Electronic Colloquium on Computational Complexity, Report No. 193 (2015)

On the Hardness of Learning Sparse Parities

Arnab Bhattacharyya[*] Ameet Gadekar[*] Suprovat Ghoshal[*] Rishi Saket[†]

November 26, 2015

Abstract

This work investigates the hardness of computing sparse solutions to systems of linear equations over $\mathbb{F}_2$. Consider the k-EvenSet problem: given a homogeneous system of linear equations over $\mathbb{F}_2$ on n variables, decide if there exists a nonzero solution of Hamming weight at most k (i.e. a k-sparse solution). While there is a simple $O(n^{k/2})$-time algorithm for it, establishing fixed parameter intractability for k-EvenSet has been a notorious open problem. Towards this goal, we show that unless k-Clique can be solved in $n^{o(k)}$ time, k-EvenSet has no $\mathrm{poly}(n) \cdot 2^{o(\sqrt{k})}$ time algorithm for all k and no polynomial time algorithm when $k = \omega(\log^2 n)$.

Our work also shows that the non-homogeneous generalization of the problem – which we call k-VectorSum – is W[1]-hard on instances where the number of equations is $O(k \log n)$, improving on previous reductions which produced $\Omega(n)$ equations. We use the hardness of k-VectorSum as a starting point to prove the result for k-EvenSet, and additionally strengthen the former to show the hardness of *approximately learning k-juntas*. In particular, we prove that given a system of $O(\exp(O(k)) \cdot \log n)$ linear equations, it is W[1]-hard to decide if there is a k-sparse linear form satisfying all the equations or any function on at most k-variables (a k-junta) satisfies at most $(1/2 + \varepsilon)$-fraction of the equations, for any constant $\varepsilon > 0$. In the setting of computational learning, this shows hardness of approximate *non-proper* learning of k-parities. In a similar vein, we use the hardness of k-EvenSet to show that that for any constant d, unless k-Clique can be solved in $n^{o(k)}$ time, there is no $\mathrm{poly}(m, n) \cdot 2^{o(\sqrt{k})}$ time algorithm to decide whether a given set of m points in $\mathbb{F}_2^n$ satisfies: (i) there exists a non-trivial k-sparse homogeneous linear form evaluating to 0 on all the points, or (ii) any non-trivial degree d polynomial P supported on at most k variables evaluates to zero on $\approx \mathbf{Pr}_{\mathbb{F}_2^n}[P(\mathbf{z}) = 0]$ fraction of the points i.e., P is *fooled* by the set of points.

Lastly, we study the approximation in the sparsity of the solution. Let the Gap-k-VectorSum problem be: given an instance of k-VectorSum of size n, decide if there exist a k-sparse solution, or every solution is of sparsity at least $k' = (1 + \delta_0)k$. Assuming ETH, we show that for some constants c_0, $\delta_0 > 0$ there is no $\mathrm{poly}(n)$ time algorithm for Gap-k-VectorSum when $k = \omega((\log \log n)^{c_0})$.

[*] Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India. Emails: {arnabb, ameet.gadekar, suprovat.ghoshal}@csa.iisc.ernet.in. AB supported in part by DST Ramanujan Fellowship.

[†] IBM Research, Bangalore, India. Email: rissaket@in.ibm.com.

ISSN 1433-8092

1 Introduction

Given a system of linear equations over $\mathbb{F}_2$, does there exist a sparse non-trivial solution? This question is studied in different guises in several areas of mathematics and computer science. For instance, in coding theory, if the system of linear equations is $\mathbf{Mx} = \mathbf{0}$ where $\mathbf{M}$ is the parity check matrix of a binary code, then the minimum (Hamming) weight of a nonzero solution is the distance of the code. This also captures the problem of determining whether a binary matroid has a short cycle, as the latter reduces to deciding whether there is a sparse nonzero $\mathbf{x}$ such that $\mathbf{Mx} = \mathbf{0}$. In learning theory, the well known sparse parity problem is: given a binary matrix $\mathbf{M}$ and a vector $\mathbf{b}$ decide whether there is a small weight nonzero vector $\mathbf{x}$ satisfying $\mathbf{Mx} = \mathbf{b}$. The version where $\mathbf{Mx}$ is required to equal $\mathbf{b}$ in most coordinates, but not necessarily all, is also well studied as the problem of learning noisy parities.

Let a vector $\mathbf{x} \in \mathbb{F}_2^n$ be called *k-sparse* if it is nonzero in at most k positions, i.e. it has Hamming weight at most k. In this work, we show that learning a k-sparse solution to a system of linear equations is fixed parameter intractable, even when (i) the number of equations is only *logarithmic* in the number of variables, (ii) the learning is allowed to be *approximate*, i.e. satisfy only 51% of the equations and, (iii) is allowed to output as hypothesis any function (junta) supported on at most k variables. We also prove variants of these results for the case when the system of equations is homogeneous, which correspond to hardness of the well known k-EvenSet problem. Note that it is always possible to recover a k-sparse solution in $O(n^k)$ time simply by enumerating over all k-sparse vectors. Our results show that for many settings of k, no substantially faster algorithm is possible for k-EvenSet unless widely believed conjectures are false. Assuming similar conjectures, we also rule out fast algorithms for learning γk-sparse solutions to a linear system promising the existence of a k sparse solutions, for some $\gamma > 1$.

In the next few paragraphs we recall previous related work and place our results in their context. Let us first formally define the basic objects of our study:

Definition 1.1. k-VectorSum*: Given a matrix* $\mathbf{M} \in \mathbb{F}_2^{m\times n}$ *and a vector* $\mathbf{b} \in \mathbb{F}_2^m$*, and a positive integer* k *as parameter, decide if there exists a* k*-sparse vector* $\mathbf{x}$ *such that* $\mathbf{Mx} = \mathbf{b}$.

Definition 1.2. k-EvenSet*: Given a matrix* $\mathbf{M} \in \mathbb{F}_2^{m\times n}$*, and a positive integer* k *as parameter, decide if there exists a* k*-sparse vector* $\mathbf{x}$ *such that* $\mathbf{Mx} = \mathbf{0}$.

Remark: In the language of coding theory, k-VectorSum is also known as the MaximumLikelihoodDecoding problem and k-EvenSet as the MinimumDistance problem.

Clearly, k-VectorSum is as hard as k-EvenSet[1]. The k-VectorSum problem was shown to be W[1]-hard[2] by Downey, Fellows, Vardy and Whittle [DFVW99], even in the special case of the vector $\mathbf{b}$ consisting of all 1's. More recently, Bhattacharyya, Indyk, Woodruff and Xie [BIWX11] showed that the time complexity of k-VectorSum is $\min(2^{\Theta(m)}, n^{\Theta(k)})$, assuming 3-SAT has no $2^{o(n)}$ time algorithm.

In contrast, the complexity of k-EvenSet remains unresolved, other than its containment in W[2] shown in [DFVW99]. Proving W[1]-hardness for k-EvenSet was listed as an open problem in Downey and Fellows' 1999 monograph [DF99] and has been reiterated more recently in lists of open problems [FM12, FGMS12]. Note that if we ask for a vector $\mathbf{x}$ whose weight is *exactly* k instead of at most k, the problem is known to be W[1]-hard [DFVW99]. Our work gives evidence ruling out efficient algorithms for k-EvenSet for a wide range of settings of k.

In the non-parameterized setting, where k is part of the input, these problems are very well-studied. Vardy showed that EvenSet (or MinimumDistance) is NP-hard [Var97]. The question of approximating k, the minimum distance of the associated code, has also received attention. Dumer, Micciancio and Sudan [DMS03] showed that if RP $\neq$ NP, then k is hard to approximate within some constant factor $\gamma > 1$. Their reduction was derandomized by Cheng and Wan [CW08, CW09], and subsequently Austrin and Khot [AK14] gave a simpler deterministic reduction for this problem. The results of [CW08, CW09] and [AK14] were further strengthened by Micciancio [Mic14].

[1] The name k-EvenSet is from the following interpretation of the problem: given a set system $\mathcal{F}$ over a universe U and a parameter k, find a nonempty subset $S \subseteq U$ of size at most k such that the intersection of S with every set in $\mathcal{F}$ has even size.

[2] Standard definitions in parameterized complexity appear in Section 2.

From a computational learning perspective, the k-VectorSum problem can be restated as: given an m-sized set of n-dimensional point and value pairs over $\mathbb{F}_2$, decide if there exists a parity supported on at most k variables (i.e. a k-parity) that is consistent with all the pairs. This has been extensively studied as a promise problem when the points are uniformly generated. Note that in this case, if $m = \Omega(n)$, there is a unique solution w.h.p and can be found efficiently by Gaussian elimination. On the other hand, for $m = O(k \log n)$, the best known running time of $O(n^{k/2})$ is given in [KS06] (credited to Dan Spielman). Obtaining a polynomial time algorithm for $m = \text{poly}(k \log n)$ would imply *attribute-efficient learning* of k-parities and is a long-standing open problem in the area [Blu96]. The best known dependence between m and the running time for this problem is described in [BGM10, BGR15]. Our work proves the hardness of k-VectorSum when $m = O(k \log n)$, showing evidence which rules out polynomial time algorithms for learning k-parity when the input is generated adversarially.

A natural question studied in this work is whether one can do better if the learning algorithm is allowed to be *non-proper* (i.e., output a hypothesis that is not a k-parity) and is allowed to not satisfy all the point-value pairs. To further motivate this problem, let us look at the case when k is not fixed. In the absence of noise, Gaussian elimination can efficiently recover a consistent parity. The harder case is the agnostic (noisy) setting which promises that there is a parity consistent with at least $1 - \varepsilon$ fraction of the point-value pairs. When the points are generated uniformly at random, one can learn the parity in time $2^{O(n/\log n)}$ [FGKP09, BKW03].

On the other hand, when the points are adversarially drawn, there is a non-proper algorithm due to Kalai, Mansour and Verbin [KMV08] that runs in time $2^{O(n/\log n)}$ and outputs a circuit C which is consistent with at least $\left(1 - \varepsilon - 2^{-n^{0.99}}\right)$ of the point value pairs. Håstad's inapproximability for Max-3LIN [Hås01] implies that learning a noisy parity in the adversarial setting is NP-hard, even for $1/2 + \varepsilon$ accuracy, for any constant $\varepsilon > 0$. Gopalan, Khot and Saket [GKS10] showed that achieving an accuracy of $1 - 1/2^d + \varepsilon$ using degree-d polynomials as hypotheses is NP-hard. Subsequently, Khot [Kho09] proved an optimal bound of $1/2 + \varepsilon$ for learning by constant degree polynomials[3]. Our work studies the intractability of approximate non-proper learning of k-parity, and extends the hardness result for k-VectorSum to learning by juntas of k variables, and for k-EvenSet to learning using constant degree polynomials on k variables.

Another interesting question in the parameterized setting is related to a gap in the sparsity parameter k, i.e. how tractable it is to learn a γk-sparse solution when the existence of a k-sparse solution is guaranteed, for some constant $\gamma > 1$ (or $\gamma < 1$ in case of a maximization problem). Previously, Bonnet et al. [BEKP15] and Khot and Shinkar [KS15] studied this problem for k-Clique, and both these works show conditional hardness results. In our work we prove a "gap in k" hardness result for k-VectorSum similar to that obtained in [BEKP15] for k-Clique.

