Machine Learning, 37, 115–130 (1999)

Projection Learning*

LESLIE G. VALIANT valiant@deas.harvard.edu
Division of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138, USA

Editor: Lisa Hellerstein

Abstract. A method of combining learning algorithms is described that preserves attribute-efficiency. It yields learning algorithms that require a number of examples that is polynomial in the number of relevant variables and logarithmic in the number of irrelevant ones. The algorithms are simple to implement and realizable on networks with a number of nodes linear in the total number of variables. They include generalizations of Littlestone's Winnow algorithm, and are, therefore, good candidates for experimentation on domains having very large numbers of attributes but where nonlinear hypotheses are sought.

Keywords: computational learning, attribute-efficient learning, irrelevant attributes, Winnow algorithm

1. Introduction

One area of stark contrast between current machine learning practice and natural biological learning is that of data preparation. In machine learning it is often found that in order to ensure success at a new learning task considerable effort has to be put into creating the right set of variables, and into eliminating redundant ones if there are large numbers of these. In biological learning, on the other hand, no explicit methods for achieving these ends have been identified. The learning process appears to have mechanisms for overcoming these problems implicitly.

This dichotomy suggests the existence of a useful class of learning algorithms that have yet to be discovered and exploited. Indeed it has been suggested that algorithms that fill this gap are fundamental to systems that learn and maintain large knowledge bases for cognitive computations (Valiant, 1998).

A simple but persuasive formulation of the problem at hand is the following: one needs algorithms that learn, as efficiently as possible, functions that depend on a small number k among a much larger number n of available variables. The necessity for this is suggested by the empirical observation that humans and other biological systems can learn from a small number of examples, using a system that has many components (i.e. $\sim 10^{10}$ neurons), and in which each component has a large number of potentially independent parameters (e.g $\sim 10^4$ synapses). On dimensionality grounds it appears that the only hope of learning from a small number of examples is to have the concept learned have small dimensionality in some sense. Perhaps the simplest expression of this restriction is to have each concept completely

*This research was supported in part by grants from the National Science Foundation, NSF-CCR-95-04436, Office of Naval Research ONR-N00014-96-1-0550 and Army Research Office ARO-DAAL-03-92-G-0115.

determined by the values of a small number of the variables. The question therefore is: just how efficiently can a function of k variables be learned in the presence of a possibly much larger number of $n - k$ irrelevant variables, when the identity of the k relevant ones is not known ahead of time.

Computational learning theory has supplied some remarkable positive results. Haussler (1988) showed that for the problem of learning a Boolean disjunction or conjunction of k variables out of n, the number of examples needed to learn in polynomial time grew essentially as $k \log n$. The fact that the dependence on n had been reduced to logarithmic, rather than linear, represented a major breakthrough. Soon afterwards Littlestone proved a similar result for a surprisingly elegant algorithm which he called Winnow (Littlestone, 1988). It resembles the perceptron algorithm in simplicity and form, but achieves its attribute-efficient behavior by having multiplicative rather than additive updates. The algorithm has been shown to be effective, for example, in natural language settings with tens of thousands of variables (Golding & Roth, 1996).

The primary purpose of this paper is to extend the known classes of algorithms that learn in this attribute-efficient sense. We do this by showing that attribute-efficient learnability can be preserved by composing algorithms that have this property in certain natural and simple ways. This result applies directly to learning certain classes of two level structures. These classes can be learned attribute-efficiently by a natural two level composition of attribute-efficient algorithms such as Winnow. The number of mistakes made by the new algorithm on any sequence of examples still depends on n only logarithmically and on k polynomially. The dependence of the computational cost on n is essentially linear and no separate transformation of the input space is needed.

The dual purpose of the paper, which we only mention here is passing but will expound more fully elsewhere, is to show that attribute-efficiency can be achieved by algorithms that satisfy the following additional constraints that have been argued to be useful for realizing the neuroidal architecture described in (Valiant, 1998). First, it is considered desirable that such algorithms be realizable on networks of elements that can each learn linear threshold functions and have additional finite state computational capabilities. Furthermore, for economy the numbers of such elements should be linear in the number of variables being represented, rather than, say, quadratic. Second, the structures that can be learned should fill some need of cognitive interest. Third, the structures should supply some welcome functionality in learning and handling relational information rather than merely propositional information, in a network setting (Valiant, 1999). The knowledge structures we describe aim to satisfy these constraints. They can be viewed as attribute-efficient alternatives to trees or production systems. The latter have been widely used to represent cognitive knowledge. Khardon (1996) has used them recently to learn action strategies. It is an open empirical question to determine to what extent the sequentiality that is implicit in general production systems can be replaced in cognitive computations by the flatter structures that we consider here.

2. Concept classes

We shall consider Boolean functions of Boolean variables $x_1, \ldots, x_n$. A *concept* will be a function from $X_n = \{0, 1\}^n$ to $\{0, 1\}$ for some finite n. We shall discuss classes of concepts,

such as the class of Boolean disjunctions. We represent such a class C as the union of stratified subclasses C_n,

$$C = \bigcup_{n \geq 1} C_n$$

where $c \in C_n$ has domain X_n as described, for example, by Kearns and Vazirani (1994).

A *projection* (or restriction) ρ of X_n is a subset of X_n, and is typically represented by a simple constraint such as $x_2 = 1$ or $x_3 = \bar{x}_4$. For a function $f : X_n \to \{0, 1\}$ the *restriction* f_ρ of f is defined as: $f_\rho(\underline{x}) = f(\underline{x})$ if $\rho(\underline{x}) = 1$, and $f_\rho(\underline{x}) = 0$ otherwise. Note that $f_\rho(\underline{x})$ can be written equivalently as $\rho(\underline{x}) f(\underline{x})$. A *projection set* R for X_n is a set of r such projections.

Projection sets are intended here to be easily enumerated sets of restrictions. For example, we could take the set of all projections that fix the product of a pair of distinct variables to 0 or 1. In that case the size r of this set would be $2 \cdot \binom{n}{2}$.

