Learning with Queries Corrupted by Classification Noise ⋆

Jeffrey Jackson [a] Eli Shamir [b,1] Clara Shwartzman [b,c]

[a] *Department of Mathematics and Computer Science, Duquesne University, Pittsburgh, PA 15282, USA*

[b] *Institute of Mathematics and Computer Science, The Hebrew University, Jerusalem 91904, Israel*

[c] *News Datacom Research, Jerusalem, Israel*

Abstract

Kearns introduced the "statistical query" (SQ) model as a general method for producing learning algorithms which are robust against classification noise. We extend this approach in several ways in order to tackle algorithms that use "membership queries", focusing on the more stringent model of "persistent noise". The main ingredients in the general analysis are:

(1) Smallness of dimension of the classes of both the target and the queries.

(2) Independence of the noise variables.

Persistence restricts independence, forcing repeated invocation of the same point x to give the same label. We apply the general analysis to get a noise-robust version of Jackson's Harmonic Sieve, which learns DNF under the uniform distribution. This corrects an error in his earlier analysis of noise tolerant DNF learning.

Key words: Machine Learning, Statistical Query, Noise Tolerance, Disjunctive Normal Form, Harmonic Sieve

⋆ Revised and expanded version of paper that appeared in Proceedings of the Fifth Israel Symposium on the Theory of Computing and Systems (1997).

[1] Partially supported by project No. I-403-001.06/95 of the German-Israeli Foundation for Scientific Research [GIF].

1 Introduction

How can one learn concepts from examples? Sampling techniques in statistics indicate a way: A teacher provides the information $\{(x, \ell = f(x))\}_{x \in S}$, where f is the target concept and S is a large enough random sample from the domain X on which the concepts are defined. Using this information, the learning device (or algorithm) computes an approximation concept h. The imprecision of h is measured by the "generalization error" $\mathbf{D}(h \neq f)$, which denotes the probability of the event $\{h \neq f\}$ with respect to the distribution $\mathbf{D}$ on the domain X.

These ingredients are the essence of the "probably approximately correct" [PAC] learning paradigm. However, there are several modes of PAC learning, with significant differences between them. In passive sampling, the independent sample S of size m is distributed according to $\mathbf{D}^m$, where $\mathbf{D}$ is the basic underlying distribution used to measure the generalization error ($\mathbf{D}$ need not be known explicitly to the learner), and no other way of getting examples is allowed.

The ability to select sample points *actively*, possibly at random using a specific distribution $\mathbf{P}$ on X, usually carries the generic name of "membership querying." Such PAC learning with active querying may well have a meaningful power gap—in extent and speed—over passive sampling. This has been proved in some cases (*e.g.*, assuming hardness of cryptographic primitives [15–17]).

In other cases, such as the DNF learning problem discussed below, the existence of this power gap has not been settled. It is worth remarking that, in some cases, with passive sampling one can simulate (with reasonable cost) certain other sampling distributions which figure in the learning algorithm (*e.g.*, by filtering [7,20]). So formal proofs of lower bounds showing a power gap are usually not easy.

In noise-robust learning, the task is to get good approximations to the noise-free target f even if the examples (passive or active) are corrupted by some noise. The main contribution of this article is in rendering certain active-query learning algorithms robust under "persistent classification noise." We adapt Kearns' technique [13], which works for algorithms that are cast to use statistical queries [SQ] *only*. An SQ query is an evaluation of an expectation $\mathbf{P}G(x, f(x))$, where G is a $\{0,1\}$-valued function. The value $\mathbf{P}G$ is required to be precise within a tolerance τ which is not very demanding. This allows an evaluation of $\mathbf{P}G$ by a moderate number of queries at random points.

Kearns' original formulation of the SQ model requires the SQ evaluation distribution $\mathbf{P}$ to coincide with the error measuring distribution $\mathbf{D}$, which clearly corresponds to passive sampling in PAC algorithms. To treat noise in *active* (membership) *querying*, we found it useful to extend the form of SQs allowed:

(1) Allow "second order" queries. This means that each PAC example consists of a pair $(x, y) \in X^2$ with label $\ell = (f(x), f(y))$. The corresponding SQ form is an expectation $\mathbf{Q}G(x, y, f(x), f(y))$.
Moreover

(2) The distribution $\mathbf{Q}$ over X^2 used in the SQs is arbitrary. Specifically, it is not necessarily a product distribution.

SQs can be—and often are—viewed as coefficients of f with respect to some basis of a space of functions over X. Passage to sample-based PAC learning involves simultaneous evaluation, within a tolerance τ, of all the SQs in the algorithm by a single large enough sample. Clearly, the natural tool to use is a Vapnik-Chervonenkis type result on "uniform approximations" of a family of expectations by empirical means over a sample.

For the class $\{\mathbf{P}G(f)\}$ we need the uniform approximation in the form given by Haussler [10], assuming finiteness of the "dimensions" of the class of targets and of the class of functions G figuring in the SQs. These are, respectively, the VC dimension (for the targets) and combinatorial dimensions (for the G-class). We do not give the definitions of dimensions here since for the specific applications (in Section 5) only one fact is needed: the dimension of a class is upper-bounded by the logarithm of its cardinality. Moreover, uniform approximation based on cardinality is also given by Haussler [10].

Now suppose that the examples are corrupted by classification noise. It is possible to transfer the effect of the noise to the evaluation of the SQ, say $\mathbf{P}G(x, f(x))$, while keeping the target $f(x)$ noise-free. In fact we show in Section 3 how to compensate for the bias (to the expected value) and accommodate the variance (caused by the random noise process) within the tolerance τ.

Noise analysis of second-order membership querying, in PAC or SQ form, forces one to closely scrutinize the standard working assumption on classification noise, namely that it affects a random label flip to each x, *independent* over all $x \in X$. What happens if the point x appears twice or more in a sample? To stipulate that the first encountered label *persists* is more realistic, but more difficult to tackle. We argue that an essential difference in "persistent classification noise" arises only when the sample point is (x, x). In Section 4 we show how to accommodate this case, and argue that other collisions usually have negligible effect.

In Section 5 we discuss, at some length, two applications: persistent noise-robustness of a "weak parity" algorithm [9] and of the "harmonic sieve" algorithm for learning DNF [11].

2 Statistical queries, empirical evaluation

Learning algorithms which use only the values of statistical queries can be made noise robust by offsetting the noise effect on the SQs. This is the basic claim. To complete the picture, one has to evaluate all the values of the SQs (provided their number is reasonable) by a sufficiently large random sample. This is done here after giving a definition of SQs.

