INFORMATION AND COMPUTATION 110, 84–95 (1994)

On Learning Monotone DNF Formulae under Uniform Distributions*

LUDEK KUCERA

Department of Applied Mathematics, Charles University, Malostranske Namesti, Prague, Czechia

ALBERTO MARCHETTI-SPACCAMELA

Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza," Via Eudossiana 18, 00184 Rome, Italy

AND

MARCO PROTASI

Dipartimento di Matematica, Università di Roma "Tor Vergata," Via O. Raimondo, 00173 Rome, Italy

We show how to learn in polynomial time monotone d-term DNF formulae (formulae in disjunctive normal form with at most d terms) using positive examples drawn from a distribution that is a generalization of the uniform distribution.

1. INTRODUCTION

Learning algorithms were introduced into the formal framework of computational complexity theory by Valiant (1984). In that paper a formal interpretation of learning as inferring concepts without explicit programming was proposed. The problem is to study learning models that "promise to be relevant both to explain human experience and to build devices that can learn."

Valiant restricts the attention to the problem of recognizing whether a concept is true or not for given data. He shows that there are several concepts that are learnable in polynomial time by protocols that use

* Work supported by EEC Esprit BRA Project ALCOM (3075) and MURST Italian project "Algoritmi e Strutture di Calcolo." The results of this paper appeared in a preliminary version at the 15th ICALP Conference, Tampere, 1988. Part of the work of the second author was performed while he was at the University of L'Aquila, L'Aquila, Italy.

0890-5401/94 $6.00

positive and negative examples of the concept. These concepts are special classes of boolean functions.

Since then many efforts have been devoted to determine the border between the (polynomial time) learnability and nonlearnability of boolean functions (see, e.g., Ehrenfeucht *et al.*, 1988; Kearns and Li, 1988; Kearns *et al.*, 1987; Littlestone, 1988; Natarajan, 1987; Pitt and Valiant, 1988; and Valiant, 1984). The computational approach to learnability has also been successfully applied to other areas (Angluin, 1987; Berman and Roos, 1987; Blumer *et al.*, 1986; Haussler, 1988). In the present paper we concentrate our attention on boolean functions in disjunctive normal form (DNF formulae).

Specific protocols considered in the literature allow for different kinds of information supply and different notions of learnability. The most common approach assumes that the learner has access to a supply of data that exemplify the concept either in a positive or in a negative way and that she/he has performed his task when she/he can recognize both positive and negative examples of the concept with sufficiently large probability.

In this approach it is assumed that examples are drawn according to a probability distribution which is fixed but unknown. However, because of the generality of the model, it can be shown that important subclasses of DNF formulae cannot be learned in polynomial time. Therefore it is often supposed that the distribution of examples is known. Some classes of boolean formulae have been shown learnable with respect to special distributions, while being non learnable in general (Kearns *et al.* 1987). Note that these results do not imply that the class of DNF formulae is not learnable.

In this paper we consider the class of monotone DNF formulae with at most d terms. It was shown that these formulae are learnable only if the learning algorithm may output hypotheses that are not necessarily DNF formulae with (at most) d terms. In (Kearns *et al.*, 1987) it was shown how to learn d-term DNF formulae by an algorithm that outputs a d–CNF formula (a formula in conjunctive normal form with at most d variables in each clause). More recently, in (Blum and Singh, 1990) it was shown that d-term DNF formulae are learnable using hypotheses that are $O(n^{d-1})$-term DNF formulae. Both of these results do not make any assumptions about the distribution of examples.

On the other hand, it was proved in (Pitt and Valiant, 1988) that, for any $d \geqslant 2$, monotone d-term DNF formulae are nonlearnable in the distribution free model if the algorithm has to output a d-term DNF formula.

In this paper we assume that positive examples are drawn according to a distribution which is a generalization of the uniform distribution. This class of distributions is known in the literature as *product distributions*. Under

this assumption we are able to prove that monotone DNF formulae with d terms are learnable in time $O(n^{d+1})$ by an algorithm that outputs a d-term monotone formula. The algorithm uses $O(n(dc^{-(d-1)})^2/\varepsilon)$ positive examples, where c is a constant depending on the probability distribution of the examples (in the case of the uniform distribution $c = \frac{1}{2}$).

