Machine Learning, 11, 37-62 (1993)

Noise-Tolerant Occam Algorithms and Their Applications to Learning Decision Trees

YASUBUMI SAKAKIBARA YASU@IIAS.FLAB.FUJITSU.CO.JP
International Institute for Advanced Study of Social Information Science (IIAS-SIS), Fujitsu Laboratories Ltd., 140, Miyamoto, Numazu, Shizuoka 410-03, Japan

Editor: David Haussler

Abstract. In the distribution-independent model of concept learning of Valiant, Angluin and Laird have introduced a formal model of noise process, called *classification noise process*, to study how to compensate for randomly introduced errors, or noise, in classifying the example data. In this article, we investigate the problem of designing efficient learning algorithms in the presence of classification noise. First, we develop a technique of building efficient robust learning algorithms, called *noise-tolerant Occam algorithms*, and show that using them, one can construct a polynomial-time algorithm for learning a class of Boolean functions in the presence of classification noise. Next, as an instance of such problems of learning in the presence of classification noise, we focus on the learning problem of Boolean functions represented by decision trees. We present a noise-tolerant Occam algorithm for k-DL (the class of decision lists with conjunctive clauses of size at most k at each decision introduced by Rivest) and hence conclude that k-DL is polynomially learnable in the presence of classification noise. Further, we extend the noise-tolerant Occam algorithm for k-DL to one for r-DT (the class of decision trees of rank at most r introduced by Ehrenfeucht and Haussler) and conclude that r-DT is polynomially learnable in the presence of classification noise.

Keywords. learning from examples, probably approximately correct learning, noisy examples, polynomial-time learnability, decision lists, decision trees

1. Introduction

The distribution-independent model introduced by Valiant (1984), which is called *probably approximately correct learning* (PAC learning, for short), has contributed to the great progress on the theoretical front for learning from examples. Several interesting classes of Boolean functions have been proved to be or not to be polynomially learnable in the PAC learning model (e.g., Blumer, Ehrenfeucht, Haussler, & Warmuth, 1989; Kearns, 1990; Natarajan, 1989; Pitt & Valiant, 1988; see also Anthony & Biggs, 1992).

However, most of the works depend strongly on the assumption of perfect, noiseless examples. This assumption is generally unrealistic, and in many situations of the real world, our observations will often be afflicted by noise. Thus it is practical to assume that the examples given to the learning algorithm contain some noise. Few works have suggested any way to make their learning algorithms noise tolerant, and two formal models of noise have been studied so far in the PAC learning model for concept learning.

One is the *malicious error model* initiated by Valiant (1985) and investigated by Kearns and Li (1988): independently for each example, the example is replaced, with some small probability, by an arbitrary example classified perhaps incorrectly. The goal of this model is to capture the worst possible case of noise process by an adversary. This model is also called *adversarial noise process* by Angluin and Laird (1988).

The other is the *classification noise process* introduced by Angluin and Laird (1988): independently for each example, the label of the example is reversed with some small probability. The goal of this model is to study the question of how to compensate for randomly introduced errors, or noise, in classifying the example data.

In this article, we consider the classification noise process to study the effect on polynomial-time learnability. We investigate the problem of *designing efficient learning algorithms* in the presence of classification noise.

We first develop a technique for building efficient robust learning algorithms in the presence of classification noise. That is the technique of finding a concept consistent, not with the given whole sample, which is the well-known technique used in learning in the absence of noise, but rather with *a certain large fraction* of the sample. We call a polynomial-time algorithm to find such a concept *a noise-tolerant Occam algorithm*, and show that by using a noise-tolerant Occam algorithm for a class of concepts, one can construct a polynomial-time algorithm for learning the class in the presence of classification noise.

Next, as an instance of such problems of learning in the presence of classification noise, we focus on the learning problem of Boolean functions represented by decision trees. Decision trees are particularly interesting because they are commonly used in AI learning algorithms, for which noise tolerance is essential. Decision trees are often used for classification tasks and as the representation of acquired knowledge in a learning system. A classification task is to assign an element of the domain to one of a specified number of disjoint classes. For example, the diagnosis of a medical condition from symptoms is a classification task, in which the classes could be either the various disease states or the possible therapies. There has been much research concerning the problem of learning decision trees (e.g., Breiman, Friedman, Olshen, & Stone, 1984; Clark & Niblett, 1989; Pagallo, 1990; Utgoff, 1989). One famous and practical example of such systems is ID3 by Quinlan (1986a). ID3 induces decision trees from examples. In the decision-tree learning problem, concepts are defined on a set of objects in which the objects are described in terms of a set of attribute-value pairs. In the case where each attribute is a Boolean variable (i.e., the value is 0 or 1), the problem of learning decision trees can be formulated as the problem of learning Boolean functions.

The problem of learning decision trees in the PAC learning model has also been studied in the absence of noise. Rivest (1987) has introduced a class of representations, called *decision lists*, for representing Boolean functions, and has shown that k-DL (the class of decision lists with conjunctive clauses of size at most k at each decision) is polynomially learnable in the PAC learning model. The decision list is a useful way of representing Boolean functions, and in fact the decision lists are an important class because k-DL properly includes other well-known techniques for representing Boolean functions such as k-CNF (formulae in conjunctive normal form with at most k literals per clause) and k-DNF (formulae in disjunctive normal form with at most k literals per term). Ehrenfeucht and Haussler (1989) have introduced the notion of the *rank* of a decison tree and have shown that for any fixed r, the class of decision trees of rank at most r, denoted r-DT, is polynomially learnable in the PAC learning model and that Rivest's result for decision lists can be interpreted as a special case of their result for rank 1. Recently, Blum (1991) has shown that every decision tree of rank r can be represented as a decision list in r-DL, that is, r-DT is a subclass of r-DL.

We begin with the decision lists for the problem of learning decision trees in the presence of classification noise. We present a noise-tolerant Occam algorithm for k-DL and hence conclude that k-DL is polynomially learnable in the presence of classification noise. This strictly increases the class of Boolean functions that are known to be polynomially learnable in the presence of classification noise: the only example of a class of Boolean functions is k-CNF that has been shown to be polynomially learnable in the presence of classification noise (Angluin & Laird, 1988) and k-DL properly includes k-CNF. Further, we extend the noise-tolerant Occam algorithm for k-DL to one for r-DT and conclude that r-DT is polynomially learnable in the presence of classification noise. Consequently, we present a new method for constructing a decision tree from noisy examples. Both results can hold at a noise rate even close to ½.

2. Probably approximately correct learning from noisy examples

Valiant (1984) has introduced the distribution-independent model of concept learning from random examples. Angluin and Laird (1988) have extended this model by introducing a noise process, called *classification noise process*, to study how to compensate for randomly introduced errors, or noise, in classifying the example data. In this section, we give a brief outline of Valiant's learnability model and the notion of polynomial learnability (Blumer, Ehrenfeucht, Haussler, & Warmuth, 1989) for Boolean functions, and define the classification noise process and the notion of polynomial learnability in the presence of classification noise for Boolean functions. Then we develop a technique of building efficient robust learning algorithms, called *noise-tolerant Occam algorithms*, and show that by using a noise-tolerant Occam algorithm for a class of Boolean functions, one can construct a polynomial-time algorithm for learning the class in the presence of classification noise.

2.1. Probably approximately correct learning for Boolean functions

We assume that there are n Boolean attributes (or variables) to be considered, and we denote the set of such variables as $V_n = \{x_1, x_2, \ldots, x_n\}$. An *assignment* $\vec{a}$ is a mapping from V_n to the set $\{0, 1\}$. Let X_n denote the set of all such assignments mapping from V_n to $\{0, 1\}$. We may also think X_n denotes the set $\{0, 1\}^n$ of all binary strings of length n. Then a *Boolean function* is defined to be a mapping from X_n to $\{0, 1\}$.

Boolean formulae are often used as useful representations for Boolean functions. The simplest Boolean formula is just a single variable. Each variable x_i ($1 \leq i \leq n$) is associated with two *literals*: x_i itself and its negation $\bar{x}_i$. A *term* is a conjunction of literals, and a *clause* is a disjunction of literals. The *size* of a term or clause is the number of its literals. Let **true** be the unique term of size 0, which always returns the value 1, and **false** be the unique clause of size 0, which always returns the value 0. Let C_k^n denote the set of all terms (Conjunctions) of size at most k over V_n, and let D_k^n denote the set of all clauses (Disjunctions) of size at most k over V_n. Thus

$$|C_k^n| = |D_k^n| \leq (2n + 1)^k = O(n^k).$$

For any fixed k, C_k^n and D_k^n have sizes polynomial in n.

A Boolean formula is in *conjunctive normal form* if it is a conjunction of clauses. We define k-CNF to be the class of Boolean formulae in conjunctive normal form with at most k literals per clause. Similarly, a Boolean formula is in *disjunctive normal form* if it is a disjunction of terms. We define k-DNF to be the class of Boolean formulae in disjunctive normal form with at most k literals per term.

A Boolean formula can be interpreted as a mapping from assignments X_n into $\{0, 1\}$. Thus each Boolean formula defines a corresponding Boolean function from X_n to $\{0, 1\}$ in a natural manner. We do not distinguish between Boolean formulae and the Boolean functions they represent.

Now we describe Valiant's learnability model for Boolean functions. First, fix a class F_n of Boolean functions over V_n and a target Boolean function f_U in F_n to be learned.

