Algorithmica 14: 70–84, 1995.

Can PAC Learning Algorithms Tolerate Random Attribute Noise?

Sally A. Goldman*
Department of Computer Science
Washington University
St. Louis, Missouri 63130

Robert H. Sloan†
Dept. of Electrical Engineering and Computer Science
University of Illinois at Chicago
Chicago, IL 60607

Abstract

This paper studies the robustness of PAC learning algorithms when the instance space is $\{0,1\}^n$, and the examples are corrupted by purely random noise affecting only the attributes (and not the labels). For *uniform attribute noise*, in which each attribute is flipped independently at random with the same probability, we present an algorithm that PAC learns monomials for any (unknown) noise rate less than 1/2. Contrasting this positive result, we show that *product random attribute noise*, where each attribute i is flipped randomly and independently with its own probability p_i, is nearly as harmful as malicious noise—no algorithm can tolerate more than a very small amount of such noise.

*Supported in part by a GE Foundation Junior Faculty Grant and NSF grant CCR-9110108. Part of this research was conducted while the author was at the M.I.T. Laboratory for Computer Science and supported by NSF grant DCR-8607494 and a grant from the Siemens Corporation. Net address: sg@cs.wustl.edu.

†Supported in part by NSF grant CCR-9108753. Part of this research was conducted while the author was at Harvard and supported by ARO grant DAAL 03-86-K-0171. Net address: sloan@eecs.uic.edu.

1 Introduction

In the problem of concept learning from examples, there is a set of objects belonging to known classes. The goal of a learning algorithm or learner is to develop a rule (or hypothesis) that properly classifies these objects after having seen some number of correctly classified objects. For example, if the objects are pieces of office furniture, then the learner's task may be to find a rule that accurately distinguishes chairs from all other types of office furniture. Clearly, a desirable feature of such a learning algorithm is the ability to succeed even when the attributes in the data contain noise.

In this paper we study the problem of learning with random noise in the attributes, under a formal model of learning known as the PAC model (defined formally in section 2). In particular, we study the case where each object is described by n binary attributes, and there are two classes, 0 and 1. (This restricted problem is referred to as boolean function learning.) In order to best study the effect of noise in the attributes, in particular, on the learning process, we assume throughout that the classification of each instance is always correctly reported. Previously, both random classification noise [1] and arbitrary, malicious attribute noise [5] have been conclusively studied within the PAC model. Note that the previous results on random classification noise can be combined with the new results presented here in a relatively straightforward manner.

For dealing with random noise affecting only the classification, the "best-agreement" rule, according to which one draws a large sample of data and then outputs the hypothesis that mislabels the fewest examples from that sample, works quite well (ignoring the issue of computation time). However, Sloan [10] has show that this "best-agreement" rule can only tolerate very small amounts of random attribute noise—suggesting that random attribute noise may be difficult to overcome. On the other hand, by using a different strategy, Shackelford and Volper [9] have obtained an algorithm that tolerates a large amount of random attribute noise (at a *known* noise rate) for learning k-DNF formulas. Thus their result suggests that random attribute noise may be like random classification noise, where large amounts of noise can sometimes be tolerated.

In order to fully understand the difficulty of overcoming random attribute noise, we first carefully examine the way in which the noise is formally modeled. Most previous work in the PAC model has considered *uniform random attribute noise*, in which each attribute is flipped independently at random with the same probability. The algorithm of Shackelford and Volper demonstrates that when the exact noise rate is known, at least for learning k-DNF formulas, a large amount of uniform random attribute noise can be tolerated. In this paper we extend that positive result by showing that even if the noise rate is *unknown*, the class of monomials is still efficiently learnable from data with a large amount of uniform random attribute noise.

Contrasting this positive result, we show that *product random attribute noise*, where each attribute i is flipped randomly and independently with its own probability p_i (all p_i are less than some given upper bound for the noise rate), is nearly as harmful as malicious noise. That is, no algorithm (regardless of sample complexity or computation time) can tolerate more than a very small amount of product random attribute noise. On the whole, these results are surprising. Intuitively, one would think that random labeling noise destroys much more information than random attribute noise, but, in fact, PAC learning is possible with large amounts of random *labeling* noise [1]. We discuss this matter further in Section 7

2 PAC Learning Model

We begin by describing the PAC ("probably approximately correct"), or distribution-free, learning model introduced by Valiant [11]. This model aims to give a precise precise model for the problem of concept learning from examples, including a definition of what it means to "do well" at learning from examples. For convenience' sake, the model is normally restricted to the case where there are only two possible classes.

