable. That is, for each block of N/k coordinates, there must be some i belonging to the block that has its corresponding $\tilde{C} \cdot x_i = 0$ constraint ignored. Otherwise, we will incur a loss larger than δ: the sum of the variables in each block must be $\Omega(1)$ due to the prior paragraph (otherwise we get a large loss for a row of D) but then summing the constraints $\tilde{C} \cdot x_i = 0$ for a block gives us a large loss. The only way to avoid this is to ignore one of the $\tilde{C} \cdot x_i = 0$ for an index i in a block. Since we can only ignore k variables and there are k blocks, it follows that each block has one ignored variable and overall, we only ignore the constraints of the form $\tilde{C} \cdot x_i = 0$. The un-ignored variables can now be made arbitrarily (polynomially) small in absolute value by setting $\tilde{C}$ to be sufficiently large since otherwise the constraint $\tilde{C} \cdot x_i = 0$ contributes more than δ loss.

The second statement we claim is that the ignored variable in each block must have its value in the range $[1 - 2\delta/\beta, 1 + 2\delta/\beta]$. This follows from the exact same reasoning used in the proof of Theorem 3.1.

To summarize, if the ignored variable is outside this range, then the corresponding row of D will induce loss larger than δ since we require the sum of all variables in a block to be close to 1. In our case, we can potentially get some contribution from the variables not ignored. However, as stated previously, their contribution to the loss can be made to be polynomially small by setting $\tilde{C}$ large enough.

Given these two statements, the proof of the reduction follows exactly as in [GV21]: the k ignored variables, one in each block, will correspond to the k-clique instance. To recap the argument, the same steps as in the proof of Theorem 3.1 (and 3.2 in [GV21]) show that we must have $(u, v) \in E$ for every x_u, x_v that are ignored. Otherwise, the loss of $\|Ax-b\|_2$ coming from the non-edge terms in the matrix C is much larger than δ already (see Page 8 in [GV21]). In addition, the proof of the fact that the k-clique instance has weight at most W also follows. This is because our vector x satisfies the same conditions required for the soundness case of Theorem 3.2; this portion of the proof in [GV21] proceeds by lower bounding the error from the matrix C alone, which is unchanged for us, and comparing it to the given hypothesis that the overall error is bounded by δ. Thus the same lower bound statements employed there also hold for us as they only consider the ignored variables. See the second half of Page 8 in [GV21].

Finally, the runtime bound required is at least $\Omega(N^{k-o(1)})$ by Theorem 3.1. Note that the final value of M is equal to $M = \binom{N}{2}+N+k^2 \leq 2N^2$. Therefore, the runtime is at least $\Omega(N^{k-o(1)}) = \Omega(M^{k/2+o(1)})$. □

4 Upper Bounds

In this section, we present upper bounds for sparse linear regression and robust linear regression. We utilize data structures for the following formulation of the c-approximate nearest neighbor problem:

Problem 4.1 (Approximate nearest neighbor)**.** *Given a set P of n points in a d-dimensional Euclidean space, the c-approximate nearest neighbor (c-ANN) problem seeks to construct a data structure that, on input query point q, reports any point $x \in P$ such that $\|x - q\|_2 \leq c \min_{p\in P} \|p - q\|_2$.*

In particular, we use the following data structure:

Theorem 4.1. *[AR15] For any fixed constant $c > 1$, there exists a data structure that solves the c-ANN problem in d-dimensional Euclidean space on n points with $O(dn^{\rho+o(1)})$ query time, $O(n^{1+\rho+o(1)} + dn)$ space, and $O(dn^{1+\rho+o(1)})$ pre-processing time, where $\rho = \frac{1}{2c^2-1}$.*

We first get an algorithm, i.e., Algorithm 1, for compressed sensing in the noise-less setting.

Theorem 4.2. *Suppose we are given $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$ such that there exists a k-sparse vector x satisfying $Ax = b$. Algorithm 1 returns a k-sparse z satisfying $\|Ax - Az\|_2 \leq \varepsilon$ in time*

$$\min_{c\geq 1} O\left(nk \cdot \left(\frac{12c \cdot d \cdot \|b\|_2}{\varepsilon}\right)^{\frac{k}{2}\cdot(1+1/(2c^2-1))}\right).$$

Proof. By scaling, we can assume b to be a unit vector without loss of generality. This means our ε term will implicitly be multiplied by a $\|b\|_2$ factor. We first show that z satisfies the approximation guarantees of Theorem 4.2. Note z is k-sparse since all y and $\tilde{y}$ considered in line 18 of Algorithm 1 are $k/2$-sparse in $\mathbb{R}^d$ (all $y's$ considered satisfy that Ay is in the image of some $k/2$ columns of A). Our task now is to show $\|Az - Ax\|_2 \leq \varepsilon$. Now let $x = x_1 + x_2$ be any division of x into the sum of two $k/2$-sparse vectors x_1 and x_2. Define $b_1 = Ax_1, b_2 = Ax_2$ so that $b = b_1 + b_2$ and let y be such that Ay is the closest point in some net $\mathcal{N}$ (considered in line 6 of the algorithm) to b_1. By the choice of our net, we know that $\|Ay - Ax_1\|_2 \leq \delta$. We have

$$\|(b - Ay) - b_2\|_2 = \|(b - Ay) - (b - Ax_1)\|_2 = \|Ay - Ax_1\|_2 \leq \delta. \tag{1}$$

Algorithm 1 Compressed Sensing Upper Bound

1: **Input:** Matrix $A \in \mathbb{R}^{n \times d}$, vector $b \in \mathbb{R}^d$, sparsity k, accuracy ε
2: **procedure** CompressedSensing-UpperBound(A, b, k, ε)
3: $\delta \leftarrow \varepsilon/((2c+2)$
4: $S \leftarrow \emptyset$
5: **for** each choice T of $k/2$ columns of A **do**
6: $\mathcal{N} \leftarrow \delta$ net over the image of A_T ▷ A_T denotes the submatrix of A with only columns in T
7: **for** each $b' = Ay \in \mathcal{N}$ **do**
8: $S \leftarrow S \cup (b', y, T)$
9: **end for**
10: **end for**
11: $\mathcal{D} \leftarrow c$-approximate nearest neighbor data structure for $\{b' \mid (b', y, T) \in S\}$
12: Best $\leftarrow \infty$
13: **for** each $(b', y, T) \in S$ **do**
14: $b'' \leftarrow b - b'$
15: $(\tilde{b}, \tilde{y}, \tilde{T}) \leftarrow$ output of $\mathcal{D}$ on query b''
16: Best $\leftarrow \min(\text{Best}, \|b' + b'' - b\|_2)$ ▷ We are extending $y, \tilde{y}$ to $k/2$ sparse vectors in $\mathbb{R}^d$ in the natural way, i.e., supported on the coordinates of T and $\tilde{T}$ respectively
17: **if** Best is updated **then**
18: Associate vector $z = y + \tilde{y}$ with Best
19: **end if**
20: **end for**
21: **Return** the vector z associated with the variable Best
22: **end procedure**

Letting $b' = Ay$ for this y and considering this choice of y and b' in the loop on line 15 of Algorithm 1, the above calculation shows the existence of a $b'' = b - b'$ such that $\|b_2 - b''\| \leq \delta$. By the construction of $\mathcal{D}$ in line 11 of Algorithm 1, it follows that $\mathcal{D}$ will output a $\tilde{b}$ on query b'' such that

$$\|b'' - \tilde{b}\|_2 \leq 2c\delta. \tag{2}$$

This is because there must exist some $\bar{y} \in \mathcal{N}$ such that $\|Ax_2 - A\bar{y}\|_2 \leq \delta$ by our net. Thus since $\|b_2 - b''\| \leq \delta$, we have $\|b'' - \bar{b}\|_2 \leq 2\delta$ by triangle inequality and (2) follows since $\mathcal{D}$ is only guaranteed to return a c-approximate neighbor.

