Computational Learning Theory: Survey and Selected Bibliography

Dana Angluin*
Yale University

1 Goals of the field

Give a rigorous, computationally detailed and plausible account of how learning can be done. Translation:

Rigorous: theorems, please.

Computationally detailed: exhibit algorithms that learn.

Plausible: with a feasible quantity of computational resources, and with reasonable information and interaction requirements.

The alert reader notices the buzzword "reasonable" — slack for a dazzling variety of models.

1.1 Definition of learning

Not now! This also is part of the goals. So far the emphasis has been on inductive learning (from examples) of concepts (binary classifications of examples) adapting the methods of analysis of algorithms and complexity theory to evaluate the resource use of proposed learning algorithms. When the examples are random, statistical methods are also important.

1.2 General resources

Directly relevant recurrent meetings are the International Workshops on Algorithmic Learning Theory, ALT [18, 19] and the annual Workshops on Computational Learning Theory, COLT [42, 63, 110, 130]. Currently, the only textbook in the field is Natarajan's [101]. Surveys by Laird [83] and Valiant [129] are valuable.

Somewhat more peripheral are the European meetings on Analogical and Inductive Inference, AII, and the AI machine learning community's annual International Conference on Machine Learning. In addition, the general AI meetings, AAAI and IJCAI, currently have a large number of papers devoted to learning, as do the neural net meetings.

1.3 Inductive inference

Inductive inference [16, 35, 80, 102] is to computational learning theory roughly as computability theory is to complexity and analysis of algorithms. Inductive inference and computability theory are historically and logically prior to and part of their polynomially-obsessed younger counterparts, share a body of techniques from recursion theory, and are a source of potent ideas and analogies in their respective fields. However, I must leave to others better qualified a systematic survey of recent progress in inductive inference.

2 The basic PAC model

The seminal paper is Valiant's [131]. In it, he proposed a new criterion of correctness for learning concepts from examples, emphasized the importance of polynomial time learning algorithms, and demonstrated that a classical algorithm learns k-CNF formulas with respect to the new criterion in polynomial time. He also emphasized the importance of coping with irrelevant attributes, introduced additional oracles for learning and gave learning algorithms for monotone DNF formulas and read-once formulas using these oracles, and gave cryptographic evidence that boolean circuits are unlearnable.

We now describe the components of the basic model introduced by Valiant; much of the work in the field can be understood as variations on this theme.

*Supported by NSF Grant CCR-9014943. Address: Department of Computer Science, Yale University, P. O. Box 2158, New Haven, CT 06520. E-mail: angluin@cs.yale.edu.

2.1 Examples

These may be boolean assignments, real numbers, points in Euclidean space, finite or infinite strings of symbols, etc. A universe X of possible examples is chosen, and a computational representation of individual examples.

2.2 Concepts

A concept is extensionally just a subset of X. An unknown *target concept* is to be learned; a single concept (intended to approximate the target concept) will be the output of the learning algorithm. A particular class C of possible target concepts is chosen; the learnability of C is investigated.

The *hypothesis space* H of a learning algorithm is the class of concepts from which its outputs are chosen. In the basic definition, C is a subclass of H, that is, we assume *adequacy of representation.* A computational representation of hypotheses from H is chosen. Since the choice of H and its computational representation strongly affects the learnability of C, the relevant notion is *learnability of C in terms of H.*

2.3 Distributions on Examples

Examples are generated independently according to a fixed but unknown probability distribution D on the universe X. Approximation of one concept c by another c' is measured with respect to D:

$$D(c \triangle c'),$$

where $c \triangle c'$ is the symmetric difference of the two sets c and c' of examples[1]. $D(c \triangle c')$ is the probability that an example drawn according to D will be classified differently by c and c', the *prediction error* of c' with respect to c and D.

2.4 Classes of distributions.

In general, we may know something (perhaps everything) about the distribution D. A class $\mathcal{D}$ of the possible distributions on X can be used to represent certain kinds of knowledge about D. In particular, a singleton class signifies that D is known to the learning algorithm.

2.5 Labelled Examples

Once the target concept c and the probability distribution D are specified, the oracle EXAMPLE is defined to take no input, draw an example x from X according to D, determine whether $x \in c$, and return $(x, +)$ if so and $(x, -)$ if not. Each call to EXAMPLE is statistically independent of every other call, and produces one labelled example of the target concept. In the terminology of AI, the learning is *supervised.* Some classes of concepts may be learned from positive examples only or negative examples only; that is, the learning algorithms ignore examples of the other sign.

[1]Measurability considerations enter here in the non-discrete case.

2.6 Learning algorithm

X, C, and H are fixed, along with their computational representations. A class $\mathcal{D}$ of distributions is fixed. A *learning algorithm* A takes as input two parameters ϵ and δ and has access to the EXAMPLE oracle determined by some $c \in C$ and some distribution D from $\mathcal{D}$. When A halts, its output is a single concept from H. The intuition — A draws labelled examples of c using the EXAMPLE oracle, and eventually conjectures hypothesis h meant to approximate c to within ϵ with respect to the distribution D. Sometimes this may not happen, but the probability that it doesn't should be less than δ.

2.7 PAC-identification

We say that the learning algorithm A PAC-identifies concepts from C in terms of H with respect to a class of distributions $\mathcal{D}$ if and only if for every distribution D in $\mathcal{D}$ and every concept $c \in C$, for all positive numbers ϵ and δ, when A is run with inputs ϵ and δ and access to the EXAMPLE oracle for D and c, it eventually halts and outputs a concept $h \in H$ such that with probability at least $1 - \delta$, $D(c \triangle h) < \epsilon$. The initials "PAC" stand for probably (except for δ) approximately (except for ϵ) correct.

2.8 Distribution-free learning

If A PAC-identifies concepts from C in terms of H with respect to the class of all possible distributions on X, then we simply say A PAC-identifies concepts from C in terms of H. The *distribution-free* requirement, that the learning algorithm work with respect to an arbitrary unknown distribution, is quite strong. However, since the performance of the output hypothesis is measured with respect to the same unknown distribution examples are drawn from, it is not impossibly strong. In practice restricted classes of distributions have not been fruitful, except in the case of the uniform distribution or the class of product distributions.

2.9 Polynomial time

We measure efficiency of the learning algorithm with respect to relevant parameters: size of examples, size of target concept, $1/\epsilon$, and $1/\delta$. "Size" of an example is usually the length of the string representing it in the selected computational representation of examples, though this is not generally used for real-valued examples. When the examples of a given concept are of uniform length, this is not problematic; otherwise, various expedients have been used.

In order to define the "size" of a target concept, we select a particular computational representation of concepts from the target class C. Then size of the target concept is usually the length of the string representing it in the representation chosen for C, though for real-valued examples it is often the number of parameters to specify the concept in the chosen representation. Different choices of representation may produce very different sizes for the same target concept c — consider the difference between representing boolean functions by circuits, boolean formulas, or DNF formulas.

As is usual, a bound polynomial in the relevant parameters is a gross indicator of computational tractability. This is all in the spirit of traditional complexity theory; nevertheless, it may get us into trouble.

2.10 Representations and complexity

We may define a *representation* $\mathcal{R}$ of a class of concepts simply as a set of ordered pairs of strings (x, u). We interpret u as specifying a concept c and x as specifying an example that is a member of c. For example, to define one representation, $\mathcal{R}_{DFA}$, of the class of regular sets over an alphabet Σ, we specify straightforward interpretations of the strings u as deterministic finite-state acceptors and the strings x as finite strings over Σ. Then $(x, u) \in \mathcal{R}_{DFA}$ if and only if the automaton represented by u accepts the string x. Representations inherit the usual definitions of complexity; for example, $\mathcal{R}_{DFA}$ is in PTIME, also in DSPACE($\log n$). Normally we restrict attention to representations in PTIME — this means that there is a uniform polynomial-time algorithm to classify the example represented by x according to the concept represented by u.

2.11 Learnability

With the background of a system of representing examples, concepts from C, and concepts from H, we may say that C is *learnable in terms of* H provided there exists a polynomial-time learning algorithm that PAC-identifies C in terms of H. We may want to say just C is *learnable*. Two conflicting definitions have been used:

1. C is learnable in terms of C.

2. C is learnable in terms of some class H with a polynomial-time representation.

Alternative (1) is desirable for positive results; it is also termed *properly learnable*. Alternative (2) is desirable for negative results; it is also termed *predictable*. We'll use "properly PAC-learnable" for (1) and "PAC-learnable" or "polynomially predictable" for (2).

2.12 Alternative definitions

The model described above is noticeably fuzzy — more of a "definition schema" than a definition. Many variants of the model have been considered. The fundamental paper of Haussler, Kearns, Littlestone and Warmuth [60, 61] provides careful definitions and systematic proofs of equivalence for a large number of alternative models, including one-button and two-button variants, the functional model, whether or not a bound is given on the size of the target concept, randomized algorithms and probabilistic halting conditions, dependence on δ, and on-line prediction from random examples. The paper is also a good source of useful proof techniques. See also the parameterization scheme proposed by Ben-David, Benedek and Mansour [23] for models of learnability.

