LEARNING DISJUNCTIONS OF CONJUNCTIONS

L.G. Valiant

Aiken Computation Laboratory
Harvard University
Cambridge, 11A 02138

ABSTRACT

The question of whether concepts expressible as disjunctions of conjunctions can be learned from examples in polynomial time is investigated. Positive results are shown for significant subclasses that allow not only propositional predicates but also some relations. The algorithms are extended so as to be provably tolerant to a certain quantifiable error rate in the examples data. It is further shown that under certain restrictions on these subclasses the learning algorithms are well suited to implementation on neural networks of threshold elements. The possible importance of disjunctions of conjunctions as a knowledge representation stems from the observations that on the one hand humans appear to like using it, and, on the other, that there is circumstantial evidence that significantly larger classes may not be learnable in polynomial time. An NP-completeness result corroborating the latter is also presented.

1. Introduction

The ability of humans to learn new concepts is remarkable and mysterious. Little progress has been made in the problem of simulating this process by computer. Worse still there is little agreement on what realistic specifications of the desired behavior of such a simulation might look like. The approach taken to this problem in a previous paper [13] was to appeal to computational complexity theory to provide some guidelines on this last question. In particular, various classes of propositional concepts were explored that could be learned from examples, or sometimes oracles, in a polynomial number of computational steps. Presumably biological systems cannot learn classes that are computationally intractable.

The overall model was hierarchical. A concept to be learned was expressed as a propositional expression in terms of concepts already known. If this expression was too complex it was computationally infeasible to learn it. The maximal complexity of such expressions that could be learned feasibly was studied.

That investigation pointed to one particular concept class that especially warranted further investigation--those that could be expressed as a disjunction of conjunctions, called disjunctive normal form (DNF). The attraction of this class is that humans appear to like it for representing knowledge as is evidenced, for example, by the success of the production system paradigm [6] and of Horn clause logics [8]. Our investigations suggest that significantly larger classes may not be learnable in polynomial time. Hence this class may turn out to have a central role.

This paper is organized as follows. In Section 2 the formal framework will be described first. It will then be shown that significant subclasses of propositional DNF expressions can be learned in polynomial time just from examples. Whether the whole class can be so learned remains an open problem.

The section following that will discuss how the above results can be extended to allow variables and relations in the manner of predicate calculus. The extension is to a very restricted part of the predicate calculus where there is only existential quantification and a limited number of variables.

In Section 4 the question as to whether the above algorithm can be made robust to errors in the data is discussed. It is shown that the learning algorithms can indeed be made robust to a small rate of even maliciously constructed errors.

The suitability of our learning algorithms to implementations on distributed "brain-like" models of computation is the subject of Section 5. In the propositional case, under certain natural restrictions, such implementations are immediate, but this does not appear to extend to the case allowing relations.

Finally in Section 6 we observe that the limits of learnability are not far off. Even if it is known that a single purely conjunctive expression explains fifty percent of the occurrences of a concept, discovering or approximating this conjunction may be NP-hard.

The word learning has been used in numerous senses in philosophy, psychology and AI. A useful distinction to make in this context is whether the available data from which the learning or induction is to be made is consistent with many varied hypothesis, or essentially with just one. In the former situation, which is often tacitly accepted in the philosophy of science, and also in machine learning

♦Supported in part by National Science Foundation grant MCS-83-02385.

(e.g. [5]), the learning problem is ill-defined because of lack of information, and the best one can hope for is to select, with good taste, from among the several plausible hypotheses. In contrast, in the latter situation, sufficient information is available in the data and the only problem is to deduce the hypothesis from it computationally. Our current work is directed at the latter situation and is motivated by the observation that distinct human individuals appear to converge remarkably to essentially the same notions about familiar concepts.

The technical idea that makes our positive results possible is the probabilistic method of specification. We assume that positive and negative examples of concepts are drawn from certain probability distributions that are determined by the real world, but are otherwise arbitrary and possibly unknown. The task of the learning procedure is to discover a recognition algorithm for the concept that gives the correct answer with high probability on further examples drawn from the same distribution.

