Noise-tolerant learning, the parity problem, and the Statistical Query model

Avrim Blum Adam Kalai Hal Wasserman

School of Computer Science
Carnegie Mellon Univeristy

PRELIMINARY DRAFT

Abstract

We describe a slightly sub-exponential time algorithm for learning parity functions in the presence of random classification noise. By applying this algorithm to the restricted case of parity functions that depend on only the first $O(\log n \log\log n)$ bits of input, we achieve the first known instance of a polynomial-time noise-tolerant learning algorithm for a concept class that is provably not learnable in the statistical query model of Kearns [Kea93]. 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.

A natural extension of the statistical query model is to allow queries about statistical properties that involve k-tuples of examples (as opposed to single examples). The second result of this paper is to show that any class of functions learnable (strongly or weakly) with k-wise queries for $k = O(\log n)$ 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.

1 Introduction

A central focus of the theoretical study of machine learning is the question: "What interesting classes of functions can be learned in polynomial time?" An equally important question, given the reality that most learning scenarios do not perfectly follow simple functional forms, is: "What classes of functions can be learned efficiently from noisy, imperfect data?" The statistical query (SQ) framework of Kearns [Kea93] is a useful, elegant model for addressing the latter question. This 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 classification noise* in the standard PAC model. (This result has been extended to more general forms of noise as well [Dec93, Dec96].) 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 almost all known learning algorithms can be formulated as SQ algorithms, and hence can be made noise-tolerant.

An interesting feature of the SQ model is that one can characterize information-theoretically (i.e., without complexity assumptions) what kinds of problems can be learned from statistical queries in (non-uniform) polynomial time [BFJ+94]. For example, the class of parity functions, which can be efficiently learned from non-noisy data in the PAC model, provably cannot be efficiently learned in the SQ model under the uniform distribution. Thus, there are problems that can be PAC-learned from non-noisy data but cannot be learned by statistical queries. On the other hand, there is no known efficient algorithm for PAC-learning parity functions in the presence of random classification noise. (This open problem is closely related to the classic coding-theory problem of decoding random linear codes.) In fact, up to now there has been no known example — not even a contrived one — of a problem learnable with random classification noise in the PAC model but not learnable by statistical queries.

In this paper, 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. 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 can also be viewed as a slightly sub-exponential 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.) 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.

The second result of this paper 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 paper, it is natural to ask whether allowing k-wise queries, for some small value of k, might increase the set of SQ-learnable functions.

We however show 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 [BFJ+94] 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.

2 Definitions and Preliminaries

A *concept* is a boolean function on an *input space*, which in this paper will generally be $\{0,1\}^n$. The output of a concept is either 0 (negative) or 1 (positive). A *concept class* is a set of concepts. We will be considering the problem of learning a target concept in the presence of *random classification noise* [AL88]. 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. (I.e., $\ell = c(x)$ 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: that is, a hypothesis function h such that $\Pr_{x \leftarrow \mathcal{D}}[h(x) = c(x)] \geq 1 - \epsilon$.

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., 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 (that is, 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 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 $1/\epsilon$.

We will 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* $\epsilon < 1/2 - 1/\text{poly}(n)$, $\mathcal{A}$ 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 non-noisy data. However, statistical queries can be simulated from data corrupted by random classification noise [Kea93]. 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 [AD93, AD98], but they are all polynomially related.

[1] Normally, one would also require polynomial dependence on $1/(1/2 - \eta)$ — in part because normally 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 $1/(1/2 - \eta)$.

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

[BFJ+94] prove that any concept class containing more than polynomially many pairwise uncorrelated functions cannot be learned even weakly in the statistical query model. Specifically, they show the following.

