Finding Correlations in Subquadratic Time, with Applications to Learning Parities and Juntas

Gregory Valiant
UC Berkeley
Berkeley, CA
gregory.valiant@gmail.com

***Abstract*—Given a set of n d-dimensional Boolean vectors with the promise that the vectors are chosen uniformly at random with the exception of two vectors that have Pearson–correlation ρ (Hamming distance $d \cdot \frac{1-\rho}{2}$), how quickly can one find the two correlated vectors? We present an algorithm which, for any constants $\epsilon, \rho > 0$ and $d >> \frac{\log n}{\rho^2}$, finds the correlated pair with high probability, and runs in time $O(n^{\frac{3\omega}{4}+\epsilon}) < O(n^{1.8})$, where $\omega < 2.38$ is the exponent of matrix multiplication. Provided that d is sufficiently large, this runtime can be further reduced. These are the first subquadratic–time algorithms for this problem for which ρ does not appear in the exponent of n, and improves upon $O(n^{2-O(\rho)})$, given by Paturi et al. [15], Locality Sensitive Hashing (LSH) [11] and the Bucketing Codes approach [6].**

Applications and extensions of this basic algorithm yield improved algorithms for several other problems:

Approximate Closest Pair: For any sufficiently small constant $\epsilon > 0$, given n vectors in $\mathbf{R}^d$, our algorithm returns a pair of vectors whose Euclidean distance differs from that of the closest pair by a factor of at most $1+\epsilon$, and runs in time $O(n^{2-\Theta(\sqrt{\epsilon})})$. The best previous algorithms (including LSH) have runtime $O(n^{2-O(\epsilon)})$.

Learning Sparse Parity with Noise: Given samples from an instance of the learning parity with noise problem where each example has length n, the true parity set has size at most $k << n$, and the noise rate is η, our algorithm identifies the set of k indices in time $n^{\frac{\omega+\epsilon}{3}k} poly(\frac{1}{1-2\eta}) < n^{0.8k} poly(\frac{1}{1-2\eta})$. This is the first algorithm with no dependence on η in the exponent of n, aside from the trivial brute-force algorithm.

Learning k-Juntas with Noise: Given uniformly random length n Boolean vectors, together with a label, which is some function of just $k << n$ of the bits, perturbed by noise rate η, return the set of relevant indices. Leveraging the reduction of Feldman et al. [7] our result for learning k-parities implies an algorithm for this problem with runtime $n^{\frac{\omega+\epsilon}{3}k} poly(\frac{1}{1-2\eta}) < n^{0.8k} poly(\frac{1}{1-2\eta})$, which improves on the previous best of $> n^{k(1-\frac{2}{2^k})} poly(\frac{1}{1-2\eta})$, from [8].

Learning k-Juntas without Noise:[1] Our results for learning sparse parities with noise imply an algorithm for learning juntas without noise with runtime $n^{\frac{\omega+\epsilon}{4}k} poly(n) < n^{0.6k} poly(n)$, which improves on the runtime $n^{\frac{\omega+1}{\omega}} poly(n) \approx n^{0.7k} poly(n)$ of Mossel et al. [13].

***Keywords*-Correlation, closest pair, nearest neighbor, locality sensitive hashing, learning parity with noise, learning juntas, metric embedding.**

This work was supported by an IBM PhD Fellowship.

[1] I am grateful to Vitaly Feldman for pointing out that an improved algorithm for learning sparse parity with noise (with a high level of noise) yields an improved algorithm for learning k-juntas *without* noise.

I. Introduction

Given a set of n d-dimensional Boolean vectors with the promise that the vectors are chosen uniformly at random from the Boolean hypercube, with the exception of two vectors that have Pearson–correlation ρ (Hamming distance $d \cdot \frac{1-\rho}{2}$), can one find the correlated pair in subquadratic time? This problem was, apparently, first posed by Leslie Valiant in 1988 as the *Light Bulb Problem* [18]. The first positive solution was provided by Paturi et al. [15], who gave an $n^{2-\Theta(\rho)}$ algorithm. This fundamental problem of finding correlated features is a problem encountered in practice throughout the sciences, and, within computer science theory, arises in various settings central to our field.

Perhaps most obviously, the Light Bulb Problem is a special case of the Boolean *Approximate Closest Pair Problem*: given a set of vectors, how can one quickly find two vectors with near-minimal Hamming distance? Surprisingly, previous algorithms obtained comparable runtimes for the Light Bulb problem and Approximate Closest Pair problem; phrased differently, in contrast to our algorithm, these algorithms do not significantly leverage the randomness in the Light Bulb Problem, and the fact that nearly all the pairwise distances are extremely concentrated near $d/2$.

The Light Bulb problem is also easily recognized as a special case of the problem of learning parity with noise, which we now describe.[2] Suppose one is given access to a sequence of examples (x, y), where $x \in \{-1, +1\}^n$ is chosen uniformly at random, and $y \in \{-1, +1\}$ is set so that $y = z \prod_{j \in S} x_i$, for some fixed, though unknown set $S \subset [n]$, where $z \in \{-1, +1\}$ is chosen to be -1 independently for each example with probability $\eta \in [0, 1/2)$. In the case where the *noise rate* $\eta = 0$, the problem of recovering the set S is easy: given n such examples, with high probability one can recover the set S by Gaussian elimination—translating this problem into a problem over $\mathbf{F}_2$, S is given simply as the solution to a set of linear equations. From an information theory standpoint, the addition of some constant amount of noise ($\eta > 0$) does not change the problem significantly; for

[2] In particular, the Light Bulb problem is the problem of learning parity with noise where the true parity has size $k = 2$. To see one direction of the reduction, given a matrix whose rows consist of examples from a noisy parity instance, if one removes the examples with odd labels, the columns in the remaining matrix will be uniformly random, except with the two correlated columns corresponding to the two indices of the true parity set.

constant $\eta < 1/2$, given $O(n)$ examples, with high probability there will only be one set $S \subset [n]$ where the parities of the corresponding set of indices of the examples are significantly correlated with the labels. From a computational standpoint, the addition of the noise seems to change the problem entirely. In contrast to the simple polynomial-time algorithm for the noise-free case, when given a small constant amount of noise, the best known algorithm, due to Blum et al. [4] takes time $2^{O(\frac{n}{\log n})}$.

This problem of learning parity with noise is, increasingly, understood to be a central problem in learning theory. Additionally, this problem reoccurs in various forms in several areas of theoretical computer science, including coding theory, and cryptography.

Our results for learning parities apply to the setting in which the true parity set S is much smaller than n, say $k := |S| = O(\log n)$. This problem of learning k-parities is especially relevant to Learning Theory, as was revealed by a series of reductions given in work of Feldman et al. [7], showing that the problem of learning k-parities (under the uniform distribution, with random classification noise) is at least as hard as the problems of learning k-juntas (where the labels are an arbitrary function of k bits), learning 2^k-term DNFs from uniformly random examples, and the variants of these problems in which the noise is adversarial (rather than random). The existence of such a reduction should not be surprising: the problem of learning a parity with noise is the problem of finding a heavy low-degree Fourier coefficient, given the promise that one exists; in the case of learning a $k = \log(n)$ sized junta, for example, one knows that there will be at most 2^k significant Fourier coefficients. Intuitively, the presence of more heavy low-degree Fourier coefficients should, if anything, facilitate the task of finding such a coefficient.