2. Definitions

We first recall some standard definitions. For a general discussion on different learning models and on their equivalence we refer to Haussler *et al.* (1988).

Let n be a natural number. A concept F is a boolean function with domain $\{0, 1\}^n$. Given a vector X, X is a positive example of the concept F if $F(X) = 1$; otherwise X is a negative example.

For any concept F there are many possible boolean formulae f such that f is consistent with the concept F. A class of representations of a concept is a set

$$B = \bigcup_n B_n$$

where, for each n, B_n is a subset of the set of all formulae with n variables $x_1, x_2, \ldots, x_n$.

For each $f \in B$, let $size(f)$ denote the smallest number of symbols needed to write a representation of f in B.

We suppose that the set $\{0, 1\}^n$ of all examples is a probability space with distribution D. Given a predicate Φ, we denote

$$\text{Prob}(\Phi) = \sum_{\Phi(X)=1} D(X).$$

If Ψ is another predicate such that $\text{Prob}(\Psi) \neq 0$ then

$$\text{Prob}(\Phi \mid \Psi) = \text{Prob}(\Phi \text{ and } \Psi)/\text{Prob}(\Psi).$$

For the sake of brevity we will use the following notation: $f = 0$ ($f = 1$) denotes the fact that f is false (true).

We always suppose that f is not constant, i.e.,

$$\text{Prob}(f = 0) \neq 0 \qquad \text{and} \qquad \text{Prob}(f = 1) \neq 0.$$

Using this assumption, we denote

$$D^+(X) = D(X)/\text{Prob}(f = 1) \qquad \text{if } X \text{ is a positive example,}$$

$$D^-(X) = D(X)/\text{Prob}(f = 0) \qquad \text{if } X \text{ is a negative example,}$$

$$\text{Prob}^+(\Phi) = \text{Prob}(\Phi \mid f = 1) \qquad \text{and} \qquad \text{Prob}^-(\Phi) = \text{Prob}(\Phi \mid f = 0).$$

We assume that the learning algorithm can call two procedures, which generate positive and negative examples, respectively. The probabilities of examples are given by D^+ and D^- and we suppose that they are independent for different calls of procedures.

Since the distribution is fixed but unknown the model is called distribution free.

DEFINITION 2.1. A function g is an ε-approximation of f (with respect to D) if

(i) $\text{Prob}^+(g(X) = 0) < \varepsilon$

(ii) $\text{Prob}^-(g(X) = 1) < \varepsilon$.

DEFINITION 2.2. A class of representations B is polynomially learnable if there exist a polynomial q and an algorithm L such that, for each n, $f \in B_n$, constants $\varepsilon > 0$ and $\theta > 0$, and distribution D, algorithm L, given a set of $q(n, \text{size}(f), 1/\varepsilon, 1/\theta)$ examples of f,

(i) outputs a formula g that, with probability $1 - \theta$, is an ε-approximation of f

(ii) runs in time polynomial in n, size(f), $1/\varepsilon$ and $1/\theta$.

Note that the distributions D^+, D^- are used twice. First they are used to generate examples which L uses to produce g and then they are used to test whether g is a good approximation of f. For the sake of brevity in the following we assume that $\varepsilon = \theta$.

As we have mentioned in the Introduction, the class of boolean formulae is not known to be learnable according to Definition 2.2. Hence we need to define suitable classes of boolean formulae.

DEFINITION 2.3. k-DNF is the class of all boolean formulae in disjunctive normal form with at most k literals per term. d-term-DNF is the class of boolean formulae in disjunctive normal form with at most d terms. Monotone DNF is the class of boolean DNF formulae with no negated literals.

Throughout the paper we use the following bounds due to Chernoff on the tails of the binomial distribution (see, for example, Bollobas, 1985, or Angluin and Valiant, 1979).