The concept classes that we investigate in this paper are defined with respect to arbitrary choices of both C and R. A *projective disjunction over* (C, R) is a function c of the form

$$c(\underline{x}) = \rho_1(\underline{x})c_1(\underline{x}) \vee \rho_2(\underline{x})c_2(\underline{x}) \vee \cdots \vee \rho_m(\underline{x})c_m(\underline{x})$$

where $\rho_1, \ldots, \rho_m \in R, c_1, \ldots, c_m \in C$ and for each $i (1 \leq i \leq m) \rho_i c = \rho_i c_i$. In other words c and c_i agree on every $\underline{x}$ such that $\rho_i(\underline{x}) = 1$. Note that the novel constraint is this last one that excludes the possibility for any $\underline{x}$ that $\rho_i(\underline{x}) = c(\underline{x}) = 1, c_i(\underline{x}) = 0$, and $c_j(\underline{x}) = \rho_j(\underline{x}) = 1$ for some $j \neq i$.

In the case that members of R are mutually exclusive, (i.e. they map disjoint subsets of X_n to 1), this constraint is satisfied automatically. For example, one could choose R to be the 2^ℓ projections defined by setting $x_1, \ldots, x_\ell$ to the 2^ℓ distinct combinations of truth values. Disjointness is not essential. One could choose R to be the set of $2n$ *single-variable projections* $\{x_i = 1, x_i = 0 \mid 1 \leq i \leq n\}$, or the set of quadratic projections $\{x_i x_j = 1 \mid 1 \leq i < j \leq n\}$. In all these cases if C is further chosen to be the set of conjunctions then the class defined is a subclass of the class of disjunctive normal form formulae. More generally, C will be chosen to be any class of functions learnable attribute-efficiently, such as the class of linear threshold functions with moderate size integer coefficients (Littlestone, 1988).

The significance of projective disjunctions is that they suggest the following natural learning strategy. For each $\rho \in R$ one applies a learning algorithm for C to the input stream of examples restricted to those satisfying ρ and obtains a hypothesis h_ρ. For each ρ that is relevant to C, a useful hypothesis h_ρ will be learned. An appropriate disjunction constructed from these will then approximate c.

We can also generalize this definition by making the same claims for an arbitrary subset $X'_n \subset X_n = \{0, 1\}^n$ rather than X_n itself. This is relevant if some of the vectors never occur as examples and c need not be defined for them. If this more general situation is acknowledged then simpler expressions sometimes suffice. For example, if $\rho_i \equiv (x_i = 1)$, then

$$\rho_1 \bar{x}_2 x_3 + \rho_2 \bar{x}_1 x_4 \equiv x_1 \bar{x}_2 x_3 + x_2 \bar{x}_1 x_4$$

is a projective disjunction. However, if no examples with $x_1 = x_2 = 1$ occur in X'_n then the expression

$$x_1x_3 + x_2x_4$$

suffices. In order to put this in a formal setting we wold define a *projective disjunction over* (C, R) *for* X'_n as we did above for the case $X'_n = X_n$, but would relax it so that $\rho_i c = \rho_i c_i$ needs to be satisfied for each i only for $\underline{x} \in X'_n$. The remainder of this paper applies to this generalized notion also.

The motivation in the cognitive setting of projective disjunctions may be characterized by the following instance: A concept may be too complex to be learned directly by Winnow. However, it may be that the concept occurs mainly in, say, just three contexts and when restricted to any one of these the concept is simple enough to be learned attribute efficiently by Winnow. Our results will state that as long as the three contexts belong to a set of contexts that can be recognized, and even if we do not know the identity of the three within the set ahead of time, the overall concept can be learned attribute efficiently nevertheless.

3. Mistake-bounded learning

We shall show for various algorithms that they learn attribute-efficiently. In particular, on any sequence of examples the number of misclassifications made depends on the number of irrelevant attributes only logarithmicly. Our results are established in the mistake-bounded model, but these can be translated to the PAC model in various standard ways (Littlestone, 1988).

Consider a learning algorithm A that is given an arbitrary sequence of examples $\underline{x}^1, \underline{x}^2, \ldots$ each a member of X_n. Suppose for some Boolean concept class C that there is a concept $c \in C_n$ that classifies the examples as $c(\underline{x}^i) \in \{0, 1\}$. An algorithm A is *online* if it behaves as follows: It maintains a hypothesis $h : X_n \to \{0, 1\}$ that is updated after each example: On seeing $\underline{x}^i$ the algorithm evaluates $h(\underline{x}^i)$. If this does not equal $c(x^i)$ then one mistake has been made. Before seeing $\underline{x}^{i+1}$, h is updated in light of the following new pieces of information: $\underline{x}^i$ and $c(\underline{x}^i)$ (as well as $h(\underline{x}^i)$ which can be deduced from $\underline{x}^i$.) We say that M is a *mistake bound* for A if for any, possibly infinitely long, sequence of examples, A never makes more than M mistakes.

Our algorithms will combine several mistake-bounded algorithms so that the output of one becomes an input for another. We shall, therefore, need to consider the more complex case that the labelling $c^*(\underline{x}^i)$ provided to A is false and not equal to $c(\underline{x}^i)$. Furthermore, we need to distinguish between false positives (i.e. $h = 1, c = 0$) and false negatives (i.e. $h = 0, c = 1$), in the case that the labels are correct.

We say for $c \in C$ on subdomain $X'_n \subseteq X_n$ that it has k *relevant* variables if there is some set $\{x_{i_1}, \ldots, x_{i_k}\}$ of k variables such that for all $\underline{x}', \underline{x}'' \in X'_n$, it is the case that $c(\underline{x}') = c(\underline{x}'')$ whenever $\underline{x}', \underline{x}''$ agree on $x_{i_1}, \ldots, x_{i_k}$. If this holds then $\{x_{i_1}, \ldots, x_{i_k}\}$ is called a *decisive* set for c on X'_n. A variable x_j is called *decisive* with respect to a *specified* decisive set if it belongs to that set. We are interested in small or minimal size decisive sets. A function on a given subdomain may have several minimal size decisive sets. (We note that some

definitions of relevance in the literature imply that if there are k relevant variables then all the remaining ones have no influence on the function. Littlestone's Winnow algorithm, however, supports our stronger definition.)

