Learning in the Presence of Malicious Errors *

Michael Kearns†
AT&T Bell Laboratories

Ming Li‡
University of Waterloo

Abstract

In this paper we study an extension of the distribution-free model of learning introduced by Valiant [23] (also known as the *probably approximately correct* or *PAC* model) that allows the presence of malicious errors in the examples given to a learning algorithm. Such errors are generated by an adversary with unbounded computational power and access to the entire history of the learning algorithm's computation. Thus, we study a worst-case model of errors.

Our results include general methods for bounding the rate of error tolerable by any learning algorithm, efficient algorithms tolerating nontrivial rates of malicious errors, and equivalences between problems of learning with errors and standard combinatorial optimization problems.

1 Introduction

In this paper, we study a practical extension to Valiant's distribution-free model of learning: the presence of errors (possibly maliciously generated by an adversary) in the sample data. The distribution-free model typically makes the idealized assumption that the oracles *POS* and *NEG* (returning positive and negative examples of the unknown target concept) always faithfully return untainted examples of the target representation drawn according to the target distributions. In many environments, however, there is always some chance that an erroneous example is given to the learning algorithm. In a training session for an expert system, this might be due to an occasionally faulty teacher; in settings where the examples are being transmitted electronically, it might be due to unreliable communication equipment.

Since one of the strengths of Valiant's model is the lack of assumptions on the probability distributions from which examples are drawn, we seek to preserve this generality by making no assumptions on the *nature* of the errors that occur. That is, we wish to avoid demanding algorithms that work under any target distributions while at the same time assuming that the errors in the examples have some "nice" form. Such well-behaved sources of error seem difficult to justify in a real

*A preliminary version of this research appears in the *Proceedings of the Twentieth Annual A.C.M. Symposium on the Theory of Computing.*

†This research was done while the author was a graduate student at Harvard University and visiting the University of California at Santa Cruz. Supported by an A.T.& T. Bell Laboratories scholarship and grants N00014-85-K-0445 and N00014-86-K-0454 from the Office of Naval Research and DCR-8606366 from the National Science Foundation. Author's address: AT&T Bell Laboratories, 600 Mountain Ave. Room 2A-423, P.O. Box 636, Murray Hill, NJ 07974-0636.

‡This research was done while the author was at Harvard University. Supported by grants N00014-85-K-0445 from the Office of Naval Research and DCR-8606366 from the National Science Foundation. Author's address: Department of Computer Science, University of Waterloo, Davis Center, Room 2339, Waterloo, Ontario N2L 3G1, Canada.

computing environment, where the rate of error may be small, but data may become badly mangled by highly unpredictable forces whenever errors do occur, for example in the case of hardware errors. Thus, we study a worst-case or *malicious* model of errors, in which the errors are generated by an adversary whose goal is to foil the learning algorithm.

The study of learning from examples with malicious errors was initiated by Valiant [24], where it is assumed that there is a fixed probability β of an error occurring independently on each request for an example. This error may be of an arbitrary nature — in particular, it may be chosen by an adversary with unbounded computational resources, and exact knowledge of the target representation, the target distributions, and the current internal state of the learning algorithm.

In this paper we study the *optimal malicious error rate* $E_{MAL}(C)$ for a representation class C — that is, the largest value of β that can be tolerated by any learning algorithm (not necessarily polynomial time) for C. Note that we expect the optimal error rate to depend on ϵ and δ (and n in the case of a parameterized target class C). An upper bound on $E_{MAL}(C)$ corresponds to a hardness result placing limitations on the rate of error that can be tolerated; lower bounds on $E_{MAL}(C)$ are obtained by giving algorithms that tolerate a certain rate of error.

Using a proof technique called the method of induced distributions, we obtain general upper bounds on $E_{MAL}(C)$ and apply these results to many representation classes. We also obtain lower bounds on $E_{MAL}^{poly}(C)$ (the largest rate of malicious error tolerated by a polynomial-time learning algorithm for C) by giving efficient learning algorithms for these same classes and analyzing their error tolerance. In several cases the upper and lower bounds on $E_{MAL}^{poly}(C)$ meet. A canonical method of transforming standard learning algorithms into error-tolerant algorithms is given, and we give approximation-preserving reductions between standard combinatorial optimization problems such as set cover and natural problems of learning with errors. Several of our results also apply to a more benign model of *classification noise* defined by Angluin and Laird [1], in which the underlying target distributions are unaltered, but there is some probability that a positive example is incorrectly classified as being negative, and vice-versa.

Several themes are brought out. One is that error tolerance need not come at the expense of efficiency or simplicity. We show that there are representation classes for which the optimal malicious error rate can be achieved by algorithms that run in polynomial time and are easily coded. For example, we show that a polynomial-time algorithm for learning monomials with errors due to Valiant [24] tolerates the largest malicious error rate possible for any algorithm that uses only positive examples, polynomial-time or otherwise. We give an efficient learning algorithm for the class of symmetric functions that tolerates the optimal malicious error rate and uses an optimal number of examples.

Another theme is the importance of using both positive and negative examples whenever errors (either malicious errors or classification noise errors) are present. Several existing learning algorithms use only positive examples or only negative examples (see e.g. Valiant [23] and Blumer et al. [5]). We demonstrate strong upper bounds on the tolerable error rate when only one type is used, and show that this rate can be provably increased when both types are used. In addition to proving this for the class of symmetric functions, we give an efficient algorithm that provides a strict increase in the malicious error rate over the positive-only algorithm of Valiant [24] for the class of monomials.

A third theme is that there are strong ties between learning with errors and more traditional problems in combinatorial optimization. We give a reduction from learning monomials with errors to a generalization of the weighted set cover problem, and give an approximation algorithm for

this problem (generalizing the greedy algorithm analyzed by several authors [7, 11, 18]) that is of independent interest. This approximation algorithm is used as a subroutine in a learning algorithm that tolerates an improved error rate for monomials. In the other direction, we prove that for the class of monomials M, approaching the optimal error rate $E_{MAL}(M)$ with a polynomial-time algorithm using hypothesis space M is at least as hard as finding an efficient approximation algorithm with an improved performance guarantee for the set cover problem. This suggests that there are classes for which the optimal error rate that can be tolerated *efficiently* may be considerably smaller than the optimal *information-theoretic* rate. The best approximation known for the set cover problem remains the greedy algorithm analyzed by Chvatal [7], Johnson [11], Lovasz [17], and Nigmatullin [18]. Finally, we give a canonical reduction that allows many learning with errors problems to be studied as equivalent optimization problems, thus allowing one to sidestep some of the difficulties of analysis in the distribution-free model. Similar results are given for the error-free model by Haussler et al. [10].

We now give a brief survey of other studies of error in the distribution-free model. Valiant [24] modified his initial definitions of learnability to include the presence of errors in the examples. He also gave a generalization of his algorithm for learning monomials from positive examples, and analyzed the rate of malicious error tolerated by this algorithm. Valiant's results led him to suggest the possibility that "the learning phenomenon is only feasible with very low error rates" (at least in the distribution-free setting with malicious errors); some of the results presented in this paper can be viewed as giving formal verification of this intuition. On the other hand, some of our algorithms provide hope that if one can somehow reliably control the *rate* of error to a small amount, then errors of an arbitrary nature can be compensated for by the learning process.

Angluin and Laird [1] subsequently modified Valiant's definitions to study a non-malicious model of errors, defined in Section 2.3 as the *classification noise model*. Their results demonstrate that under stronger assumptions on the nature of the errors, large rates of error can be tolerated by polynomial-time algorithms for nontrivial representation classes. Shackelford and Volper [21] investigate a model of random noise in the instances rather than the labels, and Sloan [22] and Laird [15] discuss a number of variants of both the malicious error and classification noise models.

2 Definitions for Distribution-free Learning

In this section we give definitions and motivation for the model of machine learning we study. This model was first defined by Valiant [23] in 1984; he then went on to generalize his definitions to allow errors in 1985 [24]. In addition to the basic definitions and notation, we give the form of Chernoff bounds we use, define the Vapnik-Chervonenkis dimension, and define a number of classes of representations whose error-tolerant learnability we will study.

2.1 Representing subsets of a domain

Concept classes and their representation. Let X be a set called a *domain* (also sometimes referred to as the *instance space*). We think of X as containing encodings of all objects of interest to us in our learning problem. For example, each instance in X may represent a different object in a particular room, with discrete attributes representing properties such as color, and continuous values representing properties such as height. The goal of a learning

algorithm is then to infer some unknown subset of X, called a *concept*, chosen from a known *concept class*.

For computational purposes we always need a way of *naming* or *representing* concepts. Thus, we formally define a *representation class over* X to be a pair (σ, C), where $C \subseteq \{0,1\}^*$ and σ is a mapping $\sigma : C \to 2^X$ (here 2^X denotes the power set of X). For $c \in C$, $\sigma(c)$ is called a *concept* over X; the image space $\sigma(C)$ is the *concept class* that is *represented* by (σ, C). For $c \in C$, we define $pos(c) = \sigma(c)$ (the *positive examples* of c) and $neg(c) = X - \sigma(c)$ (the *negative examples* of c). The domain X and the mapping σ will usually be clear from the context, and we will simply refer to the *representation class* C. We will sometimes use the notation $c(x)$ to denote the value of the characteristic function of $\sigma(c)$ on the domain point x; thus $x \in pos(c)$ ($x \in neg(c)$, respectively) and $c(x) = 1$ ($c(x) = 0$, respectively) are used interchangeably. We assume that domain points $x \in X$ and representations $c \in C$ are efficiently encoded using any of the standard schemes (see Garey and Johnson [9]), and denote by $|x|$ and $|c|$ the length of these encodings measured in bits.

Parameterized representation classes. In this paper we will study *parameterized* classes of representations. Here we have a stratified domain $X = \bigcup_{n \geq 1} X_n$ and representation class $C = \bigcup_{n \geq 1} C_n$. The parameter n can be regarded as an appropriate measure of the complexity of concepts in $\sigma(C)$ (such as the number of domain attributes), and we assume that for a representation $c \in C_n$ we have $pos(c) \subseteq X_n$ and $neg(c) = X_n - pos(c)$. For example, X_n may be the set $\{0,1\}^n$, and C_n the class of all Boolean formulae over n variables whose length is at most n^2. Then for $c \in C_n$, $\sigma(c)$ would contain all satisfying assignments of the formula c.

Efficient evaluation of representations. In general, we will be primarily concerned with learning algorithms that are computationally efficient. In order to prevent this demand from being vacuous, we need to insure that the *hypotheses* output by a learning algorithm can be efficiently evaluated as well. Thus if C is a representation class over X, we say that C is *polynomially evaluatable* if there is a (probabilistic) polynomial-time *evaluation algorithm* A that on input a representation $c \in C$ and a domain point $x \in X$ outputs $c(x)$. Note that a class being polynomially evaluatable simply means that it contains only "small" representations, that is, representations that can be written down in polynomial time. All representation classes considered here are polynomially evaluatable. It is worth mentioning at this point that Schapire [20] has shown that if a representation class is not polynomially evaluatable, then it is not efficiently learnable in our model. Thus, perhaps not surprisingly we see that classes that are not polynomially evaluatable constitute "unfair" learning problems.

Samples. A *labeled example* from a domain X is a pair $< x, b >$, where $x \in X$ and $b \in \{0,1\}$. A *labeled sample* $S = < x_1, b_1 >, \ldots, < x_m, b_m >$ from X is a finite sequence of labeled examples from X. If C is a representation class, a *labeled example of* $c \in C$ is a labeled example $< x, c(x) >$, where $x \in X$. A *labeled sample of* c is a labeled sample S where each example of S is a labeled example of c. In the case where all labels b_i or $c(x_i)$ are 1 (0, respectively), we may omit the labels and simply write S as a list of points $x_1, \ldots, x_m$, and we call the sample a *positive* (*negative*, respectively) sample.

We say that a representation h and an example $< x, b >$ *agree* if $h(x) = b$; otherwise they *disagree*. We say that a representation h and a sample S are *consistent* if h agrees with each example in S; otherwise they are *inconsistent*.

2.2 Distribution-free learning

Distributions on examples. On any given execution, a learning algorithm for a representation class C will be receiving examples of a single distinguished representation $c \in C$. We call this distinguished c the *target representation*. Examples of the target representation are generated probabilistically as follows: let D_c^+ be a fixed but arbitrary probability distribution over $pos(c)$, and let D_c^- be a fixed but arbitrary probability distribution over $neg(c)$. We call these distributions the *target distributions*. When learning c, learning algorithms will be given access to two oracles, POS and NEG, that behave as follows: oracle POS (NEG, respectively) returns in unit time a positive (negative, respectively) example of the target representation, drawn randomly according to the target distribution D_c^+ (D_c^-, respectively).

The distribution-free model is sometimes defined in the literature with a single target distribution over the entire domain; the learning algorithm is then given labeled examples of the target concept drawn from this distribution. We choose to explicitly separate the distributions over the positive and negative examples to facilitate the study of algorithms that learn using only positive examples or only negative examples. These models, however, are equivalent with respect to polynomial-time computation, as is shown by Haussler et al. [10].

Given a fixed target representation $c \in C$, and given fixed target distributions D_c^+ and D_c^-, there is a natural measure of the *error* (with respect to c, D_c^+ and D_c^-) of a representation h from a representation class H. We define $e_c^+(h) = D_c^+(neg(h))$ (i.e., the weight of the set $neg(h)$ under the probability distribution D_c^+) and $e_c^-(h) = D_c^-(pos(h))$ (the weight of the set $pos(h)$ under the probability distribution D_c^-). Note that $e_c^+(h)$ (respectively, $e_c^-(h)$) is simply the probability that a random positive (respectively, negative) example of c is identified as negative (respectively, positive) by h. If both $e_c^+(h) < \epsilon$ and $e_c^-(h) < \epsilon$, then we say that h is an *ϵ-good* hypothesis (with respect to c, D_c^+ and D_c^-); otherwise, h is *ϵ-bad*. We define the *accuracy* of h to be the value $\min(1 - e_c^+(h), 1 - e_c^-(h))$.

