Electronic Colloquium on Computational Complexity, Revision 2 of Report No. 6 (2012)

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

Gregory Valiant

February 4, 2012

Abstract

Given a set of n random d-dimensional boolean vectors with the promise that two of them are ρ-correlated with each other, how quickly can one find the two correlated vectors? We present a surprising and simple algorithm which, for any constant $\epsilon > 0$ runs in (expected) time $dn^{\frac{3\omega}{4}+\epsilon}poly(\frac{1}{\rho}) < dn^{1.8} \cdot poly(\frac{1}{\rho})$, where $\omega < 2.4$ is the exponent of matrix multiplication. This is the first subquadratic-time algorithm for this problem with polynomial dependence on $\frac{1}{\rho}$, and improves upon $O(dn^{2-O(\rho)})$, given by the Locality Sensitive Hashing approach of [4] and Bucketing Codes approach of [2].

This basic algorithm also applies to the adversarial setting, and extensions of this algorithm yield improved algorithms for several other problems:

Learning Sparse Parities with Noise: Given samples from an instance of the learning parities 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 $n^{\frac{5-\omega+\epsilon}{8-2\omega}k}poly(\frac{1}{1-2\eta}) < n^{.82k}poly(\frac{1}{1-2\eta})$. This is the first algorithm with a polynomial dependence on $\frac{1}{1-2\eta}$, aside from the trivial $O\left(\binom{n}{k}\right) \approx O(n^k)$ brute-force algorithm, and for large noise rates, improves upon the results of Grigorescu et al. [5] that give a runtime of $n^{(1+(2\eta)^2+o(1))\frac{k}{2}}poly(\frac{1}{1-2\eta})$.

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. [3] our result for learning k-parities implies an algorithm for this problem with runtime $n^{.82k}poly(2^k, \frac{1}{1-2\eta})$, which improves on the previous best of $> n^{k(1-\frac{2}{2^k})}poly(2^k, \frac{1}{1-2\eta})$, from [5].

Learning k-Juntas without Noise: [1] Given uniformly random length n boolean vectors, together with a label, which is some function of $k << n$ of the bits, return the set of relevant indices. Using a modification of the algorithm of Mossel et al. [9], employing our algorithm for learning sparse parities with noise via the reduction of Felman et al [3], we obtain an algorithm for this problem with runtime $n^{\frac{\omega(\omega-5)+\epsilon}{\omega(2\omega-7)-5}k}poly(2^k, n) < n^{.61k}poly(2^k, n)$, which improves on the previous best of $n^{.71k}poly(2^k, n)$ of Mossel et al. [9].

We stress that these runtimes differ in nature from those given by previous approaches in that the dependence of the runtimes on the correlation has been removed from the exponent of n, thereby giving nontrivial new exponents for wide ranges of correlations/noise. For this reason, in this preliminary version we favor clarity of intuition and ease of proof over optimization of the exponent; in the full version, we will give more elaborate algorithms, using more involved analyses, yielding improved exponents for all of these problems. Additionally, in the full version we will describe nontrivial extensions of these results to several other settings.

[1] I am grateful to Vitaly Feldman for drawing my attention to this corollary—that an improved algorithm for learning sparse noisy parities (with a high level of noise) yields an improved algorithm for learning k-juntas *without* noise.

ISSN 1433-8092

1 Introduction

Suppose one is given a set of n d-dimensional boolean vectors with the guarantee that all of them were chosen independently at random, with the exception of two vectors that were chosen so as to be ρ-correlated with each other, is there a subquadratic time algorithm for finding the two correlated vectors? This problem was apparently first posed by Leslie Valiant in 1988 as the Light Bulb Problem [11]. The first positive solution was provided by Paturi et al. [10]. Subsequently this problem has arisen in various settings central to our field.