Finally, we wish to express the number *fp* of false positives and the number *fn* of false negatives that algorithm A makes over any sequence of examples, for any $c \in C$ and for any $X'_n \subseteq X_n$, in terms of four parameters: the total number n of variables, the minimum number k of relevant variables of c for X'_n the number *fpl* of examples in the sequence falsely labelled as positive (i.e. $c^* = 1, c = 0$) and the number *fnl* of examples falsely labelled as negative (i.e. $c^* = 0, c = 1$). Hence we shall write the number of false positives and false negatives for A as $fp = \pi(n; k; fpl; fnl)$ and $fn = \nu(n; k; fpl; fnl)$.

In the particular case of learning monotone Boolean disjunctions it will yield better results to consider the related functions π', ν' in which the last parameter is $fnlm_S$, the *number of false negative labels with multiplicity relative to the decisive set S*. An $\underline{x}$ for which $c^*(\underline{x}) = 0$ but $x_i = 1$ for $t \geq 1$ distinct variables $x_i \in S$ (and hence $c(\underline{x}) = 1$), contributes t to $fnlm_S$ (but contributes just 1 to *fnl*.) The dual of this measure, used for learning conjunctions, is $fplm_S$ that counts the number of occurrences of $x_i = 0$ for $x_i \in S$ in examples with $c(\underline{x}) = 0$ and $c^*(\underline{x}) = 1$.

We shall adopt the convention that when an algorithm is presented with a false labelling for an example $\underline{x}$, its classification of $\underline{x}$ according to $h(\underline{x})$ will not be counted as contributing to π, ν, π' or ν'.

The paradigmatic known attribute-efficient algorithm is Littlestone's Winnow. We shall use the following instance of it, which we shall call Algorithm W: For domain $X'_n \subseteq X_n$, the hypothesis maintained after each example is "$\sum_{i=1}^{n} w_i x_i > n$". Thus $h(\underline{x}) = 1$ if $\sum w_i x_i > n$, and $h(\underline{x}) = 0$ otherwise. Initially $w_1 = w_2 = \cdots w_n = 2$. On each example $\underline{x}$: (i) if $h(\underline{x}) = c^*(\underline{x})$ no change is made to h, (ii) if $h(\underline{x}) = 1$ and $c^*(\underline{x}) = 0$ then for all i s.t. $x_i = 1$ in $\underline{x} : w_i \leftarrow w_i/2$, and (iii) if $h(\underline{x}) = 0$ and $c^*(\underline{x}) = 1$ then for all i s.t. $x_i = 1$ in $\underline{x} : w_i \leftarrow 2w_i$. An update (ii) is called a *demotion* and an update (iii) is called a *promotion*.

The following can be deduced, by an adaptation of Littlestone's analysis, for the class C of monotone disjunctions (i.e. $c = x_{i_1} \vee \cdots \vee x_{i_k}$).

Theorem 3.1. *For the class of monotone disjunctions Algorithm W has the following mistake bounds for any subdomain of* $\{0, 1\}^n$:

$$\pi'(n; k; fpl; fnlm_S) \leq 2(k \log_2 n + fnlm_S + fpl + 1),$$

and

$$\nu'(n; k; fpl; fnlm_S) \leq k \log_2 n + fnlm_S.$$

where S is any decisive set for the disjunction being learned for that subdomain, and where the values of every $x_i \notin S$ *in the examples may be arbitrarily corrupted.*

Proof: First we note that a promotion occurs only when $h(\underline{x}) = 0$ and $c^*(\underline{x}) = 1$. This can happen only if $c(\underline{x}) = 1$ and $\underline{x}$ was a false negative, or alternatively if $c(\underline{x}) = 0$ and $\underline{x}$

was falsely labelled positive. Hence

$$\#promotions = fn + fpl. \tag{1}$$

To be precise, in this proof fpl, fnl and $fnlm_S$ are measures of just those false labellings that give rise to updates, ignoring those that cause no updates. The bounds claimed in the Theorem will also hold, clearly, if all the false labellings are included in the counts.

Suppose without loss of generality that $c \equiv x_1 \vee x_2 \vee \cdots \vee x_k$ on X'_n so that $\{x_1, \ldots, x_k\}$ is a decisive set. First we observe that no coefficient w_i ever exceeds $2n$ in value since if it did it must have achieved such a value by doubling some value exceeding n for an input with $x_i = 1$ that caused a promotion. But if $w_i > n$ and $x_i = 1$ then $h(\underline{x}) = 1$, since $w_i > 0$ for all j, and promotions are only possible if $h(\underline{x}) = 0$. Next we observe that for a decisive set S and for $x_i \in S$, w_i halves only if $\underline{x}$ is falsely negatively labelled and $x_i = 1$. This is because if $x_i = 1$ a demotion occurs if $h = 1$ and $c^* = 0$. The possibility that $c = 0$ is inconsistent with a decisive $x_i = 1$, hence it must be that $c = 1$ and hence that $\underline{x}$ is falsely negatively labelled. We are now in a position to upper bound the number of promotions caused by false negatives by using these constraints on the values of w_i for $x_i \in S$. Each false negative causes at least one decisive w_i to double. Since each $w_i = 2$ initially, and at most $2n$ finally, this permits $k \log_2 n$ such promotions. Each contribution to $fnlm_S$ causes exactly one decisive w_i to halve, and therefore each one permits potentially one further promotion on false negative. Also, each false positive label may cause a promotion but has no influence on w_i. Hence

$$\#promotions \leq k\lfloor \log_2 n \rfloor + fnlm_S + fpl. \tag{2}$$

Combining (1) and (2) gives

$$fn \leq k\lfloor \log_2 n \rfloor + fnlm_S. \tag{3}$$

Now a demotion can occur only when $\sum w_i x_i > n$. Since for each i such that $x_i = 1$, w_i will be halved, it follows that $\sum w_i$ will be reduced by more than $n/2$. On the other hand, a promotion occurs only if $\sum w_i x_i \leq n$. Since for every i such that $x_i = 1$, w_i will be doubled, it follows that $\sum w_i$ can never be as small as $n/2$ since if $\sum w_i \leq n$ no demotion can occur and hence the lowest value reachable exceeds $n/2$. Hence, counting the increases and decreases in $\sum w_i$ in units of $n/2$,

$$\#demotions \leq 2 \cdot \#promotions + 2. \tag{4}$$

since otherwise the value of $\sum w_i$ would be below $n/2$.