The main assumption made by our methodology as outlined in [13] is that useful classes of learnable concepts can have simple mathematical characterizations. Although we believe that DNF expressions constitute a good and relevant knowledge representation, we don't wish to make this an equally central assumption. Alternative classes may be also worth exploring, such as those considered in [13],

As a philosophical note we observe that our probabilistic specification of learning has particular effectiveness in certain extreme empiricist or procedurist worlds. Suppose that the world is too complex to model but intelligent systems manage to cope using simple strategies (c.f. [11]). Then specifying how these strategies achieve their goals may be too complex to describe and consequently programming or discovering these strategies may be very difficult. In our model the simple strategies correspond to the recognition algorithms. The complex world is captured by the distribution, which need never be described. The important point is that the efficiency of our learning procedures is limited only by the complexity of the algorithm to be learned and not by the complexity of the distribution, which may indeed by arbitrarily complex with no ill effect. We conclude that in such worlds learning may have an unexpectedly central role that cannot be replaced by programming (c.fo Simon [12]).

2. Propositional DNF Expressions

We shall first introduce the terminology we use which is consistent with [13]. Let $\{p_1,\ldots,p_t\}$ be a set of Boolean *predicates*. Let M_0 be the set of all 3^t *monomials* (or conjuncts) over these predicates. A monomial is any product obtained by choosing for each i $(1 \leqslant i \leqslant t)$ whether p_i or $\bar{p}_i$ or neither should appear in it, and conjoining these choices. Thus if $t \geqslant 4$ then p_1p_3 and $p_1p_2\bar{p}_4$ are typical members of M_0.

Let F be a *Boolean function* over $\{p_1,\ldots,p_t\}$. More formally F is a mapping from $\{0,1\}^t \to \{0,1\}$ that specifies for each combination of truth values of the t arguments $p_1,\ldots,p_t$ of F whether F is false or true. To allow for incompletely defined inputs we shall extend F to the domain $\{0,1,*\}^t$ according to the following rule: For $v \in \{0,1,*\}^t$ $F(v) = 1$ iff $F(v') = 1$ for all $v' \in \{0,1\}^t$ such that v and v' agree on all components on which v is defined (i.e., equals 0 or 1). For example if $F(1,0,0) = 1$ and $F(1,0,1) = 1$ then we define $F(1,0,*) = 1$. If either $F(1,0,0)$ or $F(1,0,1)$ were 0 then $F(1,0,*)$ would be 0. Such an extension of a function we call the *concept* F. The criterion of truth of the concept for an incompletely defined input v is whether the defined components of v are sufficient to guarantee the truth of the associated function F.

For each concept F we define two probability distributions D^+ and D^- on its positive and negative examples. The *positive examples* are the vectors $v \in \{0,1,*\}^t$ such that $F(v) = 1$. On this set $D^-(v) = 0$ and $D^+(v) \geqslant 0$. The sum $\Sigma\, D^+(v)$ equals one when summed over this set. Similarly the *negative examples* are the vectors $v \in \{0,1,*\}^t$ such that $F(v) = 0$. On this second set $D^+(v) = 0$ and $D^-(v) \geqslant 0$. The sum $\Sigma\, D^-(v)$ equals one when summed over this set.

We define two procedures that generate positive and negative examples randomly from their respective distributions. A call of EXAMPLES^+ will generate a vector $v \in \{0,1,*\}^t$ such that $F(v) = 1$ where each such v has probability $D^+(v)$ of being produced at each call. A call of EXAMPLES^- will generate a vector v such that $F(v) = 0$ where each such v has probability $D^-(v)$ of being produced. Each call of these procedures is probabilistically independent of each other call.

A *disjunctive normal form (DNF) expression* f is an expression

$$f = \sum_{m_i \in M} m_i$$

where $M \subseteq M_0$. Thus f is the disjunction of a subset of the monomial set M_0.

Since M_0 is of size exponential in t and we will be interested in expressions that can be constructed in polynomial time, we shall need to discuss subclasses of M_0 that are of only polynomial size.

Definition. A set $M_1 \subseteq M_0$ is *polynomial generable deterministically* (p.g.d.) iff there is a deterministic algorithm that in time polynomial in t terminates and generates descriptions of all the members of M_1.

