On Learning Simple Deterministic and Probabilistic Neural Concepts

Mostefa Golea and Mario Marchand

Ottawa-Carleton Institute for Physics
University of Ottawa
Ottawa, Ont., Canada K1N 6N5
e-mail: (golea,mario)@physics.uottawa.ca

Abstract

We investigate the learnability, under the uniform distribution, of deterministic and probabilistic neural concepts that can be represented as *simple combinations* of *nonoverlapping* perceptrons with binary weights. Two perceptrons are said to be nonoverlapping if they do not share any input variables. In the deterministic case, we investigate, within the distribution-specific PAC model, the learnability of *perceptron decision lists* and *generalized perceptron decision lists*. In the probabilistic case, we adopt the approach of *learning with a model of probability* introduced by Kearns and Schapire [10] and Yamanishi [14], and investigate a class of concepts we call *probabilistic majorities* of nonoverlapping perceptrons. We give polynomial time algorithms for learning these restricted classes of networks. The algorithms work by estimating various statistical quantities that yield enough information to infer, with high probability, the target concept.

1 Introduction

Despite the excitement generated recently by neural networks, learning in these systems has proven to be very difficult from a theoretical perspective. In fact, if one adopts the PAC learning's point of view, neural networks have accumulated far more negative (non-learnability) results than positive ones [2, 3]. The reasonable approach in such situations is to look for positive results by considering restricted classes of the problem, by providing the learning algorithm with additional information in the form of queries, and/or by restricting the distribution of examples.

A restriction that has been well studied with respect to boolean formulas is the *read-once* or the μ restriction on specific distributions. In particular, positive learnability results have been obtained for read-once DNF [4, 5]

on the uniform distribution and read-once boolean formulas over the basis {AND,OR,NOT} on product distributions [6].

Nonoverlapping perceptron networks seem to be the natural neural version of read-once boolean formulas. Two perceptrons are said to be nonoverlapping if they do not share any input variables [7]. Standard techniques [4] show that under an arbitrary distribution, such networks are no easier to learn than networks with overlapping perceptrons. Recently, Hancock *et al.* [7] proved this class to be PAC learnable from examples and membership queries under an arbitrary distribution. Golea *et al.* [9] proved the PAC learnability of the class of unions (intersections) of nonoverlapping perceptrons with *binary* weights and arbitrary thresholds under the uniform distribution.

In this paper, we restrict ourselves to the case where the distribution of examples is uniform, and investigate the problem of learning concepts that can be represented as simple combinations of nonoverlapping perceptrons with binary weights and arbitrary thresholds. We will consider both *deterministic* and *probabilistic* neural concepts. In the deterministic case, we move substantially beyond the results of Golea *et al.* [9] and investigate within the distribution-specific PAC model, the learnability of *decision lists* and *generalized decision lists* built from nonoverlapping perceptrons (fig. 1). In the probabilistic case, we adopt the approach of *learning with a model of probability* introduced by Kearns and Schapire [10] and Yamanishi [14], and investigate a class of concepts we call *probabilistic majorities* of nonoverlapping perceptrons (see definitions below). We give polynomial time algorithms for learning these restricted classes of networks. The algorithms work by estimating various statistical quantities that yield enough information to infer, with high probability, the target concept. Because our algorithms are statistical in nature, they are robust against a large amount of random classification noise [11].

While the high-level structure of our algorithms is somewhat similar to Schapire's [6], the specific tests used here are new. Furthermore, under the uniform distribution, our results imply simpler algorithms for learning μ-DNF [5] and read-once boolean formulas over the basis {AND,OR,NOT} [6]. They also imply the learnability of certain classes of μ-DNF formulas where the satisfiability of a given term does not imply the satisfiability of the whole formula but only increases its likelihood. Due to space limitations, only some of the main ideas for some of the proofs are presented here. The complete proofs will appear in the full paper [8].