Let $(X, \mathbf{D})$ be a domain with probability distribution $\mathbf{D}$, Φ a class of ± 1-valued target functions on X. A statistical query (SQ) is given by:

(i) A functional $G(x, \ell)$ mapping $X \times \{-1, 1\}$ to $[-M, M]$;
(ii) A distribution $\mathbf{P}$ on X and a tolerance parameter τ; $(\mathbf{P}, \tau)$ will be common to a family of SQs indexed by I, $Q = \{G_i, i \in I\}$. Note that some algorithms may require several distinct families of SQs.

An admissible value B of an SQ G on the target f should satisfy

$$|B - \mathbf{P}G(x, f(x))| \leq \tau, \tag{2.1}$$

Kearns' original definition restricts the range of G to $\{0, 1\}$ and identifies $\mathbf{P}$ with $\mathbf{D}$—the accuracy measuring distribution. The extension here allows more flexibility, especially in converting membership-queries to SQs, without much change in the noise-robustness proof.

An SQ Learning Algorithm $\mathbf{A}$ is one which interacts with the target $f \in \Phi$ only via a family (or several families) of SQs. In a query step, $\mathbf{A}$ poses a functional G and gets [from an "oracle"] an admissible value B of $\mathbf{P}Gf = \mathbf{P}G(x, f(x))$. $\mathbf{A}$ learns with $(1 - \epsilon)$-accuracy if upon termination it produces an hypothesis h such that $\mathbf{D}(f \neq h) < \epsilon$.

Usually, an SQ-algorithm $\mathbf{A}$ is deterministic. But one can attach a $\mathbf{P}$-random sampling algorithm. The idea is to get a τ-admissible value of $\mathbf{P}Gf$ by an *average* $E_S Gf$ over a finite sample S of size m, large enough so that for any $\mathbf{P}$ and all Gf which occur in the algorithm the approximation is uniformly τ-small.

Theorem 2.1 *Let $\{G(x, -1)\}$, $\{G(x, 1)\}$, $G \in Q$, be families of functionals with range $[-M, M]$, each family of pseudo-dimension $\leq q$ [19]. Assume the VC dimension of Φ is at most d. Then*

$$\mathbf{P}^m\{|E_S Gf - \mathbf{P}G(x, f(x))| < \tau\} \geq 1 - \delta \tag{2.2}$$

for all $f \in \Phi$, $G \in Q$, [$\mathbf{P}^m$ is the product probability distribution of samples of *size m], provided*

$$m \geq O[\varphi(d+q,\delta,\tau/M)], \quad \varphi(d,\delta,\tau) = \frac{d}{\tau^2}\log\frac{d}{\tau} + \frac{1}{\tau^2}\log\frac{1}{\delta}. \tag{2.3}$$

The big-O notation is the standard one. Also, we write 'Constant' for a positive value which does not depend on the other arguments in a given estimation.

PROOF. This is a claim of uniform approximation of expectations by sample means, which originated in the works of Vapnik. The sufficient sample size bound in (2.3) is valid for a family R of dimension $O(q+d)$. But the proof for the sample bound relies only on the moderate size of a τ-net of such R w.r.t. $L^1(\mathbf{P})$ norm [19,10, Theorems 2,7]. The family R we consider here is $Q(x,\Phi)$ *or equivalently* $\{G(x,\pm 1)\cdot I(f=\pm 1), G\in Q, f\in\Phi\}$, where $I(C)$ is the indicator function of C. A τ-net for it is obtained by the cross product of $\tau/2$-nets for the factors, so that the size corresponds indeed to that of a $Constant \cdot (d+q)$-dimensional family. This, as we noted, implies the bound (2.3) on the sample size. $\square$

Remark: The composition of an SQ algorithm with the $\mathbf{P}$-sampling module should give an output h such that

$$\mathbf{P}^m\{\mathbf{D}(h\neq f)\leq\epsilon\}\geq 1-\delta,$$

the tolerance τ for the SQ part will normally depend on ϵ, where $(1-\epsilon)$ is the accuracy (w.r.t. $\mathbf{D}$). If $\mathbf{P}=\mathbf{D}$ one gets the strict-PAC learning notions. If $\mathbf{P}\neq\mathbf{D}$, the algorithm is using membership queries.

3 Statistical queries, evaluation under noise

In this section we show how to offset the effect of noise on "simple" SQs, as defined in Section 2. The more complex situation with second order SQs will be dealt with in Section 4.

The basic feature of isotropic classification noise of uniform intensity $\eta<1/2$ is

(3.1) At each point $x\in X$, the value $f(x)$ is corrupted to $f(x)\cdot\xi(x)$, the random variable $\xi(x)$ is $(-1,1)$ with probabilities $(\eta,1-\eta)$.

It is possible to offset the bias created by this noise [1]:

$$E_{\xi(x)}G(x, f(x)\xi(x)) = (1-\eta)G(x, f(x)) + \eta G(x, -f(x)), \tag{3.2}$$

where G is the functional of a SQ. A similar relation (3.2′) holds for $-f$ in place of f. From (3.2) and (3.2′), the noise-free value $G(x, f(x))$ is eliminated and expressed by

$$G(x, f(x)) = E_{\xi(x)}G^{comp}(x, f(x)\cdot\xi(x)), \tag{3.3}$$

where

$$G^{comp}(x, \ell) = (1-2\eta)^{-1}[(1-\eta)G(x,\ell) - \eta G(x, -\ell)]. \tag{3.4}$$

(3.3) shows how to offset the bias, by passing from G to the "noise-compensating" functional G^{comp}. For the value of the SQ we have then

$$\mathbf{P}Gf = \mathbf{P}[E_\xi G^{comp}(f, \xi)]. \tag{3.5}$$

The variance issue of the noise comes up when (3.5) is approximated by a sample with noise-corrupted labels. To get concentration (w.h.p., within the tolerance τ), some form of independence of the noise variables $\xi(x)$ is required. For the easiest-to-treat model (model I in §4) we assume independence of the $\xi(x)$ for all (occurrences of) x in the *sample-sequence* S.

Theorem 3.1 *Assume the dimension conditions for Theorem 2.1. Let the noise variables $\xi(x)$ be independent for all x in the sample sequence S. Let*

$$\Delta = \left|\mathbf{P}G(x, f(x)) - \frac{1}{|S|}\sum_{y\in S} G^{comp}(y, f(y)\xi(y))\right|.$$

Then

$$\mathrm{Prob}_\xi \mathbf{P}^m \{\Delta > \tau\} < \delta, \tag{3.6}$$

holds uniformly for all $f \in \Phi$, $G \in Q$, provided

$$|S| = m = \Omega[\varphi(d+q, \delta, \tau(1-2\eta)/M], \tag{3.7}$$

φ given in (2.3).