In the rest of this section we formally describe our results for k-VectorSum and k-EvenSet, and give a brief description of the techniques used to obtain them.

1.1 Our Results

Hardness of exact problems

We begin by giving a reduction from k-Clique showing the W[1]-hardness of k-VectorSum on instances which have a small number of rows.

Theorem 1.3 (W[1]-hardness of k-VectorSum)**.** *The k-VectorSum problem is W[1]-hard on instances $(\mathbf{M}, \mathbf{b})$ where $\mathbf{M} \in \mathbb{F}_2^{m \times n}$ and $\mathbf{b} \in \mathbb{F}_2^m$ such that $m = O(k \log n)$. Our reduction implies, in particular, that k-VectorSum does not admit an $n^{o(\sqrt{k})}$ time algorithm on such instances, unless k-Clique on r-vertex graphs has an $r^{o(k)}$ time algorithm.*

As far as we know, in previous proofs of the W[1]-hardness of k-VectorSum [DFVW99, CFK+15], the number of rows in the matrix output by the reduction was linear in n. Our proof is inspired by a recent proof

[3] As far as we know, this result is unpublished although it was communicated to the fourth author of this paper. We include with his permission a proof of Khot's result to illustrate some of the techniques which inspire part of this work.

of the W[1]-hardness of k-Sum [ALW14]. Also, in Appendix A, we give a simple $O(n \cdot 2^m)$ time algorithm for k-VectorSum, which suggests that m cannot be made sublogarithmic in n for hard instances.

Next, we give a hardness reduction from k-VectorSum to the k-EvenSet problem.

Theorem 1.4 (Hardness of k-EvenSet). *There is an* FPT *reduction from an instance* $(\mathbf{M}, \mathbf{b})$ *of* k-VectorSum, *where* $\mathbf{M} \in \mathbb{F}_2^{m \times n}$ *and* $\mathbf{b} \in \mathbb{F}_2^m$, *to an instance* $\mathbf{M}'$ *of* $O((k \log n)^2)$-EvenSet, *where* $\mathbf{M}' \in \mathbb{F}_2^{m' \times n'}$ *such that both* m' *and* n' *are bounded by fixed polynomials in* m *and* n.

Using Theorem 1.3, the above yields the following corollary.

Corollary 1.5. *There does not exist a* $\mathrm{poly}(n)$ *time algorithm for* k-EvenSet *when* $k = \omega(\log^2 n)$, *assuming that* k-Clique *does not have a polynomial time algorithm for any* $k = \omega(1)$. *More generally, under the same assumption,* k-EvenSet *does not admit a* $\mathrm{poly}(n) \cdot 2^{o(\sqrt{k})}$ *time algorithm for unrestricted* k.

Proof. Suppose there is a $T(n, k)$ algorithm for k-EvenSet. Chaining together the reductions in Theorem 1.3 and Theorem 1.4, we get a $T(\mathrm{poly}(n), k^4 \log^2 n)$ algorithm for k-Clique. Choosing $k = \omega(1)$ implies the first part of the corollary. For the second part, observe that if $f(x) = 2^{o(\sqrt{x})}$, then $f(k^4 \log^2 n) = n^{o(1)}$ for some $k = \omega_n(1)$. □

To the best of our knowledge, Corollary 1.5 gives the first nontrivial hardness results for parameterized k-EvenSet. Theorem 1.4 is obtained by adapting the hardness reduction for the inapproximability of MinimumDistance by Austrin and Khot [AK14] to the parameterized setting.

Hardness of non-proper and approximately learning sparse parities

The hardness for k-VectorSum proved in Theorem 1.3 can be restated in terms of W[1]-hardness of learning k-parity, i.e., linear forms depending on at most k variables[4].

Theorem 1.6 (Theorem 1.3 restated). *The following is* W[1]*-hard: given* $m = O(k \log n)$ *point-value pairs* $\{(\mathbf{y}_i, a_i)\}_{i=1}^m \subseteq \mathbb{F}_2^n \times \mathbb{F}_2$, *decide whether there exists a* k*-parity* L *which satisfies all the point-value pairs, i.e.,* $L(\mathbf{y}_i) = a_i$ *for all* $i = 1, \ldots, m$.

Next, we strengthen the above theorem in two ways. We show that the W[1]-hardness holds for learning a k-parity using a k-junta, and additionally for any desired accuracy exceeding 50%. Here, a k-junta is any function depending on at most k variables.

Theorem 1.7. *The following is W[1]-hard: for any constant* $\delta > 0$, *given* $m = O(k \cdot 2^{3k} \cdot (\log n)/\delta^3)$ *point-value pairs* $\{(\mathbf{z}_i, b_i)\}_{i=1}^m \subseteq \mathbb{F}_2^n \times \mathbb{F}_2$, *decide whether:*

YES Case: *There exists a* k*-parity which satisfies all the point-value pairs.*

NO Case. *Any function* $f : \mathbb{F}_2^n \mapsto \mathbb{F}_2$ *depending on at most* k *variables satisfies at most* $1/2 + \delta$ *fraction of the point value pairs.*

Theorem 1.7 also implies hardness for approximately learning k-juntas, in comparison to previous W[2]-hardness of *exactly* learning k-juntas shown by Arvind, Köbler and Lindner [AKL09]. Note that the current best algorithm for learning k-junta, even over the uniform distribution, takes $n^{\Omega(k)}$ time [Val12, MOS04].

We similarly strengthen Corollary 1.5 to rule out efficient algorithms for approximately learning a k-sparse solution to a homogeneous linear system using constant degree polynomials supported on at most k variables.

[4] Note that Theorem 1.3 as stated shows hardness of learning homogeneous k-parity i.e., homogeneous k-sparse linear forms (without the constant term). The result can easily be made to hold for any general k-parities by adding a point-value pair which is $(\mathbf{0}, 0)$.

Theorem 1.8. *Assume that k-Clique does not have a $\text{poly}(n)$ time algorithm for any $k = \omega(1)$. Then for any constant $\delta > 0$ and positive integer d, there is no $\text{poly}(m, n) \cdot 2^{o(\sqrt{k})}$ time algorithm to decide whether a given set of m points $\{\mathbf{z}_i\}_{i=1}^{m} \subseteq \mathbb{F}_2^n$ satisfy:*

YES Case: *There exists a nonzero k-parity L such that $L(\mathbf{z}_i) = 0$ for all $i = 1, \ldots, m$.*

NO Case. *Any non-trivial degree d polynomial $P : \mathbb{F}_2^n \mapsto \mathbb{F}_2$ depending on at most k variables satisfies $P(\mathbf{z}_i) = 0$ for at most $\left(\mathbf{Pr}_{\mathbf{z}\in\mathbb{F}_2^n}[P(\mathbf{z}) = 0] + \delta\right)$ fraction of the points.*

The proof of the above theorem relies on an application of Viola's [Vio09] pseudorandom generator for constant degree polynomials, and is inspired by Khot's [Kho09] NP-hardness of learning linear forms using constant degree polynomials.

Gap in sparsity parameter

Using techniques similar to those employed in [BEKP15], we prove the following *gap in k* hardness for k-VectorSum, i.e., hardness of Gap-k-VectorSum.

Theorem 1.9. *Assuming the Exponential Time Hypothesis, there are universal constants $\delta_0 > 0$ and c_0 such that there is no $\text{poly}(N)$ time algorithm to determine whether an instance of Gap-k-VectorSum of size N admits a solution of sparsity k or all solutions are of sparsity at least $(1+\delta_0)k$, for any $k = \omega((\log\log N)^{c_0})$. More generally, under the same assumption, this problem does not admit an $N^{O(k/\omega((\log\log N)^{c_0}))}$ time algorithm for unrestricted k.*

1.2 Our Techniques

Our first result, Theorem 1.3, is based on a gadget reduction from an n-vertex instance of k-Clique creating columns of $\mathbf{M}$ corresponding to the vertices and edges of the graph along with a target vector $\mathbf{b}$. Unlike previous reductions which used dimension linear in the number of vertices, we reuse the same set of coordinates for the vertices and edges by assigning unique logarithmic length patterns to each vertex. In total we create k columns for each vertex and $\binom{k}{2}$ columns for each edge, using $O(k^2 \log n)$ coordinates. The target vector $\mathbf{b}$ ensures that a solution always has at least $k + \binom{k}{2}$ columns, which suffices in the YES case while the NO case requires strictly more columns to sum to $\mathbf{b}$.

For proving Theorem 1.4, we homogenize the instance of Theorem 1.3 by including $\mathbf{b}$ as a column of $\mathbf{M}$. To force the solution to always choose $\mathbf{b}$ we use the approach of Austrin and Khot [AK14] who face the same issue when reducing to the MinimumDistance problem. Since we need to retain the bound on the sparsity of the solution, we cannot use their techniques directly. Instead, we construct a small length sketch of a purported sparse solution and use it as an input to a tensor code based amplification gadget used in [AK14]. Our construction however inflates the parameter k to $O((k \log n)^2)$.

The hardness of approximately learning k-parities with k-juntas given in Theorem 1.7 is obtained by transforming the instance of Theorem 1.3 using an ε-balanced code, along with an analysis of the Fourier spectrum of any k-junta on the resulting distribution. In contrast, Theorem 1.8 is obtained by using the instance of Theorem 1.4 (appropriately transformed using an ε-balanced code) as an input to Viola's construction [Vio09] of pseudorandom generators for degree d polynomials. Note that the $\exp(k)$ blowup in the reduction for Theorem 1.7 rules out its use for proving Theorem 1.8 due to the presence of a $(\log^2 n)$ factor in the sparsity parameter of the instance obtained in Theorem 1.4. On the other hand, the non-homogeneity of the k-VectorSum problem hinders the use of Viola's pseudorandom generator for proving a version (for degree d polynomials on k variables instead of k-juntas) of Theorem 1.7 which avoids the $\exp(k)$ blowup.

For Theorem 1.9, we use the improved *sparsification lemma* of Calabro, Impagliazzo and Paturi [CIP06] followed by Dinur's almost linear PCP construction [Din07] to reduce an n-variable 3-SAT instance to $2^{\varepsilon n}$ Gap-3-SAT instances with almost linear in n clauses and variables. For each instance a corresponding k-VectorSum instance is created by partitioning the clauses into k blocks and adding $\mathbb{F}_2$-valued variables for partial assignments to each block along with non-triviality and consistency equations. In the YES case setting one variable from each block to 1 (i.e. a k-sparse solution) suffices, whereas in the NO case at least γk

variables need to be set to 1, for some constant $\gamma > 1$. The parameters are such that an efficient algorithm to decide the YES and NO cases would violate the Exponential Time Hypothesis for 3-SAT.

Organization of the paper. Theorem 1.3 is proved in Section 3, and using it as the starting point Theorem 1.4 is proved in Section 4. The reduction proving Theorem 1.7 is given in 5, and starts with a restatement of Theorem 1.3. The proofs of Theorems 1.8 and 1.9 are provided in Section 6 and Section 7 respectively.

We also include in Appendix B a proof of Khot's [Kho09] result on NP-hardness of approximately learning linear forms using constant degree polynomials to illustrate the use of Viola's pseudorandom generator [Vio09] which is also used in the proof of Theorem 1.8.

In the next section we give some definitions and results which shall prove useful for the subsequent proofs.