A. Techniques

All of our results rely on fast matrix multiplication: our results for the Light Bulb problem and Learning Parity with Noise use the fact that $n \times n$ matrices may be multiplied in time $O(n^\omega)$, for $\omega < 3$. The best bound on ω is due to Virginia Vassilevska Williams [20], who showed that $\omega < 2.372$. Our results for the approximate closest pair rely on fast *rectangular* matrix multiplication; in particular, the fact shown by Coppersmith, that for any $\epsilon > 0$, for $\alpha < 0.29$, the product of an $n \times n^\alpha$ and $n^\alpha \times n$ matrix may be computed in time $O(n^{2+\epsilon})$ [5]. At the end of the paper, we briefly discuss the possibility of obtaining analogous results that do not employ fast matrix multiplication algorithms.

Our results for the approximate closest pair problem also rely on some metric embedding machinery. Roughly, our approach to finding the closest pair will 'bucket' the pairwise inner products of the vectors; if nearly all of the pairwise inner products are small, with the exception of one large inner product (corresponding to the closest pair), then we will be able to discern which bucket contains the large inner product, and very quickly search within that bucket and find the close pair. In the setting in which all vectors are random Boolean vectors, with the exception of a single pair of correlated vectors, the concentration in the pairwise inner products comes for free—almost all pairwise inner products will be very close, with the exception of the one pair of correlated vectors. In the adversarial setting of the general approximate closest pair problem, we will apply a metric embedding to obtain some concentration in the inner products between the images of 'bad' pairs of vectors. In particular, we want an embedding f for which $\langle f(u), f(v)\rangle \approx g_f(\langle u, v\rangle)$, where the function $g_f : \mathbf{R} \rightarrow \mathbf{R}$ would, ideally, map all the small inner products to tiny inner products, while roughly preserving the large inner product (corresponding to the closest pair).

If we use a single embedding for all vectors any improvement over the $n^{2-O(\epsilon)}$ runtime of LSH seems difficult, in light of classical structural results such as Schoenberg's characterization of the functions g_f which are realizable via embeddings of the spherical shell [17].

Instead of a single embedding, we randomly partition the vectors into two sets, and embed the two sets of vectors according to *different* embeddings. We carefully choose these two embeddings f, h such that $\langle f(u), h(v)\rangle \approx g_{f,h}(\langle u, v\rangle)$, where the function $g_{f,h}$ is a Chebyshev polynomial. We choose Chebyshev polynomials for their extremal properties (see Fact 12). This idea of using two embeddings to evade the limitations of using a single embedding was, perhaps, first employed by Alon and Naor [1].

II. Previous Algorithmic Work

In this section we very briefly summarize previous (theoretical) algorithmic results for these problems.

A. Light Bulbs and Close Pairs

The earliest results for the Light Bulb problem are due to Paturi et al. [15], and give a hashing-based algorithm whose runtime is $n^{2-O(\rho)}$, where ρ is the Pearson-correlation of the two correlated vectors. The Locality Sensitive Hashing approach of Indyk and Motwani [11] , and the Bucketing Codes approach of Dubiner [6] also give algorithms that run in time $n^{2-O(\rho)}$. The approach of Dubiner achieves the best constants, with a runtime of $O(n^{2-2\rho})$, in the limit as ρ gets small.

For the more general problem of finding a pair of vectors whose Hamming distance is at most a factor of $(1+\epsilon)$ times that of the distance between the closest pair, the Locality Sensitive Hashing approach [11] achieves runtime $O(n^{1+\frac{1}{1+\epsilon}})$, which is approximately $O(n^{2-\epsilon})$ for small ϵ. Subsequent work on Locality Sensitive Hashing improves this dependence for other metrics—specifically, Andoni and Indyk [2] show that this problem can be solved in time $O(n^{1+\frac{1}{(1+\epsilon)^2}}) \approx O(n^{2-2\epsilon})$ for ℓ_2 distance. For small ϵ, no algorithm achieving runtime $O(n^{2-\Omega(\epsilon)})$ has been previously described.

B. Parities, Juntas, and DNFs

For the general problem of learning noisy parities, Blum et al. give an $2^{O(\frac{n}{\log n})}$ algorithm, based on leveraging the availability of a very large number of samples to perform a structured variant of Gaussian elimination that finds sparse solutions.

For the problem of learning parities of size k, in a recent paper, Grigorescu et al. [8] adapt the approach of Hopper and Blum [9] to the noisy setting to give an algorithm that runs in time $O\left(poly(\frac{1}{1-2\eta})n^{(1+2\eta)^2+o(1))k/2}\right)$. In particular, as the noise rate goes to 0, the performance of this algorithm tends to $O(n^{k/2})$, and as the noise rate tends towards $\frac{1}{2}$, the dependency on n tends towards $O(n^k)$.

For the problem of learning juntas over the uniform distribution, Mossel et al. [13] show that size k juntas can be learned *in the absence of noise*, in time $O(n^{\frac{\omega k}{\omega+1}}) \approx O(n^{.7k})$. The above results of Grigorescu et al., via the reduction of Feldman et al. [7], yields an algorithm for learning k juntas over the uniform distribution with noise η in time

$$> n^{(1-\frac{1}{2^k})k} poly(\frac{1}{1-2\eta}).$$

For constant noise η, no algorithm running in time $O(n^{ck})$ for any constant $c < 1$ has previously been described.

For the problem of (ϵ, δ) PAC-learning s-term DNFs under the uniform distribution, the results of Grigorescu et al. imply a runtime of

$$poly(\log\frac{1}{\delta}, \frac{1}{\epsilon}, s)n^{(1-\tilde{O}(\epsilon/s)+o(1))\log\frac{s}{\epsilon}},$$

which improves upon the $O(n^{\log\frac{s}{\epsilon}})$ of Verbeurgt [19] from 1990.

III. Summary of Results

We begin by stating our main results for the Light Bulb problem, and the Approximate Closest Pair problem. We then give our results for Learning Parity with Noise, and state its corollaries for learning juntas, both with and without noise, and DNFs.

Proposition 1: For any constant $\epsilon > 0$, for sufficiently large n, given n vectors in $\{-1, 1\}^d$ chosen uniformly at random with the exception of a single pair that has inner product at least ρd, provided $d > n^{\frac{1}{4-\omega}}/\rho^2$, the correlated pair of vectors can be found in time

$$O\left(\frac{n^{\frac{5-\omega}{4-\omega}+\epsilon}}{\rho^{2\omega}}\right) < n^{1.62} \cdot poly(1/\rho),$$

where $\omega < 2.38$ is the exponent of matrix multiplication.

We stress that the runtime given in the above proposition differs in nature from those given by previous approaches in that the dependence on the correlation, ρ, has been removed from the exponent of n, yielding significant improvements in the asymptotic performance for small values of ρ.

The following more general theorem applies to any set of vectors for which most of the pairwise inner products are small; this can also be viewed as an algorithm for approximating the product of two matrices, given the promise that their product only has a moderate number of large entries.

Theorem 1: Consider a set of n vectors in $\{-1, 1\}^d$ and constants $\rho, \tau \in [0, 1]$ such that the following condition holds:

- For all but at most $n^{\frac{3}{4}\omega-\frac{1}{2}} \approx n^{1.3}$ pairs u, v of distinct vectors, $|\langle u, v\rangle| \leq \tau d$.

There is an algorithm that, with probability $1 - o(1)$, will output *all* pairs of vectors whose inner product is least ρd. Additionally, the runtime of the algorithm is

$$dn^{\frac{3\omega}{4}} \cdot n^{4\frac{\log\rho}{\log\tau}} \text{ polylog } n \leq O(dn^{1.79+4\frac{\log\rho}{\log\tau}}),$$

where $\omega < 2.38$ is the exponent of matrix multiplication.

The above algorithm, together with elementary Chernoff bounds yields the following corollary for the Light Bulb problem, in the setting in which the dimension, d, is near the information theoretic limit of $O(\frac{\log n}{\rho^2})$.

