Noise-Tolerant Learning, the Parity Problem, and the Statistical Query Model

AVRIM BLUM, ADAM KALAI, AND HAL WASSERMAN

Carnegie Mellon University, Pittsburgh, Pennsylvania

Abstract. We describe a slightly subexponential time algorithm for learning parity functions in the presence of random classification noise, a problem closely related to several cryptographic and coding problems. Our algorithm runs in polynomial time for the case of parity functions that depend on only the first $O(\log n \log\log n)$ bits of input, which provides the first known instance of an efficient noise-tolerant algorithm for a concept class that is not learnable in the Statistical Query model of Kearns [1998]. Thus, we demonstrate that the set of problems learnable in the statistical query model is a strict subset of those problems learnable in the presence of noise in the PAC model.

In coding-theory terms, what we give is a poly(n)-time algorithm for decoding linear $k \times n$ codes in the presence of random noise for the case of $k = c \log n \log\log n$ for some $c > 0$. (The case of $k = O(\log n)$ is trivial since one can just individually check each of the 2^k possible messages and choose the one that yields the closest codeword.)

A natural extension of the statistical query model is to allow queries about statistical properties that involve t-tuples of examples, as opposed to just single examples. The second result of this article is to show that any class of functions learnable (strongly or weakly) with t-wise queries for $t = O(\log n)$ is also weakly learnable with standard unary queries. Hence, this natural extension to the statistical query model does not increase the set of weakly learnable functions.

Categories and Subject Descriptors: F.2.0 [**Analysis of Algorithms and Problem Complexity**]: General; G.3 [**Probability and Statistics**]; I.5.0 [**Pattern Recognition**]: General

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Computational learning theory, machine learning, statistical queries

1. *Introduction*

An important question in the study of machine learning is: "What kinds of functions can be learned efficiently from noisy, imperfect data?" The statistical query (SQ) framework of Kearns [1998] was designed as a useful, elegant model for addressing this issue. The SQ model provides a restricted interface between a learning algorithm and its data, and has the property that any algorithm for learning in the SQ model can automatically be converted to an algorithm for learning in the presence of *random*

Authors' address: Department of Computer Science, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213, e-mail: {avrim,akalai}@cs.cmu.edu; Halwass@aol.com.

classification noise in the standard PAC model. (This result has been extended to more general forms of noise as well [Decatur 1993, 1996].) The importance of the Statistical Query model is attested to by the fact that, before its introduction, there were only a few provably noise-tolerant learning algorithms, whereas now it is recognized that a large number of learning algorithms can be formulated as SQ algorithms, and hence can automatically be made noise-tolerant.

The importance of the SQ model has led to the open question of whether there are problems learnable with random classification noise in the PAC model but not learnable by statistical queries. This is especially interesting because one can characterize information theoretically (i.e., without complexity assumptions) what kinds of problems can be learned in the SQ model [Blum et al. 1994]. For example, the class of parity functions, which *can* be learned efficiently from *non*-noisy data in the PAC model, provably cannot be learned efficiently in the SQ model under the uniform distribution. Unfortunately, no efficient non-SQ algorithm for learning them in the presence of noise is known either (this is closely related to the problem of decoding random linear codes [MacWilliams and Sloane 1977]).

In this article, we describe a polynomial-time algorithm for learning the class of parity functions that depend on only the first $O(\log n \log\log n)$ bits of input, in the presence of random classification noise of a constant noise rate. This class provably cannot be learned in the SQ model, and thus is the first known example of a concept class learnable with noise but not via statistical queries. Our algorithm has recently been shown to have applications to the problems of determining the shortest lattice vector [Ajtai et al. 2001] and its length [Kumar and Sivakumar 2001], to cryptanalysis [Wagner 2002], and to various other analyses of statistical queries [Jackson 2000].

An equivalent way of stating this result is that we are given a random $k \times n$ $\{0, 1\}$ matrix A, as well as an n-bit vector $\tilde{y}$ produced by multiplying an unknown k-bit message x by A, and then corrupting each bit of the resulting codeword $y = xA$ with probability $\eta < 1/2$. Our goal is to recover y in time poly(n). For this problem, the case of $k = O(\log n)$ is trivial because one could simply try each of the 2^k possible messages and output the nearest codeword found. Our algorithm works for $k = c \log n \log\log n$ for some $c > 0$. The algorithm does not actually need A to be random, so long as the noise is random and there is no other codeword within distance $o(n)$ from the true codeword y.

Our algorithm can also be viewed as a slightly subexponential time algorithm for learning arbitrary parity functions in the presence of noise. For this problem, the brute-force algorithm would draw $O(n)$ labeled examples, and then search through all 2^n parity functions to find the one of least empirical error. (A standard argument can be used to say that with high probability, the correct function will have the lowest empirical error [Angluin and Laird 1988].) In contrast, our algorithm runs in time $2^{O(n/\log n)}$, though it also requires $2^{O(n/\log n)}$ labeled examples. This improvement is small but nonetheless sufficient to achieve the desired separation result.

Our algorithms, unfortunately, do not have polynomial dependence on $1/(1/2 - \eta)$. An open problem left by our work is whether these results can be extended to high noise rates such as $\eta = 1/2 - 1/n$.

The second result of this article concerns a k-wise version of the Statistical Query model. In the standard version, algorithms may only ask about statistical properties of single examples (e.g., what is the probability that a random example is labeled positive and has its first bit equal to 1?) In the k-wise version, algorithms may ask about properties of k-tuples of examples (e.g., what is the probability that two random examples have an even dot-product and have the same label?) Given the first result of this article, it is natural to ask whether allowing k-wise queries, for some small value of k, might increase the set of SQ-learnable functions. What we show is that for $k = O(\log n)$, any concept class learnable from k-wise queries is also (weakly) learnable from unary queries. Thus, the seeming generalization of the SQ model to allow for $O(\log n)$-wise queries does not close the gap we have demonstrated between what is efficiently learnable in the SQ and noisy-PAC models. Note that this result is the best possible with respect to k because the results of Blum et al. [1994] imply that for $k = \omega(\log n)$, there are concept classes learnable from k-wise queries but not unary queries. On the other hand, $\omega(\log n)$-wise queries are in a sense less interesting because it is not clear whether they can in general be simulated in the presence of noise.