In the *two-button* model there are separate distributions and EXAMPLE oracles for the positive and negative examples of a concept; the model described above is the *one-button* model.

The protocol, or environment of the learning algorithm, can be different. For example, in addition to the parameters ϵ and δ the learning algorithm may also be given bounds on the length of examples and the size of the target concept. Or, instead of the parameters ϵ and δ and access to the EXAMPLE oracle, the learning algorithm may simply be given a collection of labelled examples as its input, the *functional model*.

In an on-line prediction model, the learning algorithm indefinitely repeats a cycle of: (1) requesting an example (unlabelled), (2) predicting its classification according to the target concept, (3) receiving the correct classification. Haussler, Littlestone and Warmuth [62] consider the probability of a mistake of prediction at the t-th trial when the examples are drawn according to a fixed unknown probability distribution. Haussler, Kearns, Littlestone and Warmuth [60] prove the equivalence of on-line polynomial prediction from random examples with PAC-learning using a hypothesis class H with a polynomial-time representation, justifying the identification of these two terms.

Littlestone [87] defines the *absolute mistake bound* model of prediction; the worst-case number of mistakes of prediction over any sequence of examples must be bounded by a polynomial in the length of examples and the size of the target concept. A polynomial-time algorithm in the absolute mistake bound model can be transformed into a PAC-learning algorithm for the same class of concepts. However, Blum [26] proves that if one-way functions exist then there are PAC-learnable concept classes that are not predictable in Littlestone's absolute mistake bound model by a polynomial time algorithm.

3 Occam's razor

A basic technique in the construction of PAC-learning algorithms is "Occam's razor": take a large enough set of labelled examples and find a simple enough hypothesis $h \in H$ that is consistent with the labelled sample, that is, labels each example as in the sample.

3.1 The discrete razor

Blumer, Ehrenfeucht, Haussler and Warmuth [31] quantify "large enough" and "simple enough" in terms of the length of the output hypothesis as a string. They show that if there is a polynomial-time algorithm and a constant $\beta > 0$ such that for any sample of m examples labelled according to a concept $c \in C$ the algorithm finds a hypothesis $h \in H$ consistent with the sample whose length is bounded by the product of $m^{1-\beta}$ and a polynomial in the length of c, the class C is PAC-learnable in terms of H. Note that this does not require finding the smallest hypothesis consistent with the sample; in fact, its size may depend on the sample size, but not linearly — some nontrivial data-compression must be going on. Efficient approximations of set covers and weighted set covers are useful in this context.

3.2 The continuous razor

The string-based approach does not treat real numbers. The ground-breaking paper of Blumer, Ehrenfeucht, Haussler and Warmuth [32] demonstrates that the Vapnik-Chervonenkis dimension of the hypothesis space H may be used to give a result analogous to the discrete Occam's razor. The VC dimension of a class H of concepts is the size of the largest sample that can be labelled in all possible ways by concepts from H. For example, the class of closed intervals in the real line has VC-dimension 2. If it is finite, the VC-dimension gives a polynomial bound on the number of possible labellings of a set of m examples by concepts from H. For a finite class H, an upper bound on the VC-dimension is $\log H$, a hint of why it generalizes hypothesis length.

3.3 Converse?

Does PAC-learnability imply the existence of an Occam algorithm? Board and Pitt [104] show that a converse holds in many natural classes. Another kind of converse is given by Schapire [121].

3.4 Sample size

If we drop the requirement of a polynomial-time algorithm, we concentrate on the *sample size*, the number of examples required by a learning algorithm. An upper bound on the number of samples required by a consistent learning algorithm for C in terms of C is

$$O(\frac{1}{\epsilon}\log\frac{1}{\delta} + \frac{d}{\epsilon}\log\frac{1}{\epsilon}),$$

where d is the VC-dimension of C [32]. Anthony, Biggs and Shawe-Taylor [17] improve the implied constants. Ehrenfeucht, Kearns, Haussler and Valiant give an information-theoretic lower bound of

$$\Omega(\frac{1}{\epsilon}\log\frac{1}{\delta} + \frac{d}{\epsilon})$$

examples on any algorithm for PAC-learning a concept class C of VC-dimension d.

4 Voting schemes

Voting schemes give another general class of techniques for constructing learning algorithms.

4.1 Majority vote

Barzdin and Freivalds [20] use voting schemes to achieve small numbers of errors of prediction in inductive inference. As a simple example, if H is a finite set of concepts containing the target concept, the *majority vote* strategy for on-line prediction takes the first example, x, and predicts the label "+" if x belongs to a majority of the concepts in H and predicts the label "–" otherwise. When the correct label is received, all the concepts that misclassify this example are removed from H. At each point, H contains just those concepts consistent with all the labelled examples seen so far. Since each error of prediction removes at least half the remaining elements of H, the total number of errors of prediction is bounded by $\log|H|$. Littlestone [87] gives an optimal algorithm for this problem, which votes to minimize the worst-case number of errors of prediction. As shown by Goldman, Rivest and Schapire [48] there is a close relationship between counting problems and majority vote.

4.2 Weighted majority

In a generalization of this idea, we can assign a numerical weight $w(h)$ to each hypothesis $h \in H$ and use the weighted majority vote of the hypotheses to make predictions. The value of $w(h)$ is adjusted in response to the track record of h's successful and unsuccessful predictions. Majority vote corresponds to initial weights $w(h) = 1$, where $w(h)$ is changed to 0 if h makes an error of prediction. A more complex updating scheme can permit a simpler base of hypotheses.

Littlestone's Winnow algorithms [87] use multiplicative update rules to learn linearly separable boolean functions. For example, with the hypothesis space of single variables and a simple update rule, disjunctions of k out of n variables can be learned with at most $O(k \log n)$ errors of prediction. Littlestone [88] proves strong results on the resistance of Winnow to errors in the data. Littlestone and Warmuth [90] give a general weighted majority algorithm that is robust with respect to errors in the data, and they prove bounds on the absolute mistake bound of the algorithm as a function of the bound for the best algorithm in the initial set H of algorithms. A generalization of the weighted majority algorithm is given by Vovk [135]. Littlestone, Long and Warmuth [89] use a weighting scheme to develop an efficient algorithm for the on-line prediction of linear functions with a bound on the worst case sum of squared errors that is optimal up to a constant factor, which is also robust in the presence of noise in the data.

5 Closure and reductions

As in complexity theory, closure results and problem reductions give us a means of transferring learnability (or unlearnability) results among concept classes.

5.1 Closure results

The set of all PAC-learnable classes of concepts over universe X is closed under the union of two classes. That is, we can take PAC-learning algorithms for C_1 in terms of H_1 and C_2 in terms of H_2 and construct a PAC-learning algorithm for $C_1 \cup C_2$ in terms of $H_1 \cup H_2$. The idea is to run both learning algorithms and take the output with the smaller empirical prediction error for a sufficiently large sample. The set is also clearly closed under the operation of complementing each concept in a class with respect to X.

However, consider the operation of taking two classes C_1 and C_2 and forming the class of unions $c_1 \cup c_2$ where $c_1 \in C_1$ and $c_2 \in C_2$, with a straightforward representation. For example, applying this operation to two copies of the class of monomials yields the 2-term DNF formulas. It is not known whether the set of PAC-learnable classes of concepts is closed under this operation.[2] General closure results for the set of all PAC-learnable classes of concepts are disappointingly scarce.

Kearns, Li, Pitt and Valiant [75] give some restricted boolean closure results for the set of PAC-learnable classes. Another type of closure result is given by Helmbold, Sloan and Warmuth [68] for the nested differences of intersection-closed classes of concepts, predicated on the existence of polynomial-time algorithms to return the (set-theoretically) smallest concept containing a given set of positive examples. This may be applied, for example, to the nested differences of orthogonal rectangles.

5.2 Problem reductions

Kearns, Li, Pitt and Valiant [75] give substitution-based reductions for boolean formulas that show, e.g., the monotone or read-once[3] versions of classes of boolean formulas are no easier to PAC-learn than the basic classes. For example, if monotone read-once DNF formulas are PAC-learnable, then so are general DNF formulas.

Pitt and Warmuth [107] define a general type of problem reduction that preserves polynomial-time predictability, which they term *prediction-preserving reductions*. The basic idea is that each concept c in domain A is mapped to a concept $g(c)$ in domain B (with at most a polynomial increase in length of representation) and each example x in domain A is mapped to an example $f(x)$ in domain B by a polynomial-time algorithm in such a way that for all x and c, $x \in c$ if and only if $f(x) \in g(c)$. (This is a bit too simple in general, see the paper for the correct refinements.) The effect is that if we have a polynomial-time prediction algorithm in domain B, we may compose the reduction with it to get a polynomial-time prediction algorithm for domain A.