Corollary 2: For any constant $\rho, \epsilon > 0$, there exists a constant c_ϵ dependent on ϵ such that for sufficiently large n, given an instance of the Light Bulb problem with parameters n, d, ρ with $d \geq c_\epsilon\frac{\log n}{\rho^2}$, with probability $1 - o(1)$, the pair of correlated vectors can be found in time $O(dn^{\frac{3\omega}{4}+\epsilon}) \leq O(dn^{1.79})$.

To obtain our result for the closest pair problem, we extend our approach to the Light Bulb problem in conjunction with a novel "Chebyshev metric embedding". A key step is also an intuitive though nontrivial reduction showing that the problem of finding a pair of vectors whose inner product is within an additive ϵ from the pair with maximal inner product given a set of vectors of unit length, can be used to find the $1 + \epsilon$ (multiplicatively) approximate closest pair of arbitrary vectors. We show the following theorem for the approximate closest pair problem:

Theorem 2: Given n vectors in $\mathbf{R}^d$, an approximation parameter $\epsilon > 0$, and a probability of failure $\delta > 0$, our algorithm returns a pair of vectors u, v such that with probability at least $1 - \delta$, the Euclidean distance $||u - v|| \leq (1 + \epsilon)d^*$, where d^* is the Euclidean distance between the closest pair of vectors. Additionally, the algorithm runs in time

$$\left(n^{2-\Omega(\sqrt{\epsilon})} + nd\right) poly(\log\frac{1}{\delta}, \log n).$$

For an instance of the Closest Pair problem in which the pairwise distances between *most* pairs of vectors are tightly concentrated about some value, our algorithm gives a more efficient runtime than is described by the above theorem.

A. Parities, Juntas, and DNFs

Definition 3: An *example* (x, y) from an (n, k, η)-*instance* of parity with noise, consists of $x \in \{-1, +1\}^n$, chosen uniformly at random, together with a *label* $y \in \{-1, +1\}$ defined by $y = z \cdot \prod_{i\in S} x_i$, where $z \in \{-1, +1\}$ is chosen independently of x to be -1 with probability η, for some fixed set $S \subset [n]$ with $|S| = k$.

Proposition 1 can be used to yield an algorithm for an (n, k, η) instance of parity with noise with runtime $n^{k\frac{5-\omega+\epsilon}{2(4-\omega)}} poly(\frac{1}{1-2\eta}) \approx n^{0.81} poly(\frac{1}{1-2\eta})$. We are able to slightly improve this exponent via a related, though alternate approach:

Theorem 3: For any fixed $\epsilon > 0$, for sufficiently large n and k, given examples from an (n, k, η) instance of parity with noise, with probability $1 - o(1)$, our algorithm will correctly return the true set of k parity bits. Additionally, the algorithm will run in time

$$n^{\frac{\omega+\epsilon}{3}k} poly(\frac{1}{1-2\eta}) < n^{0.80k} poly(\frac{1}{1-2\eta}).$$

The above theorem, via the reductions of Feldman et al. [7] has immediate implications for the problems of learning juntas and DNFs:

Definition 4: An *example* (x, y) from a (n, η)-*instance* of a noisy k-junta consists of $x \in \{-1, +1\}^n$, chosen uniformly at random, together with a *label* $y \in \{-1, +1\}$ defined by $y = z \cdot f(x_S)$, where $z \in \{-1, +1\}$ is chosen independently of x to be -1 with probability η, f is a fixed though unknown function $f : \{-1, +1\}^k \rightarrow \{-1, +1\}$, and x_S denotes the indices of x occurring in a fixed (though unknown) set $S \subset [n]$ with $|S| = k$.

Corollary 5: For sufficiently large n and k given access to examples from an (n, η) instance of a noisy k-junta, with constant probability our algorithm will correctly return the true set of $k' \leq k$ relevant indices, and truth table for the function. Additionally, the algorithm has runtime, and sample complexity bounded by

$$n^{\frac{\omega+\epsilon}{3}k} poly(\frac{1}{1-2\eta}) < n^{0.80k} poly(\frac{1}{1-2\eta}).$$

For learning juntas without noise, by leveraging our improved algorithm for learning sparse parities with noise in place of the brute-force-search component of the algorithm of Mossel et al. [13] (and rebalancing the tradeoff between the search for low degree Fourier coefficients, and a low degree representation as a polynomial over $\mathbf{F}_2$ in their algorithm), we can achieve an improved exponent:

Corollary 6: For sufficiently large n and k given access to examples from an (n, η) instance of a noisy k-junta with $\eta = 0$, with constant probability our algorithm will correctly return the true set of $k' \leq k$ relevant indices, and truth table for the function. Additionally, the algorithm has runtime, and sample complexity bounded by

$$n^{\frac{\omega+\epsilon}{4}k} poly(n) < n^{0.60k} poly(n).$$

Definition 7: An *example* (x, y) from a r-term DNF over n bits under the uniform distribution consists of $x \in \{-1, +1\}^n$, chosen uniformly at random, together with a *label* $y \in \{-1, +1\}$ given by a fixed (though unknown) r-term DNF applied to x.

The following corollary follows from first arguing that an analog of Theorem 3 holds in which the sample complexity has been reduced (Theorem 4), and then applying the reduction of Feldman et al.

Corollary 8: For sufficiently large n and k, there exists an algorithm that (ϵ, δ)–PAC learns r-term DNF formulae over n bits from uniformly random examples that runs in time

$$poly\left(\frac{1}{\delta}, \frac{r}{\epsilon}\right) n^{0.80 \log \frac{r}{\epsilon}},$$

where the logarithm is to the base 2.

IV. The *Expand and Aggregate* Algorithm

For the remainder of this paper, it will prove more convenient to switch from Boolean vectors to vectors with entries in $\{-1, +1\}$, as we will work with inner products (rather than Hamming distances). These two quantities are easily related, as $d_H(u, v) = \frac{d - \langle u', v' \rangle}{2}$, where u' is obtained from the Boolean vector u by replacing each 0 by -1 (and v' is obtained from v analogously).

Given an $m \times n$ matrix X with entries in $\{-1, +1\}$ whose columns are uniformly random, with the exception of two ρ-correlated columns, one naive approach to finding the correlated columns is to simply compute $W = X^t X$, the matrix whose i, jth entry is the inner product between the ith and jth columns of matrix X. With overwhelming probability, the largest off-diagonal entry of W will correspond to the correlated columns, as that entry will have value roughly $m\rho$, whereas all the other off-diagonal entries have expected value 0, and will be tightly concentrated around 0, with standard deviation $\sqrt{m}$. The obvious issue with this approach is that W has n^2 entries, precluding a subquadratic runtime. This remains an issue even if the number of rows, m, is taken to be near the information theoretic limit of $O(\frac{\log n}{\rho^2})$.

Our approach is motivated by the simple observation that if two columns of X are highly correlated, then we can compress X, by simply aggregating sets of columns. If one randomly partitions the n columns into, say, $n^{2/3}$ sets, each of size $n^{1/3}$, and then replaces each set of columns by a single vector, each of whose entries is given by the sum (over the real numbers) of the corresponding entries of the columns in the set, then we have shrunk the size of the matrix from $m \times n$, to an $m \times n^{2/3}$ matrix, Z (that now has integer entries in the range $[-n^{1/3}, n^{1/3}]$). It is still the case that most pairs of columns of Z will be uncorrelated. If, in the likely event that the two original correlated columns are assigned to distinct sets, the two columns of Z to which the two correlated columns contribute, will be *slightly* correlated. Trivially, the expected inner product of these two columns of Z is $O(\rho m)$, whereas the inner product between any two other columns of Z has expected value 0, and variance $O(n^{2/3} m)$. Thus provided $\rho m >> \sqrt{n^{2/3} m}$, and hence $m >> n^{2/3}/\rho^2$, there should be enough data to pick out the correlated columns of matrix Z, by computing $W' = Z^t Z$, and then finding the largest off-diagonal element. This computation of the product of an $n^{2/3} \times n^{2/3}/\rho^2$ matrix with its transpose, via fast matrix multiplication, is relatively cheap, taking time $n^{2\omega/3} poly(1/\rho) < n^{1.6} \cdot poly(1/\rho)$.