LEMMA 2.4. *For $0 \leqslant p \leqslant 1$ and n positive integer let $LE(p, m, \beta)$ denote the probability of at most $(1 - \beta)mp$ successes in m independent trials of a*

Bernoulli variable with probability of success p. Analogously let $GE(p, m, \beta)$ *denote the probability of at least* $(1+\beta)mp$ *successes. Then for* $0 \leqslant \beta \leqslant 1$ *we have*

(i) $LE(p, m, \beta) \leqslant e^{-\beta^2 mp/2}$

(ii) $GE(p, m, \beta) \leqslant e^{-\beta^2 mp/3}$.

Finally we also use the following inequality: $(1-x) \leqslant e^{-x}$.

3. Learning d-Term Monotone DNF Formulae under Uniform Distributions

In this section we show that if we assume that examples are drawn from the uniform distribution or from a distribution that generalizes it, we can considerably extend the class of learnable formulae. Note that Pitt and Valiant (1988) proved that monotone d-term DNF is not learnable for any $d \geqslant 2$ in the distribution free model. The distribution dependent approach was introduced in Kearns *et al.* (1987), where it was shown that if D^+ and D^- are uniform over the positive and negative examples then μ-DNF formulae (monotone DNF formulae in which each variable occurs at most once) are learnable. Other papers that study the learnability of μ-DNF formulae are by Angluin *et al.* (1990) and Hancock (1990).

We show that if d is a constant then the class of monotone d-term DNF formulae is learnable under a distribution which is a generalization of the uniform distribution using only positive examples.

From now on we suppose that a monotone formula f is given with d terms and a distribution D is defined as follows:

Each variable v is given a probability π_v and

$$D(X) = \left(\prod_{X(v)=1} \pi_v \right) \left(\prod_{X(v)=0} (1-\pi_v) \right),$$

where $X(v)$ denotes the value of v in the example X.

We suppose that there exists a constant $c > 0$ such that

$$c \leqslant \pi_v \leqslant 1-c$$

for each variable v. In the case where $c = \frac{1}{2}$ we have $\pi_v = \frac{1}{2}$ for all variables and we obtain the uniform distribution.

Note that in the case of monotone formulae a term is a nonempty set of variables. We can suppose, without loss of generality, that no term is a subset of another one, because removing a term which is a superset of another one does not change the value of the formula.

DEFINITION 3.1. A term U of a formula f is δ-weak if

$$\text{Prob}^+(U \text{ is satisfied}) < \delta.$$

A term U of formula f is δ-strong if it is not δ-weak. A formula f is δ-weak if it contains a δ-weak term.

Definition 3.1 makes it possible to distinguish between two different types of terms of the formula to be learned. Roughly speaking, a term is weak if its deletion gives a formula which is a good approximation of the original one. The following lemma formally shows that all sufficiently weak terms can be deleted without substantially changing the formula. The proof of the lemma is immediate and it is omitted.

LEMMA 3.2. *Let $\varepsilon < 1$ be a positive constant. A d-term DNF formula f can be ε-approximated by its subformula which is not (ε/d)-weak.*

Lemma 3.2 suggests that in order to learn a d-term monotone DNF formula f it is sufficient to find a formula g that contains all (ε/d)-strong terms. Before formally presenting Algorithm 3.3 for learning a d-monotone DNF formula we give an informal description.

The algorithm is based on the procedure *Infer*. This procedure takes as input a numerical parameter ϑ and returns a set $\mathcal{U}$ of sets of variables. The procedure is divided in two phases. In the first phase, after calling for a set of positive examples, it includes in $\mathcal{U}$ sets U of variables that might be terms or subsets of terms of the formula. In the second phase it removes from $\mathcal{U}$ those sets U that are subsets of other elements in $\mathcal{U}$. Crucial properties of the elements of $\mathcal{U}$ will be given in lemmas 3.6 and 3.9; lemma 3.6 guarantees that with sufficiently large probability all $(\vartheta/2d)$-weak terms and subsets of $(\vartheta/2d)$-weak terms do not belong to $\mathcal{U}$; lemma 3.9 guarantees that, with sufficiently large probability, all terms that are $(\vartheta c^{-(d-1)})$-strong belong to $\mathcal{U}$.