It is worth noting that our definitions so far assume that the hypothesis h is deterministic. However, this need not be the case; for example, we can instead define $e_c^+(h)$ to be the probability that h classifies a random positive example of c as negative, where the probability is now over both the random example and the coin flips of h. All of the results presented here hold under these generalized definitions.

When the target representation c is clear from the context, we will drop the subscript c and simply write D^+, D^-, e^+ and e^-.

In the definitions that follow, we will demand that a learning algorithm produce with high proability an ϵ-good hypothesis regardless of the target representation and target distributions. While at first this may seem like a strong criterion, note that the error of the hypothesis output is always measured with respect to the same target distributions on which the algorithm was trained. Thus, while it is true that certain examples of the target representation may be extremely unlikely to be generated in the training process, these same examples intuitively may be "ignored" by the hypothesis of the learning algorithm, since they contribute a negligible amount of error.

Learnability. Let C and H be representation classes over X. Then C is *learnable from examples by H* if there is a (probabilistic) algorithm A with access to POS and NEG, taking inputs

ϵ, δ, with the property that for any target representation $c \in C$, for any target distributions D^+ over $pos(c)$ and D^- over $neg(c)$, and for any inputs $0 < \epsilon, \delta < 1$, algorithm A halts and outputs a representation $h_A \in H$ that with probability greater than $1-\delta$ satisfies $e^+(h_A) < \epsilon$ and $e^-(h_A) < \epsilon$.

We call C the *target class* and H the *hypothesis class*; the output $h_A \in H$ is called the *hypothesis* of A. A will be called a *learning algorithm* for C. If C and H are polynomially evaluatable, and A runs in time polynomial in $1/\epsilon, 1/\delta$ and $|c|$ then we say that C is *polynomially learnable from examples by* H; if C is parameterized we also allow the running time of A to have polynomial dependence on the parameter n.

We will drop the phrase "from examples" and simply say that C is *learnable by* H, and C is *polynomially learnable by* H. We say C is *polynomially learnable* to mean that C is polynomially learnable by H for some polynomially evaluatable H. We will sometimes call ϵ the *accuracy parameter* and δ the *confidence parameter*.

Thus, we ask that for any target representation and any target distributions, a learning algorithm finds an ϵ-good hypothesis with probability at least $1-\delta$. A primary goal of research in this model is to discover which representation classes C are polynomially learnable.

We refer to Valiant's model as the *distribution-free* model, to emphasize that we seek algorithms that work for any target distributions. It is also known in the literature as the *probably approximately correct* model.

Positive-only and negative-only learning algorithms. We will sometimes study learning algorithms that need only positive examples or only negative examples. If A is a learning algorithm for a representation class C, and A makes no calls to the oracle NEG (respectively, POS), then we say that A is a *positive-only* (respectively, *negative-only*) learning algorithm, and C is *learnable from positive examples* (*learnable from negative examples*). Note that although the learning algorithm receives only one type of examples, the hypothesis output must still be accurate with respect to *both* the positive and negative distributions.

Several learning algorithms in the distribution-free model are positive-only or negative-only. The study of positive-only and negative-only learning is interesting for at least two reasons. First, it helps to quantify more precisely what kind of information is required for learning various representation classes. Second, it may be important for applications where, for instance, negative examples are rare but must be classified accurately when they do occur.

2.3 Definitions for learning with errors

Oracles with malicious errors. Let C be a representation class over a domain X, and let $c \in C$ be the target representation with target distributions D^+ and D^-. For $0 \leq \beta < 1/2$, we define two *oracles with malicious errors*, POS^{β}_{MAL} and NEG^{β}_{MAL}, that behave as follows: when oracle POS^{β}_{MAL} (respectively, NEG^{β}_{MAL}) is called, with probability $1-\beta$, a point $x \in pos(c)$ (respectively, $x \in neg(c)$) randomly chosen according to D^+ (respectively, D^-) is returned, as in the error-free model, but with probability β, a point $x \in X$ about which absolutely no assumptions can be made is returned. In particular, this point may be dynamically and maliciously chosen by an adversary who has knowledge of c, D^+, D^-, β and the internal state of the learning algorithm. This adversary also has unbounded computational resources. For

convenience we assume that the adversary does not have knowledge of the outcome of future coin flips of the learning algorithm or the points to be returned in future calls to POS^{β}_{MAL} and NEG^{β}_{MAL} (other than those that the adversary may himself decide to generate on future errors). These assumptions may in fact be removed, as our results will show, resulting in a stronger model where the adversary may choose to modify in any manner a fixed fraction β of the sample to be given to the learning algorithm. Such a model realistically captures situations such as "error bursts", which may occur when transmission equipment malfunctions repeatedly for a short amount of time.

Learning from oracles with malicious errors. Let C and H be representation classes over X. Then for $0 \leq \beta < 1/2$, we say that C is *learnable by H with malicious error rate β* if there is a (probabilistic) algorithm A with access to POS^{β}_{MAL} and NEG^{β}_{MAL}, taking inputs ϵ, δ and β_0, with the property that for any target representation $c \in C$, for any target distributions D^+ over $pos(c)$ and D^- over $neg(c)$, and for any input values $0 < \epsilon, \delta < 1$ and $\beta \leq \beta_0 < 1/2$, algorithm A halts and outputs a representation $h_A \in H$ that with probability at least $1 - \delta$ satisfies $e^+(h_A) < \epsilon$ and $e^+(h_A) < \epsilon$.

We will also say that A is a *β-tolerant learning algorithm for C*. In this definition of learning, polynomial-time means polynomial in $1/\epsilon, 1/\delta$ and $1/(1/2 - \beta_0)$, as well as polynomial in n in the case of parameterized C.

The input β_0 is intended to provide an upper bound on the error rate for the learning algorithm, since in practice we do not expect to have exact knowledge of the "true" error rate β (for instance, it is reasonable to expect the error rate to vary somewhat with time). The dependence on $1/(1/2 - \beta_0)$ for polynomial-time algorithms provides the learning algorithm with more time as the error rate approaches $1/2$, since an error rate of $1/2$ renders learning impossible for any algorithm, polynomial-time or otherwise (this is because the labels are essentially the outcomes of the flip of a fair coin). However, we will shortly see that the input β_0 and the dependence of the running time on $1/(1/2 - \beta_0)$ are usually unnecessary, since for learning under arbitrary target distributions to be possible we must have $\beta < \epsilon/(1 + \epsilon)$ (under very weak restrictions on C). This is Theorem 1. However, we include β_0 in our definitions since these dependencies may be meaningful for learning under restricted target distributions.

It is important to note that in this definition, we are *not* asking learning algorithms to "fit the noise" in the sense of achieving accuracy in predicting the behavior of the tainted oracles POS^{β}_{MAL} and NEG^{β}_{MAL}. Rather, the conditions $e^+(h_A) < \epsilon$ and $e^+(h_A) < \epsilon$ require that the algorithm find a good predictive model of the true underlying target distributions D^+ and D^-, as in the error-free model.

In general, we expect the achievable malicious error rate to depend upon the desired accuracy ϵ and confidence δ, as well as on the parameter n in the case of parameterized representation classes. We now make definitions that will allow us to study the largest rate $\beta = \beta(\epsilon, \delta, n)$ that can be tolerated by any learning algorithm, and by learning algorithms restricted to run in polynomial time.

Optimal malicious error rates. Let A be a learning algorithm for C. We define $E_{MAL}(C, A)$ to be the largest β such that A is a β-tolerant learning algorithm for C; note that $E_{MAL}(C, A)$

is actually a function of ϵ and δ (and n in the case of parameterized C). In the case that the largest such β is not well-defined (for example, A could tolerate progressively larger rates if allowed more time), then $E_{MAL}(C, A)$ is the supremum over all malicious error rates tolerated by A. Then we define the function $E_{MAL}(C)$ to be the pointwise (with respect to ϵ, δ and n in the parameterized case) supremum of $E_{MAL}(C, A)$, taken over all learning algorithms A for C. More formally, if we write $E_{MAL}(C, A)$ and $E_{MAL}(C)$ in functional form, then $E_{MAL}(C)(\epsilon, \delta, n) = \sup_A\{E_{MAL}(C, A)(\epsilon, \delta, n)\}$. Notice that this supremum is taken over *all* learning algorithms, regardless of computational complexity. We will use the notation E^{poly}_{MAL} to denote these same quantities when the quantification is only over polynomial-time learning algorithms — thus, for instance, $E^{poly}_{MAL}(C, A)$ is the largest β such that A is a β-tolerant learning polynomial-time learning algorithm for C, and $E^{poly}_{MAL}(C)$ is the largest malicious error rate tolerated by any polynomial-time learning algorithm for C.

$E_{MAL,+}(C)$ will be used to denote E_{MAL} with quantification only over positive-only learning algorithms for C. Similar definitions are made for the negative-only malicious error rate $E_{MAL,-}$, and polynomial-time positive-only and polynomial-time negative-only malicious error rates $E^{poly}_{MAL,+}$ and $E^{poly}_{MAL,-}$.

Oracles with classification noise. Some of our results will also apply to a more benign model of errors defined by Angluin and Laird [1], which we will call the *classification noise* model. Here we have oracles POS^{β}_{CN} and NEG^{β}_{CN} that behave as follows: as before, with probability $1-\beta$, POS^{β}_{CN} returns a point drawn randomly according to the target distribution D^+. However, with probability β, POS^{β}_{CN} returns a point drawn randomly according to the *negative* target distribution D^-. Similarly, with probability $1-\beta$, NEG^{β}_{CN} draws from the correct distribution D^- and with probability β draws from D^+. This model is easily seen to be equivalent (modulo polynomial time) to a model in which a learning algorithm asks for a labeled example without being allowed to specify whether this example will be positive or negative; then the noisy oracle draws from the underlying target distributions (each with equal probability), but with probability β returns an incorrect classification with the example drawn.

These oracles are intended to model a situation in which the learning algorithm's "teacher" occasionally misclassifies a positive example as negative, and vice-versa. However, this misclassification is benign in the sense that the erroneous example is always drawn according to the "natural" environment as represented by the target distributions; thus, only the classification label is subject to error. In contrast, errors in the malicious model may involve not only misclassification, but alteration of the examples themselves, which may not be generated according to any probability distribution at all. As an example, the adversary generating the errors may choose to give significant probability to examples that have zero probability in the true target distributions. We will see throughout the paper that these added capabilities of the adversary have a crucial effect on the error rates that can be tolerated.

Learning from oracles with classification noise. Let C and H be representation classes over X. Then for $0 \leq \beta < 1/2$, we say that C is *learnable by H with classification noise rate β* if there is a (probabilistic) algorithm A with access to POS^{β}_{CN} and NEG^{β}_{CN}, taking inputs ϵ, δ and β_0, with the property that for any target representation $c \in C$, for any target distributions D^+ over $pos(c)$ and D^- over $neg(c)$, and for any input values $0 < \epsilon, \delta < 1$ and

$\beta \leq \beta_0 < 1/2$, algorithm A halts and outputs a representation $h_A \in H$ that with probability at least $1 - \delta$ satisfies $e^+(h_A) < \epsilon$ and $e^+(h_A) < \epsilon$.

Polynomial time here means polynomial in $1/\epsilon, 1/\delta$ and $1/(1/2 - \beta_0)$, as well as the polynomial in n in the case of parameterized C. As opposed to the malicious case, the input β_0 is relevant here, even in the case of arbitrary target distributions, since classification noise rates approaching $1/2$ can be tolerated by polynomial-time algorithms for some nontrivial representation classes [1].

Optimal classification noise rates. Analogous to the malicious model, we define *classification noise rates* $E_{CN}, E_{CN,+}$ and $E_{CN,-}$ for an algorithm A and representation class C, as well as polynomial-time classification noise rates $E_{CN}^{poly}, E_{CN,+}^{poly}$ and $E_{CN,-}^{poly}$.

2.4 Other definitions and notation

Sample complexity. Let A be a learning algorithm for a representation class C. Then we denote by $S_A(\epsilon, \delta)$ the number of calls to the oracles POS and NEG made by A on inputs ϵ, δ; this is a worst-case measure over all possible target representations in C and all target distributions D^+ and D^-. In the case that C is a parameterized representation class, we also allow S_A to depend on the parameter n. We call the function S_A the *sample complexity* or *sample size* of A. We denote by S_A^+ and S_A^- the number of calls of A to POS and NEG, respectively.

Chernoff bounds. We shall make extensive use of the following bounds on the area under the tails of the binomial distribution. For $0 \leq p \leq 1$ and m a positive integer, let $LE(p, m, r)$ denote the probability of at most r successes in m independent trials of a Bernoulli variable with probability of success p, and let $GE(p, m, r)$ denote the probability of at least r successes. Then for $0 \leq \alpha \leq 1$,

Fact CB1. $LE(p, m, (1 - \alpha)mp) \leq e^{-\alpha^2 mp/2}$

and

Fact CB2. $GE(p, m, (1 + \alpha)mp) \leq e^{-\alpha^2 mp/3}$

These bounds in the form they are stated are from the paper of Angluin and Valiant [2]; see also Chernoff [6]. Although we will make frequent use of Fact CB1 and Fact CB2, we will do so in varying levels of detail, depending on the complexity of the calculation involved. However, we are primarily interested in Chernoff bounds for the following consequence of Fact CB1 and Fact CB2: given an event E of probability p, we can obtain an estimate $\hat{p}$ of p by drawing m points from the distribution and letting $\hat{p}$ be the frequency with which E occurs in this sample. Then for m polynomial in $1/p$ and $1/\alpha$, $\hat{p}$ satisfies $p/2 < \hat{p} < 2p$ with probability at least $1 - \alpha$. If we also allow m to depend polynomially on $1/\beta$, we can obtain an estimate $\hat{p}$ such that $p - \beta < \hat{p} < p + \beta$ with probability at least $1 - \alpha$.