Perhaps most obviously, this setting is a special case of the boolean 'All-Pairs Closest Pair' problem: given a set of vectors, how can one quickly find two vectors with near-minimal Hamming distance? The relationship between these problems, and motivation for considering the Light Bulb problem is strengthened by the fact that all proposed algorithms for the Light Bulb problem (the current one included), do not rely, fundamentally, on strong randomness, and can be adapted to perform comparably in instances where they are given an (adversarially chosen) set of vectors.

The previous best performances for this problem were given by the theoretically pleasing (and practically relevant) Locality Sensitive Hashing approach in the Hamming cube given by Gionis et al. [4], and the more recent improvement of Dubiner [2]. The Locality Sensitive Hashing approach proceeds by projecting the given vectors to a smaller-dimensional space, and then cleverly hashing so as to maximize the probability that close vectors get hashed to the same value.

Our algorithm, the *Expand and Aggregate* algorithm, takes the complete *opposite* approach. Rather than reducing the dimensionality of the vectors, we instead carefully project the vectors into a higher dimensional space. This transformation has the simple effect of exaggerating correlations: points that were not too correlated in the original space will be driven apart, and will be essentially uncorrelated in the high-dimensional space. In contrast, points that were correlated in the original space will be spread apart to a much lesser extent.

1.1 Learning Parities, Juntas, and DNFs

The Light Bulb problem is also easily recognizable a special case of the learning noisy parities problem, 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 $y = \prod_{j \in S} x_i$, for some fixed, though unknown set $S \subset [n]$. 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.

Now consider adding some noise to the labels: set $y = z \prod_{j \in S} x_i$, where $z \in \{-1, +1\}$ is chosen to be -1 independently for each example with probability $\eta \in (0, \frac{1}{2})$. This is the problem of *learning noisy parities*. From an information theory standpoint, the addition of the noise does not change the problem significantly; for constant η, 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 takes time $2^{\frac{n}{\log n}}$ [1].

This problem of learning noisy parities is, increasingly, understood to be a central problem in

[2]In particular, the Light Bulb problem is the problem of learning noisy parities, where the true parity has size 2. To see one direction of the reduction, given a matrix whose rows consist of samples from a noisy parity instance, if one simply throws out the data points that have an odd label, the columns in the remaining matrix will be uniformly random, except with the two columns corresponding to the indices of the true parity being correlated. The reduction in the other direction is only slightly more complicated.

learning theory, with implications for the problems of learning juntas, and learning DNFs. We outline these specific connections in the following section. Additionally, beyond its relevance to learning theory, this problem reoccurs in various forms in several areas of theoretical computer science, including coding theory and cryptography.

1.2 Sparse Parities, Juntas, and DNFs

Our results 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. [3], 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 main intuition, and proof approach of [3] is that the problem of learning parities with noise is the problem of finding a heavy 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. The reduction proceeds by essentially peppering the labels with random XORs, so that after the peppering process, with some decent probability, exactly *one* fourier coefficient will have survived, in which case the problem has been successfully transformed into the problem of learning a noisy parity of size k. It is worth stressing that this reduction results in a noisy parity instance with a very large noise rate—noise $\frac{1}{2} - \frac{1}{2^k}$, thus highlighting the importance of considering the learning noisy parities problem with noise-rates that approach $1/2$. We conclude this section with formal statements of these reductions.

Theorem A-1 (Feldman et al. [3])**.** *Given an algorithm that learns noisy k-parities on length n strings (under the uniform distribution) with noise rate $\eta \in [0, \frac{1}{2})$ that runs in time $T(n, k, \eta)$, there exists an algorithm that learns k-juntas under the uniform distribution with noise rate η' that runs in time*

$$O\left(k2^{2k} \cdot T(n, k, \frac{1}{2} - \frac{1-2\eta'}{2^k})\right).$$

Theorem A-2 (Feldman et al. [3])**.** *Given an algorithm that learns noisy k-parities on length n strings (under the uniform distribution) with noise rate $\eta \in [0, \frac{1}{2})$ that takes $S(n, k, \eta)$ samples and runs in time $T(n, k, \eta)$, there exists an algorithm that (ϵ, δ)–PAC learns r-term DNF formulae under the uniform distribution that runs in time*

$$\tilde{O}\left(\frac{r^4}{\epsilon^2} \cdot T\left(n, \log\left(\tilde{O}(r/\epsilon)\right), \frac{1}{2} - \tilde{O}(\epsilon/r)\right)\right) \cdot S\left(n, k, \log\left(\tilde{O}(r/\epsilon)\right), \frac{1}{2} - \tilde{O}(\epsilon/r)\right)^2\Bigg).$$