An *example* of f_U is a pair $\langle \vec{a}, l \rangle$ where $\vec{a}$ is an assignment in X_n and $l = f_U(\vec{a})$. Thus an example can be viewed as an assignment with the value of the target function f_U at the assignment. The value l is called the *label* of the example. An example $\langle \vec{a}, l \rangle$ is called a *positive example* of f_U if $l = 1$ and called a *negative example* of f_U if $l = 0$. A *sample* is a finite sequence of positive and negative examples of the target Boolean function f_U. The *size* of a sample S is the number of examples in it. A Boolean function g is said to *agree with* an example $\langle \vec{a}, l \rangle$ if $g(\vec{a}) = l$. A Boolean function is *consistent* with the given sample if it agrees with all examples in the sample.

We assume that there is an unknown and arbitrary probability distribution D on X_n. The probability of assignment $\vec{a} \in X_n$ with respect to D is denoted $\Pr_D(\vec{a})$. Random samples are assumed to be drawn independently from the domain X_n according to this probability distribution D on X_n. There is a *sampling oracle* $EX(\,)$ for the target Boolean function f_U, which has no input. Whenever $EX(\,)$ is called, it draws an assignment $\vec{a} \in X_n$ according to the distribution D, and returns $\langle \vec{a}, f_U(\vec{a}) \rangle$. We define a *learning algorithm* for the class F_n of Boolean function as an algorithm that has access to $EX(\,)$ and produces as output a Boolean function in F_n.

A learning algorithm makes a number of calls to $EX(\,)$ and then conjectures some Boolean function $g \in F_n$. The success of learning is measured by two parameters, the accuracy parameter ϵ and the confidence parameter δ, which are given as inputs to the learning algorithm. We define a notion of the difference between two Boolean functions f and g with respect to the probability distribution D as

$$d(f, g) = \sum_{f(\vec{a}) \neq g(\vec{a})} Pr_D(\vec{a}).$$

The *error* of a Boolean function g with respect to the target Boolean function f_U is $d(g, f_U)$. A successful learning algorithm is one that with high probability finds a Boolean function whose error is small. A Boolean function g is called an *ϵ-approximation* of f_U if $d(g, f_U) \leq \epsilon$ and called *ϵ-bad* otherwise.

The notion of *polynomial learnability* in Valiant's learnability model is formally defined as follows.

A class F_n of Boolean functions over V_n is *polynomially (probably approximately correctly (PAC for short)) learnable* if there exists a learning algorithm A for F_n such that for any ϵ and δ, for any target Boolean function $f_U \in F_n$, and for any distribution D on X_n, when A is given as input parameters n, ϵ, and δ and run with the sampling oracle $EX(\,)$ for f_U, the algorithm outputs a Boolean function $g \in F_n$ such that $d(g, f_U) \leq \epsilon$ with probability at least $1 - \delta$, and runs in time polynomial in n, $1/\epsilon$, and $1/\delta$.

The difficulty of learning a Boolean function that has been selected from F_n will depend on the size $|F_n|$ of F_n. We say a class F_n of Boolean functions *has polynomial representation size* if $\ln(|F_n|) = O(n^t)$ for some constant t, that is, if $\ln(|F_n|)$ is a polynomial in n. The logarithm of $|F_n|$, $\ln(|F_n|)$, may be viewed as the number of bits needed to write down an arbitrary element of F_n, using an optimal encoding.

2.2. *Classification noise process*

The classification noise process introduced for concept learning can be interpreted to be applicable to learning Boolean functions as follows.

The sampling oracle is able to draw assignments $\vec{a}$ from X_n according to the relevant distribution D without error, but the process of reporting the value of the target Boolean function f_U at the assignment $\vec{a}$—that is, $f_U(\vec{a})$—is subject to independent random mistakes with some unknown probability η; independently for each example $\langle \vec{a}, l \rangle$, $\langle \vec{a}, 0 \rangle$ is returned with $f_U(\vec{a}) = 1$ and $\langle \vec{a}, 1 \rangle$ is returned when $f_U(\vec{a}) = 0$ with probability η.

It is assumed that the rate of noise η is less than 1/2. To indicate that the sampling oracle is subject to errors of this type, we will denote it by $EX_\eta(\,)$.

Angluin and Laird (1988) have discussed the following argument: in the presence of classification noise, we should assume that there is some information about the noise rate η available to the learning algorithm, namely, an upper bound η_b such that $\eta \leq \eta_b < 1/2$, and just as the running time for polynomial-time learning is permitted in the absence of noise to be polynomial in $1/\epsilon$ and $1/\delta$, we should permit the polynomial to have $1/(1 - 2\eta_b)$ as one of its arguments.

Now we give the precise definition of *polynomial learnability in the presence of classification noise*:

A class F_n of Boolean functions over V_n is *polynomially learnable in the presence of classification noise* if there exists a learning algorithm for F_n such that for any ϵ, δ, and η ($< 1/2$), for any target Boolean function $f_U \in F_n$, and for any distribution D on X_n, when A is given as input parameters n, ϵ, δ, and η_b ($\eta \leq \eta_b < 1/2$), and run with the sampling oracle $EX_\eta(\,)$ for f_U, the algorithm outputs a Boolean function $g \in F_n$ such that $d(g, f_U) \leq \epsilon$ with probability at least $1 - \delta$, and runs in time polynomial in n, $1/\epsilon$, $1/\delta$, and $1/(1 - 2\eta_b)$.

Angluin and Laird (1988) have shown that it is not necessary to have η_b as input. That is, they have described a procedure that outputs a value η_b using calls to $EX_\eta(\,)$ such that with probability at least $1 - \delta$, η_b is between η and 1/2.

2.3. Previous research results

In the absence of noise, when given a sample of a target Boolean function f_U, the fundamental strategy that a learning algorithm takes is producing a Boolean function consistent with the sample. When the sample contains noise, this fundamental strategy may fail because there is no guarantee that such consistent Boolean functions will be found.

Angluin and Laird (1988) have proposed the simple strategy of finding a Boolean function that minimizes the number of disagreements with the given sample, and have shown that in the presence of classification noise, a learning algorithm for a class F_n of Boolean functions that outputs a Boolean function minimizing the number of disagreements PAC-learns F_n.

Let $S = \{\langle \vec{a}_1, l_1 \rangle, \langle \vec{a}_2, l_2 \rangle, \ldots, \langle \vec{a}_m, l_m \rangle\}$ be a sample drawn from an $EX_\eta(\,)$ oracle. For a Boolean function f, let $F(f, S)$ denote the number of indices j for which f disagrees with $\langle \vec{a}_j, l_j \rangle$.

Theorem 1 (Angluin & Laird, 1988) *If we draw a sample S of*

$$m \geq \frac{2}{\epsilon^2(1 - 2\eta_b)^2} \ln \left(\frac{2|F_n|}{\delta} \right)$$

examples from $EX_\eta(\,)$ for the target Boolean function f_U and find any Boolean function $g \in F_n$ that minimizes $F(g, S)$, then with probability at least $1 - \delta$, g is an ϵ-approximation of f_U.

Angluin and Laird (1988) have shown that k-CNF is polynomially learnable in the presence of classification noise for any $\eta < 1/2$.

Kearns and Li (1988) have given hardness results for learning with *malicious errors*. Let us define a class F_n of Boolean functions to be *distinct* if there are Boolean functions $f, g \in F_n$ and assignments $\vec{a}, \vec{b} \in X_n$ satisfying $f(\vec{a}) = f(\vec{b}) = 1$, $g(\vec{a}) = 0$, and $g(\vec{b}) = 1$.

Theorem 2 (Kearns & Li, 1988) *Let F_n be a distinct class of Boolean functions and ϵ the accuracy parameter. Then the largest rate of malicious error that can be tolerated by any learning algorithm for F_n is less than $\epsilon/(1 + \epsilon)$.*

Laird (1988) has shown several interesting results on handling other types of noise processes and on estimating the noise rate. Sloan (1988) has introduced and studied two new noise models between the malicious error model and the classification noise process. Recently Goldman, Kearns, and Schapire (1990) have presented learning algorithms for read-once formulas that are robust for a large amount of slightly more general classification noise under restricted distributions.

2.4. Noise-tolerant Occam algorithms

Now we develop a technique of building efficient robust learning algorithms, called *noise-tolerant Occam algorithm*, that is a generalization of Occam's Razor by Blumer, Ehrenfeucht,

Haussler, and Warmuth (1987). We show that by using a noise-tolerant Occam algorithm for a class of Boolean functions, one can construct a polynomial-time algorithm for learning the class in the presence of classification noise. We will use this fact in the following sections to show the polynomial learnability of decision lists and decision trees in the presence of classification noise.

Angluin and Laird (1988) have proposed the strategy of finding a Boolean function that minimizes the number of disagreements with the given sample. In general, however, it is a hard problem to find a Boolean function that minimizes the number of disagreements with the sample. We weaken this criterion. The following noise-tolerant Occam algorithm takes the strategy of finding a Boolean function consistent with a certain large fraction of the given sample with high probability, rather than of finding a Boolean function that minimizes the number of disagreements.

A *noise-tolerant Occam algorithm, OCCAM*$(S, \epsilon, \delta, \eta_b)$, for a class F_n of Boolean functions over V_n is an algorithm that when given as input a sufficiently large sample S of m examples drawn from $EX_\eta(\,)$ for any target Boolean function f_U, and parameters ϵ, δ, η_b,

1. produces a Boolean function $g \in F_n$ such that

$$\frac{F(g, S)}{m} \leq \eta_b + \frac{\epsilon(1 - 2\eta_b)}{4},$$

with probability at least $1 - \delta$, and
2. runs in time polynomial in n and m.