Altogether, for this y and corresponding $\tilde{y}$ from line 17 of Algorithm 1, we have

$$\begin{aligned}
\|A(y+\tilde{y}) - b\|_2 &= \|A(y+\tilde{y}) - (Ax_1 + Ax_2)\|_2 \\
&= \|(Ay - Ax_1) + (A\tilde{y} - Ax_2)\|_2 \\
&\leq \|A(y - x_1)\|_2 + \|A\tilde{y} - b_2\|_2 \\
&\leq \delta + \|A\tilde{y} - b_2\|_2 \qquad \text{(from (1))} \\
&\leq \delta + \|\tilde{b} - b''\|_2 + \|b'' - b_2\|_2 \qquad \text{(by triangle inequality and } A\tilde{y} = \tilde{b}) \\
&\leq \delta + 2c\delta + \delta \qquad \text{(from (2))} \\
&= (2c+2)\delta \\
&\leq \varepsilon,
\end{aligned}$$

or in other words, $\|A(x - z)\|_2 \leq \varepsilon$, as desired.

We now compute the runtime of Algorithm 1. The size of $\mathcal{N}$ is upper bounded by

$$|\mathcal{N}| \leq \left(\frac{3\cdot}{\delta}\right)^{k/2} \leq \left(\frac{12c}{\varepsilon}\right)^{k/2}.$$

Picking all the choice of T and looping over all the vectors in $\mathcal{N}$ in the for loop of line 5 of Algorithm 1 takes time at most $O(d^{k/2} \cdot |\mathcal{N}| \cdot nk)$. Initializing $\mathcal{D}$ on the set S is dominated by the $|S|$ many queries performed on $\mathcal{D}$, which we discuss now. Looping over S and querying $\mathcal{D}$ in the for loop on line 15 takes time $O(|S| \cdot k \cdot |S|^{1/(2c^2-1)})$ by the guarantees of approximate nearest neighbor search [AR15] in Theorem 4.1. Note that $|S| \leq d^{k/2} \cdot |\mathcal{N}|$. Putting everything together, the total runtime is

$$O\left(d^{k/2} \cdot |\mathcal{N}| \cdot nk + k \cdot |S| \cdot |S|^{1/(2c^2-1)}\right) = O\left(nk \cdot \left(\frac{12c \cdot d}{\varepsilon}\right)^{\frac{k}{2}\cdot(1+1/(2c^2-1))}\right). \qquad \square$$

By modifying the size of the net, we can also find a sparse vector z that is arbitrarily close to the sparse vector x as well.

Corollary 4.3. *Suppose we are given $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$ such that there exists a k-sparse vector x satisfying $Ax = b$. There exists an algorithm that returns a k-sparse z satisfying $\|z - x\|_2 \leq \varepsilon$ in time*

$$\min_{c\geq 1} O\left(nk \cdot \left(\frac{12c \cdot d \cdot \kappa \cdot \|b\|_2}{\varepsilon}\right)^{\frac{k}{2}\cdot(1+1/(2c^2-1))}\right)$$

where $\kappa = \sigma_{max}(A)/\sigma_{min}(A)$ where the ratio is over non-zero singular values.

Proof. We modify Algorithm 1 by setting $\varepsilon' = \varepsilon/\kappa$ in Theorem 4.2. This implies $\|Az - Ax\|_2 \leq \varepsilon/\kappa$. Letting A^+ denote the pseudo-inverse of A, we have that

$$\|x - z\|_2 = \|A^+A(x - z)\|_2 \leq \|A^+\|_2\|A(x - z)\|_2 \leq \kappa \cdot \frac{\varepsilon}{\kappa} \leq \varepsilon,$$

as desired. The runtime follows from the parameter change. $\square$

Remark 4.4. *By letting $c = 2$ and under the assumption $\|b\|_2 = O(1)$, we achieve the runtime of $nk \cdot (O(d)/\varepsilon)^{k/2\cdot(1+1/7)}$. By letting $c = \Theta(1/\sqrt{\varepsilon})$, we can achieve the runtime of $nk \cdot (O(d)/\varepsilon^{1.5})^{k/2\cdot(1+\varepsilon)}$. In general, the best choice for c depends on the relationship between d and k.*

Remark 4.5. *Note that in Corollary 4.3, we can replace the parameter κ by the largest condition number κ' of any $n \times k$ submatrix of A since $x - z$ is a k-sparse vector. This is an improvement as $\kappa' \leq \kappa$.*

Now consider the setting where x is a binary signal or each coordinate has a finite number of choices in general. This setting is motivated by a number of applications including wideband spectrum sensing, wireless networks, group testing, error correcting codes, spectrum hole detection for cognitive radios, massive Multiple-Input Multiple-Output (MIMO) channels, etc. to name a few [CRTV05, RHE14, ALLP12, DR13, EYOR13, KKLP16, Fos18, KE12, NR12, MYL$^+$10].

In this setting, we can *recover* x perfectly in roughly $O(d)^{k/2}$ time using Algorithm 2.

