Fine-Grained Complexity of Sparse Linear Regression

SPUR+ Final Paper, Summer 2020
Aparna Ajit Gupte and Kerri Lu
Mentor: Frederic Koehler
Project suggested by: Frederic Koehler

July 29, 2020

Abstract

We study the problem of sparse linear regression over the rationals from the perspective of fine-grained complexity. More precisely, we consider the noiseless/realizable setting where $Y = \langle X, w \rangle$, w is a k-sparse vector, $X \sim N(0, \Sigma)$ and the goal of the algorithm is to exactly reconstruct w given access to a sampling oracle generating iid copies of (X, Y). Under the Exponential Time Hypothesis, we show that solving this problem requires time $n^{\Omega(k)}$, give more precise bounds under stronger hypotheses, and also prove a nontrivial upper bound of the form n^{k-1} beating the trivial brute force approach.

1 Introduction

In this paper, we study the the fine-grained complexity of solving k-sparse linear regression without noise. This is a fundamental problem in machine learning, statistics, and signal processing which has seen intensive study in the past few decades. In general, this problem can be formulated in terms of matrices as

$$y = Ax + \xi$$

where ξ is some kind of noise vector, x is a k-sparse vector and $A : m \times n$ is the so-called *design matrix* and the goal is with high probability, to reconstruct x accurately given A and Y. (We explain the connection to the formulation in the abstract later.)

Both the settings with and without noise have been intensely studied; for simplicity, in this paper we focus on the case where there is no noise, in which case the goal is to reconstruct the sparse vector x (possibly up to some identifiability considerations). When x is not required to be sparse, this problem is simply the standard "fixed design regression" setup for Ordinary Least Squares, whose study dates back at least to Gauss. The motivation for considering the case where x is sparse is that it allows for reconstruction even in the "underdetermined" situation where the number of observations m is small compared to the number of parameters n. A particularly important special case of this problem is the *compressed sensing* problem, where the matrix A corresponds to some kind of physical measurement device, e.g. an MRI machine, in which case the goal is to minimize the total number of measurements (i.e. time the machine

is on) while still getting an accurate scan of the patient. See e.g. [6, 12, 15] for an overview of compressed sensing and its mathematical analysis.

In the case of compressed sensing, the measurement matrix A is partially under the control of the designer who builds the machine. Therefore, A can be chosen to have particularly nice mathematical properties which allow for the problem to be solved computationally quickly with essentially optimal statistical guarantees. In this paper, we focus on the general setting where the measurement matrix A is arbitrary. The motivation for this is that in many other settings of interest, we are not in control of the design matrix A: for example, in a typical application of regression in data science, each row of A may be a list of features corresponding to an individual person, and the entries of y correspond to a property of this person we are attempting to predict with a sparse x. In this case, the columns of A could be highly correlated (if two features of a person are highly correlated, e.g. age and education level) and this makes the situation quite different from compressed sensing, where we usually choose the columns of A to be incoherent (i.e. at almost right angles to each other).

Because of the great interest of this problem to different applications, the NP-hardness of this problem when A is general was established in the early work of Natarajan [13]. Some extensions of this work can be found, for example, in the works of Zhang et al [19, 20] where they studied how this hardness result relates to the guarantees for the famous LASSO algorithm (based on ℓ_1-constrained minimization) and also constructed counterexamples to a wide array of approaches based on regularized least squares, and another line of related work (see e.g. [1, 16]) studied the closely related problem of certifying the Restricted Invertibility Property (RIP) used in compressed sensing. On the other hand, in recent years there has also been a lot of interest in algorithms for solving this NP-hard problem and success solving instances in practice using integer programming techniques (see e.g. [2]), especially for small values of k. The existing hardness results establish that this problem is NP-hard but leave open the runtime of the fastest algorithms for this problem — in particular, the result of [13] leaves open the possibility that there exist algorithms which run in time $poly(2^k, n)$ so that the problem is *Fixed-Parameter Tractable* — informally, quickly solvable for all small values of k.

This problem has two essentially equivalent formulations: in the random design formulation, which is more learning-theoretic, we have that $Y = \langle X, w \rangle$, w is a k-sparse vector, $X \sim N(0, \Sigma)$ and the goal of the algorithm is to exactly reconstruct w from m samples $(X_i, Y_i)_{i=1}^m$. In the other, so-called fixed design formulation, we have that $y = Ax$ where x is k-sparse and the algorithm seeks to recover x given A, y. The equivalence of these two formulations is slightly nontrivial and is described in Section 5: in one direction it is given by taking the rows of A to be the samples $X_1, \ldots, X_m$ and taking $x = w$. In the Gaussian case, if we take $\Sigma = I$ then this is a well-studied setting in compressed sensing and it is known that we can solve this problem with nearly the optimal statistical efficiency (i.e. up to constant factors in the sample complexity) in polynomial time for a general range of values of k. However, as we show in this work the

problem becomes quite different when we allow an arbitrary covariance matrix Σ.

1.1 Our Results

We find upper and lower bounds for the fine-grained complexity of k-sparse linear regression over the reals, as well as over finite fields. Here, a k-sparse solution to a linear regression problem is defined to be a $n \times 1$ solution vector with k non-zero entries. It's possible to reduce to the case $m = O(k \log n)$ using a dimension reduction argument (see Section 5) so we focus on the runtime in terms of parameters n and k.

The k-Sparse Linear Regression problem is as follows:

Definition 1. k-Sparse Linear Regression (k-SLR) The k-SLR problem is to determine if there exists a k-sparse solution x to the equation $Ax = y$ where A is an $m \times n$ matrix. We write $k\text{-SLR}_{\mathbb{F}}$ to specify that we are considering the problem over the field $\mathbb{F}$.

We formulate a special case of k-SLR in terms of the Kruskal rank of a matrix, the maximum number r such that every set of r columns is linearly independent.

Definition 2. k-Kruskal Rank Estimation (k-KRE) Given a $m \times n$ matrix A, the k-KRE problem is to determine if A has Kruskal rank at most $k-1$. Equivalently, the problem is to determine if there exists a k-sparse solution $x \neq 0$ to $Ax = 0$. We write $k\text{-KRE}_{\mathbb{F}}$ to specify that we are considering the problem over the field $\mathbb{F}$.

For $k \geq 2$, we find that k-SLR and k-KRE over the reals require $n^{\Omega(k)}$ time under the Exponential Time Hypothesis, and require $n^{\Omega(\lceil k/2 \rceil)}$ under the k-sum Conjecture. We also show an upper bound of $O(n^{k-1} \log n)$ for both problems. These bounds hold whether we are promised that $y = Ax$ has a unique solution or not.

We also adapt these results from the reals to finite fields. We modify a reduction from Independent Set to Perfect Code to find that k-SLR requires $n^{\Omega(\sqrt{k})}$ for small finite fields under ETH. We also find $O(n^{\lceil k/2 \rceil} \cdot q^{\lceil k/2 \rceil})$ upper bounds for k-KRE and k-SLR over finite fields $\mathbb{F}_q$ where q is a prime power.

2 Lower Bounds

In this section, we show that k-KRE and k-SLR each require $n^{\Omega(k)}$ under the Exponential Time Hypothesis, and $\Omega(n^{\lceil k/2 \rceil})$ under the k-sum Conjecture. We also show that the unique versions of these problems, in which we are promised a unique k-sparse solution, are as hard as the general versions.