The set of all possible objects is called the *instance space*, and any subset of the instance space is called a *concept*. The particular subset of the instance space consisting of all instances in class 1 is called the *target concept*. The learner is attempting to infer this unknown target concept c given that c comes from some known class concepts, or *concept class* $\mathcal{C}$. In this paper, we concentrate on Boolean functions. In this context, $\mathcal{C} = \bigcup_{n \geq 1} \mathcal{C}_n$ is parameterized by the number of variables n, and each $c \in \mathcal{C}_n$ is a subset of the instance space $X_n = \{0,1\}^n$, or, equivalently, a zero-one function on $\{0,1\}^n$.

Throughout this paper, we assume that the n variables are $v_1, v_2, \ldots, v_n$ where the value of v_i is given by the ith bit (denoted by b_i) of the instance. A *literal* is a variable v_i or its negation $\bar{v}_i$. Observe that the instances can be viewed as an assignment to the variables since v_i is assigned the value b_i.

Two important concept classes are monomials and k-DNF. A monomial is a conjunction of literals, where each literal is either v_i or $\bar{v}_i$ for $1 \leq i \leq n$. A k-DNF formula is a disjunction of monomials, where each monomial is the conjunction of at most k literals, for k a constant.

Given any concept $c \in \mathcal{C}_n$, we say that x is a *positive instance* of c if $c(x) = 1$, and x is a *negative instance* of c if $c(x) = 0$. For example, the target concept for the class of monomials over five variables might be $v_1\bar{v}_4v_5$. Then the instance "10001" is a positive instance and "00001" is a negative instance.

In the PAC model, the learner is given access to labeled (positive and negative) examples of the target concept, drawn randomly according to some unknown *target distribution* $\mathbf{D}$ over X^n. Specifically, the learner has available to it a black box or oracle called EX such that each call to EX returns a labeled instance, or *example*, $(x, c(x))$ where x is drawn at random from $\mathbf{D}$ and labeled according to c. We call a collection of examples a *sample*. The learner is also given as input $0 < \epsilon, \delta < 1$. The learner's goal is to output with probability at least $1 - \delta$ a *hypothesis* h that has probability at most ϵ of disagreeing with c on a randomly drawn example from $\mathbf{D}$ (thus, the hypothesis has *accuracy* at least $1 - \epsilon$, or is ϵ-*good*). We shall use $\triangle$ to denote the symmetric difference between two concepts. Thus, the above can be rephrased by saying that the learner's goal is to output some representation of a concept $\hat{c}$ such that

$$\Pr\left[\mathbf{D}(c \triangle \hat{c}) > \epsilon\right] \leq \delta,$$

where the probability is over the calls to EX and any coin flips used by the learning algorithm.

If such a learning algorithm exists (that is, an algorithm that meets the goal for any $n \geq 1$, any target concept $c \in \mathcal{C}_n$, any target distribution $\mathbf{D}$, and any ϵ, δ), and that algorithm runs in polynomial time, then we say that C is *PAC learnable*. In this setting polynomial time means polynomial in n, $1/\epsilon$ and $1/\delta$.

More detailed discussions of the PAC model, and of the motivations behind it, can be found in many articles (eg. [2, 3]), including, of course, Valiant's original article [11].

3 Models for Learning with Noise

The ordinary definition of PAC learning (from noiseless data) assumes that EX returns correct data. In this paper we are concerned with the case in which our instances come from some noise oracle, instead of the usual noise-free oracle, EX. Each noise oracle represents some noise process being applied to the examples from EX. The output from the noise process is all the learner can observe. The "desired," noiseless output of each oracle would thus be a correctly labeled example (x, s), where x is drawn according to $\mathbf{D}$. We now describe the actual outputs from the following noise oracles: MAL_ν [12], RMC_ν [1], URA_ν [10], and PRA_ν.

- When MAL_ν is called, with probability $1 - \nu$, it does indeed return a correctly labeled (x, s) where x is drawn according to $\mathbf{D}$. With probability ν it returns an example (x, s) about which no assumptions whatsoever may be made. In particular, this example may be maliciously selected by an adversary who has infinite computing power, and has knowledge of the target concept, $\mathbf{D}$, ν, and the internal state of the algorithm calling this oracle. This *malicious noise* oracle models the situation where the learner usually gets a correct example, but some small fraction ν of the time the learner gets noisy examples and the nature of the noise is unknown or unpredictable. As introduced by Kearns and Li [5], we shall use $E_{MAL}(\mathcal{C}_n)$ (respectively, $E^{poly}_{MAL}(\mathcal{C}_n)$) to denote the largest malicious noise rate that can be tolerated by any learning algorithm (respectively, polynomial-time learning algorithm)for $\mathcal{C}_n$.

- When RMC$_\nu$ is called, with probability $1-\nu$, it returns a correctly labeled (x,s) where x is drawn according to $\mathbf{D}$. With probability ν it returns an example $(x,\bar{s})$. This *random misclassification noise* oracle models the situation where the learner usually gets a correct example, but some small fraction ν of the time the learner receives an example in which the label has been inverted. We shall use $E_{RMC}(\mathcal{C}_n)$ (respectively, $E^{poly}_{RMC}(\mathcal{C}_n)$) to denote the largest malicious noise rate that can be tolerated by any learning algorithm (respectively, polynomial-time learning algorithm) for $\mathcal{C}_n$.