2 Preliminaries

2.1 Parameterized Complexity

A *parameterization* of a problem is a poly(n)-time computable function that assigns an integer $k \geqslant 0$ to each problem instance x of length n (bits). The pair (x, k) is an instance of the corresponding parameterized problem. The parameterized problem is said to be *fixed parameter tractable (FPT)* if it admits an algorithm that runs in time $f(k) \cdot \text{poly}(n)$ where k is the parameter of the input, n is the size of the input, and f is an arbitrary computable function. The W-*hierarchy*, introduced by Downey and Fellows [DF95, DF99], is a sequence of parameterized complexity classes with $\text{FPT} = \text{W}[0] \subseteq \text{W}[1] \subseteq \text{W}[2] \subseteq \cdots$. It is widely believed that $\text{FPT} \neq \text{W}[1]$.

These hierarchical classes admit notions of completeness and hardness under FPT reductions i.e., $f(k) \cdot \text{poly}(n)$-time transformations from a problem A instance (x, k) where $|x| = n$, to an instance (x', k') of problem B where $|x'| = \text{poly}(n)$ and k' is bounded by $f(k)$. For example, consider the k-Clique problem: given a graph G on n vertices and an integer parameter k, decide if G has a clique of size k. The k-Clique problem is W[1]-complete, and serves as a canonical hard problem for many W[1]-hardness reductions including those in this work.

For a precise definition of the W-hierarchy, and a general background on parameterized algorithms and complexity, see [DF99, FG06, CFK+15].

2.2 Coding Theoretic Tools

Our hardness reductions use some basic results from coding theory. For our purposes, we shall be restricting our attention to *linear* codes over $\mathbb{F}_2$ i.e., those which form linear subspaces. A code $\mathcal{C} \subseteq \mathbb{F}_2^n$ is said to be a $[n, k, d]$-binary linear code if $\mathcal{C}$ forms a k-dimensional subspace of $\mathbb{F}_2^n$ such that all nonzero elements (codewords) in $\mathcal{C}$ are of Hamming weight at least d. We use weight $\text{wt}(\mathbf{x})$ of a codeword $\mathbf{x}$ to denote its Hamming weight, *distance* of a code to denote the minimum weight of any nonzero codeword, and *rate* to denote the fraction k/n. A *generator* matrix $\mathbf{G} \in \mathbb{F}_2^{n \times k}$ for $\mathcal{C}$ is such that $\mathcal{C} = \{\mathbf{G}\mathbf{x} \mid \mathbf{x} \in \mathbb{F}_2^k\}$. Also associated with $\mathcal{C}$ is a *parity check matrix* $\mathbf{G}^{\perp} \in \mathbb{F}_2^{(n-k) \times n}$ satisfying: $\mathbf{G}^{\perp}\mathbf{y} = \mathbf{0}$ *iff* $\mathbf{y} \in \mathcal{C}$. We shall use the generator and parity check matrices of well studied code constructions whose properties we state below.

Theorem 2.1 (BCH Codes, Theorem 3 [BR60])**.** *The dimension of the BCH code of block length $n = (2^m - 1)$ and distance d is at least $\left(n - \lceil \frac{d-1}{2} \rceil m\right)$.*

While the above theorem restricts the block length to be of the form $(2^m - 1)$, for general n we can use as the parity check matrix any n columns of the parity check matrix of a BCH code of the minimum length $(2^m - 1)$ greater than or equal to n. In particular, we have the following corollary tailored for our purpose.

Corollary 2.2. *For all lengths n and positive integers $k < n$, there exists a parity check matrix $\mathbf{R} \in \mathbb{F}_2^{20k \log n \times n}$ such that $\mathbf{R}\mathbf{x} \neq 0$ whenever $0 < \text{wt}(\mathbf{x}) < 18k$. Moreover, this matrix can be computed in time* poly(n, k).

The following explicit family of ε-balanced binary linear codes of constant rate was given by Alon et al. [ABN+92].

Theorem 2.3 (ε-balanced codes [ABN+92])**.** *There exists an explicit family of codes $\mathcal{C} \subseteq \mathbb{F}_2^n$ such that every codeword in $\mathcal{C}$ has normalized weight in the range $[1/2 - \varepsilon, 1/2 + \varepsilon]$, and rate $\Omega(\varepsilon^3)$, which can be constructed in time $\mathrm{poly}(n, \frac{1}{\varepsilon})$, where $\varepsilon > 0$ is an arbitrarily small constant.*

Given a linear code $\mathcal{C} \subseteq \mathbb{F}_2^n$, the product code $\mathcal{C}^{\otimes 2}$ consists of $n \times n$ matrices where each row and each column belongs to $\mathcal{C}$; equivalently, $\mathcal{C}^{\otimes 2} = \{\mathbf{G}\mathbf{X}\mathbf{G}^{\mathrm{T}} : \mathbf{X} \in \mathbb{F}_2^{k \times k}\}$ where $\mathbf{G} \in \mathbb{F}_2^{n \times k}$ is the generator matrix for the code $\mathcal{C}$. If the distance $d(\mathcal{C}) = d$, then it is easy to verify that $d(\mathcal{C}^{\otimes 2}) \geqslant d^2$. However, we shall use the following lemma from [AK14] for a tighter lower bound on the Hamming weight when the code word satisfies certain properties.

Lemma 2.4 (Density of Product Codes [AK14])**.** *Let $\mathcal{C} \subseteq \mathbb{F}_2^n$ be a binary linear code of distance $d = d(\mathcal{C})$, and let $\mathbf{Y} \in \mathcal{C}^{\otimes 2}$ be a nonzero codeword with the additional properties that $\mathrm{diag}(\mathbf{Y}) = \mathbf{0}$, and $\mathbf{Y} = \mathbf{Y}^{\mathsf{T}}$. Then, the Hamming weight of $\mathbf{Y}$ is at least $\frac{3}{2}d^2$.*

2.3 Some Useful Tools

The proof of Theorem 1.8 in Section 6 and Khot's [Kho09] result given in Appendix B use Viola's [Vio09] construction of pseudorandom generators which we describe below.

Definition 2.5. *A distribution $\mathcal{D}$ over $\mathbb{F}_2^n$ is said to* ε-fool *degree d polynomials in n-variables over $\mathbb{F}_2$ if for any degree d polynomial P:*

$$\left| \mathop{\mathbf{E}}_{\mathbf{z} \leftarrow \mathcal{D}} [e(P(\mathbf{z}))] - \mathop{\mathbf{E}}_{\mathbf{z} \leftarrow \mathcal{U}} [e(P(\mathbf{z}))] \right| \leqslant \varepsilon,$$

where $\mathcal{U}$ is the uniform distribution over $\mathbb{F}_2^n$ and $e(x) := (-1)^x$ for $x \in \{0, 1\}$.

Theorem 2.6. *Let $\mathcal{Y}_1, \ldots, \mathcal{Y}_d$ be d independent distributions on $\mathbb{F}_2^n$ that each ε-fool linear polynomials. Then the distribution $\mathcal{W} = \mathcal{Y}_1 + \cdots + \mathcal{Y}_d$ ε_d-fools degree-d polynomials where $\varepsilon_d := 16 \cdot \varepsilon^{1/2^{d-1}}$.*

Our proof of Theorem 1.9 in Section 7 uses the following improved *sparsification* lemma of Calabro, Impagliazzo, and Paturi [CIP06].

Lemma 2.7. *There is a deterministic algorithm which, for any $\varepsilon > 0$, transforms an n-variable* 3-CNF *formula F to $F_1, \ldots, F_s \in$* 3-CNF*, each on at most n variables s.t.*

1. *$s \leqslant 2^{\varepsilon n}$.*

2. *F is satisfiable if and only if at least one of $F_1, \ldots, F_s$ is satisfiable.*

3. *The number of clauses in each $F_1, \ldots, F_s$ is at most $O((1/\varepsilon)^9 \cdot n)$.*

4. *The algorithm runs in time $2^{\varepsilon n} \cdot \mathrm{poly}(n)$, where the degree of the polynomial may depend on ε.*

We shall also use in Section 7 the following reduction to Gap-3-SAT implied by the construction of almost linear sized PCPs given by Dinur [Din07].

Theorem 2.8. *There exist universal constants $\gamma_0 > 0$ and c_0, and a polynomial time reduction from a* 3-CNF *formula F on m clauses to a* 3-CNF *formula F' on at most $m(\log m)^{c_0}$ clauses such that: (i)* (YES Case) *if F is satisfiable then F' is satisfiable, and (ii)* (NO Case) *if F is unsatisfiable then at most $(1 - \gamma_0)$ fraction of the clauses of F' are satisfied by any assignment.*

3 W[1]-hardness of k-VectorSum on $O(k \log n)$ Equations

The following theorem implies Theorem 1.3.

Theorem 3.1. *There is an FPT reduction from an instance $G(V, E)$ of k-*Clique*, over n vertices and m edges, to an instance $(\mathbf{M}, \mathbf{b})$ of k'-*VectorSum*, where $\mathbf{M} \in \mathbb{F}_2^{d \times n'}$ such that $k' = O(k^2)$, $d = O(k^2 \log n)$ and n' is polynomial in n and k.*

The rest of this section is devoted to proving the above theorem. We start by observing that a k-clique in a graph $G(V, E)$ can be certified by the pair of mappings $f : [k] \mapsto V$ and $g : \binom{[k]}{2} \mapsto E$, such that $g(i, j) = (f(i), f(j)) \in E \quad \forall i, j \in [k], i < j$. Here, we use $\binom{[k]}{2}$ to represent $\{(i, j) \mid 1 \leqslant i < j \leqslant k\}$. The underlying idea behind the reduction is to construct $\mathbf{M}$ and $\mathbf{b}$ such that f and g exist *iff* there is a sparse set of columns of $\mathbf{M}$ that sums up to $\mathbf{b}$.

Construction of M and b. Let $G(V, E)$ be a k-Clique instance on $n = |V|$ vertices and $m = |E|$ edges, where $V = \{v_1, v_2, \ldots, v_n\}$. For each vertex $v_i \in V$, assign a distinct $N = \lceil \log(n + 1) \rceil$ bit nonzero binary pattern denoted by $\mathbf{q}_i \in \mathbb{F}_2^N$. We first construct a set of vectors – which shall be the columns of $\mathbf{M}$ – corresponding to the vertices and edges. The dimension over which the vectors are defined is partitioned into three sets of coordinates:

Edge-Vertex Incidence Coordinates: These consist of k slots, where each slot consists of $(k - 1)$ subslots, and each subslot in turn consists of N coordinates. In any column of $\mathbf{M}$, a subslot may either contain the N-length pattern of a vertex, or it might be all zeros.

Edge Indicator Coordinates: These are a set of $\binom{k}{2}$ coordinates corresponding to $\{(i, j) \mid 1 \leqslant i < j \leqslant k\}$, indicating whether the vector represents an edge mapped from (i, j). Any column of $\mathbf{M}$ may have at most one of these coordinates set to 1.

Vertex Indicator Coordinates: These are a set of k coordinates corresponding to indices $i \in \{1, \ldots, k\}$, which indicate whether the vector represents a vertex mapped from i. Any column of $\mathbf{M}$ may have at most one of these coordinates set to 1.