2 Definitions and Simplifying Reductions

Let $X = \{x_1, x_2..., x_n\}$ be the set of n input variables and let the input space I^n be the set $\{0,1\}^n$. We assume throughout the paper that the

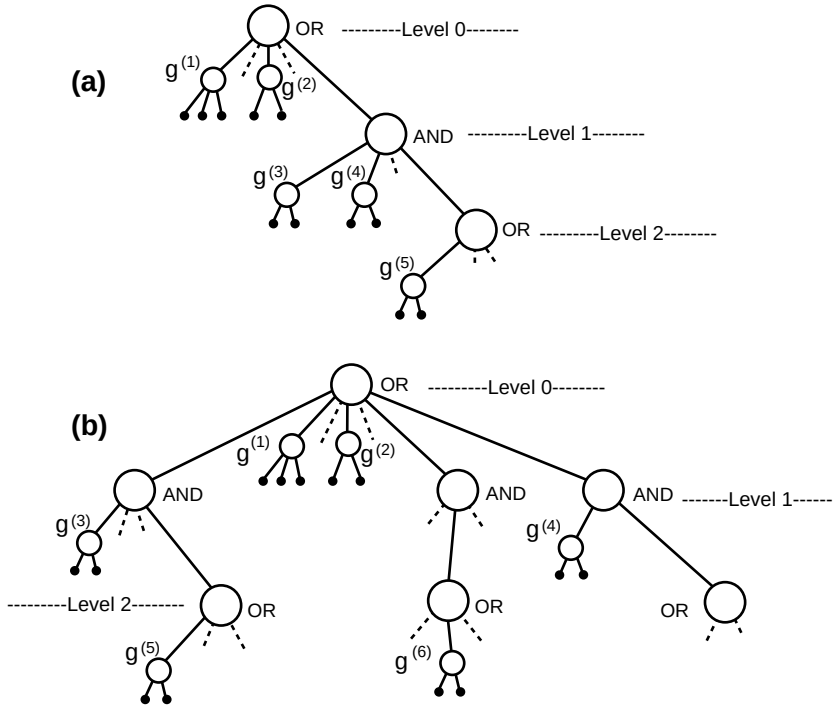

Figure 1. A perceptron decision list (a) and its generalization (b). Input units are represented as black dots, perceptrons as small circles, and OR/AND gates as large circles.

distribution D generating the examples is uniform on I^n.

We denote by $P(A)$ the probability of an event A, and by $P(A|B)$ the conditional probability of an event A given the fact that event B has been observed. All probabilities are taken with respect to D.

Let f be a boolean function defined on X. The *influence* of a variable x_i on f, denoted $Inf(x_i)$, is defined as:

$$Inf(x_i) \stackrel{\text{def}}{=} P(f = 1|x_i = 1) - P(f = 1|x_i = 0)$$

The *correlation* of a variable x_j with x_i with respect to f, denoted $C(x_i, x_j)$, is defined as

$$C(x_i, x_j) \stackrel{\text{def}}{=} \frac{P(f = 1|x_i = x_j = 1) - P(f = 1|x_i = 1, x_j = 0)}{P(f = 1|x_j = 1) - P(f = 1|x_j = 0)}$$

Intuitively, $C(x_i, x_j)$ reflects the effect of setting x_i to 1 on the influence of x_j. Note that, in general, $C(x_i, x_j) \neq C(x_j, x_i)$.

A perceptron g on X is specified by a vector of n weights w_i and a single threshold θ. For $\mathbf{x} = (x_1, x_2, ..., x_n) \in I^n$, we have:

$$g(\mathbf{x}) = \begin{cases} 1 & \text{if } \sum_{i=1}^{i=n} w_i x_i \geq \theta \\ 0 & \text{if } \sum_{i=1}^{i=n} w_i x_i < \theta \end{cases}$$

Two perceptrons are said to be nonoverlapping if they do not share any input variables. We are interested in the learnability of simple neural concepts built from nonoverlapping perceptrons with binary valued weights ($w_i = \pm 1$).

Deterministic Concepts

We generalize the notion of *decision lists*, introduced by Rivest [12], to *perceptron decision lists* (hereafter PDL). A PDL is a list of pairs:

$$(g^{(1)}, v^{(1)}), \ldots, (g^{(i)}, v^{(i)}), \ldots, (g^{(r)}, v^{(r)})$$

where each $g^{(i)}$ is a perceptron and $v^{(i)}$ is a value in $\{0, 1\}$. The last perceptron $g^{(r)}$ is the constant function $+1$. This defines a function $f : I^n \rightarrow \{0, 1\}$ as follows: for any $\mathbf{x}$, $f(\mathbf{x})$ is defined to be equal to $v^{(j)}$ where j is the least index for which $g^{(j)}(\mathbf{x}) = 1$. Compared to Rivest's decision lists, PDLs have the same structure but the complexity of the decision allowed at each node is greater.