Theorem 4.6. *Suppose we are given $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$ such that there exists a unique k-sparse vector x satisfying $Ax = b$. Furthermore, suppose that each coordinate of x must lie in the set $\{0\} \cup U$. Algorithm 2 recovers x exactly in time*

$$nk(d|U|)^{k/2} + O(1) \cdot (d|U|)^{k/2}.$$

Algorithm 2 Compressed Sensing Upper Bound for Finite Valued Signals

1: **Input:** Matrix $A \in \mathbb{R}^{n\times d}$, vector $b \in \mathbb{R}^d$, sparsity k, accuracy ε
2: **procedure** CompressedSensing-UpperBound-Finite(A, b, k, ε)
3: $S \leftarrow \emptyset$
4: **for** each choice T of $k/2$ columns of A **do**
5: **for** each $w = (w_i)_{i\in T} \in L^{k/2}$ **do**
6: $b' \leftarrow \sum_{i\in T} w_i A_{*,i}$ ▷ $A_{*,i}$ denotes the ith column of A
7: $S \leftarrow S \cup (b', T, w)$
8: **end for**
9: **end for**
10: $\mathcal{D} \leftarrow$ hashtable for the values $\{b' | (b', T, w) \in S\}$
11: Best $\leftarrow \infty$
12: **for** each $(b', T, w) \in S$ **do**
13: $b'' \leftarrow b - b'$
14: $(b'', T'', w'') \leftarrow$ output of $\mathcal{D}$ on query b'' ▷ If b'' is not in hash table, continue for loop on step 12
15: Best $\leftarrow \min(\text{Best}, \|\sum_{i\in T} w_i A_{*,i} + \sum_{i\in T''} w''_i A_{*,i} - b\|_2)$
16: **if** Best is updated **then**
17: Associate vector $z = \sum_{i\in T} w_i A_{*,i} + \sum_{i\in T''} w''_i A_{*,i}$ with Best
18: **end if**
19: **end for**
20: **Return** the vector z associated with the variable Best
21: **end procedure**

Proof. First we prove the approximation guarantee. Consider $x = x_1 + x_2$ be any decomposition of x into sum of two $k/2$-sparse vectors. Consider the for loop in step 10 of Algorithm 2. We must consider $b' = Ax_1$ at some iteration since we looped over all possible $k/2$ sparse vectors in step 4 of Algorithm 2. For this choice of b', we have that $b'' = b - b'$ satisfies $Ax_2 = b''$. Therefore, the hash table $\mathcal{D}$ will return x_2 on query b''. Finally since x_1 and x_2 have disjoint support, the vector $z = \sum_{i\in T} w_i A_{*,i} + \sum_{i\in T''} w''_i A_{*,i}$ is indeed equal to x as desired.

We now analyze the runtime. Forming the set S takes time at most $nk(d|U|)^{k/2}$ time and querying the hash table takes $O(1)$ time each in expectation. Thus the overall runtime is $nk(d|U|)^{k/2} + O(1) \cdot (d|U|)^{k/2}$. □

Remark 4.7. *In the case of binary vectors, $U = \{1\}$ so we achieve the runtime of $(nk + O(1)) \cdot d^{k/2}$.*

Using Corollary 2.6, we can also get an algorithm for robust regression, formalizing the conditions of Theorem 1.3.

Theorem 4.8. *Suppose we are given $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$ such that*

$$\min_{S,x} \|S(Ax - b)\|_2 = 0$$

where S is constrained to be a diagonal matrix with $n - k$ one entries in the diagonal and k zero entries. Algorithm 3 returns a diagonal matrix S' with $n - k$ one entries in the diagonal and k zero entries and a x' such that $\|S'(Ax' - b)\|_2 \le \varepsilon$ in time

$$\min_{c\ge 1} O\left(nk \cdot \left(\frac{12c \cdot n \cdot \|b\|_2}{\varepsilon}\right)^{\frac{k}{2}\cdot(1+1/(2c^2-1))}\right).$$

Algorithm 3 Robust Regression Upper Bound

1: **Input:** Matrix $A \in \mathbb{R}^{n \times d}$, vector $b \in \mathbb{R}^d$, sparsity k, accuracy ε
2: **procedure** RobustRegression-UpperBound(A, b, k, ε)
3: **if** Columns of A span all of $\mathbb{R}^n$ **then**
4: Let x' be such that $Ax' = b$
5: **Return** any diagonal matrix S' with $n - k$ one entries on diagonal and x'
6: **end if**
7: Let X be such that $XA = 0$ with orthonormal rows
8: $c \leftarrow -Xb$
9: $z \leftarrow$ output of Algorithm 1 on input (X, c, k, ε)
10: $x' \leftarrow \arg\min_{x'} \|Ax' - b - z\|_2$
11: $S' \leftarrow$ diagonal matrix with diagonal entry encoding the zero coordinates of z ▷ Note z has k non zero coordinates and thus, S' has $n - k$ ones on the diagonal and k zeros
12: **Return** S' and x'
13: **end procedure**

Proof. We assume we are not in the if statement case in line 3 of Algorithm 3 since otherwise we are done. Consider the quantities X and c associated with the robust regression instance arising from Corollary 2.6. Now from Corollary 2.6, we know the following statements: first, we know that all non zero singular values of X must be 1 since XX^T is the identity matrix and the non zero eigenvalues of X^TX and XX^T are the same. Second, we know that $\|z - w\|_2 \leq \varepsilon$ where $Xw = c$. This implies that if $Xz = c'$ then $\|c - c'\| \leq \varepsilon$. Therefore, we have $Xz = c + e$ or in other words, $X(z + b) = e$ for $\|e\|_2 \leq \varepsilon$. Now write $z + b = v_1 + v_2$ where v_1 is the orthogonal projection of $z + b$ onto the column span of A. We know that $Xv_1 = 0$ by definition and thus, there exists a $\tilde{x}$ such that $A\tilde{x} = v_1$. This $\tilde{x}$ satisfies

$$\|A\tilde{x} - (z + b)\|_2 = \|v_2\|_2 = \|e\|_2 \leq \varepsilon.$$

Therefore, we know that x' chosen in line 10 of Algorithm 3 satisfies $\|Ax' - b - z\|_2 \leq \varepsilon$ as well. Finally, z is a k-sparse vector since it is the output of Algorithm 1 which implies that our choice of S' in line 11 of Algorithm 3 gives us

$$\|S'(Ax' - b)\|_2 \leq \varepsilon$$

as desired.

We now compute the runtime of Algorithm 3. The runtime is dominated by the call to Algorithm 1. Note that our matrix X has all non-zero singular values equal to 1 so we can take κ in Theorem 4.2 (more specifically in Corollary 4.3) to be equal to 1. Furthermore, we can check that $\|c\|_2 \leq \|b\|_2$. □

Remark 4.9. *The same comments as in Remark 4.4 apply, except we no longer have a κ dependency and therefore do not need any assumptions on its value.*

5 Sparse PCA

In this section we present our results for the sparse PCA problem which is defined as follows. Let A be a PSD matrix of rank r. The goal of sparse PCA is to solve

$$\max_{\|v\|_2 = 1,\, \|v\|_0 \leq k} v^T A v. \tag{3}$$

Sparse PCA is known to be NP-hard to solve exactly and approximate with a $1 - \delta$ factor for some small constant $\delta > 0$ [CPR16b]. It is also known how to obtain a k-sparse unit vector v which achieves at least a $1 - \varepsilon$ approximation to the objective in time $O((4/\varepsilon)^r \cdot n \cdot k^2)$ [APKD15].

We obtain an algorithm with an improved dependence on the exponent of $1/\varepsilon$ via a novel connection to the computational geometry and our ideas from the prior sections. We need the following theorems from [Cha18] about the runtime of approximating the diameter and the bichromatic farthest pair of point sets.