Additionally, as Feldman observed, an improved algorithm for learning noisy k-parities can be used, via the reduction of Feldman et al. [3] to yield an improvement in runtime of the approach of Mossel et al [9] for the problem of learning k-juntas *without* noise. The key observation of Mossel et al. is that either a k-junta has a heavy Fourier coefficient of degree at most d, or, when represented as a polynomial over $\mathbf{F}_2$, has degree at most $k-d$. Their algorithm proceeds by brute force-searching for a heavy Fourier coefficients of order at most αk for some appropriately chosen α; if none are found, then the junta is found by solving a linear system over $n^{(1-\alpha)k}$ variables. Given an improved algorithm for learning noisy parities, using the reduction of Feldman et al., one improve upon the brute-force search component of the algorithm of Mossel et al. The following corollary quantifies this improvement.

Corollary 1. *Given an algorithm that learns noisy j-parities on length n strings (under the uniform distribution) with noise rate $\eta \in [0, \frac{1}{2})$ that runs in time $T(n, j, \eta)$, for any $\alpha \in (0, 1)$, there exists an algorithm that learns k-juntas under the uniform distribution in time*

$$\max\left(T(n, \alpha k, \frac{1}{2} - \frac{1}{2^{\alpha k}}), n^{\omega k(1-\alpha)}\right) poly(2^k, n).$$

2 Previous Algorithmic Results

In this section we very briefly summarize previous algorithmic results for these problems. A more detailed discussion of prior work is deferred to the full version.

2.1 Light Bulbs and Finding Correlated Pairs

The earliest results for the Light Bulb problem are due to Paturi et al. [10], and give a hashing-based algorithm whose runtime is roughly $O(n^{2-\rho})$, where ρ is the correlation of the two correlated vectors. The Locality Sensitive Hashing approach of Gionis et al. [4] improves upon this for correlations close to 1, though also yields an algorithm that runs in time $O(dn^{2-O(\rho)})$. The Bucketing Codes approach of Dubiner [2] also yields $O(dn^{2-O(\rho)})$.

Similarly, in the setting in which the vectors are not random, for the general closest-pair problem: given n points in $\{-1, +1\}^d$, finding a pair whose inner product is within ρd from that of the closest pair of points can be accomplished in time $O(dn^{2-O(\rho)})$.

2.2 Parities, Juntas, and DNFs

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

For the problem of learning noisy of size parities of size k, in a recent paper, Grigorescu et al. [5] adapt the approach of Hopper and Blum [6] 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. [9] 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 Theorem A-1, yields an algorithm for learning k juntas over the uniform distribution with noise η in time

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

In particular, for constant noise η, no algorithm running in time $O(n^{ck})$ for any constant $c < 1$ was previously known.

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 [12] from 1990.

3 Summary of Results

We begin by formally defining the problem, and then state our main theorem for the closest pair problem, and its corollaries for learning parities, juntas and DNFs. Again, we reiterate that in the full version of this paper, we will give improved exponents for all of these problems.

Theorem. *For any fixed $\epsilon > 0$, and $d > \frac{\log^{1+\epsilon} n}{\rho^2}$, given n uniformly random vectors in $\{-1,1\}^d$, with the exception of a pair of vectors that is ρ-correlated, with probability $1-o(1)$ our algorithm returns the correlated pair, and runs in time*

$$dn^{\frac{3\omega}{4}+\epsilon} poly(1/\rho) < dn^{1.8} poly(1/\rho),$$

where $\omega < 2.4$ is the matrix multiplication exponent.