A PDL can be converted to an equivalent feedforward neural net in an obvious manner. The result is a network of *alternating levels* of disjunction (OR) and conjunction (AND) of perceptrons (fig. 1a). A possible generalization of PDL is to allow multiple AND/OR nodes at each level (fig. 1b). Such networks represent arbitrary boolean formulas over the basis {AND,OR} whose inputs are perceptrons. We call such networks (functions) *generalized* PDL. If each variable appears at most once, the PDL is referred to as *nonoverlapping perceptron decision list* (hereafter NPDL). Similarly, we refer to generalized PDLs with the same property as generalized NPDLs. The equivalent network can be then viewed as a rooted tree. The root is the *output node* and each leaf in the tree is labeled with an input variable in a way such that no variable appears on more than one leaf. We will refer to the AND/OR nodes as *gates*.

We will assume w.l.o.g. that the input variables feed the gates only through perceptrons (may be single-variable perceptrons) and that each gate has, as input, at least one multi-variable perceptron or another gate (otherwise, it is a perceptron).

The *parent* of a variable is the perceptron to which the variable is an immediate input. We say perceptron g (or gate Γ) is an *uncle* of variable x_i if g (respectively, Γ) is an immediate input to a gate fed by x_i, but g (respectively, Γ) is not itself fed by x_i.

The *depth* of a gate Γ (or a variable x_i) is the number of gates on the path from Γ (respectively, x_i) to the output gate. We say x_i is a *zero-level* variable *with respect to* Γ if its parent is an immediate input to Γ.

The *least common ancestor* of two variables x_i and x_j, denoted lca(x_i,x_j), is the deepest gate fed by both x_i and x_j. We say two variables are *siblings* if they share a common parent (perceptron), and not siblings otherwise.

Probabilistic Concepts

A probabilistic concept (or p-concept), as defined by Kearns and Schapire [10] and Yamanishi [14], is a mapping $c : I^n \rightarrow [0, 1]$. For each $\mathbf{x} \in I^n$, $c(\mathbf{x})$ is interpreted as the probability that $\mathbf{x}$ is a positive example of the p-concept c. Thus, in the p-concept model, a labeled example is generated as follows: first, an instance $\mathbf{x}$ is chosen according to the target distribution on I^n; then, with probability $c(\mathbf{x})$, the labeled example $< \mathbf{x}, 1 >$ is observed, and with probability $1 - c(\mathbf{x})$, the labeled example $< \mathbf{x}, 0 >$ is observed. Thus, the learning algorithm has no direct access to c: the only access it has is through labeled examples $< \mathbf{x}, \sigma >$, where $\sigma \in \{0, 1\}$.

The p-concept we will investigate is what we call *probabilistic majority* of nonoverlapping perceptrons (with binary weights), defined as

$$c(\mathbf{x}) = \sum_{s=1}^{r} p_s g^{(s)}(\mathbf{x}) + p_0 \tag{2.1}$$

where $p_s \geq 0$ for $s = 0, \ldots, r$, and $0 \leq \sum_{s=0}^{r} p_s \leq 1$. Such p-concepts have a simple neural representation: a two-layer network with r hidden perceptrons and one (probabilistic) output unit that outputs 1 with a probability equal to the weighted sum of its inputs.

3 Learning Models

For learning the deterministic concepts, we adopt the PAC model [13] specified to the uniform distribution. Here the methodology is to draw a sample of a certain size labeled according to the unknown target function f and then to find a "good" approximation f' of f. The error of the hypothesis function f', with respect to the target f, is defined to be $P(f' \neq f) = P_{x \in D}(f'(\mathbf{x}) \neq f(\mathbf{x}))$. More specifically, a class of concepts F is said to be PAC learnable from examples under the uniform distribution D if there is an algorithm A such that, for the distribution D, for every $f \in F$, and any $0 < \epsilon, \delta < 1$, A runs in time polynomial in $(n, 1/\epsilon, 1/\delta)$ and outputs an hypothesis f' such that, with probability at least $(1 - \delta)$: $P(f' \neq f) < \epsilon$.