In the above definition, the probability that the algorithm $OCCAM(S, \epsilon, \delta, \eta_b)$ fails depends on the statistical fluctuation of occurrences of classification noise when the sample is drawn, as shown in the following lemma.

First we show the following important lemma for $OCCAM(S, \epsilon, \delta, \eta_b)$: if $\eta \leq \eta_b \leq \eta + \epsilon(1 - 2\eta)/2$ and F_n has polynomial representation size, then when given a sufficiently large sample S drawn from $EX_\eta(\,)$ for f_U, $OCCAM(S, \epsilon, \delta, \eta_b)$ outputs an ϵ-approximation of f_U with probability at least $1 - \delta$.

The inequality quoted below will be required in the proof to follow. Let p and r be numbers between 0 and 1, and let m be a positive integer. Let $GE(p, m, r)$ denote the probability of getting at least rm successes in m independent Bernoulli trials with probability p, and let $LE(p, m, r)$ denote the probability of at most rm successes in m independent Bernoulli trials with probability p. The following lemma bounds these quantities.

Lemma 3 (Hoeffding's Inequality (Hoeffding, 1963; Angluin & Laird, 1988)) *If* $0 \leq p \leq 1$, $0 \leq s \leq 1$, *and* m *is any positive integer, then*

$$LE(p, m, p - s) \leq e^{-2s^2m}$$

and

$$GE(p, m, p + s) \leq e^{-2s^2m}.$$

Lemma 4 *Suppose that* $\eta \leq \eta_b \leq \eta + \epsilon(1 - 2\eta)/2$. *Suppose also that* F_n *has polynomial representation size. When given a sample* S *consisting of* m *examples drawn from* $EX_\eta(\,)$ *for* f_U, $OCCAM(S, \epsilon, \delta, \eta_b)$ *outputs an* ϵ*-approximation of* f_U *with probability at least* $1 - \delta$. *The sample size* m *required is*

$$m \geq \frac{8}{\epsilon^2(1 - 2\eta_b)^2} \ln \left(\frac{2|F_n|}{\delta} \right).$$

Proof. First we consider the effect of classification noise on searching any Boolean function in F_n. We analyze the expected rate of disagreement between any Boolean function g and example sequences produced by the sampling oracle $EX_\eta(\,)$ for the target Boolean function f_U. Let $d_g = d(g, f_U)$. The probability that an example produced by $EX_\eta(\,)$ disagrees with g is

1. the probability that an example is drawn from $\{\vec{a} \in X_n \mid f_U(\vec{a}) \neq g(\vec{a})\}$ and reported correctly (which is just $d_g(1 - \eta)$)
2. plus the probability that an example is drawn from $\{\vec{a} \in X_n \mid f_U(\vec{a}) = g(\vec{a})\}$ and reported incorrectly (which is just $(1 - d_g)\eta$).

Let p_g denote the probability that an example from $EX_\eta(\,)$ disagrees with g. Then we have

$$\begin{aligned} p_g &= d_g(1 - \eta) + (1 - d_g)\eta \\ &= \eta + d_g(1 - 2\eta). \end{aligned}$$

For the target Boolean function f_U we have $p_{f_U} = \eta$, and for any ϵ-bad Boolean function g we have

$$p_g \geq \eta + \epsilon(1 - 2\eta).$$

Thus any ϵ-bad Boolean function has an expected rate of disagreement that is greater than that of the target Boolean function by at least $\epsilon(1 - 2\eta)$.

We now show that with probability at least $1 - \delta$, $OCCAM(S, \epsilon, \delta, \eta_b)$ outputs an ϵ-approximation of f_U. Let $s = \epsilon(1 - 2\eta_b)$. The probability that the target Boolean function f_U has more than $(\eta_b + s/4)m$ disagreements with a sample S of m examples drawn from $EX_\eta(\,)$ is

$$\begin{aligned} GE(\eta, m, \eta_b + s/4) &\leq GE(\eta_b, m, \eta_b + s/4) \\ &\leq e^{-2(s/4)^2 m} \end{aligned}$$

by Hoeffding's inequality lemma, and the lower bound on m implies that this is less than $\delta/2$. Hence with probability at least $1 - \delta/2$, $OCCAM(S, \epsilon, \delta, \eta_b)$ can find a Boolean function $g \in F_n$ such that $F(g, S)/m \leq \eta_b + s/4$ to output. The probability that a Boolean function with error greater than ϵ has at most $(\eta_b + s/4)m$ disagreements is at most

$$LE(\eta + \epsilon(1 - 2\eta), m, \eta_b + s/4) \leq LE(\eta_b + s/2, m, \eta_b + s/4),$$
by the assumption $\eta_b \leq \eta + \epsilon(1 - 2\eta)/2$

$$\leq e^{-2(s/4)^2 m},$$
by Hoeffding's Inequality lemma.

Since there are at most $|F_n|$ Boolean functions in F_n, the probability of producing a Boolean function with error greater than ϵ is less than

$$|F_n| \cdot e^{-2(s/4)^2 m},$$

and by the lower bound on m, it is less than $\delta/2$. Hence with probability at least $1 - \delta$, $OCCAM(S, \epsilon, \delta, \eta_b)$ outputs an ϵ-approximation of f_U. □

The above lemma indicates that $OCCAM(S, \epsilon, \delta, \eta_b)$ requires an accurate estimate η_b of the actual noise rate η for polynomial learnability. In the following, however, we show that for any upper bound $\eta_b < 1/2$, by iterating $OCCAM(S, \epsilon, \delta, \eta_b)$ for successively smaller values of η_b (down to almost 0) and picking the best Boolean function among the outputs of $OCCAM(S, \epsilon, \delta, \eta_b)$, the existence of $OCCAM(S, \epsilon, \delta, \eta_b)$ for F_n implies the polynomial learnability of F_n in the presence of classification noise.

Theorem 5 *Suppose that $\eta < 1/2$. Suppose also that F_n has polynomial representation size. If there exists a noise-tolerant Occam algorithm for F_n, then F_n is polynomially learnable in the presence of classification noise.*

Proof. We will construct by using $OCCAM(S, \epsilon, \delta, \eta_b)$ a learning algorithm for F_n that, with probability at least $1 - \delta$, outputs an ϵ-approximation of f_U from a sampling oracle $EX_\eta(\,)$ with $\eta < 1/2$. The learning algorithm, *POLY-LEARN*, is given in figure 1.

We will show that with probability at least $1 - \delta$, *POLY-LEARN* outputs an ϵ-approximation of f_U. Among the successively smaller values η_e from η_b down to almost 0, there will exist an η_e such that $\eta \leq \eta_e \leq + \epsilon(1 - 2\eta)/2$ because η_e is decreasing by $\epsilon(1 - 2\eta_b)/2$ and $\epsilon(1 - 2\eta_b) \leq \epsilon(1 - 2\eta)$. Then by the proof of lemma 4, the lower bound on m implies that with probability at least $1 - \delta/2$, there will be at least one Boolean function g_j ($1 \leq j \leq k$) in the queue Q of *POLY-LEARN* such that

$$F(g_j, S)/m \leq \eta_e + \epsilon(1 - 2\eta_e)/4$$
where $\eta \leq \eta_e \leq \eta + \epsilon(1 - 2\eta)/2$

$$\leq \eta + 3\epsilon(1 - 2\eta)/4.$$

Hence with probability at least $1 - \delta/2$, the Boolean function g_i ($1 \leq i \leq k$) in Q that minimizes $F(g_i, S)$ must have $F(g_i, S)/m \leq \eta + 3\epsilon(1 - 2\nu)/4$.

The probability that a Boolean function $g \in F_n$ with error greater than ϵ has $F(g, S)/m \leq \eta + 3\epsilon(1 - 2\eta)/4$ is at most

$$LE(\eta + \epsilon(1 - 2\eta), m, \eta + 3\epsilon(1 - 2\eta)/4) \leq e^{-2(\epsilon(1-2\eta)/4)^2 m},$$
by Hoeffding's Inequality lemma.

ALGORITHM *POLY-LEARN*
Input:

- $OCCAM(S, \epsilon, \delta, \eta_b)$ for a class F_n of Boolean functions,
- A sampling oracle $EX_\eta()$ subject to classification noise,
- A number n and positive fractions ϵ, δ, and η_b, with $0 \leq \eta \leq \eta_b < 1/2$.

Output:

A Boolean function $g \in F_n$.

Procedure:

1. Request a sample S of m examples, where
$$m \geq \frac{8}{\epsilon^2(1-2\eta_b)^2} \ln\left(\frac{2|F_n|}{\delta}\right);$$
2. $\eta_e \leftarrow \eta_b$;
3. $Q \leftarrow emptyqueue$;
4. Repeat:
   4.1. Call $OCCAM(S, \epsilon, \delta, \eta_e)$;
   4.2. Add the output to Q;
   4.3. $\eta_e \leftarrow \eta_e - \frac{\epsilon(1-2\eta_b)}{2}$;
   until $\eta_e \leq 0$;
5. Let $Q = \langle g_1, \ldots, g_k \rangle$;
6. Output a Boolean function g_i $(1 \leq i \leq k)$ in Q that minimizes $F(g_i, S)$.

Figure 1. Polynomial-time learning with classification noise.

Since there are exactly $|F_n|$ Boolean functions in F_n, the probability of producing a Boolean function with error greater than ϵ is less than

$$|F_n| \cdot e^{-2(\epsilon(1-2\eta)/4^2 m},$$

and by the lower bound on m, it is less than $\delta/2$. Hence with probability at least $1 - \delta$, *POLY-LEARN* outputs an ϵ-approximation of f_U.