The Vapnik-Chervonenkis dimension. Let C be a representation class over X. Let $Y \subseteq X$, and define

$$\Pi_C(Y) = \{Z \subseteq Y : Z = Y \cap pos(c) \text{ for some } c \in C\}.$$

If we have $\Pi_C(Y) = 2^Y$, then we say that Y is *shattered* by C. Then we define

$$\text{VCD}(C) = \max\{|Y| : Y \text{ is shattered by } C\}.$$

If this maximum does not exist, then $\text{VCD}(C)$ is infinite. The Vapnik-Chervonenkis was originally introduced in the paper of Vapnik and Chervonenkis [25] and was first studied in the context of the distribution-free model by Blumer et al. [5].

Notational conventions. Let $E(x)$ be an event and $\psi(x)$ a random variable that depend on a parameter x that takes on values in a set X. Then for $X' \subseteq X$, we denote by $\mathbf{Pr}_{x \in X'}[E(x)]$ the probability that E occurs when x is drawn uniformly at random from X'. Similarly, $\mathbf{E}_{x \in X'}[\psi(x)]$ is the expected value of ψ when x is drawn uniformly at random from X'. We also need to work with distributions other than the uniform distribution; thus if P is a distribution over X we use $\mathbf{Pr}_{x \in P}[E(x)]$ and $\mathbf{E}_{x \in P}[\psi(x)]$ to denote the probability of E and the expected value of ψ, respectively, when x is drawn according to the distribution P. When E or ψ depend on several parameters that are drawn from different distributions we use multiple subscripts. For example, $\mathbf{Pr}_{x_1 \in P_1, x_2 \in P_2, x_3 \in P_3}[E(x_1, x_2, x_3)]$ denotes the probability of event E when x_1 is drawn from distribution P_1, x_2 from P_2, and x_3 from P_3.

2.5 Some representation classes

We now define the parametrized representation classes whose error-tolerant learnability we will study. Here the domain X_n is always $\{0,1\}^n$ and the mapping σ simply maps each formula to its set of satisfying assignments. The classes defined below are all parameterized; for each class we will define the subclasses C_n, and then C is defined by $C = \bigcup_{n \geq 1} C_n$.

Monomials: The representation class M_n consists of all conjunctions of literals over the Boolean variables $x_1, \ldots, x_n$.

kCNF: For any constant k, the representation class $k\text{CNF}_n$ consists of all Boolean formulae of the form $C_1 \wedge \cdots \wedge C_l$, where each clause C_i is a disjunction of at most k literals over the Boolean variables $x_1, \ldots, x_n$. Note that $M_n = 1\text{CNF}_n$.

kDNF: For any constant k, the representation class $k\text{DNF}_n$ consists of all Boolean formulae of the form $T_1 \vee \cdots \vee T_l$, where each term T_i is a conjunction of at most k literals over the Boolean variables $x_1, \ldots, x_n$.

Symmetric Functions: A *symmetric function* over the Boolean variables $x_1, \ldots, x_n$ is a Boolean function whose output is invariant under all permutations of the input bits. Such a function can be represented by a Boolean array of size $n+1$, where the ith entry indicates whether the function is 0 or 1 on all inputs with exactly i bits set to 1. We denote by SF_n the class of all such representations.

Decision Lists: A *decision list* [19] is a list $L = < (T_1, b_1), \ldots, (T_l, b_l) >$, where each T_i is a monomial over the Boolean variables $x_1, \ldots, x_n$ and each $b_i \in \{0,1\}$. For $\vec{v} \in \{0,1\}^n$, we define $L(\vec{v})$ as follows: $L(\vec{v}) = b_j$ where $1 \leq j \leq l$ is the least value such that $\vec{v}$ satisfies the monomial T_j; if there is no such j then $L(\vec{v}) = 0$. We denote the class of all such representations by DL_n. For any constant k, if each monomial T_i has at most k literals, then we have a *k-decision list*, and we denote the class of all such representations by $k\text{DL}_n$.

3 Absolute limits on learning with errors

In this section we prove theorems bounding the achievable error rate for both the malicious error and classification noise models. These bounds are absolute in the sense that they apply to *any* learning algorithm, regardless of its computational complexity, the number of examples it uses, the hypothesis space it uses, and so on. Our first such result states that the malicious error rate must be smaller than the desired accuracy ϵ. This is in sharp contrast to the classification noise model, where Angluin and Laird [1] proved, for example, $E_{CN}^{poly}(k\text{DNF}_n) \geq c_0$ for all n and any constant $c_0 < 1/2$.

Let us call a representation class C *distinct* if there exist representations $c_1, c_2 \in C$ and points $u, v, w, x \in X$ satisfying $u \in pos(c_1), u \in neg(c_2), v \in pos(c_1), v \in pos(c_2), w \in neg(c_1), w \in pos(c_2)$, and $x \in neg(c_1), x \in neg(c_2)$.

Theorem 1 *Let C be a distinct representation class. Then*

$$E_{MAL}(C) < \frac{\epsilon}{1+\epsilon}.$$

Proof: We use a technique that we will call the *method of induced distributions*: we choose $l \geq 2$ representations $\{c_i\}_{i\in\{1,\ldots,l\}} \subseteq C$, along with l pairs of target distributions $\{D^+_{c_i}\}_{i\in\{1,\ldots,l\}}$ and $\{D^-_{c_i}\}_{i\in\{1,\ldots,l\}}$. These representations and target distributions are such that for any $i \neq j$, $1 \leq i, j \leq l$, c_j is ϵ-bad with respect to the distributions $D^+_{c_i}, D^-_{c_i}$. Then adversaries $\{ADV_{c_i}\}_{i\in\{1,\ldots,l\}}$ are constructed for generating any errors when c_i is the target representation such that the behavior of the oracle POS^{β}_{MAL} is identical regardless of which c_i is the target representation; the same is true for the oracle NEG^{β}_{MAL}, thus making it impossible for any learning algorithm to distinguish the true target representation, and essentially forcing the algorithm to "guess" one of the c_i.

In the case of Theorem 1, this technique is easily applied, with $l = 2$, as follows: let $c_1, c_2 \in C$ and $u, v, w, x \in X$ be as in the definition of distinct. Define the following target distributions for c_1:

$$\begin{aligned} D^+_{c_1}(u) &= \epsilon \\ D^+_{c_1}(v) &= 1-\epsilon \end{aligned}$$

and

$$\begin{aligned} D^-_{c_1}(w) &= \epsilon \\ D^-_{c_1}(x) &= 1-\epsilon. \end{aligned}$$

For c_2, the target distributions are:

$$\begin{aligned} D^+_{c_2}(v) &= 1-\epsilon \\ D^+_{c_2}(w) &= \epsilon \end{aligned}$$

and

$$\begin{aligned} D^-_{c_2}(u) &= \epsilon \\ D^-_{c_2}(x) &= 1-\epsilon. \end{aligned}$$

Note that these distributions are such that any representation that disagrees with the target representation on one of the points u, v, w, x is ϵ-bad with respect to the target distributions. Now if c_1 is the target representation, then the adversary ADV_{c_1} behaves as follows: on calls to POS^{β}_{MAL}, ADV_{c_1} always returns the point w whenever an error occurs; on calls to NEG^{β}_{MAL}, ADV_{c_1} always returns the point u whenever an error occurs. Under these definitions, the oracle POS^{β}_{MAL} draws a point from an *induced* distribution $I^{+}_{c_1}$ that is determined by the joint behavior of the distribution $D^{+}_{c_1}$ and the adversary ADV_{c_1}, and is given by

$$\begin{aligned} I^{+}_{c_1}(u) &= (1-\beta)\epsilon \\ I^{+}_{c_1}(v) &= (1-\beta)(1-\epsilon) \\ I^{+}_{c_1}(w) &= \beta \end{aligned}$$

where β is the malicious error rate. Similarly, the oracle NEG^{β}_{MAL} draws from an induced distribution $I^{-}_{c_1}$:

$$\begin{aligned} I^{-}_{c_1}(u) &= \beta \\ I^{-}_{c_1}(w) &= (1-\beta)\epsilon \\ I^{-}_{c_1}(x) &= (1-\beta)(1-\epsilon). \end{aligned}$$

For target representation c_2, the adversary ADV_{c_2} always returns the point u whenever a call to POS^{β}_{MAL} results in an error, and always returns the point w whenever a call to NEG^{β}_{MAL} results in an error. Then the oracle POS^{β}_{MAL} draws from the induced distribution

$$\begin{aligned} I^{+}_{c_2}(u) &= \beta \\ I^{+}_{c_2}(v) &= (1-\beta)(1-\epsilon) \\ I^{+}_{c_2}(w) &= (1-\beta)\epsilon \end{aligned}$$

and the oracle NEG^{β}_{MAL} from the induced distribution

$$\begin{aligned} I^{-}_{c_2}(u) &= (1-\beta)\epsilon \\ I^{-}_{c_2}(w) &= \beta \\ I^{-}_{c_2}(x) &= (1-\beta)(1-\epsilon). \end{aligned}$$

It is easily verified that if $\beta = \epsilon/(1+\epsilon)$, then the distributions $I^{+}_{c_1}$ and $I^{+}_{c_2}$ are identical, and that $I^{-}_{c_1}$ and $I^{-}_{c_2}$ are identical; if $\beta > \epsilon/(1+\epsilon)$, the adversary may always choose to flip a biased coin, and be "honest" (i.e., draw from the correct target distribution) when the outcome is heads, thus reducing the effective error rate to exactly $\epsilon/(1+\epsilon)$. Thus, under these distributions and adversaries, the behavior of the oracles POS^{β}_{MAL} and NEG^{β}_{MAL} is identical regardless of the target representation. This implies that any algorithm that produces an ϵ-good hypothesis for target representation c_1 with probability at least $1-\delta$ under the distributions $D^{+}_{c_1}$ and $D^{-}_{c_1}$ must fail to output an ϵ-good hypothesis for target representation c_2 with probability at least $1-\delta$ under the distributions $D^{+}_{c_2}$ and $D^{-}_{c_2}$, thus proving the theorem. □

An intuitive interpretation of the result is that if we desire 90 percent accuracy from the hypothesis, there must be less than about 10 percent error.

We emphasize that Theorem 1 bounds the achievable malicious error rate for *any* learning algorithm, regardless of computational complexity, sample complexity or the hypothesis class. Thus, for distinct C, we always have $E_{MAL}(C) \leq \epsilon/(1+\epsilon) = O(\epsilon)$. All of the representation classes studied here are distinct. We shall see in Theorem 7 of Section 4 that any hypothesis that nearly minimizes the number of disagreements with a large enough sample from POS^{β}_{MAL} and NEG^{β}_{MAL} is ϵ-good with high probability provided $\beta < \epsilon/4$. Thus, for the finite representation classes we study here (such as all the classes over the Boolean domain $\{0,1\}^n$), there is always a (possibly super-polynomial time) exhaustive search algorithm A achieving $E_{MAL}(C,A) = \Omega(\epsilon)$; combined with Theorem 1, this gives $E_{MAL}(C) = \Theta(\epsilon)$ for these classes. However, we will primarily be concerned with achieving the largest possible malicious error rate in polynomial time.

We now turn our attention to positive-only and negative-only learning in the presence of errors, where we will see that for many representation classes, the absolute bounds on the achievable error rate are even stronger than those given by Theorem 1.

Let C be a representation class. We will call C *positive t-splittable* if there exist representations $c_1, \ldots, c_t \in C$ and points $u_1, \ldots, u_t \in X$ and $v \in X$ satisfying all of the following conditions:

$$\begin{aligned} u_i &\in pos(c_j), i \neq j, 1 \leq i, j \leq t \\ u_j &\in neg(c_j), 1 \leq j \leq t \\ v &\in pos(c_i), 1 \leq i \leq t. \end{aligned}$$

Similarly, C is *negative t-splittable* if we have

$$\begin{aligned} u_i &\in neg(c_j), i \neq j, 1 \leq i, j \leq t \\ u_j &\in pos(c_j), 1 \leq j \leq t \\ v &\in neg(c_i), 1 \leq i \leq t. \end{aligned}$$

Note that if $\textsc{vcd}(C) = d$, then C is both positive and negative d-splittable. The converse does not necessarily hold.

Theorem 2 *Let C be positive t-splittable (respectively, negative t-splittable). Then for $\delta \leq 1/t$,*

$$E_{MAL,+}(C) < \frac{\epsilon}{t-1}$$

(respectively, $E_{MAL,-}(C) < \frac{\epsilon}{t-1}$).