As in [6], we view the problem of learning a p-concept c as that of inferring a good approximation of c itself. This is called in [10] *learning with*

a model of probability. Specifically, let C be the class of p-concepts. Then C is said to be learnable with a model of probability under the uniform distribution D if there is an algorithm A such that, for the distribution D, for any $c \in C$, and any $0 < \epsilon, \delta < 1$, A runs in time polynomial in $(n, 1/\epsilon, 1/\delta)$ and outputs a real-valued hypothesis c' such that, with probability at least $(1 - \delta)$: $E_{\mathbf{x} \in D}|c'(\mathbf{x}) - c(\mathbf{x})| < \epsilon$.

4 Learning Nonoverlapping Perceptron Decision Lists

In this section, we describe a polynomial time algorithm for learning NPDLs with binary weights from random examples drawn according to the uniform distribution D. Actually, what we will be learning is the equivalent network of the NPDL (fig. 1a). The next section will provide the necessary additional processing to handle generalized NPDLs (fig. 1b).

We assume w.l.o.g. that the target network contains no two successive AND or two successive OR gates, as such nodes can always be merged. We also assume w.l.o.g. that the target network has the maximum possible number of perceptrons, the minimum possible number of weights, and it contains negative weights only for those inputs that lead directly from input variables. The learning algorithm proceeds in three steps:

1. In the first step, the algorithm determines the "relevant" input variables and the values (signs) of their weights. To achieve this, for each variable x_i, the algorithm estimates its *influence* $Inf(x_i)$ using examples drawn randomly according to D. We prove that if the variable x_i is relevant, then its influence is significantly greater or less than zero, depending on whether $w_i = 1$ or $w_i = -1$. If the influence is too small, the algorithm concludes that the variable is "irrelevant" and so, neglects it in later stages. Once the weights are estimated, the algorithm reduces the target function to a monotone NPDL by simply changing x_i to $1 - x_i$ whenever $w_i = -1$.

2. In the second step, the algorithm infers the architecture of the network, *i.e.* determines which variables appear in a given level, and among those, which are siblings. To do this, the algorithm estimates the correlation $C(x_i, x_j)$ of each ordered pair of variables using examples drawn randomly according to D. We show that these correlations contain enough information to infer the architecture of the network.

3. In the last step, the algorithm estimates a threshold value for each perceptron. We will see below that the correlations provide enough information to do that.

Intuition suggests that the influence of a variable is positive (negative) if its weight is positive (negative). The following lemma strengthens this

intuition by showing that there is a measurable *gap* between the two cases. This gap is used to estimate the weight values (signs).

Lemma 1. *Let f be a NPDL. Let g be a perceptron in f and let $x_i \in g$. Let $\{\lambda^{(s)}\}$ be the set of x_i's uncles and let $\{f^{(s)}\}$ be the set of subformulas computed by these uncles. Let $a^{(s)} = 1$ if $\lambda^{(s)}$ feeds immediately an AND gate and $a^{(s)} = 0$ if $\lambda^{(s)}$ feeds immediately an OR gate. Then, if $P(g = 1)$, $P(g = 0) > \rho$ and $\prod_s P(f^{(s)} = a^{(s)}) > \gamma$,*

$$Inf(x_i) \begin{cases} > & \frac{\gamma\rho}{n+2} & \text{if} \quad w_i = 1 \\ < & -\frac{\gamma\rho}{n+2} & \text{if} \quad w_i = -1 \end{cases}$$

Proof idea: First note that:

$$Inf(x_i) = \prod_s P(f^{(s)} = a^{(s)})[\ P(g(\mathbf{x}) = 1 | x_i = 1) - P(g(\mathbf{x}) = 1 | x_i = 0)\ ]$$

The lemma then follows from Bahadur's expansion [1].□

Note that we can assume w.l.o.g. that $\rho,\ \gamma > \epsilon/2n$, for otherwise we can neglect perceptron g without introducing much error. Under this assumption, the gap is wide enough to be estimated efficiently using a sample of polynomial size. Once we determine the weight values, we reduce f to a monotone NPDL by changing x_i to $1 - x_i$ whenever $w_i = -1$. In the following, we concentrate on the monotone case.

The next step is to infer the architecture of the network. It turns out that the *correlation measure* contains enough information to determine which variables appear at a given level, and among those, which are siblings. First, some facts about the correlations.