Once one knows which two columns of Z contain the original correlated columns of W, one can simply brute-force check all pairs of those columns, which takes time $mn^{2/3}$. (One could also recurse the algorithm on the two relevant sets of $n^{1/3}$ columns, though this would not improve the asymptotic running time.) The computation, now, is dominated by the size of the initial $n^{2/3}/\rho^2 \times n > n^{1.66}$ matrix! It is also clear that the runtime of this algorithm will depend only inverse polynomially on the correlation. Optimizing the tradeoff between the size of the initial matrix, and the time spent computing the product, yields an exponent of $(5-\omega)/(4-\omega) < 1.62$.

We conclude this section by formally describing the algorithm, and giving the basic proof of correctness.

```
VECTOR AGGREGATION
Input: An $m \times n$ matrix $X$ with entries $x_{i,j} \in \{-1,+1\}$, and constant $\alpha \in (0,1]$.
Output: a pair of indices, $c_1, c_2 \in [n]$.
  • Randomly partition $[n]$ into $n^{1-\alpha}$ disjoint subsets, each of size $n^\alpha$, denoting the sets $S_1, \ldots, S_{n^{1-\alpha}}$, and form the $m \times n^{1-\alpha}$ matrix $Z$ with entries $z_{i,j} = \sum_{k \in S_j} x_{i,k}$, where the sum is taken over the reals.
  • Let $W = Z^t Z$, and denote the largest off-diagonal entry by $w_{i,j}$.
  • Using a brute-force search, taking time $O(mn^{2\alpha})$, find and output the pair
        $(c_1, c_2) := argmax_{c_1 \in S_i, c_2 \in S_j} \sum_{k=1}^{m} x_{k,c_1} x_{k,c_2}$
    .
```

The following proposition describes the performance of the above algorithm, and implies Proposition 1:

Proposition 9: For any constant $\epsilon > 0$, setting $\alpha = \frac{1}{2(4-\omega)}$ and $m = n^{2\alpha+\epsilon}/\rho^2$, the algorithm VECTOR-AGGREGATION, when given as input the value α as above and the matrix X whose columns consist of n uniformly random vectors chosen from $\{-1,+1\}^m$, with the exception of a pair of columns having inner product ρm, will output the true set of correlated columns with probability $1-o(1)$, and will run in time

$$O\left(\frac{n^{\frac{5-\omega}{4-\omega}+\epsilon}}{\rho^{2\omega}}\right) < O(n^{1.62}/\rho^{2\omega}).$$

Proof: We first verify the runtime of the algorithm. The input matrix X has size $mn = n^{1+2\alpha+\epsilon}/\rho^2$, and the creation of the matrix Z takes time linear in this size. The only remaining bottleneck is the computation of $W = Z^t Z$, which is the product of a $n^{1-\alpha} \times m$ matrix with its transpose, and hence can be computed in time $\max\left(m^\omega, (n^{1-\alpha}/m)^2 m^\omega\right) < n^{\frac{5-\omega}{4-\omega}}/\rho^{2\omega}$, where the second argument to the max operation is the case that $m < n^{1-\alpha}$.

We now verify the correctness of the algorithm. Assume without loss of generality that the true correlated columns are the first and second columns, and that the two true parity columns contribute to distinct sets (which happens with probability $> 1 - 1/n^\alpha$), and call them sets S_1, S_2, respectively. By a union bound over Chernoff bounds, with probability $1-o(1)$, for all i, j, $|z_{i,j}| \le n^{\alpha/2}\log^2 n$. Additionally, aside from the first and second columns of Z, which correspond to the sets S_1, S_2, all columns are independent, with each value $z_{i,j}$ having expectation 0, and thus by another union bound over Chernoff bounds, with probability $1-o(1)$ each off-diagonal entry of W aside from $w_{1,2}$ and $w_{2,1}$ will have magnitude at most $|z_{i,j}|^2 \cdot \sqrt{m}\log^2 n \le \frac{n^{2\alpha+\epsilon/2}}{\rho}$polylog n.

We now argue that $w_{1,2}$ will be significantly larger than this value. Indeed, $w_{1,2} = \sum_{i \in S_1, j \in S_2} \langle X_i, X_j \rangle$, where X_i denotes the ith column of matrix X, and by the above calculation, the magnitude of the contribution to this sum from all terms other than $\langle X_1, X_2 \rangle$ will be at most $\frac{n^{2\alpha+\epsilon/2}}{\rho}$polylog n, with probability $1-o(1)$. To conclude, by assumption$\langle X_1, X_2 \rangle = \rho m = n^{2\alpha+\epsilon}/\rho$, which dominates $\frac{n^{2\alpha+\epsilon/2}}{\rho}$polylog n for any constant $\epsilon > 0$ and sufficiently large n, and thus $w_{1,2}$ or $w_{2,1}$ will be the largest off-diagonal entries of W with probability $1-o(1)$, in which case the algorithm outputs the correct indices. ■

A. Projecting Up

The results of the previous section show how to solve the Light Bulb problem *provided that the points have dimension* $d \ge n^{1/(4-\omega)}/\rho^2$. What happens if d is quite small? Information theoretically, one should still be able to recover the correlated pair even for $d = O(\frac{\log n}{\rho^2})$. How can one adapt the VECTOR-AGGREGATION approach to the case when d is small? The intuition is that we should first increase the dimension of the points, and then proceed as in the large d case.

Viewing the matrix of the points as an instance of parity with noise where all examples with odd parity label have been removed, to expand the number of rows ('examples'), one can simply XOR together a few of the rows, and create a new data row that is reasonably faithful. In particular, if two columns are completely correlated, then the result of XORing together a number of rows will produce a row for which the values in the two correlated columns will be the same. If the correlation is not 1, but instead ρ, after combining q rows, the corresponding columns will only be ρ^q correlated, as XORing degrades the correlation. Recall, however, that the algorithm of the previous section was extremely noise robust, and thus we can afford to degrade the correlation considerably; for constant ρ, we can certainly take $q = o(\log n)$ without increasing the exponent of n in the runtime.

Note that as we are viewing the vectors as having entries in $\{-1,1\}$, this XORing of sets of rows is simply component-wise multiplication of the rows. Equivalently, it can be seen as replacing each column with a sample of the entries of the qth tensor power of the column.

In the context of learning parity with noise, this expansion approach was fruitfully used by Lyubashevsky [12] to show that given few examples, one can generate new "simulated" examples, that can be used in place of actual examples. In contrast to the current setting, the challenge in that work was arguing that the generated examples are actually information theoretically *indistinguishable* from new examples (with higher noise rate).

In our setting, we do not need any such strong information theoretic guarantees, and hence our results will apply more generally than the random setting of the Light Bulb problem. Our approach only requires some guarantee on the inner products of pairs of columns, which can be given by inductively applying the following trivial lemma:

Lemma 10: Given vectors $u, v, w, z \in \mathbf{R}^d$ with $\langle u, v\rangle = \rho_1 d$ and $\langle w, z\rangle = \rho_2 d$, for i, j chosen uniformly at random from $[d]$,

$$E[(u_i w_j)\cdot(v_i z_j)] = \rho_1\rho_2.$$

Phrased differently, letting $x \in \mathbf{R}^{d^2}$ be the vector whose entries are given by the d^2 entries of the outer-product uw^t, and y is given by the entries of vz^t, then $\langle x, y\rangle = \rho_1\rho_2 d^2$. Elementary concentration bounds show that provided one samples sufficiently many indices of this outer product, the inner product between the sampled vectors will be close to this expected value (normalized by the dimension).