The algorithm for learning d-term monotone DNF formulae first calls the procedure *Infer* with $\vartheta = \varepsilon c^{d-1}/d$. In this case Lemma 3.9 guarantees that, with sufficiently large probability, all terms that are (ε/d)-strong are in the output of this first call. Moreover there could be also other sets of variables that are contained in the intersection of (ε/d)-weak terms of the formula to be learned. These subsets should not appear in the output of the algorithm and must be removed. Note that Lemma 3.6 implies that these subsets belong to terms that are $(\varepsilon c^{d-1}/2d^2)$-strong. In order to eliminate these subsets the algorithm calls the procedure *Infer* for a second time with parameter $\vartheta = (\varepsilon/2)(c^{d-1}/d)^2$; in this case Lemma 3.9 guarantees that, with sufficiently large probability, in the output of this second call these are all terms that are $(\varepsilon c^{d-1}/2d^2)$-strong. Hence in the removing phase of the

second call of the procedure *Infer*, all proper subsets of $(\varepsilon c^{d-1}/2d^2)$-strong terms are removed. Therefore, with large probability, all (ε/d)-strong terms of f appear in the output of both calls of *Infer* and, conversely, if a set of variables occurs in both outputs, it is a term of f.

We use the following notation: given a positive example E and a set of variables A, we write $E(A)=0$ if $v=0$ for each $v \in A$, $E(A)=1$ if $v=1$ for all $v \in A$.

ALGORITHM 3.3. {d-term monotone DNF formulae learning algorithm}.

procedure *Infer*(input ϑ: real; output $\mathcal{U}$: set of terms);
begin
$\mathcal{U} := \varnothing$;
$r := \lceil n/\vartheta \rceil$;
draw r positive examples $E_1, \ldots, E_r$;
for each set of variables A such that $|A| \leqslant d-1$ **do**
  begin
  if the number of examples E_i such that $E_i(A)=0$ is greater than $3n/4$
  then begin
    $U := \{v \mid v=1$ for each example E_i such that $E_i(A)=0\}$
    $\mathcal{U} := \mathcal{U} \cup \{U\}$;
    end;
  end;
remove from $\mathcal{U}$ all sets U such that U is a proper subset of some other $V \in \mathcal{U}$;
end;

begin
Infer$(\varepsilon c^{d-1}/d, \mathcal{U}_1)$;
Infer$(\varepsilon(c^{d-1}/d)^2/2, \mathcal{U}_2)$;
return the formula $\mathcal{U}_1 \cap \mathcal{U}_2$;
end.

In order to prove the correctness of the algorithm we need the following definitions.

Given a formula f and a set A of variables, let

$$\mathcal{T}(A) = \{U \mid U \text{ is a term of } f,\ U \cap A = \varnothing\}.$$

DEFINITION 3.4 We say that A contributes U (to $\mathcal{U}$) if the number of examples E_i in procedure *Infer* such that $E_i(A)=0$ is at least $3n/4$ and U is the set of variables that is added to $\mathcal{U}$. We say that A contributes (to $\mathcal{U}$) if there is some U such that A contributes U.

The following fact can easily be proved.

FACT 3.5 *For each term U, $U \in f$, there is an A such that $|A| < d$ and $\mathcal{T}(A) = \{U\}$.*

Proof. Let A include one variable of $U' - U$ for each term $U' \neq U$. Note that such a variable always exists for each $U' \neq U$ since otherwise U could be omitted from f without loss of generality. ∎

LEMMA 3.6. *With probability $1 - o(1)$, if A is any set of variables such that $|A| < d$ and $\mathcal{T}(A)$ contains only $\vartheta/(2d)$-weak terms of f, then A does not contribute when Infer is called with input parameter ϑ.*

