Projection Learning

Leslie G. Valiant

TR-19-97

Center for Research in Computing Technology
Harvard University
Cambridge, Massachusetts

Projection Learning*

Leslie G. Valiant
Division of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138

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 can be viewed as strict generalizations of Littlestone's Winnow algorithm, and, therefore, appropriate to domains having very large numbers of attributes, but where nonlinear hypotheses are sought.

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, there is no evidence for the existence of explicit methods for achieving these ends. The learning process appears to proceed with the set previously known functions as the set of variables, and overcomes these problems apparently 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, 1996).

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

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 (e.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. Limiting dependence to a small number of the variables is the simplest expression of this. The question therefore is: just how efficiently can a function of k variables be learned in the presence 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 in cognitive settings with tens of thousands of variables (Golding and 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 in certain natural and simple ways algorithms that have this property. This result applies directly to certain subclasses of disjunctive normal form formulae, that we call projective DNF. 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, 1996). 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 in a network setting, rather than merely propositional information. In these three respects the knowledge structures we describe can be used as an alternative to the classes of learnable action strategies of Khardon (1996), that has the advantage of attribute efficiency. It is an open empirical question 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 $x_1, \cdots, x_n$. 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 $f : X_n \rightarrow \{0,1\}$ 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* 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

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

where $\rho_1, \cdots, \rho_m \in R, c_1, \cdots, c_m \in C$ and for each i $(1 \leq i \leq m)$ $\rho_i c = \rho_i c_i$. Note that the novel constraint is the 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 $x_1, \cdots, x_\ell$ to the 2^ℓ distinct combinations of truth values. More typically, R may be chosen 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.

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. Then for each $\rho_i \in R$ the hypothesis h_{ρ_i} can be expected to converge to $c_i \in C$. For other values of ρ nothing is guaranteed about h_ρ.

3 Mistake-Bounded Learning

We shall show for various algorithms that they learn attribute-efficiently. In particular, they converge to an approximation of the correct hypothesis after seeing a number of examples whose dependence on the irrelevant attributes is logarithmic. 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 $\underline{x}^1, \underline{x}^2, \cdots$ 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 a *mistake-bounded* 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 will, 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 $\{x_{i_1}, \cdots, x_{i_k}\}$ of k variables such that for all $\underline{x}', \underline{x}'' \in X'_n$, $c(\underline{x}') = c(\underline{x})''$ whenever $\underline{x}', \underline{x}''$ agree on $x_{i_1}, \cdots x_{i_k}$. If this is the case then $\{x_{i_1}, \cdots, x_{i_k}\}$ is called a *decisive* set for c on X'_n. A variable x_j is called *decisive* if it belongs to some decisive set.

Finally, we wish to express the number π of false positives and the number ν 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 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 $\pi(n; k; \mathit{fpl}; \mathit{fnl})$ and $\nu(n; k; \mathit{fpl}; \mathit{fnl})$.

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 sets 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. $c = x_{i_1} + \cdots + x_{i_k}$).

Theorem 3.1 *For the class of monotone disjunctions Algorithm W has the following mistake bounds, where S is any decisive set:*

$$\begin{aligned} \pi'(n; k; fpl; fnlm_S) &\leq 2(k \log_2 n + fnlm_S + fpl + 1), \text{ and} \\ \nu'(n; k; fpl; fnlm_S) &\leq k \log_2 n + fnlm_S. \end{aligned}$$

<u>**Proof**</u> 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

$$\#promotions = fn + fpl. \tag{3-1}$$

Suppose without loss of generality that $c \equiv x_1 + x_2 + \cdots + x_k$ on X'_n so that $\{x_1, \cdots, x_k\}$ 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

$$\#promotions \leq k\lfloor \log_2 n \rfloor + fnlm_S + fpl. \tag{3-2}$$

Combining (3.1) and (3.2) gives

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

Now a demotion can occur only when $\sum \underline{w}(\underline{x}) > n$. 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 $\sum \underline{w}(\underline{x}) \leq n$. 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 $n/2$,

$$\#demotions \leq 2 \cdot \#promotions + 2. \tag{3-4}$$

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

Finally we note that by the definitions

$$\#demotions = fp + fnl. \tag{3-5}$$

Combining (3.2), (3.4) and (3.5) gives

$$fp \leq 2k\lfloor \log_2 n \rfloor + 2fnlm_S + 2fpl + 2 - fnl. \quad \Box \tag{3-6}$$

We note that by Boolean duality Theorem 3.1 can be adapted to learning conjunctions. Thus in order to learn a conjunction c over $x_1, \cdots, x_n$ we learn a disjunction c' over $\bar{x}_1, \cdots, \bar{x}_n$ 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:*

$$\begin{aligned}\pi''(n;\ k; fplm_S;\ fn) &\leq k\log_2 n + fplm_S, \textit{ and}\\ \nu''(n;\ k;\ fplm_S;\ fn) &\leq 2(k\log_2 n + fplm_S + fnl + 1) \qquad \Box.\end{aligned}$$

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 $w_i \leftarrow (1 + q^{-1})w_i$ 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 Some Projection Learning Algorithms

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 μ_A and ν_A, and R is a projection set.