Definition 1 (Def. 2 of [BFJ+94]) *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 $i \neq j$,*

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

Theorem 2 (Thm. 12 of [BFJ+94]) *If $d =$ SQ-DIM$(C, \mathcal{D})$, then either the number of queries or $1/\tau$ must be at least $\frac{1}{2}d^{1/3}$ to learn C to error less than $1/2 - 1/d^3$ in the SQ model.*

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 $1/\text{poly}(n)$ tolerance. But we will 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

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. We will find it convenient to write k as the product ab (where a and b may be any desired quantities). In this case, we think of each vector in $\{0,1\}^{ab}$ as divided into *a blocks*, each b bits long. Our main result is the following.

Theorem 3 *The length-ab parity problem for noise rate η can be solved by an algorithm whose number of samples and total computation-time are* $\text{poly}\left(\left(\frac{1}{1-2\eta}\right)^{2^a}, 2^b\right)$.

Plugging in particular values for a, b, and η then allows us to derive useful corollaries. In particular, the following two corollaries are closely related. Note that Corollary 4 satisfies our main goal of an efficient noise-tolerant algorithm for a class provably not efficiently learnable in the SQ model.

Corollary 4 *Using* $a = \lceil \lg\lg n \rceil$, $b = O(\log n)$ *in Theorem 3, we find that the parity problem of length* $O(\log n \log\log n)$*, for noise rate* η *equal to any constant less than* $1/2$*, can be solved with number of samples and total computation-time* $\mathrm{poly}(n)$*. Equivalently, we have found a polynomial-time algorithm for learning noisy parity functions over* $\{0,1\}^n$ *that depend on only the first* $O(\log n \log\log n)$ *bits of input.*

Corollary 5 *Using* $a = \lceil \lg n/2 \rceil$, $b = \lceil 2n/\lg n \rceil$ *in Theorem 3, we find that the length-n parity problem, for noise rate* η *equal to any constant less than* $1/2$*, can be solved with number of samples and total computation-time* $2^{O(n/\log n)}$*.*

We begin our proof of Theorem 3 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 6 *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 $(1-\eta)(\frac{1}{2} + \frac{1}{2}(1-2\eta)^{s-1}) + \eta(\frac{1}{2} - \frac{1}{2}(1-2\eta)^{s-1}) = \frac{1}{2} + \frac{1}{2}(1-2\eta)^s$. The lemma then follows by induction. ■

The idea for our 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 6, 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.

Recall that we are viewing each example vector as consisting of a blocks, each b bits long. We then introduce the following notation.

Definition 7 *Let* V_i *be the subspace of* $\{0,1\}^{ab}$ *consisting of those vectors whose last* i *blocks have all bits equal to zero. Then 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 8 *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 $x_1, \ldots, x_s$. In these vectors, blocks $a-i+1, \ldots, a$ are all zero. Partition $x_1, \ldots, x_s$ 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 $u_1, \ldots, u_{s'}$, where $s' \geq s - 2^b$ (since we discard at most one vector per class).

What can we say about $u_1, \ldots, u_{s'}$? 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}. Secondly, 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 $u_1, \ldots, u_{s'}$ thus form the desired $(i+1)$-sample. ■

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

Proof of Theorem 3: Draw $a2^b$ labeled examples. Observe that these qualify as a 0-sample. Now apply Lemma 8, $a-1$ 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 uniform-randomly over V_{a-1}, and notice that V_{a-1} contains only 2^b distinct vectors, one of which is $(1, 0, \ldots, 0)$. Hence there is an approximately $1 - 1/e$ chance that $(1, 0, \ldots, 0)$ appears in our $(a-1)$-sample. If this does not occur, we repeat the above process with new labeled examples. Note that the expected number of repetitions is only constant.

Now, unrolling our applications of Lemma 8, observe that we have written the vector $(1, 0, \ldots, 0)$ as the sum of 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 6. Hence the sum of the labels gives us the correct value of $(1, 0, \ldots, 0) \cdot c$ with probability $\frac{1}{2} + \frac{1}{2}(1-2\eta)^{2^{a-1}}$.