Example 2.1. For any positive integer k the class $M^{(k)}$ of monomials consisting of k or fewer predicates (negated or not) is p.g.d.

It would appear to be more flexible, and hence advantageous in practice, if the monomial set were not so constrained, *a priori*, but were to be deduced

from calls of EXAMPLES+ or EXAMPLES" for the given concept F. This is feasible if appropriate conditions hold for the distributions, such as the following:

Definition. Distributions {D+,D-} have polynomially generable monomial sets (p.g.m.s.) if there is an algorithm that can call EXAMPLES+ and EXAMPLES-, runs in time polynomial in t and h, and generates a set $M_1 \subseteq M_0$ of monomials that with probability at least $(1-h^{-1})$ has the following property:

$$\sum D^+(v) < h^{-1}$$

where summation is over every v such that $v \not\rightarrow m$ for any $m \in M_1$. [N.B. I "$v \not\rightarrow m$", s elsewhere, v and m are identified with Boolean functions in the obvious way. Thus v ->m means that the truth values defined in v make m true,]

The motivation for this last definition is to justify certain strategies for finding an M1 that would otherwise appear to be unjustified heuristics.

Example 2.2. Suppose that for some small parameter $\lambda > 0$ we called EXAMPLES+ a certain number of times and put into M1 every monomial that appears as a submonomial of the positive examples with frequency at least λ. For some distributions this set would be exponentially large. For certain others, which one could define, the set would be only polynomially large.

Example 2.3. A good practical possibility might be to let M1 be the intersection of some M(k) of Example 2.1 with the set generated in Example 2.2. For example if we choose k =3 and $\lambda = 0$ then the monomial set is simply all conjuncts of sets of up to three predicates that are ever satisfied simultaneously in some positive example.

The quantitative aspects of our learning strategies are captured by the function L(h,S): Suppose we have an urn containing a large number (possibly infinite) of marbles of at most S kinds. We want to sample a number x of marbles (with replacement) so that with probability at least 1 -h-1 the sample is such that if a further marble is picked at random then the probability that this new marble is not already represented in the previous sample is less than h-1. In [13] it is shown that there is a function L(h,S) <2h(S + loge h) that is an upper bound on this x for all possible relative frequencies of the S kinds of marbles.

Our basic results about learning DNF expressions from examples alone will be expressed with respect to our previous definitions of a concept $F(p_1,\ldots,p_t)$ having distributions D+, D- where $F = \sum m_i$ with summation over some $M \subseteq M_0$. It is essentially the dual of Theorem A in [13].

Theorem 2.1. There is an algorithm E calling EXAMPLES+ and EXAMPLES" such that (i) whether $M \subseteq M_1$ for some M1 that is p.g.d. or (ii) whether {D+,D~} has a p.g.m.s., E will generate a DNF formula g in time polynomial in h and t that will have probability at least $(1-h^{-1})$ of having each of the following properties:

(a) $$\sum_{g(v)=0} D^+(v) < h^{-1}$$

and

(b) $$\sum_{g(v)=1} D^-(v) < h^{-1} \quad .$$

[In case (i) EXAMPLES$^+$ is not necessary and the sum ΣD^+ in (a) is zero.]

Proof. Algorithm E is simply the following:

```
begin  Generate M1  [in case (i) deterministically, in case (ii) probabilistically to have the
                     desired properties];
       L ← L(h,|M1|);
       For i ← 1 to L do
       begin v ← EXAMPLES⁻;
             For all m ∈ M1 s.t. v → m
               M1 ← M1 - m;
       end
       g ← Σ m summed over m ∈ M1;
end
```

To verify (a) we note that in case (i) the monomials of F consist only of members of M1 and will always be contained in the monomial set of g. Hence $g(v) = 0 \Rightarrow F(v) = 0 \Rightarrow D^+(v) = 0$. In case (ii) some monomials of F will be missing from g because they were already missing from M1. But by the definition of p.g.m.s. the probability $\Sigma D^+(v)$ summed over all v such that $v \not\rightarrow m$ for any $m \in M_1$, is bounded by h-1 with probability at least $1-h^{-1}$. Hence $\Sigma D^+(v)$ summed over v such that g(v) =0 is bounded by the same quantity.