- The oracle URA$_\nu$ makes sense only when the instance space is $\{0,1\}^n$ (i.e., we are learning boolean functions). The oracle URA$_\nu$ calls EX and obtains some $(b_1\cdots b_n, s)$. URA$_\nu$ then adds noise to this example by independently flipping each bit b_i to $\bar{b}_i$ with probability ν for $1 \le i \le n$. Note that the label of the "true" example is never altered by URA$_\nu$. This *uniform random attribute noise* oracle models a situation where the attributes of the examples are subject to noise, but that noise is as benign as possible. For example, the attributes might be sent over a noisy channel. We shall use $E_{URA}(\mathcal{C}_n)$ (respectively, $E^{poly}_{URA}(\mathcal{C}_n)$) to denote the largest uniform random attribute noise rate that can be tolerated by any learning algorithm (respectively, polynomial-time learning algorithm) for $\mathcal{C}_n$.

- The oracle PRA$_\nu$ also only applies when we are learning boolean functions. This oracle calls EX and obtains some $(b_1\cdots b_n, s)$. The oracle PRA$_\nu$ then adds noise by independently flipping each bit b_i to $\bar{b}_i$ with some fixed probability $\nu_i \le \nu$ for each $1 \le i \le n$. This *product random attribute noise* oracle provides a more general model of random attribute noise than URA$_\nu$[1]. We shall use $E_{PRA}(\mathcal{C}_n)$ (respectively, $E^{poly}_{PRA}(\mathcal{C}_n)$) to denote the largest product random attribute noise rate that can be tolerated by any learning algorithm (respectively, polynomial-time learning algorithm) for $\mathcal{C}_n$.

The noise oracles we will focus on here are URA$_\nu$ and PRA$_\nu$.

[1] Technically, PRA$_\nu$ specifies a family of oracles, each member of which is specified by n variables, $\nu_1,\ldots,\nu_n$, where $0 \le \nu_i \le \nu$.

4 Review of PAC Learning From Noisy Data

The method most commonly used for PAC learning in the presence of noise is to pick a concept that has the best (or at least very good) agreement with a sample of data corrupted by noise. It has been shown that for both discrete [1] and continuous [6] instance spaces, the hypothesis that minimizes disagreements meets the PAC criterion when the examples are modified by random labeling noise. Sloan [10] has extended those results to the case of malicious labeling noise. Similarly, Kearns and Li [5] have shown that this method of minimizing disagreements can tolerate a small amount of malicious noise in discrete instance spaces. So these results specify the amount of noise that can be tolerated ignoring the issue of computation time. (Of course, if a hypothesis minimizing disagreements can be found in polynomial time then the above techniques produce efficient learning algorithms.) Finally, as Blumer et al. mention [2], their VC dimension methods can be used to prove that this minimal disagreement method also works for handling small amounts of malicious noise in continuous instance spaces. In the case of uniform random attribute noise, if one uses the minimal disagreement method, then the minimum error rate obtainable (i.e. the minimum "epsilon") is bounded below by the noise rate [10]. We note that for arbitrary adversarial malicious noise, that is the maximum noise rate that *any* algorithm can tolerate [5].

Although the method of minimizing disagreements is not effective against random attribute noise, there are techniques for coping with uniform random attribute noise. In particular, Shackelford and Volper [9] have an algorithm that tolerates large amounts of random attribute noise for learning k-DNF formulas. That algorithm, however, has one very unpleasant requirement: it must be given the exact noise rate (or at least a very good estimate of the noise rate) as an input. Littlestone [7] has looked at how his Winnow algorithm can tolerate several different models of attribute noise. He first considers an adversarial model of attribute noise. Assuming that the uncorrupted instances are linearly separable, under this noise model, the mistake bound for Winnow just increases by an additive term proportional to the weighted sum of the number of errors occurring in each relevant attribute. Next he considers the situation in which there is a large amount of redundant information provided by attributes that are separately indicative of the correct classification of each instance. In this case he shows that a large amount of noise can be handled where the irrelevant attributes are affected by arbitrary adversarial noise and the relevant attributes are affected by random noise independently of one another.

In Table 1 we summarize the results (from previous work and this paper) about PAC learning monomials and k-DNF formulas from the noise oracles discussed in the Section 3.