Proof: The proof follows from the independence of i, j, the facts that $E[u_i v_i] = \rho_1, E[w_j z_j] = \rho_2$, and the basic fact that the expectation of the product of independent random variables is the product of their expectations. ■

We now restate Theorem 1, which applies more generally than the Light Bulb problem, and can be viewed as an algorithm for approximating the product of two matrices, given the promise that their product has a small number of large entries.

Theorem 1: Consider a set of n vectors in $\{-1, 1\}^d$ and constants $\rho, \tau \in [0, 1]$ such that the following condition holds:

- For all but at most $n^{\frac{3}{4}\omega - \frac{1}{2}} \approx n^{1.3}$ pairs u, v of distinct vectors, $|\langle u, v\rangle| \le \tau d$.

With probability $1 - o(1)$, the algorithm EXPAND AND AGGREGATE when given as input the matrix of vectors, ρ and τ, will output all pairs of vectors with inner product at least ρd. Additionally, the runtime of the algorithm is

$$dn^{\frac{3\omega}{4}} \cdot n^{4\frac{\log\rho}{\log\tau}} \text{ polylog}\, n \le O(dn^{1.79 + 4\frac{\log\rho}{\log\tau}}),$$

where $\omega < 2.38$ is the exponent of matrix multiplication.

```
EXPAND AND AGGREGATE
Input: An $m \times n$ matrix $X$ with entries $x_{i,j} \in \{-1,+1\}$, $\epsilon \in (0,1)$, and $\rho, \tau \in (0,1)$, with $\rho > \tau$
Output: Two indices $c_1, c_2 \in [n]$.
  • Let $m' = n^{\frac{3}{4} + 2\frac{\log\rho}{\log\tau}} \log^5 n$, and $q = \frac{\log n}{-\log\tau}$.
  • If $m' \ge n$, do the $O(mn^2)$ time brute-force search.
  • Otherwise, we create an $m' \times n$ matrix $Y$ with entries in $\{-1,+1\}$:
      – For each of the $m'$ rows of $Y$, select a list $t_1, \ldots, t_q$ with each $t_i$ selected uniformly at random from $[m]$, and set the $j$th component of the corresponding row to be $\prod_{i=1}^{q} x_{t_i,j}$.
  • Let $c_1, c_2$ be the output of algorithm VECTOR-AGGREGATION on input $Y$ with the parameter $\alpha = \frac{1}{4}$, where the algorithm is modified to brute-force-search for each of the top $n^{\frac{3}{4}\omega - \frac{1}{2}}$ entries of $W$.
```

The intuition of the above algorithm is that the matrix Y resulting from the XOR expansion step has the property that the expected inner product between any two "bad" columns is bounded in magnitude by $m'\tau^q = m'\frac{1}{n}$, and the expected inner product of a "good" pair of vectors will be $m'\rho^q = m' n^{-\frac{\log\rho}{\log\tau}} >> m'\frac{1}{n}$. We now hope to argue that the inner products of the "bad" vectors are closely concentrated about their expectations, in which case the Vector Aggregation step of the algorithm will find a "good" pair of vectors. The minor technical issue is that the entries of matrix Y resulting from the XOR expansion step are not independent. Even if we start with an instance of the Light Bulb problem—while the expansion step, intuitively, can be thought of allowing us to pretend that we were given much more data than we were, the added dimensions are far from independent. This lack of independence harms the exponent slightly—we leverage the randomness of the partitioning of the Vector Aggregation algorithm to obtain slightly worse concentration than we would obtain in the truly random setting. This results in a worse exponent of ≈ 1.79 instead of ≈ 1.62 as in Proposition 9, though this discrepancy can, perhaps, be removed via a tighter analysis.

The proof of Theorem 1 relies on the following simple concentration result for the sum of the entries of a random submatrix of a specified size:

Lemma 11: Given an $s \times s$ matrix X, with entries bounded in magnitude by b, let $S_1, S_2 \subset [s]$ be two sets of size h chosen uniformly at random. Define the random variable $y := \sum_{i \in S_1, j \in S_2} X_{i,j}$. Then

$$Pr\left[|y - E[y]| > b \cdot h^{3/2}\log h\right] = o(1/poly(h)).$$

Proof: First consider selecting set S_1, and then selecting set S_2. Let $z_{S_1} := E[y|S_1]$ denote the expected value of y given the choice of S_1. We now argue that $Pr[|z_{S_1} - E[z_{S_1}]| \ge b \cdot h^{3/2}\log h] = o(1/poly(h))$. To see this, let $p_i = \frac{\sum_{j=1}^{s} x_{j,i}}{s}$ denote the average weight of the ith column, and thus $z_{S_1} = h\sum_{i \in S_1} p_i$. The probability of z_{S_1} deviating from its expectation by more than some value is easily seen to be dominated by the process of choosing the h contributing p_i's with replacement from the set $\{p_1, \ldots, p_s\}$, in which case a standard Chernoff bound applies, yielding that $Pr[|z_{S_1} - E[z_{S_1}]| \ge b \cdot h^{3/2}\log h] < e^{-\Theta(\log^2 h)} = o(1/poly(h))$.

We now argue that, with high probability, the value of y will be closely concentrated around z_{S_1}. In analogy with the above, fixing a set S_1 fixes the average weight of each row of the restriction of matrix X to the columns indexed by elements of S_1. An identical analysis to the above (over the randomness in the choice of S_2 rather than S_1) yields the desired lemma. ■

We now prove Theorem 1.

Proof of Theorem 1: The runtime of EXPAND AND AGGREGATE is dominated by the multiplication of an $n^{3/4} \times m'$ matrix with its transpose, and thus trivially, takes time at most $O\left(n^{\frac{3\omega}{4}} \cdot (\frac{m'}{n^{3/4}})^2\right)$.

To verify the correctness of the algorithm, we first proceed under the assumption that the only pair of columns with inner product greater than τd is the pair with inner product at least ρd. First observe that for a pair of vectors with inner product bounded in magnitude by τd, after the degree q XORing expansion, by Lemma 10, the magnitude of the expected inner product is at most $m' \cdot \tau^q \leq m'/n \leq 1$, which is negligible in comparison to the variance of this quantity. By a union bound over Chernoff bounds, with probability $1 - o(1)$ all such inner products will have magnitude at most $\sqrt{m'} \log n < n^{\frac{3}{8} + \frac{\log \rho}{\log \tau}} \log^{7/2} n := \beta$. Since the inner product of two sums of sets of vectors is simply the sum of the pairwise inner products between the elements of the sets, by Lemma 11, the contribution from these uncorrelated columns to each entry of the product of the aggregated matrix and its transpose—matrix W—calculated in the VECTOR-AGGREGATION stage of the algorithm will be bounded by

$$(n^{1/4})^{3/2} \log n \cdot \beta = n^{\frac{3}{4} + \frac{\log \rho}{\log \tau}} \log^{9/2} n$$

On the other hand, the inner product of the expanded pair of correlated vectors will, with probability $1 - o(1)$, be at least

$$\frac{1}{2} m' \rho^q = \frac{1}{2} m' n^{-\frac{\log \rho}{\log \tau}} = \frac{1}{2} n^{\frac{3}{4} + \frac{\log \rho}{\log \tau}} \log^5 n,$$

which dominates the contribution of the uncorrelated vectors for sufficiently large n.