1.1. MAIN IDEAS. The standard way to learn parity functions without noise is based on the fact that if an example can be written as a sum (mod 2) of previously seen examples, then its label must be the sum (mod 2) of those examples' labels [Helmbold et al. 1992]. So, once one has found a basis, one can use that to deduce the label of *any* new example (or, equivalently, use Gaussian elimination to produce the target function itself).

In the presence of noise, this method breaks down. If the original data had noise rate $1/4$, say, then the sum of s labels has noise rate $1/2 - (1/2)^{s+1}$. This means we can add together only $O(\log n)$ examples if we want the resulting sum to be correct with probability $1/2 + 1/poly(n)$. Thus, if we want to use this kind of approach, we need some way to write a new test example as a sum of only a *small number* of training examples.

Let us now consider the case of parity functions that depend on only the first $k = \log n \log\log n$ bits of input. Equivalently, we can think of all examples as having the remaining $n - k$ bits equal to 0. Gaussian elimination will in this case allow us to write our test example as a sum of k training examples, which is too many. Our algorithm will instead write it as a sum of $k/\log k = O(\log n)$ examples, which gives us the desired noticeable bias (that can then be amplified).

Notice that if we have seen $poly(n)$ training examples (and, say, each one was chosen uniformly at random), we can argue existentially that for $k = c \log n \log\log n$, one should be able to write any new example as a sum of just $O(\log\log n)$ training examples, since there are $n^{\Omega(\log\log n)} \gg 2^k$ subsets of this size and these subsets are pairwise independent. So, while our algorithm is finding a smaller subset than Gaussian elimination, it is not doing best possible. If one *could* achieve, say, a constant-factor approximation to the problem "given a set of vectors, find the smallest subset that sums to a given target vector" then this would improve the value of k we can handle in polynomial time from $O(\log n \log\log n)$ to $O(\log^2 n)$. Equivalently, this would allow one to learn parity functions over n bits in time $2^{O(\sqrt{n})}$, compared to the $2^{O(n/\log n)}$ time of our algorithm.

2. *Definitions and Preliminaries*

A *concept* is a boolean function on an *input space*, which in this article will generally be $\{0, 1\}^n$. We will be considering the problem of learning a target concept in the presence of *random classification noise* [Angluin and Laird 1988]. In this model, there is some fixed (known or unknown) noise rate $\eta < 1/2$, a fixed (known or unknown) probability distribution $\mathcal{D}$ over $\{0, 1\}^n$, and an unknown target concept c. The learning algorithm may repeatedly "press a button" to request a labeled example. When it does so, it receives a pair (x, ℓ), where x is chosen from $\{0, 1\}^n$ according to $\mathcal{D}$ and ℓ is the value $c(x)$, but "flipped" with probability η. That is, $\ell = c(x)$ with probability $1 - \eta$, and $\ell = 1 - c(x)$ with probability η. The goal of the learning algorithm is to find an *ϵ-approximation* of c: a hypothesis function h such that $\Pr_{x \leftarrow \mathcal{D}}[h(x) = c(x)] \geq 1 - \epsilon$.

A *concept class* is a set of concepts. We say that a concept class C is *efficiently learnable in the presence of random classification noise* under distribution $\mathcal{D}$ if there exists an algorithm $\mathcal{A}$ such that for any $\epsilon > 0, \delta > 0, \eta < 1/2$, and any target concept $c \in C$, the algorithm $\mathcal{A}$ with probability at least $1 - \delta$ produces an ϵ-approximation of c when given access to $\mathcal{D}$-random examples which have been labeled by c and corrupted by noise of rate η. Furthermore, $\mathcal{A}$ must run in time polynomial in n, $1/\epsilon$, and $1/\delta$.[1]

A *parity function c* is defined by a corresponding vector $c \in \{0, 1\}^n$; the parity function is then given by the rule $c(x) = x \cdot c \pmod 2$. We say that *c depends on only the first k bits of input* if all nonzero components of c lie in its first k bits. So, in particular, there are 2^k distinct parity functions that depend on only the first k bits of input. Parity functions are especially interesting to consider under the uniform distribution $\mathcal{D}$, because under that distribution parity functions are pairwise uncorrelated.

2.1. The Statistical Query Model.

The Statistical Query (SQ) model can be viewed as providing a restricted interface between the learning algorithm and the source of labeled examples. In this model, the learning algorithm may only receive information about the target concept through *statistical queries*. A statistical query is a query about some property Q of labeled examples (e.g., that the first two bits are equal and the label is positive), along with a tolerance parameter $\tau \in [0, 1]$. When the algorithm asks a statistical query (Q, τ), it is asking for the probability that predicate Q holds true for a random correctly labeled example, and it receives an approximation of this probability up to $\pm\tau$. In other words, the algorithm receives a response $\hat{P}_Q \in [P_Q - \tau, P_Q + \tau]$, where $P_Q = \Pr_{x \leftarrow \mathcal{D}}[Q(x, c(x))]$. We also require each query Q to be polynomially evaluable (i.e., given (x, ℓ), we can compute $Q(x, \ell)$ in polynomial time).

Notice that a statistical query can be simulated by drawing a large sample of noiseless data and computing an empirical average, where the size of the sample would be roughly $O(1/\tau^2)$ if we wanted to assure an accuracy of τ with high probability.

A concept class C is *learnable from statistical queries* with respect to distribution $\mathcal{D}$ if there is a learning algorithm $\mathcal{A}$ such that for any $c \in C$ and any $\epsilon > 0$, $\mathcal{A}$ produces an ϵ-approximation of c from statistical queries; furthermore, the running time, the number of queries asked, and the inverse of the smallest tolerance used must be polynomial in n and $1/\epsilon$.