Proof. In order for A to contribute, there must exist some $U \in \mathcal{T}(A)$ (and hence U is $\vartheta/(2d)$-weak) such that U is satified by at least $3n/4$ positive examples E_i for which $E_i(A) = 0$. Since $A \cap U = \varnothing$ for each $U \in \mathcal{T}(A)$, and by independence of the variables values, the probability that any given U is satisfied by E_i is independent of whether $E_i(A) = 0$. So, for any given E_i,

$$\text{Prob}^+(\text{there exists } U \text{ s.t. } U \in \mathcal{T}(A),\ U \text{ is satified by } E_i) \leqslant d\frac{\vartheta}{2d} = \frac{\vartheta}{2},$$

and therefore, by Chernoff's bound, with $m = n/\vartheta$, $p = \vartheta/2$, and $\beta = 1/2$, and using the fact that $3n/4 = (1 + 1/2)(\vartheta/2)(n/\vartheta)$, we can bound the probability that A contributes as follows:

$$\text{Prob}^+\left(\text{the number of examples verifying } E(A) = 0 \text{ is at least } \frac{3n}{4}\right)$$

$$\leqslant \exp\left(-\frac{1}{12}\frac{n}{\vartheta}\frac{\vartheta}{2}\right) = \exp\left(-\frac{n}{24}\right) = o(n^{-d}).$$

By summing over all sets A we obtain the Lemma. ∎

As we have already observed the correctness of the algorithm hinges on Lemma 3.9; before proving this lemma we need the following lemmas.

Given ϑ let us define $\kappa(\vartheta) = \vartheta c^{-(d-1)}$. In the following, for the sake of brevity, we simply write κ.

LEMMA 3.7. *With probability $1 - o(1)$, for each set of variables A such that $|A| < d$, if $\mathcal{T}(A)$ contains a κ-strong term, then A contributes when* Infer *is called with input parameter ϑ.*

Proof. Assume that A is such that $\mathcal{T}(A)$ contains a κ-strong term U of f. Since probabilities of occurrences of $v=1$ for different variables v are independent, we have for a positive example E

$$\begin{aligned}&\text{Prob}^+(U \text{ is satisfied by } E \text{ and } E(A)=0)\\ &\quad=\frac{\text{Prob}(U \text{ is satisfied by } E \text{ and } E(A)=0 \text{ and } f=1)}{\text{Prob}(f=1)}\end{aligned}$$

(since $U \cap A = \varnothing$ and because of the independence of the variables assignments)

$$\begin{aligned}&=\frac{\text{Prob}(U \text{ is satisfied by } E \text{ and } f=1)}{\text{Prob}(f=1)}\,\text{Prob}(E(A)=0)\\ &=\text{Prob}^+(U \text{ is satisfied by } E)\prod_{v\in A}\text{Prob}(v=0)\\ &\geqslant \kappa c^{d-1}=\vartheta\end{aligned}$$

and therefore, applying Chernoff's bounds with $m=n/\vartheta$, $p=\vartheta$ and $\beta=1/4$, and using the fact that $3n/4=(1-1/4)\,\vartheta n/\vartheta$, we bound the probability that A does not contribute as follows

$$\begin{aligned}&\text{Prob}(\text{for less than } 3n/4 \text{ examples } E_iU \text{ is satisfied by } E_i \text{ and } E_i(A)=0)\\ &\leqslant \exp\left(-\frac{1}{32}\,\vartheta\frac{n}{\vartheta}\right)=\exp\left(-\frac{n}{32}\right)=o(n^{-d}),\end{aligned}$$

By summing over all sets A we obtain the Lemma. ∎

LEMMA 3.8. *With probability* $1-o(1)$, *if* A *is any set such that* $|A|<d$ *and if* U *is any* $\vartheta/(2d)$*-strong term in* $\mathcal{T}(A)$, *then the contribution of* A, *when* Infer *is called with input parameter* ϑ, *is a subset of* U.