PROOF. We decompose Δ into

$$\begin{aligned} &\mathbf{P}G(x, f(x)) - \tfrac{1}{m}\sum_{y\in S} G(y, f(y)) \\ &-\tfrac{1}{m}\sum_{y\in S}[G^{comp}(y, f(y)\xi(y)) - G(y, f(y))] = A + B \end{aligned} \tag{3.8}$$

where A comprises the first two terms and B is the rest. Note that ξ does not enter into A. Therefore, by Theorem 2.1, (3.7) implies that

$$\mathbf{P}^m\{|A| \geq \tau/2\} \leq \delta/2. \tag{3.9}$$

Now $B = \frac{1}{m}\sum_{k=1}^{m} b_k(\xi(y_k))$, where the terms in this average are independent, have 0 expectation (w.r.t. ξ) by (3.3), and have range of size $Constant(1-2\eta)^{-1}$. Using Hoeffding's inequality [19, Appendix B]

$$\mathrm{Prob}_\xi\{B \geq \tau/8\} \leq \exp\{-m\tau^2(1-2\eta)\cdot Constant\}. \tag{3.10}$$

This probability estimate holds for a single f. Multiplying both sides of (3.10) by the size $N(\tau/8, \ell^1[Q(\Phi)|_S])$ of $\tau/8$-net (cf. [10]), one gets a probability estimate for the event holding for *any* net element. Then upon relaxing $\tau/8$ to $\tau/2$ on the left hand side of (3.10) we get, uniformly for all $f \in \Phi$, $G \in Q$

$$\begin{aligned} &\mathrm{Prob}_\xi\{B \geq \tau/2\} \\ &\leq \exp\{ \quad -Constant\ m\tau^2(1-2\eta) \\ &\qquad\qquad +Constant\ (d+q)\log[(d+1)\tau(1-\eta)]\}, \end{aligned} \tag{3.11}$$

the positive term in the exponent comes from the size estimate for the $\tau/8$ net mentioned above. Now to make (3.10) less than $\delta/2$, m has to be large as stated in (3.7), and then (3.9) holds too, and the theorem is proved. □

Notice that estimating A involved only $\mathbf{P}^m$ of the noise-free sample, while that of B involved only the independence of the noise variables $\xi(x)$.

Theorem 3.1 holds also for *variable noise rate* $\eta(x)$. The compensating functional in (3.4), offsetting the noise bias is locally defined at x; the approximation of the SQs' values by noise-corrupted sample averages in (3.7), (3.8) have the same proof. A problem arises in implementing an *efficient* noise-robust algorithm when the noise level $\eta(x)$ is not known. If η is constant then, as Kearns [13] noted, the algorithm should be repeated with several sufficiently close different values of η. Then among the candidate outputs, the one with minimum disagreement with the given sample is chosen.

Noise-robustness proofs for Kearns' type SQ-algorithm were given before [13,1]. The proof of Theorem 3.1 here is more systematic and as we'll see, will capture extended SQs and more difficult noise models (including variable rate as noted above). In particular $\mathbf{P}$-sampling when $\mathbf{P} \neq \mathbf{D}$ falls under membership queries. In an extreme case where $\mathbf{P}$ is concentrated at x^* (a delta function), the $\mathbf{P}$-sample keeps asking for the value (label) at x^*. Then for model I below of "full sample independence," the correct noise-free label is simply obtained by majority. In such cases it is more realistic to consider the "persistent noise" model (model III below). In connection with second order queries, we also consider a pair-consistent model (model II below).

4 Second order queries, classification noise models

Queries of order $r = 2$ arise when a distribution $\mathbf{Q}$ on X^2 is used for sampling. The SQ form is

$$\mathbf{Q}Ff = \mathbf{Q}[F(x, y, f(x), f(y))]. \tag{1}$$

Such queries can express second moments—correlations between shifted values of f—and they are needed for the efficient learning algorithms we discuss in Section 5. The issue of noise-free uniform approximation of $\{\mathbf{Q}Ff\}$ is essentially the same for $r = 2$ (or $r \geq 2$) as was the approximation of $\{\mathbf{P}Ff\}$ for $r = 1$, *i.e.*, the analogue of Theorem 2.1 holds.

Note that one random draw from $\mathbf{Q}$ gives a pair (x, y), with corresponding example $(x, y, f(x), f(y))$. The noisy version of this example is denoted by $(x, y, f(x)\xi(x), f(y)\xi(y))$. To specify precisely a classification noise model for second-order queries, we distinguish three cases, based on the extent of independence of the noise variables $\xi(x)$, especially when the same x occurs at different places in the sample.

Model I: Independent. Any set of occurrences of $\xi(x)$ is independent.

Model II: Pair-consistent. If the pair (x, y) is drawn (according to the pair distribution $\mathbf{Q}$) and $x = y$, then $\xi(x) = \xi(y)$. In other words, the noisy example corresponding to such a pair is $(x, x, f(x)\xi, f(x)\xi)$, where ξ is the constant $+1$ or -1 chosen (once) according to the noise variable $\xi(x)$. The noise variables $\xi(x)$ are otherwise independent. Specifically, if a given x_0 occurs in two different pairs, the noise value ($+1$ or -1) applied to the label $f(x_0)$ may be different in each pair. Technically, we should write ξ as a function of a pair (x, y) in this model, but for notational simplicity we will continue to write $\xi(x)$ and $\xi(y)$ below with the understanding that when the noise is applied to an example $(x, y, f(x), f(y))$ with $x = y$ then $\xi(x) = \xi(y)$.

Model III: Persistent. Any set of occurrences of $\xi(x)$ with distinct values of x is independent. But once a given x_0 appears in *any* pair drawn, the value $\xi(x_0)$—and hence the label $f(x_0)\xi(x_0)$—persists for this example (so this model guarantees pair-consistency) *and* for all future examples containing x_0. As discussed further below, in this model we find it convenient to think of

$\xi(x)$ not as a random variable but instead as a randomly chosen *function* of x.

Thus each model allows (strictly) more dependence between the noise variables than the preceding model. Note also that while we will primarily be interested in these models as applied to second order examples, Models I and III both apply equally well to first order examples (Model II reduces to Model I in this case).

Our goal is to learn in Model III, the persistent classification noise model introduced (for first order examples) in [8]. This noise model is closely related to Model I, which (in its first order form) is a standard noise model for PAC learning without membership queries. However, Model I is not appropriate for membership query algorithms, in which a simple resampling strategy easily eliminates the noise effects. By making the classification noise persist, we again have a noise model which is nontrivial even if membership queries are available, but also a model for which analysis is feasible.