For example, we reduce general DNF formulas to monotone DNF formulas as follows. Formula ϕ over the variables $X_1, X_2, \ldots$ is mapped to the monotone formula $g(\phi)$ over the variables $X_1, Y_1, X_2, Y_2, \ldots$ by substituting Y_i for each occurrence of $\bar{X}_i$. Example $x = b_1 b_2 \cdots b_n$ signifying the assignment $X_i = b_i$ is mapped to the example

$$f(x) = b_1 \bar{b}_1 b_2 \bar{b}_2 \cdots b_n \bar{b}_n.$$

It is clear that the assignment x satisfies ϕ if and only if the assignment $f(x)$ satisfies $g(\phi)$. These transformations do not in general preserve special distributions (e.g., product distributions), so the distribution-free requirement is important here.

Pitt and Warmuth define *prediction-completeness* of a representation of concepts $\mathcal{R}$ over a set of such representations in the usual way, and prove, for example, that the class $\mathcal{R}_{DFA}$ of regular sets represented by deterministic finite acceptors is prediction-complete over DSPACE($\log n$), and the class $\mathcal{R}_{NFA}$ of regular sets represented by nondeterministic finite acceptors is prediction-complete over NSPACE($\log n$). Since the

[2] The set of *properly* PAC-learnable classes is not, as evidenced by the example given [105].

[3] A read-once or μ-formula contains at most one occurrence of each variable.

class $\mathcal{R}_{BF}$ of boolean formulas is in DSPACE($\log n$), this result implies that polynomial predictability of dfas would imply polynomial predictability of boolean formulas.

Pitt and Warmuth also give several examples of concept classes prediction-complete over PTIME by reductions from the class of all boolean circuits. By a similar technique, Long and Warmuth [91] prove that the class of convex polytopes specified as the convex hull of vertices is prediction-complete over PTIME. Schapire [119] considers the *pattern languages* [6] and exhibits a prediction-preserving reduction of nondeterministic boolean circuits to the pattern languages. (Note that the pattern languages have an NP-complete membership problem, and are therefore not necessarily a PTIME representation.)

6 What is PAC-learnable?

6.1 Classes of boolean formulas

Valiant [131] shows monomials and k-CNF formulas are properly PAC-learnable using only positive examples. Haussler [57] gives an algorithm using an approximate cover and Occam's razor that properly PAC-learns k-CNF using both positive and negative examples, in which the dependence on *irrelevant attributes* (that is, variables not appearing the target concept) is logarithmic rather than polynomial.

Littlestone [87] shows that k-CNF formulas are polynomial-time predictable by an on-line algorithm with logarithmic dependence on irrelevant attributes. His algorithm uses a weighted majority of clauses and gives a worst-case bound on the number of mistakes of prediction for any sequence of examples. He applies the technique more generally to linearly separable boolean formulas.

By constructing Occam algorithms, Rivest [116] shows that k-decision lists are properly PAC-learnable, and Ehrenfeucht and Haussler [36] show that rank k decision trees are properly PAC-learnable. Blum and Singh [29] show that for a fixed k, the class of all concepts denoted by $f(T_1, \ldots, T_k)$ where f is any boolean function on k arguments and the T_i are monomials, is PAC-learnable in terms of the class of general DNF formulas. The questions remain open of whether general CNF and DNF formulas or general decision trees are PAC-learnable.

6.2 Geometric & algebraic concepts

Blumer, Ehrenfeucht, Haussler and Warmuth show by means of their continuous version of Occam's razor that classes such as axis-parallel rectangles in E^n, open or closed halfspaces in E^n, or, for fixed k, the set of all halfspaces in E^n defined by surfaces of degree at most k are properly PAC-learnable. Baum [21] shows that for fixed k, unions or intersections of halfspaces in E^k are PAC-learnable. Long and Warmuth [91] give a reduction to prove the polynomial predictability of classes consisting of a union of a fixed number of flats, and an Occam algorithm for predicting fixed finite unions of boxes.

Abe [1] proves that the class of semilinear sets of dimensions 1 and 2 with unary coding is PAC-learnable by means of an Occam algorithm. Helmbold, Sloan and Warmuth [67] give an efficient on-line algorithm for predicting membership in an integer lattice, which is applied to learn rational lattices, cosets of lattices, and a subclass of the commutative regular languages. By the closure result for nested differences of intersection closed classes, they also show that nested differences of these classes are polynomially predictable [68].

7 What isn't PAC-learnable?

If RP = NP, then by the discrete Occam's razor, every PTIME representation of concepts H is properly PAC-learnable. (Use any convenient NP oracle to find a shortest hypothesis in H consistent with a given set of labelled examples.) Thus, non-learnability results are relative to unproved complexity theoretic or cryptographic assumptions.

7.1 If RP $\neq$ NP ...

So far, all the nonlearnability results based on NP $\neq$ RP have been *representation-dependent*. That is, they rely on the restriction that hypotheses must come from the class H. The general form of these results is: "If RP $\neq$ NP, then concept class H is not properly PAC-learnable." This does not preclude H being PAC-learnable in terms of some other class H'.

Pitt and Valiant [105] give several non-learnability results of this type. They show that if NP $\neq$ RP then k-term DNF formulas are not properly PAC-learnable (for $k \geq 2$), nor are boolean threshold formulas nor read-once formulas. Jerrum [71] similarly shows that a simple class of formulas invariant under cyclic shifts of the variables is not properly PAC-learnable.

Note that concepts representable by k-term DNF formulas are also learnable by k-CNF formulas, which are PAC-learnable. Here is a case in which H is not PAC-learnable by H (if NP $\neq$ RP), but H is PAC-learnable by a larger class H'. Blum and Singh [29] exhibit a generalization of this phenomenon to arbitrary boolean functions of k terms. Making the target class smaller or the hypothesis class larger cannot make learning harder; however, the opposite changes may make learning harder.

The basic lemma, due to Pitt and Valiant, is that if the problem of deciding whether there is a hypothesis in H consistent with an arbitrary labelled set of examples is NP-complete, then H is not properly PAC-learnable unless NP = RP. To see this, suppose A is an algorithm to PAC-learn H in terms of H. Let S be an arbitrary labelled set of examples and consider the distribution that assigns probability $1/|S|$ to each example from S, and zero probability to all other examples. Suppose we run A with $\epsilon < 1/|S|$ and $\delta = 1/2$, and this distribution on examples (labelled as they are in S.)

If there is a hypothesis $h \in H$ consistent with S, then with probability at least $1/2$ A must halt and output some $h' \in H$ that is ϵ-close to h. But by the definition of the distribution and ϵ, any concept ϵ-close to h must agree with h on all the examples from S, i.e., in this case h' is consistent with S. On the other hand, if there is no hypothesis $h \in H$ consistent with S, A will not output one. Thus our NP complete problem is in RP.

7.2 The pattern languages

Schapire [119] shows that the pattern languages are not polynomially predictable assuming the class of sets recognized by deterministic polynomial sized circuits is a proper subclass of the class of sets recognized by nondeterministic polynomial sized circuits. What's the catch? As noted above, the membership problem for pattern languages is NP-complete, so they are not necessarily a PTIME representation. In particular, it is conceivable that the pattern languages could be properly PAC-learnable yet not polynomially predictable. This is analogous to the distinction between identification and prediction in inductive inference.

7.3 Cryptographic assumptions

Stronger results may be had, apparently at the cost of stronger assumptions. The results are stronger: they claim that certain classes of concepts are not polynomially predictable — the representation of output concepts doesn't matter (as long as it is in PTIME.) The stronger assumptions and basic constructions are borrowed from public-key cryptography. It is logical that cryptography (which tries to make unpredictable things ever easier to compute) and computational learning theory (which tries to make more powerful classes of concepts predictable) should meet along certain frontiers.

Valiant [131] observes that the construction of a pseudo-random function by Goldreich, Goldwasser and Micali [50] is also the construction of a class of unpredictable boolean circuits. Thus, if one-way functions exist, the class of all boolean circuits is not polynomially predictable. Since the representation class of boolean circuits is in PTIME, Long and Warmuth's reduction shows that if one-way functions exist, convex polytopes in E^n represented by their vertices are not polynomially predictable. It is open whether the class of convex polytopes in E^n represented as an intersection of halfspaces is polynomially predictable.

Kearns and Valiant [78] show that more specific cryptographic assumptions imply that certain less powerful classes of concepts are not polynomially predictable. In particular, they show that assuming the intractability of any of the three problems (1) deciding quadratic residuosity modulo a composite (2) inverting RSA or (3) factoring Blum integers, the class of boolean formulas is not polynomially predictable, nor is the class of finite depth feedforward networks of threshold gates. Using Pitt and Warmuth's prediction-preserving reduction of boolean formulas to dfas, the same result applies to dfas.

The basic ideas may be summarized as follows. Imagine a secure public-key cryptosystem to encode single bit messages. For each pair of keys (e, d), the set of strings that decode to 1 should be unpredictable — given a polynomial number of examples of strings decoding to 1 and to 0 (which we can generate for ourselves, since this is a public-key system), we should have no polynomial advantage in guessing whether a new encoding of a coin flip decodes to 1 or 0. That is, the class of concepts

$$C_{(e,d)} = \{x : d(x) = 1\}$$

should be not polynomially predictable.