Proof. Assume that A is such that $\mathcal{T}(A)$ contains a $\vartheta/(2d)$-strong term U; furthermore let v be a variable such that $v \notin U$. Let $B=A\cup\{v\}$. Analogously to the preceeding lemma we obtain

$$\text{Prob}^+(U=1 \text{ and } E(B)=0)=\text{Prob}^+(U=1)\prod_{v\in B}\text{Prob}(v=0)\geqslant\frac{\vartheta}{2d}c^d;$$

therefore we obtain

$$\begin{aligned}&\text{Prob}(\text{at least one of } r \text{ examples } E_i \text{ verifies } U=1 \text{ and } E(B)=0)\\ &\quad=1-\text{Prob}(\text{no one of } r \text{ examples } E_i \text{ verifies } U=1 \text{ and } E(B)=0)\\ &\geqslant 1-\left(1-\frac{\vartheta}{2d}c^d\right)^{n/\vartheta}\end{aligned}$$

$$\geqslant 1 - \exp\left(-\frac{\vartheta}{2d} c^d \frac{n}{\vartheta}\right)$$

$$= 1 - \exp\left(-\frac{c^d}{2d} n\right)$$

$$= 1 - o(n^{-d}).$$

Hence if A contributes then, with probability $1 - o(n^{-(d-1)})$, no variable $v \notin U$ is in its contribution. ∎

LEMMA 3.9. *With probability* $1 - o(1)$, *the output of Infer contains all terms* U *such that* U *is a* κ*-strong term of* f *and no proper subset of a* κ*-strong term of* f.

Proof. Let U be a κ-strong term and consider the case where A is chosen such that $\mathcal{T}(A) = \{U\}$. In this case if $E_i(A) = 0$, then U is satisfied by E_i; hence the contribution of A (if any) must contain the whole set U. Lemmas 3.7 and 3.8 imply that, with very large probability, the contribution of A exists and is equal to U. The lemma follows by observing that the removing phase of the procedure eliminates all subsets of κ-strong terms. ∎

We are now able to prove the correctness of Algorithm 3.3

THEOREM 3.10. *With probability* $1 - o(1)$, *Algorithm* 3.3 *gives an* ε*-approximation of* f.

Proof. It follows that, with probability $1 - o(1)$:

If U is an (ε/d)-strong term of f then, by Lemma 3.9, it appears in the output of both calls of Procedure *Infer* by Algorithm 3.3;

If U is an $((\varepsilon/2d^2)\, c^{d-1})$-strong term of f, which is ε/d-weak, then it might appear in the output of both calls of Procedure *Infer* that are performed by Algorithm 3.3.

Conversely, if U appears in the output of Algorithm 3.3, then it appears in the output of the first call of *Infer* and, hence, it is a contribution of some A such that $\mathcal{T}(A)$ contains an $((\varepsilon/2d^2)\, c^{d-1})$-strong term (see Lemma 3.6). Therefore, by Lemma 3.8 and by observing that $\varepsilon/d > ((\varepsilon/2d^2)\, c^{d-1})$, U is a subset of some $((\varepsilon/2d^2)\, c^{d-1})$-strong term. However, by Lemma 3.9, proper subsets of $((\varepsilon/2d^2)\, c^{d-1})$-strong terms do not appear in the output of the second call of *Infer*. Thus, with very large probability, the output of Algorithm 3.3 contains only sets that are $((\varepsilon/2d^2)\, c^{d-1})$-strong

terms of f. Since there are at most d terms of f which are not terms of g, and all of them are ε/d-weak, then by applying Lemma 3.2 we obtain

$$\text{Prob}^-(g=1)=0,$$

$$\text{Prob}^+(g=0)\leqslant d\varepsilon/d\leqslant\varepsilon. \quad \blacksquare$$

The complexity of Algorithm 3.3 is stated in the following theorem whose proof can easily be obtained.

THEOREM 3.11. *Algorithm* 3.3 *has time complexity* $O(n^{d+1}/\varepsilon)$ *and uses* $O((n/\varepsilon)(dc^{-(d-1)})^2)$ *positive examples.*

ACKNOWLEDGMENTS

We thank an anonymous referee for careful reading of the manuscript and for many suggestions that improved the presentation of the paper.