2.1 Preliminaries

We first present the known results that we later use to obtain lower bounds for our problem. For completeness, we provide proofs because they are short, and to show that they preserve uniqueness.

In theoretical computer science, the k-SAT and subset sum problems are NP hard.

Definition 3 (k-SAT)**.** The k-SAT problem is to determine whether a k-CNF Boolean formula is satisfiable.

Definition 4 (k-SUM)**.** Given a set of n numbers, the k-SUM problem is to determine whether there exists a subset of k numbers that sum to zero.

The subset sum problem is equivalent to k-SUM for arbitrary k. Given a set of n numbers, the subset sum problem is to determine whether there exists any subset of those numbers that sum to zero.

Throughout this paper, we use two computational hardness assumptions based on k-SAT: the Exponential Time Hypothesis and the Strong Exponential Time Hypothesis.

Definition 5 (ETH)**.** The Exponential Time Hypothesis states that 3-SAT cannot be solved in subexponential time in the worst case. [8]

Definition 6 (SETH)**.** The Strong Exponential Time Hypothesis states that for every ϵ there is a k such that k-SAT cannot be solved in $O(2^{(1-\epsilon)n})$. [4]

2.1.1 Khachiyan's subset sum reduction

[11] gives a reduction from k-subset sum to k-KRE. For context, the connection of the work of [11] with NP hardness of Kruskal rank and SLR has been observed in [12] and for the closely related problem of RIP certification, previously in [1].

Theorem 1. *[11] Given any instance $\mathcal{A}$ of k-subset sum on n real numbers, there exists an k-KRE problem such that $\mathcal{A}$ has an k-subset sum iff the Kruskal rank of a set of k vectors is at most $k-1$.*

Proof. In the following proof, we use the weird moment curve for the real numbers α_i:

$$\Gamma'(\alpha_i) = [1, \alpha_i, \alpha_i^2, ..., \alpha_i^{k-2}, \alpha_i^k].$$

Given a subset I of the distinct real numbers α_1, α_2, ..., α_n, we have that $\sum_{i\in I} \alpha_i = 0$ iff the vectors $\{\Gamma'(\alpha_i)\}_{i\in I}$ are linearly dependent. This is because the determinant of the matrix with columns $\{\Gamma'(\alpha_i)\}_{i\in I}$ is

$$\left(\sum_{i\in I} \alpha_i\right) \prod_{i,j\in I, i<j} (\alpha_i - \alpha_j),$$

which is zero if and only if $\sum_{i\in I} \alpha_i = 0$.

If there is a subset I such that $|I| = k$ and $\sum_{i \in I} \alpha_i = 0$, the vectors $\{\Gamma'(\alpha_i)\}_{i \in I}$ are linearly dependent, so the Kruskal rank of $\{\Gamma'(\alpha_i)\}_{i \in I}$ is at most $k-1$. Otherwise, the Kruskal rank of $\{\Gamma'(\alpha_i)\}_{i \in I}$ is k. Thus, deciding whether the Kruskal rank is k or at most $k-1$ is NP-hard. □

Corollary 1. *If k-KRE is in time $O(n^{f(k)})$, then k-SUM is in $O(n^{f(k)})$.*

2.1.2 Known results from fine-grained complexity

We present a known lower bound for k-sum from [14].

Theorem 2. *[14] ETH implies k-sum requires $n^{\Omega(k)}$ time.*

Proof. Let F be the 3-CNF instance of 3-SAT, with n variables and m clauses. By the Sparsification Lemma [9], we can assume that $m = O(n)$. We first reduce to an instance F' of 1-IN-3-SAT (in which exactly one literal in each clause needs to be true for the formula to be satisfied). For each clause $(x \vee y \vee z)$ in F, construct the clauses $(x \vee a \vee d) \wedge (y \vee b \vee d) \wedge (a \vee b \vee e) \wedge (c \vee d \vee f) \wedge (z \vee c)$ in F'. It can be checked that this reduction preserves uniqueness.

F' has 6 new variables per clause and 5 clauses per clause, so F' has $O(m)$ variables and clauses. We partition the variables into k groups of size n/k size: $V_1, \ldots, V_k$ and consider all possible $2^{n/k}$ partial assignments for each group. For each partial assignment ϕ for each group, we assign a number in base $(k+1)$.

Each number has a group section of k digits and a clause section of $5m$ digits. In the group section for a partial assignment ϕ of group V_i, the i^{th} digit is 1 and all other digits are 0. The target t has 1 for each digit corresponding to a group. This forces the solution to the k-SUM problem to pick one partial assignment from each group.

In the clause section, the i^{th} digit records the number of literals that ϕ sets to true. If there are more than one true literals in any clause, omit ϕ. The target t has 1 for each digit corresponding to a clause. This forces the assignment to satisfy exactly one variable in each clause. This reduction also preserves uniqueness of solution.

By this reduction, we get that if k-sum on $N = k \cdot 2^{n/k}$ numbers of $O(k \log k \log N)$ bits can be solved in $O(N^{\delta k}) = O(2^{\delta n} \cdot k^{O(k)})$ time for all $\delta > 0$, then ETH is false. □

Next, we present a known upper bound for k-SUM from [18].

Lemma 1. *2-sum on n numbers is in $O(n)$ (randomized) time.*

Proof. Given a list $L = \{a_i\}$, we create a new list $L' = \{-a_i \mid a_i \; in L\}$ and check if $L \cap L'$ is empty or not. This can be done in linear time with the help of a hash table. □

Lemma 2. *[18] k-sum on n numbers (on k parts) reduces to 2-sum on 2 parts with $n^{\lceil k/2 \rceil}$ numbers each.*

Proof. We enumerate the sums of all possible triples from the first $\lfloor k/2 \rfloor$ parts and from the last $\lceil k/2 \rceil$ parts to obtain two lists L, L'. We now want to check for every $l_i \in L$, if $-l_i \in L'$. From above, this can be done in linear time in the size of the lists. □

Corollary 2. *k-sum is in $O(n^{\lceil k/2 \rceil})$ time.*

Conjecture 1 (k-Sum Conjecture)**.** It is conjectured that k-sum cannot be solved in $O(n^{\lceil k/2 \rceil - \epsilon})$ time for any $k \geq 2$ and $\epsilon > 0$.

2.2 New Results

We combine Khachiyan's subset sum reduction with the known reduction from 3-SAT to k-SUM to find a new lower bound for k-KRE.

Theorem 3. *ETH implies that k-KRE requires $n^{\Omega(k)}$.*

Proof. Combining Corollary 1 and Theorem 2, ETH implies that k-KRE requires time $n^{\Omega(f(k)}$ where $f(k) \geq k$. □

We find another lower bound for k-KRE under the k-sum conjecture.

Theorem 4. *The k-SUM conjecture implies that k-KRE requires $\Omega(n^{\lceil k/2 \rceil})$.*

Proof. Corollary 1 implies that under the k-SUM conjecture, k-SLR requires time $\Omega(n^{f(k)})$ where $f(k) \geq \lceil k/2 \rceil$. □

Next, we show that k-SLR is as hard as k-KRE. [12] gives a reduction from k-KRE to $(k-1)$-SLR, but we find a more efficient reduction that preserves k.