Thus, each vector is a concatenation of $k(k - 1)N$ edge-vertex incidence bits, followed by $\binom{k}{2}$ edge indicator bits and k vertex indicator bits, so that $d = k(k - 1)N + \binom{k}{2} + k = O(k^2 \log n)$. For ease of notation, let S_l^j represent the N-sized subset of coordinates belonging to the subslot l of slot j where $j \in [k]$ and $l \in [k - 1]$. We define $\mathbf{q}_i(S_l^j) \in \mathbb{F}_2^d$ to be the vector which contains the pattern of vertex v_i in S_l^j, and is zero everywhere else. For $1 \leqslant i < j \leqslant k$, let $\boldsymbol{\delta}_{i,j} \in \mathbb{F}_2^d$ be the vector which has a 1 at the edge indicator coordinate corresponding to (i, j), and is 0 everywhere else. Similarly, $\boldsymbol{\delta}_i \in \mathbb{F}_2^d$ is the indicator vector which has its ith vertex indicator coordinate set to 1, everything else being 0. Use these components we construct the vertex and edge vectors as follows.

Vertex Vectors: For each vertex $v_i \in V$ and $j \in [k]$, we introduce a vector $\boldsymbol{\eta}(v_i, j) \in \mathbb{F}_2^d$ which indicates that vertex v_i is mapped from index (slot) j i.e., $f(j) = v_i$. The vector is constructed as follows: populate each of the $(k-1)$ subslots of the jth slot with the pattern of vertex v_i (which is $\mathbf{q}_i$), and set its jth vertex indicator coordinate to 1. Formally, $\boldsymbol{\eta}(v_i, j) := \sum_{l=1}^{k-1} \mathbf{q}_i(S_l^j) + \boldsymbol{\delta}_j$. For each vertex there are k vertex vectors resulting in a total of nk vertex vectors.

Edge Vectors: For each edge $e = (v_{i_1}, v_{i_2}) \in E$ where $i_1 < i_2$, and $1 \leqslant j_1 < j_2 \leqslant k$, we introduce a vector that indicates that the pair of indices (slots) (j_1, j_2) is mapped to (v_{i_1}, v_{i_2}) i.e., $g(j_1, j_2) = (v_{i_1}, v_{i_2})$. We construct the vector $\boldsymbol{\eta}(e, j_1, j_2) \in \mathbb{F}_2^d$ as follows: populate $S_{j_2-1}^{j_1}$ with the pattern of vertex v_{i_1}, and $S_{j_1}^{j_2}$ with the pattern of vertex v_{i_2}. Additionally, we set the edge indicator coordinate corresponding to (j_1, j_2) to 1. The vector is formally expressed as, $\boldsymbol{\eta}(e, j_1, j_2) := \mathbf{q}_{i_1}(S_{j_2-1}^{j_1}) + \mathbf{q}_{i_2}(S_{j_1}^{j_2}) + \boldsymbol{\delta}_{j_1,j_2}$. Intuitively, for the lower ordered vertex v_{i_1}, $\boldsymbol{\eta}(e, j_1, j_2)$ cancels out the $(j_2 - 1)$th subslot of slot j_1, and for the higher ordered vertex v_{i_2}, it cancels out the j_1th subslot of its j_2th slot. Note that we are treating (v_{i_1}, v_{i_2}) as an *unordered* pair since $i_1 < i_2$. Therefore, for each edge $e \in E$, and for each choice of $1 \leqslant j_1 < j_2 \leqslant k$, we introduce one edge vector. Hence, there are a total of $m \cdot \binom{k}{2}$ edge vectors in the set.

The vertex and edge vectors constructed above constitute the columns of $\mathbf{M}$. The target vector $\mathbf{b}$ ensures that (i) every solution must have at least k vertex vectors, and $\binom{k}{2}$ edge vectors and (ii) the vectors must cancel each other out in the Edge-Vertex Incidence coordinates. Formally, $\mathbf{b} = \sum_{i\in[k]} \boldsymbol{\delta}_i + \sum_{1\leqslant i<j<k} \boldsymbol{\delta}_{i,j}$. In other words, all the edge and vertex indicator coordinates of $\mathbf{b}$ are set to 1, and everything else to 0.

3.1 YES case

We show that if $G(V,E)$ has a k-Clique, then there exists a set of $k+\binom{k}{2}$ columns of $\mathbf{M}$ that sum to $\mathbf{b}$. Assume that $v_{i_1}, v_{i_2}, \ldots, v_{i_k}$ form a k-clique where $i_1 < i_2 < \cdots < i_k$. We select k vertex vectors $\{\boldsymbol{\eta}(v_{i_j}, j)\}_{j\in[k]}$, and $\binom{k}{2}$ edge vectors $\{\boldsymbol{\eta}(e, j_1, j_2) \mid e = (v_{i_{j_1}}, v_{i_{j_2}}), 1 \leqslant j_1 < j_2 \leqslant k\}$. Since the k vertices form a clique, these vectors always exists. Observe that for any fixed $j \in [k]$, (i) for $\ell = 1, \ldots, j-1$, $\boldsymbol{\eta}(v_{i_j}, j)$ and $\boldsymbol{\eta}(e, \ell, j)$ have the same pattern $\mathbf{q}_{i_j}$ in subslot ℓ of slot j, where $e = (v_{i_\ell}, v_{i_j})$, and (ii) for $\ell = j+1, \ldots, k$, $\boldsymbol{\eta}(v_{i_j}, j)$ and $\boldsymbol{\eta}(e, j, \ell)$ have the same pattern $\mathbf{q}_{i_j}$ in subslot $(\ell-1)$ of slot j, where $e = (v_{i_j}, v_{i_\ell})$. Thus, the $k+\binom{k}{2}$ selected vectors sum to zero on all but the vertex and edge indicator coordinates and thus sum up to $\mathbf{b}$.

3.2 NO Case

Suppose for a contradiction that $\mathcal{S}$ is a subset of columns of $\mathbf{M}$ that sum to $\mathbf{b}$ and that $|\mathcal{S}| \leqslant k + \binom{k}{2}$.

Proposition 3.2. *There are exactly k vertex vectors corresponding to indices (slots) $i \in [k]$ in $\mathcal{S}$. Also, there are exactly $\binom{k}{2}$ edge vectors, one for each pair (i,j) $(1 \leqslant i < j \leqslant k)$ of slots, in $\mathcal{S}$.*

Proof. This follows from the observation that there are $k + \binom{k}{2}$ nonzero indicator coordinates in the target $\mathbf{b}$, and each (edge or vertex) vector contributes exactly one nonzero (edge or vertex) indicator coordinate. Therefore, by a counting argument, k vertex vectors, one each for the indices (slots) $i \in [k]$, must contribute to the k vertex indicator bits. Similarly, $\binom{k}{2}$ edge vectors, one for each pair of slots (i,j) $(1 \leqslant i < j \leqslant k)$, must contribute to the $\binom{k}{2}$ edge indicator bits. □

The above proposition implies that for each pair of vertex vectors there is exactly one edge vector which has a common populated subslot with each of them. So there are exactly $(k-1)$ edge vectors which share a populated subslot with any given vertex vector in $\mathcal{S}$.

Since the k vertex vectors in $\mathcal{S}$ populate distinct slots, in total $k(k-1)$ subslots are populated by the sum of the k vertex vectors. Note that any edge vector populates exactly 2 subslots. Thus, for the $\binom{k}{2} = k(k-1)/2$ edge vectors in $\mathcal{S}$ to sum up to the values in $k(k-1)$ subslots, it must be that no two edge vectors overlap in the same slot-subslot combination.

Thus, for each vertex vector there are exactly $(k-1)$ edge vectors which share distinct populated subslots with it, and these edge vectors must cancel out the corresponding subslots i.e., have the same pattern in the shared subslot as that of the vertex vector. In other words, for any two vertex vectors corresponding to slots i and j respectively $(i < j)$, the edge vector corresponding to the pair (i,j) must cancel one subslot from each one of the two vertex vectors. This is possible only if (i) the k vertex vectors correspond to distinct vertices in G, and (ii) each pair of these vertices have an edge between them for the corresponding edge vector to exist. This implies that G has a k-clique which is a contradiction.

4 Parameterized Reduction for the k-EvenSet problem

The following is a restatement of Theorem 1.4.

Theorem 4.1 (Hardness of k-EvenSet)**.** *Given an instance $(\mathbf{M}, \mathbf{t})$ of k-VectorSum, where $\mathbf{M} \in \mathbb{F}_2^{m\times n}$ and $\mathbf{t} \in \mathbb{F}_2^m$, there is a* poly$(m,n)$ *time reduction to an instance $\mathbf{M}'$ of $O(k^2 \log^2 n)$-EvenSet, where $\mathbf{M}' \in \mathbb{F}_2^{m'\times n'}$ such that m' and n' are polynomial in n and m.*

The rest of this section is devoted to proving the above theorem. The next few paragraphs give an informal description of the reduction. We then define the variables and equations of the k-EvenSet instance, and analyze the completeness and soundness of the reduction.

4.1 Reduction Overview

Let $\mathbf{Mx} = \mathbf{t}$ be a hard instance of k-VectorSum i.e., in the YES case there exists a k-sparse solution, whereas in the NO case all solutions have Hamming weight at least $(k+1)$. We homogenize this affine system by replacing the target vector $\mathbf{t}$ by $a_0\mathbf{t}$ for some $\mathbb{F}_2$-variable a_0, where $a_0\mathbf{t}$ is a coordinate-wise multiplication of $\mathbf{t}$ with the scalar a_0. Clearly, if all $(k+1)$-sparse (including a_0 as a variable) solutions to $\mathbf{Mx} = a_0\mathbf{t}$ have $a_0 = 1$ then the hardness of k-VectorSum implies the desired hardness result for k-EvenSet. However, this may not be true in general: there could exist a k-sparse $\mathbf{x}$ such that $\mathbf{Mx} = \mathbf{0}$. The objective of our reduction therefore, is to ensure that any solution to $\mathbf{Mx} = a_0\mathbf{t}$ that has $a_0 = 0$ with a k-sparse $\mathbf{x}$, must have significantly large weight in other auxiliary variables which we shall add in the construction.

Towards this end, we borrow some techniques from the proof of the inapproximability of MinimumDistance by Austrin and Khot [AK14]. Using transformations by suitable codes we first obtain a $K = O(k \log n)$-length *sketch* $\mathbf{y} = (y_1, \ldots, y_K)$ of $\mathbf{x}$, such that $\mathbf{y}$ is of normalized weight nearly $1/2$ when $\mathbf{x}$ is k-sparse but nonzero. We then construct a codeword $\mathbf{Y} \in \mathbb{F}_2^{K \times K}$, which is intended to be the product codeword $\mathbf{y}\mathbf{y}^T$. However, this relationship cannot be expressed explicitly in terms of linear equations. Instead, for each pair of coordinates $(i,j) \in [K] \times [K]$, we introduce functions $Z_{ij} : \mathbb{F}_2 \times \mathbb{F}_2 \mapsto \mathbb{F}_2$ indicating the value taken by the pair (y_i, y_j) along with constraints that relate the Z_{ij} variables to codewords $\mathbf{y}$ and $\mathbf{Y}$. In fact, the explicit variables $\{Z_{ij}\}$ determine both $\mathbf{y}$ and $\mathbf{Y}$ which are implicit. The constraints also satisfy the key property: if $\mathbf{x}$ is k-sparse, then the number of nonzero Z_{ij} variables is significantly larger when $a_0 = 0$ than when $a_0 = 1$. This forces all sparse solutions to set $a_0 = 1$, which gives us the desired separation in sparsities between the YES and NO cases.