Proof: The proof is by the method of induced distributions. We prove only the case that C is positive t-splittable; the proof for C negative t-splittable is similar. Let $c_1, \ldots, c_t \in C$ and $u_1, \ldots, u_t, v \in X$ be as in the definition of positive t-splittable. For target representation c_j, define the target distributions $D^+_{c_j}$ over $pos(c_j)$ and $D^-_{c_j}$ over $neg(c_j)$ as follows:

$$\begin{aligned} D^+_{c_j}(u_i) &= \frac{\epsilon}{t-1}, 1 \leq i \leq t, i \neq j \\ D^+_{c_j}(v) &= 1 - \epsilon \end{aligned}$$

and

$$D^-_{c_j}(u_j) = 1.$$

For target representation c_j, the errors on calls to POS^{β}_{MAL} are generated by an adversary ADV_{c_j} who always returns the point u_j whenever an error occurs. Then under these definitions, POS^{β}_{MAL} draws a point from a distribution $I^+_{c_j}$ induced by the distribution $D^+_{c_j}$ and the adversary ADV_{c_j}. This distribution is

$$\begin{aligned} I^+_{c_j}(u_i) &= (1-\beta)\frac{\epsilon}{t-1}, 1 \leq i \leq t, i \neq j \\ I^+_{c_j}(v) &= (1-\beta)(1-\epsilon) \\ I^+_{c_j}(u_j) &= \beta. \end{aligned}$$

If $\beta = (1-\beta)(\epsilon/(t-1))$, then the induced distributions $I^+_{c_j}$ are all identical for $1 \leq j \leq t$. Solving, we obtain $\beta = (\epsilon/(t-1))/(1+\epsilon/(t-1)) < \epsilon/(t-1)$. Now let $\beta \geq \epsilon/(t-1)$, and assume A is a β-tolerant positive-only learning algorithm for C. If c_j is the target representation, then with probability at least $1-\delta$, $u_i \in pos(h_A)$ for some $i \neq j$, otherwise $e^+(h_A) \geq \epsilon$ under the induced distribution $I^+_{c_j}$. Let k be such that

$$\mathbf{Pr}[u_k \in pos(h_A)] = \max_{1 \leq i \leq t}\{\mathbf{Pr}[u_i \in pos(h_A)]\}$$

where the probability is taken over all sequences of examples given to A by the oracle POS^{β}_{MAL} and the coin tosses of A. Then we must have

$$\mathbf{Pr}[u_k \in pos(h_A)] \geq \frac{1-\delta}{t-1}.$$

Choose $\delta < 1/t$. Then with probability at least δ, $e^-(h_A) = 1$ when c_k is the target representation, with distributions $D^+_{c_k}$ and $D^-_{c_k}$ and adversary ADV_{c_k}. This contradicts the assumption that A is a β-tolerant learning algorithm, and the theorem follows. □

Note that the restriction $\delta < 1/t$ in the proof of Theorem 2 is apparently necessary, since a learning algorithm may always randomly choose a u_j to be a positive example, and make all other u_i negative examples; the probability of failing to learn under the given distributions is then only $1/t$. It would be interesting to find a different proof that removed this restriction, or to prove that it is required.

As in the case of Theorem 1, Theorem 2 is an upper bound on the achievable malicious error rate for *all* learning algorithms, regardless of hypothesis representation, number of examples used or computation time. For any representation class C, by computing a value t such that C is t-splittable, we can obtain upper bounds on the positive-only and negative-only error rates for that class. As examples, we state such results as corollaries for a few of the representation classes studied here. Even in cases where the representation class is known to be not learnable from only positive or only negative examples in polynomial time (for example, it is shown in Kearns et al. [13] that monomials are not polynomially learnable from negative examples), the bounds on $E_{MAL,+}$ and $E_{MAL,-}$ are relevant since they also hold for algorithms that do not run in polynomial time.

Corollary 3 *Let M_n be the class of monomials over $x_1, \ldots, x_n$. Then*

$$E_{MAL,+}(M_n) < \frac{\epsilon}{n-1}$$

and

$$E_{MAL,-}(M_n) < \frac{\epsilon}{n-1}.$$

Corollary 4 *For fixed k, let $k\mathrm{DNF}_n$ be the class of kDNF formulae over $x_1, \ldots, x_n$. Then*

$$E_{MAL,+}(k\mathrm{DNF}_n) = O\left(\frac{\epsilon}{n^k}\right)$$

and

$$E_{MAL,-}(k\mathrm{DNF}_n) = O\left(\frac{\epsilon}{n^k}\right).$$

Corollary 5 *Let SF_n be the class of symmetric functions over $x_1, \ldots, x_n$. Then*

$$E_{MAL,+}(\mathrm{SF}_n) < \frac{\epsilon}{n-1}$$

and

$$E_{MAL,-}(\mathrm{SF}_n) < \frac{\epsilon}{n-1}.$$

Proofs of these corollaries follow from the Vapnik-Chervonenkis dimension of the representation classes and Theorem 2. Note that the proof of Theorem 2 shows that these corollaries actually hold for any fixed ϵ and n.

We note that Theorem 2 and its corollaries also hold for the classification noise model. To see this it suffices to notice that the adversaries ADV_{c_j} in the proof of Theorem 2 simulated the classification noise model. Thus, for classification noise we see that the power of using both positive and negative examples may be dramatic: for kCNF we have $E_{CN}^{poly}(k\mathrm{CNF}_n) \geq c_0$ for any $c_0 < 1/2$ due to Angluin and Laird [1] but $E_{CN,+}(k\mathrm{CNF}_n) = O(\epsilon/n^k)$ by Theorem 2. (Kearns et al. [13] show that kCNF is not learnable in polynomial time from negative examples even in the error-free model.) In fact, we can give a bound on $E_{CN,+}$ and $E_{CN,-}$ that is weaker but more general, and applies to almost any representation class. Note that by exhaustive search techniques, we have that for any small constant α, $E_{CN}(C) \geq 1/2 - \alpha$ for any finite representation class C. Thus the following result demonstrates that for representation classes over finite domains in the classification noise model, the advantage of using both positive and negative examples is almost always significant.

We will call a representation class C *positive* (respectively, *negative*) *incomparable* if there are representations $c_1, c_2 \in C$ and points $u, v, w \in X$ satisfying $u \in pos(c_1), u \in neg(c_2), v \in pos(c_1), v \in pos(c_2)$ (respectively, $v \in neg(c_1), v \in neg(c_2)$), $w \in neg(c_1), w \in pos(c_2)$.

Theorem 6 *Let C be positive (respectively, negative) incomparable. Then*

$$E_{CN,+}(C) < \frac{\epsilon}{1+\epsilon}$$

(respectively, $E_{CN,-}(C) < \frac{\epsilon}{1+\epsilon}$).

Proof: By the method of induced distributions. We do the proof for the case that C is positive incomparable; the proof when C is negative incomparable is similar. Let $c_1, c_2 \in C$ and $u, v, w \in X$ be as in the definition of positive incomparable. For target representation c_1, we define distributions

$$\begin{aligned} D^+_{c_1}(u) &= \epsilon \\ D^+_{c_1}(v) &= 1-\epsilon \end{aligned}$$

and

$$D^-_{c_1}(w) = 1.$$

Then in the classification noise model, the oracle POS^{β}_{CN} draws from the induced distribution

$$\begin{aligned} I^+_{c_1}(u) &= (1-\beta)\epsilon \\ I^+_{c_1}(v) &= (1-\beta)(1-\epsilon) \\ I^+_{c_1}(w) &= \beta. \end{aligned}$$

For target representation c_2, define distributions

$$\begin{aligned} D^+_{c_2}(v) &= 1-\epsilon \\ D^+_{c_2}(w) &= \epsilon \end{aligned}$$

and

$$D^-_{c_2}(u) = 1.$$

Then for target representation c_2, oracle POS^{β}_{CN} draws from the induced distribution

$$\begin{aligned} I^+_{c_2}(u) &= \beta \\ I^+_{c_2}(v) &= (1-\beta)(1-\epsilon) \\ I^+_{c_2}(w) &= (1-\beta)\epsilon. \end{aligned}$$

For $\beta = \epsilon/(1+\epsilon)$, distributions $I^+_{c_1}$ and $I^+_{c_2}$ are identical. Any positive-only algorithm learning c_1 under $D^+_{c_1}$ and $D^-_{c_1}$ with probability at least $1-\delta$ must fail with probability at least $1-\delta$ when learning c_2 under $D^+_{c_2}$ and $D^-_{c_2}$. □

Thus, for positive (respectively, negative) incomparable C, $E_{CN,+}(C) = O(\epsilon)$ (respectively, $E_{CN,-}(C) = O(\epsilon)$). All of the representation classes studied here are both positive and negative incomparable. Note that the proof of Theorem 6 depends upon the assumption that a learning algorithm has only an upper bound on the noise rate, not the exact value; thus, the *effective* noise rate may be less than the given upper bound. This issue does not arise in the malicious model, where the adversary may always choose to draw from the correct target distribution with some fixed probability, thus reducing the effective error rate to any value less than or equal to the given upper bound.

4 Efficient error-tolerant learning

Given the absolute upper bounds on the achievable malicious error rate of Section 3, we now wish to find *efficient* algorithms tolerating a rate that comes as close as possible to these bounds, or give evidence for the computational difficulty of approaching the optimal error rate. In this section we give efficient algorithms for several representation classes and analyze their tolerance to malicious errors.

We begin by giving a generalization of Occam's Razor [4] for the case when errors are present in the examples.

Let C and H be representation classes over X. Let A be an algorithm accessing POS^{β}_{MAL} and NEG^{β}_{MAL}, and taking inputs $0 < \epsilon, \delta < 1$. Suppose that for target representation $c \in C$ and $0 \leq \beta < \epsilon/4$, A makes m calls to POS^{β}_{MAL} and receives points $u_1, \ldots, u_m \in X$, and m calls to NEG^{β}_{MAL} and receives points $v_1, \ldots, v_m \in X$, and outputs $h_A \in H$ satisfying with probability at least $1 - \delta$:

$$|\{u_i : u_i \in neg(h_A)\}| \leq \frac{\epsilon}{2}m \quad (1)$$

$$|\{v_i : v_i \in pos(h_A)\}| \leq \frac{\epsilon}{2}m. \quad (2)$$

Thus, with high probability, h_A is consistent with at least a fraction $1 - \epsilon/2$ of the sample received from the faulty oracles POS^{β}_{MAL} and NEG^{β}_{MAL}. We will call such an A a *β-tolerant Occam algorithm for C by H*.

Theorem 7 *Let $\beta < \epsilon/4$, and let A be a β-tolerant Occam algorithm for C by H. Then A is a β-tolerant learning algorithm for C by H; the sample size required is $m = O(1/\epsilon \ln 1/\delta + 1/\epsilon \ln |H|)$. If A is such that only Condition 1 (respectively, Condition 2) above holds, then $e^+(h_A) < \epsilon$ (respectively, $e^-(h_A) < \epsilon$) with probability at least $1 - \delta$.*

Proof: We prove the statement where A meets Condition 1; the case for Condition 2 is similar. Let $h \in H$ be such that $e^+(h) \geq \epsilon$. Then the probability that h agrees with a point received from the oracle POS^{β}_{MAL} is bounded above by

$$(1 - \beta)(1 - \epsilon) + \beta \leq 1 - \frac{3\epsilon}{4}$$

for $\beta < \epsilon/4$. Thus the probability that h agrees with at least a fraction $1 - \epsilon/2$ of m examples received from POS^{β}_{MAL} is

$$LE\left(\frac{3\epsilon}{4}, m, \frac{\epsilon}{2}m\right) \leq e^{-m\epsilon/24}$$

by Fact CB1. From this it follows that the probability that *some* $h \in H$ with $e^+(h) \geq \epsilon$ agrees with a fraction $1 - \epsilon/2$ of the m examples is at most $|H|e^{-m\epsilon/24}$. Solving $|H|e^{-m\epsilon/24} \leq \delta/2$, we obtain $m \geq 24/\epsilon(\ln |H| + \ln 2/\delta)$. This proves that any h meeting Condition 1 is with high probability ϵ-good with respect to D^+, completing the proof. □

To demonstrate that the suggested approach of finding a nearly consistent hypothesis is in fact a feasible one, we note that if c is the target representation, then the probability that c fails to agree with at least a fraction $1 - \epsilon/2$ of m examples received from POS^{β}_{MAL} is

$$GE\left(\frac{\epsilon}{4}, m, \frac{\epsilon}{2}m\right) \leq \frac{\delta}{2}$$

for $\beta \leq \epsilon/4$ and m as in the statment of Theorem 7 by Fact CB2.

Thus, in the presence of errors of any kind, finding an $\epsilon/2$-good hypothesis is as good as learning, provided that $\beta < \epsilon/4$. This fact can be used to prove the correctness of the learning algorithms of the following two theorems due to Valiant.

Theorem 8 *(Valiant [24]) Let M_n be the class of monomials over $x_1, \ldots, x_n$. Then*

$$E^{poly}_{MAL,+}(M_n) = \Omega\left(\frac{\epsilon}{n}\right).$$

Theorem 9 *(Valiant [24]) For fixed k, let $k\mathrm{DNF}_n$ be the class of kDNF formulae over $x_1, \ldots, x_n$. Then*

$$E^{poly}_{MAL,-}(k\mathrm{DNF}_n) = \Omega\left(\frac{\epsilon}{n^k}\right).$$

Similar results are obtained by duality for the class of disjunctions (learnable from negative examples) and kCNF (learnable from positive examples); that is, $E^{poly}_{MAL,-}(1\mathrm{DNF}_n) = \Omega(\epsilon/n)$ and $E^{poly}_{MAL,+}(k\mathrm{CNF}_n) = \Omega(\epsilon/n^k)$. Note that the class of monomials (respectively, kDNF) is not polynomially learnable even in the error-free case from negative (respectively, positive) examples [13]

Combining Corollaries 8 and 9 with Corollaries 3 and 4 we have $E^{poly}_{MAL,+}(M_n) = \Theta(\epsilon/n)$ and $E^{poly}_{MAL,-}(k\mathrm{DNF}_n) = \Theta(\epsilon/n^k)$, thus proving that the algorithms of Valiant [24] tolerate the optimal malicious error rate with respect to positive-only and negative-only learning. The algorithm given in the following theorem, similar to those of Valiant [24], proves an analogous result for efficiently learning symmetric functions from only one type of examples in the presence of errors.