Lemma 3. *There exists a randomized reduction from k-KRE to k-SLR.*

Proof. Given an instance of k-KRE in the form $0 = Ax$, we construct an instance of k-SLR. We add a row of random numbers to A to get the $(m+1) \times n$ matrix B. Then $0 \neq Bx$ with probability 1. If we rescale B and set y equal to a vector of all zeros with 1 for the last entry, then $y = Bx$ with probability 1. □

Therefore, the same lower bounds hold for k-KRE and k-SLR.

Theorem 5. *ETH implies that k-SLR requires $n^{\Omega(k)}$. The k-SUM conjecture implies that k-SLR requires $\Omega(n^{\lceil k/2 \rceil})$.*

2.2.1 Uniqueness of solution

Finally, we show that the reductions from k-SAT to the k-SLR problem preserve uniqueness. That is, k-SLR is hard even when we are guaranteed a unique solution.

Definition 7 (Unique k-SLR)**.** Given a k-SLR problem with the guarantee that $y = Ax$ has at most one k-sparse solution x, the Unique k-SLR problem is to determine whether there exists a k-sparse solution.

We define the Unique k-SAT and Unique k-SUM problems as follows.

Definition 8 (Unique k-SAT)**.** Given a k-SAT problem that is known to have either 0 or 1 satisfying assignments, the Unique k-SAT problem is to determine the number of satisfying assignments.

Definition 9 (Unique k-SUM)**.** Given a set S of n integers, the Unique k-SUM problem is to determine whether a unique subset of k integers sum to 0.

[3] finds that if ETH holds for general 3-SAT, then it holds for Unique 3-SAT.

Theorem 6. *[3] ETH implies that Unique 3-SAT cannot be solved in $O(2^{\delta n})$ for every $\delta > 0$.*

The reduction from Theorem 2 preserves uniqueness, so we can apply it to the Unique k-SUM case. Theorem 2 allows us to reduce from Unique 3-SAT to Unique 1-IN-3-SAT, then to Unique k-SUM, to get the following result.

Theorem 7. *ETH implies Unique k-sum requires $n^{\Omega(k)}$ time.*

These results, combined with Khachiyan's subset sum reduction, imply that Unique k-SLR is as hard as general k-SLR, under ETH.

3 Upper Bound

In this section, we present algorithms to solve the k-SLR$_{\mathbb{R}}$ and k-KRE$_{\mathbb{R}}$ problems in $O(n^{k-1} \log n)$ time, beating the trivial brute force approach that takes $O(n^k)$ time (Theorems 9, 10). These algorithms build off of the algorithms we present for the 2-sparse case. When $k = 2, 3$, these algorithms are optimal and meet the lower bound of $\Omega(n^{\lceil k/2 \rceil})$ presented in Section 2, up to a $\log n$ factor. We then provide some evidence that the lower bound is not tight for general k, through a reduction to the INTERSECTING SUBSPACES problem, for which we prove a SETH-based lower bound.

The naive brute force algorithm is to search over all possible $O(n^k)$ supports of size k, consider only the corresponding columns of A and solve the resulting linear system. This would take $O(n^k)$ time. As stated before, the key insight for improving this algorithm occurs in the case $k = 2$ where one-dimensional subspaces play an important role; the following definition is key.

Definition 10. For each non zero vector $u \in \mathbb{R}$, we can choose a unique representative element rep$[u]$ of the one-dimensional subspace $U = \{au \mid a \in \mathbb{R}\}$ spanned by u as

$$\text{rep}[u] = \frac{1}{u_{i_1}} \cdot u$$

where u_{i_1} is the first non zero entry in the vector u.

In the following algorithm and for the rest of this section, we use the following notation. If A is a $m \times n$ matrix, $S = \{i_1, \ldots, i_k\} \subseteq n$ and x is a n-dimensional vector, we define A_S, x_S as follows:

$$A_S = [A_{i_1}, A_{i_2}, \ldots, A_{i_k}]$$

$$x_S = (x_{i_1}, x_{i_2}, \ldots, x_{i_k})$$

To ensure this notation is unique, we require that $i_j < i_k$ whenever $j < k$. We define $x_{\overline{S}}$ as the vector of entries of x not in x_S.

Algorithm 1: SOLVE-2-KRE$_{\mathbb{R}}$

input : $A \in \mathbb{R}^{m \times n}$
output: 2-sparse non zero solution x to $Ax = 0$

for $i \leftarrow 1$ **to** n **do**
  $B[i] \leftarrow \text{rep}[A_i]$;
Sort columns in B in lexicographic order. Let σ be s.t. $B[i] = \text{rep}[A_{\sigma(i)}]$;
for $i \leftarrow 2$ **to** n **do**
  if $B[i] = B[i-1]$ **then**
    $S \leftarrow \{\sigma(i), \sigma(i-1)\}$;
    Solve for $x_S : A_S x_S = 0$;
    $x_{\overline{S}} \leftarrow 0$;
    return x
return *No Solution*

Lemma 4. SOLVE-2-KRE$_{\mathbb{R}}$ *solves* 2-*KRE*$_{\mathbb{R}}$ *in time* $O(mn \log n)$.

Proof. A solution to 2-KRE$_{\mathbb{R}}$ for $A \in \mathbb{R}^{m \times n}$ exists if and only if A has two linearly dependent column vectors. Two vectors $u, v \in \mathbb{R}$ are linearly dependent if and only if $\text{rep}[u] = \text{rep}[v]$. If two column vectors A_i, A_j are linearly dependent, then $\text{rep}[A_i], \text{rep}[A_j]$ will appear consecutively after sorting. □

Definition 11. Given a $m \times n$ matrix A, a set $M \subseteq [n]$ is called a maximal pairwise independent support if

1. for all $i, j \in M$, A_i, A_j are linearly independent, and
2. for all $i \in [n] \setminus M$, there exists $j \in M$ such that A_i, A_j are linearly dependent.

Algorithm 2: FIND-MPIS

input : $A \in \mathbb{R}^{m \times n}$
output: The maximal pairwise independent support M of A

for $i \leftarrow 1$ **to** n **do**
  $B[i] \leftarrow \text{rep}[A_i]$;
Sort columns in B in lexicographic order. Let σ be st.
 $B[i] = \text{rep}[A_{\sigma(i)}]$;
Initialize $M \leftarrow \{\sigma(1)\}$;
for $i \leftarrow 2$ **to** n **do**
  if $B[i] \neq B[i-1]$ **then**
    $M \leftarrow M \cup \{\sigma(i)\}$
return M

Lemma 5. FIND-MPIS *finds the maximal pairwise independent support of a matrix* $A \in \mathbb{R}^{m \times n}$ *in time* $O(mn \log n)$.