Further, there are at most $\lceil 1/\epsilon(1 - 2\eta_b) \rceil$ repetitions of calling $OCCAM(S, \epsilon, \delta, \eta_e)$, and hence k is at most $\lceil 1/\epsilon(1 - 2\eta_b) \rceil$ in *POLY-LEARN*. These are executed in time polynomial in $1/\epsilon$, $1/(1 - 2\eta_b)$, and the running time of $OCCAM(S, \epsilon, \delta, \eta_e)$, which is bounded by a polynomial in n, $1/\epsilon$, $1/\delta$, and $1/(1 - 2\eta_b)$. Searching a Boolean function g_i in Q that minimizes $F(g_i, S)$ is executed in time polynomial m, $1/\epsilon$, and $1/(1 - 2\eta_b)$,

since there are at most $\lceil 1/\epsilon(1 - 2\eta_b)\rceil$ Boolean functions in Q. Therefore *POLY-LEARN* runs in time polynomial in n, $1,/\epsilon$, $1/\delta$, and $1/(1 - 2\eta_b)$. □

In contrast to theorem 2 for the malicious error model, the noise rate is independent of the desired accuracy ϵ, and a noise rate close to 1/2 is achievable in theorem 5.

3. An efficient robust algorithm for learning decision lists

Since it is common in the machine learning literature to consider concepts defined on a set of objects in which the objects are described in terms of a set of Boolean attribute-value pairs, the problem of learning Boolean functions from examples has been widely studied both theoretically and empirically. Recently Rivest (1987) has introduced a useful way, called *decision lists*, to represent Boolean functions and to perform classification tasks, and has shown that k-DL (the class of decision lists with conjunctive clauses of size at most k at each decision) is polynomially learnable in Valiant's learnability model. However, Rivest's learning algorithm for k-DL is not robust for noisy data, and he has left open to study the problem of whether the Boolean functions in k-DL can be learned efficiently when the classifications of the given examples may be erroneous with some small probability.

In this section, we give the affirmative answer for this open problem. We present a noise-tolerant Occam algorithm for k-DL and hence conclude that k-DL is polynomially learnable in the presence of classification noise. This strictly increases the class of Boolean functions that are known to be polynomially learnable in the presence of classification noise: The only example of a class of Boolean functions that has been shown to be polynomially learnable in the presence of classification noise is k-CNF (Angluin & Laird, 1988), and k-DL properly includes k-CNF.

3.1. Decision lists

A *decision list* is a list L of pairs

$$\langle (t_1, v_1), (t_2, v_2), \ldots, (t_r, v_r)\rangle$$

where each t_i is a term over V_n, each v_i is a value in $\{0, 1\}$, and the last term t_r is the unique term **true**. A decision list L defines a Boolean function as follows: For any assignment $\vec{a} \in X_n$, $L(\vec{a})$ is defined to be equal to the value v_i where i is the *least* index such that $t_i(\vec{a}) = 1$. (Such an item always exists, since the last term is always **true**.) Let k-DL denote the class of all Boolean functions defined by decision lists, where each term in the list is of size at most k (i.e., each t_i is in C_k^n). As k increases, the class k-DL becomes increasingly expressive. Note that k-DL is closed under complementation (negation).

We may think of a decision list as an extended "**if — then — elseif — ... else —**" rule. Or we may think of decision lists as a linearly ordered set of *production rules*. For example, the decision list

$$L = \langle (x_1x_2, 1), (\bar{x}_2x_3x_5, 0), (x_3x_4, 1), (\textbf{true}, 0)\rangle$$

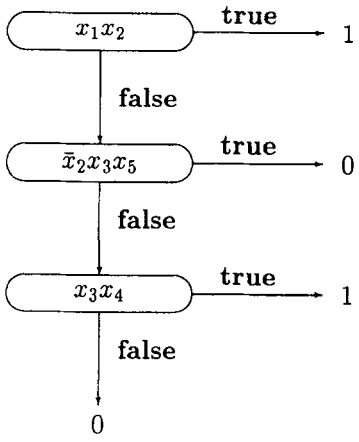

Figure 2. Diagram of the decision list $\langle (x_1x_2, 1), (\bar{x}_2x_3x_5, 0), (x_3x_4, 1), (\text{true}, 0)\rangle$.

may be diagrammed as in figure 2. For example, $L(1, 0\ 1, 1, 1) = 0$; this value is specified by the second pair in the decision list.

When we wish to emphasize the number of variables upon which a class of Boolean functions depends, we will indicate this in parentheses after the class name, as in k-CNF(n), k-DNF(n), or k-DL(n).

Theorem 6 (Rivest, 1987) *For $0 < k < n$, k-CNF(n) and k-DNF(n) are proper subclasses of k-DL(n). For $0 < k < n$ and $n > 2$, (k-CNF(n) $\cup$ k-DNF(n)) is a proper subclass of k-DL(n).*

3.2. *Efficient robust learning of k-DL*

Now we consider an efficient robust algorithm for learning decision lists in the presence of classification noise. We show that there exists a noise-tolerant Occam algorithm for k-DL, and hence k-DL is polynomially learnable in the presence of classification noise.

Let L_U denote the target decision list in k-DL(n) over V_n to be learned. Let (t, v) be a pair in $C_k^n \times \{0, 1\}$. We say a pair (t, v) *disagrees with* an example $\langle \vec{a}, l\rangle$ if $t(\vec{a}) = 1$ and $v \neq l$. We say a pair (t, v) is *correct w.r.t.* a sample S drawn from $EX_\eta(\,)$ for L_U if for every example $\langle \vec{a}, l\rangle$ in S such that $t(\vec{a}) = 1$, $v = L_U(\vec{a})$.

We begin with the trivial fact observed by Rivest (1987) in the absence of noise that *if a decision list is consistent with a sample S, then it is consistent with any subset of S.* This can be restated in the presence of classification noise as follows: *If a sample S is drawn from $EX_\eta(\,)$ for a decision list L_U, then for any nonempty subset S' of S, there exists a correct pair (t, v) w.r.t. S' such that $t(\vec{a}) = 1$ for at least one example $\langle \vec{a}, l\rangle$ in S'.*

Let $S = \{\langle \vec{a}_1, l_1\rangle, \langle \vec{a}_2, l_2\rangle, \ldots, \langle \vec{a}_m, l_m\rangle\}$ be a sample drawn from an $EX_\eta(\,)$ oracle. For a decision list L, let $F(L, S)$ denote the number of indices j for which L disagrees with $\langle \vec{a}_j, l_j\rangle$. For a pair $(t, v) \in C_k^n \times \{0, 1\}$, let $F((t, v), S)$ denote the number of indices

j for which (t, v) disagrees with $\langle \vec{a}_j, l_j \rangle$, and let $T((t, v), S)$ denote the number of indices j for which $t(\vec{a}_j) = 1$.

Now we present an efficient robust learning algorithm for k-DL. Our noise-tolerant Occam algorithm *NODL* for k-DL is shown in figure 3. Here we try to give the intuitive explanation of the algorithm *NODL*. We say a pair (t, v) *explains* an example $\langle \vec{a}, l \rangle$ if $t(\vec{a}) = 1$. Given the sample S, *NODL* proceeds by identifying the pairs of the decision list in order. *NODL* selects as the first pair of the decision list any pair (t,v) of $C_k^n \times \{0, 1\}$ if it disagrees with a small fraction of the examples that are explained by it in S or if the number of examples explained by it is below a given threshold. *NODL* then proceeds to delete from S any example explained by the chosen pair (t, v), and to construct the remainder of the decision list in the same way using the remaining part of S.

Lemma 7 *Suppose that NODL is given a sample S of m examples drawn from $EX_\eta(\,)$ and parameters n, k, ϵ, and η_b. If NODL outputs a decision list L (not "none"), then*

$$\frac{F(L, S)}{m} \leq \eta_b + \frac{\epsilon(1 - 2\eta_b)}{4}.$$

Proof. Let $L = \langle (t_1, v_1), \ldots, (t_r, v_r) \rangle$. For the i-th item (t_i, v_i) of L, let $F_0((t_i, v_i), S)$ denote the number of examples $\langle \vec{a}, l \rangle$ in S such that (t_i, v_i) disagrees with $\langle \vec{a}, l \rangle$ and i is the least index such that $t_i(\vec{a}) = 1$, and let $T_0((t_i, v_i), S)$ denote the number of examples $\langle \vec{a}, l \rangle$ in S for which i is the least index such that $t_i(\vec{a}) = 1$. Any decision list L output by *NODL* has the property that for any i $(1 \leq i \leq r)$,

$$\frac{F_0((t_i, v_i), S)}{T_0((t_i, v_i), S)} \leq \eta_b + \frac{\epsilon(1 - 2\eta_b)}{8}$$

or

$$\frac{T_0((t_i, v_i), S)}{m} \leq Q_l.$$

Therefore the total number $F(L, S)$ of disagreements of L with S is at most the sum of $\Sigma_{i=1}^r F_0((t_i, v_i), S)$ and $Q_l mr/2$, which are bounded by the quantities estimated in the following inequalities:

$$\sum_{i=1}^{r} F_0((t_i, v_i), S) \leq \sum_{i=1}^{r} (\eta_b + \epsilon(1 - 2\eta_b)/8) \cdot T_0((t_i, v_i), S)$$

$$= (\eta_b + \epsilon(1 - 2\eta_b)/8) \cdot \sum_{i=1}^{r} T_0((t_i, v_i), S)$$

$$= (\eta_b + \epsilon(1 - 2\eta_b)/8)m$$

ALGORITHM *NODL*
Input:

- A sample S of m examples drawn from $EX_\eta()$ subject to classification noise,
- Numbers n, k and positive fractions ϵ, and η_b, with $0 \leq \eta \leq \eta_b < 1/2$.