This result does not depend crucially on the randomness of the vectors. We improve the exponent, and prove analogous statements for the adversarial model, and in the context of the general 'closest-pair' problem in the full version of this paper.

Definition 2. *An* example (x, y) *from an* (n, k, η)-instance *of noisy parity, 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 $|S| = k$.*

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

$$n^{k\frac{5-\omega+\epsilon}{8-2\omega}} poly(\frac{1}{1-2\eta}) < n^{.82k} poly(\frac{1}{1-2\eta}),$$

where $\omega < 2.4$ is the matrix multiplication exponent.

The above theorem has immediate implications for the problems of learning juntas and DNFs:

Definition 3. *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 $f : \{-1,+1\}^k \to \{-1,+1\}$, and x_S denotes the indices of x occurring in a fixed (though unknown) set $S \subset [n]$ with $|S| = k$.*

The above theorem together with Theorem A-1 immediately imply the following corollary:

Corollary 4. *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*

$$n^{.82k} poly(2^k, \frac{1}{1-2\eta}).$$

The above theorem together with Corollary 1 yields the following corollary for learning juntas without noise, where the exponent is obtained by selecting the value of α in the statement of Corollary 1 so as to equate the two arguments of the max operation:

Corollary 5. *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 $k' \le k$ relevant indices, and truth table for the function. Additionally, the algorithm has runtime, and sample complexity bounded by*

$$n^{\frac{\omega(\omega-5)+\epsilon}{\omega(2\omega-7)-5}k} poly(2^k, n) < n^{.61k} poly(2^k, n).$$

Definition 6. *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.*

To obtain the following corollary one must show that the one can still perform our algorithm given a polynomial number of examples. This is not difficult, and the details are deferred to the full version.

Corollary 7. *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*

$$poly\left(\frac{1}{\delta}, \frac{r}{\epsilon}\right) n^{.82 \log \frac{r}{\epsilon}},$$

where the logarithm is to the base 2.

3.1 Discussion

A discussion of the implications of these results for other problems (closest-pair, nearest-neighbor search, compressed matrix multiplication, etc.) will be given in the full version of the paper.

4 The Vector-Aggregation Algorithm

Given an $m \times n$ matrix X with entries in $\{-1, +1\}$ whose columns are uniformly random, with the exception of two correlated columns, one obvious approach to finding the correlated columns is 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 entry of W will correspond to the correlated columns. The obvious issue with this approach is that W has n^2 entries. This remains an issue even if the number of rows, m is taken to be near the information theoretic limit of $O(\log n)$, assuming constant correlation.

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. It is still the case that most pairs of columns of Z will be uncorrelated. If, in the likely event that the two original columns are assigned to distinct sets, the two columns of Z to which the two correlated columns contribute, will be *slightly* correlated. It is not hard to show that this correlation will be $O(1/n^{1/3})$, and thus provided $m = n^{2/3+\epsilon}$, for any constant ϵ, 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 element. This computation, via fast matrix multiplication, is relatively cheap, taking time $O(n^{2\omega/3}) < O(n^{1.6})$. Once one knows which two columns of Z contain the correlated columns of W, one can simply brute-force check all pairs of those columns, in time $mn^{2/3}$. The computation, now, is dominated by the size of the initial $n^{2/3+\epsilon} \times n \approx n^{1.66}$ 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 $(5-\omega)/(4-\omega) < 1.62$.