Proof. Two vectors $u, v \in \mathbb{R}$ are linearly dependent if and only if $\text{rep}[u] = \text{rep}[v]$. If two column vectors A_i, A_j are linearly dependent, then $\text{rep}[A_i], \text{rep}[A_j]$ will appear consecutively after sorting. □

Algorithm 3: SOLVE-2-SLR$_{\mathbb{R}}$

input : $A \in \mathbb{R}^{m \times n}, y \in \mathbb{R}^m$
output: 2-sparse solution x to $Ax = y$

Check that no 1-sparse solution exists;
$M \leftarrow$ FIND-MPIS(A);
Solve for 2-sparse $x'_M : P_{y^\perp} A_M x'_M = 0$ using SOLVE-2-KRE$_{\mathbb{R}}$;
$x'_{\overline{M}} \leftarrow 0$;
if *solution* x' *exists* **then**
  $V \leftarrow \text{supp}(x')$;
  Solve for $x_V : A_V x_V = y$;
  $x_{\overline{V}} \leftarrow 0$;
  return x
return *No Solution*

Theorem 8. SOLVE-2-SLR$_{\mathbb{R}}$ *solves* 2-*SLR*$_{\mathbb{R}}$ *in* $O(mn \log n)$ *time.*

Proof. Assume that no 1-sparse solution exists. Let $P_{y^\perp}$ be the projection matrix onto the orthogonal complement of y.

Claim 1. Then a solution x with support $\{i, j\}$ to 2-SLR$_{\mathbb{R}}$ exists if and only if A_i, A_j are linearly independent and $P_{y^\perp} A_i, P_{y^\perp} A_j$ are linearly dependent.

Proof. Suppose $x_i A_i + x_j A_j = y$. Since there is no 1-sparse solution, A_i, A_j have to be linearly independent. Further,

$$P_{y^\perp} x_i A_i + P_{y^\perp} x_j A_j = P_{y^\perp} y = 0$$

Thus $P_{y^\perp} A_i, P_{y^\perp} A_j$ are linearly dependent.

Conversely, suppose that A_i, A_j are linearly independent and $P_{y^\perp} A_i, P_{y^\perp} A_j$ are linearly dependent. Writing A_i in terms of its components, $A_i = a_i y + P_{y^\perp} A_i$, we get that there exist $\tilde{x_i}, \tilde{x_j}$

$$\tilde{x_i}(A_i - a_i y) + \tilde{x_j}(A_j - a_j y) = 0$$

Since A_i, A_j are linearly independent, we get that

$$\tilde{x_i} A_i + \tilde{x_j} A_j = (\tilde{x_i} a_i + \tilde{x_j} a_j) y \neq 0$$

Therefore a 2-sparse solution to $Ax = y$ with support $\{i, j\}$ exists. □

Computing the maximal pairwise independent support M of A ensures that we consider only pairwise independent columns. By Claim 1 the support of solution x' to $P_{y^\perp} A x' = 0$ is also the support to the solution of $Ax = y$ if it exists. □

Algorithm 4: SOLVE-k-SLR$_{\mathbb{R}}$

input : $A \in \mathbb{R}^{m \times n}, y \in \mathbb{R}^m, k \in \mathbb{N}$
output: k-sparse solution x such that $Ax = y$

foreach $T \subseteq [n]$ *st.* $|T| = k - 2, \ker(A_T) = \{0\}$ **do**
  U $\leftarrow \text{span}(A_T)$;
  Solve for 2-sparse $x'_{\overline{T}} : P_{U^\perp} A_{\overline{T}} x'_{\overline{T}} = P_{U^\perp} y$ using SOLVE-2-SLR$_{\mathbb{R}}$;
  $x'_T \leftarrow 0$;
  if *solution* x' *exists* **then**
    Solve $x_S : A_S x_S = y$ where $S = U \cup \text{supp}(x')$;
    $x_{\overline{S}} \leftarrow 0$;
    return x
return *No Solution*

Lemma 6. *Suppose x is a k-sparse vector and $T \subseteq \text{supp}(x)$, $|T| = k - 2$. Let $U = \text{span}(A_T)$. Then x solves $Ax = y$ if and only if*

$$P_{U^\perp} A_{\overline{U}} x'_{\overline{U}} = P_{U^\perp} y$$

has a 2-sparse solution, where $x'_U = 0$.

Proof. Suppose x is a k-sparse solution with support $\{i_1, \ldots, i_k\}$ and suppose $T = \{i_1, \ldots, i_{k-2}\}$. Then

$$\sum_{j=1}^{k} x_{i_j} A_{i_j} = y$$

Since $U = \text{span}(A_T)$ by definition, $P_{U^\perp} A_i = 0$ for $i \in T$. Therefore

$$\sum_{j=1}^{k} P_{U^\perp} x_{i_j} A_{i_j} = x_{i_{k-1}} P_{U^\perp} A_{i_{k-1}} + x_{i_k} P_{U^\perp} A_{i_k} = P_{U^\perp} y$$

Therefore $P_{U^\perp} A_{\overline{U}} x'_{\overline{U}} = P_{U^\perp} y$ has a 2-sparse solution with support $\{i_{k-1}, i_k\}$.

Now we show the converse. Suppose that $P_{U^\perp} A_{\overline{U}} x'_{\overline{U}} = P_{U^\perp} y$ has a 2-sparse solution with support $\{i_{k-1}, j_k\}$.

$$x_{i_{k-1}} P_{U^\perp} A_{i_{k-1}} + x_{i_k} P_{U^\perp} A_{i_k} = P_{U^\perp} y \tag{1}$$

Since $y_U - x_{i_{k-1}}(A_{i_{k-1}})_U - x_{i_k}(A_{i_k})_U \in U$, there exist real numbers x_{i_j}, for $j = 1, \ldots, k-2$ such that

$$\sum_{j=1}^{k-2} x_{i_j} A_{i_j} = y_U - x_{i_{k-1}}(A_{i_{k-1}})_U - x_{i_k}(A_{i_k})_U \tag{2}$$

Adding equations 1 and 2, we get a k-sparse solution to $Ax = y$.

$$\sum_{j=1}^{k} x_{i_j} A_{i_j} = y$$

□

Theorem 9. SOLVE-k-SLR$_\mathbb{R}$ *solves* k*-SLR*$_\mathbb{R}$ *in* $O(mn^{k-1} \log n)$ *time, for* $k \geq 2$.

Proof. SOLVE-k-SLR$_\mathbb{R}$ loops over all the $O(n^{k-2})$ subsets $T \subseteq [n]$ of size $k-2$ and checks if $P_{U^\perp} A_{\overline{T}} x'_{\overline{T}} = P_{U^\perp} y$ has a 2-sparse solution. Computing an orthogonal basis for each subspace U using the Gram-Schmidt process takes $O(mk^2)$ time. Projecting the remaining $O(n)$ column vectors onto $U^\perp$ takes $O(mkn)$ time. By Lemma 4 and Theorem 8, finding a 2-sparse solution can be done in $O(mn \log n)$ time. The correctness of this algorithm follows from Lemma 6. □

Theorem 10. k*-KRE*$_\mathbb{R}$ *can be solved in* $O(n^{k-1} \log n)$ *time, for* $k \geq 2$.

Proof. $Ax = 0$ has a k-sparse solution if and only if $A_{[n]\setminus\{i\}} x'_{[n]\setminus\{i\}} = A_i$ has a $(k-1)$-sparse solution for some $i \in \text{supp}(x)$. We use brute force over all $i \in [n]$ entries of x to find an index in the support of x. For each index i, the problem then reduces to the $(k-1)$-SLR problem, which is solvable in $O(mn^{k-2} \log n)$ time. Thus, $0 = Ax$ with k-sparse x can be solved in $O(mn^{k-1} \log n)$ time. □

Remark 1. Given n vectors in $\mathbb{Q}^m$ we can check if there are two linearly dependent vectors in $O(n)$ time. We can do this by re-scaling the vectors to make the first non-zero entry 1 and use hashing to search for duplicates.