This means that, if we repeat the above process $\text{poly}((\frac{1}{1-2\eta})^{2^a}, b)$ times, using new labeled examples each time, we can determine $(1, 0, \ldots, 0) \cdot c$ with probability of error exponentially small in ab. In other words, we can determine the first bit of c with very high probability. And of course, by cyclically shifting all examples, the same algorithm may be employed to find each bit of c. Thus, with high probability we can determine c using number of examples and total computation-time $\text{poly}((\frac{1}{1-2\eta})^{2^a}, 2^b)$. ■

3.1 Extension to other distributions

While the uniform distribution is in this case the most interesting, in fact we can extend our algorithm to work over any distribution. In this case, instead of actually producing the target parity function, the algorithm, given a test example, will directly write that test example as a sum of a small number of the given training examples, thereby producing a prediction of its label. (We then can do this multiple times using different portions of the training data to increase accuracy.) One way to view this algorithm — and, in fact, our algorithm for the uniform case — is as a kind of Gaussian elimination in which each entry in the matrix, rather than being an element of the field $\mathbf{F}_2$, is an element of the vector space $\mathbf{F}_2^b$. In particular, instead of choosing a row that begins with a 1 and subtracting it from all other such rows, what we do is choose one row for each initial b-bit block observed: we then use these (at most $2^b - 1$) rows to zero out all the others. We then move on to the next b-bit block. If we think of this as an online algorithm, then each new example seen either gets captured as a new row in the matrix (and there are at most $a(2^b - 1)$ of them) or else it passes all the way through the matrix and is given a prediction. Details will appear in the full paper.

3.2 Discussion

Corollary 5 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. For the point of view of coding theory, it would be useful to have an algorithm which takes time $2^{o(n)}$ and number of examples $\text{poly}(n)$ or even $O(n)$. 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, for example, $2^{O(\sqrt{n})}$.

It would also be desirable to reduce our algorithm's dependence on η. This dependence comes from Lemma 6, with $s = 2^{a-1}$. Thus, in the case of Corollary 4, where $a = \lceil \lg \lg n \rceil$ and $b = O(\log n)$, this dependence is $(\frac{1}{1-2\eta})^{\log n} = n^{\log(1/(1-2\eta))}$; so we attain polynomial time only if η is a constant less than 1/2. However, we find that a simple adjustment of parameters allows for the case $\eta = 1/2 - 1/\text{polylog}(n)$. Indeed, as the following corollary shows, we can do somewhat better, allowing for any noise rate η of form $1/2 - 2^{-(\log n)^{1-\varepsilon}}$.

Corollary 9 *Let ε be any positive constant. Using $a = \lceil \varepsilon \lg \lg n \rceil$, $b = O(\log n)$ in Theorem 3, we find that the parity problem of length $O(\log n \log \log n)$, for noise rate $\eta \leq 1/2 - 2^{-(\lg n)^{1-\varepsilon}}$, can be solved with number of samples and total computation-time* $\text{poly}(n)$.

4 Limitations 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: i.e., from probabilities of form $\Pr_{x_1,x_2}[Q(x_1, c(x_1), x_2, c(x_2))]$. 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.

Theorem 10 *Let $k = O(\log n)$, and assume that there exists a* $\text{poly}(n)$*-time algorithm using k-wise statistical queries which weakly learns a concept class C under distribution $\mathcal{D}$. That is, this algorithm learns from approximations of probabilities of form $\Pr_{x_1,\ldots,x_k}[Q(x_1, c(x_1), \ldots, x_k, c(x_k))]$, where Q is a polynomially evaluable predicate. Then there exists a* $\text{poly}(n)$*-time algorithm which weakly learns the same class using only unary queries.*

Sketch of proof: The original algorithm has access to approximations, correct to plus-or-minus any desired polynomial fraction, of probabilities of form $\Pr_{x_1,\ldots,x_k}[Q(x_1, c(x_1), \ldots, x_k, c(x_k))]$ (where all probabilities are over the given distribution $\mathcal{D}$). We now consider the problem of simulating such a k-wise query using only unary queries. What we will show is that either our simulation succeeds, or else in failing it finds a unary query which distinguishes the target function from random; in the latter case, the discovered query can then be used directly for weak-learning.