[1] Typically, one would also require polynomial dependence on $1/(1/2 - \eta)$—in part because typically this is easy to achieve (e.g., it is achieved by any statistical query algorithm). Our algorithms run in polynomial time for any *fixed* $\eta < 1/2$, but have a super-polynomial dependence on $1/(1/2 - \eta)$.

We also want to talk about *weak learning.* An algorithm $\mathcal{A}$ weakly learns a concept class C if for any $c \in C$ and for *some* $\epsilon < 1/2 - 1/\mathrm{poly}(n)$, $\mathcal{A}$ produces an ϵ-approximation of c. That is, an algorithm weakly learns if it can do noticeably better than guessing.

The statistical query model is defined with respect to nonnoisy data. However, statistical queries can be simulated from data corrupted by random classification noise [Kearns 1998]. Thus, any concept class learnable from statistical queries is also PAC-learnable in the presence of random classification noise. There are several variants to the formulation given above that improve the efficiency of the simulation [Aslam and Decatur 1998a, 1998b], but they are all polynomially related.

One technical point: we have defined statistical query learnability in the "known distribution" setting (algorithm $\mathcal{A}$ knows distribution $\mathcal{D}$); in the "unknown distribution" setting, $\mathcal{A}$ is allowed to ask for random unlabeled examples from the distribution $\mathcal{D}$. This prevents certain trivial exclusions from what is learnable from statistical queries.

2.2. AN INFORMATION-THEORETIC CHARACTERIZATION. It is proven in [Blum et al. 1994] that any concept class containing more than polynomially many pairwise uncorrelated functions cannot be learned even weakly in the statistical query model. Specifically,

Definition 1 [*Blum et al.* 1994, *Def.* 2]. For concept class C and distribution $\mathcal{D}$, the *statistical query dimension* SQ-DIM(C, $\mathcal{D}$) is the largest number d such that C contains d concepts $c_1, \ldots, c_d$ that are nearly pairwise uncorrelated: specifically, for all $i \neq j$,

$$\left| \Pr_{x \leftarrow D}[c_i(x) = c_j(x)] - \Pr_{x \leftarrow D}[c_i(x) \neq c_j(x)] \right| \leq \frac{1}{d^3}.$$

THEOREM 1 [BLUM ET AL. 1994, THM. 12]. *In order to learn C to error less than* $1/2 - 1/d^3$ *in the SQ model, where* $d = \text{SQ-DIM}(C, \mathcal{D})$*, either the number of queries or* $1/\tau$ *must be at least* $\frac{1}{2}d^{1/3}$.

Note that the class of parity functions over $\{0, 1\}^n$ that depend on only the first $O(\log n \log\log n)$ bits of input contains $n^{O(\log\log n)}$ functions, all pairs of which are uncorrelated with respect to the uniform distribution. Thus, this class cannot be learned (even weakly) in the SQ model with polynomially many queries of $1/\mathrm{poly}(n)$ tolerance. But we now show that there nevertheless exists a polynomial-time PAC-algorithm for learning this class in the presence of random classification noise.

3. *Learning Parity with Noise*

3.1. LEARNING OVER THE UNIFORM DISTRIBUTION. For ease of notation, we use the "length-k parity problem" to denote the problem of learning a parity function over $\{0, 1\}^k$, under the uniform distribution, in the presence of random classification noise of rate η.

THEOREM 2. *Let a and b be positive integers such that* $ab \geq k$. *Then the length-k parity problem can be solved with sample-size and total computation time* $\mathrm{poly}((\frac{1}{1-2\eta})^{2^a}, 2^b)$.

COROLLARY 3. *The length-k parity problem, for noise rate η equal to any constant less than* $1/2$, *can be solved with sample-size and total computation-time* $2^{O(k/\log k)}$.

PROOF. Plugging in $a = \frac{1}{2}\log k$ and $b = 2k/\log k$ into Theorem 2 yields the desired $2^{O(k/\log k)}$ bound for constant noise rate η. □

Thus, in the presence of noise we can learn parity functions over $\{0, 1\}^n$ in time and sample size $2^{O(n/\log n)}$, and we can learn parity functions over $\{0, 1\}^n$ that only depend on the first $k = O(\log n \log\log n)$ bits of the input in time and sample size *poly*(n).

We begin our proof of Theorem 2 with a simple lemma about how noise becomes amplified when examples are added together. For convenience, if x_1 and x_2 are examples, we let $x_1 + x_2$ denote the vector sum mod 2; similarly, if ℓ_1 and ℓ_2 are labels, we let $\ell_1 + \ell_2$ denote their sum mod 2.

LEMMA 4. *Let* $(x_1, \ell_1), \ldots, (x_s, \ell_s)$ *be examples labeled by c and corrupted by random noise of rate* η. *Then* $\ell_1 + \cdots + \ell_s$ *is the correct value of* $(x_1 + \cdots + x_s) \cdot c$ *with probability* $\frac{1}{2} + \frac{1}{2}(1 - 2\eta)^s$.

PROOF. Clearly true when $s = 1$. Now assume that the lemma is true for $s - 1$. Then the probability that $\ell_1 + \cdots + \ell_s = (x_1 + \cdots + x_s) \cdot c$ is

$$(1-\eta)\left(\frac{1}{2} + \frac{1}{2}(1-2\eta)^{s-1}\right) + \eta\left(\frac{1}{2} - \frac{1}{2}(1-2\eta)^{s-1}\right) = \frac{1}{2} + \frac{1}{2}(1-2\eta)^s.$$

The lemma then follows by induction. □

The idea for the algorithm is that, by drawing many more examples than the minimum needed to learn information-theoretically, we will be able to write basis vectors such as $(1, 0, \ldots, 0)$ as the sum of a relatively small number of training examples—substantially smaller than the number that would result from straightforward Gaussian elimination. In particular, for the length $O(\log n \log\log n)$ parity problem, we will be able to write $(1, 0, \ldots, 0)$ as the sum of only $O(\log n)$ examples. By Lemma 4, this means that, for any constant noise rate $\eta < 1/2$, the corresponding sum of labels will be polynomially distinguishable from random. Hence, by repeating this process as needed to boost reliability, we may determine the correct label for $(1, 0, \ldots, 0)$, which is equivalently the first bit of the target vector c. This process can be further repeated to determine the remaining bits of c, allowing us to recover the entire target concept with high probability.