Algorithm 5: SOLVE-k-KRE$_{\mathbb{R}}$

input : $A \in \mathbb{R}^{m \times n}, k \in \mathbb{N}$
output: k-sparse solution x such that $Ax = 0$

foreach $i \in [n]$ **do**
  Solve for $(k-1)$-sparse $x'_{[n]\setminus\{i\}} : A_{[n]\setminus\{i\}} x'_{[n]\setminus\{i\}} = A_i$ using SOLVE-$(k-1)$-SLR$_{\mathbb{R}}$;
  $x'_i \leftarrow 0$;
  if *solution x' exists* **then**
    $S \leftarrow \{i\} \cup \text{supp}(x')$;
    Solve for $x_S : A_S x_S = 0$;
    $x_{\overline{S}} \leftarrow 0$;
    return x
return *No Solution*

3.1 Connections to Intersecting Subspaces

Is there a natural way to improve the algorithmic upper bound to $n^{\lceil k/2 \rceil}$, matching the lower bound? Motivated by this question, we define the Intersecting Subspaces problem. A natural way to come up with a faster algorithm to solve k-KRE would be to reduce to the Intersecting Subspaces problem, taking inspiration from the optimal algorithm for k-Sum [18]. The randomized reduction we present from k-KRE creates an Intersecting Subspaces instance with $O(n^{\lceil k/2 \rceil})$ subspaces. If Intersecting Subspaces can be solved in $O(n^c)$ time where $c < 2$, then this reduction would give us a $O(n^{c'k})$ time algorithm for even k and a $O(n^{c'(k+1)})$ time algorithm for k-KRE$_{\mathbb{R}}$ for odd k, where $c' < 1$. We show that for large dimensions, Intersecting Subspaces cannot be solved in subquadratic time if SETH is true.

Definition 12. Intersecting Subspaces Suppose we are given two sets S_1, S_2 each of size n. Suppose S_1 contains k_1-dimensional subspaces of $\mathbb{R}^d$ and S_2 contains k_2-dimensional subspaces of $\mathbb{R}^d$. The problem is to determine if there exist two subspaces $U \in S_1, V \in S_2$ such that they have a non-trivial intersection.

We note that this problem can be solved over the reals in $O(n \log n)$ time when $k_1 = k_2 = 1$, with an algorithm similar to 2-KRE$_{\mathbb{R}}$. However, when either k_1 or k_2 is larger than 1, we can prove a SETH-based quadratic lower bound.

Lemma 7. *If* Intersecting Subspaces *on $O(n)$ subspaces can be solved in time $f(n)$, then k-KRE can be solved in time $f(n^{\lceil k/2 \rceil})$.*

Proof. We randomly assign the column vectors of A to one of two buckets A_1, A_2, with equal probability. If there exists a k-sparse x such that $Ax = 0$, then the probability that $\lfloor k/2 \rfloor$ of the vectors in the support of x are assigned to A_1 and $\lceil k/2 \rceil$ of them are assigned to A_2 is $1/2^k$. After $O(1)$ repeated trials, we can assume that this is true.

We then construct set S_1 of all $O(n^{\lfloor k/2 \rfloor})$ subspaces spanned by sets of $\lfloor k/2 \rfloor$ vectors from A_1. These subspaces have dimension at most $\lfloor k/2 \rfloor$. Similarly we construct set S_2 of all $O(n^{\lceil k/2 \rceil})$ subspaces spanned by sets of $\lceil k/2 \rceil$ vectors from A_1. These subspaces have dimension at most $\lceil k/2 \rceil$. There exists a k-sparse solution to $Ax = 0$ if and only if there exist subspaces $U \in S_1, V \in S_2$ with non-trivial intersection. □

Definition 13 (Orthogonal Vectors)**.** Suppose we are given a set S containing n vectors. The problem is to determine if there exist two vectors $u, v \in S$ such that $\langle u, v \rangle = 0$.

Lemma 8. *[Lemma A.1 of [17]] Suppose there is a $\delta > 0$ and an algorithm $\mathcal{A}$ such that for all $c \geq 1$, $\mathcal{A}$ solves* Orthogonal Vectors *on n vectors in $\{0,1\}^{c \log n}$ over the integers, in $O(n^{2-\delta})$ time. Then the Strong Exponential Time Hypothesis is false.*

Corollary 3. *Suppose* Intersecting Subspaces *over $\mathbb{R}^d$ with $2 < k_1 + k_2 \leq d, d = \Omega(\log n)$ can be solved in $O(n^{2-\epsilon})$ time for some $\epsilon > 0$. Then the Strong Exponential Time Hypothesis is false.*

Proof. We provide a padding argument that reduces from Intersecting Subspaces with k_1, k_2, d to Intersecting Subspaces with $k_1' \geq k_1, k_2' \geq k_2, d' \geq \max\{k_1' + k_2', d\}$. We add $d' - d$ more coordinates so the ambient space is $\mathbb{R}^{d'}$.

We then replace each subspace $U \in S_1$ with the subspace spanned by $\text{basis}(U) \cup \left\{e_{d+1}, \ldots, e_{d+k_1'-k_1}\right\}$. We replace each subspaces $V \in S_2$ with the subspaces spanned by $\text{basis}(V) \cup \left\{e_{d+k_1'-k_1+1}, \ldots, e_{d+k_1'-k_1+k_2'-k_2}\right\}$. □

Lemma 9. *Suppose* Intersecting Subspaces *over $\mathbb{R}^d$ with $2 < k_1 + k_2 \leq d, d = \Omega(\log n)$ can be solved in $O(n^{2-\epsilon})$ time for some $\epsilon > 0$. Then the Strong Exponential Time Hypothesis is false.*

Proof. We provide a reduction from Orthogonal Vectors to Intersecting Subspaces. We are given a set S of n vectors in R^d as the Orthogonal Vectors instance. Let $S_1 = S$ and $S_2 = \left\{u^\perp \mid u \in S\right\}$ where $u^\perp$ is the orthogonal complement of vector u.

Then there exist two orthogonal vectors $u, v \in S$ if and only if there is a non-trivial intersection between $u \in S_1$ and $v^\perp \in S_2$. By Lemma 8 and Lemma 3, SETH implies that Intersecting Subspaces over $\mathbb{R}^d$ with $2 < k_1 + k_2$ cannot be solved in $O(n^{2-\epsilon})$ time for any $\epsilon > 0$. □

4 Finite Fields

We consider the k-KRE$_\mathbb{F}$ and k-SLR$_\mathbb{F}$ problems where the finite field $\mathbb{F} = \mathbb{F}_q$ for a prime power $q = p^\ell$. We provide an algorithm for Intersecting Subspaces over $\mathbb{F}_q$ that takes $O(n \cdot q^{\max\{k_1, k_2\}})$ time. This gives us algorithms for k-SLR$_\mathbb{F}$ and k-KRE$_\mathbb{F}$ that run in time $O(n^{\lceil k/2 \rceil} \cdot q^{\lceil k/2 \rceil})$ time. This is a significant speed up when $q = o(n)$. For finite fields with large characteristic, however,

the optimal algorithm would be a version similar to that presented in Section 3. Theorem 1.7 of [17] (presented here as Theorem 11) gives us a lower bound on Intersecting Subspaces over finite fields. In Subsection 4.3 we provide a lower bound of $n^{\Omega(\sqrt{k})}$ to the k-SLR$_{\mathbb{F}}$ problem.