In the rest of this section we formally describe the algorithm, and give the simple proof of correctness.

```
AGGREGATE ROWS
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 $Y$ with entries $y_{i,j} = \sum_{k \in S_j} x_{i,k}$, where the sum
      is taken over the reals.
    • Create the $m \times n^{1-\alpha}$ matrix $Z$, where $z_{i,j} := 1$ if $x_{i,j} \geq 0$, and $z_{i,j} = -1$ if $x_{i,j} < 0$.
    • 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 the pair
          $$(c_1, c_2) := argmax_{c_1 \in S_1, c_2 \in S_2} \sum_{k=1}^{m} x_{k,c_1} x_{k,c_2}$$
      .
    • Output:  $c_1, c_2$.
```

Proposition 8. *For any constant* $\epsilon > 0$*, setting* $m = n^{2\alpha+\epsilon}\frac{1}{(1/2-\eta)^{2+\epsilon}}$*, the algorithm* VECTOR-AGGREGATION*, when given as input the matrix* X *consisting of* m *positively labeled examples from an* $(n, 2, \eta)$*-instance of noisy parity, and parameter* $\alpha \in (0, 1]$*, will run in time*

$$O\left(\max\left(\frac{n^{1+2\alpha+\epsilon}}{(1/2-\eta)^{2+\epsilon}}, \frac{n^{2(1-(3-\omega)\alpha+\epsilon)}}{(1/2-\eta)^{(2+\omega)(2+\epsilon)}}, \frac{n^{\omega(2\alpha+\epsilon)}}{(1/2-\eta)^{\omega(2+\epsilon)}}\right)\right)$$

and return the correct indices with probability $1 - o(1)$*, where* $\omega < 2.4$ *is the exponent of matrix multiplication. Setting* $\alpha = .31$ *and taking* ϵ *small yields a runtime bounded by*

$$O(\frac{n^{1.63}}{(1/2-\eta)^9}).$$

Proof. The input matrix X has size $mn = n^{1+2\alpha+\epsilon}\frac{1}{(1/2-\eta)^{2+\epsilon}}$, and the creation of the matrix Y takes time linear in this size. The only remaining bottleneck is the computation of $W = Z^t Z$. If $m < n^{1-\alpha}$, then this product can be computed by performing $\left(n^{1-\alpha}/m\right)^2$ fast square matrix multiplications, each taking time $O(m^{\omega}) = O(\frac{n^{\omega(2\alpha+\epsilon)}}{(1/2-\eta)^{\omega(2+\epsilon)}}$, and thus (ignoring the ϵ for the sake of clarity) the desired product W is computed in time

$$O\left(\frac{n^{2(1-3\alpha+\omega\alpha)}}{(1/2-\eta)^{2(\omega+2)}}\right).$$

In the case that $m > n^{1-\alpha}$, we crudely bound the time to compute W by $O(m^{\omega})$, which is the third argument to the *max* in the statement of the theorem.

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 with probability $1 - o(1)$, the two true parity columns contribute to distinct sets, and call them sets S_1, S_2, respectively. Also, for convenience, assume without loss of generality that n^{α} is an odd integer, and thus $y_{i,j} \neq 0$, and $Pr[z_{i,j} = 1] = Pr[y_{i,j} > 0] = \frac{1}{2} = Pr[z_{i,j} = -1]$. For $i \notin \{1, 2\}$, and any $j \neq i$, trivially, for any $k \in [m]$, $Pr[z_{k,i} = z_{k,j}] = \frac{1}{2}$.

We now argue that $Pr[z_{k,1} = z_{k,2}] = \frac{1}{2} + O(n^{-\alpha})$. To see this, consider first assigning the values to the $2(n^{\alpha} - 1)$ elements of matrix X in the kth row, for the columns indexed by all the elements of $S_1 \cup S_2$, with the exception of the two correlated columns. This assignment determines the value of $y_{k,1}$ [and $y_{k,2}$] up to ± 1, and thus it determines the value of $z_{k,1}$ [and $z_{k,2}$] in all cases, except when $z_{k,1} = 0$ [$z_{k,2} = 0$.] Let $a_1 = \sum_{i \in S_1, i \neq 1} x_{k,i}$, and let $a_2 = \sum_{i \in S_2, i \neq 2} x_{k,i}$, and thus $y_{k,1} = a_1 + x_{k,1}$. If $a_1 \neq 0$, then $|a_1| \geq 2$, and thus $Pr[z_{k,1} = z_{k,2} | a_1 \neq 0 \text{ or } a_2 \neq 0] = \frac{1}{2}$. On the other hand, $Pr[z_{k,1} = z_{k,2} | a_1 = a_2 = 0] = 1 - \eta$, and thus we conclude