Lemma 2. *Let f be a monotone NPDL. Let x_i and x_j be two influential variables in f. Assume that $x_i \in g$. Then*

1. *If x_i and x_j are siblings:*

$$C(x_i, x_j) = C_I = \frac{P(g = 1 | x_i = x_j = 1) - P(g = 1 | x_i = 1, x_j = 0)}{P(g = 1 | x_j = 1) - P(g = 1 | x_j = 0)}\ .$$

2. *If x_i and x_j are not siblings, x_i is not deeper than x_j, and lca(x_i,x_j) is an OR:*

$$C(x_i, x_j) = C_{II} = \frac{1 - P(g = 1 | x_i = 1)}{1 - P(g = 1)}\ .$$

3. *If x_i and x_j are not siblings, x_i is not deeper than x_j, and lca(x_i,x_j) is an AND:*

$$C(x_i, x_j) = C_{III} = \frac{P(g = 1|x_i = 1)}{P(g = 1)} \ .$$

Moreover, if g contains p variables and has a renormalized threshold v:

$$C_I = 2\frac{v-1}{p-1} \ ; \ C_{II} = 1 - \frac{\left(\begin{array}{c} p-1 \\ v-1 \end{array} \right)}{\sum_{i=0}^{v-1} \left(\begin{array}{c} p \\ i \end{array} \right)} \ ; \ C_{III} = 1 + \frac{\left(\begin{array}{c} p-1 \\ v-1 \end{array} \right)}{\sum_{i=v}^{p} \left(\begin{array}{c} p \\ i \end{array} \right)} \quad (4.1)$$

$$C_I - C_{II} \geq \frac{1}{p^2} \geq \frac{1}{n^2} \qquad \text{for} \quad v \geq 2 \quad (4.2)$$

$$C_I - C_{III} \leq -\frac{1}{2p} \leq -\frac{1}{2n} \qquad \text{for} \quad v \leq p-1 \quad (4.3)$$

Proof idea: The different expressions of $C(x_i, x_j)$ can be derived using the inclusion-exclusion property and the fact that perceptrons in f do not share any variables. The derivation of eq. 4.1 is straightforward. Eqs. 4.2 and 4.3 follow using Bahadur's expansion [1].□

Note that the correlation gaps established in this lemma are wide enough to be estimated using a polynomial number of examples.

Let $X' \subseteq X$. We call two variables x_i and x_j *OR-potential-siblings* (w.r.t. X') if $C(x_i, x_j) \geq C(x_i, x_k)$ for all $x_k \in X'$, and *AND-potential-siblings* (w.r.t. X') if $C(x_i, x_j) \leq C(x_i, x_k)$ for all $x_k \in X'$. The following lemma enables us to determine which variables appear in a given level, and among those, which are siblings.

Lemma 3. *Let f be a monotone NPDL. Let Γ be a gate in f and let $X' \subseteq X$ be the set of all variables that feed Γ. Let $x_i \in X'$. Then, x_i is not a zero-level variable with respect to Γ iff there exist x_j, $x_k \in X'$ such that, for some permutation $\{i_1, i_2, i_3\}$ of $\{i, j, k\}$:*
x_{i_1} and x_{i_2} are OR-potential-siblings (w.r.t. X'),
x_{i_2} and x_{i_3} are OR-potential-siblings (w.r.t. X'), but
x_{i_1} and x_{i_3} are not OR-potential-siblings (w.r.t. X').

Moreover, if x_i is a zero-level variable with respect to Γ, and x_i and x_j are OR-potential-siblings (w.r.t. X'), then x_i and x_j are siblings in f.
The same holds if we replace OR by AND in the lemma.

Proof: We need to consider the different possible situations and apply the facts established in lemma 2. The details will appear elsewhere [8].□

Assume that the output gate Γ is an OR. To decide whether a variable x_i is a zero-level w.r.t. Γ, we determine the set of its OR-potential-siblings (w.r.t to X), call it T_i. If x_i is a zero-level variable, then every pair of variables in T_i will themselves be OR-potential-siblings. But if x_i is not a zero-level variable, then some pair in T_i will not be OR-potential siblings. There may be several different sets of zero-level siblings determined in this manner, which will form the various perceptrons that feed *immediately* the output gate [1]. A similar process applies if the output gate is an AND, using the AND-potential-siblings technique. We repeat this recursively, removing from X the variables already used and noting that the gates switches from OR to AND (or vice versa).