So the question comes down to: determine "small" classes of concepts sufficient to represent the decoding function in specific cryptosystems. Except this isn't enough — e.g., we don't know of any way to compute quadratic residuosity modulo a composite with a log depth circuit or a polynomial-sized boolean formula. Here Kearns and Valiant supply a very clever idea — move some tasks that are computationally onerous but cryptographically irrelevant into the "input." Put another way, create additional "features" that reduce the computational complexity of the decoding function but not its cryptographic strength. The relevant features in each case are the successive squares of the input string x modulo the composite N that is part of the key. This does not affect (modulo polynomial-time computation) the cryptographic security of the predicate, but it suffices to make the remaining part of the computation feasible with a log depth circuit (and therefore a polynomial-sized boolean formula.)

8 Errors and noise

Potential applications of learning algorithms will have to cope with data contaminated with errors both systematic and random. In the work described below, the assumption is that there is a correct target concept to be approximated within ϵ despite the errors in the examples. Various models of error in the EXAMPLE oracle have been studied.

8.1 Malicious errors

Valiant [128] defines *malicious errors* as follows. A coin flip with success probability β determines which calls to EXAMPLE will be affected by errors. When there is no error, EXAMPLE returns a correctly chosen labelled example as before. The result when an error occurs may be any example whatsoever with correct or incorrect sign, assumed to be generated by a malicious adversary. Valiant gives an algorithm to PAC-learn k-DNF formulas over n variables using only negative examples that tolerates a malicious error rate on the order of ϵ/n^k.

Kearns and Li [73, 74] prove that, under very weak conditions on the concept class, no learning algorithm can overcome a malicious error rate of $\beta = \epsilon/(1+\epsilon)$ or larger. They also show that for algorithms using only negative examples, no PAC-learning algorithm for k-DNF formulas can overcome an error rate of $\beta = c\epsilon/n^k$ for some $c > 0$. Of course, in the presence of errors there may be no hypothesis consistent with all the examples, so the simple Occam's razor does not apply. Kearns and Li give a generalization of Occam's razor in which it suffices to find a hypothesis consistent with a large fraction (at least $1 - \epsilon/2$) of the examples.

8.2 Less malicious errors

Angluin and Laird [14, 82] define a model of errors called *classification noise.* As in Valiant's model, a coin flip with success probability β determines which calls to EXAMPLE will be affected by error. When an error occurs, the example is still drawn correctly according to the distribution D, but it is returned with its sign reversed. This kind of error is particularly benign — Angluin and Laird give an algorithm that PAC-learns k-CNF formulas for any noise rate $\beta < 1/2$. In this case, the running time of the algorithm is allowed to grow polynomially in the inverse of $(\beta - 1/2)$.

Shackelford and Volper [123] consider a model of *attribute noise* for concepts with n boolean attributes. In their model, each example is potentially affected by noise in reporting its attributes. That is, each example is drawn correctly according to D and is then reported with the correct sign but with each of the n bits of the example flipped with probability $\beta < 1/2$. Shackelford and Volper give a procedure to overcome the effects of such noise provided β is known, which gives a polynomial-time algorithm that PAC-learns k-DNF formulas assuming β is known. The running time depends polynomially on the inverse of $(1/2 - \beta)$ in this case as well. Goldman and Sloan have shown how to remove the assumption that β is known for the case of learning 1-DNF.

Sloan [125, 126] defines also *malicious misclassification noise*, which is similar to misclassification noise except that when an error occurs, an adversary may choose not to reverse the sign of the example. This can model the situation in which certain examples are more likely to be misclassified than others. For a natural variant of attribute noise in which different attributes may have different rates of noise (each rate bounded by β), Goldman and Sloan have shown that under very weak assumptions about the concept class, no learning algorithm can tolerate a noise rate of $\beta = \epsilon/2$ or larger. Thus attribute noise with differing rates is essentially as bad as malicious errors.

9 Distributions, revisited

Recall that in the basic PAC-learning model, a learning algorithm has to be prepared to cope with an arbitrary unknown distribution on examples: the distribution-free requirement. Results described in this section show just how strong that requirement is, and propose ways of weakening it.

9.1 "Weak" is not so weak

The parameters δ, bounding the failure probability of the learning algorithm, and ϵ, bounding the prediction error of the hypothesis output when the learning algorithm succeeds, have very different roles in the learning protocol. To what extent may each be "boosted"? Is there a procedure to take a learning algorithm that achieves a mediocre failure probability (or prediction error) and improve it?

The answer is straightforward for δ — we can re-run the algorithm several times and take the "best-looking" hypothesis — that is, the one with the best empirical prediction error over a sufficient number of examples [60]. However, it is not at all straightforward for ϵ.

Kearns and Valiant [78] introduce a model called *weak learning*, in which it is sufficient to produce an output concept h such that

$$D(h \triangle c) < \frac{1}{2} - \frac{1}{p(n,s)},$$

where c is the target concept, p is a fixed polynomial, n is the length of examples, and s is the size of the target concept. Thus, h performs slightly (by an inverse polynomial) better than chance when used to predict c's labelling of examples drawn according to D. Their results show that even a weak learning algorithm for boolean formulas could be used to get a polynomial-time algorithm for any of the three basic cryptographic problems they consider.

Schapire [121, 122] proves this is no fluke: surprisingly enough, weak learnability implies PAC-learnability (not necessarily with the same hypothesis space.) His method exploits the distribution-free requirement by constructing filtered versions of the basic distribution that focus on the "weaknesses" of output hypotheses and force enough improvement that an output consisting of a majority vote of three hypotheses exhibits an improved prediction error. This can be iterated sufficiently many times to achieve any given prediction error ϵ. Schapire's results have a variety of consequences, including a strong partial converse of Occam's razor, bounds on the space complexity of learning, and bounds on the expected number of mistakes in the on-line model of prediction. Freund [41] gives an alternative construction, in which the final output hypothesis is a single majority vote of a large collection of hypotheses from the original class. Goldman, Kearns and Schapire [47] investigate the sample complexity of weak learning, which can be quite different from the sample complexity of PAC-learning.

9.2 Restricted classes of distributions

Suppose the learning algorithm "knows" the distribution D on examples, or at least a restricted class $\mathcal{D}$ of distributions from which it may be drawn: how much does this help? In several specific cases it does seem to help: learning algorithms have been devised for certain problems assuming the uniform distribution or the class of product distributions that significantly improve on the results known for the distribution free case. Benedek and Itai [25] consider the general situation of learning with respect to a fixed, known distribution and prove results characterizing learnability with respect to a fixed D.

9.3 Polynomial-time algorithms

Kearns and Pitt [76] give a polynomial-time algorithm for PAC-learning k-variable patterns in terms of disjunctions of k-variable patterns under the following class of distributions. The distribution on negative examples is arbitrary, and the distribution on positive examples is the product of k arbitrary distributions, each supplying one string to be substituted for a variable of the pattern.

As noted earlier, read-once boolean formulas are no easier to PAC-learn in the distribution free case than general boolean formulas, which may be difficult indeed, by the results of Kearns and Valiant [78]. However, the reduction does not preserve distributions. Read-once and read-k-times restrictions appear to interact particularly favorably with the uniform distribution and product distributions, and also, with membership queries (see below.) In the case of the read-once restriction, the reason appears to be that changing the value of a single variable affects only the path of gates from the unique occurrence of that variable to the root of the formula (viewed as a tree.)

Kearns, Li, Pitt and Valiant [75] show that read-once DNF formulas are PAC-learnable with respect to the uniform distribution, as do Pagallo and Haussler [103]. Goldman, Kearns and Schapire [49] show that some restricted classes of read-once formulas are PAC-learnable with respect to certain fixed simple product distributions. Schapire [120] significantly generalizes these results by giving an algorithm that PAC-learns the class of probabilistic read-once formulas with respect to the class of product distributions. The class of probabilistic read-once formulas properly generalizes the class of read-once formulas, and provides an interesting example of a class of p-concepts, defined and studied by Kearns and Schapire [77].

A $k\mu$-formula has at most k occurrences of each variable. Hancock and Mansour [56] give an algorithm that PAC-learns monotone $k\mu$-DNF formulas with respect to the class of product distributions.

9.4 AC^0 in quasi-polynomial time

Linial, Mansour and Nisan [86] consider learning the class AC^0 of constant depth circuits over the basis of AND, OR, and NOT with unbounded fan-in, applying Fourier spectrum techniques. Using a representation of boolean functions as linear combinations of parity functions of subsets of the input, they show that functions in AC^0 are well approximated with respect to the uniform distribution by their lower-order terms in this representation. (Intuitively, because AC^0 cannot compute good approximations to the parity of a large set of inputs.) This is used to derive a straightforward PAC-learning algorithm for AC^0 functions with respect to the uniform distribution that has time and sample complexity $O(n^{polylog(n)})$, *quasi-polynomial.* Furst, Jackson and Smith [43] improve this result to allow the class of product distributions on the boolean attributes in place of the uniform distribution. Verbeurgt [133] gives a simpler algorithm to PAC-learn DNF formulas with respect to the uniform distribution whose running time is quasi-polynomial, but whose sample complexity is polynomial.