$$Pr[z_{k,1} = z_{k,2}] = \frac{1}{2}(1 - Pr[a_1 = a_2 = 0]) + (1-\eta)Pr[a_1 = a_2 = 0] > \frac{1}{2} + \left(\frac{1}{2} - \eta\right)\frac{1}{4n^{\alpha}},$$

using the basic fact that if we flip n^{α} unbiased coins, we obtain $\lfloor n^{\alpha}/2 \rfloor$ heads with probability at least $\frac{1}{2n^{\alpha/2}}$.

Consider $W = Z^t Z$. By the above, we have

$$E[w_{1,2}] = 2m\left(\frac{1}{2} - \eta\right)\frac{1}{4n^{\alpha}} > \frac{1}{2}n^{\alpha+\epsilon}\frac{1}{(1/2-\eta)^{1+\epsilon}} > \frac{1}{2}m^{(1+\epsilon)/2}.$$

Trivially, all other entries $w_{i,j}$ with $i < j$ have $E[w_{i,j}] = 0$. The variance of these quantities is at most m and thus from the independence of the different rows, we may apply standard Chernoff bounds to yield the proposition. □

5 The *Expand-and-Aggregate* Algorithm

The results of the previous section show how to solve the closest-pair problem in the randomized setting with a single correlated pair, *provided that the points have dimension* $d \geq n^{2/3+\epsilon}$. What happens if d is quite small? Given the above approach, the intuition for what to do is clear: simply increase the dimension of the points.

Viewing the matrix of the points as an instance of the noisy-parity problem 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 roughly 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. It is easy to see that, provided we start with $\log^{1+\epsilon} n$ rows, and the initial correlation is a constant, we will be able to expand the number of rows sufficiently to apply the aggregation approach of the previous section.

This expansion approach was first fruitfully used by Lyubashevsky [8] to show that given few noisy parity 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 new instances are actually information theoretically *indistinguishable* from new examples (with higher noise rate). To prove this strong indistinguishability, Lyubashevsky employed the "Leftover Hash Lemma" of Impagliazzo and Zuckerman [7]. In our setting, we do not need any such strong information theoretic guarantees, we simply need guarantees on the inner products of pairs of columns.

Additionally, for clarity we only state the theorem in the randomized setting, and provide a crude analysis with yields an exponent of 1.8, as opposed to the exponent of 1.63 of the previous section.

Expand Vectors
Input: An $m \times n$ matrix X with entries $x_{i,j} \in \{-1,+1\}$, a positive integer m', and a positive integer $q < m$.
Output: A $m' \times n$ 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 jth component of the corresponding row to be $\prod_{i=1}^{q} x_{t_i,j}$.

The following proposition argues that the result of applying the above algorithm to sufficiently many rows of data will faithfully yield a much larger set of examples which have the essential properties we require.

Proposition 9. *For any constants $\epsilon, \beta, c > 0$, let $m > \frac{\log^{1+\epsilon} n}{\rho^2}$, and let X be an $m \times 2$ matrix with entries in $\{-1,+1\}$, assigned independently at random to be $+1$ with probability $1/2$, and let Y be the result of applying algorithm* Expand Vectors *to matrix X, with $q = c\frac{\log n}{\log \frac{1}{\rho}}$ and $m' = n^{\beta}$ then*

$$Pr\left[|\sum_{i=1}^{m'} y_{i,1} y_{i,2}| > 10\sqrt{m' \log n} \right] = o(\frac{1}{n^2}).$$

The following lemma is the core of the proof of the above proposition"

Lemma 10. *Let $z = \prod_{i=1}^{s} w_i$, where each $w_i \in \{-1,+1\}$ is chosen independently to be 1 with probability $\frac{1}{2} + \alpha$. Then $Pr[z = 1] = \frac{1}{2} + 2^{s-1}\alpha^s$.*