Output:

A decision list L in k-DL(n) such that $F(L,S)/m \leq \eta_b + \epsilon(1-2\eta_b)/4$, or "none".

Procedure:

1. Calculate the following:

$$\begin{aligned} C_k^n &= \{t \mid t \text{ is a term over } V_n \text{ with at most } k \text{ literals}\}, \\ M &= |C_k^n|, \\ Q_I &= \frac{\epsilon(1-2\eta_b)}{4M}; \end{aligned}$$

2. $SS \leftarrow S$; $CC \leftarrow C_k^n \times \{0,1\}$; $L \leftarrow$ *emptylist*; $i \leftarrow 1$;
3. Repeat:
   - 3.1. If SS is empty, then output L and halt;
   - 3.2. Find a pair (t,v) in CC such that
$$\frac{F((t,v),SS)}{T((t,v),SS)} \leq \eta_b + \frac{\epsilon(1-2\eta_b)}{8};$$
   - 3.3. If no such pair can be found, then find a pair (t,v) in CC such that
$$\frac{T((t,v),SS)}{m} \leq Q_I$$
   and $v = 1$ if $F((t,1),SS) \leq F((t,0),SS)$ and $v = 0$ otherwise;
   - 3.4. If no such desired pair can be found, then stop and return "none";
   - 3.5. Let T denote those examples in SS which make t true;
   - 3.6. Add the pair (t,v) to L as the i-th item of the decision list L;
   - 3.7. $SS \leftarrow SS - T$; $CC \leftarrow CC - (t,v)$; $i \leftarrow i+1$;

   until it halts.

Figure 3. Efficient robust learning of k-DL.

and

$$\frac{1}{2} \cdot Q_l m \cdot r \leq \frac{1}{2} \frac{\epsilon(1 - 2\eta_b)}{4M} mM$$

$$= (\epsilon(1 - 2\eta_b)/8)m$$

since $r \leq M$. Hence $F(L, S)/m \leq \eta_b + \epsilon(1 - 2\eta_b)/4$. □

Lemma 8 *Suppose that $\eta \leq \eta_b < 1/2$. Suppose also that S consists of m examples drawn from $EX_\eta(\,)$ for the target decison list L_U in k-DL(n) over V_n and*

$$m \geq \frac{128M}{\epsilon^3(1 - 2\eta_b)^3} \ln\left(\frac{2^{M+1}M}{\delta}\right).$$

Then with probability at least $1 - \delta/2$, NODL outputs a decision list L.

Proof. First we prove that with probability at least $1 - \delta/2$, a sample S is drawn such that for all choices $t_1, t_2, \ldots, t_j$ of terms in C_k^n and for all $t \in C_k^n$, whenever $T((t, v), R) \geq Q_l m$, the number of occurrences of classification noise in $\{\langle \vec{a}, l\rangle \in R \mid t(\vec{a}) = 1\}$ is at most $(\eta_b + \epsilon(1 - 2\eta_b)/8) \cdot T((t, v), R)$, where $R = S - \{\langle \vec{a}, l\rangle \in S \mid t_1(\vec{a}) = 1\} - \ldots - \{\langle \vec{a}, l\rangle \in S \mid t_j(\vec{a}) = 1\}$. Let $t_1, t_2, \ldots, t_j$ in C_k^n and $t \in C_k^n$ be fixed. By Hoeffding's inequality lemma, whenever $T((t, v), R) \geq Q_l m$, the probability that classification noise occurs more than $(\eta_b + \epsilon(1 - 2\eta_b)/8) \cdot T((t\ v), R)$ times is at most

$$GE(\eta, T((t, v), R), \eta_b + \epsilon(1 - 2\eta_b)/8) \leq GE(\eta_b, T((t, v), R), \eta_b + \epsilon(1 - 2\eta_b)/8)$$

$$\leq e^{-2(\epsilon(1-2\eta_b)/8)^2 T((t,v),R)}$$

$$\leq e^{-2(\epsilon(1-2\eta_b)/8)^2 Q_l m},$$

and the lower bound on m implies that this is less than $\delta/(2^{M+1} M)$. Since there are at most 2^M choices of sequences of terms in C_k^n and M terms in C_k^n, the probability that for all choices $t_1, t_2, \ldots, t_j$ of sequences of terms in C_k^n and for all $t \in C_k^n$, the number of occurrences of classification noise in $\{\langle \vec{a}, l\rangle \in R \mid t(\vec{a}) = 1\}$ is at most $(\eta_b + \epsilon(1 - 2\eta_b)/8) \cdot T((t\ v), R)$ is at least $1 - \delta/2$. This completes the proof.

Next we assume that for all choices $t_1, t_2, \ldots, t_j$ of sequences of terms in C_k^n and for all $t \in C_k^n$, the number of occurrences of classification noise in $\{\langle \vec{a}, l\rangle \in R \mid t(\vec{a}) = 1\}$ is at most $(\eta_b + \epsilon(1 - 2\eta_b)/8) \cdot T((t,v), R)$ whenever $T((t, v), R) \geq Q_l m$. We show that this assumption implies that *NODL* outputs a decision list L. For any stage, say i, in the repetition in *NODL* where *SS* is not empty, there is at least one correct pair (t, v) w.r.t. *SS* in *CC*. If $T((t, v), SS)/m \leq Q_l$, then *NODL* can select the pair (t, v) as the i-th item of L. If $T((t, v), SS)/m \geq Q_l$, by the above assumption, the correct pair (t, v) has at most $(\eta_b + \epsilon(1 - 2\eta_b)/8) \cdot T((t, v), SS)$ disagreements with *SS*. Then *NODL* can select the pair (t, v) as the i-th item of L. Hence eventually *NODL* halts and outputs a decision list L.

Therefore with probability at least $1 - \delta/2$, *NODL* outputs a decision list L. □

Theorem 9 *NODL is a noise-tolerant Occam algorithm for k-DL(n).*

Proof. Since $|C_k^n| \leq (2n + 1)^k$, it is clear that *NODL* runs in time polynomial in n and m. Then it is straightforward from lemmas 7 and 8. □

We quote the following important lemma by Rivest (1987).

Lemma 10 (Rivest, 1987) *k-DL(n) has polynomial representation size.*

Proof. $|k\text{-}DL(n)| = O(3^{|C_k^n|}(|C_k^n|)!)$. This implies that $\ln(|k\text{-DL}(n)|) = O(n^t)$ for some constant t. □

Now we have the main theorem.

Theorem 11 *k-DL(n) is polynomially learnable in the presence of classification noise.*

Proof. It is straightforward from theorems 9 and 5 and lemma 10. □

Schapire (1991) had independently achieved this result by using probabilistic decision lists, which are a probabilistic analog of (deterministic) decision lists. Schapire has shown that a special class of probabilistic decision lists with conjunctive clauses of size at most k at each decision can be learned efficiently and the result can be applied to learn ordinary decision lists when the supplied examples are noisy.

4. An efficient robust algorithm for learning decision trees

Recently Ehrenfeucht and Haussler (1989) have introduced the notion of the *rank* of a decision tree; they have shown that the class of all decision trees of rank at most r on V_n is polynomially learnable in Valiant's learnability model and that Rivest's result for decision lists can be interpreted as a special case of their result for rank 1. However, their learning algorithm is not robust for noisy data. Few empirical works (e.g., Quinlan, 1986b) for decision trees have suggested any way to make their learning algorithms noise tolerant, and there has been no theoretical treatment for learning decision trees from noisy examples so far.

In this section, we extend the noise-tolerant Occam algorithm for decision lists to one for decision trees and conclude that the class of all decision trees of rank at most r on V_n is polynomially learnable in the presence of classification noise. This result can hold at a noise rate even close to 1/2.

Consequently, we present a new method for constructing a decision tree from noisy examples. Here we try to give the intuitive explanation of our method. In the course of constructing a decision tree from a given sample in a top-down manner, the learning algorithm first chooses some variable that best divides examples of the sample into their attached labels' classes and then partitions the examples according to the values of that variable. This process of growing the decision tree is recursively applied to each partitioned subset of the sample with the terminating procedure. In the absence of noise, the learning algorithm of finding a consistent decision tree terminates the process when all examples in the current subset have the same label. In contrast to it, in the presence of classification noise, our new learning algorithm terminates the process when

- a certain large fraction of the examples in the current subset already have the same label, or
- the number of examples in the subset is below a given threshold.

Let $S = \{\langle \vec{a}_1, l_1 \rangle, \langle \vec{a}_2, l_2 \rangle, \ldots, \langle \vec{a}_m, l_m \rangle\}$ be a given sample. For a decision tree T, let $F(T, S)$ denote the number of indices j for which T disagrees with $\langle \vec{a}_j, l_j \rangle$. Let x be a variable in V_n. Assume $x = x_i$, where $1 \leq i \leq n$. Let S_0^x denote the set of all examples $\langle \vec{a}, l \rangle$ in a sample S such that $\vec{a} = (a_1, \ldots, a_n)$ and $a_i = 0$, and S_1^x denote the set of all examples $\langle \vec{a}, l \rangle$ in S such that $\vec{a} = (a_1, \ldots, a_n)$ and $a_i = 1$. We say a variable x is *informative* (on S) if both S_0^x and S_1^x are nonempty.