Theorem 10 *Let SF_n be the class of symmetric functions over $x_1, \ldots, x_n$. Then*

$$E^{poly}_{MAL,+}(\mathrm{SF}_n) = \Omega\left(\frac{\epsilon}{n}\right).$$

Proof: Let $\beta \leq \epsilon/8n$. The positive-only algorithm A maintains an integer array P indexed $0, \ldots, n$ and initialized to contain 0 at each location. A takes m (calculated below) examples from POS^{β}_{MAL}, and for each vector $\vec{v}$ received, increments $P[index(\vec{v})]$, where $index(\vec{v})$ is the number of bits set to 1 in $\vec{v}$. The hypothesis h_A is defined as follows: all vectors of index i are contained in $pos(h_A)$ if and only if $P[i] \geq (\epsilon/4n)m$; otherwise all vectors of index i are negative examples of h_A.

Note that h_A can disagree with at most a fraction $(\epsilon/4n)(n+1) < \epsilon/2$ of the m vectors received from POS^{β}_{MAL}, so $e^{+}(h_A) < \epsilon$ with high probability by Theorem 7. To prove that $e^{-}(h_A)$ with high probability, suppose that all vectors of index i are negative examples of the target representation (call such an i a *negative index*). Then the probability that a vector of index i is received on a call to POS^{β}_{MAL} is at most $\beta \leq \epsilon/8n$, since this occurs only when there is an error on a call to POS^{β}_{MAL}. Thus the probability of receiving $(\epsilon/4n)m$ vectors of index i in m calls to POS^{β}_{MAL} is

$$GE\left(\frac{\epsilon}{8n}, m, \frac{\epsilon}{4n}m\right) \leq e^{-m\epsilon/24n}$$

by Fact CB2. The probability that *some* negative index is classified as a positive index by h_A is thus at most

$$(n+1)e^{-m\epsilon/24n} \leq \frac{\delta}{2}$$

for $m = O((n/\epsilon)(\ln n + \ln 1/\delta))$. Thus with high probability, $e^{-}(h_A) = 0$, completing the proof. □

Thus, with Corollary 5 we have $E^{poly}_{MAL,+}(\mathrm{SF}_n) = \Theta(\epsilon/n)$. We can give a dual of the above algorithm to prove $E^{poly}_{MAL,-}(\mathrm{SF}_n) = \Theta(\epsilon/n)$ as well. The number of examples required by the

algorithm of Theorem 10 is a factor of n larger than the lower bound given by Ehrenfeucht et al. [8] for the error-free case; whether this increase is necessary for positive-only algorithms in the presence of malicious errors is an open problem.

The next theorem demonstrates that using both positive and negative examples can significantly increase the tolerated error rate in the malicious model.

Theorem 11 *Let* SF_n *be the class of symmetric functions over* $x_1, \ldots, x_n$. *Then*

$$E_{MAL}^{poly}(\mathrm{SF}_n) = \Omega(\epsilon).$$

Proof: Algorithm A maintains two integer arrays P and N, each indexed $0, \ldots, n$ and initialized to contain 0 at each location. A first takes m (calculated below) examples from POS_{MAL}^{β} and for each vector $\vec{v}$ received, increments $P[index(\vec{v})]$, where $index(\vec{v})$ is the number of bits set to 1 in $\vec{v}$. A then takes m examples from NEG_{MAL}^{β} and increments $N[index(\vec{v})]$ for each vector $\vec{v}$ received. The hypothesis h_A is computed as follows: all vectors of index i are contained in $pos(h_A)$ if and only if $P[i] \geq N[i]$; otherwise, all vectors of index i are contained in $neg(h_A)$.

We now show that for sufficiently large m, A is an $\epsilon/8$-tolerant Occam algorithm. For $0 \leq i \leq n$, let $d_i = \min(P[i], N[i])$. Then $d = \sum_{i=0}^{n} d_i$ is the number of vectors in the sample of size $2m$ with which h_A disagrees. Now for each i, either $P[i]$ or $N[i]$ is a lower bound on the number e_i of malicious errors received that have index i; let $e = \sum_{i=0}^{n} e_i$. Note that $e \geq d$. Now the probability that e exceeds $(\epsilon/4)(2m)$ in m calls POS_{MAL}^{β} and m calls to NEG_{MAL}^{β} for $\beta \leq \epsilon/8$ is

$$GE(\frac{\epsilon}{8}, 2m, \frac{\epsilon}{4}2m) \leq \delta$$

for $m = O(1/\epsilon \ln 1/\delta)$ by Fact CB2. Thus, with high probability the number of disagreements d of h_A on the examples received is less than $(\epsilon/2)m$. This shows that A is an $\epsilon/8$-tolerant Occam algorithm for SF, and thus is a learning algorithm for SF by Theorem 7 for $m = O(1/\epsilon \ln 1/\delta + n/\epsilon)$. □

Thus, by Theorems 1 and 11 we have $E_{MAL}^{poly}(\mathrm{SF}_n) = \Theta(\epsilon)$ in contrast with $E_{MAL,+}^{poly}(\mathrm{SF}_n) = \Theta(\epsilon/n)$ and $E_{MAL,-}^{poly}(\mathrm{SF}_n) = \Theta(\epsilon/n)$, a provable increase by using both types of examples. This is also our first example of a nontrivial class for which the optimal error rate $\Theta(\epsilon)$ of Theorem 1 can be achieved by an efficient algorithm. Furthermore, the sample complexity of algorithm A above meets the lower bound (within a constant factor) for the error-free case given by Ehrenfeucht et al. [8]; thus we have an algorithm with optimal sample complexity that tolerates the largest possible malicious error rate. This also demonstrates that it may be difficult to prove general theorems providing hard trade-offs between sample size and error rate.

We note that the proof of Theorem 11 relies only on the fact that there is a small number of equivalence classes of $\{0,1\}^n$ (namely, the sets of vectors with an equal number of bits set to 1) on which each symmetric function is constant. The same result thus holds for any Boolean representation class with this property.

Now that we have given some simple and efficient error-tolerant algorithms, we turn to the more abstract issue of general-purpose methods of making algorithms more tolerant to errors. It is reasonable to ask whether for an arbitrary representation class C, polynomial learnability of C implies polynomial learnability of C with malicious error rate β, for some nontrivial value of β that depends on C, ϵ and δ. The next theorem answers this in the affirmative by giving an efficient technique for converting any learning algorithm into an error-tolerant learning algorithm.

Theorem 12 *Let A be a polynomial-time learning algorithm for C with sample complexity $S_A(\epsilon, \delta)$, and let $s = S_A(\epsilon/8, 1/2)$. Then for $\epsilon \leq 1/2$,*

$$E_{MAL}^{poly}(C) = \Omega\left(\frac{\ln s}{s}\right).$$

Proof: We describe a polynomial-time algorithm A' that tolerates the desired error rate and uses A as a subroutine. Note that S_A (and hence, s) may also depend upon n in the case of parameterized C.

Algorithm A' will run algorithm A many times with accuracy parameter $\epsilon/8$ and confidence parameter $1/2$. The probability that no errors occur during a single such run is $(1-\beta)^s$. For $\beta \leq \ln s/s$ we have

$$(1-\beta)^s \geq \left(1 - \frac{\ln s}{s}\right)^s \geq \frac{1}{s^2}.$$

(This lower bound can be improved to $1/s^\alpha$ for any constant $\alpha > 1$ provided there is a sufficiently small constant upper bound on ϵ.) Thus, on a single run of A there is probability at least $(1-\delta)1/s^2 = 1/2s^2$ that no errors occur and A outputs an $\epsilon/8$-good hypothesis h_A (call a run of A when this occurs a *successful* run). A' will run A r times. In r runs of A, the probability that no successful run of A occurs is at most

$$\left(1 - \frac{1}{2s^2}\right)^r < \frac{\delta}{3}$$

for $r > 2s^2 \ln 3/\delta$. Let $h_A^1, \ldots, h_A^r$ be the hypotheses output by A on these r runs. Suppose h_A^i is an ϵ-bad hypothesis with respect to the target distributions; without loss of generality, suppose $e^+(h_A^i) \geq \epsilon$. Then the probability that h_A^i agrees with an example returned by the oracle POS_{MAL}^β is then at most $(1-\beta)(1-\epsilon) + \beta \leq 1 - 3\epsilon/4$ for $\beta \leq \epsilon/8$. Thus, the probability that h_A^i agrees with at least a fraction $1 - \epsilon/2$ of m examples returned by POS_{MAL}^β is

$$LE(\frac{3\epsilon}{4}, m, \frac{\epsilon}{2}m) \leq e^{-m\epsilon/24}$$

by Fact CB1. Then it follows that the probability that *some* h_A^i with $e^+(h_A^i) \geq \epsilon$ agrees with a fraction $1 - \epsilon/2$ of the m examples returned by POS_{MAL}^β is at most

$$re^{-m\epsilon/24} < \frac{\delta}{3}$$

for $m = O(1/\epsilon \ln r/\delta)$. Using Fact CB2, it can be shown that for $\beta \leq \epsilon/8$ the probability of an $\epsilon/8$-good h_A^i failing to agree with at least a fraction $1 - \epsilon/2$ of the m examples is smaller than $\delta/3$.

Thus, if A is run r times and the resulting hypotheses are tested against m examples from both POS_{MAL}^β and NEG_{MAL}^β, then with probability at least $1 - \delta$ the hypothesis with the fewest disagreements is in fact an ϵ-good hypothesis. Note that if A runs in polynomial time, A' also runs in polynomial time. □

Note that the trick used in the proof of Theorem 12 to eliminate the dependence of the tolerated error rate on δ is general: we may always set $\delta = 1/2$ and run A repeatedly to get a good hypothesis with high probability (provided we are willing to sacrifice a possible increase in the number of examples used). This technique has also been noted in the error-free setting by Haussler et al. [10].

It is shown by Ehrenfeucht et al. [8] that any learning algorithm A for a representation class C must have sample complexity

$$S_A(\epsilon, \delta) = \Omega\left(\frac{1}{\epsilon}\left(\ln\frac{1}{\delta} + \mathrm{VCD}(C)\right)\right).$$

Suppose that a learning algorithm A achieves this optimal sample complexity. Then applying Theorem 12, we immediately obtain an algorithm for C that tolerates a malicious error rate of

$$\Omega\left(\frac{\epsilon}{\mathrm{VCD}(C)}\ln\frac{\mathrm{VCD}(C)}{\epsilon}\right).$$

This rate is also the best that can be obtained by applying Theorem 12. By applying this technique to the algorithm of Valiant [23] for the class of monomials in the error-free model, we obtain the following corollary:

Corollary 13 *Let M_n be the class of monomials over $x_1, \ldots, x_n$. Then*

$$E^{poly}_{MAL}(M_n) = \Omega\left(\frac{\epsilon}{n}\ln\frac{n}{\epsilon}\right).$$

This improves the malicious error rate tolerated by the polynomial-time algorithm of Valiant [24] in Theorem 8 by a logarithmic factor. Furthermore, since $E^{poly}_{MAL,+}(M) = \Theta(\epsilon/n)$ this proves that, as in the case of symmetric functions, using both oracles improves the tolerable error rate. Similarly, a slight improvement over the malicious error rate given in Theorem 9 for kDNF can also be shown. For decision lists, we can apply the algorithm of Rivest [19] and the sample size bounds given by Ehrenfeucht et al. [8] to obtain the following:

Corollary 14 *Let $k\mathrm{DL}_n$ be the class of k-decision lists over $x_1, \ldots, x_n$. Then*

$$E^{poly}_{MAL}(k\mathrm{DL}_n) = \Omega\left(\frac{\epsilon}{n^k}\right).$$

Despite the small improvement in the tolerable error rate for monomials of Corollary 13, there is still a significant gap between the absolute upper bound of $\epsilon/(1+\epsilon)$ on the achievable malicious error rate for monomials implied by Theorem 1 and the $\Omega(\epsilon/n\ln n/\epsilon)$ polynomial-time error rate of Corollary 13. We now describe further improvements that allow the error rate to primarily depend only on the number of *relevant* variables. We describe an algorithm tolerating a larger error rate for the class M^s_n of monomials with at most s literals, where s may depend on n, the total number of variables. Our algorithm will tolerate a larger rate of error when the number s of relevant attributes is considerably smaller than the total number of variables n. Other improvements in the performance of learning algorithms in the presence of many irrelevant attributes are investigated by Littlestone [16] and Blum [3].

We note that by applying Theorem 2 we can show that even for M^1_n, the class of monomials of length 1, the positive-only and negative-only malicious error rates are bounded by $\epsilon/(n-1)$. This is again an absolute bound, holding regardless of the computational complexity of the learning algorithm. Thus, the positive-only algorithm of Valiant [24] in Theorem 8 cannot exhibit an improved error rate when restricted to the subclass M^s_n for *any* value of s.

Our error-tolerant learning algorithm for monomials is based on an approximation algorithm for a generalization of the set cover problem that we call the partial cover problem, which is defined below. This approximation algorithm is of independent interest and has found application in other learning algorithms [14, 26]. Our analysis and notation rely heavily on the work of Chvatal [7]; the reader may find it helpful to read his paper first.

The Partial Cover Problem:

Input: Finite sets $S_1, \ldots, S_n$ with positive real costs $c_1, \ldots, c_n$, and a positive fraction $0 < p \leq 1$. We assume without loss of generality that $\bigcup_{i=1}^{n} S_i = \{1, \ldots, m\} = T$ and we define $J = \{1, \ldots, n\}$.