To describe the algorithm for the length-k parity problem, it will be convenient to view each example as consisting of a blocks, each b bits long (so, $k = ab$) where a and b will be chosen later. We then introduce the following notation.

Definition 2. Let V_i be the subspace of $\{0, 1\}^{ab}$ consisting of those vectors whose last i blocks have all bits equal to zero. An i*-sample* of size s is a set of s vectors independently and uniformly distributed over V_i.

The goal of our algorithm will be to use labeled examples from $\{0, 1\}^{ab}$ (these form a 0-sample) to create an i-sample such that each vector in the i-sample can be written as a sum of at most 2^i of the original examples, for all $i = 1, 2, \ldots, a - 1$. We attain this goal via the following lemma.

LEMMA 5. *Assume we are given an i-sample of size s. We can in time $O(s)$ construct an $(i+1)$-sample of size at least $s - 2^b$ such that each vector in the $(i+1)$-sample is written as the sum of two vectors in the given i-sample.*

PROOF. Let the i-sample be $x_1, \dots, x_s$. In these vectors, blocks $a-i+1, \dots, a$ are all zero. Partition $x_1, \dots, x_s$ based on their values in block $a-i$. This results in a partition having at most 2^b classes. From each nonempty class p, pick one vector x_{j_p} at random and add it to each of the other vectors in its class; then discard x_{j_p}. The result is a collection of vectors $u_1, \dots, u_{s'}$, where $s' \geq s - 2^b$ (since we discard at most one vector per class).

What can we say about $u_1, \dots, u_{s'}$? First of all, each u_j is formed by summing two vectors in V_i which have identical components throughout block $a-i$, "zeroing out" that block. Therefore, u_j is in V_{i+1}. Second, each u_j is formed by taking some x_{j_p} and adding to it a random vector in V_i, subject only to the condition that the random vector agrees with x_{j_p} on block $a-i$. Therefore, each u_j is an independent, uniform-random member of V_{i+1}. The vectors $u_1, \dots, u_{s'}$ thus form the desired $(i+1)$-sample. □

Using this lemma, we can now prove our main theorem.

PROOF OF THEOREM 2. Draw $a2^b$ labeled examples. Observe that these qualify as a 0-sample. Now apply Lemma 5, $a-1$ times, to construct an $(a-1)$-sample. This $(a-1)$-sample will have size at least 2^b. Recall that the vectors in an $(a-1)$-sample are distributed independently and uniformly at random over V_{a-1}, and notice that V_{a-1} contains only 2^b distinct vectors, one of which is $(1, 0, \dots, 0)$. Hence, there is at least a $1 - 1/e$ chance that $(1, 0, \dots, 0)$ appears in our $(a-1)$-sample. If this does not occur, we repeat the above process with new labeled examples. Note that the expected[2] number of repetitions is constant (it is at most $1/(1-1/e)$).

Now, unrolling our applications of Lemma 5, observe that we have written the vector $(1, 0, \dots, 0)$ as the sum of at most 2^{a-1} of our labeled examples—and we have done so without examining their labels. Thus, the label noise is still random, and we can apply Lemma 4. Hence, the sum of the labels gives us the correct value of $(1, 0, \dots, 0) \cdot c$ with probability at least $\frac{1}{2} + \frac{1}{2}(1-2\eta)^{2^{a-1}}$.

We can amplify our success probability by repeating the above process using new labeled examples each time and taking majority vote. If we do this $\text{poly}((\frac{1}{1-2\eta})^{2^a}, b)$ times, Hoeffding bounds guarantee that we will determine $(1, 0, \dots, 0) \cdot c$ with probability of error exponentially small in ab. This gives us the first bit of c. By cyclically shifting all examples, we can use the same method to find *every* bit of c. Since we use $a2^b$ samples per round, this means that with high probability we can determine c using a number of examples and total computation-time $\text{poly}((\frac{1}{1-2\eta})^{2^a}, 2^b)$. □

3.2. EXTENSION TO OTHER DISTRIBUTIONS.

While the uniform distribution is in this case the most interesting, we can extend our algorithm to work over any distribution, just as parity *without* noise can be learned for an arbitrary distribution [Helmbold et al. 1992]. We give two methods: the first follows the same lines as

[2] Standard techniques can be used to convert an algorithm with polynomial *expected* runtime into one with polynomial runtime that works with high probability.

the uniform distribution algorithm above; the second views learning as an online process and gives a mistake-bounded algorithm.

3.2.1. *Version 1: Pairing Similar Examples.* In the case of the uniform distribution, we were able to write the vector $(1, 0, 0, \ldots, 0)$ as the sum of at most 2^a labeled examples, where each k-bit example was divided into a blocks of width b. We repeated that process several times and took majority vote. Here, we cannot necessarily do that—for instance, it could be that all examples in the support of our distribution $\mathcal{D}$ have a 0 in the first coordinate.

Instead, with high probability, we are able to write a random vector from $\mathcal{D}$ as the sum of at most 2^a vectors. Again, we do this repeatedly and take majority vote. This will allow us to generate a polynomial-size *noiseless* data set from $\mathcal{D}$ and use standard noiseless parity learning algorithms. Note that, in the noiseless case, one cannot always find the exact parity function since the function might be underdetermined. Instead, one finds a parity function that agrees with all of the training data, and with high probability this will be correct on nearly all of $\mathcal{D}$.

The basic primitive is similar: we repeatedly write an example as the sum of at most 2^a labeled examples.

LEMMA 6. *Given an example x drawn from $\mathcal{D}$ and m additional examples also from $\mathcal{D}$, with probability at least* $1 - a2^{a+b}/m$, *we can write x as the sum of at most 2^a of the labeled examples, in time* $\text{poly}(m, k)$.