4.2 Constraints

Let $\mathbf{Mx} = \mathbf{t}$ be the instance of k-VectorSum over $\mathbb{F}_2$, in n variables and m equations. We homogenize this system by introducing a new $\mathbb{F}_2$-variable a_0 so that the new system of equations is then given by

$$\mathbf{Mx} = a_0\mathbf{t}, \tag{1}$$

where the $a_0\mathbf{t}$ is the coordinate wise product of $\mathbf{t}$ with the scalar a_0. We also add the following additional constraints and variables.
Linear Sketch Constraints : Let $\mathbf{R} \in \mathbb{F}^{k' \times n}$ be the parity check matrix of a $[n, n-k', 18k]$ linear code, where $k' = 20k \log n$, as defined in Corollary 2.2. Define $\boldsymbol{\eta}$ to be a k'-length sketch of $\mathbf{x}$ using $\mathbf{R}$ as,

$$\boldsymbol{\eta} = \mathbf{Rx}. \tag{2}$$

Mixing Constraints : Let $\mathbf{C} \in \mathbb{F}_2^{K \times k'}$ be the generator matrix of a linear code $\mathcal{C} \subseteq \mathbb{F}_2^K$ as defined in Theorem 2.3 where $\mathcal{C}$ has relative distance $\frac{1}{2} - \varepsilon$ and rate $\Omega(\varepsilon^3)$ for some small $\varepsilon > 0$ and $K = \frac{k'}{\Omega(\varepsilon^3)} \leqslant \frac{20k \log n}{c\varepsilon^3}$, for some constant $c > 0$. We add the constraint

$$\mathbf{y} = \mathbf{C}\boldsymbol{\eta} = \mathbf{CRx}. \tag{3}$$

Product Code Constraints : Let $\mathcal{C}^{\otimes 2} := \mathcal{C} \bigotimes \mathcal{C}$ be the product code with relative distance $\left(\frac{1}{2} - \varepsilon\right)^2$, constructed from $\mathcal{C}$. Let $\mathbf{Y} = \{Y_{ij}\}_{1 \leqslant i,j \leqslant K} \in \mathbb{F}_2^{K \times K}$ be such that $\mathbf{Y} = \mathbf{y}\mathbf{y}^{\mathsf{T}}$. To represent this relation linearly, we introduce variables $\{Z_{ij}(a,b)\}_{a,b \in \mathbb{F}_2}$ for each $1 \leqslant i,j \leqslant K$, which are intended to indicate the value assigned to the pair (y_i, y_j) i.e., $Z_{ij}(a,b) = \quad \{y_i = a, y_j = b\}$. For each $(i,j) \in [K] \times [K]$ we add the following equations,

$$\begin{aligned} Z_{ij}(0,0) + Z_{ij}(0,1) + Z_{ij}(1,0) + Z_{ij}(1,1) &= a_0 & (4) \\ Z_{ij}(1,0) + Z_{ij}(1,1) &= y_i & (5) \\ Z_{ij}(0,1) + Z_{ij}(1,1) &= y_j & (6) \\ Z_{ij}(1,1) &= Y_{ij}. & (7) \end{aligned}$$

Furthermore, we add the constraints

$$\mathbf{QY} = \mathbf{0}, \tag{8}$$

where $\mathbf{Q}$ is the parity check matrix for the product code $\mathcal{C}^{\otimes 2}$, and

$$\begin{aligned} Y_{ij} &= Y_{ji} && \forall i \neq j, && (9)\\ Y_{ii} &= y_i && \forall i \in [K], && (10) \end{aligned}$$

so that $\mathbf{Y}$ preserves the diagonal entries and symmetry of $\mathbf{yy}^T$. Finally, we introduce $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^{r-1}$ and constraints

$$\mathbf{x}^i = \mathbf{x} \qquad \forall i \in [r-1], \tag{11}$$

where $r = \frac{K^2}{16} \leqslant \frac{25k(\log n)^2}{c^2\varepsilon^6}$. These $r-1$ explicit copies of the vector $\mathbf{x}$ are used to balance the Hamming weight of the final solution. Observe that all the variables described above are linear combinations of $a_0, \{Z_{ij}(\cdot,\cdot)\}_{i,j\in[k]}$ and the coordinates of the vectors $\mathbf{x}$ and $\{\mathbf{x}^i\}_{i\in[r-1]}$. Hence, we analyze the sparsity of the solution restricted to these explicit variables. The total number of variables considered is $4K^2 + r\cdot n + 1$.

Remark: The key difference between [AK14] and our reduction is in Equation (2) which constructs a small ($O(k\log n)$)-length sketch of the n-length vector $\mathbf{x}$. This helps us contain the blowup in the sparsity of the solution to $O(k^2\log^2 n)$ instead of $O(n)$.

4.3 Completeness

In the YES case, setting $a_0 = 1$ we obtain a k-sparse $\mathbf{x}$ such that $\mathbf{Mx} = a_0\mathbf{t} = \mathbf{t}$. Furthermore, for each $i,j \in [K]$, exactly one of the Z_{ij} variables would be nonzero. Hence, we have a solution of weight $K^2 + rk + 1$.

4.4 Soundness

Since the solution has to be non-trivial, at least one of $a_0, \mathbf{x}, \mathbf{y}, \mathbf{Y}$ must be nonzero. Note that when $\mathbf{x} = \mathbf{0}$, $\mathbf{y} = \mathbf{0}$ since $\mathbf{y}$ is a homogeneous linear transformation of $\mathbf{x}$. Moreover, we may assume that the weight of $\mathbf{x}$ is at most $\frac{K^2+1}{r} + k + 1 < 18k$ by our setting of r, otherwise the total weight of the solution would be at least $r\cdot\left(\frac{K^2+1}{r} + k + 1\right) \geqslant K^2 + r(k+1) + 1$ due to the copies of $\mathbf{x}$ and we would be done. The construction of $\mathbf{y}$ along with the upper bound of $18k$ on the weight of $\mathbf{x}$ constrains $\mathbf{y}$ to be nonzero when $\mathbf{x}$ is nonzero. Thus, the only three cases we need to consider are:

Case (i): $a_0 = 1$. In this case, any solution $\mathbf{x}$ to $\mathbf{Mx} = a_0\mathbf{t} = \mathbf{t}$ has weight at least $k+1$. Furthermore, for each $i,j \in [K]$, at least one of the four Z_{ij} variables must be nonzero since $a_0 = 1$. Hence, the total Hamming weight of the solution is at least $K^2 + r(k+1) + 1$.

Case (ii): $a_0 = 0, \mathbf{x} \neq \mathbf{0}, \mathbf{y} \neq \mathbf{0}$. By construction, since $\mathbf{y}$ is nonzero it has weight $\geqslant \left(\frac{1}{2} - \varepsilon\right)K$. Therefore, for at least $1 - \left(\frac{1}{2} + \varepsilon\right)^2 \geqslant \frac{3}{4} - 2\varepsilon$ fraction of the pairs $(i,j) \in [K]\times[K]$, either $y_i = 1$ or $y_j = 1$. Observe that for each such pair, at least two Z_{ij} variables are set to 1. Thus, the weight of any solution in this case is at least $2\left(\frac{3}{4} - 2\varepsilon\right)K^2 = \left(\frac{3}{2} - 4\varepsilon\right)K^2$.

Case (iii): $a_0 = 0, \mathbf{x} = \mathbf{0}, \mathbf{y} = \mathbf{0}, \mathbf{Y} \neq \mathbf{0}$. We have that $\operatorname{diag}(\mathbf{Y}) = \mathbf{y} = \mathbf{0}$, $\mathbf{Y}$ is symmetric and it belongs to the product code $\mathcal{C}^{\otimes 2}$ (as enforced by Equations (8) and (9)). Then by lemma 2.4, the weight of $\mathbf{Y}$ is at least $\left(\frac{3}{8} - 3\varepsilon\right)K^2$. Observe that for each $i,j \in [K]$ such that $Y_{ij} = 1$, Equations (4)-(7) force all four Z_{ij} variables to be set to 1. Hence, the number of nonzero Z_{ij}'s are at least $\left(\frac{3}{2} - 12\varepsilon\right)K^2$.

The above analysis yields that in contrast to the YES case which admits a $(K^2 + rk + 1)$-sparse solution, in the NO case all solutions are of weight at least

$$\min\left\{\left(K^2 + r(k+1) + 1\right), \left(\left(\frac{3}{2} - 12\varepsilon\right)K^2\right)\right\} \geqslant K^2 + r(k+1) + 1$$

by choice of the parameter r. Thus, solving the d-EvenSet problem with $d = K^2 + rk + 1 = O(k^2(\log n)^2)$ solves the k-VectorSum instance $\mathbf{Mx} = \mathbf{t}$.

5 Hardness of Learning k-Parities using k-Juntas

The hardness for k-VectorSum proved in Theorem 1.3 can be restated in terms of W[1]-hardness of learning k-parities, i.e. linear forms depending on at most k-variables.

Theorem 5.1. *The following is W[1]-hard: given $r = O(k \log n)$ point-value pairs $\{(\mathbf{y}_i, a_i)\}_{i=1}^r \subseteq \mathbb{F}_2^n \times \mathbb{F}_2$, decide whether there exists a homogeneous linear form L supported on at most k variables which satisfies all the point-value pairs, i.e. $L(\mathbf{y}_i) = a_i$ for all $i = 1, \ldots, t$.*

Combining the above with a small bias linear code we induce an approximation gap for learning k-parities along with extending the result to non-homogeneous linear forms.

Theorem 5.2. *The following is W[1]-hard: for any $\varepsilon > 0$ depending only on k, given $t = O(k \log n/\varepsilon^3)$ point-value pairs $\{(\mathbf{z}_i, b_i)\}_{i=1}^t \subseteq \mathbb{F}_2^n \times \mathbb{F}_2$, decide whether:*

YES Case: *There exists a homogeneous linear form supported on at most k variables which satisfies all the point-value pairs.*

NO Case. *Any linear form supported on at most k variables satisfies a fraction in the range $[1/2 - \varepsilon, 1/2 + \varepsilon]$ of the point value pairs.*

Proof. Let $\mathbf{W} = \{W_{ij}\} \in \mathbb{F}_2^{t \times r}$ be the generator matrix of an ε-balanced linear code given by Theorem 2.3, where $t = O(r/\varepsilon^3)$. Given an instance $\{(\mathbf{y}_j, a_j)\}_{j=1}^r$ from Theorem 5.1, let

$$\mathbf{z}_i = \sum_{j=1}^r W_{ij}\mathbf{y}_j, \qquad \text{and} \qquad b_i = \sum_{j=1}^r W_{ij}a_j,$$

for $i = 1, \ldots, t$.

In the YES case, there is a homogeneous linear form L^* that satisfies all $\{(\mathbf{y}_j, a_j)\}_{j=1}^r$ and thus satisfies linear combinations of these point-value pairs, in particular $\{(\mathbf{z}_i, b_i)\}_{i=1}^t$.