To conclude, we consider the case that there might be at most $n^{\frac{3}{4}\omega - \frac{1}{2}}$ pairs of columns with inner product $> \tau d$. With probability $1 - o(1)$, all the pairwise correlations between the sets of vectors to which the most correlated pair get grouped will be at most τd. Additionally, as there are at most $n^{\frac{3}{4}\omega - \frac{1}{2}}$ pairs of vectors whose inner products have magnitude greater than τd, there will be at most this many entries of W that are larger than $n^{\frac{3}{4} + \frac{\log \rho}{\log \tau}} \log^{9/2} n$, with probability $1 - o(1)$. Thus one could modify the VECTOR-AGGREGATION algorithm by performing the $O(dn^{1/2})$ time brute-force search for each of the $n^{\frac{3}{4}\omega - \frac{1}{2}}$ largest entries of the matrix W of the VECTOR-AGGREGATION algorithm, taking total time $O(dn^{\frac{3}{4}\omega})$. The probability of success can be boosted by repetition. ■

V. THE CHEBYSHEV EMBEDDING, AND CLOSEST-PAIR PROBLEM

We now abstract and refine the main intuitions behind the EXPAND AND AGGREGATE algorithm, to yield our algorithm for the general approximate closest pair problem. The VECTOR-AGGREGATION algorithm of the previous section relies, crucially, on the tight concentration around 0 of the inner products of the uncorrelated vectors. In the case of Proposition 1, this concentration came "for free", because we assumed that the dimensionality of the data was large $\approx n^{.6}$. To obtain Theorem 1, we needed to work to obtain sufficiently tight concentration. In particular, we performed a metric embedding $f : \{-1,+1\}^d \to \{-1,+1\}^m$, with the crucial property that for an appropriately chosen integer q, for $u, v \in \{-1,+1\}^d$,

$$\frac{\langle f(u), f(v) \rangle}{m} \approx \left(\frac{\langle u, v \rangle}{d} \right)^q .$$

The key property of this mapping $x \to x^q$ is that if one pair of vectors has an inner product that is a factor of $(1+\epsilon)$ larger than than that of any other pair, after performing this mapping, the inner product of the image of this pair will now be a factor of $(1+\epsilon)^q$ larger than that of the images of any other pair of vectors; thus the "gap" has been significantly expanded. Of course, we can not take q to be arbitrarily large, as we would like to maintain a subquadratic amount of data and thus $m << n$, and the variance in the inner products that arises from the subsampling process (choosing which subsets of the rows to XOR) will be $O(m)$. Thus if q is so large that the $O(\sqrt{m})$ standard deviation in the inner product dominates the $m\rho^q$ inner product of the images of the correlated pair, the algorithm will fail.

A simple calculation shows that if we try to obtain an algorithm for the $(1+\epsilon)$ Approximate Closest Pair problem via this EXPAND AND AGGREGATE approach, we would end up with an algorithm with runtime $n^{2-O(\epsilon)}$. Can we do any better? To simplify the exposition, assume that we are told that there is a "good" pair of vectors with inner product at least $(1+\epsilon)d/2$, and that all other pairs of vectors are "bad" and have inner product in the range $[-d/2, d/2]$. In order to improve upon this runtime of $n^{2-O(\epsilon)}$, we need an improved embedding—one that damps the magnitudes of the "bad" pairs of vectors as much as possible, while preserving the inner product between the closest pair. Specifically, we seek a mapping $f_c : \{-1,+1\}^d \to \{-1,+1\}^m$ with the following properties:

- For all $u, v \in \{-1,+1\}^d$, if $\langle u, v \rangle \geq (1+\epsilon)d/2$, then $\langle f_c(u), f_c(v) \rangle \geq c$.
- For all $u, v \in \{-1,+1\}^d$, if $\langle u, v \rangle \in [-d/2, d/2]$, then $\langle f_c(u), f_c(v) \rangle$ is as small as possible.
- For all $u \in \{-1,+1\}^d$, $f_c(u)$ can be computed reasonably efficiently.

The dimension of the image, m, is not especially important, as we could always simply choose a random subset of the dimensions to project onto while roughly preserving the inner products (provided this can all be computed efficiently). In general, it is not clear what the optimal such embedding will be, or how extreme a "gap amplification" we can achieve. In the following section, we show how to construct one family of natural embeddings which allow us to give an algorithm for the $(1+\epsilon)$ Approximate Closest Pairs problem with runtime $n^{2-\Omega(\sqrt{\epsilon})}$.

A. *Embedding via Monic Polynomials*

For the remainder of the chapter, it will prove convenient to consider vectors with unit Euclidean norm; hence the Boolean vectors will be scaled by a factor of $1/\sqrt{d}$. Given a monic degree q polynomial P, with q real roots $r_1, \ldots, r_q \in (-1, 1)$ we wish to give a mapping $f : \mathbf{R}^d \to \mathbf{R}^m$ such that $\langle f(u), f(v) \rangle \approx P(\langle u, v \rangle)$.

Constructing such a mapping in general is not possible: a classical result of Schoenberg from the 1940s [17] characterizes the set of functions $g : \mathbf{R} \to \mathbf{R}$ which have the property that for any d, there exists $f : S^{d-1} \to \mathbf{R}^m$ such that $\langle f(u), f(v)\rangle = g(\langle u, v\rangle)$ for all $u, v \in S^{d-1}$, where S^{d-1} denotes the d-dimensional spherical shell. In particular, he showed that a necessary and sufficient condition for such functions g is that their Taylor expansion about 0 has exclusively nonnegative coefficients (and converges uniformly).

Given that realizing a general degree q polynomial P via a single embedding seems improbably, we now briefly describe how to construct *two* mappings, $f, g : \mathbf{R}^d \to \mathbf{R}^m$ with the property that

$$\langle f(u), g(v)\rangle \approx P(\langle u, v\rangle) \cdot \frac{1}{2^q}.$$

Note that for the purposes of the closest pair problem, such a pair of embeddings are just as good as a single embedding.

Lemma 10 shows that if we can construct such embeddings for the polynomials Q_1 and Q_2, then by simply taking the component-wise products of pairs of rows, we can obtain an embedding for the polynomial $Q(x) = Q_1(x)Q_2(x)$. Thus all that remains is showing that we can construct embeddings for the degree-1 polynomials $Q(x) = \frac{x-r_i}{2}$ for each root r_i of the desired polynomial P.

The mappings for $Q(x) = \frac{x+1}{2}$ is obtained by simply adding d additional dimensions to each vector, populated with $+1$'s, thus sending an inner product of cd to an inner product (in $2d$-dimensional space) of $cd + d = \frac{c+1}{2}(2d)$. Generally, for $Q(x) = \frac{x-r_i}{2}$, the mapping f will simply add d additional dimensions populated by $+1'$s. The mapping g will add d additional dimensions where the first $\frac{1-r_i}{2}d$ dimensions are populated with $+1$s, and the remaining $\frac{1+r_i}{2}d$ dimensions are populated with -1's. Given these mappings, an inner product of cd will yield an inner product of $cd + \frac{1-r_i}{2}d - \frac{1+r_i}{2}d = \frac{c-r_i}{2}(2d)$, as desired. Given this tool, the question is now *which polynomials should we use?*

The following fact suggests an embedding which, at least among a certain class of embeddings, will be optimal.[3]

Fact 12: (see e.g. Thm. 2.37 of [16]) For any $x \notin [-1, 1]$,

$$T_q(x) = \max\left\{|p(x)| : p \in \mathcal{P}_q \text{ and } sup_{y\in[-1,1]}|p(y)| \le 1\right\},$$

where T_q is the degree q Chebyshev polynomial (of the first kind), and $\mathcal{P}$ denotes the set of all degree q polynomials with real coefficients.

Perhaps the most surprising aspect of this fact is that a single polynomial, T_q captures this extremal behavior for all x.