PROOF. First, let the multiset S be the union of $\{x\}$ with the additional random examples. As above, we think of each k-bit example as divided into a blocks of width b. Do the following:

(1) Pick two examples from S that agree on their last nonzero block.
(2) Remove these two examples from S and replace them by their sum.
(3) Repeat until there are no more such pairs.

When the above procedure halts, we will have a number of 0 vectors and at most $a(2^b - 1)$ other vectors, because there can be at most one vector for each of the $2^b - 1$ possible nonzero blocks in a positions. Each vector will be the sum of at most 2^a original examples because the "tree" of additions producing any given vector has depth at most a. Thus, all but at most $a2^{a+b}$ examples will be in sets of size at most 2^a that sum to 0. Any example in one of the sets that sum to 0 can be written as the sum of the remaining $\leq 2^a - 1$ examples. Since we included x as a random member of S, by symmetry the probability that x is not written as the sum of $\leq 2^a - 1$ examples is at most $1 - a2^{a+b}/m$. □

Note that if we wanted to be slightly more efficient and more similar to the previous section, we could group the examples into equivalence classes in Step (1) and subtract a single example from all the others in its class. However, the above lemma is enough to show,

THEOREM 7. *The length-k parity problem over an arbitrary distribution $\mathcal{D}$, for constant noise rate $\eta < 1/2$, can be solved with number of samples and total computation-time* $2^{O(k/\log k)}$.

PROOF. We use a standard noiseless parity learning algorithm that with probability $1 - \delta/3$ returns a parity function that is correct to within $1 - \epsilon$ over $\mathcal{D}$. Suppose this algorithm requires $t = \text{poly}(k, 1/\epsilon, 1/\delta)$ examples.

We draw t *unlabeled* examples and s noisy data sets of size m. By the above lemma, with probability at least $1-(st)a2^{a+b}/m$, we can write each of the unlabeled examples as the sum of at most 2^a noisy examples from each of the s data sets. Thus, we succeed in doing this with probability at least $1-\delta/3$, provided that

$$m \geq \frac{3(st)a2^{a+b}}{\delta}.$$

Now, consider a single unlabeled example. We decide its label by majority vote over the s independent ways we have of representing it as a sum of at most 2^a examples. By Lemma 4, each one will be correct with probability at least $\frac{1}{2}+\frac{1}{2}(1-2\eta)^{2^a}$. By Chernoff bounds, our majority will be correct with probability at least $1-\frac{\delta}{3t}$, provided that

$$s \geq \frac{3\log\dfrac{2t}{\delta}}{(1-2\eta)^{2^{a+1}}}.$$

Thus, for the above m and s, our probability of being off by more than ϵ is at most $\delta/3+\delta/3+t\delta/(3t)=\delta$, where the three terms are for the failure of parity learner, failure to write some example as sum of 2^a noisy examples, and the failure of majority to be correct on some example. Again, using $a=\frac{1}{2}\log k$ and $b=2k/\log k$ gives s, m in poly$(1/\epsilon, 1/\delta, 2^{k/\log k})$ for constant noise rate η. $\square$

3.2.2. *Version 2: Online Block Gaussian Elimination.* We now give an alternative proof that requires a bit more notation, but is in some sense cleaner because it works in the online mistake-bound setting [Littlestone 1988, 1989]. That is, we dispense with the distribution altogether, and allow an adversary to present the learner with an arbitrary *sequence* of examples, one at a time. Given a new test example, the algorithm will output either "I don't know" or a prediction of the label. In the former case, the algorithm is told the correct label, flipped with probability η. The claim is that the algorithm will, with high probability, be correct in all its predictions, and furthermore will output "I don't know" only a small fraction of the time. In the coding-theoretic view, this corresponds to producing a $1-o(1)$ fraction of the desired codeword, where the remaining entries are left blank. This allows us to recover the full codeword so long as no other codeword is within relative distance $o(1)$.

The algorithm is essentially a form of Gaussian elimination, but where each entry in the matrix is an element of the vector space $\mathbf{F}_2^b$ rather than an element of the field $\mathbf{F}_2$ (again $k=ab$). We will form a matrix M with a process to be described later, but it will have the following properties. Let $x[j]$ denote the jth block of vector x, so x has length ab and $x[j]$ has length b. The matrix M has rows $M_{i,v}$ indexed by two parameters, $1\leq i\leq a$ and $v\in\{0,1\}^b$, where $M_{i,v}$ is a length ab vector with the properties:

$$M_{i,v}[j]=0 \quad \text{for} \quad j=1,2,\ldots,i-1, \quad \text{and} \quad M_{i,v}[i]=v$$

Once we have such an M, given any vector x, we can write it as the sum of examples from M as follows. Let $x_0=x$ and

$$x_i = x_{i-1} + M_{i,x_{i-1}[i]}, \tag{1}$$

for $1 \leq i \leq a$, so that $x_a = 0$. Unravelling, we have written x as the sum of a vectors from M. Notice that x_i has all 0's for the first i blocks.

Instead of forming a single matrix M, we form N such matrices, $M^{(1)}, M^{(2)}, \ldots, M^{(N)}$, as follows. Each matrix begins empty. For each new example x we get, we try to write x as the sum of elements of matrix $M^{(1)}$ according to Eq. (1). If we fail, this means that for some $1 \leq i \leq a$, $M^{(1)}_{i,x_{i-1}[i]}$ is still empty in $M^{(1)}$. In this case, we use x_{i-1} to fill in that entry in the matrix, which is valid because x_{i-1} has 0's for the first $i-1$ blocks.