Finally we note that by the definitions

$$\#demotions = fp + fnl. \tag{5}$$

Combining (2), (4) and (5) gives

$$fp \leq 2k\lfloor \log_2 n \rfloor + 2\, fnlm_S + 2\, fpl + 2 - fnl. \tag{6}$$

□

We note that by Boolean duality Theorem 3.1 can be adapted to learning conjunctions. Thus in order to learn a conjunction c over $x_1, \ldots, x_n$ we learn a disjunction c' over $\bar{x}_1, \ldots, \bar{x}_n$ that equals the negation of c, and negate the learned function. Thus the bounds of Theorem 3.1 apply for the dual false positive and false negative counts π'' and ν'', that take label parameters $fplm_S$ and fnl. Also we need to interchange positive and negative everywhere.

Corollary 3.2. *For the class of monotone conjunctions the dual of Algorithm W has the following mistake bounds where S is any decisive set*:

$$\pi''(n; k; fplm_S; fnl) \leq k \log_2 n + fplm_S,$$

and

$$\nu''(n; k; fplm_S; fnl) \leq 2(k \log_2 n + fplm_S + fnl + 1). \qquad \square$$

We note that these algorithms can be made to learn conjunctions and disjunctions in which negated and unnegated variables may occur arbitrarily, by, for example, introducing separate variables for x_i and its negation $\bar{x}_i$ for each i. This will double the number of irrelevant variables but not increase the number of relevant ones (Littlestone, 1988). Also, the algorithm itself can be tuned in various ways by modifying the update function. For example, it can be verified that if in Algorithm W for conjunctions the promotions are replaced by $w_i \leftarrow (1+q^{-1})w_i$ and demotions are unchanged, then the mistakes made will be a factor of q times less sensitive to false positive labels, but q times more sensitive to false negative labels.

4. A projection learning algorithm

We first consider learning in a domain X_n over (C, R) where C is a concept class learnable by an algorithm A with mistake bounds π_A and ν_A, and R is a projection set.

As defined in Section 2, a disjunction over (C, R) can be expressed as

$$c(\underline{x}) = \rho_1(\underline{x})c_1(\underline{x}) \vee \rho_2(\underline{x})c_2(\underline{x}) \vee \cdots \vee \rho_m(\underline{x})c_m(\underline{x})$$

where $c_i \in C$ and $\rho_i \in R$. In learning such a function, the learner does not know the identities of the projections $\rho_1, \ldots, \rho_m$ that are relevant to the particular disjunction being learned.

Below we describe Algorithm Y for learning such disjunctions, and analyze it. This online algorithm uses its one examples stream and learns a separate hypothesis h_ρ for each of the r projections $\rho \in R$. To learn h_ρ it updates its hypothesis for h_ρ according to algorithm A for each example $\underline{x}$ such that $\rho(\underline{x}) = 1$, and ignores $(\underline{x})$ if $\rho(\underline{x}) = 0$. The label of $\underline{x}$ for h_ρ is taken to be 1 if and only if $c(\underline{x}) = 1$. The hypothesis of Algorithm Y is then updated according to an algorithm B for learning disjunctions over the domain $V_r = \{0, 1\}^r$, where the r components v_i are taken as $\rho(\underline{x})h_\rho(\underline{x})$ for the various $\rho \in R$ and the truth value of the disjunction is taken as $c(\underline{x})$. More formally:

Algorithm Y.

Initialize h_ρ for each $\rho \in R$.
Initialize h for Algorithm B.
For each example $\underline{x}$:
 Let $v_\rho = \rho(\underline{x})h_\rho(\underline{x})$ for each ρ.
 Output the value of h on $\{v_\rho\}$ as the predicted value.
 For each ρ such that $\rho(\underline{x}) = 1$ update h_ρ according
  to A given $\underline{x}$ and label $c(\underline{x})$.
 For each ρ let $v_\rho = \rho(\underline{x})h_\rho(\underline{x})$.
 Update h according to B given $\{v_\rho \mid \rho \in R\}$
  and label $c(\underline{x})$.

We now prove the following, where we extend the notation for π and ν to have two parameters, r and n, to describe the total numbers of variables, and two more m and k, to describe the numbers of the relevant ones, instead of just one parameter each.

Theorem 4.1. *Suppose that Algorithm A and B have respective mistake bounds π_A, ν_A, π_B, and ν_B. Then for learning projective disjunctions over (C, R) where there are at most m disjuncts out of $r = |R|$ projections, and in each c_i at most k of the n variables are relevant, Algorithm Y has mistake bounds*:

$$\pi_Y(r, n; m, k; 0; 0) \leq \pi'_B(r; m; m \cdot \nu_A(n; k; 0; 0); m \cdot \pi_A(n; k; 0; 0)),$$

and

$$\nu_Y(r, n; m, k; 0; 0) \leq \nu'_B(r; m; m \cdot \nu_A(n; k; 0; 0); m \cdot \pi_A(n; k; 0; 0)).$$

Proof: In the course of running Algorithm Y we regard h_ρ for $\rho = \rho_i$ $(1 \leq i \leq m)$ as learning c_i from the domain X_n restricted to $\rho = 1$. Then calls of Algorithm A never get false labellings, since, by definition, if $\rho_i(\underline{x}) = 1$ then $c(\underline{x})$ is the correct labelling for $c_i(\underline{x})$. Hence for each ρ the hypothesis h_ρ produces at most $fp = \pi_A(n; k; 0; 0)$ false positives, and at most $fn = \nu_A(n; k; 0; 0)$ false negatives.