Theorem 5.1. *Consider $P \subset \mathbb{R}^d$ with $|P| = n$. Let*

$$\operatorname{diam}(P) = \max_{x,y \in P} \|x - y\|_2$$

denote the diameter of P. There exists an algorithm which computes x', y' such that

$$\|x' - y'\|_2 \geq (1 - \varepsilon) \operatorname{diam}(P)$$

in time $\widetilde{O}(nd/\sqrt{\varepsilon} + (1/\varepsilon)^{d/2+1})$.

Theorem 5.2. *Consider $P, Q \subset \mathbb{R}^d$ with $|P \cup Q| = n$. There exists an algorithm which computes $x' \in P$ and $y' \in Q$ such that*

$$\|x' - y'\|_2 \geq (1 - \varepsilon) \max_{x \in P, y \in Q} \|x - y\|$$

in time $\widetilde{O}(nd/\sqrt{\varepsilon} + (1/\varepsilon)^{d/2+1})$.

We now show Theorem 1.5. The main idea behind our algorithm is to note that since $A = B^T B$, we have $v^T A v = \|Bv\|_2^2$. Now we employ a trick from our previous algorithmic result: we partition v into the difference of two $k/2$ sparse vectors $x_1 - x_2$. This gives us $v^T A v = \|Bx_1 - Bx_2\|_2^2$. Thus, maximizing the original quadratic form over A reduces to maximizing the distance between the points $\{Bx_1\}$ and $\{Bx_2\}$ where x_1 and x_2 range over all $k/2$ sparse vectors. To do this, we invoke the `Bichromatic-Farthest-Pair` guarantee from [Cha18] to find the best pair of $k/2$ sparse vectors, say $y_1, -y_2$.

One technical issue is we must ensure the supports of y_1 and y_2 are *disjoint*. Since we have to maximize over unit norm vectors v, if the supports of y_1 and y_2 overlap then we cannot say anything about the norm of $y_1 - y_2$. To get around this, we first randomly partition $[n]$ into two disjoint sets and only search over $k/2$ sparse y_1 with support completely contained in the first set and vice versa for y_2. We formalize the argument below.

Theorem 5.3. *Algorithm 4 returns a unit vector u satisfying $\|u\|_0 \leq k$ such that*

$$u^T A u \geq (1 - \varepsilon) \max_{\|v\|_2 = 1, \|v\|_0 \leq k} v^T A v$$

with probability $1 - \exp(-\Omega(k\varepsilon^2))$. The runtime is

$$\widetilde{O}\left(\frac{k^2}{\varepsilon} \cdot \left(r\left(\frac{n\kappa}{\varepsilon}\right)^{k(1+\varepsilon)/2} + \left(\frac{1}{\varepsilon}\right)^{r/2+1}\right)\right)$$

where κ is the ratio of the largest and smallest non-zero singular values of A.

Proof. We first prove the approximation guarantee. Let v be the optimum k-sparse unit vector. Note that with probability $1 - \exp(-\Omega(k\varepsilon^2))$, via a Chernoff bound, both U_1 and U_2 contain at least $k(1-\varepsilon)/2$ number of support indices of v. We now condition in this event. Let $v = x_1 - x_2$ be a decomposition of v into the difference of two vectors such that the support of x_1 is contained entirely in U_1 and similarly, the support of x_2 is contained entirely in U_2. We know that we will loop over some $z \in \mathcal{N}'$ in step 9 and some T_1 in step 12 of Algorithm 4 which satisfies $|\|x_1\|_2^2 - z| \leq \varepsilon/2$ and T exactly encodes the support of x_1. Similarly, we will loop over some T_2 in step 20 of Algorithm 4 such that T_2 exactly encodes the support of x_2. Now let y_1 and y_2 satisfy the following:

- $|\|y_1\|_2^2 - \|x_1\|_2^2| \leq \varepsilon/2$, $|\|y_2\|_2^2 - \|x_2\|_2^2| \leq \varepsilon/2$,
- the supports of y_1 and y_2 exactly match those of x_1 and x_2 respectively, and
- $\|y_1 - x_1\|_2 \leq \delta$, $\|y_2 - x_2\|_2 \leq \delta$.

Such a y_1, y_2 exist because of the net $\mathcal{N}$ and we looped over all choices k_1, k_2 such that $k_1 + k_2 = k$. We know that $w = y_1 - y_2$ satisfies

$$\|w\|_2^2 = \|y_1\|_2^2 + \|y_2\|_2^2 \geq 1 - \varepsilon,$$

since y_1, y_2 have disjoint support. Furthermore, we have

$$\|B(x_1 - y_1)\| \leq \varepsilon/\kappa$$

and

$$\|B(x_2 - y_2)\| \leq \varepsilon/\kappa$$

by our choice of δ in the net $\mathcal{N}$. Furthermore,

$$v^T A v = v^T B^T B v = \|Bv\|_2^2 = \|Bx_1 - Bx_2\|_2^2$$

so by the above calculations,

$$w^T A w = \|By_1 - By_2\|_2^2 \geq (1 - \varepsilon)\|Bx_1 - Bx_2\|_2^2 = v^T A v.$$

The guarantees of `Bichromatic-Farthest-Pair` imply that we find a w' such that

$$w'^T A w' \geq (1 - \varepsilon) w^T A w \geq (1 - O(\varepsilon)) v^T A v$$

in step 29 of Algorithm 4. Furthermore, w' has norm at least $1 - \varepsilon$ by our requirements on y_1 and y_2 in Algorithm 4 so we get the desired approximation.

We now analyze the runtime. The runtime is consists of guessing over k_1 a k_2 with is $O(k^2)$ time and guessing over z which has $O(1/\varepsilon)$ choices. Looping over the choices of k_1 columns (or k_2 columns) is time $O(n^{k(1+\varepsilon)/2}(\kappa/\varepsilon)^{k(1+\varepsilon)/2})$. The total number of points in the each instance of `Bichromatic-Farthest-Pair` used is $O(n^{k(1+\varepsilon)/2}(\kappa/\varepsilon)^{k(1+\varepsilon)/2})$ and all vectors in the instance are in dimension r. Invoking Theorem 5.2, the overall runtime is $\widetilde{O}(k^2/\varepsilon \cdot (r(n\kappa/\varepsilon)^{k(1+\varepsilon)/2} + (1/\varepsilon)^{r/2+1}))$. □

Remark 5.4. *Note that we can think of the parameter k as much smaller than r. Thus the dominant term in our runtime is $(1/\varepsilon)^{r/2+1}$ which improves upon the dependence of $(4/\varepsilon)^r$ as $(4/\varepsilon)^r \gg (1/\varepsilon)^{r/2+1}$.*

5.1 Sparse PCA with Limited Alphabet

In this section, we consider a slight variation of `SparsePCA` where the entries in v are limited to a small alphabet. Formally, we consider the problem of

$$\max_{\forall i: v_i \in \{-L, \ldots, L\},\, \|v\|_0 \leq k} v^T A v$$

where A is a $n \times n$ PSD matrix of rank r. This formulation is motivated by its connection the the Densest k-Subgraph problem where we wish to maximize $v^T A v$ over vectors v with a limited range of choices per coordinate but the matrix A is not necessarily PSD which holds in our case; see [CPR16a] for more information about the Densest k-Subgraph problem.