Output: $J^* \subseteq J$ such that

$$|\bigcup_{j \in J^*} S_j| \geq pm$$

(we call such a J^* a *p-cover* of the S_i) and such that $cost_{PC}(J^*) = \sum_{j \in J^*} c_j$ is minimized.

Following Chvatal [7], for notational convenience we identify a partial cover $\{S_{j_1}, \ldots, S_{j_s}\}$ with the index set $\{j_1, \ldots, j_s\}$.

The partial cover problem is NP-hard, since it contains the set cover problem as a special case ($p = 1$) [9]. We now give a greedy approximation algorithm G for the partial cover problem.

Algorithm G:

Step 1. Initialize $J^* = \emptyset$.

Step 2. If $|\bigcup_{j \in J^*} S_j| \geq pm$ then halt and output J^*, since J^* is a p-cover.

Step 3. Set $q = pm - |\bigcup_{j \in J^*} S_j|$ (thus q is the number of still-uncovered elements that we must cover in order to have a p-cover). For each $j \notin J^*$, if $|S_j| > q$, delete any $|S_j| - q$ elements from S_j (delete excess elements from any remaining set that covers more than q elements).

Step 4. Find a k minimizing the ratio $c_k/|S_k|$. Add k to J^*, and replace each S_j by $S_j - S_k$. Return to Step 2.

Chvatal [7] shows that the greedy algorithm for the set cover problem cannot do better than $H(m)$ times the cost of an optimal cover, where $H(m) = \sum_{i=1}^{m} 1/i = \Theta(\log m)$. By a padding argument, this can also be shown to hold for algorithm G above, for any fixed p. We now prove that G can always achieve this approximation bound within a constant factor.

Theorem 15 *Let I be an instance of partial cover and let $opt_{PC}(I)$ denote the cost of an optimal p-cover for I. Then the cost of the p-cover J^* produced by algorithm G satisfies*

$$cost_{PC}(J^*) \leq (2H(m) + 3) opt_{PC}(I).$$

Proof: Let J_{opt} be an optimal p-cover (i.e., $cost_{PC}(J_{opt}) = opt_{PC}(I)$). Let

$$T_{opt} = \bigcup_{j \in J_{opt}} S_j$$

(these are the elements covered by J_{opt}) and

$$T^* = \bigcup_{j \in J^*} S_j$$

(these are the elements covered by J^*) where J^* is the p-cover output by algorithm G. Notice that $|T_{opt}| \geq pm$ since J_{opt} is a p-cover.

Let S_j^r be set of elements remaining in the set S_j immediately before Step 3 in algorithm G is executed for the rth time (i.e., at the start of the rth iteration of Steps 2-4). By appropriate renaming of the S_j, we may assume without loss of generality that $J^* = \{1, \ldots, r\}$ (recall that J^* is the set of *indices* of sets chosen by algorithm G) immediately after Step 4 is executed for the rth time (i.e., at the end of the rth iteration of Steps 2-4). Let $J^* = \{1, \ldots, t\}$ when G halts, so there are a total of t iterations.

Define $T^{**} = T^* - S'_t$, where S'_t is the union of all elements deleted from the set S_t on all executions of Step 3. Intuitively, T^{**} consists of those elements that algorithm G "credits" itself with having covered during its execution (as opposed to those elements regarded as "excess" that were covered because G may cover more than the required minimum fraction p). We say that a set S_j is *at capacity* when in Step 3, $|S_j| \geq q$. Note that once S_j reaches capacity, it remains at capacity until it is chosen in Step 4 or until G halts. This is because if l elements are removed from S_j on an execution of Step 4, the value of q in Step 3 will decrease by at least l on the next iteration. Furthermore, since G halts the first time a set at capacity is chosen, and by the above definitions S_t is the last set chosen by G, we have that $T^{**} = \bigcup_{r=1}^{t} S_r^r$. Thus we have $|S'_t| = |T^*| - pm$ and $|T^{**}| = pm$.

The set S_r^r can be regarded as the set of previously uncovered elements that are added to T^{**} on the rth iteration. We wish to amortize the cost c_r over the elements covered. For each $i \in T^*$, we define a number y_i, which is intuitively the cost we paid to put i in T^*:

$$y_i = \begin{cases} c_r/|S_r^r| & \text{if for some } r,\ i \in S_r^r \\ 0 & i \text{ is not in } T^{**} \end{cases}$$

Since for $i \in T^* - T^{**}$, $y_i = 0$, we have

$$\begin{aligned} \sum_{i \in T^{**}} y_i &= \sum_{i \in T^*} y_i \\ &= \sum_{r=1}^{t} \sum_{i \in S_r^r} y_i \\ &= \sum_{r=1}^{t} c_r \\ &= \sum_{j \in J^*} c_j \\ &= cost_{PC}(J^*). \end{aligned}$$

Thus to bound $cost_{PC}(J^*)$, we now bound $\sum_{i \in T^{**}} y_i$ in two parts, first bounding $\sum_{i \in T^{**} - T_{opt}} y_i$ and then bounding $\sum_{i \in T^{**} \cap T_{opt}} y_i$.

Lemma 16

$$\sum_{i \in T^{**} - T_{opt}} y_i \leq (H(m) + 2) opt_{PC}(I).$$

Proof: If $T^{**} \subseteq T_{opt}$ then the lemma follows trivially. We therefore assume $T^{**} \not\subseteq T_{opt}$. Since $|T_{opt}| \geq pm$ and $|T^{**}| = pm$, this implies $T_{opt} - T^{**} \neq \emptyset$. Pick $j \in J_{opt}$ such that

$$\frac{c_j}{|S_j - T^{**}|}$$

is minimized. Now

$$\begin{aligned} \frac{opt_{PC}(I)}{|T_{opt} - T^{**}|} &= \frac{\sum_{i \in J_{opt}} c_i}{|\bigcup_{i \in J_{opt}} (S_i - T^{**})|} \\ &\geq \frac{\sum_{i \in J_{opt}} c_i}{\sum_{i \in J_{opt}} |S_i - T^{**}|} \\ &\geq \frac{c_j}{|S_j - T^{**}|}. \end{aligned}$$

Thus

$$opt_{PC}(I) \geq |T_{opt} - T^{**}| \frac{c_j}{|S_j - T^{**}|}.$$

Let r_0 be the first execution of Step 3 in which $|S_j| > q$ (i.e., S_j reaches capacity on the r_0th iteration). We will analyze the behavior of G before and after the r_0th iteration separately. Let T_0^{**} denote the set of elements that were added to T^{**} prior to the r_0 iteration. For each $i \in T_0^{**} - T_{opt}$, the cost y_i must satisfy

$$y_i \leq \frac{c_j}{|S_j - T^{**}|}$$

because otherwise G would have already added S_j to J^*. Since $|T_{opt} - T^{**}| \geq |T^{**} - T_{opt}|$ we have

$$\begin{aligned} \sum_{i \in T_0^{**} - T_{opt}} y_i &\leq \sum_{i \in T_0^{**} - T_{opt}} \frac{c_j}{|S_j - T^{**}|} \\ &\leq |T_{opt} - T^{**}| \frac{c_j}{|S_j - T^{**}|} \\ &\leq opt_{PC}(I). \end{aligned}$$

For iterations $r \geq r_0$, whenever an element i is added to $T_1^{**} = T^{**} - T_0^{**}$, an element is deleted from S_j in Step 3, since S_j is at capacity. We charge y_i to this element as follows:

$$\begin{aligned} \sum_{i \in T_1^{**} - T_{opt}} y_i &\leq \sum_{i \in T_1^{**}} y_i \\ &= \sum_{r=r_0}^{t} \sum_{i \in S_r^r} y_i \\ &\leq \sum_{r=r_0}^{t-1} \sum_{i \in S_r^r} y_i + c_j \end{aligned}$$

(because on iteration t, both S_j and S_t are at capacity, so $c_t \leq c_j$)

$$\leq \sum_{r=r_0}^{t-1} \frac{c_r}{|S_r^r|} |S_j^r - S_j^{r+1}| + c_j$$

(because since S_j is at capacity, $|S_j^r - S_j^{r+1}| = |S_r^r|$)

$$\leq \sum_{r=r_0}^{t-1} \frac{c_j}{|S_j^r|} |S_j^r - S_j^{r+1}| + c_j$$

(because otherwise G would have chosen S_j at time r)

$$\begin{aligned} &= c_j \sum_{r=r_0}^{t-1} \frac{1}{|S_j^r|} |S_j^r - S_j^{r+1}| + c_j \\ &\leq c_j H(|S_j|) + c_j \\ &= c_j (H(|S_j|) + 1). \end{aligned}$$

Combining the two parts, we have

$$\begin{aligned} \sum_{i \in T^{**} - T_{opt}} y_i &= \sum_{i \in T_0^{**} - T_{opt}} y_i + \sum_{i \in T_1^{**} - T_{opt}} y_i \\ &\leq opt_{PC}(I) + c_j (H(m) + 1) \\ &\leq (H(m) + 2) opt_{PC}(I). \end{aligned}$$

□(Lemma 16)

Lemma 17

$$\sum_{i \in T^{**} \cap T_{opt}} y_i \leq (H(m) + 1) opt_{PC}(I).$$

Proof: We generalize the idea used by Chvatal [7]. For $j \in J_{opt}$ and $S_j \bigcap T^{**} \neq \emptyset$,

$$\begin{aligned} \sum_{i \in S_j \cap T^{**}} y_i &= \sum_{r=1}^{t} \sum_{i \in S_j \cap S_r^r} y_i \\ &\leq \sum_{r=1}^{t-1} \frac{c_r}{|S_r^r|} |S_j^r - S_j^{r+1}| + c_j \end{aligned}$$

(because the average cost of elements in S_t is lower than in S_j, and we are summing over at most $|S_j|$ elements)

$$\leq \sum_{r=1}^{s} \frac{c_j}{|S_j^r|} |S_j^r - S_j^{r+1}| + c_j$$

(where $s = \min\{\max\{k : S_j^k \neq \emptyset\}, t\}$)

$$\begin{aligned} &= c_j \sum_{r=1}^{s} \frac{1}{|S_j^r|} |S_j^r - S_j^{r+1}| + c_j \\ &\leq c_j H(|S_j|) + c_j \\ &= c_j (H(|S_j|) + 1). \end{aligned}$$

Now by the above,

$$\begin{aligned} \sum_{i \in T^{**} \cap T_{opt}} y_i &\leq \sum_{j \in J_{opt}} \sum_{i \in S_j \cap T^{**}} y_i \\ &\leq \sum_{j \in J_{opt}} (H(m)+1) c_j \\ &\leq (H(m)+1) opt_{PC}(I). \end{aligned}$$

□(Lemma 17)

Combining Lemmas 16 and 17, we have

$$\begin{aligned} \sum_{j \in J^*} c_j &= \sum_{i \in T^*} y_i \\ &= \sum_{i \in T^{**}} y_i \\ &= \sum_{i \in T^{**} - T_{opt}} y_i + \sum_{i \in T^{**} \cap T_{opt}} y_i \\ &\leq (H(m)+2) opt_{PC}(I) + (H(m)+1) opt_{PC}(I) \\ &= (2H(m)+3) opt_{PC}(I). \end{aligned}$$

This completes the proof of Theorem 15. □

We now use algorithm G as a subroutine in constructing our error-tolerant learning algorithm for M_n^s.

Theorem 18 *Let M_n^s be the class of monomials over $x_1, \ldots, x_n$ containing at most s literals. Then*

$$E_{MAL}^{poly}(M_n^s) = \Omega\left(\frac{\epsilon}{s \log \frac{s \log n}{\epsilon}}\right).$$

Proof: We construct an Occam algorithm A for M_n^s that tolerates the desired malicious error rate, and uses the algorithm G for the partial cover problem as a subroutine.

Let $0 \leq \beta < \epsilon/8$, and let $c \in M_n^s$ be the target monomial. A first takes m_N points from the oracle NEG_{MAL}^{β}, where $m_N = O(1/\epsilon \ln 1/\delta + 1/\epsilon \ln |M_n^s|)$ as in the statement of Theorem 7. Let S denote the multiset of points received by A from NEG_{MAL}^{β}. For $1 \leq i \leq n$, define the multisets

$$S_i^0 = \{\vec{v} \in S : v_i = 0\}$$

and

$$S_i^1 = \{\vec{v} \in S : v_i = 1\}.$$

We now define a pairing between monomials and partial covers of the set S as follows: the literal x_i is paired with the partial cover consisting of the single set S_i^0 and the literal $\overline{x}_i$ is paired with the partial cover consisting of the single set S_i^1. Then any monomial c is paired with the partial cover obtained by including exactly those S_i^0 and S_i^1 that are paired with the literals appearing in c. Note that the multiset $neg(c) \cap S$ contains exactly those vectors that are covered by the corresponding partial cover.

Now with high probability, there must be some collection of the S_i^0 and S_i^1 that together form a $1 - \epsilon/2$ cover of S: namely, if (without loss of generality) the target monomial $c \in M_n^s$ is

$$c = x_1 \cdots x_r \overline{x}_{r+1} \cdots \overline{x}_s$$

then with high probability the sets

$$S_1^0, \ldots, S_r^0, S_{r+1}^1, \ldots, S_s^1$$

form a $1 - \epsilon/2$ cover of S, since for $\beta \leq \epsilon/8$, the probability that the target monomial c disagrees with a fraction larger than $\epsilon/2$ of a sample of size m_N from NEG_{MAL}^{β} can be shown to be smaller than $\delta/2$ by Fact CB2.