4.1 Intersecting Subspaces

Definition 14. For each non zero vector $u \in \mathbb{F}^m$, we can choose a unique representative element rep$[u]$ to be the first vector in the one-dimensional subspace $U = \{au \mid a \in \mathbb{F}\}$ spanned by u, when all the vectors are sorted in lexicographic order.

Lemma 10. *2-KRE$_{\mathbb{F}}$ can be solved in time $O(n)$.*

Proof. For each column A_i of A, we compute rep$[A_i]$. By hashing, we can determine if there exist $i, j \in [n]$ such that rep$[A_i] =$ rep$[A_j]$.

The correctness of the algorithm follows from the fact that two vectors $u, v \in \mathbb{F}^m$ are linearly dependent if and only if rep$[u] =$ rep$[v]$. □

Lemma 11. Intersecting subspaces *with $|S_1| = |S_2| = n$ in $\mathbb{F}^m$ can be solved in time $O(n \cdot q^{\max\{k_1,k_2\}})$.*

Proof. We enumerate all the $n \cdot q^{k_1}$ vectors in S_1 to get list L_1, and all the $n \cdot q^{k_2}$ vectors in S_2 to get list L_2. There exist subspaces $U \in S_1, V \in S_2$ if and only iff there are non zero $u \in L_1, v \in L_2$ that are linearly dependent. This can be checked by a method very similar to that in Lemma 10 in time linear in the size of the lists L_1, L_2. Thus Intersecting Subspaces$_{\mathbb{F}}$ can be solved in time $O(n \cdot q^{\max\{k_1,k_2\}})$. □

The following theorem by [17] when combined with the reduction in Lemma 9 gives us a lower bound on the runtime of Intersecting Subspaces over finite fields (Corollary 4).

Theorem 11. *[Theorem 1.7 of [17]] Suppose there is an $\epsilon > 0$ and a function $f : \mathbb{N} \to \mathbb{N}$ such that $f(x)/(x/\log x) \to 0$ and for infinitely many $\mathbb{F} = \mathbb{F}_q$,* Orthogonal Vectors *is solvable in time $n^{2-\epsilon} \cdot d^{f((p-1)\ell)}$. Then SETH is false.*

Corollary 4. *Suppose there is an $\epsilon > 0$ and a function $f : \mathbb{N} \to \mathbb{N}$ such that $f(x)/(x/\log x) \to 0$ and for infinitely many $\mathbb{F} = \mathbb{F}_q$,* Intersecting Subspaces *is solvable in time $n^{2-\epsilon} \cdot \max\{k_1, k_2\}^{f((p-1)\ell)}$. Then SETH is false.*

4.2 Upper Bound

By reducing to the problem of Intersecting Subspaces, we obtain faster algorithms for k-KRE and k-SLR over the finite fields.

Theorem 12. *k-KRE$_{\mathbb{F}}$ can be solved in time $O(n^{\lceil k/2 \rceil} q^{\lceil k/2 \rceil})$.*

Proof. By combining the reduction in Lemma 7 and the algorithm described in Lemma 11, we get a $O(n^{\lceil k/2 \rceil} q^{\lceil k/2 \rceil})$ time algorithm. □

Corollary 5. *k-SLR$_{\mathbb{F}}$ can be solved in time $O(n^{\lceil k/2 \rceil} q^{\lceil k/2 \rceil})$.*

Proof. The algorithm for k-SLR$_{\mathbb{F}}$ is similar to that in Theorem 12. We first randomly divide the columns of A into two buckets A_1, A_2 as in Lemma 7. Let S_1 be the collection of all the subspaces spanned by $\lfloor k/2 \rfloor$-sized subsets from A_1 and y. Let S_2 be the collection of all subspaces spanned by $\lceil k/2 \rceil$-sized subspaces from A_2. We then enumerate all the vectors spanned by these subspaces to obtain two lists L_1, L_2 and check for linear dependencies. We consider only those vectors in L_1 which have nonzero coefficient of y when expressed as a linear combination of the columns in A_1 and y. □

4.3 Lower Bound for k-SLR

The lower bound derived for k-SLR in Section 2 does not hold for small finite fields, since the k-Sum conjecture does not hold over small finite fields.

We observe that the Perfect Code problem is very similar to the problem of sparse linear regression when the solution vector x is constrained to be in $\{0,1\}^n$. We adapt a reduction from Independent Set to Perfect Code given by [7] (Theorem 4.1) to make it more robust so that we can reduce to k-SLR.

Definition 15 (k-Independent Set)**.** An independent set in a graph $G = (V, E)$ is a set of vertices $V' \subset V$ such that no two vertices in V' are adjacent. The k-Independent Set problem is to determine whether there exists an independent set of k vertices in G.

Definition 16. The neighborhood $N(v)$ of a vertex v in $G = (V, E)$ is the set of vertices adjacent to v. The closed neighborhood $N[v]$ is $\{v\} \cup N(v)$.

Definition 17 (k-Perfect Code)**.** A perfect code in a graph $G = (V, E)$ is an independent set of vertices $V' \subset V$ such that for every vertex $v \in V$, there is exactly one vertex in $N[v] \cup V'$. The k-Perfect Code problem is to determine whether there exists a perfect code of k vertices in G.

Theorem 13. *Suppose k-SLR over finite fields can be solved in $n^{o(\sqrt{k})}$ time, then we can solve k-Independent Set in $n^{o(k)}$ time.*

Proof. We first reduce from an arbitrary instance $G = (V, E)$ of k-Independent Set to an instance $H = (V', E')$ of $(k' = \binom{k}{2} + k + 1)$-Perfect Code such that G has an independent set of size k if and only if H has a perfect code of size k'.

Let the vertex set V of G be $\{1, \ldots, n\}$.

We construct the vertex set V' of H to be the union of the following sets of vertices:

$$
\begin{aligned}
V_0 &= \{h_1, h_2, h_3\} \\
V_1 &= \{a_s \mid 0 \le s \le k' + 1\} \\
V_2 &= \{b_i \mid 1 \le i \le k\} \\
V_3 &= \{c_i \mid 1 \le i \le k\} \\
V_4 &= \{d_{i,u} \mid 1 \le i \le k; u \in V\} \\
V_5 &= \{e[i,j,u] \mid 1 \le i < j \le k; u \in V\} \\
V_6 &= \{f[i,j,u,v] \mid 1 \le i < j \le k; u, v \in V\} \\
V_7 &= \{g_{i,j} \mid 1 \le i < j \le k\}
\end{aligned}
$$

The edge set E' of H is the union of the following sets of edges:

$$
\begin{aligned}
E_0 &= \{\{h_1, h_2\}, \{h_1, h_3\}, \{h_2, h_3\}\} \\
E_1 &= \{\{a_0, a_i\} \mid i = 1, 2\} \\
E_2 &= \{\{a_0, b_i\} \mid 1 \le i \le k\} \\
E_3 &= \{\{b_i, c_i\} \mid 1 \le i \le k\} \\
E_4 &= \{\{c_i, d_{i,u}\} \mid 1 \le i \le k, u \in V\} \\
E_5 &= \{\{d_{i,u} di, v\} \mid 1 \le i \le k; u, v \in V\} \\
E_6 &= \{\{d_{i,u}, e[i,j,u]\} \mid 1 \le i < j \le k, u \in V\} \\
E_7 &= \{\{d_{j,v}, e[i,j,u]\} \mid 1 \le i < j \le k, v \in N[u]\} \\
E_8 &= \{\{e[i,j,x], f[i,j,u,v]\} \mid 1 \le i < j \le k; x \ne u; x \notin N[v]\} \\
E_9 &= \{\{f[i,j,u,v], f[i,j,x,y]\} \mid 1 \le i < j \le k, u \ne x \text{ or } v \ne y\} \\
E_{10} &= \{\{f[i,j,u,v], g_{i,j}\} \mid 1 \le i < j \le k\} \\
E_{11} &= \{\{h_1, g_{i,j}\} \mid 1 \le i < j \le k\}
\end{aligned}
$$

The construction of H is shown in Figure 1.