To illustrate the general approach of our algorithm, for the example above in which all the "bad" inner products are in the range $[-d/2, d/2]$, we will construct an embedding corresponding to the monic polynomial $P(x) = T_q(2x)/2^{2q-1}$, where $T_q(x)$ is the qth Chebyshev polynomial (of the first kind). Note that since $T_q(x)$ has q roots, all in the interval $[-1, 1]$, the polynomial $P(x)$ will also have q real roots in the interval $[-1/2, 1/2]$. The corresponding mappings f, g, constructed as described above, will have the property that $\langle f(u), g(v)\rangle = P(\langle u, v\rangle)/2^q$. In particular, we will have the following two properties, the second of which will be the source of the $\sqrt{\epsilon}$ term in the $n^{2-\Omega(\sqrt{\epsilon})}$ runtime of our Approximate Closest Pair algorithm:

- For u, v with $\langle u, v\rangle \in [-d/2, d/2]$, $\langle f(u), f(v)\rangle \le \frac{1}{2^{3q-1}}$.
- For u, v with $\langle u, v\rangle \ge (1+\epsilon)d/2$, $\langle f(u), f(v)\rangle \ge \frac{e^{q\sqrt{\epsilon}}}{2^{3q-1}}$.

We will end up choosing $q = O(\log n)$, and hence the above multiplicative gap of $e^{q\sqrt{\epsilon}} = n^{O(\sqrt{\epsilon})}$, hence we will be able to aggregate sets of $n^{O(\sqrt{\epsilon})}$ vectors. We will ensure that the image of the original vectors have dimension $m < n^{0.29}$, hence the most computationally expensive step of our algorithm will be the computation of the product of an $n^{1-O(\sqrt{\epsilon})} \times m$ matrix and an $m \times n^{1-O(\sqrt{\epsilon})}$ matrix, using fast rectangular matrix multiplication with the following performance guarantee:

Fact 13 (Coppersmith [5]): For any constant $\delta > 0$, provided $\alpha < .29$, the product of an $n \times n^\alpha$ with an $n^\alpha \times n$ matrix can be computed in time $O(n^{2+\delta})$.

There are several technical challenges in realizing our general result for the $(1+\epsilon)$ approximate closest pair problem, particularly in the Euclidean setting. We first show that it suffices to consider the case that all vectors lie on the unit sphere, and then show that it suffices to find a pair of vectors whose inner product is additively within ϵ from that of the pair with maximal inner product. One of the difficulties of this reduction is the case in which most of the vectors are extremely close together ($<< 2^{-n}$). Our algorithm and proof of Theorem 2 are given in the full version.

VI. Learning Sparse Parities with Noise

The problem of finding a ρ-correlated pair of Boolean vectors from among n random vectors is easily seen to be equivalent to solving the Learning Parity with Noise problem, in the special case that the size of the true parity is $k = 2$; the correspondence between the correlation ρ and noise rate η is given by $\eta = 1/2 - \rho/2$. To see one direction of the equivalence, note that given an instance of such a parity with noise problem, if one removes all examples that have label 1, one will be left with a set of examples in which the two true parity indices are correlated. One could thus use the algorithm of Theorem 1 to find the pair of parity indices in time $n^{1.62}poly(\frac{1}{1/2-\eta})$.

In general, given an algorithm for solving the parity with noise problem for parities of some fixed size c in time $O(n^\alpha)$, one may attempt to adapt it to obtain an algorithm for the parity with noise problem for parities of any value k that runs in time $O(n^{k\frac{\alpha}{c}})$ by performing the following transformation: for each length n example with label ℓ, transform it into

[3]We note that better constants would be obtained by replacing Chebyshev polynomials of the first kind, with Chebyshev polynomials of the second kind, as we want to minimize the ℓ_1 norm of the inner products of the images of the "bad" vectors, rather than the ℓ_∞ norm, though the difference is a small constant, and the analysis is easier in the case of Chebyshev polynomials of the first kind.

a length $N = \binom{n}{k/c} \approx n^{k/c}$ example, where each index represents the XOR of some set of k/c of the indices of the original example. If the original set of examples contained a set of k indices whose XOR is correlated with the labels, then the transformed examples will contain (several) sets of c indices whose XOR is correlated with the labels. One can now simply apply the original algorithms for finding parities of size c to the transformed set of examples, to yield a runtime of $O\left((n^{k/c})^{\alpha}\right)$. The one minor difficulty, of course, is that the transformed examples are no longer uniformly random bit strings, though most algorithms should be robust to the type of dependencies that are introduced by this transformation.

The above transformation motivates the search for improved algorithms for finding small constant–sized parities ($k = 2, 3, 4, \ldots$.) Given the existence of a subquadratic time algorithm for the case $k = 2$, a natural hope is that one can design better and better algorithms for larger k, perhaps with the eventual hope of yielding an $n^{o(k)}$ algorithm. In the remainder of this section, we describe an algorithm for the case $k = 3$, with runtime $n^{\omega+\epsilon} poly(\frac{1}{1/2-\eta})$, which yields an algorithm for parities of size k with runtime $n^{\frac{\omega+\epsilon}{3}k} poly(\frac{1}{1/2-\eta}) \approx n^{0.80k} poly(\frac{1}{1/2-\eta})$. While the constant in the exponent is only ≈ 0.02 better than what is yielded by Theorem 1, this slightly different approach may be of independent interest.

Our $k = 3$ algorithm is also based on using fast matrix multiplication to find a pair of correlated vectors, though does not require any column aggregation. The crux of the approach is that a parity function has reasonably heavy low-degree Fourier coefficients if one changes from the uniform distribution over the Boolean hypercube to a slightly biased product distribution. The required bias is very small, thereby allowing one to efficiently subsample the uniform examples so as to produce a distribution with the desired bias. In the remainder of this section we describe the main idea of the algorithm.

A. A Little Bias Goes a Long Way

Given an example x, y from an instance of learning parity with noise with three parity bits, with $x \in \{-1, +1\}^n$ and $y \in \{-1, +1\}$, for any i, $Pr[x_i = 1|y = 1] = 1/2$. Similarly, $Pr[x_i x_j = 1|y = 1] = 1/2$ for distinct $i, j \in [n]$. The improved algorithm for finding parities rests on the following observation about parity sets of size 3: if the bits of x are not chosen uniformly at random, but instead are chosen independently to be 1 with probability $\frac{1}{2} + \alpha$, for some small bias α, then the above situation no longer holds. In such a setting, it is still the case that $Pr[x_i = 1|y = 1] \approx \frac{1}{2} + \alpha$, irrespective of whether i is in the true parity set or not. However,

$$Pr[x_i x_j = 1|y = 1] = \begin{cases} \frac{1}{2} + \Theta(\alpha) & \text{if } i \text{ and } j \text{ in parity set,} \\ \frac{1}{2} + \Theta(\alpha^2) & \text{if } i \text{ or } j \text{ not in parity set.} \end{cases}$$

The punchline of the above discrepancy is that very small biases—even a bias of $\alpha = 1/\sqrt{n}$ can be quite helpful. Given such a bias, for any pair $i, j \in [n]$, for sufficiently large n, even $n^{1.01}$ examples will be sufficient to determine whether i and j are both in the parity set by simply measuring the correlation between the ith and jth indices for examples with odd label, namely estimating $Pr[x_i x_j|y = 1]$ based on the examples. How does one compute these $\binom{n}{2}$ correlations in time $o(n^3)$? By (fast) matrix multiplication. It is worth stressing that, provided this argument is sound, the resulting algorithm will be extremely noise-robust, since the discrepancy between $Pr[x_i x_j|y = 1]$ in the cases that i, j are both parity bits and the case that they are not, will degrade linearly as $\eta \to 1/2$.