Let S be the decisive set consisting of the v_ρ corresponding to $\rho = \rho_i (1 \leq i \leq m)$. We regard Algorithm B as learning the disjunction of this set. Calls of Algorithm B may get false labellings. While for each $\underline{x}$ the supplied label $c(\underline{x})$ is correct, the supplied input $\underline{v}(x)$ will be corrupted if some of the h_ρ are incorrect. As long as h_ρ is correct for $\rho \in S$ the labelling $c(\underline{x})$ remains correct for calls of B. Hence if fp, fn are upper bounds on the mistakes for each h_ρ there will be at most $m \cdot fp$ examples where a decisive bit of $\underline{v}$ is a false positive, and at most $m \cdot fn$ examples where a decisive bit of $\underline{v}$ is a false negative. The former may cause false negative labellings and the latter false positive labellings of the disjunction being learned by Algorithm B. They contribute to $fnlm_S$ and to fpl respectively. Note that the values of v_ρ for $\rho \notin S$ are irrelevant in this calculation. In other words, the behavior of

these other h_ρ may be arbitrarily noisy, adversarial or inconsistent without ill effect! Hence

$$\pi_Y(r, n; m, k; 0; 0) \leq \pi'_B(r; m; m \cdot \nu_A(n; k; 0; 0); m \cdot \pi_A(n; k; 0; 0)),$$

and

$$\nu_Y(r, n; m, k; 0; 0) \leq \nu'_B(r; m; m \cdot \nu_A(n; k; 0; 0); m \cdot \pi_A(n; k; 0; 0)). \quad \square$$

Corollary 4.1. *If both of Algorithms A and B have the mistake bounds given in Theorem* 3.1 *for Algorithm W, then*

$$\pi_Y(r, n; m, k; 0; 0) \leq 6mk \log_2 n + 2m \log_2 r + 4m + 2,$$
$$\nu_Y(r, n; m, k; 0; 0) \leq 2mk \log_2 n + m \log_2 r + 2m.$$

Proof: From Theorem 3.1,

$$\pi_A(n; k; 0; 0) \leq 2k \log_2 n + 2,$$

and

$$\nu_A(n; k; 0; 0) = \nu'_A(n; k; 0; 0) \leq k \log_2 n.$$

Applying Theorem 3.1 again to Algorithm B:

$$\pi'_B(r; m; m \cdot \nu_A(n; k; 0; 0); m \cdot \pi_A(n; k; 0; 0))$$
$$\leq 2(m \log_2 r + m(2k \log_2 n + 2) + mk \log_2 n + 1),$$

and

$$\nu'_B(r; m; m \cdot \nu_A(n; k; 0; 0); m \cdot \pi_A(n; k; 0; 0)) \leq m \log_2 r + m(2k \log_2 n + 2). \quad \square$$

Corollary 4.2. *If Algorithms A and B have the mistake bounds of Corollary* 3.2 *and Theorem* 3.1 *respectively, then*

$$\pi_Y(r, n; m, k; 0; 0) \leq 6mk \log_2 n + 2m \log_2 r + 4m + 2,$$
$$\nu_Y(r, n; m, k; 0; 0) \leq mk \log_2 n + m \log_2 r.$$

Proof: From Corollary 3.2

$$\pi_A(n; k; 0; 0) \leq k \log_2 n,$$

and

$$\nu_A(n; k; 0; 0) = \nu''(n; k; 0; 0) \leq 2k \log_2 n + 2.$$

Applying Theorem 3.1 to Algorithm B,

$$\pi'_B(r; m; m \cdot \nu_A(n; k; 0; 0); m \cdot \pi_A(n; k; 0; 0))$$
$$\leq 2(m \log_2 r + m(2k \log_2 n + 2) + mk \log_2 n + 1)$$
$$\nu'_B(r; m; m \cdot \nu_A(n; k; 0; 0); m \cdot \pi_A(n; k; 0; 0)) \leq m \log_2 r + mk \log_2 n. \quad \square$$

From the above statements we can deduce attribute efficient learnability for a variety of *projective* classes. We can consider R to contain only conjunctive constraints (e.g. $x_i x_j = 1$). Then if C is conjunctions then the resulting class is a class of *projective* DNF, a subclass of disjunctive normal form. We can equally consider for C the class of disjunctions or the class of Boolean functions that are separable by linear inequalities with small integer coefficients. Also in either case the class R can be broadened beyond conjunctions to, among others, disjunctions or threshold functions. Finally, our composition construction can be iterated more than once.

5. Sequential structures

Production systems, or sequences of condition-action rules have been frequently suggested as appropriate for representing cognitive computations (Newell & Simon, 1972). These can be formalized skeletally as decision lists (Rivest, 1987; Khardon, 1996). No polynomial time learning algorithm is known that can learn decision lists attribute efficiently in the strong sense so far considered here, that the computation time is polynomial in all the parameters, and sample complexity is polynomial in m the number of relevant variables, and logarithmic in r, the number of irrelevant ones. If this sense is relaxed to *weak* attribute efficiency, where the dependence of the sample complexity on m is allowed to be exponential while the dependence on r is still logarithmic, then a positive result holds: It is easy to write a decision list of m relevant variables as a linear inequality on m variables with integer coefficients whose magnitudes sum to 2^m. Further, variants of Winnow (Littlestone, 1988) can learn linear threshold functions with integer coefficients over r variables, with the number of mistakes growing proportionally to $\log_2 r$ times the square of the sum of the magnitudes of the coefficients.

We can define decision-lists over (C, R), in analogy with the disjunctions we defined earlier (cf. Bshouty, Tamon, & Wilson, 1996), as

$$c = \rho_1 c_1 \vee \bar{\rho}_1 \rho_2 c_2 \vee \bar{\rho}_1 \bar{\rho}_2 \rho_3 c_3 \vee \cdots \vee \bar{\rho}_1 \bar{\rho}_2 \cdots \bar{\rho}_{m-1} \rho_m c_m$$

and attempt to understand their learnability. Note that if the learner knows the identities of $\rho_1, \rho_2, \ldots, \rho_m$ ahead of time then the task would be equivalent to learning projective disjunctions, since the projections $\{\bar{\rho}_1 \cdots \bar{\rho}_{i-1} \rho_i \mid 1 \leq i \leq m\}$ are mutually exclusive and could be chosen as R. Attribute-efficient learning even in the weak sense is of interest here since it is plausible that in cognitive computations the depth m of the list is some small number.