For a relaxation of this version, we can also obtain an algorithm with an exponentially better dependence on ε than the result from [APKD15] via a novel connection to the computational geometry problem of `Diameter`.

Algorithm 4 Sparse PCA Upper Bound

Input: PSD Matrix $A \in \mathbb{R}^{n \times n}$ of rank r, sparsity k, accuracy ε
procedure SparsePCA-UpperBound(A, k, ε)
  Compute B such that $B^T B \leftarrow A$ ▷ B is a $r \times n$ matrix
  $\kappa \leftarrow \sigma_{\max}(A)/\sigma_{\min}(A)$ ▷ The min / max is over non-zero singular values
  $\delta \leftarrow \varepsilon/\kappa$
  $U_1 \cup U_2 \leftarrow$ random partition of $[n]$ into two disjoint subsets
  $\mathcal{N}' \leftarrow \varepsilon/2$-net of the unit interval $[0, 1]$
  for each choice of k_1, k_2 such that $k_1 + k_2 = k$ and $k_1, k_2 \geq k(1-\varepsilon)/2$ **do**
    for each choice of $z \in \mathcal{N}'$ **do**
      $S_1 \leftarrow \emptyset$
      $S_2 \leftarrow \emptyset$
      for every choice T of k_1 columns of B restricted to the indices in U_1 **do** ▷ B_T is $r \times k_1$ matrix restricted to columns in T
        $\mathcal{N} \leftarrow \delta$-net of ball of radius 1 in $\mathbb{R}^{k_1}$
        for each $y \in \mathcal{N}$ **do**
          if $\|y\|_2^2 \in [z - \varepsilon/2, z + \varepsilon/2]$ **then**
            $S_1 \leftarrow S_1 \cap \{B_T y\}$
          end if
        end for
      end for
      for every choice T of k_2 columns of B restricted to the indices in U_1 **do** ▷ B_T is $r \times k_2$ matrix restricted to columns in T
        $\mathcal{N} \leftarrow \delta$-net of ball of radius 1 in $\mathbb{R}^{k_2}$
        for each $y \in \mathcal{N}$ **do**
          if $\|y\|_2^2 \in [1 - z - \varepsilon/2, 1 - z + \varepsilon/2]$ **then**
            $S_2 \leftarrow S_2 \cap \{-B_T y\}$
          end if
        end for
      end for
      $(B_{T_1} y_1, -B_{T_2} y_2) \leftarrow$ solution of `Bichromatic-Farthest-Pair` on the pair (S_1, S_2)
      $w = (y_1 - y_2)/\|y_1 - y_1\|_2$
      Keep track of the maximum value of $w^T A w$ encountered
    end for
  end for
  Return w that maximizes $w^T A w$ over all w's observed
end procedure

Theorem 5.5. *Algorithm 5 returns a u satisfying $\|u\|_0 \leq k$ such that*

$$u^T A u \geq (1 - \varepsilon) \max_{\forall i : v_i \in \{-L, \ldots, L\}, \|v\|_0 \leq k} v^T A v$$

and all the entries of u are in the set $\{-2L, \ldots, 2L\}$ in time $\widetilde{O}((1/\varepsilon)^{r/2+1} + rk \cdot ((2L+1)n)^{k/2}/\sqrt{\varepsilon})$.

Proof of Theorem 5.5. We prove correctness first. Consider the optimal v and let $v = x_1 + x_2$ for $k/2$ sparse vectors x_1, x_2. Note that

$$\text{OPT} = v^T A v = v^T B^T B v$$

Algorithm 5 Sparse PCA Limited Alphabet Upper Bound

Input: PSD Matrix $A \in \mathbb{R}^{n\times n}$ of rank r, sparsity k, accuracy ε
procedure SparsePCA-UpperBound-Limited(A, k, ε)
  Compute B such that $B^TB \leftarrow A$ ▷ B is a $r \times n$ matrix
  $S \leftarrow \emptyset$
  for every choice T of $k/2$ columns of B **do** ▷ B_T is $r \times k/2$ matrix restricted to columns in T
    for each $w = (w_i)_{i\in T} \in \{-L, \ldots, L\}^{k/2}$ **do**
      $y \leftarrow \sum_{i\in T} w_i A_{*,i}$ ▷ $A_{*,i}$ denotes the ith column of A
      $S \leftarrow S \cup \{B_T y\}$
    end for
  end for
  $(By, By') \leftarrow$ solution of `Diameter` on S using Theorem 5.1
  Return $u = y - y'$
end procedure

$$= x_1^TB^TBx_1 + 2x_1B^TBx_2 + x_2^TB^TBx_2$$
$$= \|Bx_1\|_2^2 + \|Bx_2\|_2^2 + 2\langle Bx_1, Bx_2\rangle.$$

Letting $x_2' = -x_2$, we get

$$v^TAv = \|Bx_1 - Bx_2'\|_2^2.$$

Now consider y, y' returned by Algorithm 5. From the guarantees of `Diameter`, it follows that

$$u^TBu = \|By - By'\|_2^2 \geq (1-\varepsilon)\|Bx_1 - Bx_2'\|_2^2 = (1-\varepsilon)\text{OPT}.$$

Finally, u is k sparse and its entries belong to $\{-2L, \cdots, 2L\}$. The runtime follows from Theorem 5.1 using the fact that $|S| = ((2L+1)d)^{k/2}$. □

Acknowledgments

Eric Price is supported by NSF awards CCF-2008868, CCF-1751040 (CAREER), and NSF IFML 2019844. Sandeep Silwal is supported by an NSF Graduate Research Fellowship under Grant No. 1745302, NSF TRIPODS program (award DMS-2022448), NSF award CCF-2006798, and Simons Investigator Award. Samson Zhou is supported by a Simons Investigator Award of David P. Woodruff.