Two algorithms, one in the absence of noise and one in the presence of noise, are illustrated in figure 4. The algorithm *RFT* is extended to have two extra parameters and modified stopping conditions.

The correctness of our new method is theoretically proved in the framework of PAC learning as shown in the following. The adequate input values for two parameters Q_F and Q_I in *RFT* can be calculated from ϵ, η_b, $|S|$, and the rank r of the target decision tree.

4.1. Decision trees

First we give formal definitions of decision trees and their rank, as introduced by Ehrenfeucht and Haussler (1989).

A *decision tree* is a binary tree where each internal node is labeled with a variable and each leaf is labeled with 0 or 1. A decision tree is a useful way to represent a Boolean function. The class $\mathcal{T}_n$ of *decision trees* over V_n is defined recursively as follows:

1. If T is the binary tree consisting of only a root node labeled either 0 or 1, then $T \in \mathcal{T}_n$. (We will abbreviate this case by simply saying "$T = 0$" or "$T = 1$".)
2. If T_0, $T_1 \in \mathcal{T}_n$ and $x \in V_n$, then the binary tree with root labeled x, left subtree T_0, and right subtree T_1 is in $\mathcal{T}_n$. (We will refer to the left subtree as the 0-*subtree* and the right subtree as the 1-*subtree*.)

A decision tree $T \in \mathcal{T}_n$ defines a Boolean function f_T as follows:

1. If $T = 0$ then f_T is the constant function 0, and if $T = 1$ then f_T is the constant function 1.
2. Else if x_i is the variable labeled at the root node, T_0 the 0-subtree, and T_1 the 1-subtree, then for any assignment $\vec{a} = (a_1, \ldots, a_n)$, if $a_i = 0$ then $f_T(\vec{a}) = f_{T_0}(\vec{a})$, else $f_T(\vec{a}) = f_{T_1}(\vec{a})$.

For example, the decision tree in figure 5 represents the Boolean function $x_1x_2 \vee x_3$.

A decision tree is *reduced* if each variable appears at most once in any path from the root to a leaf.

The *rank* of a decision tree T, denoted $r(T)$, is defined as follows:

1. If $T = 0$ or $T = 1$, then $r(T) = 0$.
2. Else if r_0 is the rank of the 0-subtree of T and r_1 is the rank of the 1-subtree, then

ALGORITHM $FT(S)$

Input: A sample S.

Output: A decision tree T.

Procedure:

1. If all pairs $\langle \vec{a}, l \rangle$ in S have $l = 1$, stop and return the decision tree $T = 1$;
   If all pairs $\langle \vec{a}, l \rangle$ in S have $l = 0$, stop and return the decision tree $T = 0$;
2. For some informative variable $x \in V_n$
   (a) Let $T_0^x = FT(S_0^x)$ and $T_1^x = FT(S_1^x)$;
   (b) Stop and return the decision tree with root labeled x, left subtree T_0^x and right subtree T_1^x;

1. Algorithm for constructing a consistent decision tree.

ALGORITHM $RFT(S, Q_F, Q_I)$

Input: A sample S, a positive integer Q_I, and a positive fraction Q_F.

Output: A decision tree T.

Procedure:

1. If $F(1,S)/|S| \leq Q_F$, stop and return the decision tree $T = 1$;
   If $F(0,S)/|S| \leq Q_F$, stop and return the decision tree $T = 0$;
2. If $|S| \leq Q_I$ and $F(1,S) \leq F(0,S)$, stop and return the decision tree $T = 1$;
   If $|S| \leq Q_I$ and $F(0,S) \leq F(1,S)$, stop and return the decision tree $T = 0$;
3. For some informative variable $x \in V_n$
   (a) Let $T_0^x = RFT(S_0^x, Q_F, Q_I)$ and $T_1^x = RFT(S_1^x, Q_F, Q_I)$;
   (b) Stop and return the decision tree with root labeled x, left subtree T_0^x and right subtree T_1^x;

2. Algorithm for constructing a decision tree consistent with a large fraction of S.

Figure 4. Two algorithms for constructing a decision tree.

$$r(T) = \begin{cases} \max(r_0, r_1) & \text{if } r_0 \neq r_1 \\ r_0 + 1 (= r_1 + 1) & \text{otherwise.} \end{cases}$$

Let r-DT(n) denote the set of all Boolean functions over V_n represented by decision trees of rank at most r.

It is easily verified that every function in r-DT(n) can be represented by a reduced decision tree of rank at most r.

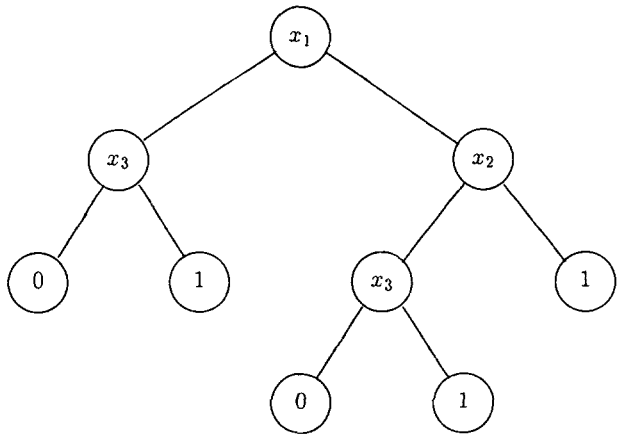

Figure 5. A decision tree representation for $x_1x_2 \vee x_3$.

We quote the following important lemma by Ehrenfeucht and Haussler (1989).

Lemma 12 (Ehrenfeucht & Haussler, 1989)
1. Let $L(n, r)$ denote the maximum number of leaves of any reduced decision trees over V_n of rank r. Then

$$L(0, r) = 1 \text{ for all } r \geq 0,$$

$$L(n, 0) = 1 \text{ for all } n \geq 0,$$

$$L(n, r) = L(n - 1, r) + L(n - 1, r - 1) \text{ for all } n, r \geq 1.$$

Further, $L(n, r)$ is bounded by $(en/r)^r$ for all $n \geq r \geq 1$.
2. If $r = 0$, then $|r\text{-}DT(n)| = 2$. If $r < n$, then $|r\text{-}DT(n)| \leq (8n)^{(en/r)^r}$.

4.2. Efficient robust learning of decision trees of rank r

Now we present a noise-tolerant Occam algorithm for r-DT(n) and hence conclude that r-DT(n) is polynomially learnable in the presence of classification noise.

We give a noise-tolerant Occam algorithm for r-DT(n) in figures 6 and 7. The subroutine *RFINDT* that finds a decision tree of rank at most r consistent with a certain large fraction of the given sample is an extension of the algorithm *NODL* for k-DL and based on the FIND procedure by Ehrenfeucht and Haussler (1989) to find a decision tree of rank at most r consistent with the given sample in the absence of noise.

Now we show that *NODT* is a noise-tolerant Occam algorithm for decision trees of rank r, and hence the class of decision trees of rank at most r is polynomially learnable in the

ALGORITHM *NODT*
Input:

- A sample S of m examples drawn from $EX_\eta()$ subject to classification noise,
- Integers $n, r \geq 0$ and positive fractions ϵ, and η_b, with $0 \leq \eta \leq \eta_b < 1/2$.

Output:

A decision tree T of rank at most r such that $F(T,S)/m \leq \eta_b + \epsilon(1-2\eta_b)/4$, or "none".

Procedure:

1. Calculate the following:

$$Q_F = \eta_b + \frac{\epsilon(1-2\eta_b)}{8};$$
$$Q_I = \frac{\epsilon(1-2\eta_b)}{4(en/r)^r}|S|;$$

2. Let T =*RFINDT*(S, r, Q_F, Q_I);
3. Output T and halt.

Figure 6. Efficient robust learning of decision trees of rank r.

presence of classification noise for any $n \geq r \geq 0$. Throughout the following sequence of lemmas and theorems, we assume that $n \geq r \geq 0$.

Lemma 13 *Suppose* $Q_F = \eta_b + \epsilon(1-2\eta_b)/8$ *and* $Q_I = \epsilon(1-2\eta_b)|S|/4(en/r)^r$. *If RFINDT*$(S, r, Q_F, Q_I)$ *outputs a decision tree* T *(not "none"), then* T *is a decision tree of rank at most* r *and*

$$\frac{F(T, S)}{|S|} \leq \eta_b + \frac{\epsilon(1-2\eta_b)}{4}.$$

Proof. First, it is clear that if *RFINDT*(S, r, Q_F, Q_I) returns a decision tree (which occurs either in step 1, 4b, or 4c), then by the definitions of reduced and rank, it will be a reduced decision tree over V_n of rank at most r.

Next, any decision tree T output by *RFINDT* has the following property. For any leaf, say j, of T, let $S(j)$ denote the set of all examples in S that reach the leaf j. Then $F(T, S(j))/|S(j)| \leq Q_F$ or $|S(j)| \leq Q_I$. Therefore the total number $F(T, S)$ of disagreements of T with S is at most the sum of $\Sigma_{\text{all leaves } j} F(T, S(j))$ and $Q_I(en/r)^r/2$, which are bounded by the quantities estimated in the following inequalities:

ALGORITHM $RFINDT(S, r, Q_F, Q_I)$
Input:

A sample S, integers $r, Q_I \geq 0$ and a positive fraction Q_F.

Output:

A decision tree T of rank at most r or "none".