Suppose the constructed graph H has a perfect code C of size $k' = \binom{k}{2} + k + 1$. We show that G has an independent set of size k. Since a_i for all $i \ge 1$ are attached to a_0, none of them can be in C. It must be true that $a[0] \in C$ to cover all the other nodes in V_1. $a[0]$ covers all the nodes in V_2, so none of the nodes in V_2, V_3 can be present in C. To cover the nodes in V_3, exactly one node from each of the k cliques in V_4 has to be present in C.

Let the set of vertices in G corresponding to $V_4 \cap C$ be called I.

Claim 2. *I is an independent set of size k in G.*

Proof. Suppose for contradiction there exist nodes $u, v \in I$ such that $v \in N[u] \cup \{u\}$. Then WLOG for some $i, j, 1 \le i < j \le k$, $d[i,u], d[j,v] \in C$. Both of these vertices $d[i,u], d[j,v]$ are adjacent to $e[i,j,u]$, so C is not a perfect code. □

Conversely, we show that if G contains an independent set $J = \{u_1, \ldots, u_k\}$ of size k, then H has a perfect code of size k'. The set

$$C_J = \{a[0]\} \cup \{d[i, u_i] : 1 \leq i \leq k\} \cup \{f[i, j, u_i, u_j] : 1 \leq i < j \leq k\}$$

is a perfect code of H with size k'.

We now reduce from the constructed instance H, k' of Perfect Code to k'-SLR.

Given a graph H as constructed above, with n vertices, construct matrix $A \in \{0,1\}^{n \times n}$ with rows and columns indexed by the vertices of the graph. For all $i \neq j$, $A_{ij} = 1$ if vertices i and j are adjacent and $A_{ij} = 0$ otherwise. For all i, $A_{ii} = 1$. The graph contains a perfect code of k' vertices iff $Ax = \vec{1}$ has a k'-sparse solution x. Each entry x_u in the support of x corresponds to a vertex u in the perfect code.

Consider the vertices in V_1. Since we are looking for a k'-sparse solution, there is at least one $i \geq 1$ such that $x_{a[i]} = 0$. This implies $x_{a_0} = 1, x_{a_i} = 0, i \geq 1$. Further, $x_{b_i} = -x_{c_i}$

Considering a_0 and the vertices in V_0, we get the following equations:

$$\begin{aligned} x_{h_1} + x_{h_2} + x_{h_3} &= 0 \\ x_{h_1} + x_{h_3} &= 0 \\ x_{h_1} + x_{h_2} &= 0 \end{aligned}$$

Therefore,

$$x_{h_1} = x_{h_2} = x_{h_3} = 0.$$

Considering the vertex h_1, we get that $\sum_{i,j} x_{g_{i,j}} = 0$. Suppose there exist i, j such that $x_{g_{i,j}} = 0$. Then there must exist nodes $u, v \in V$ such that $x_{f[i,j,u,v]} \neq 0$, and $f[i, j, u, v]$ is in the support of x. This implies that at least $\binom{k}{2}$ vertices from $V_6 \cup V_7$ have to be in the support of x.

We can now add at most k nodes from the remaining graph to the support of x.

We show that at least one node from each of the k cliques in V_4 need to be present in the support. Suppose for contradiction that there exists an i such that no node of the form $d[i, u]$ is in the support of x. This implies that both $b[i], c[i]$ need to be in the support of x. Since we are looking for a k'-sparse solution, this is not possible. It then follows that $x_u = 1$ for all nodes u in the support of x. □

By combining Theorem 4.7 in [5], and Theorem 13, we get the following lower bound for k-SLR over finite fields.

Corollary 6. *Suppose k-SLR over finite fields can be solved in $n^{o(\sqrt{k})}$ time. Then, ETH is false.*

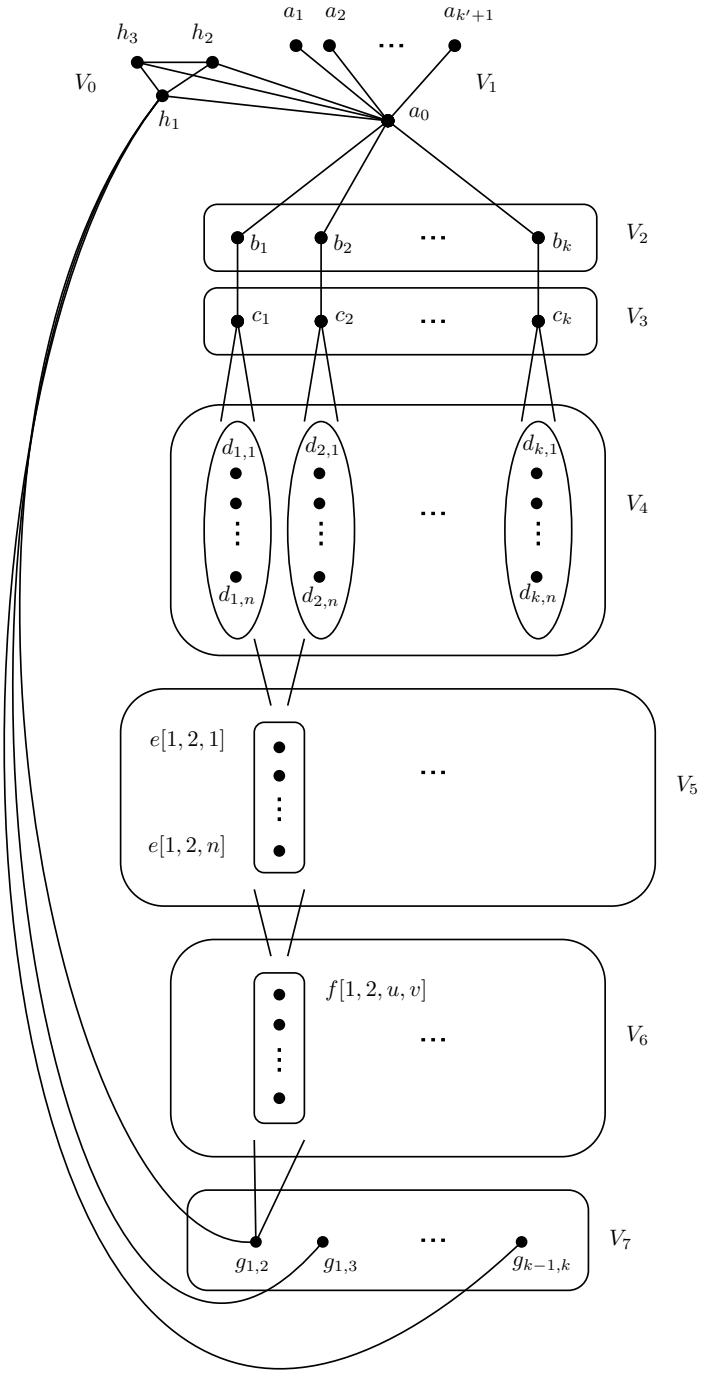

Figure 1: Reduction from Independent Set to Perfect Code

5 Gaussian Random Design

In this section, we sketch the computational equivalence of three different-looking formulations of the sparse linear regression problem. The reductions between the different forms take time $poly(n, k)$.