Noise Model		Monomials	k-DNF Formulas
Malicious Noise:	E_{MAL}^{poly}	$\geq \frac{\epsilon}{n} \ln \frac{n}{\epsilon}$ [5]	$\geq \frac{\epsilon}{n^k} \ln \frac{n^k}{\epsilon}$ [5]
	E_{MAL}	$< \frac{\epsilon}{1+\epsilon}$ [5]	$< \frac{\epsilon}{1+\epsilon}$ [5]
Random Misclassification Noise:	E_{RMC}^{poly}	$\geq \frac{1}{2} - \omega$ [1] for any $\omega > 0$	$\geq \frac{1}{2} - \omega$ [1] for any $\omega > 0$
	E_{RMC}	$< 1/2$ [1]	$< 1/2$ [1]
Uniform Random Attribute Noise: (noise rate known)	E_{URA}^{poly}	1 [9]	1 [9]
Uniform Random Attribute Noise: (noise rate *not* known)	E_{URA}^{poly}	$\geq \frac{1}{2} - \omega$ [§ 5] for any $\omega > 0$	open
	E_{URA}	$< 1/2$	$< 1/2$
Product Random Attribute Noise:	E_{PRA}	$< 2\epsilon$ [§ 6]	$< 2\epsilon$ [§ 6]

Table 1: This table summarizes the results known about learning monomials and k-DNF formulas under various models of noise. Note that the upper bounds correspond to polynomial-time algorithms that can tolerate the given noise rate, and the lower bounds correspond to information-theoretic proofs that no algorithm can tolerate the given noise rate.

5 Learning Monomials From Noisy Data

In this section we present an algorithm for learning monomials from data corrupted with uniform random attribute noise with any noise rate less than 1/2.

The key idea we exploit is the following: Imagine that the literal v_1 is included in the target concept. Then whenever the learner receives a positive instance with the first bit off, it must be that the bit was on in the "noise free" instance, and flipped by the noise oracle. Hence, with high probability, the ratio

$$\frac{\text{number of times bit 1 is off in positive instance}}{\text{total number of positive instances}} \tag{1}$$

provides a good estimate of the noise rate (given a sufficiently large set of positive instances). Notice also that if v_1 is *not* in the formula, then the expectation of the ratio specified in (1) still is bounded below by the noise rate. Thus we will estimate the noise rate to be the minimum, over all literals, of the ratio specified in (1) for the literal v_1.

To compute the actual sample size needed to make these estimates, we use Hoeffding's Inequality [4] (also referred as a form of Chernoff bounds) as stated below:

Lemma 1 (Hoeffding's Inequality) *Let $Y_1, \ldots, Y_m$ be a sequence of m independent Bernoulli trials, each succeeding with probability p. Let $S = Y_1 + \cdots + Y_m$ be the random variable describing the total number of successes. Then for $0 \leq \gamma \leq 1$, the following holds:*

$$\Pr[|S - pm| > \gamma m] \leq 2e^{-2m\gamma^2} \tag{2}$$

Also, for $0 \leq \alpha \leq p$,

$$\Pr[S \leq \alpha m] \leq e^{-2m(\alpha - p)^2} \tag{3}$$

Once we know how to obtain a good estimate for the noise rate, we can apply Angluin and Laird's [1] technique of successive approximation to obtain an upper bound on the noise rate that is sufficiently close to the actual noise rate. Finally, using this estimate of the noise rate, we apply Shackelford and Volper's [9] algorithm when specialized to the case of monomials. However, the correctness proof provided by Shackelford and Volper assumes the *exact* noise rate is provided. While one could extend their proof to apply when given only a sufficiently close estimate of the noise rate, we give instead a direct proof of correctness that is easier to follow and yields a better bound on the time and sample complexity.

Let the literals be numbered from 1 to $2n$, and for each literal i, let

$$\begin{aligned} q_i &= \Pr[\text{Literal } i \text{ is off in a random positive instance from EX (noiseless data)}] \\ p_i &= \Pr[\text{Literal } i \text{ is off in a random positive instance from URA}_\nu]. \end{aligned}$$

Our goal is to output a conjunction that contains every literal that is in the target monomial, and no literal with a high value of q_i. Of course, we cannot directly estimate the q_i's since we only see examples from URA_ν. Our method is to accurately estimate all the p_i's using examples from URA_ν and then use these estimates to determine which literals have a high value for q_i and should thus be excluded. Observe that for all i

$$\begin{aligned} p_i &= q_i(1-\nu) + (1-q_i)\nu \\ &= \nu + q_i(1-2\nu) \end{aligned} \tag{4}$$

Thus for any literal i that is in the target monomial, $p_i = \nu$. Furthermore, since $\nu < 1/2$, for any literal i that is not in the target monomial $p_i \geq \nu$. We will show that by accurately estimating all the p_i's we can obtain a good estimate for the noise rate by simply taking the minimum estimated value over all the p_i's. We then output the conjunction of all literals i having values of p_i close to that minimum.