To show (b) suppose to the contrary that the sum $\Sigma D^-(v)$ is at least h^{-1}. Suppose we consider each $m \in M_1$ that is not a monomial of F to be a marble and that a call of v<-EXAMPLES" is interpreted as taking a sample of at least one marble, (i.e. the submonomials of v cannot be monomials of F). Then a f $t L(|M_1|,h)$ J c h calls with probability at least $(1-h^{-1})$ the set of monomials in M1 that are not in F and have not been chosen have total probability of occurrence

$$\sum_{g(v)=1} D^-(v) < h^{-1}$$

Algorithms in the style of algorithm E are natural in several contexts in inductive inference [2]. In the current context it is made noteworthy by its provably good behavior for all distributions.

3. DNF Expressions with Relations

In AI applications there is often need for referring to several distinct objects and to expressing relationships among them. For sheer

expressive power the full predicate calculus is very useful, of course, but it clearly includes much power, such as arbitrary alternations of quantifiers, that are little used. It is an interesting question to delineate the minimal part of the predicate calculus that is still expressive enough for conventional AI applications. Only a few attempts have been made in this direction [4].

Here we are concerned with the question of how much the class of expressions that is learnable can be extended to allow relations without sacrificing polynomial time learnability. We shall now allow *object variables* x1,.,.,xr. Instead of propositional predicates we will allow relation predicates P1,...,Pt each of which has a number, its arity, that defines the number of arguments it has. A *monomial* is now an existentially quantified conjunction of relational predicates, such as:

$$\exists x_1 \exists x_2 \ldots \exists x_r p_1(x_2,x_3)\bar{p}_2(x_1,x_4,x_5)p_4( )$$

For notational brevity the existential quantification will be suppressed.

This notation is sufficient to express a predicate on a scene that depends on the existence of certain subobjects in the scene that obey certain relations among themselves.

For a fixed r and a fixed set of predicates p1,...,pt we can define the class MQ of all monomials that can be formed from then in the obvious way. We regard two monomials are identical iff they can be obtained from each other by permuting the variable names.

Definition. A set $M_1 \subseteq M_0$ is polynomial generable deterministically (p.g.d.) iff there is a deterministic algorithm that in time polynomial in t and r terminates and generates all the members of M1.

Example 3.1. For any fixed numbers r and k the set of monomials consisting at most k predicates, negated or not, is p.g.d. In fact there are no more than (2r+1t)k of them0

In treating the probabilistic generation of monomials from examples a problem would arise if we had to relate the variables {xi} to the primitive input vectors {v}. In the case under consideration, however, this whole problem can be completely finessed. Let D+, D- be probability distributions on the positive and negative examples of the primitive input vectors {v}. We extend D+, D- to the domain of monomials Mo as follows: D+(m) is the probability that the conjunctive relation m holds for a random output v of EXAMPLES+. Similarly for D-.

Definition. Distributions {D+,D-} have polynomially generably monomial sets (p.g.m.s.) if there is an algorithm calling EXAMPLES+ and EXAMPLES" that runs in time polynomial in h, t and r and generates a set $M_1 \subseteq M_0$ of monomials that with probability at least- $(1"-h^{-1})$ has the following property:

$$\sum D^+(v) < h^{-1}$$

where summation is over every v such that $v \not\Rightarrow m$ for any $m \in M-\hat{}$ [i.e., as in the propositional case the monomials of F missing from M1 account for a small fraction of the probability space {v}. Now, however, $v \Rightarrow m$ has the complex meaning that the monomial expression m is true of primitive input v.].

Examples 3.2 and 3.3. The analogues of Examples 2.2 and 2.3 can be defined in the obvious way.

Theorem 3.1. The statement and proof are identical to those of Theorem 2.1 except that the monomial set M1. has the more general interpretation allowing relations. o

In summary, the learning algorithm we are suggesting, if we use Example 3.3 with $\lambda = 0$ as the monomial set, is the following: List all monomials with up to r object variables and k predicates, that hold in at least one observed positive example. Eliminate from this list all monomials that occur in some observed negative example. Take the disjunction of the remaining monomials as the desired expression.