The above probability can be rewritten as:

$$\sum_{\ell_1,\ldots,\ell_k\in\{0,1\}} \Pr_{x_1,\ldots,x_k}\left[c(x_1)=\ell_1 \,\&\ldots\&\, c(x_k)=\ell_k \,\&\, Q(x_1,\ell_1,\ldots,x_k,\ell_k)\right]. \tag{1}$$

This is a sum of $2^k = \mathrm{poly}(n)$ probabilities, so we can approximate each constituent probability separately. Hence let us fix $\ell_1,\ldots,\ell_k$ and focus hereafter on approximating a probability of form:

$$\Pr_{x_1,\ldots,x_k}\left[c(x_1)=\ell_1 \,\&\ldots\&\, c(x_k)=\ell_k \,\&\, Q(x_1,\ell_1,\ldots,x_k,\ell_k)\right]. \tag{2}$$

This probability can in turn be rewritten as the product of $k+1$ constituent probabilities, namely:

$$\Pr_{x_1,\ldots,x_k}\left[Q(x_1,\ell_1,\ldots,x_k,\ell_k)\right]\cdot$$
$$\prod_{i=1}^{k} \Pr_{x_1,\ldots,x_k}\left[c(x_i)=\ell_i \mid c(x_1)=\ell_1 \,\&\cdots\&\, c(x_{i-1})=\ell_{i-1} \,\&\, Q(x_1,\ell_1,\ldots,x_k,\ell_k)\right].$$

Once again, we will approximate the constituent probabilities individually. We start by approximating $\Pr_{x_1,\ldots,x_k}[Q(x_1,\ell_1,\ldots,x_k,\ell_k)]$ (this is easy, as the probability does not depend on c), then proceed in order from $i=1$ up to k. If at any point the product of the probabilities calculated so far is very small, we halt, returning zero as our approximation for (2).

Hereafter we fix i and focus on approximating a probability of form:

$$\Pr_{x_1,\ldots,x_k}\left[c(x_i)=\ell_i \mid c(x_1)=\ell_1 \,\&\cdots\&\, c(x_{i-1})=\ell_{i-1} \,\&\, Q(x_1,\ell_1,\ldots,x_k,\ell_k)\right]. \tag{3}$$

To approximate this value, we sample from $\mathcal{D}$ to generate $\bar{z}^{(1)},\ldots,\bar{z}^{(t)}$, a list of $(k-1)$-tuples of unlabeled examples, where t is of large polynomial size. Each $\bar{z}^{(j)}$ is of the form $(z_1^{(j)},\ldots,z_{i-1}^{(j)},z_{i+1}^{(j)},\ldots,z_k^{(j)})$, where each $z_m^{(j)}$ is a $\mathcal{D}$-random unlabeled example. (We think of each $\bar{z}^{(j)}$ as specifying values for all of $x_1,\ldots,x_k$ except x_i.) Corresponding to each $\bar{z}^{(j)}$ we introduce probabilities $S^{(j)}$ and $T^{(j)}$, defined as follows:

$$\begin{aligned} S^{(j)} &:= \Pr_{x_i}\left[Q(z_1^{(j)},\ell_1,\ldots,x_i,\ell_i,\ldots,z_k^{(j)},\ell_k)\right], \\ T^{(j)} &:= \Pr_{x_i}\left[c(x_i)=\ell_i \,\&\, Q(z_1^{(j)},\ell_1,\ldots,x_i,\ell_i,\ldots,z_k^{(j)},\ell_k)\right]. \end{aligned}$$