Theorem 1 *There exists an algorithm to PAC learn the class of monomials over n variables given data from URA_ν that has sample complexity of*

$$O\left(\frac{n^2}{(1-2\nu)^2\epsilon^3}\ln\frac{n}{\delta} + \frac{1}{(1-2\nu)^2}\ln\frac{n}{\delta(1-2\nu)}\right),$$

and time complexity of

$$O\left(\frac{n^3}{(1-2\nu)^2\epsilon^3}\ln\frac{n}{\delta} + \frac{n}{(1-2\nu)^2}\ln\frac{n}{\delta(1-2\nu)}\right).$$

Proof: In Figure 1 we present Algorithm A, which achieves the bounds stated in the theorem. In step 1, the algorithm obtains an upper bound ν_b on the noise rate ν. (Note that this step could fail to halt with probability zero, so strictly speaking this is not a finite procedure.) In step 2 the algorithm computes how many positive examples are needed to obtain a sufficienlty accurate estimate of the noise rate from the ratio in (1) above. Then in step 3 a large enough sample is drawn so that with high probability the desired number of positive examples are obtained. (If enough positive are not obtained then the algorithm simply halts and outputs that it has failed.) Next in steps 4 and 5, the algorithm estimates the noise to be the minimum over all literals of the fraction of the time that a literal is off in the positive examples. Finally, in step 6 the algorithm outputs as its hypothesis the conjunctions of all literals that occur infrequently enough in the positive examples.

It follows from Theorem 3 of Angluin and Laird [1] that with probability at least $1-\delta/2$, step 1 of Algorithm A terminates with $r \leq 1 + \left\lceil \log_2 \frac{1}{1-2\nu} \right\rceil$ and $\nu \leq \nu_b < 1/2$.

Inputs: n, ϵ, δ, access to URA$_\nu$ (ν unknown).
Output: Some monomial (possibly "FALSE").

1. **Repeat for** $r = 1, 2, 3, \ldots$
   (a) Put $\hat{\nu}_b = \frac{1}{2} - (1/2)^{r+1}$
   (Comment: $\hat{\nu}_b$ is current guess at an upper bound on unknown ν.)
   (b) Draw $m_r = 2^{2r+3} \ln \frac{n 2^{r+3}}{\delta}$ examples from URA$_\nu$
   (c) For each literal i
   $\hat{p}_i =$ (number of times literal i is off in sample)$/m_r$
   (d) $\hat{p} = \min_i \hat{p}_i$
   (e) **Until** $\hat{p} < \hat{\nu}_b - 2^{-(r+2)}$
2. Let $m = \dfrac{32n^2}{(1-2\hat{\nu}_b)^2\epsilon^2} \ln \dfrac{16n}{\delta}$
3. Draw $m' = \max\left\{\dfrac{2m}{\epsilon}, \dfrac{2}{\epsilon^2}\ln\dfrac{4}{\delta}\right\}$ examples from URA$_\nu$
   (a) If m positive instances are not obtained, halt and output "FALSE"
   (b) Else let S be a sample of m positive examples (from the sample drawn)
4. **For** each literal i
   $\hat{p}_i =$ (number of times literal i is off in S)$/m$.
5. $\hat{\nu} = \min\limits_i \hat{p}_i$.
6. **Output** conjunction of all literals i such that $\hat{p}_i \leq \hat{\nu} + \epsilon(1-2\nu_b)/4n$

Figure 1: Algorithm A for PAC learning monomials under random attribute noise with an unknown noise rate.

Thus, with probability at least $1-\delta/2$, the total sample complexity needed for step 2 is at most

$$\sum_{r=1}^{1+\lceil \log_2 \frac{1}{1-2\nu}\rceil} 2^{2r+3} \ln\left(\frac{n2^{r+3}}{\delta}\right) = O\left(\left(\frac{1}{1-2\nu}\right)^2 \ln\frac{n}{\delta(1-2\nu)}\right).$$

Now we need to show that the value m' in step 3 is large enough that with high probability either either Algorithm A gets m positive examples or the the hypothesis FALSE is ϵ-good. Let p_+ denote the probability of drawing a positive example from URA_ν. (Note that since the noise process does not affect the labels, p_+ is also the probability of drawing a positive example directly from EX.) We now apply Hoeffding's Inequality to show that if $p_+ \geq \epsilon$ then using a sample of size $m' = \max\left\{\frac{2m}{\epsilon}, \frac{2}{\epsilon^2}\ln\frac{4}{\delta}\right\}$ ensures with probability at least $1-\delta/4$ that the algorithm will obtain at least m positive examples in step 4. Of course, if $p_+ < \epsilon$ then the hypothesis FALSE is ϵ-good.