1. (Gaussian Random Design I) Given access to a sampling oracle generating i.i.d. pairs (X, Y) with $X \sim N(0, \Sigma)$ and $Y = \langle w, X \rangle$ with w being k-sparse, return with high probability a k-sparse w' such that $\langle w, X \rangle = \langle w', X \rangle$ almost surely.

2. (Gaussian Random Design II) If $m = \Omega(k \log(n/\delta))$ and we are given access to $A : m \times n, y$ such that

   $$y = Ax,$$

   the rows of A are sampled from $N(0, \Sigma)$ and the unknown x is k-sparse, return with probability at least $1-\delta$ (over the randomness of A) a k-sparse x' such that $A(x - x') = 0$.

3. (Fixed Design) Given access to an arbitrary $A : m \times n$ and y such that x is k-sparse and $y = Ax$, find a k-sparse solution x' such that $A(x - x') = 0$.

To demonstrate the equivalence, we give a reduction from problem 1 to 2, 2 to 3, and 3 to 1. For 1 to 2 we use the sampling oracle to generate the matrix A with $x = w$ and use the algorithm from 2 to find x' such that $A(x - x') = 0$. Now we use some standard Lemmas which follow from concentration of Wishart matrices; more detailed proofs of these Lemmas can be found in e.g. [10].

Lemma 12. *Suppose $X_1, \ldots, X_m \sim N(0, \Sigma)$ with $\Sigma : d \times d$ and let $\hat{\Sigma} := \frac{1}{m} \sum_{i=1}^{m} X_i$. Then with probability at least $1 - \delta$, $\frac{1}{2}\hat{\Sigma} \preceq \Sigma \preceq 2\hat{\Sigma}$ provides that $m = \Omega(d + \log(2/\delta))$*

Proof. This is a standard result about concentration of Wishart matrices. When $\Sigma = I$ this is Theorem 4.6.1 in [15]. Otherwise, it follows by making a change of basis to reduce to the case $\Sigma = I$. □

Lemma 13. *Suppose $X_1, \ldots, X_m \sim N(0, \Sigma)$ with $\Sigma : d \times d$ and let $\hat{\Sigma} := \frac{1}{m} \sum_{i=1}^{m} X_i$. Then with probability at least $1 - \delta$, every $k \times k$ submatrix of $\hat{\Sigma}$ is a 2-spectral approximation to the correspoding $k \times k$ submatrix of Σ provides $m = \Omega(k \log(d/\delta))$.*

Proof. This follows from Lemma 12 and union bounding over all of the $\binom{d}{k}$ possible choices of $k \times k$ submatrix. □

Based on the last Lemma, we see that with high probability over the randomness of A, $A(x - x') = 0$ implies that $\langle w, X \rangle = \langle x', X \rangle$ almost surely, so taking $w' = x'$ solves the problem. For 2 to 3, clearly an algorithm for problem 3 can solve problem 2. For 3 to 1, define matrix B by adding y as an additional

column to A so that $B(x,-1)=0$ and observe solving this is equivalent to solving $(x,-1)B^TB(x,-1)=0$. Defining $\Sigma = B^TB$ and letting $X \sim N(0,\Sigma)$ we see this is the same as asking for a solution to $\langle (x,-1), X\rangle = 0$ or equivalently $\langle x, X_{\ (n+1)}\rangle = X_{n+1}$ which can be solved using an algorithm for problem 1 as we can generate arbitrarily many samples from $N(0,\Sigma)$ efficiently.

6 Acknowledgements

This paper was written during the 2020 Summer Program in Undergraduate Research (SPUR+) at the Massachusetts Institute of Technology. We would like to thank our mentor Frederic Koehler, our SPUR+ advisors Prof. Ankur Moitra and Prof. David Jerison, and Prof. Slava Gerovitch for directing the research program.

References

[1] Afonso S Bandeira, Edgar Dobriban, Dustin G Mixon, and William F Sawin. Certifying the restricted isometry property is hard. *IEEE transactions on information theory*, 59(6):3448–3450, 2013.

[2] Dimitris Bertsimas, Angela King, and Rahul Mazumder. Best subset selection via a modern optimization lens. *The annals of statistics*, pages 813–852, 2016.

[3] Chris Calabro, Russell Impagliazzo, Valentine Kabanets, and Ramamohan Paturi. The complexity of unique k-sat: An isolation lemma for k-cnfs. *Journal of Computer and System Sciences*, 74(3):386–393, 2008.

[4] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. The complexity of satisfiability of small depth circuits. pages 75–85, 2009.

[5] Jianer Chen, Xiuzhen Huang, Iyad A Kanj, and Ge Xia. Linear fpt reductions and computational lower bounds. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 212–221, 2004.

[6] David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

[7] Rod G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness ii: On completeness for w[1].

[8] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

[9] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280)*, pages 653–662. IEEE, 1998.

[10] Jonathan Kelner, Frederic Koehler, Raghu Meka, and Ankur Moitra. Learning some popular gaussian graphical models without condition number bounds. *arXiv preprint arXiv:1905.01282*, 2019.

[11] Leonid Khachiyan. On the complexity of approximating extremal determinants in matrices. *Journal of Complexity*, 11(1):138–153, 1995.

[12] Ankur Moitra. *Algorithmic aspects of machine learning*. Cambridge University Press, 2018.

[13] Balas Kausik Natarajan. Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24(2):227–234, 1995.

[14] Mihai Pătraşcu and Ryan Williams. On the possibility of faster sat algorithms. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1065–1075. SIAM, 2010.

[15] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

[16] Tengyao Wang, Quentin Berthet, and Yaniv Plan. Average-case hardness of rip certification. In *Advances in Neural Information Processing Systems*, pages 3819–3827, 2016.

[17] Ryan Williams and Huacheng Yu. Finding orthogonal vectors in discrete structures. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1867–1877. SIAM, 2014.

[18] Virginia Vassilevska Williams and Ryan Williams. 6.s078 fine-grained algorithms and complexity - spring 2018, 2018.

[19] Yuchen Zhang, Martin J Wainwright, and Michael I Jordan. Lower bounds on the performance of polynomial-time algorithms for sparse linear regression. In *Conference on Learning Theory*, pages 921–948, 2014.

[20] Yuchen Zhang, Martin J Wainwright, Michael I Jordan, et al. Optimal prediction for sparse linear models? lower bounds for coordinate-separable m-estimators. *Electronic Journal of Statistics*, 11(1):752–799, 2017.