Proof. Letting $p = \frac{1}{2} - \alpha$, we have the following:

$$\begin{aligned} Pr[z=1] - Pr[z=-1] &= \sum_{i=0}^{s} (-1)^i p^i (1-p)^{s-i} \binom{s}{i} \\ &= ((1-p) - p)^s \\ &= (1-2p)^s = 2^s \alpha^s, \end{aligned}$$

□

Proof of Proposition 9. By elementary Chernoff bounds,

$$Pr[|\{i : x_{i,1} = x_{i,2}\}| \in (\frac{m}{2} - 3\sqrt{m}\sqrt{\log n}, \frac{m}{2} + 3\sqrt{m}\sqrt{\log n}] = 1 - o(1/n^2).$$

Assuming that $|\{i : x_{i,1} = x_{i,2}\}| \in \{\frac{m}{2} - 3\sqrt{m}\sqrt{\log n}, \frac{m}{2} + 3\sqrt{m}\sqrt{\log n}$, we now apply Lemma 10 to the length m vector whose ith element is $x_{i,1}x_{i,2}$, to yield that

$$Pr[y_{i,1} = y_{i,2}] \in (\frac{1}{2} - \left(\frac{6\sqrt{\log n}}{\sqrt{m}}\right)^q, \frac{1}{2} + \left(\frac{6\sqrt{\log n}}{\sqrt{m}}\right)^q).$$

Plugging in the values of m and m' yields that $Pr[y_{i,1} = y_{i,2}] \in (\frac{1}{2} - o(1/poly(n)), \frac{1}{2} + o(1/poly(n)))$. Thus, by the independence of the events that $y_{i,1} = y_{i,2}$ and $y_{j,1} = y_{j,2}$ (for a fixed matrix X), standard Chernoff bounds yield the desired statement. □

Expand and Aggregate
Input: An $m \times n$ matrix X with entries $x_{i,j} \in \{-1,+1\}$, $\epsilon \in (0,1)$, and $\rho \in (0,1)$.
Output: Two indices $c_1, c_2 \in [n]$.

- Let Y be the output of algorithm Expand Vectors on input X, $m' = n^{3/4+2\epsilon}$, and $q = \frac{\epsilon \log n}{\log \rho}$.
- Let c_1, c_2 be the output of algorithm Aggregate Rows on input Y, and $\alpha = \frac{1}{4}$.

Theorem 1. *For any constant $\epsilon > 0$, and $d \geq \frac{\log^{1+\epsilon} n}{\rho^2}$, given a set of n vectors in $\{-1,+1\}^d$, such that $n-1$ of the vectors are chosen uniformly at random, and one vector is chosen so as to be ρ-correlated with one of the $n-1$ other vectors, with probability $1-o(1)$, the algorithm* Expand and Aggregate *when given as input the matrix of examples, sufficiently small constant ϵ, and ρ, will output the indices of the two correlated vectors. Additionally, the runtime of the algorithm will be $O(n^{1.8})$.*

It is worth stressing that, 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 quite far from independent. This is in sharp contrast to the implementation of a similar idea in the context of learning large noisy parities, in work of Lyubashevsky [8]. Nevertheless, this lack of independence does not harm us, as we can still leverage the randomness from the *aggregation* step.

The following lemma will be helpful.

Lemma 11. *Given an arbitrary boolean $s \times s$ matrix X, 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 $Pr[|y - E[y]| \geq h^{3/2} \log h] = o(1/poly(h))$.*

Proof. Consider first choosing 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, and we claim that $Pr[|z_{S_1} - E[z_{S_1}]| \geq h^{3/2} \log h] = o(1/poly(h))$. 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 $z_{S_1} = h \sum_{i \in S} p_i$. The probability of deviating from the 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 $\{p_1, \ldots, p_s\}$, which is maximized when half the p_i's are 0 and the other half are 1, in which case the variance is at most h, and a Chernoff bound applies, yielding the claim.