The last step of the algorithm is to estimate the threshold of each (monotone) perceptron g. Two cases are possible:

case 1: g is a single-variable perceptron. Then the only possible threshold value is 1.

case 2: g is a multi-variable perceptron. Let p be the number of variables in g and let v be its threshold. Let x_i, $x_j \in g$. Then:

$$C(x_i, x_j) = C_I = 2\frac{v-1}{p-1}$$

Thus, a good estimation of $C(x_i, x_j)$ yields a good estimation of the threshold v (and hence the original threshold).

Theorem 4. *The class of NPDL with binary weights are PAC learnable under the uniform distribution. The sample complexity is* $O(\frac{n^8}{\epsilon^4} log(n^2/\delta))$ *and the time complexity is* $O(\frac{n^{10}}{\epsilon^4} log(n^2/\delta))$.

5 Learning Generalized Nonoverlapping Perceptron Decision Lists

We extend the results of the previous section to handle the case where the target function f is a generalized NPDL (fig. 1b). The basic ideas behind the algorithm are just like those of the preceding section. First, we note that lemmas 1, 2, and 3 hold also for generalized NPDLs. In fact, the problem of learning generalized NPDLs presents only one difficulty that is not encountered in learning NPDLs: because a given level in the network may contain more than one gate, we need to determine not only which variables appear at that level and which are siblings, but also which variables

[1] This argument can be phrased in terms of type-graphs that are usually used for read-once formulas.

appear in *same* subtree rooted at one of the gates of that level. This extra difficulty can be solved fairly easily. The following subroutine, when called with a set X' and a variable x_i, returns the set $S \subseteq X'$ of all variables that feed some AND gate fed by x_i:

AND-test:

1. Set $S = \{x_i\}$.
2. For each variable $x_j \in X'$ and $x_j \notin S$:
   If there exists $x_k \in S$ such that $C(x_k, x_j) + C(x_j, x_k) > 2$, set $S = S \cup x_j$ and Go to 2.
3. If no variable can be added, return S.

Likewise, the following subroutine, when called with a set X' and a variable x_i, returns the set $S \subseteq X'$ of all variables that feed some OR gate fed by x_i:

OR-test:

1. Set $S = \{x_i\}$.
2. For each variable $x_j \in X'$ and $x_j \notin S$:
   If there exists $x_k \in S$ such that $C(x_k, x_j) + C(x_j, x_k) < 2$, set $S = S \cup x_j$ and Go to 2.
3. If no variable can be added, return S.

The intuition behind the above subroutines is that if two variables meet at an AND gate, setting one of them to 1 increases the influence of the other, whereas if they meet at an OR gate, setting one of them to 1 decreases the influence of the other. The following lemma strengthens this intuition by showing that there is measurable gap between the two cases.

Lemma 5. *Let f be a monotone generalized NPDL. Let x_i and x_j be two variables in f. Then, if x_i and x_j are not siblings*

$$C(x_i, x_j)+C(x_j, x_i) > 2+\frac{1}{2}\min(Inf(x_i), Inf(x_j)) \quad \text{if } lca(x_i, x_j) \text{ is an AND}$$

$$C(x_i, x_j)+C(x_j, x_i) < 2-\frac{1}{2}\min(Inf(x_i), Inf(x_j)) \quad \text{if } lca(x_i, x_j) \text{ is an OR}$$

Proof: Will appear elsewhere [8].□

Assume w.l.o.g. that the output gate Γ is an OR (fig. 1b). The zero-level variables w.r.t. Γ are determined as in the previous section, using the

OR-potential-siblings technique. These variables are then removed and the **AND-test** is invoked to determine which set of variables feed the *same* AND gate in the next level. There may be several different sets determined in this manner; each set S_i will be assigned an AND gate Γ_i in that level. Then, for each set S_i we use the AND-potential-siblings technique to determine which variables are zero-level w.r.t. Γ_i. This may yield several different subsets, which will form the various perceptrons that feed *immediately* the gate Γ_i. We repeat this subdivision process recursively, removing the variables already used and noting that the gates switches from OR to AND (or vice versa). The threshold of each perceptron can be estimated by the same method used in the previous section.

Finally, the sample and time complexities are the same as that for NPDLs.