For completeness we shall describe here some positive results for restricted cases that follow from known methods. They are similar to ones referred to by Blum and Singh (1990)

and also discussed by Kivinen, Mannila and Ukkonen (1992) and by Dhagat and Hellerstein (1994).

We consider decision lists over (C, R), where $C = \{0, 1\}$ is the class consisting of the two constant functions zero and one. In other words we have standard decision-lists with conditions $\rho \in R$ at the nodes, and constants 0 or 1 at the leaves. We define the *degree* of a decision list of m decision points (i.e. internal nodes) to be d if d of the leaves have value 1, and the remainder have value 0. We say that it has t *alternations* if the sequence of constants at the leaves can be partitioned into no more than $t + 1$ subsequences, where each subsequence is all 0 or all 1. Thus, for example, a decision list of degree d has at most $2d$ alternations. We note also that, by convention, the last internal node always leads to two leaves with distinct labels, and hence the last sequence of leaves having the same label consists always of just a single leaf.

Theorem 5.1 gives an improved result for this class in the case that d or t is significantly smaller than m. One motivation for considering the small degree case comes from decision-list systems where the leaves are labelled from a larger set than $\{0, 1\}$, and each label recommends an action. We may have a large decision list of m leaves where each label occurs a small number d of times. Then each label can be considered to define its own degree d decision list.

For the purposes of the proof below it is convenient to reverse the ordering of the ρ_i, and to denote by $j_1, j_2, \ldots, j_d$ the indices of the c_i such that $c_i = 1$. Then a degree d decision list can be written as:

if ($\rho_i = 1$ for any $i > j_d$) $c = 0$;
  else if ($\rho_{j_d} = 1$) $c = 1$;
    else if ($\rho_i = 1$ for any i such that $j_{d-1} < i < j_d$) $c = 0$;
      else if ($\rho_{j_{d-1}} = 1$) $c = 1$;
        else if ($\rho_i = 1$ for any i such that $j_{d-2} < i < j_{d-1}$) $c = 0$;
                    $\vdots$
          else if ($\rho_{j_1} = 1$) $c = 1$;
            else $c = 0$.

We can show the following, which implies polynomial time learnability with mistake bounds depending on the relevant variables as $(2m/d)^d$ and $2t(m/t)^t$ rather than the general 2^m bound mentioned earlier. (We note that respective bounds of $O(m^d)$ and $O(m^t)$ can be obtained more simply.)

Theorem 5.1. *A decision list over* $(\{0, 1\}, R)$ *with m leaves and degree d can be expressed as a linear inequality* $\sum w_i \rho_i > \tau$ *where* τ *and every coefficient* w_i *is an integer, and the sum of the magnitudes of the* $|w_i|$ *is less than* $(2m/d)^d$. *If the decision list has t alternations then the sum of the magnitudes of the* $|w_i|$ *is less than* $2(m/t + 1)^t$.

Proof: We shall show the first upper bound by induction on d. Let $L_{d-1}(\underline{\rho}) > 0$ be a linear inequality for expressing the condition that $c = 1$ in the decision list obtained by deleting $\rho_m, \rho_{m-1}, \ldots, \rho_{j_d+1}, \rho_{j_d}$, from the decision list. We shall derive from this a

linear inequality $L_d(\underline{\rho})$ for the original decision list in two stages. First we add back ρ_{j_d}, and second we add back the remaining nodes $\rho_m, \rho_{m-1}, \ldots, \rho_{j_d+1}$. For the first stage we consider the inequality $L_d^*(\underline{\rho}) > 0$ where

$$L_d^*(\underline{\rho}) = L_{d-1}(\underline{\rho}) - \rho_{j_d} \cdot \left(\min \left\{ \min_{\substack{\underline{\rho} \text{ s.t.} \\ \rho_{j_d}=1}} \{L_{d-1}(\underline{\rho})\}, 0 \right\} - 1 \right) \tag{7}$$

Clearly if $\rho_{j_d} = 0$ then for all such $\underline{\rho}$ $L_d^*(\underline{\rho}) = L_{d-1}(\underline{\rho})$, as desired, and if $\rho_{j_d} = 1$ then $L_d^*(\underline{\rho}) > 0$ for all $\underline{\rho}$ with that constraint, again as needed. (Note that we may minimize over $\{0, 1\}$ values of the ρ_j for vectors $\underline{\rho}$ consistent with the domain X_n' from which the examples are drawn. Some choices of X_n' may yield better bounds than the general case that we analyze here. Note also that if the minimal value of L_{d-1} is positive, then the value zero is used instead of it.) For the second stage we consider the inequality $L_d(\underline{\rho}) > 0$ where

$$L_d(\underline{\rho}) = L_d^*(\underline{\rho}) - \sum_{i=j_d+1}^{m} \rho_i \cdot \left(\max \left\{ \max_{\substack{\underline{\rho} \text{ s.t.} \\ \rho_i=1}} \{L_d^*(\underline{\rho})\}, 0 \right\} \right). \tag{8}$$

Then if $\rho_i = 0$ for all $i\,(j_d + 1 \le i \le m)$ then for all such $\underline{\rho}$ $L_d(\underline{\rho}) = L_d^*(\rho)$. If $\rho_i = 1$ for some such i, then $L_d(\underline{\rho}) \le 0$. In both cases this is as needed. (Note that here we are assuming similarly that if the maximum value of L_d^* is negative, then the value zero is used instead of it.)