Persistent noise is a good model of much empirical research in machine learning, where the goal is often to find a relatively simple approximator to some unknown target function which is assumed to be much more complex than any function in the class of approximators. All that is desired in this setting is a reasonable approximator, not one that attempts to capture every nuance of the target. In our Model III, the goal is similarly to find a good approximation to the underlying noiseless function, rather than to the noisy function that we can query directly. The noisy examples can be considered to be produced by additional complexity in the target function that we choose to ignore for the sake of producing a simple approximator. Viewed this way, the persistent noise model could also be thought of as defining a type of *agnostic learning* [14], which has the general goal of finding the best approximator within a class of functions H to a target function which is not necessarily contained within H.

A subtle but important distinction between Models I and III is illustrated by the following technical difficulty encountered in Model III but not in I. Typically, in the PAC model we require that a learning algorithm succeed with probability $1 - \delta$ for any $\delta > 0$ (i.e., succeed with arbitrarily high confidence) at producing a good approximator to the target. This is generally required of the algorithm even if (non-persistent) classification noise is present. However, it is unreasonable to allow δ to be arbitrarily near zero in the persistent noise model, because in this model $\xi(x)$ is a random [noise] *function* and not a random variable. So, for example, if we are extremely unlucky in a given run of the learning algorithm and $\xi(x) = -1$ for all x (which occurs with probability $\eta^{2^n} > 0$), then we will be trying to learn the target f from an oracle for $\bar{f}$, a hopeless task given a class of approximators that is closed under complement. In Model III there is nothing a learning algorithm can do to avoid such a situation, unlike Model I, where in the limit a large enough set of examples will always "average out" the noise with arbitrarily high probability. Therefore, unlike standard PAC learning, we must impose a positive lower bound on δ in the persistent model, although this bound will be negligible.

Note that if a learning algorithm uses only first order SQs and the (first order) sampling distribution $\mathbf{P}$ has the property that in any $\mathbf{P}$-sample there is a negligible chance of a repetition of some x, then an algorithm that learns in model I will also learn in model III. An important case where this holds is when $\mathbf{P}$ is (roughly) uniform over a domain of exponential size.

For second order SQs, model II will replace model I in the approximation role. What differentiates models I and II in the second order setting is the case of equality ("diagonal draw") within a pair produced by a single draw from $\mathbf{Q}$. Our motivation for considering Model II is that, in our applications, the distributions $\mathbf{Q}$ are sometimes such that the chance of a diagonal draw occurring is non-negligible.

Model II is a useful intermediate between the other two models in the second order setting. As in the independent noise model, we can reasonably require a learning algorithm in this model to achieve arbitrarily high confidence, which simplifies the analysis. And, analogously with the first order case above, if the distributions $\mathbf{Q}$ used by a learning algorithm have the property that in any polynomial-size sample there is negligible chance that any two distinct pairs contain an identical member, then an algorithm that learns in the pair-consistent noise model will clearly also learn in the persistent noise model. It turns out that the two learning algorithms we consider in the Applications section use distributions having this property. Thus we will obtain persistent noise results for these algorithms by starting with a relatively clean analysis in the pair-consistent model.

The noise-compensating functional we will produce for 2nd order SQs under model II will assume different forms for $x = y$ and for $x \neq y$.

Let us write the four equations expressing

$$E_\xi F(x, y, \pm f(x)\xi(x), \pm f(y)\xi(y))$$

in terms of the four noise-free values of F. In the symmetric case where $F(\cdot, \cdot, \ell, \ell') = F(\cdot, \cdot, \ell', \ell)$ (which we assume below in view of the applications) there are just three equations expressing the responses A, B, C to queries to a noisy oracle in terms of the responses a, b, c to queries to a noiseless oracle. We must also consider the fact that the pair-consistent noise function ξ behaves differently in the case of a pair (x, y) with $x = y$ and the case $x \neq y$. So we define

$$
\begin{aligned}
a &= F(x, y, f(x), f(y)) \\
b &= F(x, y, -f(x), f(y)) \\
c &= F(x, y, -f(x), -f(y))
\end{aligned}
$$

$$
\begin{aligned}
A &= E_\xi F(x, y, f(x)\xi(x), f(y)\xi(y)) \\
B &= E_\xi F(x, y, -f(x)\xi(x), f(y)\xi(y)) \\
C &= E_\xi F(x, y, -f(x)\xi(x), -f(y)\xi(y))
\end{aligned}
$$

for an arbitrary pair (x, y), and

$$
\begin{aligned}
\alpha &= F(x, y, \ell, \ell') \\
\beta &= F(x, y, -\ell, \ell') \\
\gamma &= F(x, y, -\ell, -\ell')
\end{aligned}
$$

for fixed $\ell, \ell' \in \{-1, +1\}$.

Now we consider the two cases $x = y$ and $x \neq y$ separately, as the noise behaves differently in each case. In the first case, we obtain

$$
\begin{aligned}
A &= (1-\eta)a + \eta c \\
C &= \eta a + (1-\eta)c
\end{aligned} \tag{4.3}
$$

and in the second

$$
\begin{aligned}
A &= (1-\eta)^2 a + 2\eta(1-\eta)b + \eta^2 c \\
B &= ((1-\eta)^2 + \eta^2)b + \eta(1-\eta)(a+c) \\
C &= \eta^2 a + 2\eta(1-\eta)b + (1-\eta)^2 c.
\end{aligned} \tag{4.4}
$$

(4.3) can be solved as in the previous section, and solving (4.4) gives the value of the noiseless functional in terms of the expected values of the noisy functionals at a given point (x, y) such that $x \neq y$:

$$
a = \frac{1}{2(1-2\eta)} \left[A - C + \frac{[(1-\eta)^2 + \eta^2](A+C) - 4\eta(1-\eta)B}{1-2\eta} \right].
$$

Thus, analogously with the first-order analysis, we have that

$$
F(x, y, f(x), f(y)) = E_{\xi(x),\xi(y)} F^{comp}(x, y, f(x)\cdot\xi(x), f(y)\cdot\xi(y))
$$

where

$$F^{comp}(x, y, \ell, \ell') = \begin{cases} \frac{(1-\eta)\alpha - \eta\gamma}{1-2\eta} & \text{if } x = y \\ \frac{1}{2(1-2\eta)} \left[\alpha - \gamma + \frac{[(1-\eta)^2+\eta^2](\alpha+\gamma) - 4\eta(1-\eta)\beta}{1-2\eta}\right] & \text{otherwise.} \end{cases}$$

Theorem 4.1 *The analogue of Theorem 3.1 holds for SQ families of the second-order under the pair consistency noise model.*

Indeed, once we computed the compensating functional for this case, the proof of Theorem 3.1 carries over verbatim.