For the NO case, consider any linear form $L(\mathbf{x}) + c$. Since the homogeneous part L does not satisfy all pairs $\{(\mathbf{y}_j, a_j)\}_{j=1}^r$, it will satisfy a fraction in the range $[1/2 - \varepsilon, 1/2 + \varepsilon]$ of the pairs $\{(\mathbf{z}_i, b_i)\}_{i=1}^t$, due the lower and upper bounds bound on the weight of the nonzero codewords in the column space of $\mathbf{W}$. Thus, the linear form $L(\mathbf{x}) + c$ also satisfies a fraction in the range $[1/2 - \varepsilon, 1/2 + \varepsilon]$ of the point-value pairs $\{(\mathbf{z}_i, b_i)\}_{i=1}^t$. □

As we show below, using a small enough bias ε in the above construction, one can strengthen the hardness result to learning k-parities with *k-juntas*, i.e. functions depending only on a subset of at most k variables.

Theorem 5.3. (Theorem 1.7 restated) *The following is W[1]-hard: for any constant $\delta > 0$, given $t = O(k \cdot 2^{3k} \cdot \log n/\delta^3)$ point-value pairs $\{(\mathbf{z}_i, b_i)\}_{i=1}^t \subseteq \mathbb{F}_2^n \times \mathbb{F}_2$, decide whether:*

YES Case: *There exists a homogeneous linear form supported on at most k variables which satisfies all the point-value pairs.*

NO Case. *Any function $f : \mathbb{F}_2^n \mapsto \mathbb{F}_2$ depending on at most k variables satisfies at most $1/2 + \delta$ fraction of the point value pairs.*

Proof. The construction of $\mathcal{Z} = \{(\mathbf{z}_i, b_i)\}_{i=1}^t$ is exactly the same as in the proof of Theorem 5.2 taking $\varepsilon = \delta \cdot 2^{-k}$. The YES case follows directly as before.

For the NO case, let $f : \mathbb{F}_2^n \mapsto \mathbb{F}_2$ be a function depending only a subset $S \subseteq [n]$ of coordinates where $|S| \leqslant k$. Define an extension $g : \mathbb{F}_2^{n+1} \mapsto \mathbb{F}_2$ as $g(x_1, \ldots, x_n, x_{n+1}) := f(x_1, \ldots, x_n) + x_{n+1}$. For convenience we shall abuse notation to denote $(\mathbf{z}, b) = (z_1, \ldots, z_n, b)$ where $\mathbf{z} = (z_1, \ldots, z_n) \in \mathbb{F}_2^n$ and $b \in \mathbb{F}_2$. To complete the proof we need to show that,

$$\left| \underset{(\mathbf{z},b)\in\mathcal{Z}}{\mathbf{E}} [e(g(\mathbf{z}, b))] \right| \leqslant 2\delta, \tag{12}$$

where $e(x) := (-1)^x$. For some real values C_α ($\alpha \subseteq [n+1]$), the Fourier expansion of $e(g)$ is given by,

$$e(g) = \sum_{\alpha \subseteq [n+1]} C_\alpha \chi_\alpha.$$

Since $e(g(x_1, \ldots, x_{n+1})) = e(f(x_1, \ldots, x_n) + x_{n+1})$ and f depends only on coordinates in S, it is easy to see that the Fourier spectrum of $e(g)$ is supported only on characters χ_α such that $\alpha \subseteq S \cup \{n+1\}$. Further, since $e(g(x_1, \ldots, x_{n+1}))$ changes sign on flipping x_{n+1}, $C_\alpha \neq 0 \Rightarrow (n+1) \in \alpha$. Thus,

$$e(g) = \sum_{\substack{\alpha \subseteq S \cup \{n+1\} \\ (n+1) \in \alpha}} C_\alpha \chi_\alpha. \tag{13}$$

Observe that for any α in the sum above, $\chi_\alpha(x_1, \ldots, x_n, b) = e(L(x_1, \ldots, x_n) + b)$ where L is a homogeneous linear form supported on at most k variables. For any such α, the NO case of Theorem 5.2 implies,

$$\left| \mathop{\mathbf{E}}_{(\mathbf{z}, b) \in \mathcal{Z}} [\chi_\alpha(\mathbf{z}, b)] \right| \leqslant 2\varepsilon \tag{14}$$

Using the above along with Equation (13) yields,

$$\begin{aligned} \left| \mathop{\mathbf{E}}_{(\mathbf{z}, b) \in \mathcal{Z}} [e(g(\mathbf{z}, b))] \right| \quad &\leqslant \quad (2\varepsilon) \cdot \sum_{\substack{\alpha \subseteq S \cup \{n+1\} \\ (n+1) \in \alpha}} |C_\alpha| \\ &\leqslant \quad (2\varepsilon) \cdot 2^k = 2\delta, \end{aligned}$$

where the last inequality is because there are at most 2^k subsets α in the sum on the RHS of Equation (13) and each $|C_\alpha| \leqslant 1$ since $e(g)$ is a $\{-1, 1\}$-valued function. □

6 Proof of Theorem 1.8

We first prove the following strengthening of Theorem 1.4 along the same lines as Theorem 5.2 in Section 5.

Theorem 6.1 (Hardness of approximate k-EvenSet)**.** *For any constant $\varepsilon > 0$, given an instance $(\mathbf{A}, \mathbf{b})$ of k'-VectorSum, where $\mathbf{A} \in \mathbb{F}_2^{m' \times n'}$ and $\mathbf{b} \in \mathbb{F}_2^{m'}$, there is an* FPT *reduction to an instance $\mathbf{B} \in \mathbb{F}_2^{m \times n}$ of k'-EvenSet for some $k = O((k' \log n')^2)$, such that*

YES Case: *There is a nonzero k-sparse vector $\mathbf{x}$ which satisfies $\mathbf{Bx} = 0$.*

NO Case. *For any nonzero k-sparse vector $\mathbf{x}$ the weight of $\mathbf{Bx}$ is in the range $[1/2 - \varepsilon, 1/2 + \varepsilon]$.*

Here both m and n are bounded by fixed polynomials in m' and n'.

Proof. Let $\mathbf{M} \in \mathbb{F}_2^{r \times n}$ be the instance of k-EvenSet obtained by applying Theorem 1.4 to the instance $(\mathbf{A}, \mathbf{b})$ of k'-VectorSum we start with. As in the proof of Theorem 5.2 let $\mathbf{W} \in \mathbb{F}_2^{m \times r}$ be the generator matrix of an ε-balanced linear code given by Theorem 2.3, where $m = O(r/\varepsilon^3)$. Taking $\mathbf{B} := \mathbf{WM}$ completes the proof. □

It is easy to see that the uniform distribution on the rows of the matrix $\mathbf{B}$ fools all linear forms (with error ε) over k variables. Viola's result [Vio09] (Theorem 2.6) implies that for any constant d, taking d-wise sums of the rows of $\mathbf{M}$ yields a distribution which fools all degree d polynomials with error $16 \cdot \varepsilon^{1/2^{d-1}}$. Taking ε to be a small enough constant yields the following theorem which implies Theorem 1.8.

Theorem 6.2. *For any constants $\delta > 0$ and positive integer d, given an instance $(\mathbf{A}, \mathbf{b})$ of k'-VectorSum, where $\mathbf{A} \in \mathbb{F}_2^{m' \times n'}$ and $\mathbf{b} \in \mathbb{F}_2^{m'}$, there is an* FPT *reduction to a set of m points $\{\mathbf{z}_i\}_{i=1}^m \subseteq \mathbb{F}_2^n$ such that for some $k = O((k' \log n')^2)$,* YES Case: *There exists a k-parity L such that $L(\mathbf{z}_i) = 0$ for all $i = 1, \ldots, m$.*

NO Case. *Any degree d polynomial $P : \mathbb{F}_2^n \mapsto \mathbb{F}_2$ depending on at most k variables satisfies $P(\mathbf{z}_i) = 0$ for at most $\left(\mathbf{Pr}_{\mathbf{z}\in\mathbb{F}_2^n}[P(\mathbf{z}) = 0] + \delta\right)$ fraction of the points.*

In the above m and n are bounded by fixed polynomials in m' and n'.

7 Hardness Reduction for Gap-k-VectorSum

In this section we first prove the following theorem.

Theorem 7.1. *For universal constants $\delta_0 > 0$ and c_0, and any arbitrarily small constant $\varepsilon > 0$, there is a $2^{O(\varepsilon n)}$-time Turing reduction from an n-variable* 3-CNF *formula F to $s \leqslant 2^{\varepsilon n}$ instances $I_1, \ldots, I_s$ of* Gap-k-VectorSum, *each of size at most $O(k^2 \cdot 2^{O(n'/k)})$ where $n' = c_\varepsilon n(\log n)^{c_0}$ for some constant c_ε depending on ε, such that*

YES Case. *If F is satisfiable, then at least one instance I_j ($j \in [s]$) admits a solution of sparsity k.*

NO Case. *If F is unsatisfiabe, then no instance I_j admits a solution of sparsity $\leqslant (1 + \delta_0)k$.*

Proof. Let F be a 3-CNF formula on n-variables. We use Lemma 2.7 to obtain 3-CNF fomulas $H_1, \ldots H_s$ for $s \leqslant 2^{\varepsilon n}$, each on at most n variables and $O((1/\varepsilon)^9 \cdot n)$ clauses. Using Theorem 2.8, each H_j ($j \in [s]$) is separately transformed into a Gap-3-SAT instance F_j with at most $n' = c_\varepsilon n(\log n)^{c_0}$ clauses (and variables), for some $c_\varepsilon = O((1/\varepsilon)^9)$. Each F_j is now reduced to an instance I_j of Gap-k-VectorSum as follows.

Fix $j \in [s]$ and let $\mathcal{C}(F_j)$ denote the set of clauses of F_j. We may assume that $|\mathcal{C}(F_j)|$ is divisible by k by adding up to k dummy clauses which are always satisfied. Let $B_1^j, \ldots, B_k^j$ be any arbitrary partition of $\mathcal{C}(F_j)$ into k equal sized subsets, and let $X(B_i^j)$ be the set of variables contained in the clauses B_i^j. Let $\mathrm{SAT}(B_i^j) \subseteq \{0,1\}^{X(B_i^j)}$ be the set of assignments to the variables $X(B_i^j)$ that satisfy all the clauses in B_i^j, for $1 \leqslant i \leqslant k$. For each $\alpha \in \mathrm{SAT}(B_i^j)$ we introduce an $\mathbb{F}_2$-valued variable z_α, and add the following equation,

$$\sum_{\alpha\in\mathrm{SAT}(B_i^j)} z_\alpha = 1, \tag{15}$$

for each $i = 1, \ldots, k$. We also add equations to ensure consistency of the solution across different blocks of variables. For each pair of distinct blocks B_i^j and $B_{i'}^j$ and each assignment to their common variables $\sigma \in \{0,1\}^{X(B_i^j)\cap X(B_{i'}^j)}$ we add the following equation.

$$\sum_{\substack{\alpha\in\mathrm{SAT}(B_i^j)\\ \sigma=\alpha|_{X(B_i^j)\cap X(B_{i'}^j)}}} z_\alpha = \sum_{\substack{\beta\in\mathrm{SAT}(B_{i'}^j)\\ \sigma=\beta|_{X(B_i^j)\cap X(B_{i'}^j)}}} z_\beta. \tag{16}$$

It is easy to see that $|X(B_i^j)| = O(n'/k)$ and $|\mathrm{SAT}(B_i^j)| \leqslant 2^{O(n'/k)}$ for $1 \leqslant i \leqslant k$. Thus, the above construction of I_j has at most $k \cdot 2^{O(n'/k)}$ variables, and at most $k^2 \cdot 2^{O(n'/k)}$ equations. The constant c_0 is the same as in Theorem 2.8 and δ_0 shall be chosen later.