10 Equivalence queries

Often it is convenient to develop learning algorithms using *equivalence queries* [8], usually in combination with other types of queries. The input to an equivalence query is a hypothesis $h \in H$, and the output is either "yes", if h is extensionally the same as the target concept c, or a *counterexample* x consisting of an arbitrarily chosen example classified differently by h and the target concept c. Thus a counterexample is an arbitrary element of $(h \triangle c)$.

In the equivalence query model, the criterion for successful learning is *exact identification*, that is, the learning algorithm must halt and output a hypothesis exactly equivalent to the target concept. The assumption is that the counterexamples are arbitrarily chosen by an adversary, though as Maass [92] points out, randomized learning algorithms necessitate care in specifying the type of adversary.

Since equivalence queries are dependent upon the hypothesis class H and its representation, we say C is *exactly identified in terms of* H. When we omit "in terms of," we imply that $H = C$. The term *extended equivalence queries* has also been used to signal the situation that $H \neq C$.

Equivalence queries in effect provide "direct access" to counterexamples, and may at first seem too powerful. However, a polynomial-time learning algorithm developed using equivalence queries can be transformed into an algorithm in the absolute mistake bound model [87] or in the PAC-model [8]. The idea for the first transformation is to run the learning algorithm until it makes an equivalence query with a hypothesis h, and then to use h to predict the labels of examples until (if ever) there is a mistake of prediction, say on example x. Then the suspended learning algorithm is resumed, with x as the counterexample returned by the equivalence query.

For the second transformation, we substitute for each equivalence query an "approximate equivalence test" that consists of checking the hypothesis h against a sufficiently large set of labelled examples drawn from EXAMPLE. If the examples are all correctly classified, we stop and declare success. Otherwise, any incorrectly classified example will serve as the counterexample.

Many of the known PAC-learnable discrete concept classes can be exactly learned in polynomial time using only equivalence queries. Blum [26] shows that this is not true in general if one-way functions exist. Maass and Turan [93] give polynomial-time algorithms for learning discrete geometric concepts using equivalence queries only. Yokomori [137] gives a polynomial-time algorithm for learning *very simple grammars* using only equivalence queries.

Angluin [10] shows that no polynomial-time algorithm can learn DNF formulas (resp., dfas, nfas, cfgs) in terms of DNF formulas (resp., dfas, nfas, cfgs) using only equivalence queries. The idea of is that if hypotheses are constrained to be polynomial size DNF formulas (or dfas, nfas, or cfgs) then particularly uninformative counterexamples may be chosen, enforcing very slow progress towards exact identification.

11 Active learning: positive

In the basic PAC model, as in the absolute mistake bound model and the equivalence query model, the selection of examples is not under the control of the learning algorithm; the model is one of *passive learning.* If we permit the learning algorithm control over the selection of examples, we get a more *active* model, in which certain classes of concepts may be easier to learn.

Valiant [131] considers specific oracles designed to give the learner more information about the target concept, and demonstrates the learnability of monotone DNF formulas and μ-formulas with respect to certain of these oracles. Angluin introduces membership and equivalence queries [8], and other types of queries [9]. Gasarch and Smith [44] consider queries in the context of inductive inference.

11.1 Membership queries

We may permit the learning algorithm access to another oracle, MEMBER, which takes as input an example x and returns as output the classification of x with respect to the target concept c. Such a query is called a *membership query.* In this setting we may define PAC-learning with membership queries in the obvious way.

The transformation sketched in the previous section shows that a polynomial-time algorithm that exactly identifies C in terms of H using equivalence and membership queries can be converted to a PAC-learning algorithm for C in terms of H with membership queries.

11.2 Automata and formal languages

Angluin [8] gives a polynomial-time algorithm for learning deterministic finite state acceptors using membership and equivalence queries. Sakakibara [118] generalizes this result to deterministic bottom up tree automata. Ishizaka [70] gives a polynomial-time algorithm that exactly identifies the class of simple deterministic context free grammars in terms of general context free grammara using membership and equivalence queries. Maler and Pnueli [95] give an efficient algorithm to learn a subclass of the infinitary regular sets using membership and equivalence queries.

Rivest and Schapire [111, 112, 113] consider the problem of a robot navigating in an unknown environment and attempting to construct an accurate map of that environment. For the case of finite state environments with deterministic actions, they give polynomial-time algorithms to construct a perfect model of the unknown environment, even in the absence of an operation to reset the robot to a known state. One of the corollaries of their results is a new and more efficient algorithm for learning dfas using equivalence and membership queries.

11.3 Geometric concepts

Bultman and Maass [34] give efficient algorithms for identifying a variety of discrete geometric concepts using only membership queries. Baum [22] demonstrates the power of membership queries and random examples for learning concepts describable by certain kinds of neural nets. In particular, he sketches a polynomial-time algorithm to learn the intersection of m halfspaces in n dimensions using random examples and membership queries.[4]

11.4 Subclasses of CNF and DNF

Angluin [7] gives a polynomial-time algorithm that exactly identifies k-term DNF formulas using equivalence and membership queries. Blum and Rudich [28] show that k-term DNF formulas can be exactly identified in terms of general DNF formulas by a randomized algorithm that uses membership and equivalence queries and runs in expected time $O(n \cdot 2^{O(k)})$. This means that DNF formulas of $O(\log n)$ terms are PAC-learnable with membership queries.

Valiant [131] gives an algorithm that can be viewed as exactly learning monotone DNF formulas in polynomial time using equivalence and membership queries [9]. A propositional Horn sentence is a CNF formula with at most one positive literal per clause. Angluin, Frazier and Pitt [11] give a polynomial-time algorithm that exactly identifies the class of propositional Horn sentences using membership and equivalence queries. However, "more" nonmonotonicity, e.g., two positive literals per clause, yields a problem no easier than predicting general CNF or DNF formulas with membership queries, which remains open.

11.5 Read-once formulas

Angluin, Hellerstein, and Karpinski [12] give a polynomial-time algorithm that exactly identifies the class of general read-once boolean formulas using membership and equivalence queries. Subsequent results have demonstrated the surprising power of membership queries to aid in learning read-once formulas over a variety of more powerful bases.

Raghavan and Schach [109] give a polynomial-time algorithm to learn single-contact switch configurations using equivalence and membership queries. This class of boolean functions properly includes the read-once boolean functions, and Raghavan and Schach's algorithm improves the time bound of the Angluin, Hellerstein and Karpinski algorithm.

A result due independently to Hancock [51] and Hellerstein and Karpinski [65] shows that there is a polynomial-time algorithm using membership and equivalence queries to learn read-once formulas over the basis of NOT and threshold gates, which is also a proper generalization of the read-once boolean formulas. Hancock [52] gives a polynomial-time algorithm using membership and equivalence queries to learn μ-formula decision trees, another proper generalization of read-once boolean formulas.

Hancock and Hellerstein [55] give polynomial algorithms using membership and equivalence queries that exactly identify read-once formulas over extended bases and fields. These results have recently been extended and improved by Bshouty, Hancock, and Hellerstein [33]. Hancock, Golea and Marchand [54] give a polynomial-time algorithm to learn nonoverlapping perceptron networks (or read-once formulas over a weighted threshold basis) using random examples and membership queries.

11.6 $k\mu$-formulas

Generalizing the read-once or μ restriction to allow two or a bounded number of occurrences of each variable, there has also been progress. Hancock gives polynomial-time algorithms to PAC-identify 2μ-DNF formulas and $k\mu$-decision trees using random examples and membership queries [53]. For the first class, Aizenstein and Pitt [4] prove the stronger result that 2μ-DNF formulas are exactly identifiable in polynomial time using equivalence and membership queries. Predicting 3μ-DNF formulas with membership queries is no easier than predicting general DNF formulas with membership queries [53], so 2 seems to be the limit of this line of attack. The status of 2μ-boolean formulas of greater structural complexity is open.

11.7 Errors in membership queries

Errors in the responses to membership queries have not yet been much studied. Sakakibara [117] defines a model in which answers to queries are subject to random independent noise, which he shows can be effectively removed by repeating the query sufficiently often. Angluin and Slonim [15] consider a model in which a fixed but randomly chosen fraction of membership queries can be answered "I don't know" and the answers are *persistent*, that is, do not change when queried again. They demonstrate a polynomial-time algorithm to learn monotone DNF formulas in this model.

[4]There is a technical constraint on the interaction of the concept and the distribution on examples that prevents certain pathological conditions.