Procedure:

1. If $F(1,S)/|S| \leq Q_F$, stop and return the decision tree $T = 1$;
   If $F(0,S)/|S| \leq Q_F$, stop and return the decision tree $T = 0$;
2. If $|S| \leq Q_I$ and $F(1,S) \leq F(0,S)$, stop and return the decision tree $T = 1$;
   If $|S| \leq Q_I$ and $F(0,S) \leq F(1,S)$, stop and return the decision tree $T = 0$;
3. If $r = 0$, stop and return "none";
4. For each informative variable $x \in V_n$
   (a) Let $T_0^x = RFINDT(S_0^x, r-1, Q_F, Q_I)$ and $T_1^x = RFINDT(S_1^x, r-1, Q_F, Q_I)$;
   (b) If both recursive calls are successful (i.e., neither $T_0^x =$"none", nor $T_1^x =$"none"), then stop and return the decision tree with root labeled x, 0-subtree T_0^x and 1-subtree T_1^x;
   (c) If one recursive call is successful but the other is not, then
      i. Reexecute the unsuccessful recursive call with rank bound r instead of $r-1$;
      ii. If the reexecuted call is now successful, then let T be the decision tree with root labeled x, 0-subtree T_0^x and 1-subtree T_1^x, else let $T =$"none";
      iii. Stop and return T;
5. Stop and return "none".

Figure 7. Finding a decision tree of rank r.

$$\sum_{\text{all leaves } j} F(T, S(\langle j \rangle)) \leq \sum_{\text{all leaves } j} Q_F \cdot |S(j)|$$

$$= Q_F \cdot \sum_{\text{all leaves } j} |S(j)|$$

$$= (\eta_b + \epsilon(1 - 2\eta_b)/8)|S|$$

and

$$\frac{1}{2} \cdot Q_I \cdot \left(\frac{en}{r}\right)^r \leq \frac{1}{2} \frac{\epsilon(1 - 2\eta_b)}{4(en/r)^r} |S|(en/r)^r$$

$$= (\epsilon(1 - 2\eta_b)/8)|S|$$

since T is reduced and the maximum number of leaves of any reduced decision trees over V_n of rank r is bounded by $(en/r)^r$ by lemma 12. Hence $F(T, S)/|S| \leq \eta_b + \epsilon(1 - 2\eta_b)/4$. □

Lemma 14 *Let T be a reduced decision tree over V_n of rank r, S be a sample consistent with T, and x be a variable that appears in T. Let T_0^x (T_1^x) denoted the decision tree obtained by replacing every subtree with root labeled x of T by the 0-subtree (1-subtree) of that subtree. Then T_0^x (T_1^x) is a reduced decision tree of rank at most r consistent with S_0^x (S_1^x).*

Proof. The proof is straightforward from the definitions of "rank" and "reduced," and the observation that if a decision tree T is consistent with a sample S, then T is consistent with any subset of S, and for every example $\langle \vec{a}, l \rangle$ in S_0^x (S_1^x) and for $x_i = x$ and $\vec{a} = (a_1, \ldots, a_n)$, $a_i = 0$ ($a_i = 1$). □

For a term t over V_n, let $var(t) = \{x \in V_n \mid x \text{ or its negation } \bar{x} \text{ appears in } t\}$. For a sample S and a term t, let $S_{[t]} = \{\langle \vec{a}, l \rangle \in S \mid t(\vec{a}) = 1\}$.

We say a decision tree T is *correct w.r.t.* a sample S drawn from $EX_\eta(\,)$ for a decision tree T_U if for every example $\langle \vec{a}, l \rangle$ in S, $f_T(\vec{a}) = f_{T_U}(\vec{a})$.

Lemma 15 *Suppose $Q_F = \eta_b + \epsilon(1 - 2\eta_b)/8$ and $Q_I = \epsilon(1 - 2\eta_b)|S|/4(en/r)^r$. Suppose also that S consists of m examples drawn from $EX_\eta(\,)$ for the target decision tree over V_n of rank r and*

$$m \geq \frac{128(en/r)^r}{\epsilon^3(1 - 2\eta_b)^3} \ln\left(\frac{2 \cdot 3^n}{\delta}\right).$$

Then with probability at least $1 - \delta/2$, RFINDT(S, r, Q_F, Q_I) outputs a decision tree T.

Proof. Let T_U denote the target reduced decision tree over V_n of rank r to be learned, and a sample S is drawn from $EX_\eta(\,)$ for T_U.

First, we prove that with probability at least $1 - \delta/2$, a sample S is drawn such that for all terms t over V_n, whenever $|S_{[t]}| \geq Q_I$, the number of occurrences of classification noise in $S_{[t]}$ is at most $Q_F \cdot |S_{[t]}|$. Let a term t over V_n be fixed. By Hoeffding's inequality lemma, whenever $|S_{[t]}| \geq Q_I$, the probability that classification noise occurs more than $Q_F \cdot |S_{[t]}|$ times is at most

$$\begin{aligned} GE(\eta, |S_{[t]}|, Q_F) &\leq GE(\eta_b, |S_{[t]}|, \eta_b + \epsilon(1 - 2\eta_b)/8) \\ &\leq e^{-2(\epsilon(1-2\eta_b)/8)^2|S_{[t]}|} \\ &\leq e^{-2(\epsilon(1-2\eta_b)/8)^2 Q_I}, \end{aligned}$$

and the lower bound on m implies that this is less than $\delta/(2 \cdot 3^n)$. Since there are at most 3^n terms over V_n, the probability that for all terms t over V_n, the number of occurrences of classification noise in $S_{[t]}$ is at most $Q_F \cdot |S_{[t]}|$ is at least $1 - \delta/2$. This completes the proof.

Second, we assume that for all terms t over V_n, the number of occurrences of classfication noise in $S_{[t]}$ is at most $Q_F \cdot |S_{[t]}|$ whenever $|S_{[t]}| \geq Q_I$. We show that this assumption implies that

> for a term t over V_n, if there is a correct reduced decision tree T over $V_n - var(t)$ of rank at most r' w.r.t. $S_{[t]}$, then $RFINDT(S_{[t]}, r', Q_F, Q_I)$ outputs a decision tree. (1)

We prove it by induction on r' and the number i of variables in $V_n - var(t)$.

If $r' = 0$ or $i = 0$, a correct reduced decision tree T of rank at most r' w.r.t. $S_{[t]}$ consists of only a root node labeled either 0 or 1 (i.e., $T = 0$ or $T = 1$). By the above assumption, $|S_{[t]}| \leq Q_I$, or $F(0, S_{[t]}) \leq Q_F \cdot |S_{[t]}|$, or $F(1, S_{[t]}) \leq Q_F \cdot |S_{[t]}|$. Hence *RFINDT* outputs a decision tree.

Next suppose that (1) holds for $r' - 1$ and i, and for r' and $i - 1$. In the case that a correct decision tree T w.r.t. $S_{[t]}$ consists of only a root node (i.e., $T = 0$ or $T = 1$), by the above assumption, *RFINDT* outputs a decision tree $T = 0$ or $T = 1$. In the case that T has more than two nodes, we prove that RFINDT cannot return "none" in step 5 and 4c. First we prove that *RFINDT* cannot reach step 5 and return "none." Let y be the variable labeled at the root node in a correct reduced decision tree T of rank at most r'. Since *RFINDT* will eventually find the variable y when both recursive calls for any other informative variable x in step 4a are not successful, we assume that *RFINDT* chooses the variable y as an informative variable in step 4. (In the case that the variable y is not informative on $S_{[t]}$, either $S_{[t]_0^y}$ or $S_{[t]_1^y}$ is empty. We assume that $S_{[t]_0^y}$ is empty. Then $S_{[t]} = S_{[t]_1^y} = S_{[t \wedge y]}$ and by lemma 14, the 1-subtree of T is a correct reduced decision tree over $V_n - var(t \wedge y)$ of rank at most r' w.r.t. $S_{[t \wedge y]}$. By the inductive hypothesis, $RFINDT(S_{[t \wedge y]}, r', Q_F, Q_I)$ outputs a decision tree, and so does $RFINDT(S_{[t]}, r', Q_F, Q_I)$.) By the definition of rank,

1. both the 0-subtree and the 1-subtree of T have the rank $r' - 1$, or
2. either the 0-subtree or the 1-subtree of t has the rank at most $r' - 1$ and the other has the rank r'.

In both cases, by lemma 14, the 0-subtree of T is a correct reduced decision tree over $V_n - var(t) - \{y\}$ w.r.t. $S_{[t]_0^y}$, and the 1-subtree is a correct reduced decision tree over $V_n - var(t) - \{y\}$ w.r.t. $S_{[t]_1^y}$. By the inductive hypothesis, both recursive calls *RFINDT* for $S_{[t]_0^y}$ and $S_{[t]_1^y}$ are successful, and hence $RFINDT(S_{[t]}, r', Q_F, Q_I)$ outputs a decision tree.

Next we prove that *RFINDT* cannot return "none" in step 4c. Assume that in step 4a, $RFINDT(S_{[t]_0^x}, r' - 1, Q_F, Q_I)$ is successful for an informative variable x. Note that any informative variable does not appear in the term t. If x does not appear in T, then T is a correct reduced decision tree over $V_n - var(t) - \{x\}$ of rank at most r' w.r.t. $S_{[t]_1^x} = S_{[t \wedge x]}$. If x appears in T, then by lemma 14, T_1^x is a correct reduced decision tree over $V_n - var(t) - \{x\}$ of rank at most r' w.r.t. $S_{[t]_1^x}$. By the inductive hypothesis, $RFINDT(S_{[t]_1^x}, r', Q_F, Q_I)$ returns a decision tree. This completes the proof of (1).