As defined in § 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 $\rho_1, \cdots, \rho_m$ that are relevant to the particular disjunction being learned.

Below we describe Algorithm Y for learning such disjunctions, and analyze it. The 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 true value of the disjunction is taken as $c(\underline{x})$. More formally:

Algorithm Y.

```
Initialize $h_\rho$ for each $\rho \in R$, and initialize $h$ for Algorithm $B$. For each example $\underline{x}$:
  For each $\rho$ such that $\rho(\underline{x}) = 1$ updated $h_\rho$ according
     to A given $\underline{x}$, and label $c(\underline{x})$.
  For each $\rho$ 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 disjunctions over (C,R) where there are at most m disjuncts out of $r = \mid R \mid$ projections, and in each c_i at most k of the n variables are relevant, Algorithm Y has mistake bounds:*

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

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

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. Note that the values of v_ρ for $\rho \notin S$ are irrelevant in this calculation. Hence

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

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

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,

$$\pi_A(n;\ k;\ 0;\ 0) \leq 2k \log_2 n + 2, \mathit{and}$$
$$\nu_A(n;\ k;\ 0;\ 0) = \nu'_A(n;\ k;\ 0;\ 0) \leq k \log_2 n.$$

Applying Theorem 3.1 again to Algorithm B:

$$\begin{aligned} &\pi'_B(r;\ m;\ m \cdot \nu_A(n;\ k;\ 0;\ 0);\quad m \cdot \pi_A(n;\ k;\ 0;\ 0)) \\ &\quad \leq 2(m \log_2 r + m(2k \log_2 n + 2) + mk \log_2 n + 1), \text{ and} \\ &\nu'_B(r;\ m;\ m \cdot \nu_A(n;\ k;\ 0;\ 0);\quad m \cdot \pi_A(n;\ k;\ 0;\ 0)) \\ &\quad \leq m \log_2 r + m(2k \log_2 n + 2). \qquad \Box \end{aligned}$$

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

$$\begin{aligned} \pi_Y(r, n;\ m, k;\ 0;\ 0) &\leq 6mk \log_2 n + 2m \log_2 r + 4m + 2, \\ \nu_Y(r, n;\ m, k;\ 0;\ 0) &\leq mk \log_2 n + m \log_2 r. \end{aligned}$$

Proof From Corollary 3.2

$$\begin{aligned} \pi_A(n;\ k;\ 0;\ 0) &\leq k \log_2 n, \text{ and} \\ \nu_A(n;\ k;\ 0;\ 0) = \nu''(n;\ k;\ 0;\ 0) &\leq 2k \log_2 n + 2. \end{aligned}$$

Applying Theorem 3.1 to Algorithm B,

$$\begin{aligned} &\pi'_B(r;\ m;\ m \cdot \nu_A(n;\ k;\ 0;\ 0);\ m \cdot \pi_A(n;\ k;\ 0;\ 0)) \\ &\quad \leq 2(m \log_2 r + m(2k \log_2 n + 2) + mk \log_2 n + 1) \\ &\nu'_B(r;\ m;\ m \cdot \nu_A(n;\ k;\ 0;\ 0);\ m \cdot \pi_A(n;\ k;\ 0;\ 0)) \\ &\quad \leq m \log_2 r + mk \log_2 n. \qquad \Box \end{aligned}$$

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. 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.

We envisage that Algorithm Y would be used in applications where the number of variables is very large. Consider, for example, the following setting. For large values of n it will not be practicable, even for a certain small integer g_0, to create the set of all conjunctions of combinations of g variables for $g \geq g_0$ since the size of this set would grow as n^g. Hence even in cases in which the concept class being learned can be viewed as disjunctions over conjunctions of some fixed number g of variables, it may not be practicable to learn these as disjunctions over n^g variables. For example, if R is the set of single variable disjunctions, and C the set of conjunctions with k relevant variables, then the associated disjunctive normal form would have conjunctions of size $k+1$, and these could then not be learned through simple disjunctions if $k+1 \geq g_0$. On the other hand, Algorithm Y, with Winnow instantiated at both levels, works with only rn weights, which in this case would be $2n^2$, rather than the potentially prohibitive n^{k+1}.

5 Sequential Structures

Production systems, or sequences of condition-action rules have been frequently suggested as appropriate for representing cognitive computations (Newell and 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, 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 bounded in magnitude by 2^m. Further, variants of Winnow (Littlestone, 1988) can learn these with sample complexity growing proportionally to $\log_2 r$ times the maximum magnitude of the coefficients.

We can define decision-lists over (C, R), in analogy with the disjunctions we defined earlier (cf. Bshouty, Tamon and 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 $\rho_1, \rho_2, \cdots, \rho_m$ 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. 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.

The following gives a positive result for a restricted case. 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 branches to be d if d of the leaves have value 1, and the remainder have value 0. Theorem 5.1 gives an improved result for this class in the case that d is significantly smaller than m. One motivation for considering this 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 $j_1, j_2, \cdots, j_d$ 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 $i$ $j_{d-1} < i < j_d$) $c = 0$;
         else if ($\rho_{j_{d-1}} = 1$) $c = 1$;
            else if ($\rho_i = 1$ for any $i$ $j_{d-2} < i < j_{d-1}$) $c = 0$;
                              ⋮
               else if ($\rho_{j_1} = 1$) $c = 1$;
                else $c = 0$.
```