Summarizing, we have produced a second-order noise-compensating functional that can be used to simulate a statistical query oracle given a pair-consistent noise oracle. We have also noted that, for learning algorithms that adhere to certain query distribution constraints, this simulation can (with very high probability) be performed using a persistent noise oracle. In the remaining section we shall discuss applications to specific learning algorithms.

5 Applications

In this section we apply our techniques to developing noise-tolerant versions of two well-known membership query learning algorithms. The first of these is an algorithm originally presented by Goldreich and Levin [9] which we call the Weak Parity, or `WP`, algorithm (it has also been called the `KM` algorithm [2,21] by researchers in learning theory because it was first applied to prove learnability results by Kushilevitz and Mansour [18]). `WP` is essentially an agnostic learning algorithm [14] that finds the parity functions that are best correlated (with respect to the uniform distribution) with a given target function. The second algorithm is the Harmonic Sieve (`HS`), an algorithm that efficiently learns the class of DNF expressions with respect to the uniform distribution [11]. Our analysis corrects a deficiency in an earlier attempt at producing a noise-tolerant version of `HS` [11,12].

As both algorithms utilize Fourier analysis, we briefly discuss the multi-dimensional discrete Fourier transform before analyzing these learning algorithms.

5.1 *The Fourier transform*

For each set $A \subseteq \{1, \ldots, n\}$ we define the function $\chi_A : \{0,1\}^n \to \{-1, +1\}$ as

$$\chi_A(x) = (-1)^{\sum_{i \in A} x_i} = 1 - 2\left(\sum_{i \in A} x_i \bmod 2\right)$$

where x_i represents the ith bit in the instance x. That is, $\chi_A(x)$ is the $\{-1, +1\}$-valued function that is 1 when the parity of the bits in x indexed by A is even and is -1 otherwise. Every function $f : \{0,1\}^n \to \mathbf{R}$ can be uniquely expressed as a linear combination of parity functions: $f = \sum_A \hat{f}(A) \cdot \chi_A$, where $\hat{f}(A) = \mathbf{E}[f(x) \cdot \chi_A(x)]$ and the expectation is uniform over the instances x. We call the vector of coefficients $\hat{f}$ the *Fourier transform* of f. Note that for Boolean ($\{-1, +1\}$-valued) f, $\hat{f}(A)$ represents the correlation of f and χ_A with respect to the uniform distribution.

It can be shown that for any f and g mapping $\{0,1\}^n$ into the reals, $\mathbf{E}[fg] = \sum_A \hat{f}(A)\hat{g}(A)$. As a corollary of this we have Parseval's identity: $\mathbf{E}[f^2] = \sum_A \hat{f}^2(A)$. For Boolean f it follows that $\sum_A \hat{f}^2(A) = 1$. This implies that for any Boolean f and any $0 < \theta \leq 1$, $|\{A \mid \hat{f}(A) \geq \theta\}| \leq \theta^{-2}$. We call Fourier coefficients of a function f exceeding a threshold θ the θ-*heavy* Fourier coefficients of f.

5.2 *A noise-tolerant* `WP`

We now apply our noise-tolerance techniques to the Weak Parity algorithm. We begin with a brief discussion of the original algorithm. It should be noted that the algorithm is probabilistic—it uses sampling to obtain estimates of various quantities—and therefore has non-zero probability of failure. However, as with many learning algorithms, the running time of `WP` depends only inverse-logarithmically on the desired failure probability, and therefore the failure probability can effectively be made extremely small. Furthermore, we will develop an SQ version of the algorithm that fails with probability zero, as the SQ model specifies that with certainty we receive requested estimates from the SQ oracle within a specified error tolerance. Therefore, for expositional simplicity, we will ignore the possibility of failure of the `WP` algorithm in the following discussion.

Given a membership oracle for an arbitrary Boolean function f and a threshold $\theta > 0$, `WP` will find a set S of Fourier coefficients including all of the θ-heavy coefficients of f. Furthermore, all of the coefficients in S will be $(\theta/\sqrt{2})$-heavy.

The algorithm runs in time polynomial in n and θ^{-1}.

Conceptually, the algorithm is quite simple. It begins by partitioning the Fourier coefficients of the target function into two subsets each of size 2^{n-1}: those coefficients $\hat{f}(A)$ such that $1 \in A$ and those such that $1 \notin A$. It then estimates (as described below) the sum of squares of the coefficients in each subset. If either or both of these subsets has sum of squares greater than θ^2, the algorithm recurses on the subset(s) by partitioning each into two subsets according to whether or not $2 \in A$ and estimating the sum of squares of the coefficients in each resulting subset of size 2^{n-2}. This continues for n levels; at level n, we are testing the sum of squares of subsets of size 1 (*i.e.*, individual coefficients) against the threshold squared. Those coefficients that survive this test are the desired θ-heavy coefficients.

Because the magnitude of a Fourier coefficient of f represents the (possibly negative) correlation of the corresponding parity function with f, any parity function corresponding to a θ-heavy coefficient for θ inverse-polynomially large in n is a weak approximator for f with respect to uniform. Thus `WP` can be used as the basis of a uniform-distribution weak learning algorithm for the class PL_∞ [4] of functions which have at least one Fourier coefficient of magnitude inverse-polynomially large in n. Specifically, if we run `WP` with an appropriate inverse-polynomial threshold then it will be guaranteed to return a non-empty set of coefficients for which all of the corresponding parity functions (or their negations, for those with negative coefficients) are weak approximators to f. Therefore, we may choose any of these parity functions as the weak hypothesis returned by the learning algorithm, and hence the name Weak Parity algorithm.

Actually, because our estimates of the sums of squares of various subsets are in general not perfect, we estimate to within a tolerance of $\theta^2/4$ and recurse on a subset if our estimate is at least $3\theta^2/4$. This guarantees that the set S of coefficients returned by `WP` has the heaviness properties claimed above. Also note that by Parseval's there will be at most $2/\theta^2$ subsets recursed on at each of the n levels of the recursion. Therefore, as long as the estimates of the sums of squares can all be performed efficiently, the algorithm runs in polynomial time.