Now let $|L|$ denote the sum of the magnitudes of the coefficients of the linear form L. Then the two stages of the construction imply respectively that

$$|L_d^*| \le 2|L_{d-1}| + 1,$$

and

$$|L_d| \le (m - j_d + 1)|L_d^*|.$$

For the basis of the induction let $L_1(\underline{\rho}) > 0$ where

$$L_1(\underline{\rho}) = \rho_{j_1} - \sum_{j_1 < i < j_2} \rho_i.$$

Then clearly if $\rho_i = 1$ for any $i\,(j_1 < i < j_2)$ then $L_1(\underline{\rho}) \le 0$, while otherwise if $\rho_{j_1} = 1$ then $L_1(\underline{\rho}) > 0$. If all these ρ values are zero, then $L_1(\underline{\rho}) \le 0$ also as required. This gives the basis inequality $|L_1| \le j_2 - j_1$.

Putting together these inequalities gives for $d > 1$ that $|L_d|$ is upper bounded by a quantity l_d satisfying

$$l_i < (j_{i+1} - j_i)(2l_{i-1} + 1)$$

for $d \geq i \geq 2$ where $j_{d+1} = m + 1$ and $l_1 = j_2 - j_1$. In other words there exist quantities $x_1, \ldots, x_d$ where $x_i = j_{i+1} - j_i \geq 1$, such that $\sum x_i = j_{d+1} - j_1 \leq m$,

$$l_i < x_i(2l_{i-1} + 1) \quad \text{for } i = 2, \ldots, d, \text{ and } l_1 = x_1$$

Multiplying out gives:

$$\begin{aligned} l_d &< x_d(2x_{d-1}(2x_{d-2} \cdots 2x_2(2x_1 + 1) \cdots) + 1) \\ &\leq \sum_{i=1}^{d} 2^{d-i} x_d \cdots x_i \\ &< 2^d x_d \cdots x_1 \\ &\leq (2m/d)^d \text{ by the arithmetic-geometric mean inequality.} \end{aligned}$$

We can analyze the more general case of t alternations by the same method. We use (8) above for sequences of 0-leaves, and an adaptation of (7) for sequences of 1-leaves. Suppose that the kth sequence of identical leaves correspond to $\rho_{j_{k-1}+1}, \ldots, \rho_{j_k}$ where $j_0 = 0$ and $j_{t+1} = m + 1$. We define the linear inqualitites $\{L_k\}$ inductively. If the tth such sequence consists of 0-leaves we have

$$L_t(\underline{\rho}) = L_{t-1}(\underline{\rho}) - \sum_{i=j_t+1}^{m} \rho_i \left(\max \left\{ \max_{\substack{\underline{\rho} \text{ s.t.} \\ \rho_i = 1}} \{L_{t-1}(\underline{\rho})\}, 0 \right\} \right)$$

and it consists of 1-leaves we have

$$L_t(\underline{\rho}) = L_{t-1}(\underline{\rho}) - \sum_{i=j_t+1}^{m} \rho_i \left(\min \left\{ \min_{\substack{\underline{\rho} \text{ s.t.} \\ \rho_i = 1}} \{L_{t-1}(\underline{\rho})\}, 0 \right\} - 1 \right)$$

The last sequence is always a single leaf. If it is a 0-leaf then it can be ignored. Hence the worst case is a 1-leaf, which gives a base equation of

$$L_1(\rho) = 1.$$

Hence if l_{i+1} is the sum of the magnitudes of the coefficients (but excluding the constant term) for i alternations, then $l_i < (x_i + 1)(l_{i-1} + 1)$ for $i = 2, \ldots, t + 1$, and $l_1 = 0$, for some $x_2, \ldots, x_{t+1}$ such that $x_2 + \cdots + x_{t+1} \leq m$. Putting $y_i = x_i + 1$ gives the recurrence $l_i < y_i(l_{i-1} + 1)$ where now $y_2 + \cdots + y_{t+1} \leq m + t$. Hence

$$l_{t+1} < \sum_{k=2}^{t+1} \prod_{i=k}^{t+1} y_i.$$

Using the fact that $y_i \geq 2$ for all i and the arithmetic-geometric mean inequality gives the claimed bound. □

A number of concept classes that are natural applications of our Algorithm Y and Theorem 4.1 can be learned more simply by appropriately recoding the input representation and applying Winnow and Theorem 5.1. We shall consider the class of projective disjunctions over (C, R), for various C, R and apply the following corollary of Theorem 9 and Example 6 of Littlestone (1989):

Theorem 5.2. (*Littlestone*): *For some constant κ a variant of Winnow can learn any inequality $\sum_{i=1}^{n} w_i x_i > \tau$, where $w_1, \ldots, w_n$ are integral and the sum of the magnitudes of the w_i equals Z, with mistake bound no more than $\kappa Z^2 \log n$.*

Suppose first that C can be expressed as a class of linear separators that are learnable by Winnow attribute efficiently, and that $R = \{\rho_1, \ldots, \rho_r\}$ is a set of restrictions that are *disjoint* (i.e. $\forall \underline{x} \in X_n' \rho_i(\underline{x}) = \rho_j(\underline{x}) = 1 \Rightarrow i = j$). Then it is easy to see that if we define $y_i = \bar{\rho}_1 \bar{\rho}_2 \cdots \bar{\rho}_{i-1} \rho_i \bar{\rho}_{i+1} \cdots \bar{\rho}_r$ for $1 \leq i \leq r$ then replacing ρ_i by y_i in any expression f being learned leaves f invariant on X_n'. We now introduce a new set of rn variables $\{z_{ij} \mid 1 \leq i \leq n, 1 \leq j \leq r\}$ where $z_{ij} = x_i y_j$. It is easy to see that a linear inequality suffices where the coefficients of the z_{ij} variables are equal to those of $\rho_j c_j$ when these are learned separately.