If we are successful in writing x as the sum of entries from $M^{(1)}$, we apply the same to $M^{(2)}, M^{(3)}, \ldots, M^{(N)}$ stopping as soon as we find a matrix for which we cannot do our Gaussian elimination and outputting "I don't know." If we do not stop, then we will successfully have written x as the sum of exactly a entries from each of the matrices. Note that each matrix element $M_{i,v}$ is in turn the sum of at most 2^{i-1} examples that we have seen so far. Therefore, we choose the label of x according to the sum of the labels corresponding to these examples (which we can simply store along with $M_{i,v}$), and output the majority vote over all N sums. By doing this, we observe the following:

LEMMA 8. *The following hold for the above procedure:*

(1) *"I don't know" is output at most $Na2^b$ times.*

(2) *The probability of ever erring on a prediction is at most:*

$$2^k \exp\left(-\frac{N}{2}(1-2\eta)^{2^{a+1}}\right).$$

PROOF. Every time we output "I don't know," we fill in one matrix entry, and each of the N matrices has $a2^b$ entries $M_{i,v}$ because there are a possibilities for i and 2^b possibilities for v.

Now, suppose we make a prediction on some example x. Then we have written x as the sum of a entries from each matrix, with one entry for each $1 \leq i \leq a$. Since each $M_{i,v}$ is the sum of at most 2^{i-1} previous examples, we have written x as the sum of at most 2^a entries N times and taken a majority. By Lemma 4, each sum is a correct prediction with probability $\frac{1}{2} + \frac{1}{2}(1-2\eta)^{2^a}$. Thus the majority of N of these, by Chernoff bounds, is incorrect with probability at most $\exp(-\frac{N}{2}(1-2\eta)^{2^{a+1}})$. Finally, there are only 2^k different possible x's. All predictions for the same x will be the same. This gives the quantity in the lemma. $\square$

Again, choosing $a = \frac{1}{2}\log k$ and $b = 2k/\log k$ allows us to have $N = 2^{O(\sqrt{k})} \log\frac{1}{\delta}$ and have no errors with probability at least $1-\delta$. However, we would say "I don't know" up to $Na2^b = 2^{O(k/\log k)}\log\frac{1}{\delta}$ times. Thus, the number of examples we need in order to give a correct answer on a $1-\epsilon$ fraction is $2^{O(k/\log k)}(\frac{1}{\epsilon}\log\frac{1}{\delta})$.

4. *Limits of O(log n)-wise Queries*

We return to the general problem of learning a target concept c over a space of examples with a fixed distribution $\mathcal{D}$. A limitation of the statistical query model is that it permits only what may be called *unary* queries. That is, an SQ algorithm can access c only by requesting approximations of probabilities of form $\Pr_x[Q(x, c(x))]$, where x is $\mathcal{D}$-random and Q is a polynomially evaluable predicate. A natural question is whether problems not learnable from such queries can be learned, for example, from binary queries: that is, from probabilities of form $\Pr_{x_1,x_2}[Q(x_1, x_2, c(x_1), c(x_2))]$. The following theorem demonstrates that this is not possible, proving that $O(\log n)$-wise queries are no better than unary queries, at least with respect to weak-learning.

We assume in the discussion below that all algorithms also have access to individual *unlabeled* examples from distribution $\mathcal{D}$, as is usual in the SQ model. Also, for a k-tuple of examples $\vec{x} = x_1, x_2, \ldots, x_k$, let $c(\vec{x}) = (c(x_1), c(x_2), \ldots, c(x_k))$.

THEOREM 9. *Let $k = O(\log n)$, and assume that there exists a poly(n)-time algorithm using k-wise statistical queries that weakly learns a concept class C under distribution $\mathcal{D}$. That is, this algorithm learns from approximations of probabilities of form* $\Pr_{\vec{x}}[Q(\vec{x}, c(\vec{x}))]$, *where Q is a polynomially evaluable predicate, and $\vec{x}$ is a k-tuple of examples. Then there exists a poly(n)-time algorithm that weakly learns the same class using only unary queries, under $\mathcal{D}$.*

PROOF. We consider each k-wise query $Q(\vec{x}, c(\vec{x}))$ employed by the original algorithm. We will argue that either we can approximate it to within the required accuracy τ, or we can weak learn c to within error $1/2 - \epsilon$, where $\epsilon = \tau/2^{k+2}$. The first thing our algorithm will do is use Q to construct several candidate weak hypotheses. It then uses unary statistical queries to test whether each of these hypotheses is in fact noticeably correlated with the target. If none of them appear to be good, it uses this fact to estimate the value of the k-wise query. We prove that for any k-wise query, with high probability we either succeed in finding a weak hypothesis or we output a good estimate of the k-wise query. In either case, we succeed in our goal of weakly learning c using only unary queries.

Without loss of generality, let us assume that $\Pr_x[c(x) = 1] \in [1/2-\epsilon, 1/2+\epsilon]$. Otherwise, weak-learning is easy by just predicting all examples are positive or all examples are negative. This assumption implies that if a hypothesis h satisfies $|\Pr_x[h(x) = 1 \wedge c(x) = 1] - \frac{1}{2}\Pr_x[h(x) = 1]| \geq \epsilon$, then either $h(x)$ or $1 - h(x)$ is a weak hypothesis. To see this, the error of $h(x)$ is,

$$Pr_x[h(x) \neq c(x)] = Pr_x[c(x) = 1] + Pr_x[h(x) = 1] - 2Pr_x[h(x) = 1 \wedge c(x) = 1].$$

The first term in the right-hand side is in $[1/2 - \epsilon, 1/2 + \epsilon]$ while the rest of the right-hand side has absolute value at least 2ϵ, giving an error $\notin [1/2-\epsilon, 1/2+\epsilon]$, so h or $1 - h$ is a weak hypothesis.