The key to the `WP` algorithm, then, is showing how to estimate the sums of squares of certain subsets of Fourier coefficients efficiently. The subsets of interest are of the form $\{\hat{f}(A) \mid A \cap [k] = B\}$, where $[k] = \{1, \ldots, k\}$ and $B \subseteq [k]$. Let C_B represent such a subset, and let D_k be the distribution on X^2 that places zero weight on all pairs (x, y) such that the last $n-k$ bits of x and y are not equal and is uniform over all other pairs. Then Goldreich and Levin [9] showed that the sum of squares of coefficients in any such C_B is given by

$$\mathbf{E}[f(x)f(y)\chi_B(x \oplus y)]$$

where the expectation is taken over (x, y) drawn according to D_k. In typical learning-theoretic applications of this algorithm, it is assumed that a membership oracle for f is available and these expectations are estimated using calls to this oracle.

Taking $Q(x, y, f(x), f(y)) = f(x)f(y)\chi_B(x \oplus y)$ shows that WP can be converted to a weak second-order SQ learning algorithm for PL_∞ [21]. We next wish to show that this algorithm can be simulated by a pair-consistent algorithm. To facilitate this, we will limit the class of functions to be learned so that we have a class with polynomial VC dimension. Specifically, we will assume that f is a polynomial-size DNF, *i.e.*, a Boolean function expressible as a DNF with at most polynomial in n terms. Every such function is contained in the class $\widehat{PT}_1$ of functions expressible as the majority vote of polynomially many parity functions [11]. And there are only $2^{poly(n)}$ many such functions [3]. Thus the VC dimension of this class is polynomial in n.

Furthermore, since there are only 2^n parity functions χ_B and n different distributions D_k, the pseudo-dimension of the families of queries is polynomial in n (for fixed tolerance τ). Therefore, given a pair-consistent oracle of a known noise rate η we can simulate the SQ algorithm from a polynomial-size sample using our second-order compensating functional.

Finally, note that if we draw a pair (x, y) according to D_k then the probability of seeing either x or y in a subsequent pair is exponentially small in n for all D_k. This follows because D_k can be sampled by the following process:

- Draw $n - k$ bits c, k bits a, and k bits b uniformly at random
- Construct x by concatenating a and c, y by concatenating b and c.

Since either k or $n - k$ is at least $n/2$, the probability that a subsequent x' or y' will match either x or y is negligible. Thus, for this algorithm's choice of (membership) query distributions D_k, there is negligible probability that a polynomial-size sample sequence drawn from a persistent noisy oracle will differ noticeably from a sample returned by a pair-consistent noisy oracle. Therefore, the above pair-consistent algorithm for weakly learning DNF also tolerates persistent classification noise.

5.3 A noise-tolerant HS

In this section we extend the Harmonic Sieve algorithm—which learns DNF with respect to the uniform distribution using noiseless membership queries—so that it tolerates persistent classification noise in the membership queries. As above, we begin with a brief outline of the original algorithm, and again we ignore the possibility of failure to simplify the explanation. We then extend the algorithm in several steps. First, we rewrite the algorithm so that it uses second-order statistical queries rather than membership queries. Next, we show that the SQ algorithm can be simulated by a pair-consistent algorithm operating on a polynomial-size sample. Finally, we argue that for this algorithm samples drawn from pair-consistent and persistent noisy oracles are, with extremely high probability, indistinguishable, giving the desired result.

5.3.1 The Harmonic Sieve

The Harmonic Sieve consists of two primary components: a boost-by-filtering algorithm of Freund [6,7] and an extension of the `WP` algorithm. We consider the latter algorithm first.

The Harmonic Sieve needs a generalized version of `WP` which finds the heavy Fourier coefficients of certain non-Boolean functions. Specifically, given a function $g : \{0,1\}^n \to \mathbf{R}$ and a threshold θ, we want an algorithm `WP`$'$ that efficiently finds all of the Fourier coefficients $\hat{g}(A)$ of magnitude at least $\theta/\sqrt{2}$. It has been shown [11] that the `WP` algorithm can in fact be applied with almost no modification to perform this task.

Specifically, we can use exactly the same recursive splitting algorithm to isolate the large Fourier coefficients, and the test for each split can be computed by estimating the same expectation (with g replacing f) as before. There are only two differences. First, because the function g may have greater variance than the Boolean function f, we may need to draw a larger sample to estimate the expectation. Second, because our bound on the number of subsets of Fourier coefficients split at each level of the `WP` recursion was based in part on the expected value of the square of the target function, there may be more subsets at each level when a real-valued function g is the target rather than a Boolean f. However, for the g's produced by the Harmonic Sieve while learning a DNF expression of size polynomial in n, the magnitude of g will be inverse polynomial in the accuracy ϵ required of the learning algorithm. This in turn implies a polynomial bound on both the variance of g and on the number of subsets at each level of the `WP` recursion.

The other component of the Harmonic Sieve is a modification of a hypothesis-boosting algorithm due to Freund. We give only the basic ideas here; the reader interested in a detailed discussion is referred to [12]. Given a weak learning algorithm for a function class and the task of producing a strong hypothesis with respect to uniform, the hypothesis booster first uses the weak learner to produce a weak hypothesis with respect to uniform. It then determines a number k of stages that will be performed (how this is determined is discussed below). For each stage $0 < i < k$ it defines a new distribution and invokes the weak learner against the distribution to find a weak hypothesis w_i (w_0 is the initial weak hypothesis learned with respect to uniform). The final strong hypothesis is a majority vote over all of the weak hypotheses produced. The distribution defined at stage i of this process is given by

$$D_i(x) = \frac{\frac{1}{2^n}\alpha_i(k, x, f(x), w_0(x), w_1(x), \ldots, w_{i-1}(x))}{\mathbf{E}_y[\alpha_i(k, y, f(y), w_0(y), \ldots)]}$$

where α_i is an explicit, efficiently computable function of its parameters, $0 \leq \alpha_i \leq 1$ for all possible i and all valid parameters, and the expectation is over uniform y.

The Harmonic Sieve modifies this booster in the following way. At each stage $i > 0$ the algorithm simulates an oracle $D'_i(x)$ that given any x returns an estimate of the weight assigned to x by the distribution D_i. Specifically, the algorithm estimates $\mathbf{E}_y[\alpha_i(k, y, f(y), w_0(y), \ldots)]$ by sampling; call this estimate E_{α_i}. Then D'_i is defined as:

$$D'_i(x) = \frac{\frac{1}{2^n}\alpha_i(k, x, f(x), w_0(x), w_1(x), \ldots, w_{i-1}(x))}{E_{\alpha_i}}.$$

Note that an oracle for D'_i can be simulated given a membership oracle for f. The Harmonic Sieve requires that the estimate E_{α_i} be made with tolerance inverse polynomial in ϵ, the specified accuracy required of the algorithm's hypothesis, and a "cut-off" condition of the booster guarantees that the magnitude of E_{α_i} is at least inverse polynomial in ϵ.