We can view the drawing of the sample as a sequence of m' independent Bernoulli trials, with a success corresponding to drawing a positive example (which occurs with probability p_+). Let S be the total number of positive examples in the sample. Thus by applying Hoeffding's Inequality (as given in Inequality (3) with $\alpha = p_+/2$, we get

$$\begin{aligned} \Pr[S < m'p_+/2] &\leq e^{-2m'\left(\frac{p_+}{2}-p_+\right)^2} \\ &= e^{\frac{-m'p_+^2}{2}}. \end{aligned}$$

Setting $e^{\frac{-m'p_+^2}{2}} \leq \delta/4$ and solving for m' gives that $m' \geq \frac{2}{p_+^2}\ln\frac{4}{\delta}$. In addition we want $m'p_+/2 \geq m$, so we conclude that a sample of size

$$m' = \max\left\{\frac{2m}{p_+}, \frac{2}{p_+^2}\ln\frac{4}{\delta}\right\}$$

suffices to ensure with probability at least $1-\delta/4$, that a sample of m' randomly drawn examples contains at least m positive examples.

Next we show that if at least m positive examples are obtained in step 3, then the estimates $\hat{p}_i$ of the p_i obtained in step 4 are accurate with high probability. Again, by repeatedly applying Hoeffding's Inequality (as given in Inequality (2) with $\gamma = \epsilon(1-2\nu_b)/8n$), it can be shown that by using a sample of size

$$m = \frac{32n^2}{(1-2\nu_b)^2\epsilon^2}\ln\frac{16n}{\delta} = O\left(\frac{n^2}{(1-2\nu)^2\epsilon^2}\ln\frac{n}{\delta}\right),$$

the probability that *all* of the estimates $\hat{p}_i$ are within $\epsilon(1-2\nu_b)/8n$ of their true value p_i is at least $1-\delta/4$. That is,

$$\Pr\left[\bigwedge_{(1\leq i\leq 2n)} p_i - \frac{\epsilon(1-2\nu_b)}{8n} < \hat{p}_i < p_i + \frac{\epsilon(1-2\nu_b)}{8n}\right] \geq 1-\delta/4. \tag{5}$$

For the remainder of this proof, we assume that $\nu \leq \nu_b$, at least m positive examples are obtained in step 3, and all the $\hat{p}_i$'s are within the tolerance specified in Inequality 5 (these conditions are all satisfied with probability at least $1 - \delta$). Then:

1. The estimate $\hat{\nu}$ of ν is accurate. Namely,

$$\nu - \frac{\epsilon(1 - 2\nu_b)}{8n} < \hat{\nu} < \nu + \frac{\epsilon(1 - 2\nu_b)}{8n}. \tag{6}$$

2. Any literal that is in the target monomial will be placed in Algorithm A's hypothesis.

3. For any literal i that is *not* in the target monomial but is placed in Algorithm A's hypothesis $q_i < \epsilon/2n$.

Since for all i, $p_i \geq \nu$ and for any literal in the target monomial $p_i = \nu$, item 1 above easily follows from the fact that for all i, $\hat{p}_i$ is within $\epsilon(1 - 2\nu_b)/8n$ of p_i.

For for any literal i that is in the target monomial, $p_i = \nu$, and thus by Inequalities 6 and 5

$$\hat{p}_i < \hat{\nu} + \frac{\epsilon(1 - 2\nu_b)}{4n}.$$

Hence for every literal i that is in the target monomial, $\hat{p}_i$ will satisfy the inequality in step 6 of the algorithm and thus be placed in the hypothesis.

Finally, to prove that item 3 above holds, we show that if $q_i \geq \epsilon/2n$ then $\hat{p}_i$ will not satisfy the inequality in step 6. Applying equation (4) we get

$$p_i \geq \nu + \frac{\epsilon(1 - 2\nu_b)}{2n}.$$

Finally since $\hat{p}_i$ and $\hat{\nu}$ are within the given tolerance of their true values, it follows that:

$$\hat{p}_i > \hat{\nu} + \frac{\epsilon(1 - 2\nu_b)}{4n}.$$

Thus the choice of literals made by the algorithm in step 6 ensures that every literal i in the output formula has $q_i < \epsilon/2n$.

To complete the proof, observe that literals in the output monomial are a superset of the literals in the target monomial. Therefore, the algorithm's hypothesis is false whenever the target concept is false. Since every literal i in the output has $q_i < \epsilon/2n$ and there are at most $2n$ literals in the output formula, the probability that the output is false when the target concept is true is at most ϵ.

□

Remark: We can obtain a very similar result for a noise model where with probability $1 - \nu$ the example is noise free, and with probability ν a single one of the n bits is picked at random and flipped. The key observation now is that if literal v_1 is in the target monomial, then fraction of the time that bit 1 is off in positive examples is now an estimate of the noise rate divided by n, and, when v_1 is not in the target monomial, that fraction is still (in the limit) a lower bound on $\nu\ n$.

6 Product Random Attribute Noise

In this section we show that product random attribute noise makes PAC learning almost as difficult as malicious noise.