We can show the following, which implies polynomial time learnability with sample complexity depending on the relevant variables as $(2m/d)^d$ rather than the general 2^m bound mentioned earlier.

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

Proof We shall show this upper bound by induction on d. Let $L_{d-1}(\underline{\rho}) \geq 0$ be a linear inequality for expressing the condition that $c = 1$ in the decision list obtained by deleting $\rho_m, \rho_{m-1}, \cdots, \rho_{j_d+1}, \rho_{j_d}$, 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 $\rho_m, \rho_{m-1}, \cdots, \rho_{j_d+1}$. For the first stage we consider the inequality $L_d^*(\underline{\rho}) \geq 0$ where

$$L_d^*(\underline{\rho}) = L_{d-1}(\underline{\rho}) - \rho_{j_d} \cdot \min\{ \min_{\substack{\underline{\rho}\ s.t. \\ \rho_{j_d}=1}} \{L_{d-1}(\underline{\rho})\}, 0\}. \tag{5-1}$$

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 $L_d^*(\underline{\rho}) \geq 0$ for all $\underline{\rho}$ with that constraint, again as needed for that circuit. (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 $L_d(\underline{\rho}) \geq 0$ where

$$L_d(\underline{\rho}) = L_d^*(\underline{\rho}) - \sum_{i=j_d+1}^{m} \rho_i \cdot (\max\{\max_{\substack{\underline{\rho}\ s.t.\\ \rho_i=1}}\{L_d^*(\underline{\rho})\}, 0\} + 1). \tag{5-2}$$

Then if $\rho_i = 0$ for all i $(j_d + 1 \leq i \leq m)$ then for all such $\underline{\rho}$ $L_d(\underline{\rho}) = L_d^*(\rho)$. If $\rho_i = 1$ for some such i, then $L_d(\underline{\rho}) < 0$. 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 $\mid L \mid$ denote the sum of the magnitudes of the coefficients of the linear form L. Then the two stages of the construction imply respectively that

$$\mid L_d^* \mid \leq\ 2 \mid L_{d-1} \ \mid \ \text{ and } \ \mid\ L_d \mid \leq\ (m - j_d + 1) \mid L_d^* \mid + m - j_d.$$

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

$$L_1(\underline{\rho}) = \rho_{j_1} - 1 - \sum_{j_1 < i < j_2} \rho_i.$$

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

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

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

for $d \geq i \geq 1$ where $j_{d+1} = m + 1$ and $l_0 = 0$. In other words there exist quantities $x_1, \cdots, x_d$ where $x_i = j_{i+1} - j_i \geq 1$, such that $\sum x_i = j_{d+1} - j_1 \leq m$,

$$l_i < x_i(2l_{i-1} + 1) \text{ for } i = 1, \cdots, d.$$

Multiplying out gives:

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

It is possible that there are significant limits to the attribute-efficient learnability of sequential structures such as decision lists. In that case, in the context of cognitive computations attention needs to be redirected towards flatter structures, such as projective disjunctions.

Acknowledgment

I am grateful to Amos Beimel for his comments on this paper.

References

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

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

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

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

Kearns, M.J. and 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. and Warmuth, M.K. (1995). The perceptron algorithm vs. Winnow: linear vs. logarithmic mistake bounds where few input variables are relevant. In *Proc. 8th ACM Conference on Computational Learning Theory*, 289-296.

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

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

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

Valiant, L.G. (1996). A neuroidal architecture for cognitive computation. Technical Report TR-11-96, Center for Research in Computing Technology, Harvard University.