References

[ALLP12] Erik Axell, Geert Leus, Erik G. Larsson, and H. Vincent Poor. Spectrum sensing for cognitive radio : State-of-the-art and recent advances. *IEEE Signal Processing Magazine*, 29:101–116, 2012. 12

[APKD15] Megasthenis Asteris, Dimitris S. Papailiopoulos, Anastasios Kyrillidis, and Alexandros G. Dimakis. Sparse PCA via bipartite matchings. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems*, pages 766–774, 2015. 5, 14, 16

[AR15] Alexandr Andoni and Ilya P. Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC*, pages 793–801, 2015. 10, 12

[AW08] Arash A. Amini and Martin J. Wainwright. High-dimensional analysis of semidefinite relaxations for sparse principal components. In *2008 IEEE international symposium on information theory*, pages 2454–2458, 2008. 5

[BGM20] Tamara Broderick, Ryan Giordano, and Rachael Meager. An automatic finite-sample robustness metric: Can dropping a little data change conclusions. *arXiv preprint arXiv:2011.14999*, page 16, 2020. 23

[BJK15] Kush Bhatia, Prateek Jain, and Purushottam Kar. Robust regression via hard thresholding. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems*, pages 721–729, 2015. 4, 24

[BJKK17] Kush Bhatia, Prateek Jain, Parameswaran Kamalaruban, and Purushottam Kar. Consistent robust regression. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, pages 2110–2119, 2017. 4, 24

[BP21] Ainesh Bakshi and Adarsh Prasad. Robust linear regression: optimal rates in polynomial time. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 102–115, 2021. 4

[CAT+20] Yeshwanth Cherapanamjeri, Efe Aras, Nilesh Tripuraneni, Michael I. Jordan, Nicolas Flammarion, and Peter L. Bartlett. Optimal robust linear regression in nearly linear time. *CoRR*, abs/2007.08137, 2020. 4

[CBZ+22] Agniva Chowdhury, Aritra Bose, Samson Zhou, David P. Woodruff, and Petros Drineas. A fast, provably accurate approximation algorithm for sparse principal component analysis reveals human genetic variation across the world. In *Research in Computational Molecular Biology - 26th Annual International Conference, RECOMB, Proceedings*, pages 86–106, 2022. 5

[CDS98] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM J. Sci. Comput.*, 20(1):33–61, 1998. 4

[CDWZ20] Agniva Chowdhury, Petros Drineas, David P. Woodruff, and Samson Zhou. Approximation algorithms for sparse principal component analysis. *CoRR*, abs/2006.12748, 2020. 5

[Cha18] Timothy M. Chan. Applications of chebyshev polynomials to low-dimensional computational geometry. *J. Comput. Geom.*, 9:3–20, 2018. 15

[CJ95] Jorge Cadima and Ian T. Jolliffe. Loading and correlations in the interpretation of principle compenents. *Journal of applied Statistics*, 22(2):203–214, 1995. 5

[CP86] Thomas F Coleman and Alex Pothen. The null space problem i. complexity. *SIAM Journal on Algebraic Discrete Methods*, 7(4):527–537, 1986. 5

[CPR16a] Siu On Chan, Dimitris Papailliopoulos, and Aviad Rubinstein. On the approximability of sparse PCA. In *Proceedings of the 29th Conference on Learning Theory, COLT*, pages 623–646, 2016. 5, 16

[CPR16b] Siu On Chan, Dimitris Papailliopoulos, and Aviad Rubinstein. On the approximability of sparse pca. In *COLT*, 2016. 14

[CRTV05] Emmanuel J. Candès, Mark Rudelson, Terence Tao, and Roman Vershynin. Error correction via linear programming. In *FOCS 2005*, 2005. 12

[CT07] Emmanuel Candes and Terence Tao. The dantzig selector: Statistical estimation when p is much larger than n. *The annals of Statistics*, 35(6):2313–2351, 2007. 4

[CW15] Kenneth L. Clarkson and David P. Woodruff. Sketching for M-estimators: A unified approach to robust regression. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 921–939, 2015. 2

[CWW19] Kenneth L. Clarkson, Ruosong Wang, and David P. Woodruff. Dimensionality reduction for tukey regression. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, 2019. 2

[CYW19] Yichen Chen, Yinyu Ye, and Mengdi Wang. Approximation hardness for A class of sparse optimization problems. *J. Mach. Learn. Res.*, 20:38:1–38:27, 2019. 4

[dBEG08] Alexandre d'Aspremont, Francis Bach, and Laurent El Ghaoui. Optimal solutions for sparse principal component analysis. *Journal of Machine Learning Research*, 9(7), 2008. 5

[dGJL07] Alexandre d'Aspremont, Laurent El Ghaoui, Michael I. Jordan, and Gert R. G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. *SIAM Rev.*, 49(3):434–448, 2007. 5

[DH14] Laurent Demanet and Paul Hand. Scaling law for recovering the sparsest element in a subspace. *Information and Inference: A Journal of the IMA*, 3(4):295–309, 2014. 5, 25

[dKNS20] Tommaso d'Orsi, Pravesh K. Kothari, Gleb Novikov, and David Steurer. Sparse PCA: algorithms, adversarial perturbations and certificates. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 553–564, 2020. 5

[DKS19] Ilias Diakonikolas, Weihao Kong, and Alistair Stewart. Efficient algorithms and lower bounds for robust linear regression. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 2745–2754, 2019. 4

[DMA97] Geoff Davis, Stephane Mallat, and Marco Avellaneda. Adaptive greedy approximations. *Constructive approximation*, 13(1):57–98, 1997. 4

[DR13] Przemyslaw Dymarski and Rafal Romaniuk. Sparse signal modeling in a scalable celp coder. *21st European Signal Processing Conference (EUSIPCO 2013)*, pages 1–5, 2013. 12

[EYOR13] Alexander Ens, Adnan Yousaf, Thomas Ostertag, and Leonhard Michael Reindl. Optimized sinus wave generation with compressed sensing for radar applications, 2013. 12

[FF93] LLdiko E. Frank and Jerome H. Friedman. A statistical view of some chemometrics regression tools. *Technometrics*, 35(2):109–135, 1993. 4

[FKT15] Dean P. Foster, Howard J. Karloff, and Justin Thaler. Variable selection is hard. In *Proceedings of The 28th Conference on Learning Theory, COLT*, volume 40, pages 696–709, 2015. 2, 4

[FL01] Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96(456):1348–1360, 2001. 4

[Fos18] Sophie Marie Fosson. Non-convex lasso-kind approach to compressed sensing for finite-valued signals. *arXiv: Optimization and Control*, 2018. 12

[GV21] Aparna Gupte and Vinod Vaikuntanathan. The fine-grained hardness of sparse linear regression, 2021. 2, 4, 8, 9, 10

[HIM18] Sariel Har-Peled, Piotr Indyk, and Sepideh Mahabadi. Approximate sparse linear regression. In *45th International Colloquium on Automata, Languages, and Programming, ICALP*, volume 107, pages 77:1–77:14, 2018. 4

[HK70] Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970. 2

[JLST21] Arun Jambulapati, Jerry Li, Tselil Schramm, and Kevin Tian. Robust regression revisited: Acceleration and improved estimation rates. *arXiv preprint arXiv:2106.11938*, 2021. 4

[Jol95] Ian T. Jolliffe. Rotation of principal components: choice of normalization constraints. *Journal of Applied Statistics*, 22(1):29–35, 1995. 5

[JTU03] Ian T. Jolliffe, Nickolay T. Trendafilov, and Mudassir Uddin. A modified principal component technique based on the lasso. *Journal of computational and Graphical Statistics*, 12(3):531–547, 2003. 5

[KBV09] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009. 24

[KE12] Nazim Burak Karahanoglu and Hakan Erdogan. A* orthogonal matching pursuit: Best-first search for compressed sensing signal recovery. *Digit. Signal Process.*, 22:555–568, 2012. 12