Kearns and Li [5] defined a concept class $\mathcal{C}$ to be *distinct* if there exist concepts $c_2, c_2 \in \mathcal{C}$ and instance $u, v, w, x \in X$ satisfying that $u \in c_1, u \notin c_2, v \in c_1, v \in c_2, w \notin c_1, w \in c_2, x \notin c_1, x \notin c_2$. (I.e., neither c_1 nor c_2 is a subset of the other; they have nonempty intersection, and their union is not the whole instance space. This is a formal definition of a concept class being nontrivial.) Then they show that for any distinct concept class, it is impossible to PAC learn to accuracy ϵ with examples from MAL_ν unless

$$\nu < \epsilon/(1+\epsilon).$$

Thus, our result for product random attribute noise is similar, with a slightly weaker bound.

Theorem 2 *Let $\mathcal{C}$ be any distinct concept class over $\{0,1\}^n$. It is possible to PAC learn $\mathcal{C}$ to accuracy ϵ with examples from PRA_ν only if $\nu < 2\epsilon$.*

Proof: We use the method of induced distributions [5].

Let c_1, c_2 be the concepts that exist in $\mathcal{C}$ according to the definition of a distinct concept class. For ease of exposition we shall let $c_1 = v_1$ (the first attribute) and $c_2 = v_2$ (the second attribute). We shall let $u = \{100\cdots0\}$, $v = \{110\cdots0\}$, $w = \{010\cdots0\}$, $x = \{000\cdots0\}$. We shall place zero probability weight in $\mathbf{D}$ on all other instances. Thus without loss of generality we may assume that these are the only 4 instances and that $\nu_k = 0$ for $3 \le k \le n$.

Fix some value of ν in the range $0 \le \nu < 1/2$. Consider a distribution $\mathbf{D}$ which assigns weight $(1-\nu)/2$ to 00 and 11 and weight $\nu/2$ to 01 and 10. In Table 1 we show the two noise-free probability distributions on examples obtained by labeling the instances drawn from $\mathbf{D}$ according to concept c_1 or concept c_2.

Now consider what happens under the following two learning problems:

1. For the first learning problem let c_1 be the target concept, let $\mathbf{D}$ be the distribution on instances (so $\mathbf{D}_1$ is the noise-free distribution on examples), and let the noise oracle be PRA_ν with $\nu_1 = \nu$ and $\nu_2 = 0$. The observed distribution on examples is shown in the last column of Table 1.

Labeled Instance	$\mathbf{D}_1$	$\mathbf{D}_2$	Observed dist.
$(x, -)$	$(1-\nu)/2$	$(1-\nu)/2$	$(1-\nu)^2/2$
$(x, +)$	0	0	$\nu^2/2$
$(w, -)$	$\nu/2$	0	$\nu(1-\nu)/2$
$(w, +)$	0	$\nu/2$	$\nu(1-\nu)/2$
$(u, -)$	0	$\nu/2$	$\nu(1-\nu)/2$
$(u, +)$	$\nu/2$	0	$\nu(1-\nu)/2$
$(v, -)$	0	0	$\nu^2/2$
$(v, +)$	$(1-\nu)/2$	$(1-\nu)/2$	$(1-\nu)^2/2$

Table 2: Two induced noise-free probability distributions obtained from distribution $\mathbf{D}$ for concept class $\mathcal{C}$, and the distribution obtained after noise for the two learning problems used in the proof of Theorem 2.

2. For the second learning problem, let c_2 be the target concept, let $\mathbf{D}$ be the distribution (so $\mathbf{D}_2$ is the noise-free distribution on examples), and let the noise oracle be PRA$_\nu$ with $\nu_1 = 0$ and $\nu_2 = \nu$. The observed distribution on examples is also the one shown in the last column of Table 1.

These two learning problems have an identical probability distribution on the observed (noisy) samples. Therefore, no PAC learning algorithm has any basis for distinguishing between these two scenarios. Thus with probability at least $1-\delta$ the learning algorithm must output a concept c such that:

$$\mathbf{D}(c_1 \bigtriangleup c) < \epsilon \text{ and } \mathbf{D}(c_2 \bigtriangleup c) < \epsilon.$$

By the triangle inequality it follows that:

$$\mathbf{D}(c_1 \bigtriangleup c) + \mathbf{D}(c_2 \bigtriangleup c) \geq \mathbf{D}(c_1 \bigtriangleup c_2).$$

Thus with probability at least $1-\delta$ the learning algorithm must output a concept c such that $\mathbf{D}(c_1 \bigtriangleup c_2) < 2\epsilon$. Finally, since $\mathbf{D}(c_1 \bigtriangleup c_2) = \nu$, no learning algorithm can succeed unless $\nu < 2\epsilon$. $\square$

Remark: Notice that the lower bound of Theorem 2 is an information-theoretic representation-independent hardness result. No algorithm, regardless of either its sample size or computation time, can escape this bound. Furthermore, we have made no assumptions on the representation class from which the hypothesis may be selected—this bound on the tolerable noise rate holds for *any* hypothesis the learner may output.