4. Robustness

How much tolerance to erroneous data should we require of a learning system? On the one hand experience suggests that learning is difficult enough from error-free data and becomes well nigh impossible if the data is seriously flawed. This suggests the possibility that the learning phenomenon is only feasible with very low error rates. On the other hand some robustness is clearly essential and it is reasonable to investigate the maximum error rate that can be compensated for.

In this section we will modify Algorithm E and show that the modified form Algorithm E*, is resistant to a quantifiable, if low, error rate, whether in the propositional or extended relational case of Section 3.

Definition. If A is an oracle then we say that oracle B is "oracle A with error rate δ" if at each call of B (i) with probability $1-\delta$ it calls oracle A correctly, and (ii) with probability δ it produces an arbitrary answer possibly chosen maliciously by an adversary.

The modified algorithm that has access only to an oracle "EXAMPLES" with error rate $\delta(t,h)$" will now perform $T(|M_1|,h)$ cycles and will depend on a parameter $\varepsilon(t,h)$, where δ,ε and T are related below. 2 is an integer array indexed by the members of M1 and initialized to zero. Algorithm E* is the following:

```
begin  Generate M1;  T ← T(|M1|,h);
       For i ← 1 to T do
       begin  v ← "EXAMPLES" with error δ";
              For all m ∈ M s.t. v ⇒ m
                Z[m] ← Z[m] + 1;
       end
```

g ← Σ m with summation over all m s.t. Z[m] < εT;

end

The following analogue to Theorems 2.1 and 3.1 can be proved. We shall restrict ourselves to the deterministic case. The other case, when *M1* has p.g.m.s., can be treated similarly by detecting any maliciously generated monomials having too many submonomials by their low frequency of occurrence. Thus the result refers to an arbitrary function F that can be expressed as the sum of monomials from M1.

Theorem 4.1. There is an algorithm E* calling EXAMPLES" with error rate $\delta = (4h|M_1|)^{-1}$ that will generate a DNF formula g in time polynomial in h and t such that with probability at least $(1 - h^{-1})$ g has the following properties

(a) $$\sum_{g(v)=0} D^+(v) = 0$$

and

(b) $$\sum_{g(v)=1} D^-(v) < h^{-1}$$

Proof. We will show that the described algorithm E* suffices if $\varepsilon = 2\delta$ and $T = 36h|M_1|\log(|M_1|h)$.

(a) A monomial m of F can be missing from g only because it has been removed from M1 by having occurred with frequency greater than E in the negative examples. This can only happen if the oracle behaved erroneously at least ET times in T trials where the probability of each such occurrence is at most $\delta = \varepsilon/2$. By the binomial distribution (e.g. [3]) the probability of this happening is less than $\exp(-1/2 \cdot \varepsilon T) < \exp(-4 \log(|M_1|h)) < |M_1|h^{-1}$ under the assumed bounds on T and δ. Hence the probability of this happening to any m in F is at most h-1.

(b) A monomial m such that $D^-(v) \geq 0$ for some v such that m =>v can be included in g if there have been fewer than £T negative examples containing m. Now if some m €M1 is implied with frequency at least 2ε by D- then the probability of having found at most εT negative examples in T trials each with probability of success at least $2\varepsilon(1 - \delta)$ [i.e., allowing for the 6 error rate] can be bounded above by $\exp(-(1/3) \cdot 4\delta T(1-\delta) \cdot [(1-\delta)/(2-2\delta)]^2) < \exp(-\delta T/9)$. Hence the probability that at least one such monomial with frequency at least 2e has been wrongly included in g is at most $M_1 \cdot \exp(-\delta T/9) \leq h^{-1}$ under the assumed bounds. Hence if only those monomials remain that have frequency less than 2ε then their total contribution to $\Sigma D^-(v)$ is at most $|M_1| \cdot (2h|M_1|)^{-1} \leq h^{-1}$. D

5. Naive Neural Modelling

The methodology behind this study is that of determining the largest classes of "concepts", that can be learned in a feasible total number of computation steps. Since learning is exhibited by biological systems it is an interesting afterthought to determine whether the algorithms discovered happen to be well suited to implementation on a "brain-like" model of computation, such as considered by Feldman [6] or Ackley *et al. [1]*, for example.