Let $w_{S_1,S_2} := y - E[y|S_1]$. 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 S_1, and an identical analysis to the above (over the randomness in the choice of S_2, rather than S_1 yields the desired claim. □

Proof of Theorem 1. For sufficiently small ϵ, the runtime is bounded by the matrix multiplication step in the Aggregate Rows step of the algorithm. This operation multiplies an $n^{3/4+2\epsilon} \times n^{3/4}$ matrix by its transpose, and thus takes time $O(n^{3\omega/4+6\epsilon}) < O(n^{1.8})$, for sufficiently small ϵ, and using the fact that $\omega < 2.4$.

We now verify the correctness of the algorithm. Consider the $n \times n$ matrix C, whose entry $C_{i,j}$ is the inner product of the ith and jth columns of matrix Y. By a union bound over the n^2 entries, and Proposition 9, with probability $1 - o(1)$, aside from the two entries of C corresponding to the inner products of the correlated columns, the magnitude of $C_{i,j}$ will be bounded in magnitude by $10\sqrt{m \log n}$, where m is the number of rows of matrix Y.

Consider an arbitrary entry $w_{i,j}$ of the matrix W computed in the aggregation step. This entry is given by the inner product between the sum of two sets of n^α vectors, namely the columns of Y indexed sets S_i, S_j. By the distributive property, $w_{i,j}$ is simply $\sum_{k \in S_i, \ell \in S_j} C_{k,\ell}$, where C is the matrix of inner products defined in the previous paragraph. Thus by Lemma 11, excluding the contribution from the two correlated columns, with probability at least $1 - o(1/poly(n))$, $w_{i,j}$ will differ from its expectation by at most $10\sqrt{m \log n} \cdot n^{3\alpha/2} \log n$.

By Lemma 10, the correlation between the two correlated columns after the Expand Vectors portion of the algorithm will be, in expectation, at least $\rho^q = 1/n^\epsilon$. And thus, with probability $1-o(1)$, their inner product will contribute at least $\frac{mn^{-\epsilon}}{2} \geq n^{3/4+\epsilon}$ to some $w_{i,j}$, in which case that $w_{i,j}$ will be one of the two maximal entries of W, and the algorithm will terminate successfully. □

5.1 Learning Noisy Parities of Size k

The above results immediately yields an algorithm for learning parities of size k. For each length n example with even label, we transform it into a length $N = \binom{n}{k/2} \approx n^{k/2}$ vector, where each index represents the XOR of some set of $k/2$ of the indices of the original example. If the original example contained a set of k indices whose parity was biased towards being even, the new length N vector will contain $\binom{k}{k/2}$ pairs of indices (corresponding to different partitions of the k parity indices into two sets of size $k/2$), such that the parity of each pair is biased towards being even. Once one finds one such pair, one has found the set of parity indices. We defer the details to the full version.

Acknowledgements

I am very grateful to Vitaly Feldman for pointing out that algorithms for learning sparse noisy parities, together with the reduction of Feldman et al. [3] can be used to improve the runtime of the algorithm for learning k-juntas without noise of Mossel et al. [9], yielding Corollary 5.

References

[1] 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.

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

[3] V. Feldman, P. Gopalan, S. Khot, and A. Ponnuswami. New results for learning noisy parities and halfspaces. In *Proc. Foundations of computer Science (FOCS)*, 2006.

[4] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, pages 518–529, 1999.

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

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

[7] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *Proc. Foundations of computer Science (FOCS)*, pages 248–253, 1989.

[8] 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.

[9] 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.

[10] Ramamohan Paturi, Sanguthevar Rajasekaran, and John H. Reif. The light bulb problem. In *COLT'89*, pages 261–268, 1989.

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

[12] K. A. Verbeurgt. Learning dnf under the uniform distribution in quasipolynomial time. In *COLT*, pages 314–326, 1990.

ECCC ISSN 1433-8092
http://eccc.hpi-web.de