RECEIVED April 19, 1989; FINAL MANUSCRIPT RECEIVED December 17, 1991

REFERENCES

1. ANGLUIN, D., FRAZIER, M., AND PITT, L. (1990) Learning conjunctions of Horn clauses, *in* "Proceedings, 31st Symposium on Foundations of Computer Science."
2. ANGLUIN, D., HELLERSTEIN, L., AND KARPINSKI M. (1989), "Learning Read-Once Formulas with Queries," Technical Report, ICSI.
3. ANGLUIN, D., AND VALIANT, L. G. (1979), Fast probabilistic algorithms for Hamiltonian circuits and matchings, *J. Comput. System. Sci.* **18**, 155–193.
4. BENEDEK, G. M., AND ITAI, A. (1988), Non uniform learnability, *in* "Proceedings, 15th Conference on Automata, Languages and Programming," Lecture Notes in Computer Science, Vol. 317, Springer Verlag, Berlin/New York.
5. BERMAN, P., AND ROOS, R. (1987), Learning one-counter languages in polynomial time, *in* "Proceedings, 28th Symposium on Foundations of Computer Science."
6. BLUM, A., AND SINGH, M. (1990), Learning functions of k terms, *in* "Proceedings, 3rd Workshop on Computational Learning Theory," Morgan Kauffman.
7. BLUMER, A., EHRENFEUCHT, A., HAUSSLER, D., AND WARMUTH, M. (1986), Classifying learnable geometric concepts with the Vapnik–Chervonenkis dimension, *in* "Proceedings, 18th ACM Symposium on Theory of Computing."
8. BOLLOBAS, B. (1985), "Random Graphs," Academic Press, London.
9. EHRENFEUCHT, A., HAUSSLER, D., KEARNS, M., AND VALIANT, L. (1989), A general lower bound on the number of examples needed for learning, *Inform. and Comput.* **82**, 247–261.
10. FULK, M. A., AND CASE, J. (Eds.) (1990), "Proceedings, 3rd Workshop on Computational Learning Theory," Morgan Kauffman.
11. HANCOCK, T. R. (1990), Identifying μ-formulae decision trees with queries, *in* "Proceedings, 3rd Workshop on Computational Learning Theory, Morgan Kauffman.
12. HAUSSLER, D. (1988), Quantifying inductive bias: AI learning algorithms and Valiant's learning framework, *Artifical Intelligence* **36**.

13. HAUSSLER, D., KEARNS, L., LITTLESTONE, N., AND WARMUTH, M. (1988), Equivalence of models for polynomial learnability, *in* "Proceedings 1st Workshop on Computational Learning Theory," Morgan Kauffman.
14. KEARNS, M., AND LI, M. (1988), Learning in the presense of malicious errors, *in* "Proceedings, 20th ACM Symposium on Theory of Computing."
15. KEARNS, M., LI, M., PITT, L., AND VALIANT, L. G. (1987), On the learnability of Boolean formulae, *in* "Proceedings 19th ACM Symposium on Theory of Computing."
16. KEARNS, M. AND VALIANT, L. G. (1987), Cryptographic limitations on learning boolean formulae and finite automata, *in* "Proceedings, 21th ACM Symposium on Theory of Computing."
17. LINIAL, N., MANSOUR, Y., AND RIVEST, R. L. (1988), Results in learnability and the Vapnik–Chervonenkis dimension, *in* "Proceedings, 29th Symposium on Foundations of Computer Science."
18. LITTLESTONE, N. (1988), Learning quickly when irrelevant attributes abound: A new linear treshold algorithm, *Machine Learning* **2**.
19. NATARAJAN, B. K. (1987), On learning Boolean functions, *in* "Proceedings, 19th ACM Symposium on Theory of Computing."
20. PITT, L., AND VALIANT, L. G. (1988), Computational limitations on learning from examples, *J. Assoc. Comput. Mach.* **35**.
21. RIVEST, R. HAUSSLER, D., AND WARMUTH, M. (Eds.) (1988), "Proceedings, 2nd Workshop on Computational Learning Theory," Morgan Kauffman.
22. VALIANT, L. G. (1984), A theory of the learnable, *Comm. ACM* **27**.