In this section we observe that a partial implementation of our algorithm on networks of linear threshold devices can be found that is noteworthy for its simplicity. In the propositional case it is able to learn DNF expressions if the distributions are such that the monomial sets are generable in a certain sense on these networks. While this class is still plausible and accommodates robustness exactly as in Section 4, it does require more restricted distributions than before and does not allow for the relations of Section 3.

The essential idea is that if the input nodes of the network are exposed to the vectors constituting the examples, then the monomials in the monomial set M1 will be learned in unsupervised fashion at various distinct nodes in the network. The final disjunction over these is learned in supervised mode at a specified node. To accomplish the unsupervised stage symmetry in the network is broken by assuming randomness in the network (either of the connections or of the weights).

We consider the following distributed model of computation. An instance of a net is described by a triple (V,E,A) where V is a set of n nodes {1,2,...,n), $E \subseteq V \times V$ is a set of directed edges and A:E --> [0,1] labels each edge (i,j) with a numerical weight aij. Each node i has a state s. that at any time has value 0 or 1.

A set of nodes, say {l,...,t} is regarded as constituting the input. When an example is presented to the net, for learning or recognition, the states s1,...,st are set to the Boolean values corresponding to the truth values of the propositional predicates p ,...,p ,

A node j "fires" (i.e., sj =1) in a time interval if in the previous interval

$$\sum_{(i,j) \in E} a_{ij} s_i \geq 1 \quad .$$

A node continues to fire as long as the above condition holds.

The learning mechanism allowed is the following: When a new example is presented we wait for the states to stabilize and assume that they do. A weight aij can be changed only if node i or node j is firing. Its new value depends only on the current values of si Sj and on the values of aij and aji before the presentation of the current example. The new value of aij becomes operative only when a further example has been presented.

We consider two senses in which a net learns a function F(p1,...,pt) as a consequence of a training session of examples. F has been learned in *unsupervised* mode if the weights have been so modified during the training session that some node j now fires whenever (i.e. almost whenever) F(s1...,st) - 1 . F has been learned in *supervised* mode if the same has been achieved but in the course

of the training session a distinguished node, say j was controlled by an external omnipotent agent and made to fire if and only if $F(S1,\ldots,st) = 1$ was true of the training example (thus overriding possibly the threshold condition above).

For implementing counters on the weights we use an arbitrary continuous strictly monotone function $\sigma[0,1] \to [0,1]$ such that for any $c \in (0,1)$ $\sigma^r(c) \to 1$ and $\sigma^{-r}(c) \to 0$ as $r \to \infty$. [e.g., $\sigma(x) = 2x - x^2$, $\sigma^{-1}(x) = 1 - \sqrt{1-x}$ would do].

First, we notice that if F is a simple monotone disjunction (e.g., p1 vp3 vp7) or a simple monotone conjunction then it can be learned in supervised mode at any node j to which all the relevant inputs are connected. The learning rule for disjunctions involves changing those a_{ij} where $(i,j) \in E$ provided $F=0$ as follows:

$$a_{ij} = \begin{cases} \sigma(a_{ij}) & \text{if } F=0 \text{ and } s_i = 0 \\ \sigma^{-\alpha}(a_{ij}) & \text{if } F=0 \text{ and } s_i = 1 \end{cases}.$$

By duality learning a conjunction is achieved by

$$a_{ij} = \begin{cases} \sigma(a_{ij}) & \text{if } F=1 \text{ and } s_i = 1 \\ \sigma^{-\alpha}(a_{ij}) & \text{if } F=1 \text{ and } s_i = 0 \end{cases}.$$

In both cases $\alpha + 1$ is the inverse of the $\varepsilon = 2\delta$ of Theorem 4.1. If each aij is initialized to 1/2, say, then each a_{ij} will tend to 0 or 1 as desired according to wherner its frequency in the corresponding examples is less than or greater than ε.