Thus, A will input the sets $S_1^0, \ldots, S_n^0, S_1^1, \ldots, S_n^1$ and the value $p = 1 - \epsilon/2$ to algorithm G. The costs for these sets input to G are defined below. However, note that regardless of these costs, if h_G is the monomial paired with the p-cover output by G, then since $|neg(h_G) \cap S| \geq (1 - \epsilon/2)m_N$ (where $neg(h_G) \cap S$ is interpreted as a multiset), $e^-(h_G) < \epsilon$ with high probability by Theorem 7. We now show that for β as in the statement of the theorem, we can choose the costs input to G so as to force $e^+(h_G) < \epsilon$ as well.

For any monomial c, let $p(c)$ denote the probability that c disagrees with a vector returned by POS_{MAL}^{β} [1], and let $cost_{PC}(c)$ denote the cost of the partial cover that is paired with c. To determine the costs of the sets input to G, A next samples POS_{MAL}^{β} enough times (determined by application of Facts CB1 and CB2) to obtain an estimate for $p(x_i)$ and $p(\overline{x}_i)$ for $1 \leq i \leq n$ that is accurate within a multiplicative factor of 2 — that is, if $\hat{p}(x_i)$ is the estimate computed by A, then $p(x_i)/2 \leq \hat{p}(x_i) \leq 2p(x_i)$ with high probability for each i. The same bounds hold for the estimate $\hat{p}(\overline{x}_i)$. Then the cost for set S_i^0 input to G by A is $\hat{p}(x_i)$ and the cost for set S_i^1 is $\hat{p}(\overline{x}_i)$.

Note that for any monomial $c = x_1 \cdots x_r \overline{x}_{r+1} \cdots \overline{x}_s$, we have with high probability

$$\begin{aligned} p(c) &\leq p(x_1) + \cdots + p(x_r) + p(\overline{x}_{r+1}) + \cdots + p(\overline{x}_s) \\ &\leq 2\hat{p}(x_1) + \cdots + 2\hat{p}(x_r) + 2\hat{p}(\overline{x}_{r+1}) + \cdots + 2\hat{p}(\overline{x}_s) \\ &= 2cost_{PC}(c). \end{aligned}$$

By Theorem 18, the output h_G of G must satisfy

$$cost_{PC}(h_G) \leq (H(m_N) + 2)cost_{PC}(c_{opt}) \tag{3}$$

[1] Note that technically this probability may not be well-defined since the behavior of the oracle POS_{MAL}^{β} may depend on the entire history of the computation so far. If this is the case, however, we may use the following trick: rather than running the algorithm using POS_{MAL}^{β}, we instead take a sufficiently large number of examples l from POS_{MAL}^{β}, and then run the algorithm using a uniform distribution over these l examples (treated as a multiset, not a set). The algorithm may need to be run more than once in order to find an appropriate setting of the error parameter used; this technique is detailed and shown correct in Theorem 20. For the rest of the proof, therefore, we assume without loss of generality that $p(c)$ is well-defined.

where c_{opt} is the monomial paired with a p-cover of minimum cost. But for the target monomial c we have

$$p(c) \leq \beta \tag{4}$$

$$2sp(c) \geq cost_{PC}(c) \tag{5}$$

where Equation 4 holds absolutely and Equation 5 holds with high probability, since c contains at most s literals.

From Equations 3, 4 and 5 we obtain with high probability

$$\begin{aligned} p(h_G) &\leq 2cost_{PC}(h_G) \\ &\leq 2(H(m_N)+2)cost_{PC}(c_{opt}) \\ &\leq 2(H(m_N)+2)cost_{PC}(c) \\ &\leq 4sp(c)(H(m_N)+2) \\ &\leq 4s\beta(H(m_N)+2). \end{aligned}$$

Thus, if we set

$$\beta = \frac{\epsilon}{4s(H(m_N)+2)} = \Omega(\frac{\epsilon}{s \log m_N})$$

then $e^+(h_G) < \epsilon$ with high probability by Theorem 7. We can remove the dependence of β on δ by method used in the proof of Theorem 12, thus obtaining an error rate of

$$\Omega\left(\frac{\epsilon}{s \log \frac{s \log n}{\epsilon}}\right)$$

completing the proof. □

As an example, if $s = \sqrt{n}$ then Theorem 18 gives

$$E^{poly}_{MAL}(M^{\sqrt{n}}_n) = \Omega(\frac{\epsilon}{\sqrt{n} \log \frac{n}{\epsilon}})$$

as opposed to the the bound of $\Omega(\epsilon/n \ln \epsilon/n)$ of Theorem 13.

Littlestone [16] shows that the Vapnik-Chervonenkis dimension of M^s_n is $\Theta(s \ln(1+n/s))$. Since the algorithm of Valiant [23] can be modified to have optimal sample complexity for M^s_n, by applying Theorem 12 to this modified algorithm we obtain

$$E^{poly}_{MAL}(M^s_n) = \Omega\left(\frac{\epsilon \ln(\frac{s}{\epsilon}\ln(1+\frac{n}{s}))}{s \ln(1+\frac{n}{s})}\right).$$

This lower bound on $E^{poly}_{MAL}(M^s_n)$ is incomparable to that of Theorem 18. We may decide at run time which algorithm will tolerate the larger error rate, thus giving

$$E^{poly}_{MAL}(M^s_n) = \Omega\left(\min\left(\frac{\epsilon \ln(\frac{s}{\epsilon}\ln(1+\frac{n}{s}))}{s \ln(1+\frac{n}{s})}, \frac{\epsilon}{s \log \frac{s \log n}{\epsilon}}\right)\right).$$

By using transformation techniques similar to those described Kearns et al. [13] it can be shown that the algorithm of Theorem 18 (as well as that obtained from Theorem 12) can be used to obtain an improvement in the error rate over the negative-only algorithm of Valiant [24] for the class $k\text{DNF}_{n,s}$ of kDNF formulae with at most s terms. Briefly, the appropriate transformation regards a kDNF formulae as a 1DNF formulae in a space of $\Theta(n^k)$ variables, one variable for each of the possible terms (monomials) of length at most k.

5 Limits on efficient learning with errors

In Section 3, we saw that there was an absolute bound of $\epsilon/(1+\epsilon)$ on the achievable malicious error rate for most interesting representation classes. It was also argued there that, at least for our finite representation classes over $\{0,1\}^n$, this bound could always be achieved by a super-polynomial time exhaustive search learning algorithm. Then in Section 4 we gave polynomial-time learning algorithms that in some cases achieved the optimal error rate $O(\epsilon)$, but in other cases fell short. These observations raise the natural question of whether for some classes it is possible to prove bounds stronger than $\epsilon/(1+\epsilon)$ on the malicious error rate for learning algorithms constrained to run in polynomial time. In particular, for parameterized representation classes, under what conditions must the error rate tolerated by a polynomial-time learning algorithm decrease as the number of variables n increases? If we informally regard the problem of learning with malicious errors as an optimization problem where the objective is to maximize the achievable error rate in polynomial time, and $\epsilon/(1+\epsilon)$ is the optimal value, then we might expect such hardness results to take the form of hardness results for the approximation of *NP*-hard optimization problems. This is the approach we pursue in this section.

By reducing standard combinatorial optimization problems to learning problems, we state theorems indicating that efficiently learning with an error rate approaching $\Theta(\epsilon)$ is eventually as hard as approximations for *NP*-hard problems.

In Section 4 we gave an error-tolerant algorithm for learning monomials by monomials that was based on an approximation algorithm for a generalization of set cover. Our next theorem gives a reduction in the opposite direction: an algorithm learning monomials by monomials and tolerating a malicious error rate approaching $\Theta(\epsilon)$ can be used to obtain an improved approximation algorithm for set cover.

Theorem 19 *Let M_n be the class of monomials over $x_1, \ldots, x_n$. Suppose there is a polynomial-time learning algorithm A for M_n using hypothesis space M_n such that*

$$E^{poly}_{MAL}(M_n, A) = \frac{\epsilon}{r(n)}.$$

Then there is a polynomial-time algorithm for the weighted set cover problem that outputs (with high probability) a cover whose cost is at most $2r(n)$ times the optimal cost, where n is the number of sets.

Proof: We describe an approximation algorithm A' for set cover that uses the learning algorithm A as a subroutine. Given an instance I of set cover with sets $S_1, \ldots, S_n$ and costs $c_1, \ldots, c_n$, let $J_{opt} \subseteq \{1, \ldots, n\}$ be an optimal cover of $T = \bigcup_{j=1}^{n} S_j = \{1, \ldots, m\}$, where we identify a cover $\{S_{j_1}, \ldots, S_{j_s}\}$ with its index set $\{j_1, \ldots, j_s\}$. Let $cost_{SC}(J)$ denote the set cover cost of any cover J of T, and let $opt_{SC}(I) = cost_{SC}(J_{opt})$. As in the proof of Theorem 18, we pair a cover $\{j_1, \ldots, j_s\}$ of T with the monomial $x_{j_1} \cdots x_{j_s}$ over the variables $x_1, \ldots, x_n$. Let c_{opt} be the monomial paired with the optimal cover J_{opt}.

The goal of A' is to simulate algorithm A with the intention that c_{opt} is the target monomial, and use the monomial h_A output by A to obtain the desired cover of T. The examples given to A on calls to NEG^{β}_{MAL} during this simulation will be constructed so as to guarantee that the collection of sets paired with h_A is actually a cover of T, while the examples given to A on calls to POS^{β}_{MAL} guarantee that this cover has a cost within a multiplicative factor of $2r(n)$ of the optimal cost.

We first describe the examples A' generates for A on calls to NEG^{β}_{MAL}. For each $i \in T$, let $\vec{u}_i \in \{0,1\}^n$ be the vector whose jth bit is 0 if and only if $i \in S_j$, and let the multiset U be $U = \bigcup_{i \in T}\{\vec{u}_i\}$. Then $\{j_1, \ldots, j_s\}$ is a cover of T if and only if $U \subseteq neg(x_{j_1} \cdots x_{j_s})$. In particular, we must have $U \subseteq neg(c_{opt})$. Thus, define the target distribution D^- for c_{opt} to be uniform over U. Note that this distribution can be generated in polynomial time by A'. On calls of A to NEG^{β}_{MAL}, A' will simply draw from D^-; thus if we regard c_{opt} as the target monomial, there are no errors in the negative examples. A' will simulate A with accuracy parameter $\epsilon \leq 1/|U|$, thus forcing A to output an hypothesis monomial h_A such that $U \subseteq neg(h_A)$; by the above argument, this implies that the collection of sets paired with the monomial h_A is a cover of T. Note that $|U|$ (and therefore $1/\epsilon$) may be super-polynomial in n, but it is polynomial in the size of the instance I.

We now describe the examples A' generates for A on calls to POS^{β}_{MAL}. Instead of defining the target distribution D^+ for c_{opt}, we define an *induced* distribution I^+ from which the oracle POS^{β}_{MAL} will draw. Thus, I^+ will describe the joint behavior of the underlying distribution D^+ on c_{opt} and an adversary generating the malicious errors. For each $1 \leq j \leq n$, let $\vec{v}_j \in \{0,1\}^n$ be the vector whose jth bit is 0, and all other bits are 1. Let $I^+(\vec{v}_j) = c_j$ for each j, where c_j is the cost of the set S_j, and we assume without loss of generality that $\sum_{j=1}^{n} c_j \leq \epsilon/r(n)$ (if not, we can normalize the weights without changing the relative costs of covers). We complete the definition of I^+ by letting $I^+((1,\ldots,1)) = 1 - \sum_{j=1}^{n} c_j$. Then the probability that a monomial $x_{i_1} \cdots x_{i_s}$ disagrees with a point drawn from POS^{β}_{MAL} is exactly $c_{i_1} + \cdots + c_{i_s}$, the cost of the corresponding cover. Thus since $opt_{SC}(I) \leq \sum_{j=1}^{n} c_j \leq \epsilon/r(n) = \beta$, I^+ is an induced distribution for c_{opt} with malicious error rate β. Note that I^+ can be generated by A' in polynomial time. When A requests an example from POS^{β}_{MAL}, A' will simply draw from I^+.

A' will run algorithm A many times with the oracles POS^{β}_{MAL} and NEG^{β}_{MAL} for c_{opt} described above, each time with a progressively smaller value for the accuracy parameter, starting with $\epsilon = 1/|U|$.

Now if $opt_{SC}(I) << \epsilon/r(n)$, then algorithm A may output a monomial h_A whose corresponding cover has a cost much larger than $4r(n) \cdot opt_{SC}(I)$, since h_A is only guaranteed to satisfy $\epsilon^+(h_A) < \epsilon$. We solve this problem by repeated scaling: A' first runs algorithm A with the oracles POS^{β}_{MAL} and NEG^{β}_{MAL} as they have been described. After each run, A' divides the accuracy parameter ϵ by 2, so that on some run $\epsilon/2r(n) \leq opt_{SC}(I) \leq \epsilon/r(n)$. On this run, we may regard I^+ as an induced distribution on the positive examples of c_{opt}, with malicious error rate at most $\beta = \epsilon/r(n) \leq 2opt_{SC}(I)$. Then the error $\epsilon^+(h_A)$ on the underlying distribution D^+ over $pos(c_{opt})$ is at most $\epsilon \leq 2r(n)opt_{SC}(I)$. The desired cover is thus the one paired with the monomial h_A. Note that without knowing c_{opt}, we have no way of knowing what the underlying target distribution D^+ is, but it is enough to know that I^+ is a "close" distribution. The only problem with the simulation described occurs when $opt_{SC}(I) << \sum_{j=1}^{n} c_j$, in which case it may take a super-polynomial number of runs of A to guarantee $\epsilon/2r(n) \leq opt_{SC}(I) \leq \epsilon/r(n)$. We solve this by preprocessing: before running the described simulation, A' runs the greedy approximation algorithm analyzed by Chvatal [7] on the set cover instance I, and removes any set whose cost is larger than the entire cost of the greedy cover. Then for the new (smaller) instance I', every cost is within a multiplicative factor of $\log m$ of every other cost. □

Thus, if $r(n) << \log n$, then Theorem 19 says that a polynomial time algorithm A for M_n (using hypothesis space M_n) tolerating $E^{poly}_{MAL}(M_n, A) \geq \epsilon/r(n)$ would imply a significant breakthrough in approximation algorithms for set cover, since the best algorithm for this problem remains the

greedy method analyzed by Chvatal and others [7, 11, 17, 18]. Note that the proof of Theorem 19 in fact shows the result holds for the class of *monotone* monomials.