7.1 YES Case

If F is satisfiable, the for some $j^* \in [s]$, H_{j^*} is satisfiable and therefore F_{j^*} is satisfiable. Let π be a satisfying assignment for F_{j^*}. In I_{j^*}, for each $B_i^{j^*}$ ($1 \leqslant i \leqslant k$) we set the variable z_α corresponding to the projection $\alpha \in \mathrm{SAT}(B_i^{j^*})$ of π on to the variables $X(B_i^{j^*})$ to 1, and the rest of the variables to 0. Clearly, this satisfies Equations (15) and (16) since π is a satisfying assignment. As we set exactly one variable per block to 1, I_{j^*} admits a solution of sparsity k.

7.2 NO Case

If F is not satisfiable, then none of $H_1, \ldots, H_s$ are satisfiable and thus, for each $j = 1, \ldots, s$, at most $(1-\gamma_0)$ fraction of the clauses of F_j are satisfiable by any assignment. Fix any $j \in [s]$. Since $B_1^j, \ldots, B_k^j$ is a balanced partition of $\mathcal{C}(F_j)$, any assignment to the variables of F_j can satisfy all the clauses of at most $(1-\gamma_0)$ fraction of B_i^j, $1 \leqslant i \leqslant k$.

Consider a solution to I_j of sparsity at most $(1+\delta_0)k$. By Equation (15), this solution must set an odd number of variables in $\{z_\alpha \mid \alpha \in \mathrm{SAT}(B_i^j)\}$ to 1, for $1 \leqslant i \leqslant k$. Let $S \subseteq [k]$ consist of all indices $i \in [k]$ such that exactly one variable z_{α^i} for some $\alpha^i \in \mathrm{SAT}(B_i^j)$ is set to 1. Thus, the sparsity is at least $|S| + 3(k - |S|)$, which is at most $(1+\delta_0)k$ by our assumption. By rearranging we get $|S| \geqslant (1 - \delta_0/2)k$. Further, Equation (16) implies that for any $i, i' \in S$, the assignments α^i and $\alpha^{i'}$ are consistent on their common variables. Thus, there is an assignment to the variables in $\cup_{i \in S} X(B_i^j)$ that satisfies all the clauses of B_i^j for $i \in S$. Choosing $\delta_0 = \gamma_0$ yields a contradiction to our assumption, and therefore no I_j admits a solution of sparsity at most $(1+\delta_0)k$. □

Theorem 7.1 proved above implies the following restatement of Theorem 1.9.

Theorem 7.2. *Assuming the Exponential Time Hypothesis, there are universal constants $\delta_0 > 0$ and c_0 such that there is no $\mathrm{poly}(N)$ time algorithm to determine whether an instance of* Gap-k-VectorSum *of size N admits a solution of sparsity k or all solutions are of sparsity at least $(1+\delta_0)k$, for any $k = \omega((\log\log N)^{c_0})$. More generally, under the same assumption, this problem does not admit an $N^{O(k/\omega((\log\log N)^{c_0}))}$ time algorithm for unrestricted k.*

Proof. For the first part of the theorem, assume for a contradiction that such an algorithm exists. In Theorem 7.1, the size of each Gap-k-VectorSum instance constructed is at most N, where $\log N = O(\log k + n'/k) = O(\log k) + O(c_\varepsilon n (\log n)^{c_0}/k)$. Here n is the number of variables in the 3-CNF formula F. Thus, choosing $k = \omega((\log\log N)^{c_0})$ implies $k = \omega((\log n)^{c_0})$. Note that in the reduction k is bounded by $\mathrm{poly}(n)$. Our supposed algorithm would decide each Gap-k-VectorSum instance in time $\mathrm{poly}(k, 2^{O(n'/k)}) = 2^{o(n)}$. Applying this to all the instances of Gap-k-VectorSum would decide the n-variable 3-CNF formula F in time $2^{(\varepsilon+o(1))n}$ for all constants $\varepsilon > 0$, which contradicts the ETH.

A similar analysis proves the second part (unrestricted k) of the theorem. □

Acknowledgements

We thank Subhash Khot for his permission to include his proof for the hardness of learning parities using polynomials. Also, thanks to Ryan Williams for encouraging and stimulating conversations and to Rocco Servedio for bringing [AKL09] to our attention .

References

[ABN$^+$92] Noga Alon, Jehoshua Bruck, Joseph Naor, Moni Naor, and Ron M Roth. Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs. *IEEE Trans. Inform. Theory*, 38(2):509–516, 1992.

[AK14] Per Austrin and Subhash Khot. A simple deterministic reduction for the gap minimum distance of code problem. *IEEE Trans. Inform. Theory*, 60(10):6636–6645, 2014.

[AKL09] Vikraman Arvind, Johannes Köbler, and Wolfgang Lindner. Parameterized learnability of juntas. *Theor. Comp. Sci.*, 410(47-49):4928–4936, 2009.

[ALW14] Amir Abboud, Kevin Lewi, and Ryan Williams. Losing weight by gaining edges. In *Proc. 22nd Annual European Symposium on Algorithms*, pages 1–12. Springer, 2014.

[BEKP15] Edouard Bonnet, Bruno Escoffier, Eun Jung Kim, and Vangelis Th. Paschos. On subexponential and fpt-time inapproximability. *Algorithmica*, 71(3):541–565, 2015.

[BGM10] Harry Buhrman, David García-Soriano, and Arie Matsliah. Learning parities in the mistake-bound model. *Inform. Process. Lett.*, 111(1):16–21, 2010.

[BGR15] Arnab Bhattacharyya, Ameet Gadekar, and Ninad Rajgopal. On learning k-parities with and without noise. *arXiv preprint arXiv:1502.05375*, 2015.

[BIWX11] Arnab Bhattacharyya, Piotr Indyk, David P Woodruff, and Ning Xie. The complexity of linear dependence problems in vector spaces. In *Proc. 2nd Innovations in Computer Science*, pages 496–508, 2011.

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, 2003.

[Blu96] Avrim Blum. On-line algorithms in machine learning. In *Workshop on on-line algorithms, Dagstuhl*, pages 305–325. Springer, 1996.

[BR60] R. C. Bose and Dwijendra K. Ray-Chaudhuri. On A class of error correcting binary group codes. *Information and Control*, 3(1):68–79, 1960.

[CFK$^+$15] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms.* Springer International Publishing, 2015.

[CIP06] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. A duality between clause width and clause density for SAT. In *21st Annual IEEE Conference on Computational Complexity*, pages 252–260, 2006.

[CW08] Qi Cheng and Daqing Wan. Complexity of decoding positive-rate reed-solomon codes. In *Proc. 35th Annual International Conference on Automata, Languages, and Programming*, pages 283–293. Springer, 2008.

[CW09] Qi Cheng and Daqing Wan. A deterministic reduction for the gap minimum distance problem. In *Proc. 41st Annual ACM Symposium on the Theory of Computing*, pages 33–38. ACM, 2009.

[DF95] Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness I: basic results. *SIAM J. on Comput.*, 24(4):873–921, 1995.

[DF99] Rodney G Downey and Michael Ralph Fellows. *Parameterized complexity.* Springer Science & Business Media, 1999.

[DFVW99] Rod G Downey, Michael R Fellows, Alexander Vardy, and Geoff Whittle. The parametrized complexity of some fundamental problems in coding theory. *SIAM J. on Comput.*, 29(2):545–570, 1999.

[Din07] Irit Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3), 2007.

[DMS03] Ilya Dumer, Daniele Micciancio, and Madhu Sudan. Hardness of approximating the minimum distance of a linear code. *IEEE Trans. Inform. Theory*, 49(1):22–37, 2003.

[FG06] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory.* Springer Verlag, 2006.

[FGKP09] Vitaly Feldman, Parikshit Gopalan, Subhash Khot, and Ashok Kumar Ponnuswami. On agnostic learning of parities, monomials, and halfspaces. *SIAM J. on Comput.*, 39(2):606–645, 2009.

[FGMS12] Michael R Fellows, Jiong Guo, Dániel Marx, and Saket Saurabh. Data reduction and problem kernels (Dagstuhl Seminar 12241). *Dagstuhl Reports*, 2(6):26–50, 2012.

[FM12] Fedor V Fomin and Dániel Marx. FPT suspects and tough customers: Open problems of Downey and Fellows. In *The Multivariate Algorithmic Revolution and Beyond*, pages 457–468. Springer, 2012.

[GKS10] Parikshit Gopalan, Subhash Khot, and Rishi Saket. Hardness of reconstructing multivariate polynomials over finite fields. *SIAM J. on Comput.*, 39(6):2598–2621, 2010.

[Hås01] Johan Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001.

[Kho06] Subhash Khot. Ruling out PTAS for graph min-bisection, dense k-subgraph, and bipartite clique. *SIAM J. on Comput.*, 36(4):1025–1071, 2006.

[Kho09] Subhash Khot. personal communication, 2009.

[KMV08] Adam Tauman Kalai, Yishay Mansour, and Elad Verbin. On agnostic boosting and parity learning. In *Proc. 40th Annual ACM Symposium on the Theory of Computing*, pages 629–638. ACM, 2008.

[KS06] Adam R Klivans and Rocco A Servedio. Toward attribute efficient learning of decision lists and parities. *J. Mach. Learn. Res.*, 7:587–602, 2006.

[KS15] Subhash Khot and Igor Shinkar. On hardness of approximating the parameterized clique problem. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:13, 2015. http://eccc.hpi-web.de/report/2015/013.

[LW05] Michael Luby and Avi Wigderson. Pairwise independence and derandomization. *Foundation and Trends in Theoretical Computer Science*, 1(4):237–301, 2005.

[Mic14] Daniele Micciancio. Locally dense codes. In *Proc. 29th Annual IEEE Conference on Computational Complexity*, pages 90–97. IEEE, 2014.

[MOS04] Elchanan Mossel, Ryan O'Donnell, and Rocco A. Servedio. Learning functions of k relevant variables. *J. Comp. Sys. Sci.*, 69(3):421–434, November 2004.

[Val12] Gregory Valiant. Finding correlations in subquadratic time, with applications to learning parities and juntas. In *Proc. 53rd Annual IEEE Symposium on Foundations of Computer Science*, pages 11–20. IEEE, 2012.

[Var97] Alexander Vardy. The intractability of computing the minimum distance of a code. *IEEE Trans. Inform. Theory*, 43(6):1757–1766, 1997.

[Vio09] Emanuele Viola. The sum of D small-bias generators fools polynomials of degree D. *Computational Complexity*, 18(2):209–217, 2009.