For conciseness, we use the following notational shortcuts, where x is an example and $\vec{z} = z_1, \ldots, z_k$ is a k-tuple of examples:

$$\begin{aligned} \vec{z}_{a\ldots b} &= z_a, z_{a+1}, \ldots, z_b \\ \vec{z}_{\frac{x}{i}} &= z_{1\ldots i-1}, x, z_{i+1\ldots k}. \end{aligned}$$

We now generate a set of candidate hypotheses by choosing one random k-tuple of unlabeled examples $\vec{z}$. For each $1 \leq i \leq k$ and $\vec{\ell} \in \{0, 1\}^k$, we define candidate hypotheses $h_{\vec{z},i,\vec{\ell}}$ as follows:

$$h_{\vec{z},i,\vec{\ell}}(x) = 1 \quad \text{if} \quad Q(\vec{z}_{\frac{x}{i}}, \vec{\ell}),\ 0 \quad \text{otherwise},$$

and then use a unary statistical query to tell if $h_{\vec{z},i,\vec{\ell}}(x)$ or $1 - h_{\vec{z},i,\vec{\ell}}(x)$ is a successful weak hypothesis for c. As noted above, we will have found a weak hypothesis if

$$\left|\Pr_x\big[Q(\vec{z}_{\frac{x}{i}}, \vec{\ell}) \wedge c(x) = 1\big] - \frac{1}{2}\Pr_x\big[Q(\vec{z}_{\frac{x}{i}}, \vec{\ell})\big]\right| \geq \epsilon.$$

In this case, we say the ith coordinate *matters*. We repeat this process for $O(1/\epsilon)$ randomly chosen k-tuples $\vec{z}$. We now consider two cases.

Case I. Suppose that the ith coordinate matters to the k-wise query Q for some i and $\vec{\ell}$. By this, we mean there is at least an ϵ chance of the above inequality holding for random $\vec{z}$. Then, with high probability, we will discover such a $\vec{z}$ and thus weak learn.

Case II. Suppose, on the contrary, that for no i or $\vec{\ell}$ does the ith coordinate matter, that is, the probability of a random z satisfying the above inequality is less than ϵ. This means that

$$\mathbf{E}_{\vec{z}}\left[\left|\Pr_x\left[Q(\vec{z}_{\frac{x}{i}}, \vec{\ell}) \wedge c(x) = 1\right] - \frac{1}{2}\Pr_x\left[Q(\vec{z}_{\frac{x}{i}}, \vec{\ell})\right]\right|\right] < 2\epsilon. \tag{2}$$

Consider the following quantity for any $\vec{b} \in \{0, 1\}^{i-1}$:

$$\mathbf{E}_{\vec{z}}\left[\left|\Pr_x\left[Q(\vec{z}_{\frac{x}{i}}, \vec{\ell}) \wedge c(\vec{z}_{1\ldots i-1}, x) = \vec{b}, 1\right] - \frac{1}{2}\Pr_x\left[Q(\vec{z}_{\frac{x}{i}}, \vec{\ell}) \wedge c(\vec{z}_{1\ldots i-1}) = \vec{b}\right]\right|\right].$$

We can see that the above quantity is smaller than the left-hand side of (2) and thus less than 2ϵ by considering it as an average over various $\vec{z}$. For those $\vec{z}$ for which $c(\vec{z}_{1\ldots i-1}) = \vec{b}$, the terms are the same, and for the rest of the terms, the above quantity is 0. Next, because $|E[R]| \le E[|R|]$ for any random variable R,

$$\left|\mathbf{E}_{\vec{z}}\left[\Pr_x\left[Q(\vec{z}_{\frac{x}{i}}, \vec{\ell}) \wedge c(\vec{z}_{1\ldots i-1}, x) = \vec{b}, 1\right] \right.\right.$$
$$\left.\left. -\frac{1}{2}\Pr_x\left[Q(\vec{z}_{\frac{x}{i}}, \vec{\ell}) \wedge c(\vec{z}_{1\ldots i-1}) = \vec{b}\right]\right]\right| < 2\epsilon.$$

Since z_i and x are from the same distribution,

$$\left|\Pr_{\vec{z}}[Q(\vec{z}, \vec{\ell}) \wedge c(\vec{z}_{1\ldots i}) = \vec{b}, 1] - \frac{1}{2}\Pr_{\vec{z}}[Q(\vec{z}, \vec{\ell}) \wedge c(\vec{z}_{1\ldots i-1}) = \vec{b}]\right| < 2\epsilon.$$

By a straightforward inductive argument on i, we conclude that for every $\vec{b} \in \{0, 1\}^k$,

$$\left|\Pr_{\vec{z}}[Q(\vec{z}, \vec{\ell}) \wedge c(\vec{z}) = \vec{b}] - \frac{1}{2^k}\Pr_{\vec{z}}[Q(\vec{z}, \vec{\ell})]\right| < 4\epsilon\left(1 - \frac{1}{2^k}\right).$$

This fact now allows us to estimate our desired k-wise query $\Pr_{\vec{z}}[Q(\vec{z}, c(\vec{z}))]$. In particular,

$$\Pr_{\vec{z}}[Q(\vec{z}, c(\vec{z}))] = \sum_{\vec{\ell} \in \{0,1\}^k} \Pr_{\vec{z}}[Q(\vec{z}, \vec{\ell}) \wedge c(\vec{z}) = \vec{\ell}].$$

We approximate each of the $2^k = \text{poly}(n)$ terms corresponding to a different $\vec{\ell}$ by using *unlabeled* data to estimate $\frac{1}{2^k}\Pr_{\vec{z}}[Q(\vec{z}, \vec{\ell})]$. Adding up these terms gives us a good estimate of $\Pr_{\vec{z}}[Q(\vec{z}, c(\vec{z}))]$ for the chosen $\epsilon = \tau/2^{k+2}$, with high probability. □

4.1. DISCUSSION. In the above proof, we saw that either the data is statistically "homogeneous" in a way that allows us to simulate the original learning algorithm with unary queries, or else we discover a "heterogeneous" region that we can exploit with an alternative learning algorithm using only unary queries. Thus any concept class that can be learned from $O(\log n)$-wise queries can also be weakly learned from unary queries. Note that Aslam and Decatur [1998a] have shown that weak-learning statistical query algorithms can be boosted to strong-learning algorithms, if they weak-learn over *every* distribution. Thus, any concept class which can be (weakly or strongly) learned from $O(\log n)$-wise queries over *every* distribution can be strongly learned over every distribution from unary queries.