The weak learner boosted by the Harmonic Sieve is, as might be expected, `WP`$'$. At each stage i, `HS` simulates a membership oracle for the function

$$\begin{aligned} g'_i(x) &= 2^n f(x) D'_i(x) \\ &= \frac{f(x)\alpha_i(k, x, f(x), w_0(x), w_1(x), \ldots, w_{i-1}(x))}{E_{\alpha_i}}. \end{aligned}$$

It can be shown that the inverse-polynomially heavy Fourier coefficients of g'_i correspond exactly to the parity functions that are inverse-polynomially well correlated with f with respect to D_i. Furthermore, it can be shown that any Boolean function expressible as a DNF with polynomially many in n terms has a non-empty set of inverse-polynomially well correlated parity functions [11]. Finally, note that g'_i is polynomially bounded in ϵ as required by `WP`$'$ due to the cut-off condition of the booster. Therefore, `WP`$'$ is an appropriate weak learner for the Harmonic Sieve to boost.

The number of boosting stages k required to achieve a final hypothesis h such that $\Pr(f \neq h) \leq \epsilon$ is inverse polynomial in the number of terms in the smallest DNF representation of f and depends logarithmically on ϵ^{-1}. Specifically, letting s represent the number of terms in f, $k = O(s^2 \log \epsilon^{-1})$. Thus, if we assume that the target class is the set of all DNF expressions with at most a fixed polynomial in n number of terms, then k can be computed directly.

Jackson [11,12] extended the basic Harmonic Sieve in a number of ways, showing among other things that DNF can be learned with respect to certain nonuniform distribution classes and that some geometric concepts can be learned by a generalization of the Sieve. He also considered noise tolerance and showed that, with respect to the uniform distribution, the original `WP` algorithm weakly learns DNF despite persistent classification noise. In fact, `WP` will return essentially the same parity function as the weak approximator whether or not noise is present; noise simply causes the algorithm to require more time (the increase is inverse polynomial in the noise rate's difference from $\frac{1}{2}$). This means that the first boosting stage of the Harmonic Sieve will produce the same weak approximator whether or not the target membership oracle is noisy. At first glance, this fact might seem to imply that the distribution generated by `HS` at its second boosting stage would be the same whether or not the target was noisy. Jackson showed that if this *was* the case at the second stage, then the generalized `WP` used as the weak learner by `HS` would again return the same hypothesis whether or not the target was noisy, and in fact this would continue for subsequent stages as well. However, the distribution at the second stage (and succeeding stages) is dependent not only on the weak hypothesis produced but also on the target, and specifically on any *noise* present in the target. Jackson overlooked this fact and mistakenly claimed that his results concerning noise tolerance of the generalized `WP` implied that an unmodified `HS` would also strongly learn DNF despite persistent classification noise.

5.3.2 Generalizing `HS`

While we do not directly address the question of noise tolerance of the original `HS` algorithm, we will present a generalized Harmonic Sieve that, by using a compensating functional, learns DNF with respect to uniform despite persistent classification noise. First, we show how to learn DNF using second-order statistical queries, and then we will build on this to obtain the persistent noise result.

Notice that in the original Harmonic Sieve, membership queries are used at each boosting stage i in two ways: to compute E_{α_i}, and within `WP'` to find the heavy Fourier coefficients of g'_i. E_{α_i} can obviously be computed using statistical queries rather than membership queries (use $Q_{\alpha_i}(x, y, f(x), f(y)) = \alpha_i(k, y, f(y), w_0(y), \ldots)$). Recalling our earlier discussion about `WP'`, it is also clear that if we had a second-order statistical query oracle for g'_i then we could find this function's heavy Fourier coefficients. As with `WP`, the queries we would use would be of the form $Q(x_1, x_2, g'_i(x_1), g'_i(x_2)) = g'_i(x_1)g'_i(x_2)\chi_B(x_1 \oplus x_2)$. But note that given x_1, x_2, $f(x_1)$, and $f(x_2)$, we can efficiently compute $g'_i(x_1)$ and $g'_i(x_2)$. Therefore, we can use a query

$$Q'(x_1, x_2, f(x_1), f(x_2)) = \frac{f(x)\alpha_i(k, x, f(x), w_0(x), \ldots)f(y)\alpha_i(k, y, f(y), w_0(y), \ldots)}{E^2_{\alpha_i}}$$

to a second-order oracle for f to simulate the associated query to the oracle for g'_i. This completes the proof that DNF can be learned with respect to uniform from a second-order statistical query oracle.

There is a somewhat subtle point here that is worth noting. We are making use of the fact that the query given to the statistical query oracle has (conceptually) access to the true target function; it is only the resulting expected value of the query that is corrupted by noise. This means that the computations of $\alpha_i()$ needed to compute $g'_i(x)$ can be thought of as being performed using the noiseless function f. This is critical to the proper computation of the distribution D'_i and thus to the overall operation of the learning algorithm. It is precisely on this point that Jackson's earlier attempt to obtain a noise-tolerant DNF algorithm [11] broke down, as his analysis neglected the fact that the computation of a "true" α_i depends on access to a noiseless f.

We next wish to show that we can still learn DNF given access to a pair-consistent noisy oracle rather than a second-order SQ oracle. Now, of course, we no longer have a "true" function f available to us. Nevertheless, the noise-compensating functional of Theorem 4.1 allows us to accurately simulate queries to the SQ oracle. And this simulation can be performed efficiently given a polynomial bound on the pseudo-dimension of the query class, which we consider next.

When learning DNF expressions bounded by a size s polynomial in n, a crude bound on the total number of queries of this form that the algorithm might use is $2^{2ni}|\Phi|^2 2^k$, where $|\Phi| = 3^{ns}$ is the size of the space of target functions, because the target and i parity functions (weak hypotheses) of earlier stages of the boosting process determine (efficiently) the g'_i. Now i runs up to $O(s^2 \log \epsilon^{-1})$. Thus the logarithm of the number of possible queries at level k of the `WP'` recursion is polynomial in n, s, and $\log \epsilon^{-1}$, and so is the sample size needed for uniform approximation of all of the expectations involved in the SQs, using also the facts that the number of SQ-families is just n, the number of levels of the recursive splitting process in `WP'`, and the range $[-M, M]$ of the functionals, like the bound for g'_i, has inversely polynomial growth in ϵ.