Theorem 18 took an approximation algorithm for an optimization problem (the partial cover problem), and used it as a subroutine in obtaining an error-tolerant learning algorithm for M_n^s. Theorem 19 proved that when learning algorithms are restricted to hypothesis class M, any learning algorithm for M yields an algorithm for set cover with only a constant factor blowup in the approximation. Thus, we see that there are strong ties between learning with errors and approximating combinatorial optimization problems. Our goal now is to generalize and strengthen these ideas. We show that for any representation class C, the problem of learning C with errors is equivalent to a combinatorial optimization problem with only a constant factor blowup in the approximation in each direction of the reduction.

For domain X, define a *balanced sample* of X to be a sample

$$S = < x_1, 1 >, \ldots, < x_m, 1 >, < y_1, 0 >, \ldots, < y_m, 0 >$$

where $x_i, y_i \in X, 1 \leq i \leq m$. If C is a representation class over X and $c \in C$, define

$$cost_{MD}(c, S) = |\{< x_i, 1 > \in S : x_i \in neg(c)\}| + |\{< y_i, 0 > \in S : y_i \in pos(c)\}| + 1.$$

Thus, $cost_{MD}(c, S)$ is simply one more than the number of disagreements between the balanced sample S and the representation c. We now define the following optimization problem for C:
The Minimize Disagreements Problem for C (denoted $MD(C)$):

Input: Balanced sample S of X.

Output: Representation $c \in C$ such that $cost_{MD}(c, S)$ is minimized.

Theorem 20 *Let C be a representation class over X. If there exists a polynomial-time algorithm A' for $MD(C)$ that outputs $h_{A'} \in C$ such that $cost_{MD}(h_{A'}, S)$ is at most r times the optimal cost, then C is learnable by C by an algorithm A that runs in time polynomial in $1/\epsilon, 1/\delta$ and $\ln |C|$, and satisfies*

$$E_{MAL}^{poly}(C, A) \geq \frac{\epsilon}{8r}.$$

Conversely, if algorithm A learns C by C in polynomial time with error rate $E_{MAL}^{poly}(C, A) \geq \epsilon/r$, then there exists a polynomial-time algorithm A' for $MD(C)$ that outputs (with high probability) $h_{A'} \in C$ such that $cost_{MD}(h_{A'}, S)$ is at most $2r$ times the optimal cost.

Proof: Let S be a balanced sample of X, and let A' be an approximation algorithm for $MD(C)$ such that the output $h_{A'}$ satisfies $cost_{MD}(h_{A'}, S) \leq r \cdot opt_{MD}(S)$, where

$$opt_{MD} = \min_{h \in C}(cost_{MD}(h, S)).$$

Let $\beta = \epsilon/8r$. To learn C by C in polynomial time with error rate β, we take m random examples $x_1, \ldots, x_m$ from the oracle POS_{MAL}^{β} and m random examples $y_1, \ldots, y_m$ from the oracle NEG_{MAL}^{β}, where m is as in the statement of Theorem 7. Let S be the balanced sample consisting of the x_i and y_j. Now with probability at least $1 - \delta$, the target representation $c \in C$ disagrees with fewer than

$4\beta m$ elements of S by Fact CB2, so $opt_{MD}(S) \leq 4\beta m$ with high probability. Thus, algorithm A', when given S as input, will satisfy $cost_{MD}(h_{A'}, S) \leq r(4\beta m) = (\epsilon/2)m$. This implies that $h_{A'}$ can disagree with at most a fraction $\epsilon/2$ of the x_i and at most a fraction $\epsilon/2$ of the y_i. By Theorem 7, $h_{A'}$ is an ϵ-good hypothesis with high probability.

For the other direction, we use an algorithm A for learning C by C with $\beta = \epsilon/r$ to obtain an approximation algorithm for $MD(C)$ as follows: given the balanced sample S, let $h_{opt} \in C$ be such that $cost_{MD}(h_{opt}, S) = opt_{MD}(S)$ and assume without loss of generality that $m/r \geq opt_{MD}(S)$ (otherwise *any* hypothesis has cost at most $2r$ times the optimal). Define

$$c_0 = \max\{|\{x_i \in S : x_i \in neg(h_{opt})\}|, |\{y_i \in S : y_i \in pos(h_{opt})\}|\}.$$

Note that $opt_{MD}(S) \geq c_0 \geq opt_{MD}(S)/2$. Now let I^+ be the uniform distribution over the x_i, and let I^- be the uniform distribution over the y_i. Then I^+ and I^- can be regarded as induced distributions for h_{opt} with error rate $\beta' = c_0/m$. I^+ is induced by the joint behavior of the uniform distribution D^+ over $\{x_i \in S : x_i \in pos(h_{opt})\}$, and an adversary that draws a point uniformly from $\{x_i \in S : x_i \in neg(h_{opt})\}$; I^- can be decomposed over the y_i in a similar fashion.

Algorithm A' runs algorithm A many times, starting with accuracy parameter $\epsilon = 1$, and drawing from I^+ on each call to POS^{β}_{MAL} and from I^- on each call to NEG^{β}_{MAL}. Note that if h_A is an ϵ-good hypothesis with respect to D^+ and D^-, then we have $cost_{MD}(h_A, S) \leq 2\epsilon m + opt_{MD}(S)$. After each run, A' divides ϵ by 2. On some run of A, $\epsilon/r \leq c_0/2m$, and for this run we have $cost_{MD}(h_A, S) \leq (r+1)opt_{MD}(S) \leq 2r\, opt_{MD}(S)$, as desired. □

The first direction of this equivalence is also given by Blumer et al. [5]. Note that this equivalence as it is stated is *representation-based*, in the sense that it relies on the learning algorithm representing its hypothesis as a monomial. With more technical definitions for the problem $MD(C, H)$, we can in fact give a straightforward generalization of Theorem 20 for the problem of learning C by H in the presence of malicious errors, giving an equivalent optimization problem. In addition to simplifying the analysis of learning with errors in the distribution-free model — we only need to look at the equivalent optimization problem — these results allow us to weaken our restrictions on the adversary generating the errors. In particular, since there is no guarantee in the Minimize Disagreements problem on how the errors in the input sample are generated, it can be shown that the adversary gains no power by being allowed to see all coin flips of the learning algorithm, and all examples to be received by the learning algorithm *before* he generates the errors. This allows our model to incorporate faults such as *error bursts*, where all examples are in error for a short amount of time.

Figures 1 and 2 summarize some of the results in this paper.

References

[1] D. Angluin, P. Laird.
Learning from noisy examples.
Machine Learning, 2(4), 1988, pp. 343-370.

[2] D. Angluin, L.G. Valiant.
Fast probabilistic algorithms for Hamiltonian circuits and matchings.
Journal of Computer and Systems Sciences, 18, 1979, pp. 155-193.

[3] A. Blum.
Learning in an infinite attribute space.
Proceedings of the 22nd A.C.M. Symposium on the Theory of Computing, 1990, pp. 64-72.

[4] A. Blumer, A. Ehrenfeucht, D. Haussler, M. Warmuth.
Occam's razor.
Information Processing Letters, 24, 1987, pp. 377-380.

[5] A. Blumer, A. Ehrenfeucht, D. Haussler, M. Warmuth.
Learnability and the Vapnik-Chervonenkis dimension.
Journal of the A.C.M., 36(4), 1989, pp. 929-965.

[6] H. Chernoff.
A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations.
Annals of Mathematical Statistics, 23, 1952, pp. 493-509.

[7] V. Chvatal.
A greedy heuristic for the set covering problem.
Mathematics of Operations Research, 4(3), 1979, pp. 233-235.

[8] A. Ehrenfeucht, D. Haussler, M. Kearns. L.G. Valiant.
A general lower bound on the number of examples needed for learning.
Information and Computation, 82(3), 1989, pp. 247-261.

[9] M. Garey, D. Johnson.
Computers and intractability: a guide to the theory of NP-completeness.
Freeman, 1979.

[10] D. Haussler, M. Kearns, N. Littlestone, M. Warmuth.
Equivalence of models for polynomial learnability.
Proceedings of the 1988 Workshop on Computational Learning Theory, Morgan Kaufmann Publishers, 1988, pp. 42-55, and University of California at Santa Cruz Information Sciences Department, technical report number UCSC-CRL-88-06, 1988.

[11] D. Johnson.
Approximation algorithms for combinatorial problems.
Journal of Computer and Systems Sciences, 9, 1974, pp. 256-276.

[12] M. Kearns, M. Li.
Learning in the presence of malicious errors.
Proceedings of the 20th A.C.M. Symposium on the Theory of Computing, 1988, pp. 267-280.

[13] M. Kearns, M. Li, L. Pitt, L.G. Valiant.
On the learnability of Boolean formulae.
Proceedings of the 19th A.C.M. Symposium on the Theory of Computing, 1987, pp. 285-295.

[14] M. Kearns, L. Pitt.
A polynomial-time algorithm for learning k-variable pattern languages from examples.
Proceedings of the 1989 Workshop on Computational Learning Theory, Morgan Kaufmann Publishers, 1989, pp. 57-71.

[15] P. Laird.
Learning from good and bad data.
Kluwer Academic Publishers, 1988.

[16] N. Littlestone.
Learning quickly when irrelevant attributes abound: a new linear threshold algorithm.
Machine Learning, 2(4), 1988, pp. 245-318, and *Proceedings of the 28th I.E.E.E. Symposium on the Foundations of Computer Science*, 1987, pp. 68-77.

[17] L. Lovasz.
On the ratio of optimal integral and fractional covers.
Discrete Math, 13, 1975, pp. 383-390.

[18] R. Nigmatullin.
The fastest descent method for covering problems.
Proceedings of a Symposium on Questions of Precision and Efficiency of Computer Algorithms, Kiev, 1969 (in Russian).

[19] R. Rivest.
Learning decision lists.
Machine Learning, 2(3), 1987, pp. 229-246.

[20] R. Schapire.
On the strength of weak learnability.
Proceedings of the 30th I.E.E.E. Symposium on the Foundations of Computer Science, 1989, pp. 28-33.

[21] G. Shackelford, D. Volper.
Learning k-DNF with noise in the attributes.
Proceedings of the 1988 Workshop on Computational Learning Theory, Morgan Kaufmann Publishers, 1988, pp. 97-105.

[22] R. Sloan.
Types of noise in data for concept learning.
Proceedings of the 1988 Workshop on Computational Learning Theory, Morgan Kaufmann Publishers, 1988, pp. 91-96.

[23] L.G. Valiant.
A theory of the learnable.
Communications of the A.C.M., 27(11), 1984, pp. 1134-1142.

[24] L.G. Valiant.
Learning disjunctions of conjunctions.
Proceedings of the 9th International Joint Conference on Artificial Intelligence, 1985, pp. 560-566.

[25] V.N. Vapnik, A.Ya. Chervonenkis.
On the uniform convergence of relative frequencies of events to their probabilities.
Theory of Probability and its Applications, 16(2), 1971, pp. 264-280.

[26] K. Verbeurgt.
Learning DNF under the uniform distribution in quasi-polynomial time.
To appear, *Proceedings of the 1990 Workshop on Computational Learning Theory*, Morgan Kaufmann Publishers, 1990.

	$E_{MAL,+}(C)$ and $E_{MAL,-}(C)$	$E_{MAL}(C)$	$E_{CN,+}(C)$ and $E_{CN,-}(C)$	$E_{CN}(C)$
Upper bound on the optimal error rate	$\epsilon/(t(C)-1)$	$\epsilon/(1+\epsilon)$	$\epsilon/(1+\epsilon), \epsilon/(t(C)-1)$	$1/2$ [1]

Figure 1: Summary of general upper bounds on the optimal error rates for the malicious and noise models. We denote by $t(C)$ the largest value of t such that C is (positive or negative) t-splittable.

Class C	$E^{poly}_{MAL,+}(C)$ $E^{poly}_{MAL,-}(C)$ and $E^{poly}_{CN,+}(C)$ $E^{poly}_{CN,-}(C)$	$E^{poly}_{MAL}(C)$	$E^{poly}_{CN}(C)$
Upper bound		$O(\epsilon)$	
M_n	$\Theta(\epsilon/n)$		$\Theta(1)$
Lower bound	[24]	$\Omega(\ln(n/\epsilon)\epsilon/n)$	[1]
Upper bound		$O(\epsilon)$	
M^s_n	$\Theta(\epsilon/n)$	$\Omega((\epsilon/s)\ln((s/\epsilon)\ln(1+n/s))/\ln(1+n/s))$	$\Theta(1)$
Lower bound	[24]	$\Omega((\epsilon/s)(1/\log((s\ln n)/\epsilon)))$	[1]
Upper bound SF_n Lower bound	$\Theta(\epsilon/n)$	$\Theta(\epsilon)$	$\Theta(1)$

Figure 2: Summary of upper and lower bounds on the optimal polynomial time error rate (malicious and noise models) for the classes of monomials M_n, monomials of length s M^s_n, and symmetric functions SF_n.