It is worth noting here that k-wise queries can be used to solve the length-k parity problem. One simply asks, for each $i \in \{1, \ldots, k\}$, the query: "what is the probability that k random examples form a basis for $\{0, 1\}^k$ and, upon performing Gaussian elimination, yield a target concept whose ith bit is equal to 1?" Thus, k-wise queries cannot be reduced to unary queries for $k = \omega(\log n)$. On the other hand, it is not at all clear how to simulate such queries in general from noisy examples.

5. *Conclusion*

In this article, we demonstrate a separation between the set of problems efficiently learnable from noisy data, and the set of problems learnable in the Statistical Query model. We do this by producing a slightly sub-exponential time algorithm for learning parity functions in the presence of random noise. By scaling down the size of the functions, we get a specific parity problem that can be PAC-learned from noisy data in time poly(n), as compared to time $n^{\Omega(\log \log n)}$ for the best SQ algorithm. Even though this separation is small, it suggests the possibility of interesting new noise-tolerant PAC-learning algorithms which go beyond the SQ model.

Corollary 3 demonstrates that we can solve the length-n parity learning problem in time $2^{o(n)}$. However, it must be emphasized that we accomplish this by using $2^{O(n/\log n)}$ labeled examples. From the point of view of coding theory, it would be useful to have an algorithm that takes time $2^{o(n)}$ but uses only poly(n) or even $O(n)$ examples. We do not know if this can be done. Also of interest is the question of whether our time-bound can be improved from $2^{O(n/\log n)}$ to, say, $2^{O(\sqrt{n})}$.

It would also be desirable to reduce our algorithm's dependence on η. The dependence comes from Lemma 4, with $s = 2^{a-1}$. For instance, consider the problem of learning parity functions that depend on the first k bits of input for $k = O(\log n \log \log n)$. In this case, if we set $a = \lceil \frac{1}{2} \log \log n \rceil$ and $b = O(\log n)$ in Theorem 2, the running time is polynomial in n, with dependence on η of $(\frac{1}{1-2\eta})^{\sqrt{\log n}}$. This allows us to handle η as large as $1/2 - 2^{-\sqrt{\log n}}$ and still have polynomial running time. While this can be improved slightly, we do not know how to solve the length-$O(\log n \log \log n)$ parity problem in polynomial time for η as large as $1/2 - 1/n$ or even $1/2 - 1/n^{\varepsilon}$. What makes this interesting is that it is an open question (Kearns, personal communication) whether noise tolerance can in general be boosted; this example suggests why such a result may be nontrivial.

In the second half of this article, we examine an extension to the SQ model in which one is allowed queries of arity k. We have shown that for $k = O(\log n)$, any concept class learnable in the SQ model with k-wise queries is also (weakly) learnable with unary queries. On the other hand, the results of Blum et al. [1994] imply this is not the case for $k = \omega(\log n)$. An interesting open question is whether every concept class learnable from $O(\log n \log \log n)$-wise queries is also PAC-learnable in the presence of classification noise. If so, then this would be a generalization of the first result of this paper.

REFERENCES

AJTAI, M., KUMAR, R., AND SIVAKUMAR, D. 2001. A sieve algorithm for the shortest lattice vector problem. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*. ACM, New York.

ANGLUIN, D., AND LAIRD, P. 1988. Learning from noisy examples. *Mach. Learn. 2*, 4, 343–370.

ASLAM, J. A., AND DECATUR, S. E. 1998a. General bounds on statistical query learning and PAC learning with noise via hypothesis boosting. *Inf. Comput. 141*, 2 (Mar.), 85–118.

ASLAM, J. A., AND DECATUR, S. E. 1998b. Specification and simulation of statistical query algorithms for efficiency and noise tolerance. *J. Comput. Syst. Sci. 56*, 2 (Apr.), 191–208.

BLUM, A., FURST, M., JACKSON, J., KEARNS, M., MANSOUR, Y., AND RUDICH, S. 1994. Weakly learning DNF and characterizing statistical query learning using fourier analysis. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing* (May). ACM New York, pp. 253–262.

DECATUR, S. E. 1993. Statistical queries and faulty PAC oracles. In *Proceedings of the 6th Annual ACM Workshop on Computational Learning Theory*. ACM, New York.

DECATUR, S. E. 1996. Learning in hybrid noise environments using statistical queries. In *Learning from Data: Artificial Intelligence and Statistics V*, D. Fisher and H.-J. Lenz, Eds. Springer-Verlag, New York.

HELMBOLD, D., SLOAN, R., AND WARMUTH, M. 1992. Learning integer lattices. *SIAM J. Comput. 21*, 2, 240–266.

MACWILLIAMS, F., AND SLOANE, N. 1977. The Theory of Error-Correcting Codes. North-Holland, Amsterdam, The Netherlands.

JACKSON, J. 2000. On the efficiency of noise-tolerant PAC algorithms derived from statistical queries. In *Proceedings of the 13th Annual Workshop on Computational Learning Theory.*

KEARNS, M. 1998. Efficient noise-tolerant learning from statistical queries. *J. ACM*, *45*, 6 (Nov.), 983–1006.

KUMAR, R., AND SIVAKUMAR, D. 2001. On polynomial approximations to the shortest lattice vector length. In *Proceedings of the 12th Annual Symposium on Discrete Algorithms*.

LITTLESTONE, N. 1988. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Mach. Learn. 2*, 285–318.

LITTLESTONE, N. 1989. From online to batch learning. In *Proceedings of the 2nd Annual ACM Conference on Computational Learning Theory*. ACM, New York, pp. 269–284.

WAGNER, D. 2002. A generalized birthday problem. In *Proceedings in Advances in Cryptology—CRYPTO 2002.* Lecture Notes in Computer Science, vol. 2442. Springer-Verlag, New York, pp. 288–304.

RECEIVED OCTOBER 2000; REVISED NOVEMBER 2002; ACCEPTED NOVEMBER 2002