Theorem 6. *The class of generalized NPDL with binary weights are PAC learnable under the uniform distribution. The sample complexity is of the* $O(\frac{n^8}{\epsilon^4} log(n^2/\delta))$ *and the time complexity is of* $O(\frac{n^{10}}{\epsilon^4} log(n^2/\delta))$.

5.1 Learning Probabilistic Majorities of Nonoverlapping Perceptrons

Recall that, when learning a p-concept, we have no direct access to c: the only access we have is through labeled examples $< \mathbf{x}, \sigma >$. So in what follows, the different estimations are taken with respect to the label σ.

Let the target c be a probabilistic majority of nonoverlapping perceptrons defined by (2.1). We are interested in learning, with a model of probability, the equivalent neural network of c. The learning algorithm proceeds in four steps:

1. Estimating the weight values using the influences. Once this is done, the target p-concept is reduced to its monotone form by simply changing x_i to $1 - x_i$ whenever $w_i = -1$.
2. Inferring the architecture, *i.e.* which variables are siblings (appear in the same perceptron).
3. Estimating the threshold of each perceptron.
4. Estimating the different probabilities $p_0, p_1, \ldots, p_r$.

The following lemma establish that there exists a measurable gap in terms of the influence between the two cases $w_i = 1$ and $w_i = -1$.

Lemma 7. *Let c be a probabilistic majority of nonoverlapping perceptrons as in eq. 2.1. Let $g^{(s)}$ be a perceptron in c and let $x_i \in g^{(s)}$. Then if*

$P(g^{(s)} = 1)$, $P(g^{(s)} = 0) > \rho$, *and* $p_s > \gamma$,

$$Inf(x_i) \begin{cases} > \frac{\gamma\rho}{n+2} & \text{if } w_i = 1 \\ < -\frac{\gamma\rho}{n+2} & \text{if } w_i = -1 \end{cases}$$

Again, we can assume w.l.o.g. that ρ, $\gamma > \epsilon/2n$, for otherwise we can neglect perceptron $g^{(s)}$ without introducing much error. Under this assumption, the gap is wide enough to be estimated efficiently using a sample of polynomial size.

Once we have an estimate of the weight values, we reduce c to its monotone form by changing x_i to $1 - x_i$ whenever $w_i = -1$. The next step is to infer the architecture of the monotone p-concept c.

Lemma 8. *Let c be a monotone probabilistic majority of nonoverlapping perceptrons. Let g be a (monotone) perceptron in c with p variables and a threshold v. Let $x_i \in g$ and let x_j be another variable in c. Then*

$$C(x_i, x_j) = \begin{cases} 2\frac{v-1}{p-1} & \text{if } x_j \in g \\ 1 & \text{if } x_j \notin g \end{cases}$$

One implication of this lemma is that as long as g is not a majority function ($v \neq (p+1)/2$), we can determine whether or not x_i and x_j are siblings. But if g is a majority function ($v = (p+1)/2$), estimating $C(x_i, x_j)$ yields no information on whether or not x_i and x_j are actually siblings [2]. To overcome this difficulty, we introduce a new measure of correlation that depends on triples of variables. More precisely, we define the correlation of a variable x_k with two variables x_i and x_j, denoted $C(x_i, x_j, x_k)$, as

$$C(x_i, x_j, x_k) \stackrel{\text{def}}{=} \frac{P(\sigma = 1 | x_i = x_j = x_k = 1) - P(\sigma = 1 | x_i = x_j = 1, x_k = 0)}{P(\sigma = 1 | x_k = 1) - P(\sigma = 1 | x_k = 0)}$$

The following lemma shows that this new correlation measure solves our problem.

Lemma 9. *Let c be a monotone probabilistic majority of nonoverlapping perceptrons. Let x_i, x_j, and x_k be three variables in c and assume that the parents of these variables compute majority functions. Assume that $x_i \in g$ where g is a perceptron with p variables. Then*

$$C(x_i, x_j, x_k) = \begin{cases} 1 - \frac{1}{p-2} & \text{if } x_j, x_k \in g \\ 1 & \text{otherwise} \end{cases}$$

[2] This is related to the fact that the amplification function is independent of the level at which the two variables meet[6].

To infer the architecture, we first estimate the correlation $C(x_i, x_j)$ of each (ordered) pair of variables. Whenever $C(x_i, x_j)$ is strictly different from one, we conclude that x_i and x_j are siblings. For the unsolved cases, we estimate the correlation $C(x_i, x_j, x_k)$ of each (ordered) triple of variables. We conclude that x_i, x_j, and x_k are siblings if $C(x_i, x_j, x_k)$ is strictly less than one, and not siblings otherwise.