A A simple $O(n \cdot 2^m)$-time algorithm for k-VectorSum

Let $(\mathbf{M}, \mathbf{b})$ be an instance of k-VectorSum where $\mathbf{M} \in \mathbb{F}_2^{m \times n}$ and $\mathbf{b} \in \mathbb{F}_2^m$. Construct a graph G on vertex set $V = \mathbb{F}_2^m$ and edge set given by,

$$E = \left\{ \{\mathbf{u}, \mathbf{v}\} \in \binom{V}{2} \mid \mathbf{u} + \mathbf{v} \text{ is a column of } \mathbf{M} \right\}.$$

We say that an edge $\{\mathbf{u}, \mathbf{v}\} \in E$ is labeled by the column $\mathbf{u} + \mathbf{v}$ of $\mathbf{M}$. Clearly, if there is a vector $\mathbf{x}$ of Hamming weight at most k such that $\mathbf{Mx} = \mathbf{b}$ then there is a path of length at most k in G from $\mathbf{0}$ to $\mathbf{b}$ given by choosing the edges labeled by the columns corresponding to the non-zero entries of $\mathbf{x}$ in any sequence. On the other hand, if there is a path in G from $\mathbf{0}$ to $\mathbf{b}$ of length at most k, then there is a sequence of at most k columns (with possible repetitions) of $\mathbf{M}$ which sum up to $\mathbf{b}$. Cancelling out even number of repetitions of any column yields a subset of at most k distinct columns of $\mathbf{M}$ that sum up to $\mathbf{b}$. Thus, deciding k-VectorSum reduces to determining whether there is a path of length at most k from $\mathbf{0}$ to $\mathbf{b}$.

The size of V is 2^r and of E is at most $n \cdot 2^m$, and the graph can be constructed in time $O(n \cdot 2^m)$. Doing a Breadth First Search yields a running time of $O(n \cdot 2^m)$.

B Optimal hardness of learning parities using degree d polynomials

The hardness reduction in this section is due to Khot [Kho09].

The starting point of the reduction is the MinimumDistance problem over $\mathbb{F}_2$: given a matrix $\mathbf{A} \in \mathbb{F}_2^{m\times n}$, find a nonzero vector $\mathbf{z} \in \mathbb{F}_2^n$ to minimize $\overline{\text{wt}}(\mathbf{Az})$ where $\overline{\text{wt}}(\mathbf{a})$ is the normalized hamming weight of $\mathbf{a}$. The latter quantity denotes the distance of the code given by the linear span of the columns of $\mathbf{A}$.

Below we restate the hardness of MinimumDistance as proved by Austrin and Khot [AK14] along with an additional guarantee satisfied by their reduction in the YES case which shall prove useful for the subsequent application.

Theorem B.1. [[AK14]] *There is a universal constant $\zeta \in (0, 1/5)$ such that given a matrix $\mathbf{A} \in \mathbb{F}_2^{m\times n}$, it is NP-hard to distinguish between the following:*

YES Case. *There is a vector $\mathbf{z} = (z_1, \ldots, z_n) \in \mathbb{F}_2^n$ such that $z_1 = 1$ and $\overline{\text{wt}}(\mathbf{Az}) \leqslant \zeta$.*

NO Case. *For any nonzero vector $\mathbf{z} \in \mathbb{F}_2^n$, $\overline{\text{wt}}(\mathbf{Az}) \geqslant 5\zeta$.*

The following is an easy consequence of above obtained by tensoring the instance of the above theorem.

Theorem B.2. *There is a universal constant $\zeta \in (0, 1/5)$ such that for any constant positive integer K, given a matrix $\mathbf{A} \in \mathbb{F}_2^{m\times n}$, it is NP-hard to distinguish between the following:*

YES Case. *There is a vector $\mathbf{z} = (z_1, \ldots, z_n) \in \mathbb{F}_2^n$ such that $z_1 = 1$ and $\overline{\text{wt}}(\mathbf{Az}) \leqslant \zeta^K$.*

NO Case. *For any nonzero vector $\mathbf{z} \in \mathbb{F}_2^n$, $\overline{\text{wt}}(\mathbf{Az}) \geqslant 5^K\zeta^K$.*

From the above we obtain – using techniques similar to those used in [Kho06] – the following hardness of distinguishing between an MinimumDistance instance of distance ε vs. $1/2 - \varepsilon$.

Theorem B.3. *For any positive constant $\varepsilon > 0$, given a matrix $\mathbf{B} \in \mathbb{F}_2^{m\times n}$, it is NP-hard to distinguish between the following:*

YES Case. *There is a vector $\mathbf{z} = (z_1, \ldots, z_n) \in \mathbb{F}_2^n$ such that $z_1 = 1$ and $\overline{\text{wt}}(\mathbf{Bz}) \leqslant \varepsilon$.*

NO Case. *For any nonzero vector $\mathbf{z} \in \mathbb{F}_2^n$, $1/2 - \varepsilon \leqslant \overline{\text{wt}}(\mathbf{Bz}) \leqslant 1/2 + \varepsilon$.*

Proof. Let $\mathbf{A} \in \mathbb{F}_2^{m\times n}$ be an instance of MinimumDistance from Theorem B.2 for a large value of K. Let $\mathbf{A}_i, i = 1, \ldots, m$ be the rows of $\mathbf{A}$. Let G be a regular expander on m vertices labeled from $[m]$, with degree $D = \left(\frac{4}{5^K\zeta^K}\right)^{10}$ and second largest eigenvalue $\lambda \leqslant D^{0.9}$, which can be constructed efficiently. Note that $\lambda/D \leqslant (5^K\zeta^K)/4$.

Let $t = 1/(2^K\zeta^K)$ and consider all t length random walks $[i_1, i_2, \ldots, i_t]$ in G. There are $m \cdot D^t$ of such walks. For each such walk we add 2^t rows in $\mathbf{B}$ given by

$$\left\{\sum_{j=1}^{t} s_j A_{i_j} \;\middle|\; s_1, \ldots, s_t \in \mathbb{F}_2\right\}.$$

In total, there are $m' = m \cdot (2D)^t$ rows in $\mathbf{B}$ and n columns.

YES Case

Let $\mathbf{z}$ be the vector given in the YES case of Theorem B.2, such that there are at most ζ^K fraction of rows $\mathbf{A}_i$ satisfying $\langle \mathbf{A}_i, \mathbf{z} \rangle = 1$. For a random walk in G, the probability that it contains an index corresponding to any such row is at most $t\zeta \leqslant 1/2^K$. Thus, $\overline{\mathrm{wt}}(\mathbf{Bz}) \leqslant 1/2^K$.

NO Case

Consider any $\mathbf{z} \in \mathbb{F}_2^n$. From the NO case of Theorem B.2, we have that at least $5^k\zeta^K$ fraction of the rows $\mathbf{A}_i$ satisfy $\langle \mathbf{A}_i, \mathbf{z} \rangle = 1$. Let I be the set of the indices corresponding to these rows. Using the analysis in Section 4.7 of [LW05] we can bound the probability that a t length random walk $[i_1, \ldots, i_t]$ in G does not contain an index from I as follows.

$$\begin{aligned}
\mathbf{Pr}\left[i_1 \notin I, i_2 \notin I, \ldots, i_t \notin I\right] &\leqslant \left(\sqrt{1 - 5^K\zeta^K} + \frac{\lambda}{D}\right)^t \\
&\leqslant \left(\sqrt{1 - 5^K\zeta^K} + \frac{5^K\zeta^K}{4}\right)^t \\
&\leqslant \left(1 - \frac{5^K\zeta^K}{4}\right)^{1/(\zeta^K 2^K)} \\
&\leqslant e^{-(5/2)^K/4} \leqslant 2^{-2^K}.
\end{aligned}$$

Thus, at least $1 - 2^{-2^K}$ fraction of the random walks contain an index from I. For any such walk, exactly half of the 2^t linear combinations of the corresponding rows result in a row $\mathbf{B}_r$ of $\mathbf{B}$ such that $\langle \mathbf{B}_r, \mathbf{z} \rangle = 0$. Thus, $(1/2)(1 - 2^{-2^K}) \leqslant \overline{\mathrm{wt}}(\mathbf{Bz}) \leqslant (1/2)(1 + 2^{-2^K})$. Choosing K to be large enough completes the proof. □

B.1 Main Theorem

The following theorem shows optimal hardness of learning linear forms by degree d polynomials and subsumes the weaker results in [GKS10].

Theorem B.4. *For any constants $\delta > 0$ and positive integer d, given a set of point-value pairs $\{(\mathbf{x}_i, y_i)\}_{i=1}^M \subseteq \mathbb{F}_2^M \times \mathbb{F}_2$, it is NP-hard to distinguish whether*

YES Case. *There is a linear form $\ell^* : \mathbb{F}_2^N \mapsto \mathbb{F}_2$ such that for at least $(1 - \delta)$ fraction of the pairs $(\mathbf{x}, y)$, $\ell^*(\mathbf{x}) = y$.*

NO Case. *Any degree d polynomial $p : \mathbb{F}_2^N \mapsto \mathbb{F}_2$ satisfies $p(\mathbf{x}) = y$ for at most $(1/2 + \delta)$ fraction of the point-value pairs.*

Proof. Let $\mathbf{B} \in \mathbb{F}_2^{m \times n}$ be the matrix given by Theorem B.3 for a parameter $\varepsilon > 0$ to be chosen small enough. For each combination of d rows of $\mathbf{B}$ we add a point-value pair in our instance as follows. For $\mathbf{r} = (r_1, \ldots, r_n)$ obtained by summing some d rows of $\mathbf{B}$ we add $((r_2, \ldots, r_n), r_1)$ to our instance. Thus, $M = m^d$ and $N = n - 1$.

YES Case

Let $\mathbf{z} = (z_1, \ldots, z_n)$ with $z_1 = 1$ be the vector from the YES case of Theorem B.3 satisfying $\langle \mathbf{z}, \mathbf{b} \rangle = 0$ for at least $(1 - \varepsilon)$ fraction of the rows $\mathbf{b}$ of $\mathbf{B}$. Defining $\ell^*(\mathbf{x}) = \sum_{j=1}^{n-1} z_{j+1}x_j$, we obtain that for $(1 - \varepsilon)$ fraction of the rows $(b_1, \ldots, b_n)$ of B, $\ell^*(b_2, \ldots, b_n) = b_1$. Thus, for at least $(1 - \varepsilon d)$ fraction of the point-value pairs $(\mathbf{x}, y)$ of the instance constructed above, $\ell^*(\mathbf{x}) = y$.

NO Case

In the NO case, the uniformly random distribution over the rows of $\mathbf{B}$ fools every linear form $\ell : \mathbb{F}_2^n \mapsto \mathbb{F}_2$ with bias ε. By the result of Viola [Vio09] (Theorem 2.6), the distribution given by uniformly random

d-wise sums of of the rows of $\mathbf{B}$ fools all degree d polynomials $q : \mathbb{F}_2^n \mapsto \mathbb{F}_2$ with bias $\varepsilon_d = 16 \cdot \varepsilon^{1/2^{d-1}}$. Let $(r_1, \ldots, r_n)$ be a random element from the latter distribution, and let $q(r_1, \ldots, r_n) = r_1 + p(r_2, \ldots, r_n)$ where p is a degree d polynomial. Since q is linear in the first coordinate, the bias of q under the uniform distribution is 0, and $p(r_2, \ldots, r_n) \neq r_1$ for at least $1/2 - \varepsilon_d$ fraction of $(r_1, \ldots, r_n)$.

Thus, for any degree d polynomial p, for at least $1/2 - \varepsilon_d$ fraction of the point-value pairs $(\mathbf{x}, y)$ constructed in our instance $p(\mathbf{x}) \neq y$.

To complete the reduction we choose ε to be small enough so that $\max\{\varepsilon d, 16 \cdot \varepsilon^{1/2^{d-1}}\} \leqslant \delta$.

□

ECCC ISSN 1433-8092

http://eccc.hpi-web.de