12 Active learning: negative

12.1 Lower bounds

Maass and Turan [94] present general lower bounds on the number of membership and equivalence queries required for exact identification of all concepts from a class $\mathcal{C}$. In particular, they show this quantity is bounded below by $\frac{1}{7}$ of the Vapnik-Chervonenkis dimension of $\mathcal{C}$. They also give a lower bound in terms of the number of equivalence queries to identify elements of $\mathcal{C}$ using arbitrary subsets of the domain as hypotheses. In effect, these results establish that membership queries do not (even in pathological cases) confer an extraordinary advantage over computationally unrestricted algorithms using only examples.

12.2 Reductions

Generalizing Pitt and Warmuth's definitions, Angluin and Kharitonov [13] define prediction with respect to random examples and membership queries, and a reduction that preserves prediction with membership queries. In addition to the function g that maps concepts in domain A to concepts in domain B, and the function f that maps examples in domain A to examples in domain B, there is also a function h that maps examples in domain B to answers or examples in domain A. Intuitively, h is the inverse of f, so that examples queried in domain B may be transformed into examples to be queried in domain A. However, the examples queried in domain B may not be in the range of f, then the function h must itself supply an answer, typically a constant $+$ or $-$ for all examples not in the range of f.

With this new reduction, the class of dfas is apparently not complete over DSPACE($\log n$), however, the class of finite unions of dfas or two-way dfas is complete over DSPACE($\log n$). Also, general boolean formulas can be reduced to 3μ-boolean formulas. Hence, predicting 3μ-boolean formulas or finite unions of dfas or two-way dfas with membership queries is as hard as predicting boolean formulas with membership queries.

12.3 Implications of cryptography

Generalizing the results of Kearns and Valiant [78], Angluin and Kharitonov [13] use results and techniques from public-key cryptography to show limitations on the classes of concepts that are PAC-learnable using membership queries. Using Naor and Yung's construction of a public-key encryption system secure against chosen cyphertext attack [97], they show that assuming the intractability of (1) recognizing quadratic residues modulo a composite, (2) inverting RSA encryption, or (3) factoring Blum integers, there is no PAC-learning algorithm with membership queries for several concept classes, including general boolean formulas, constant depth threshold circuits, 3μ-boolean formulas, finite unions or intersections of deterministic finite acceptors, 2-way deterministic finite acceptors, nondeterministic finite acceptors, and context-free grammars.

They also show that if there exist one-way functions that cannot be inverted by polynomial-sized circuits, an application of existing secure signature schemes can be used to show that CNF and DNF formulas formulas are either PAC-learnable without membership queries, or are not PAC-learnable even with membership queries. This result shows that under fairly weak cryptographic assumptions membership queries won't help with learning CNF or DNF formulas.

Consequently, classes such as CNF and DNF formulas, or nondeterministic finite acceptors and context-free grammars, which have so far resisted PAC-learning with membership queries, appear to be out of reach.

12.4 Nonclosure results

The "folk wisdom" that finite conjunctions or disjunctions of concepts from a learnable class may be unlearnable is also supported by the results above. For example, though dfas and read-once boolean formulas are PAC-learnable with membership queries, the results above give cryptographic evidence that finite intersections or unions of dfas are not, and conjunctions or disjunctions of as few as three read-once boolean formulas are not.

13 Generalizations of the PAC model

Haussler [58] considers a powerful decision-theoretic generalization of PAC-learning to settings in which the rules to be learned are not necessarily boolean-valued nor deterministic, and adequacy of representation is not necessarily assumed. He proves very general results on the sample sizes sufficient for learning in such domains, using appropriate generalizations of the VC-dimension, with specific application to the problem of learning in terms of neural nets.

In one application of this approach, Kearns and Schapire [77] define a *p-concept* to be a map c from X to $[0,1]$, where $c(x)$ is interpreted as the probability that c classifies x positively. In this learning paradigm, examples are drawn according to an unknown distribution D on X and then stochastically classifed as positive or negative by an unknown p-concept c. They distinguish the goals of (1) finding a good prediction rule, that is, a decision rule whose prediction error is within ϵ of the Bayes optimal rule, and (2) finding a *good model of probability*, that is, a good approximation h to the target rule in the sense that $|h(x) - c(x)|$ is small for most inputs x with respect to D.

Yamanishi [136] defines a *stochastic rule* similarly and considers the problem of learning stochastic decision lists. Abe, Takeuchi and Warmuth [2] investigate relations among various definitions of "distance" between two p-concepts, with particular emphasis on the Kullback-Liebler divergence. Fischer, Pölt, and Simon [37] define related notions of multiplicative, additive or linear pac-estimability of a class of distributions. Abe and Warmuth [3] consider the concrete problem of approximating a distribution using a stochastic automaton.

14 Other models

At this point the reader may feel that the field is coherent, and the models settled; ***this impression is wrong!*** The goal stated in Section 1 is yet very distant, and the major part of the vitality of the field lies in its ability to generate new models, approaches, formalizations. We therefore point, possibly at the future:

Valiant [132]: a model of *neuroids*, neurons with state, and task-specific learning algorithms. Aldous and Vazirani [5]: an extension of the PAC model to examples generated using a Markov chain. Rivest and Sloan [114]: a model of learning a concept from subconcepts and an algorithm to learn boolean circuits, see also Kivinen [79]. Vitter and Lin [134]: a model of parallel learning. Natarajan [98]: a model of learning from exercises. Li and Vitanyi [85]: a theory of learning "simple" concepts from "simple" distributions based on program-size complexity. Floyd [40]: a model of space-bounded learning. Rivest and Sloan [115]: a Bayesian model of scientific theories and experiments. Ben-David, Itai and Kushilevitz [24]: a model of learning using estimates of "distance" from the target. Li [84]: a model of learning a string, motivated by DNA sequencing, see also Jiang and Li [72]. Helmbold and Long [66]: a model of learning concepts that change over time. Blum, Hellerstein and Littlestone [27]: dealing efficiently with infinite attribute spaces. Maass [92]: a model of worst-case "oblivious" example sequences and the power of randomized algorithms in this setting. Models of teaching have been defined and investigated by Goldman and Kearns [45, 46] and Shinohara and Miyano [124].

15 Open problems

In order of increasing strength: Are decision trees PAC-learnable? Is DNF or CNF PAC-learnable? Are intersections or unions of half spaces in E^n PAC-learnable? For membership queries: Determine the bases over which read-once formulas are PAC-learnable with membership queries. Determine which classes of 2μ-formulas are PAC-learnable with membership queries. Of course, the basic open problem is to account for the possibility of learning.

16 Comments

Thanks to Lenny Pitt for help improving the paper. With luck there will be another, more complete, version of this paper. Therefore, corrections, comments, suggestions, complaints, and updated references are welcome.

References

[1] N. Abe. Polynomial learnability of semilinear sets. In *Proceedings of the Second Annual Workshop on Computational Learning Theory*, pages 25–40. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1989.

[2] N. Abe, J. Takeuchi, and M. Warmuth. Polynomial learnability of probabilistic concepts with respect to the Kullback-Leibler divergence. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 277–289. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991.

[3] N. Abe and M. Warmuth. On the computational complexity of approximating distributions by probabilistic automata. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 52–66. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[4] H. Aizenstein and L. Pitt. Exact learning of read-twice DNF formulas. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, pages 170–179. IEEE Computer Society Press, 1991.

[5] D. Aldous and U. Vazirani. A Markovian extension of Valiant's learning model. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 392–396. IEEE Computer Society Press, 1990.

[6] D. Angluin. Finding patterns common to a set of strings. *J. Comp. Sys. Sci.*, 21:46–62, 1980.

[7] D. Angluin. Learning k-term DNF formulas using queries and counterexamples. Technical report, Yale University, YALE/DCS/RR-559, 1987.

[8] D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987.

[9] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

[10] D. Angluin. Negative results for equivalence queries. *Machine Learning*, 5:121–150, 1990.

[11] D. Angluin, M. Frazier, and L. Pitt. Learning conjunctions of Horn clauses. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 186–192. IEEE Computer Society Press, 1990.

[12] D. Angluin, L. Hellerstein, and M. Karpinski. Learning read-once formulas with queries. Technical report, University of California at Berkeley, Report No. 89/528, 1989. (Also, International Computer Science Institute Technical Report TR-89-05099.) *JACM*, to appear.

[13] D. Angluin and M. Kharitonov. When won't membership queries help? In *Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing*, pages 444–454. ACM Press, 1991.

[14] D. Angluin and P. Laird. Learning from noisy examples. *Machine Learning*, 2:343–370, 1988.

[15] D. Angluin and D. Slonim. Learning monotone DNF with an incomplete membership oracle. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 139–146. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991.

[16] D. Angluin and C. Smith. Inductive inference: theory and methods. *Comput. Surveys*, 15:237–269, 1983.

[17] M. Anthony, N. Biggs, and J. Shawe-Taylor. The learnability of formal concepts. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 246–257. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[18] S. Arikawa, S. Goto, S. Ohsuga, and T. Yokomori, editors. *Proceedings of the First International Workshop on Algorithmic Learning Theory.* Japanese Society for Artificial Intelligence, Tokyo, October 8-10, 1990.