The next step is to estimate the threshold of each perceptron. This can be done exactly as in section 4. The final step is to estimate the different probabilities $p_0, \ldots, p_r$. For this, note that

$$p_s \;=\; P(\sigma = 1 | g^{(s)} = 1) - P(\sigma = 1 | g^{(s)} = 0) \quad \text{for} \quad s = 1, \ldots, r.$$

$$p_0 \;=\; P(\sigma = 1) - \sum_{s=1}^{r} p_s P(g^{(s)} = 1)$$

Since we have already determined the different perceptrons $g^{(1)} \ldots, g^{(r)}$, the different probabilities $p_0, p_1, \ldots, p_r$ can be estimated easily using the above expressions.

Theorem 10. *The class of probabilistic majorities of nonoverlapping perceptrons with binary weights is learnable with a model of probability under the uniform distribution. The sample complexity is* $O(\frac{n^8}{\epsilon^4} log(n^2/\delta))$.

6 Conclusion

As we will outline in the full paper, the techniques developed in the previous sections can be extended to learn other (deterministic/probabilistic) nonoverlapping neural concepts on the uniform distribution.

The general class of nonverlapping perceptron networks has recently been shown to be PAC learnable from examples and membership queries [7]. It is still an open problem whether this class is PAC learnable from examples only on the uniform distribution. A more tractable problem would be to extend the techniques of this paper to handle the class of nonverlapping perceptron networks with binary weights and arbitrary thresholds.

Finally, it is important to investigate if these techniques can be made to work when the requirement of nonoverlapping is relaxed by going to the 2μ or read-twice case.

Acknowledgments

We thank Thomas Hancock for helpful comments on earlier drafts of this paper. Work supported by NSERC grant OGP0122405.

Bibliography

1. Bahadur R.,"Some Approximations to the Binomial Distribution Function", *Annals Math. Stat.*, Vol.31, (1960), 43–54.

2. Blum A. and Rivest R.L., "Training a 3-node neural network is NP-complete", in *Proc. of the 1st Workshop on Computational Learning Theory*, Morgan Kaufman, pp. 9–18, 1988.

3. Lin J.H. and Vitter J.S., "Complexity results on learning by neural nets", *Machine Learning*, *Vol. 6*, pp. 211–230, 1991.

4. Kearns M., Li M., Pitt L., and Valiant L., "On the learnability of boolean formulas", in *Proceedings of the 9th Annual ACM Symposium on Theory of Computing*, New York, NY, 1987.

5. Pagallo G. and Haussler D., "A greedy method for learning μDNF functions under the uniform distribution". Technical Report UCSC-CRL-89-12, Santa Cruz: Dept. of Computer and Information Science, University of California at Santa Cruz, 1989.

6. Schapire R.E., *The Design and Analysis of Efficient Learning Algorithms*, Cambridge MA: MIT Press, 1992.

7. Hancock T., Golea M., and Marchand M., "Learning Nonoverlapping Perceptron Networks From Examples and Membership Queries", *To appear* in *Machine Learning*.

8. Golea M., Marchand M., and Hancock T.R., "On Learning μ-Perceptron Networks On the Uniform Distribution", submitted to *Neural Networks*.

9. Golea M., Marchand M., and Hancock T.R., "On Learning μ-Perceptron Networks with Binary Weights", *Advances in Neural Information Processing Systems*, *Vol. 5*, pp. 591–598, 1993.

10. Kearns M. and Schapire R.E. " Efficient Distribution-free Learning of Probabilistic Concepts", in *Proceedings of the 31st Symposium on Foundations of Computer Science*, pp. 382, 1990.

11. Kearns M., "Efficient Noise-Tolerant Learning from Statistical Queries", AT&T manuscript 1992.

12. Rivest R.L., "Learning Decision Lists", *Machine Learning*, *Vol. 2*, pp. 229, 1987.

13. Valiant L.G., "A theory of the learnable", *Communications of the ACM*, *Vol. 27*, pp. 1134–1142, 1984.

14. Yamanishi K., "A Learning Criteria for Stochastic Rules", *Machine Learning*, vol. 9, 165–203, 1992.