Second, let us consider the case that C is the set of conjunctions. Corollary 4.1 gives a mistake bound of $O(mk \log(rn))$ for learning a projective disjunction of m conjunctions under r projections where at most k variables out of the n appear in any conjunction. The question arises whether this bound can be obtained more directly, even in the nonprojective case. In particular as a referee of this paper has observed, a projective disjunction $\bigvee_i (\rho_i \bigwedge_j x_{i_j})$ can be expressed as a decision list with two alternations as follows: starting from the root there is a sequence of internal nodes representing $\rho_i \bar{x}_{i_j}$ for all i, j that lead to 0-labelled leaves when these conditions are satisfied. Then come a sequence of internal nodes representing ρ_i for all i, that lead to 1-labelled leaves. Finally there is a 0-labelled leaf. It appears, however, that applying to this construction any of the known algorithms mentioned for learning decision lists with a fixed number of alternations yields worse bounds than Corollary 4.1. For example, a direct application of Theorems 5.1 and 5.2 would give a larger bound of $O((mk)^4 \log(rn))$.

Finally, for this same concept class an alternative coding is to have a new variable for each of the $\Omega(n^k)$ possible k-conjunctions of variables and then use Winnow. However, if $k > 2$ this introduces an unacceptably high dependence, as compared with Algorithm Y, of the computational cost on n.

6. Conclusion

If a concept cannot be learned satisfactorily because it does not approximate a function in any class for which a learning algorithm is available, it is natural to do the following. One tries a variety of restrictions of the function to various subdomains in the hope that the functions on some of these restrictions will be simple, and will become accessible to the available algorithms. If such favorable restrictions cover most cases then it remains to put together into a single hypothesis the hypotheses for the various restrictions, in order to

derive a hypothesis for the whole function. This general methodology we call projection learning.

In this paper we described an algorithm for this that exhibits the desired properties in a certain context. In particular we showed some conditions under which attribute-efficient learnability of the subdomains was sufficient to guarantee attribute-efficient learning over the whole domain. Mistake-bounded analysis was found to be effective, though these phenomena may also be amenable to analysis directly in the PAC model.

The algorithm described suggests a broader variety of algorithms that might be tried as a heuristics for difficult learning problems. For example, if the variable set is very large, as may be the case in cognitive or computational biology applications, it may be more practicable to use as the set of projections a much smaller set of carefully preselected restrictions, and not fully exploit attribute-efficiency at the level of the projections.

We expect that projection learning might find a use within a broader toolkit of methods. As observed in the previous section, several classes of functions that can be learned by the basic algorithm described can also be learned by Winnow itself, though sometimes with quantitatively inferior bounds, after the variable set has been recoded to an appropriate larger set. We have not been able to usefully characterize the domains in which our methods outperform others. However, Winnow appears to be particularly useful in practice in noisy environments where linear inequalities with moderate size integer coefficients provide adequate approximations to the function being learned. Hence applications having projections with these properties look the most promising.

The development of efficient learning algorithms for particular representations can be expected to influence the representations of choice used in computer systems that perform cognitive computations. While sequential structures such as production systems and decision lists have been advocated for their expressiveness, it is possible, though not proven, that there are fundamental impediments to their attribute-efficient learnability. If these apparent impediments remain insurmountable, then it would appear that in the context of cognitive computations attention will need to be redirected to the flatter, possibly projective, structures suggested by this paper.

Acknowledgment

I am grateful to Amos Beimel and to several referees for some unusually helpful comments on this paper.

References

Blum, A. (1992). Learning Boolean functions in an infinite attribute space. *Machine Learning*, *9*(4), 373–386.

Blum, A., Hellerstein, L., & Littlestone, N. (1995). Learning in the presence of finitely or infinitely many irrelevant attributes. *JCSS*, *50*, 32–40.

Blum, A., & Singh, M. (1990). Learning functions of *k* terms. *Proc. 3rd Annual Workshop on Computational Learning Theory* (pp. 145–153).

Bshouty, N., & Hellerstein, L. (1998). Attribute-efficient learning in query and mistake-bounded models. *JCSS*, *56*, 310–319.

Bshouty, N. H., Tamon, C., & Wilson, D. K. (1996). On learning width two branching programs. *Proc. 9th ACM Conference on Computational Learning Theory* (pp. 224–227).

Bshouty, N. H., Tamon, C., & Wilson, D. K. (1995). On learning decision trees with large output domain. *Proc. 8th ACM Conf. on Computational Learning Theory* (pp. 190–197).

Dhagat, A., & Hellerstein, L. (1994). PAC learning with irrelevant attributes. *Proc. IEEE Symp. on Foundations of Computer Science* (pp. 64–74).

Golding, A. R., & Roth, D. (1996). Applying Winnow to context-sensitive spelling correction. *Proc. 13th International Conference on Machine Learning* (pp. 182–190). San Francisco, CA: Morgan Kaufmann.

Haussler, D. (1988). Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence*, *36*(2), 177–222.

Jackson, J., & Craven, M. (1996). Learning sparse perceptrons. In *Advances in Neural Information processing* (Vol. 8) (pp. 654–660). MIT Press.

Kearns, M. J., & Vazirani, U. V. (1994). *An introduction to computational learning theory*. MIT Press.

Khardon, R. (1996). Learning to take actions. *Proc. 1996 AAAI* (pp. 787–792).

Kivinen, J., Mannila, H., & Ukkonen, E. (1992). Learning hierarchical rule sets. *Proc. 5th ACM workshop on Computational Learning Theory* (pp. 37–44). New York, NY: ACM Press.

Kivinen, J., & Warmuth, M. K. (1995). The perceptron algorithm vs. Winnow: Linear vs. logarithmic mistake bounds where few input variables are relevant. *Proc. 8th ACM Conference on Computational Learning Theory* (pp. 289–296). New York: ACM Press.

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

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

Rivest, R.L. (1987). Learning decision lists. *Machine Learning*, *2*(3), 229–246.

Valiant, L. G. (1998). A neuroidal architecture for cognitive computation. *ICALP 98, Proceeding of the 25th International Colloquium on Automata, Languages and Programming* (pp. 642–669). Aarhus, Denmark, July 13–17: Springer-Verlag.

Valiant, L. G. (1999). Robust logics. *Proc. 31st ACM Symposium on Theory of Computing* (pp. 642–651). New York: ACM Press.

Received January 12, 1998

Accepted April 30, 1999

Final Manuscript April 30, 1999