To recognize disjunctions of conjunctions we will show how certain monomial sets M1 can be generated in unsupervised mode from examples. A disjunction over any subset of *M1* can then be learned by the above method at any node that is connected to all the nodes computing M1 .

The basic idea is illustrated by the simple case of wanting to learn the monomial P1P2 . . . Pk from a distribution D* of examples when the nodes $1,2,\ldots,k,\ldots,d$, are all connected to some node j >d„ [Note that here D* will be a distribution that typically includes both positive and negative examples of F and no-one informs node j as to which is the case.] To accomplish this we define a as above but scaled to the range [0,(k-1/2)1], and choose it so that $|\sigma(x)-x|$ is always suitably small compared with k-1. If each aik is initialized to $(k-1/4)^{-1}$ then the following learning algorithm, for $1 \leq i \leq d$, will achieve the task

$$a_{ij} = \begin{cases} \sigma(a_{ij}) & \text{if } s_j = 1 \text{ and } s_i = 1 \\ \sigma^{-1}(a_{ij}) & \text{if } s_j = 1 \text{ and } s_i = 0 \end{cases}$$

provided the distribution D* has favorable properties such as the following:

If Y is the event that at least k of $s_1,\ldots,s_d$ are 1 then

$\text{Prob}(s_1=s_2=\ldots=s_k=1 \mid Y) > \frac{1}{2} + \varepsilon$ and

$\text{Prob}(s_\ell=1 \mid Y) < \frac{1}{2} - \varepsilon$ for any ℓ $(k < \ell \leq d)$.

The idea is that for the majority of inputs $s_\ell = 1$ for $\ell < k$. These will cause s- to fire and hence increase the values of $a_{\ell j}$ to tend towards $(k-1/2)^{-1}$. For the remaining values of ℓ $s_\ell = 0$ for the majority of inputs and hence alj will tend to zero as desired.

To make arbitrary monomials of length k (i.e., M(K)) learnable in unsupervised mode, one approach is to make the network random so that for each set of k out of the n nodes, there will be at least one, and at most a few, nodes to which all k are connected. Connecting each node to a random set of about $n^{1-1/k}$ others achieves this. Then any monomial $m \in M^{(k)}$ can be learnt at some node, and, equally importantly, will be learnt at only a few, thus allowing the other nodes to learn different monomials. In this way about n distinct monomials from M(K) can be learnt. To make all this possible D* has to be so defined that at each node at most one monomial from M(K) has overriding influence in the sense of the previous paragraph.

To make the above concrete consider the case k = 2 and suppose that we want P1P2 to be learnable. Then D* has to be such that if we pick a set of $\sqrt{n}$ Pi's to include P1/P2 but to be otherwise random, then with high likelihood the following holds: If v €D* is chosen randomly to have at least two of the chosen pi true, then with probability greater than a half these will include p1 and P2*

To summarize we have shown that a restricted version of Algorithm E of Section 2 can be implemented on a network of threshold elements. It will build up recognizers for a set M* of monomials that occur in examples provided the members of M* are suitably separated by D* in the manner indicated above. The network can then learn arbitrary disjunctions over M* in supervised mode.

6. Simple Rules of Thumb May be Hard to Learn

There is evidence that certain significant extensions to the concept classes here or in [13] render them computationally intractable. One source of evidence is cryptography as described in the latter paper. Here we shall describe an NP-completeness result that shows that extensions in a certain other direction are probably computationally intractable also.

In the learnable classes considered a simple formula was assumed to account for a hundred percent of all positive examples of a concept. Suppose that the simple formula is now a single monomial that accounts for just fifty percent, and the rest is accounted for by a function whose complexity is unspecified. We shall show that in this situation it is NP-hard to discover either this monomial, or

one that approximates it in the sense of accounting for $(1/2 - h^{-1})$ of the distribution for arbitrarily large h.

Theorem 6.1, Given a function F expressed as a sum of monomials and distributions D+ and D- of positive and negative examples, it is NP-hard to determine whether a monomial m of the following kind exists, or to construct one if it does:

$$\sum_{v \Rightarrow m} D^+(v) \geq \frac{1}{2} \quad \text{and} \quad \sum_{v \Rightarrow m} D^-(v) = 0$$

[The descriptions of D and D do not need to be counted in the input size.]