7 Final Thoughts

We have studied the robustness of PAC learning algorithms under several forms of random attribute noise. We presented a new algorithm for learning monomials that tolerates large amounts of uniform random attribute noise (any noise rate less than 1/2) without any prior knowledge of the noise rate. An intriguing open question is whether one can PAC learn k-DNF formulas under uniform random attribute noise for an *unknown* noise rate.

However, we feel that our negative result for the more realistic (or at least more general) product random attribute noise oracle makes it clear that, in general, under the PAC learning model random attribute noise is quite harmful. In fact, product random attribute noise is significantly more harmful than *malicious* labeling noise generated by a powerful adversary [10], and nearly as harmful as truly malicious noise. This result was surprising to the authors. One expects to only be able to tolerate a small amount of truly malicious noise—it is obviously the worst sort of noise possible. Yet, one would expect that labeling noise would be worse than random attribute noise.

7.1 An empirical test

Indeed, one empirical test of the effects of random noise on concept learning from examples by the ID-3 system concluded that in practice, classification noise is much more harmful than attribute noise [8]. Unfortunately, that experiment was not directly comparable to our work. One can view learning from examples as consisting of two phases: a training phase (in our case, getting examples from the oracle EX), and a performance phase where the learner's output is evaluated for accuracy. We, and others who have studied noise in the PAC learning setting, have studied the case where the training data is corrupted by noise, but the data used for evaluating the learning in the performance phase is noise-free. Note that this is the only sensible paradigm for studying *malicious* noise; for random noise, this is one of several reasonable possibilities.

Quinlan's empirical study took a different approach: He assumed that the same random noise was present in both the training and performance phases of the learning [8]. That approach virtually guarantees that classification noise will seem to be the most harmful, since each percentage point increase in classification noise for the data for the performance phase should lead to roughly a full percentage point decrease in the learner's performance.

Thus the difference between our results and Quinlan's empirical study may simply be caused by studying different problems. Another possibility is that the difference stems from the PAC model's focus on worst cases. It may be that for a "typical" learning-from-examples problem, classification noise is more harmful that attribute noise, but that for the "hardest" learning-from-examples problems, attribute noise is more harmful.

7.2 Future research directions

More research is needed on both the theoretical and the applied sides. There are several problems concerning random noise that we in the PAC community should examine. One, obviously, is what results we get studying the situation where the data has the same random noise in both the training and performance phases, and whether this case makes more or less sense that the case we studied in this paper.

Another problem is finding good "meta-algorithms" for PAC learning for the cases where we can make substantial progress. We would like to find a way to convert any correct PAC learning algorithm into one that is robust against noise. Kearns and Li have such an algorithm for very small amounts of malicious noise [5]. This problem is open, however, for both random classification noise and uniform attribute noise.

Acknowledgments

We would like to thank Les Valiant for suggesting this line of research. We thank Dana Angluin for pointing out an error (and providing a correction to this error) in our original statement of Theorem 2. We also thank several anonymous reviewers for their many useful comments. In particular, we thank them for pointing out that we could combine our techniques with those from Angluin and Laird to obtain a good estimate for the noise rate.

References

[1] Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

[2] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, 1989.

[3] David Haussler, Michael Kearns, Nick Littlestone, and Manfred K. Warmuth. Equivalence of models for polynomial learnability. *Information and Computation*, 95(2):129–161, December 1991.

[4] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963.

[5] Michael Kearns and Ming Li. Learning in the presence of malicious errors. *SIAM Journal on Computing*, 22(4):807–837, August 1993.

[6] Philip D. Laird. *Learning from Good and Bad Data*. Kluwer international series in engineering and computer science. Kluwer Academic Publishers, Boston, 1988.

[7] Nicholas Littlestone. Redundant noisy attributes, attribute errors, and linearythreshold learning using winnow. In *Fourth Workshop on Computational Learning Theory*, pages 147–156, 1991.

[8] J. Ross Quinlan. The effect of noise on concept learning. In *Machine Learning, An Artificial Intelligence Approach (Volume II)*, chapter 6, pages 149–166. Morgan Kaufmann, 1986.

[9] George Shackelford and Dennis Volper. Learning k-DNF with noise in the attributes. In *First Workshop on Computatinal Learning Theory*, pages 97–103, Cambridge, Mass. August 1988. Morgan Kaufmann.

[10] Robert H. Sloan. Types of noise in data for concept learning. In *First Workshop on Computational Learning Theory*, pages 91–96. Morgan Kaufmann, 1988.

[11] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.

[12] Leslie G. Valiant. Learning disjunctions of conjunctions. In *Proceedings IJCAI-85*, pages 560–566. International Joint Committee for Artificial Intelligence, Morgan Kaufmann, August 1985.