Finally, recall that the `WP'` algorithm boosted by `HS` is essentially the `WP` algorithm discussed in Section 5.2. In particular, WP′ makes random draws according to the same distributions D_k that are used by WP. Thus, the analysis of WP in that section also gives that for WP′ there is negligible probability that pair-consistent and persistent noisy oracles will produce noticeably different samples. Therefore, HS—with WP′ modified to use our second-order noise-compensating functional—learns DNF with respect to uniform from a persistent noisy oracle.

5.4 An alternate analysis

Actually, it turns out that for these particular applications we do not need the full generality of our approach to producing noise-tolerant algorithms. In particular, using the $x \neq y$ (off-diagonal) second-order compensating functional suffices to learn DNF in a noise-tolerant fashion with respect to uniform. Since all that changes is the compensating functional we use, once we have shown that the new functional suffices for pair-consistent noise-tolerant learning, persistent noise tolerance follows immediately from our earlier argument. Thus we focus here on briefly outlining a proof that the $x \neq y$ functional suffices for simulating second-order SQs. The idea is that, while a $x \neq y$ functional will not in general simulate SQs as well as the original functional, the errors introduced do not cause deviation in the algorithm's final output (with high probability).

The reason we included the $x = y$ part in the original second-order functional is because in the independent noise model we can have a pair x, y with $x = y$ and $f(x) \neq f(y)$, but this cannot occur in the pair-consistent model. The purpose, then, is to allow us to obtain an estimate in the pair-consistent model of a result we would obtain in the independent noise model. Now in the WP algorithm we are interested in estimating quantities of the form $\mathbf{E}[f(x)f(y)\chi_B(x \oplus y)]$. First, note that this is always a non-negative value, since it represents a sum of squares. Also observe that if $x = y$ and $f(x) \neq f(y)$ then $f(x)f(y)\chi_B(x \oplus y) = -1$. On the other hand, if $x = y$ and $f(x) = f(y)$ then $f(x)f(y)\chi_B(x \oplus y) = +1$. Thus the primary effect of using only the $x \neq y$ part in the compensating functional in the pair-consistent noise model is that the expectations will be overestimated (more positive) relative to the values obtained in the independent model.

Now recall that in WP, $\Pr(x = y) = 2^{-i}$, where i represents the level of the recursion. Thus, if WP is given an inverse-polynomially in n large threshold, then the only levels at which $\Pr(x = y)$ is non-negligible are the top $O(\log n)$ levels. But at these levels, the only effect of an overestimate of $\mathbf{E}[f(x)f(y)\chi_B(x \oplus y)]$ is to potentially cause WP to recurse on some subsets on which it would not otherwise recurse. However, at lower levels of the recursion, where the effect of the overestimation will be minimal, all descendants of such subsets will be eliminated because it will be discovered that they fall below threshold. Thus the only impact of these overestimates is that the algorithm might do somewhat more work than it would if a better estimate was available. However, the algorithm still runs efficiently since the extraneous subsets only appear in the top $O(\log n)$ levels. A similar argument applies to $\mathtt{WP}'$. Thus the somewhat simpler off-diagonal compensating functional is sufficient for the WP and Harmonic Sieve applications.

References

[1] J. A. Aslam and S. E. Decatur. General bound on statistical query learning and PAC learning with noise via hypothesis boosting, in *Proceedings of the 34th Annual Symposium on Foundations of Computer Science* (1993) 282–291.

[2] 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* (1994) 253–262.

[3] J. Bruck. Harmonic analysis of polynomial threshold functions. *SIAM Journal of Discrete Mathematics* **3(2)** (May 1990) 168–177.

[4] J. Bruck and R. Smolensky. Polynomial threshold functions, AC^0 functions and spectral norms, in *Proceedings of the 31st Annual Symposium on Foundations of Computer Science* (1990) 632–641.

[5] N. H. Bshouty. Exact learning via the monotone theory, in *Proceedings of the 34th Annual Symposium on Foundations of Computer Science* (1993) 302–311.

[6] Y. Freund. Boosting a weak learning algorithm by majority, in *Proceedings of the Third Annual Workshop on Computational Learning Theory* (1990) 202–216.

[7] Y. Freund. *Data Filtering and Distribution Modeling Algorithms for Machine Learning.* PhD thesis, University of California at Santa Cruz, Sept. 1993. Available as technical report UCSC-CRL-93-37.

[8] S. A. Goldman, M. J. Kearns, and R. E. Schapire. Exact identification of read-once formulas using fixed points of amplification functions. *SIAM Journal on Computing* **22(4)** (Aug. 1993) 705–726. Preliminary version appeared in *Proceedings of the 31st Symposium on Foundations of Computer Science* (1990) 193–202.

[9] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions, in *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing* (1989) 25–32.

[10] D. Haussler. Decision theoretic generalizations of the pac model. *Information and Computation* **100** (1992) 78–150.

[11] J. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution, in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science* (1994) 42–53.

[12] J. C. Jackson. *The Harmonic Sieve: A Novel Application of Fourier Analysis to Machine Learning Theory and Practice.* PhD thesis, Carnegie Mellon University, Aug. 1995. Available as technical report CMU-CS-95-183.

[13] M. J. Kearns. Efficient noise-tolerant learning from statistical queries, in *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing* (1993) 392–401.

[14] M. J. Kearns, R. E. Schapire, and L. M. Sellie. Toward efficient agnostic learning, in *Fifth Annual Workshop on Computational Learning Theory* (1992) 341–352.

[15] M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory* (MIT Press, 1994).

[16] M. Kharitonov. Cryptographic lower bounds for learnability of Boolean functions on the uniform distribution, in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory* (1992) 29–36.

[17] M. Kharitonov. Cryptographic hardness of distribution-specific learning, in *Proceedings of the 25th Annual ACM Symposium on Theory of Computing* (1993) 372–381.

[18] E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. *SIAM Journal on Computing* **22(6)** (Dec. 1993) 1331–1348. Earlier version appeared in *Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing* (1991) 455–464.

[19] D. Pollard. *Convergence of Stochastic Processes* (Springer-Verlag, 1984).

[20] R. E. Schapire. The strength of weak learnability. *Machine Learning* **5** (1990) 197–227.

[21] E. Shamir and C. Shwartzman. Learning by extended statistical queries and its relation to pac learning, in *Springer Lectures Notes in AI 904* (Computational Learning Theory, EuroCOLT '95, Barcelona) 357–366.

Correspondence author address:

Jeff Jackson
Math and Computer Science Dept.
Duquesne University
600 Forbes Ave
Pittsburgh, PA 15282-1754
USA

jackson@mathcs.duq.edu