It should now be intuitively clear how to extend this approach from the small-biased setting to the setting in which the examples are generated uniformly at random, since a bias of $1/\sqrt{n}$ is quite modest. In particular, with constant probability, a random length–n example will have at least $\frac{n}{2} + \sqrt{n}$ positive indices, thus simply filtering the examples by removing those with fewer than $n/2$ positive indices should be sufficient to instill the necessary bias (at the minor expense of independence). To simplify the proof, we instead argue that we can subsample the examples in such a way that the resulting set of examples is information theoretically indistinguishable from a set of examples chosen according to the $1/\sqrt{n}$-biased product distribution.

In order to obtain the corollary for learning DNF formulae from random examples, via the reduction of Feldman et al. [7], we also need to reduce the sample complexity. We accomplish this by employing the XORing trick to simulate extra examples, and leverage the "Leftover Hash Lemma" of Impagliazzo and Zuckerman [10], as was done by Lyubashevsky [12]. In our setting, the analysis is slightly more delicate, as we cannot XOR together especially large sets of examples without degrading the correlation that we are hoping to detect, and thus we are not able to argue that the larger set of constructed examples is information theoretically indistinguishable from a set of actual examples (with higher noise rate). Nevertheless, we show that our algorithm can tolerate the dependencies that this process introduces. We state the theorem, and defer the full description of the algorithm and proof of correctness to the full version.

Theorem 4: For any fixed $\epsilon > 0$, for sufficiently large n and k, examples from an (n, k, η) instance of parity with noise, with probability $1 - o(1)$, our algorithm will correctly return the true set of k parity bits. Additionally, the algorithm will run in time

$$n^{\frac{\omega+\epsilon}{3}k} poly(\frac{1}{1-2\eta}) < n^{0.80k} poly(\frac{1}{1-2\eta}),$$

and requires at most $n \cdot poly(\frac{k \log n}{1-2\eta})$ examples.

VII. Further Directions: Beyond Fast Matrix Multiplication

Beyond the more obvious open questions posed by this work, one very relevant direction for future work is to give practical algorithms with subquadratic asymptotic runtimes. For example:

Does there exist an algorithm for finding a pair of 0.05*-correlated Boolean vectors from among* $n = 100,000$ *uniformly random Boolean vectors that significantly beats brute-force-search,* in practice*?*

There are two natural approaches to this question. The first is to try to improve practical fast matrix multiplication implementations. While the algorithms of this work rely on fast matrix multiplication, they do not require an especially accurate multiplication. In particular, our algorithms would still succeed if they used a noisy matrix multiplication, or even an algorithm that "misplaced" a constant fraction of the cross-terms. (For example, for $n \times n$ matrices A, B, C, in computing $AB = C$, the entry $c_{i,j}$ should be the sum of n cross terms $a_{i,k} \cdot b_{k,j}$; our algorithms would be fine if only, say, half of these cross terms ended up contributing to $c_{i,j}$.) Tolerating such "sloppiness" seems unlikely to allow for faster asymptotic bounds on the runtime (at least within the Coppersmith–Winograd framework), though it may significantly reduce the overhead on some of the more practically expensive components of the Coppersmith-Winograd framework.

The second approach to yielding a practical algorithm would be to avoid fast matrix multiplication entirely. Our Expand and Aggregate algorithm seems natural (if many pairwise inner products are extremely small, we should "bucket" them in such a way that we can process them in bulk, yet still be able to detect which bucket contains the large inner product). Nevertheless, if one replaces the fast matrix multiplication step with the naive quadratic-time multiplication, one gets no improvement over the quadratic brute-force search. It seems that no clever bucketing schemes (in the "aggregation" step, one need not simply add the vectors over the reals...), or fancy embeddings can remove the need for fast matrix multiplication.

One intuitive explanation for the difficulty of avoiding fast matrix multiplication is via the connection between finding correlations, and learning parity with noise. The statistical query (SQ) lower bound of Blum et al. [3], informally, implies that any algorithm that will beat brute-force-search must be highly non-SQ; in particular, it must perform nontrivial operations that intertwine at least $\log n$ rows of the matrix whose columns are the given vectors. Fast matrix multiplication is clearly such an algorithm.

Given this intuitive need for a non-SQ algorithm, perhaps the most likely candidate for an off-the-shelf algorithm that might replace fast matrix multiplication, is the Fast Fourier Transform. In a recent paper, Pagh gives an extremely clean (and practically viable) algorithm for computing [or approximating] the product of two matrices given the promise that their product is *sparse* [or has small Frobenius norm after one removes a small number of large entries] [14]. The algorithmic core of Pagh's approach is the computation of a Fourier transform. Perplexingly, despite the fact that Pagh's results specifically apply to the type of matrix products that we require in the above algorithms, they seem unable to yield subquadratic algorithms for our problems.

Acknowledgements

I am very grateful to Vitaly Feldman for pointing out that algorithms for learning sparse parities with noise can be used to improve the runtime of the algorithm of Mossel et al. [13] for learning k-juntas *without* noise, which yields Corollary 6. I am also grateful to Alexandr Andoni for discussing the Approximate Closest Pair problem with me, and to Vitaly Feldman, Adam Kalai, Aleksander Madry, and Paul Valiant for discussing the Parity with Noise problem with me over the past few years.

REFERENCES

[1] N. Alon and A. Naor. Approximating the cut-norm via Grothendiecks inequality. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 72–80, 2004.

[2] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 459–468, 2006.

[3] A. Blum, M. Furst, J. Jackson, M. Kearns, Y. Mansour, and S. Rudich. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 253–262, 1994.

[4] A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM (JACM)*, 50(4):507–519, 2003.

[5] D. Coppersmith. Rectangular matrix multiplication revisited. *Journal of Complexity*, 13(1):42–49, 1997.

[6] M. Dubiner. Bucketing coding and information theory for the statistical high dimensional nearest neighbor problem. *CoRR*, abs/0810.4182, 2008.

[7] V. Feldman, P. Gopalan, S. Khot, and A. Ponnuswami. New results for learning noisy parities and halfspaces. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2006.

[8] E. Grigorescu, L. Reyzin, and S. Vempala. On noise-tolerant learning of sparse parities and related problems. In *The 22nd International Conference on Algorithmic Learning Theory (ALT)*, 2011.

[9] N. J. Hopper and A. Blum. Secure human identification protocols. In *ASIACRYPT*, pages 52–66, 2001.

[10] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 248–253, 1989.

[11] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, 1998.

[12] V. Lyubashevsky. The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In *RANDOM*, pages 378–389, 2005.

[13] E. Mossel, R. O'Donnell, and R. Servedio. Learning functions of k relevant variables. *Journal of Computer and System Sciences*, 69(3):421–434, 2004.

[14] R. Pagh. Compressed matrix multiplication. In *"Innovations in Theoretical Computer Science (ITCS)"*, 2012.

[15] Ramamohan Paturi, Sanguthevar Rajasekaran, and John H. Reif. The light bulb problem. In *Conference on Learning Theory (COLT)*, pages 261–268, 1989.

[16] T.J. Rivlin. *The Chebyshev Polynomials*. John Wiley and Sons, 1974.

[17] I.J. Schoenberg. Positive definite functions on spheres. *Duke Mathematical Journal*, 9(1):96–108, 1942.

[18] L. Valiant. Functionality in neural nets. In *First Workshop on Computational Learning Theory*, pages 28–39, 1988.

[19] K. A. Verbeurgt. Learning DNF under the uniform distribution in quasipolynomial time. In *Conference on Learning Theory (COLT)*, pages 314–326, 1990.

[20] V. Vassilevska Williams. Multiplying matrices faster than Coppersmith–Winograd. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, 2012.