Proof, We reduce the BALANCED COMPLETE BIPARTITE SUBGRAPH problem proved NP-complete by Garey and Johnson ([7], p. 196) to this problem. In the former problem we are given a bipartite graph $G = (V = V_1 \cup V_2, E \subseteq V_1 \times V_2)$ where $|V_1| = |V_2|$ and we have to determine whether there are subsets $U_1 \subseteq V_1$ and $U_2 \subseteq V_2$ both of size $|V_1|/2$ such that $U_1 \times U_2 \subseteq E$.

Suppose we interpret the nodes V_1 as $\{p_1, \ldots, p_t\}$ and define for each node $q_i \in \{q_1, \ldots, q_t\} = V_2$ a monomial m_i such that $m_i = \Pi p_j$ with conjunction over all j such that $(j,i) \in E$. Let $D^+(v_i) = t^{-1}$ for each v_i that is completely defined and has exactly those predicates true that occur in m_i. Also let $D^-(v) > 0$ whenever fewer than t/2 predicates are true.

Then the distribution ensures that any m satisfying the theorem must have at least t/2 predicates defined as true, and, also, must imply at least t/2 of the mi's. But this is exactly the condition that the graph G has a (t/2) x (t/2) complete bipartite subgraph.

The above shows that determining whether the desired m exists or not is NP-hard. Constructing it on the assumption that this good rule of thumb exists is therefore also NP-hard since the decision procedure reduces to this. o

Also if no such m exists then the value of $\Sigma D^+(v)$ summed over all $v \Rightarrow m$ is at most $(1-t^{-1})/2$ and the constant 1/2 cannot be approached to within h-1 uniformly in a number of steps polynomial in h and t (unless NP-completeness does not imply intractability).

Finally note that it is not known whether the size of the largest balanced complete bipartite subgraph can be approximated to arbitrarily good multiplicative constant factors in polynomial time. Hence it is possible that the above reduction establishes a stronger result than claimed, namely that finding monomials accounting for $(1/2 - \varepsilon)$ of the distribution for sufficiently small constant e is also difficult.

REFERENCES

[1] D.H. Ackley, G.E. Hinton, and T.J. Sejnowski. "A Learning Algorithm for Bolzmann Machines'.', Cognitive Science 9;1 (1985) 147-169.

[2] D. Angluin and C.H. Smith. "Inductive Inference: Theory and Methods", Computing Surveys 15:3 (1983) 237-269.

[3] D. Angluin and L.G. Valiant. "Probabilistic Algorithms for Hamiltonian Circuits and Matchings", J. Comp. Syst. Sci. 18:2 (1979) 155-193.

[4] R.J. Brachman and H.J. Levesque. "The Tractability of Subsumption in Frame-based Description Languages", Proc. AAAI-84, Austin, Texas (1984) 34-37.

[5] T.G. Dietterich and R.S. Michalski. "A Comparative Review of Selected Methods for Learning from Examples'.', in [9], pp. 41-81.

[6] J.A. Feldman and D.H. Ballard. "Connectionist Models and Their Properties", Cognitive Science 6 (1982) 205-254.

[7] M.R. Garey and D.S. Johnson. Computers and Intractability, Freeman Press, San Francisco (1979).

[8] R. Kowalski. Logic for Problem Solving. North Holland, New York (1979).

[9] R.S. Michalski, J.G. Carbonell, and T.M. Mitchell. Machine Learning. Tioga Press, Palo Alto, CA (1983).

[10] A. Newell and H.A. Simon. Human Problem Solving. Prentice Hall, Englewood Cliffs, NJ (1972).

[11] W.E. Reichardt and T. Poggio. "Visual Control of Flight in Flies", In Recent Theoretical Developments in Neurobiology. W.E. Reichardt, V.B. Mountcastle and T. Poggio (eds.) (1979).

[12] H.A. Simon. "Why Should Machines Learn", In 19], pp. 25-37.

[13] L.G. Valiant. "A Theory of the Learnable", CACM 27:11 (1984) 1134-1142.