Note that we can efficiently approximate each of the probabilities $S^{(j)}$ and $T^{(j)}$: indeed, $S^{(j)}$ does not depend on the target concept, while $T^{(j)}$ requires only a unary query. Now consider the fraction

$$\frac{\sum_{j\in\mathcal{R}} T^{(j)}}{\sum_{j\in\mathcal{R}} S^{(j)}}, \tag{4}$$

where the summation is over $\mathcal{R} = \{j\colon c(z_m^{(j)}) = \ell_m \text{ for all } m<i\}$. Note that our algorithm cannot tell which j belong to $\mathcal{R}$ and which do not, because we do not have direct access to c; nonetheless, we may assume that $|\mathcal{R}|$ is not too small and indeed that the denominator of this fraction is not close to zero. The reason is that if this denominator were small, that would (with high probability) imply that $\Pr_{x_1,\ldots,x_k}[c(x_1)=\ell_1 \,\&\ldots\&\, c(x_{i-1})=\ell_{i-1} \,\&\, Q(x_1,\ell_1,\ldots,x_k,\ell_k)]$ is small, which would have caused our algorithm to halt before reaching this point. But observe that, if the denominator

is not too small and t is of sufficiently large polynomial size, (4) will with high probability be a good approximation for (3). We now distinguish two cases:

Case I: we find that, for all $j \in \{1, \ldots, t\}$, $|2T^{(j)} - S^{(j)}|$ is small. Then the value of (4) is approximately 1/2. (We know this to be true even though we do not know which values of j are in $\mathcal{R}$.) Hence we may return 1/2 as our approximation for (3).

Case II: we find some j such that $|2T^{(j)} - S^{(j)}|$ is large. This means that we have discovered a significantly large region of $\mathcal{D}$-random instances, namely $\{x\colon Q(z_1^{(j)}, \ell_1, \ldots, x, \ell_i, \ldots, z_k^{(j)}, \ell_k)\}$, over which the probability that $c(x) = 1$ is skewed away from 1/2. But then we can abandon our effort to simulate the original learning algorithm, and can instead use this new information to directly predict the value of $c(x)$ with probability significantly greater than 1/2. ∎

4.1 Discussion

In the above proof, we saw that either the data is statistically "homogeneous" in a way which allows us to simulate the original learning algorithm with unary queries, or else we discover a "heterogeneous" region which 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 [AD93] 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 paper we have addressed the classic problem of learning parity functions in the presence of random noise. We have shown that parity functions over $\{0,1\}^n$ can be learned in slightly sub-exponential time, but only if many labeled examples are available. It is to be hoped that future research may reduce both the time-bound and the number of examples required.

Our result also applies to the study of statistical query learning and PAC-learning. We have given the first known noise-tolerant PAC-learning algorithm which can learn a concept class not learnable by any SQ algorithm. The separation we have established between the two models is rather small: we have shown that a specific parity problem can be PAC-learned from noisy data in time poly(n), as compared to time $n^{O(\log \log n)}$ for the best SQ algorithm. This separation may well prove capable of improvement and worthy of further examination. Perhaps more importantly, the topic of interesting new noise-tolerant PAC-learning algorithms which go beyond the SQ model is now open for study.

We have also examined an extension to the SQ model in terms of allowing 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 [BFJ+94] 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

[AD93] J. A. Aslam and S. E. Decatur. General bounds on statistical query learning and PAC learning with noise via hypothesis boosting. In *Proceedings of the 34th Annual Symposium on Foundations of Computer Science*, pages 282–291, November 1993.

[AD98] J. A. Aslam and S. E. Decatur. Specification and simulation of statistical query algorithms for efficiency and noise tolerance. *Journal of Computer and System Sciences*, 56(2):191–208, April 1998.

[AL88] D. Angluin and P. Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

[BFJ+94] 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 26th Annual ACM Symposium on Theory of Computing*, pages 253–262, May 1994.

[Dec93] S. E. Decatur. Statistical queries and faulty PAC oracles. In *Proceedings of the 6th Annual ACM Workshop on Computational Learning Theory*. ACM Press, 1993.

[Dec96] S. E. Decatur. Learning in hybrid noise environments using statistical queries. In D. Fisher and H.-J. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics V*. Springer Verlag, 1996.

[Kea93] M. Kearns. Efficient noise-tolerant learning from statistical queries. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 392–401, 1993.