[KKK19] Sushrut Karmalkar, Adam R. Klivans, and Pravesh Kothari. List-decodable linear regression. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 7423–7432, 2019. 4

[KKLP16] Sandra Keiper, Gitta Kutyniok, Dae Gwan Lee, and Götz E. Pfander. Compressed sensing for finite-valued signals. *arXiv: Optimization and Control*, 2016. 12

[KKM18] Adam R. Klivans, Pravesh K. Kothari, and Raghu Meka. Efficient algorithms for outlier-robust regression. In *Conference On Learning Theory, COLT*, pages 1420–1430, 2018. 4

[LW15] Po-Ling Loh and Martin J. Wainwright. Regularized m-estimators with nonconvexity: Statistical and algorithmic theory for local optima. *The Journal of Machine Learning Research*, 16(1):559–616, 2015. 4

[Mag17] Malik Magdon-Ismail. Np-hardness and inapproximability of sparse PCA. *Inf. Process. Lett.*, 126:35–38, 2017. 5

[Mah15] Sepideh Mahabadi. Approximate nearest line search in high dimensions. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 337–354, 2015. 4

[MWA06] Baback Moghaddam, Yair Weiss, and Shai Avidan. Generalized spectral bounds for sparse LDA. In *Machine Learning, Proceedings of the Twenty-Third International Conference ICML*, pages 641–648, 2006. 5

[MYL+10] Jia Meng, Wotao Yin, Husheng Li, Ekram Hossain, and Zhu Han. Collaborative spectrum sensing from sparse observations using matrix completion for cognitive radio networks. *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3114–3117, 2010. 12

[Nat95] Balas K. Natarajan. Sparse approximate solutions to linear systems. *SIAM J. Comput.*, 24(2):227–234, 1995. 4

[NR12] Ukash Nakarmi and Nazanin Rahnavard. Bcs: Compressive sensing for binary sparse signals. *MILCOM 2012 - 2012 IEEE Military Communications Conference*, pages 1–5, 2012. 12

[OWZ15] Ryan O'Donnell, Yi Wu, and Yuan Zhou. Hardness of max-2lin and max-3lin over integers, reals, and large cyclic groups. *ACM Trans. Comput. Theory*, 7(2):9:1–9:16, 2015. 1, 2, 5, 6

[RHE14] Marco Rossi, Alexander M. Haimovich, and Yonina C. Eldar. Spatial compressive sensing for mimo radar. *IEEE Transactions on Signal Processing*, 62:419–430, 2014. 12

[SBRJ19] Arun Sai Suggala, Kush Bhatia, Pradeep Ravikumar, and Prateek Jain. Adaptive hard thresholding for near-optimal consistent robust regression. In *Conference on Learning Theory, COLT*, pages 2892–2897, 2019. 4, 24

[SH08] Haipeng Shen and Jianhua Z. Huang. Sparse principal component analysis via regularized low rank matrix approximation. *Journal of multivariate analysis*, 99(6):1015–1034, 2008. 5

[SKPB12] Christoph Studer, Patrick Kuppinger, Graeme Pope, and Helmut Bölcskei. Recovery of sparsely corrupted signals. *IEEE Trans. Inf. Theory*, 58(5):3115–3130, 2012. 4

[Tib96] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996. 2, 4

[TWZ+22] Murad Tukan, Xuan Wu, Samson Zhou, Vladimir Braverman, and Dan Feldman. New coresets for projective clustering and applications. In *International Conference on Artificial Intelligence and Statistics, AISTATS*, 2022. 2

[ZH05] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320, 2005. 5

[ZHT06] Hui Zou, Trevor Hastie, and Robert Tibshirani. Sparse principal component analysis. *Journal of computational and graphical statistics*, 15(2):265–286, 2006. 5

[ZJS20] Banghua Zhu, Jiantao Jiao, and Jacob Steinhardt. Robust estimation via generalized quasi-gradients. *CoRR*, abs/2005.14073, 2020. 4

A NP Hardness Result

We give an alternate proof of NP hardness for robust regression based on exact cover.

Problem A.1 (Exact Cover)**.** *Given a collection S of subsets of X, determine if there exists a sub collection S' of S such that every member of X belongs to exactly one set in S'.*

Problem A.2 (Robust Regression, Zero Cost Decision Version)**.** *Given $A \in \mathbb{R}^{n\times d}, b \in \mathbb{R}^n$, and integer $0 < k \leq n$, determine if there exists $T \subset [n]$ satisfying $|T| = k$ such that*

$$\min_{y\in\mathbb{R}^d} \|(Ay - b)_T\| = 0 \tag{4}$$

where $(Ax - b)_T$ denotes that we only measure the loss on the coordinates in T. The coordinates not in T are called ignored*.*

Lemma A.1. *Problem A.1 is reducible to Problem A.2.*

Proof. Consider an exact cover instance given by S, X. Let $n = 2|S| + |X|$ and $d = |S|$. We form the matrix $A \in \mathbb{R}^{n\times d}$ as follows. We have a variable y_i for the i th set in S for all i. The first $2|S| \times |S|$ block of A will be the constraints $y_i = 0$ and $1 - y_i = 0$. This also defines the b vector for this part of the matrix. The next $|X| \times |S|$ block of A will be the indicator matrix for the sets in S. That is, each column will be a $\{0, 1\}$ vector indicating which elements of X are in the set corresponding to the column. The part of the b vector for this block of A will all $1's$. Finally, we set $k = |X| + |S|$.

We now claim that Eq. (4) is equal to 0 iff an exact cover exists. In particular, we claim that Eq. (4) is equal to 0 iff y is the indicator vector for which sets in S to pick to be part of S', the exact cover. First, note that if an exact cover exists, letting $y_i = 1$ for the sets that are part of S' (and thus ignoring the $y_i = 0$ constraints), and letting $y_i = 0$ for sets that are not part of S' (and again ignoring the $1 - y_i = 0$ constraints) results in $(Ay - b)_T = 0$. Note that we have chosen to ignore exactly $n - k = |S|$ many constraints, one for each y_i.

We now show the other direction. Suppose that Eq. (4) holds. We first show that y must only have $0, 1$ entries. Let's focus on the first $2|S|$ constraints of A. If both of the constraints $y_i = 0$ and $1 - y_i = 0$ are not ignored for some i, then we automatically induce a non zero cost. Since we are assuming Eq. (4) holds, it implies that all variables y_i must only have one of $y_i = 0$ or $1 - y_i = 0$ present in T for all i and furthermore, only these types of constraints must be ignored in T. Therefore, $y \in \{0, 1\}^n$ and y represents an indicator vector for the second $|X| \times |S|$ block of A. Since the b vector for this block is the all 1's vector, this automatically implies that the sets chosen by y forms an exact cover of X, as desired. □

B Simple Counter Examples to Natural Algorithms for Robust Regression

In this section, we present particularly simple counter examples to natural algorithms for robust regression which have also been studied in applied works.