[19] S. Arikawa, A. Maruoka, and T. Sato, editors. *Proceedings of the Second International Workshop on Algorithmic Learning Theory.* Japanese Society for Artificial Intelligence, Tokyo, October 23-25, 1991.

[20] J. M. Barzdin and R. V. Freivalds. On the prediction of general recursive functions. *Sov. Math. Dokl.*, 13:1224–1228, 1972.

[21] E. Baum. On learning a union of half spaces. *Journal of Complexity*, 6:67–101, 1990.

[22] E. Baum. Polynomial time algorithms for learning neural nets. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 258–272. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[23] S. Ben-David, G. Benedek, and Y. Mansour. A parameterization scheme for classifying models of learnability. In *Proc. of the Second Annual Workshop on Computational Learning Theory*, pages 285–302. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1989.

[24] S. Ben-David, A. Itai, and E. Kushilevitz. Learning by distances. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 232–245. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[25] G. Benedek and A. Itai. Learnability by fixed distributions. In *Proceedings of the 1988 Workshop on Computational Learning Theory*, pages 80–90, 1988.

[26] A. Blum. Separating distribution-free and mistake-bound learning models over the boolean domain. In *Proc. 31st Annual Symposium on Foundations of Computer Science*, pages 211–218. IEEE Computer Society Press, 1990.

[27] A. Blum, L. Hellerstein, and N. Littlestone. Learning in the presence of finitely or infinitely many irrelevant attributes. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 157–166. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991.

[28] A. Blum and S. Rudich. Fast learning of k-term DNF formulas with queries. In *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing.* ACM Press, 1992.

[29] A. Blum and M. Singh. Learning functions of k terms. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 144–153. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[30] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Classifying learnable geometric concepts with the Vapnik-Chervonenkis dimension. In *Proc. 18th ACM Symposium on Theory of Computing*, pages 273–282. ACM Press, 1986.

[31] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, 1987.

[32] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *J. ACM*, 36:929–965, 1989.

[33] N. Bshouty, T. Hancock, and L. Hellerstein. Learning arithmetic read-once formulas. In *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing.* ACM Press, 1992.

[34] W. Bultman and W. Maass. Fast identification of geometric objects with membership queries. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 337–352. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991.

[35] J. Case and C. Smith. Comparison of identification criteria for machine inductive inference. *Theor. Comp. Sci.*, 25:193–220, 1983.

[36] A. Ehrenfeucht and D. Haussler. Learning decision trees from random examples. In *Proceedings of the 1988 Workshop on Computational Learning Theory*, pages 182–194, 1988.

[37] P. Fischer, S. Polt, and H. Simon. Probably almost Bayes decisions. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 88–94. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991.

[38] P. Fischer and H. Simon. On learning ring-sum expansions. *SIAM J. Comput.*, 21:181–192, 1992.

[39] S. Floyd. *On Space-bounded Learning and the Vapnik-Chervonenkis Dimension.* PhD thesis, University of California, Berkeley, 1989. Issued as ICSI TR-89-061.

[40] S. Floyd. Space-bounded learning and the Vapnik-Chervonenkis dimension. In *Proc. of the Second Annual Workshop on Computational Learning Theory*, pages 349–364. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1989.

[41] Y. Freund. Boosting a weak learning algorithm by majority. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 202–216. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[42] M. Fulk and J. Case, editors. *Proceedings of the Third Annual Workshop on Computational Learning Theory.* Morgan Kaufmann Pubishers, Inc., San Mateo, CA, Rochester, NY, August 6-8, 1990.

[43] M. Furst, J. Jackson, and S. Smith. Improved learning of AC^0 functions. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 317–325. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991.

[44] W. Gasarch and C. Smith. Learning via queries. In *Proc. 29th Annual Symposium on Foundations of Computer Science*, pages 130–137. IEEE Computer Society Press, 1988.

[45] S. Goldman. *Learning Binary Relations, Total Orders, and Read-Once Formulas.* PhD thesis, MIT, 1990. Issued as MIT/LCS/TR-483.

[46] S. Goldman and M. Kearns. On the complexity of teaching. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 303–314. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991.

[47] S. Goldman, M. Kearns, and R. Schapire. On the sample complexity of weak learning. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 217–231. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[48] S. Goldman, R. Rivest, and R. Schapire. Learning binary relations and total orders. In *Proceedings of the Thirtieth Annual Symposium on Foundations of Computer Science*, pages 46–51. IEEE Computer Society Press, 1989.

[49] S. A. Goldman, M. J. Kearns, and R. E. Schapire. Exact identification of circuits using fixed points of amplification functions. In *Proc. 31st Annual Symposium on Foundations of Computer Science*, pages 193–202. IEEE Computer Society Press, 1990.

[50] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. In *Proc. 25th Annual Symposium on Foundations of Computer Science*, pages 464–479. IEEE, 1984.

[51] T. Hancock. Identifying μ-formula decision trees with queries. Technical report, Harvard University Center for Research in Computing Technology, TR-16-90, 1990.

[52] T. Hancock. Identifying μ-formula decision trees with queries. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 23–37. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[53] T. Hancock. Learning 2μDNF formulas and $k\mu$ decision trees. In *Proceedings of the Fourth Annual Workshop on Computational Learning The-*

ory, pages 199–209. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991.

[54] T. Hancock, M. Golea, and M. Marchand. Learning nonoverlapping perceptron networks from examples and membership queries. Technical report, Harvard University Center for Research in Computing Technology, TR-26-91, 1991.

[55] T. Hancock and L. Hellerstein. Learning read-once formulas over fields and extended bases. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 326–336. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991.

[56] T. Hancock and Y. Mansour. Learning monotone $k\mu$ DNF formulas on product distributions. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 179–183. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991.

[57] D. Haussler. Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence*, 36:177–221, 1988.

[58] D. Haussler. Generalizing the PAC model: sample size bounds from metric dimension-based uniform convergence results. In *Proceedings of the Thirtieth Annual Symposium on Foundations of Computer Science*, pages 40–45. IEEE Computer Society Press, 1989.

[59] D. Haussler. Learning conjunctive concepts in structural domains. *Machine Learning*, 4:7–40, 1989.

[60] D. Haussler, M. Kearns, N. Littlestone, and M. Warmuth. Equivalence of models for polynomial learnability. In *Proc. of the 1988 Workshop on Computational Learning Theory*, pages 42–55. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1988.

[61] D. Haussler, M. Kearns, N. Littlestone, and M. Warmuth. Equivalence of models for polynomial learnability. Technical report, University of California, Santa Cruz, UCSC-CRL-88-06, 1988.

[62] D. Haussler, N. Littlestone, and M. Warmuth. Predicting $\{0,1\}$-functions on randomly drawn points. In *Proc. 29th Symposium on Foundations of Computer Science*, pages 100–109. IEEE Computer Society Press, 1988.

[63] D. Haussler and L. Pitt, editors. *Proceedings of the 1988 Workshop on Computational Learning Theory.* Morgan Kaufmann Pubishers, Inc., San Mateo, CA, Boston, MA, August 3-5, 1988.

[64] L. Hellerstein. *On Characterizing and Learning Some Classes of Read-once Functions.* PhD thesis, University of California, Berkeley, 1989.

[65] L. Hellerstein and M. Karpinski. Computational complexity of learning read-once formulas over different bases. Technical report, International Computer Science Institute, Berkeley, CA, TR-91-014, 1991.

[66] D. Helmbold and P. Long. Tracking drifting concepts using random examples. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 13–23. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991.

[67] D. Helmbold, R. Sloan, and M. Warmuth. Learning integer lattices. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 288–302. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[68] D. Helmbold, R. Sloan, and M. Warmuth. Learning nested differences of intersection-closed classes. *Machine Learning*, 5:165–196, 1990.

[69] O. Ibarra and T. Jiang. Learning regular languages from counterexamples. In *Proc. of the 1988 Workshop on Computational Learning Theory*, pages 371–385. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1988.

[70] H. Ishizaka. Learning simple deterministic languages. In *Proceedings of the Second Workshop on Computational Learning Theory*, pages 162–174. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1989.

[71] M. Jerrum. Simple translation-invariant concepts are hard to learn. Technical report, University of Edinburgh, Department of Computer Science, CSR-12-91, 1991.

[72] T. Jiang and M. Li. On the complexity of learning strings and sequences. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 367–371. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991.

[73] M. Kearns. *The Computational Complexity of Machine Learning.* PhD thesis, Harvard University, 1989. To be published by MIT Press in the ACM Distinguished Dissertation Series.

[74] M. Kearns and M. Li. Learning in the presence of malicious errors. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 267–280. ACM Press, 1988.

[75] M. Kearns, M. Li, L. Pitt, and L. Valiant. On the learnability of boolean formulae. In *Proc. 19th ACM Symposium on Theory of Computing*, pages 285–295. ACM Press, 1987.