Since T_U is a correct reduced decision tree over V_n of rank r w.r.t. S, $RFINDT(S, r, Q_F, Q_I)$ outputs a decision tree with probability at least $1 - \delta/2$. □

Lemma 16 *For any nonempty sample S drawn from $EX_\eta(\,)$ for a decision tree over V_n and $r \geq 0$, the running time of RFINDT(S, r, Q_F, Q_I)* is $O(|S|(n+1)^{2r})$.

Proof. The proof is almost same as the one for the time analysis of the procedure FIND by Ehrenfeucht and Haussler (1989).

Let $t(i, r)$ be the maximum running time needed for *RFINDT*(S, r, Q_F, Q_I) when S is drawn from $EX_\eta(\,)$ for a decision tree over V_n of rank at most r and at most i variables are informative on S. Let $m = |S|$.

If $i = 0$ or $r = 0$, then $t(i, r)$ is clearly $O(m)$. If $r \geq 1$, then the time required to test whether $F(1, S)/|S| \leq Q_F$ or $F(0, S)/|S| \leq Q_F$ (step 1), to test whether $|S| \leq Q_I$ (step 2), to determine which variables are informative (step 4), and to perform the other miscellaneous tests is $O(mn)$. Each of the two recursive calls for an informative variable x in step 4a takes time at most $t(i-1, r-1)$, since the variable x is no longer informative in either S_0^x or S_1^x. Since there are at most i informative variables on S, these calls are made at most i times in the course of the loop of step 4, which gives a total time for all executions of step 4a of at most $2it(i-1, r-1)$. The only remaining step is 4(c)i, where a recursive call is made either to *RFINDT*(S_0^x, r, Q_F, Q_I) or *RFINDT*(S_1^x, r, Q_F, Q_I) for some informative variable x. This takes time at most $t(i-1, r)$. Since this step terminates the loop, this call is made at most once. It follows that for $r \geq 1$,

$$t(i, r) \leq O(mn) + 2i \cdot t(i-1, r-1) + t(i-1, r),$$

which is bounded by $O(mn(i+1)^{2r-1} + m(i+1)^{2r})$ (see Ehrenfeucht & Haussler, 1989).

Since $i \leq n$ and $m = |S|$, this implies that the running time for *RFINDT*(S, r, Q_F, Q_I) is $O(|S|(n+1)^{2r})$. □

Theorem 17 *NODT is a noise-tolerant Occam algorithm for decison trees of rank r.*

Proof. It is straightforward from lemmas 13, 15, and 16. □

Now we have the main theorem.

Theorem 18 *r-DT(n) is polynomially learnable in the presence of classification noise.*

Proof. Since *r-DT*(n) has polynomial representation size by lemma 12, it is straightforward from theorems 5 and 17. □

Elomaa and Kivinen (1991) had independently achieved this result based on our result (theorem 5) for noise-tolerant Occam algorithms and by using a similar technique to ours shown in section 3.

Recently Blum (1991) has shown that every decision tree of rank r can be represented as a decision list in r-DL, that is, r-DT(n) is a subclass of r-DL(n). The idea to construct a decision list in r-DL corresponding to an aribitrary decision tree of rank r is as follows: Find a leaf that is closest to the root in the decision tree. Form a term that corresponds to the path from the root to the leaf. Add a pair of the term and the label of the leaf to the decision list. Delete the leaf and the node closest to the leaf from the tree. Repeat the above procedure until no leaf exists.

Combined with this result, it is straightforward to show that r-DT(n) is polynomially learnable in the presence of classification noise in terms of r-DL(n) by using *NODL*. Now it is interesting for us to compare the sample size needed by *NODT* with the one by *NODL*. By lemma 15, *NODT* gives the sample size $m \geq 128(en/r)^r/\epsilon^3(1 - 2\eta_b)^3 \ln (2 \cdot 3^n/\delta)$. By lemma 8, *NODL* gives the sample size $m \geq 128M/\epsilon^3(1 - 2\eta_b)^3 \ln (2^{M+1} M/\delta)$, where $M = O(n^r)$. Thus *NODT* might give a better bound than *NODL*.

5. Conclusions

We have focused on the problem of learning Boolean functions represented by decision trees from noisy examples. We have developed a technique of building efficient robust learning algorithms, called *noise-tolerant Occam algorithms*, in the presence of classification noise, and have shown that by using a noise-tolerant Occam algorithm for a class of Boolean functions, one can construct a polynomial-time algorithm for learning the class in the presence of classification noise. Next we have presented a noise-tolerant Occam algorithm for k-DL and hence conclude that k-DL is polynomially learnable in the presence of classification noise. This strictly increases the class of Boolean functions that are known to be polynomially learnable in the presence of classification noise: the only example of a class of Boolean functions that has been shown to be polynomially learnable in the presence of classification noise is k-CNF (Angluin & Laird, 1988), and k-DL properly includes k-CNF. Further, we have extended the noise-tolerant Occam algorithm for decision lists to one for decision trees, and we conclude that the class of decision trees of rank at most r is polynomially learnable in the presence of classification noise.

It is important for us to have empirical studies and to see how well our algorithms work in a practical situation. Quinlan's (1986b) research is a good empirical study of the effects of different sorts of noise on learning decision trees. Mingers (1989) has studied an empirical comparison of several pruning methods for learning decision trees. It is an interesting future problem to evaluate our algorithms empirically and to compare the results to Quinlan's results and to the results using those pruning methods for handling noisy data.

Acknowledgments

The author would like to thank the anonymous referees for their careful reviewing and helpful comments. The author would also like to thank Manfred Warmuth, Masako Takahashi, Philip Laird, and Jyrki Kivinen for their useful suggestions and comments. The author would especially like to thank Robert Schapire for pointing out an error in lemma 8 on an earlier draft.

References

Angluin, D., & Laird, P. (1988). Learning from noisy examples. *Machine Learning* 2, 343-370.

Anthony, M., & Biggs, N. (1992). *Computational learning theory.* Cambridge Tracts in Theoretical Computer Science. Cambridge: Cambridge University Press.

Blum, A. (1991). Basic argument for rank-k DTs being contained in k-DLs. Unpublished manuscript.

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M.K. (1987). Occam's razor. *Information Processing Letters, 24*, 377–380.

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M.K. (1989). Learnability and the Vapnik–Chervonenkis dimension. *Journal of the ACM, 36*, 929–965.

Breiman, L., Friedman, J.H., Olshen, R.A., & Stone, C.J. (1984). *Classification and regression trees*. Wadsworth Statistics/Probability Series. California: Wadsworth International.

Clark, P., & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning, 3*, 261–283.

Ehrenfeucht, A., & Haussler, D. (1989). Learning decision trees from random examples. *Information and Computation, 82*, 231–246.

Elomaa, T., & Kivinen, J. (1991). Learning decision trees from noisy examples (Report A-1991-3). Helsinki: University of Helsinki, Department of Computer Science.

Goldman, S.A., Kearns, M.J., & Schapire, R.E. (1990). Exact identification of circuits using fixed points of amplification functions. *Proceedings of the Thirty-First IEEE Symposium on Foundations of Computer Science* (pp. 193–202). IEEE Computer Society Press.

Hoeffding W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association, 58*, 13–30.

Kearns, M.J. (1990). *The computational complexity of machine learning*. Cambridge, MA: MIT Press.

Kearns, M., & Li, M. (1988). Learning in the presence of malicious errors. *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing* (pp. 267–280). New York: The Association for Computing Machinery.

Laird, P.D. (1988). *Learning from good and bad data*. Boston, MA: Kluwer Academic Publishers.

Mingers, J. (1989). An empirical comparison of pruning methods for decision tree induction. *Machine Learning, 4*, 227–243.

Natarajan, B.K. (1989). On learning sets and functions. *Machine Learning, 4*, 67–97.

Pagallo, G.M. (1990). *Adaptative decision tree algorithms for learning from examples* (Technical Report UCSC-CRL-90-27). Doctoral dissertation, Department of Computer and Information Sciences, University of California, Santa Cruz, CA.

Pith, L., & Valiant, L.G. (1988). Computational limitations on learning from examples. *Journal of the ACM, 35*, 965–984.

Quinlan, J.R. (1986a). Induction of decision trees. *Machine Learning, 1*, 81–106.

Quinlan, J.R. (1986b). The effect of noise on concept learning. In R.S. Michalski, J.G. Carbonell, & T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Vol. 2). Los Altos, CA: Morgan Kaufmann.

Rivest, R.L. (1987). Learning decision lists. *Machine Learning, 2*, 229–246.

Schapire, R.E. (1991). *The design and analysis of efficient learning algorithms* (Technical Report MIT/LCS/TR-493). Doctoral dissertation, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA.

Sloan, R. (1988). Types of noise in data for concept learning. *Proceedings of the 1988 Workshop on Computational Learning Theory* (pp. 91–96). San Mateo, CA: Morgan Kaufmann.

Utgoff, P.E. (1989). Incremental induction of decision trees. *Machine Learning, 4*, 161–186.

Valiant, L.G. (1984). A theory of the learnable. *Communications of the ACM, 27*, 1134–1142.

Valiant, L.G. (1985). Learning disjunctions of conjunctions. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 560–566). Los Angeles, CA: Morgan Kaufmann.

Received July 2, 1991
Accepted August 11, 1992
Final Manuscript August 20, 1992