Greedy algorithm. First consider a greedy algorithm which fits a best fit linear regression on all data points and removes the k data points with the largest residuals, such as in Algorithm 1. Variants of this algorithm have been used in applied works such as [BGM20] to find the 'most influential' data points in econometric data analysis.

Now consider the following (x, y) data pairs: $(0, 10), (1, 0), (2, 0), (3, 0)$. We can generalize this example to any total number of data points by having multiple copies of each data point. Consider the simplest $k = 1$ case of robust regression where we wish to remove one point to minimize the regression loss. In the example given, it is clear that if we remove $(0, 10)$, zero loss is achieved by the line $y = 0$. The best fit on all of the points is given by $y = 7 - 3x$. We can check the residual for the $(0, 10)$ data point is 3 while the residual for $(1, 0)$ is 4. Thus, the greedy algorithm removes $(1, 0)$ which results in a suboptmial algorithm which performs arbitrarily worse compared to the true solution with 0 loss.

1. Given $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^d$, $y = \arg\min_x \|Ax - b\|_2$.
2. Let S be the $n \times n$ identity matrix and let $p_1, \ldots, p_k$ be the indices of the k coordinates of $Ay - b$ largest in magnitude.
3. For $i \in [k]$, set $S_{p_i,p_i} = 0$.
4. Output S and y.

Figure 1: Greedy algorithm for robust regression

Alternating minimization. We now consider another natural algorithm which performs alternating minimization: starting from an arbitrary S, it optimizes for x given the choice of S. Then using the resulting x, it optimizes for S and continues in this loop for a specified number of iterations. See Algorithm 2 for more details.

1. Given $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^d$, and number of iterations T, set S to be an arbitrary $n \times n$ diagonal matrix with $n - k$ ones on the diagonal and k zeros.
2. While # of iterations $< T$:
   (a) Set $y = \arg\min_x \|SAx - Sb\|_2$. (Optimize over x)
   (b) Let $p_1, \ldots, p_k$ be the indices of the k coordiantes $Ay - b$ largest in magnitude.
   (c) For $i \in [k]$, set $S_{p_i,p_i} = 0$. (Optimize over S)
3. Output S and $y = \arg\min_x \|SAx - Sb\|_2$.

Figure 2: Alternating minimization algorithm for robust regression

This class of algorithms is widely used in practice; for example, it was a key component in the winning submission for the Netflix Prize Competition [KBV09]. Alternating algorithms have also been considered for robust regression in the distributional setting [BJK15, BJKK17, SBRJ19]. Alternating minimization algorithms are especially useful where one is interested in minimizing a complex function of various parameters with the property that minimizing over specific subsets of the variables is tractable. Indeed, this is the case here: given S, finding x is just an instance of linear least squares with no restrictions and given x, the best S is given by discarding the k datapoints with the largest loss.

Our example for the greedy algorithm again serves as a simple counter example for the proposed alternating minimization algorithm for the most basic case of $k = 1$ in the robust regression problem. Suppose we start with the matrix S which removes or ignores the point $(1, 0)$. Doing so gives us the best fit line $y = 9.29x - 3.57$. However for this line, one can check that the point $(1, 0)$ would still have the largest residual among all four points. Therefore, the alternating minimization algorithm would not make any further progress as it would continue to select the point $(1, 0)$ to remove in all future iterations. We can check that if we started by removing the point $(2, 0)$ instead, the point $(1, 0)$ would still have the largest residual among all four data points in the resulting best fit line. Thus, we are back in the first case considered. If we start by removing $(3, 0)$, then $(3, 0)$ will have the largest residual among all four data points so the alternating minimization algorithm is again stuck. Therefore, the alternating minimization algorithm is guaranteed to return a suboptmial solution if we do not initialize S with the optimal choice.

C Polynomial-time Algorithm for Planted Instance of Robust Regression

In this section, we show that if the columns of the input matrix A are generated from a normal distribution and the measurement vector b has Hamming distance at most k from a planted solution b' that lies in the column span of A, then there is a polynomial time algorithm that solves the robust regression problem:

Theorem C.1. *Let C be a fixed constant and $k \leq C\sqrt{n}\log n$. Let the columns of an input matrix $A \in \mathbb{R}^{n\times d}$ be drawn independent and identically distributed from $\mathcal{N}(0, I_n)$. Let $b' \in \mathbb{R}^n$ lie in the column span of A. Then given a vector b such that $\|b - b'\|_0 \leq k$, there exists an algorithm that solves n linear programs and then uses polynomial time to solve the sparse linear regression problem with probability at least $2/3$, i.e., the algorithm finds a diagonal matrix $S \in \mathbb{R}^{n\times n}$ with $n - k$ nonzero entries along that diagonal that are set to 1 and a vector $x \in \mathbb{R}^d$ such that $\|S(Ax - b)\| = 0$.*

Our result is motivated by the following result of [DH14] which solve the problem of sparsest non-zero vector in a subspace in a planted setting as well.

Lemma C.2 (Theorem 1 in [DH14])**.** *Given a basis of vectors $w_1, \ldots, w_{d+1} \in \mathbb{R}^n$ for a subspace spanned by vectors $v, v_1, \ldots, v_d \in \mathbb{R}^n$, where $v_i \sim \mathcal{N}(0, I_n)$ for all $i \in [d]$, then there exists an absolute constant $C > 0$ and an algorithm that solves n linear programs and uniquely recovers the vector v with probability at least $2/3$, for $\|v\|_0 \leq C\sqrt{n}\log n$.*

Proof of Theorem C.1. Given a matrix A whose columns $u_1, \ldots, u_d \sim \mathcal{N}(0, I_n) \in \mathbb{R}^n$ and a vector $b \in \mathbb{R}^n$ such that there exists a vector $b' \in \mathbb{R}^n$ in the column span of A with $\|b-b'\|_0 \leq k$, we construct the vectors $w_1, \ldots, w_{d+1}$ by taking an arbitrary basis over the $d + 1$ vectors $b, u_1, \ldots, u_d$. The vectors $w_1, \ldots, w_{d+1}$ also form a basis for the subspace spanned by the vectors $b - b', u_1, \ldots, u_d$, since b' is in the column span of A and thus spanned by $u_1, \ldots, u_d$. Since $\|b - b'\|_0 \leq k$, then by Lemma C.2, there exists an algorithm that solves n linear programs and uniquely recovers the vector $b - b'$ with probability at least $2/3$. Because we are given b as input, we can thus determine the vector b', as well as a vector $x \in \mathbb{R}^d$ such that $Ax = b'$. By setting S to be the diagonal matrix S with at most k zeros and at least $n - k$ ones on the diagonal such that the zero entries on the diagonal of S are located precisely in the coordinates for which $b - b'$ is nonzero, then we have $\|S(Ax - b)\| = \|S(b - b')\| = 0$, since $S(b - b') = 0^n$. □