[76] M. Kearns and L. Pitt. A polynomial-time algorithm for learning k-variable pattern languages from examples. In *Proceedings of the Second Annual Workshop on Computational Learning Theory*, pages 57–70. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1989.

[77] M. Kearns and R. Schapire. Efficient distribution-free learning of probabilistic concepts. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 382–391. IEEE Computer Society Press, 1990.

[78] M. Kearns and L. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. In *Proc. 21st ACM Symposium on Theory of Computing*, pages 433–444. ACM Press, 1989.

[79] J. Kivinen. Reliable and useful learning. In *Proceedings of the Second Workshop on Computational Learning Theory*, pages 365–380. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1989.

[80] R. Klette and R. Wiehagen. Research in the theory of inductive inference by GDR mathematicians - a survey. *Information Sciences*, 22:149–169, 1980.

[81] E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. In *Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing*, pages 455–464. ACM Press, 1991.

[82] P. Laird. *Learning From Good Data and Bad.* PhD thesis, Yale University, 1987. Published by Kluwer Academic Publishers, 1988.

[83] P. Laird. A survey of computational learning theory. In R. Banerji, editor, *Formal Techniques in Artificial Intelligence: A Sourcebook*, pages 173–215. Elsevier Science Publishers, 1990.

[84] M. Li. Towards a DNA sequencing theory: learning a string. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 125–134. IEEE Computer Society Press, 1990.

[85] M. Li and P. Vitanyi. A theory of learning simple concepts under simple distributions and average case complexity for the universal distribution. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 34–39. IEEE Computer Society Press, 1989.

[86] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier transform, and learnability. In *Proceedings of the Thirtieth Annual Symposium on Foundations of Computer Science*, pages 574–579. IEEE Computer Society Press, 1989.

[87] N. Littlestone. Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[88] N. Littlestone. Redundant noisy attributes, attribute errors, and linear-threshold learning using Winnow. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 147–156. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991.

[89] N. Littlestone, P. Long, and M. Warmuth. Online learning of linear functions. In *Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing*, pages 465–475. ACM Press, 1991.

[90] N. Littlestone and M. Warmuth. The weighted majority algorithm. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 256–261. IEEE Computer Society Press, 1989.

[91] P. Long and M. Warmuth. Composite geometric concepts and polynomial predictability. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 273–287. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[92] W. Maass. On-line learning with an oblivious environment and the power of randomization. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 167–175. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991.

[93] W. Maass and G. Turan. On the complexity of learning from counterexamples. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 262–267. IEEE Computer Society Press, 1989.

[94] W. Maass and G. Turan. On the complexity of learning from counterexamples and membership queries. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 203–210. IEEE Computer Society Press, 1990.

[95] O. Maler and A. Pnueli. On the learnability of infinitary regular sets. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 128–136. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991.

[96] S. Miyano, A. Shinohara, and T. Shinohara. Which classes of elementary formal systems are polynomial-time learnable? In *Proceedings of the Second Workshop on Algorithmic Learning Theory*, pages 139–150. Japanese Society for Artificial Intelligence, 1991.

[97] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 427–437. Association for Computing Machinery, 1990.

[98] B. Natarajan. On learning from exercises. In *Proceedings of the Second Workshop on Computational Learning Theory*, pages 72–87. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1989.

[99] B. Natarajan. On learning sets and functions. *Machine Learning*, 4:67–97, 1989.

[100] B. K. Natarajan. On learning boolean functions. In *Proc. 19th ACM Symposium on Theory of Computing*, pages 296–304. ACM Press, 1987.

[101] B. K. Natarajan. *Machine Learning: a Theoretical Approach.* Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991.

[102] D. Osherson, M. Stob, and S. Weinstein. *Systems That Learn.* MIT Press, Cambridge, MA, 1986.

[103] G. Pagallo and D. Haussler. A greedy method for learning μ-DNF functions under the uniforn distribution. Technical report, University of California at Santa Cruz, UCSC-CRL-89-12, 1989.

[104] L. Pitt and R. Board. On the necessity of Occam algorithms. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 54–63. ACM Press, 1990.

[105] L. Pitt and L. Valiant. Computational limitations on learning from examples. *J. ACM*, 35:965–984, 1988.

[106] L. Pitt and M. Warmuth. The minimum consistent DFA problem cannot be approximated within any polynomial. In *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, pages 421–432. ACM Press, 1989.

[107] L. Pitt and M. Warmuth. Prediction-preserving reducibility. *J. of Computer and System Sciences*, 41:430–467, 1990.

[108] S. Porat and J. Feldman. Learning automata from ordered examples. In *Proc. of the 1988 Workshop on Computational Learning Theory*, pages 386–396. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1988.

[109] V. Raghavan and S. Schach. Learning switch configurations. In *Proceedings of Third Annual Workshop on Computational Learning Theory*, pages 38–51. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[110] R. Rivest, D. Haussler, and M. Warmuth, editors. *Proceedings of the Second Annual Workshop on Computational Learning Theory.* Morgan Kaufmann Pubishers, Inc., San Mateo, CA, Santa Cruz, CA, July 31- August 2, 1989.

[111] R. Rivest and R. Schapire. Diversity-based inference of finite automata. In *Proc. 28th IEEE Symposium on Foundations of Computer Science*, pages 78–87. IEEE Computer Society Press, 1987.

[112] R. Rivest and R. Schapire. A new approach to unsupervised learning in deterministic environments. In *Proc. of the 4th International Workshop on Machine Learning*, pages 364–375. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1987.

[113] R. Rivest and R. Schapire. Inference of finite automata using homing sequences. In *Proc. 21st ACM Symposium on Theory of Computing*, pages 411–420. ACM Press, 1989.

[114] R. Rivest and R. Sloan. Learning complicated concepts reliably and usefully. In *Proc. of the 1988 Workshop on Computational Learning Theory*, pages 69–79. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1988.

[115] R. Rivest and R. Sloan. A new model for inductive inference. In *Proceedings of the Second Conference on Theoretical Aspects of Reasoning About Knowledge*, pages 13–27. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1988.

[116] R. L. Rivest. Learning decision lists. *Machine Learning*, 2:229–246, 1987.

[117] Y. Sakakibara. On learning from queries and counterexamples in the presence noise. *Information Processing Letters*, to appear.

[118] Y. Sakakibara. Learning context-free grammars from structural data in polynomial time. *Theoretical Computer Science*, pages 223–242, 1990.

[119] R. Schapire. Pattern languages are not learnable. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 122–129. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[120] R. Schapire. Learning probabilistic read-once formulas on product distributions. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 184–198. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991.

[121] R. E. Schapire. The strength of weak learnability. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 28–33. IEEE Computer Society Press, 1989.

[122] R. E. Schapire. *The Design and Analysis of Efficient Learning Algorithms.* PhD thesis, MIT, 1991. Issued as MIT/LCS/TR-493.

[123] G. Shackelford and D. Volper. Learning k-DNF with noise in the attributes. In *Proceedings of the 1988 Workshop on Computational Learning Theory*, pages 97–103. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1988.

[124] A. Shinohara and S. Miyano. Teachability in computational learning. *New Generation Computing*, 8:337–347, 1991.

[125] R. Sloan. Types of noise in data for concept learning. In *Proceedings of the 1988 Workshop on Computational Learning Theory*, pages 91–96. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1988.

[126] R. H. Sloan. *Computational Learning Theory: New Models and Algorithms.* PhD thesis, MIT, 1989. Issued as MIT/LCS/TR-448.

[127] L. Valiant. Deductive learning. *Phil. Trans. Roy. Soc. Lond. A*, 312:441–446, 1984.

[128] L. Valiant. Learning disjunctions of conjunctions. In *Proc. 9th IJCAI*, pages 560–566. IJCAI, 1985.

[129] L. Valiant. A view of computational learning theory. In C. W. Gear, editor, *Computation & Cognition: Proceedings of the First NEC Research Symposium*, pages 32–51. SIAM, 1991.

[130] L. Valiant and M. Warmuth, editors. *Proceedings of the Fourth Annual Workshop on Computational Learning Theory.* Morgan Kaufmann Pubishers, Inc., San Mateo, CA, Santa Cruz, CA, August 5-7, 1991.

[131] L. G. Valiant. A theory of the learnable. *C. ACM*, 27:1134–1142, 1984.

[132] L. G. Valiant. Functionality in neural nets. In *Proceedings of the 1988 Workshop on Computational Learning Theory*, pages 28–39. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1988.

[133] K. Verbeurgt. Learning DNF under the uniform distribution in quasi-polynomial time. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 314–326. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[134] J. Vitter and J.-H. Lin. Learning in parallel. *Information and Computation*, pages 179–202, 1992.

[135] V. Vovk. Aggregating strategies. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 371–383. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[136] K. Yamanishi. A learning criterion for stochastic rules. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 67–81. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[137] T. Yokomori. Polynomial-time learning of very simple grammars from positive data. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 213–227. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991.