Identification Criteria and Lower Bounds for Perceptron-like Learning Rules

Michael Schmitt
Institute for Theoretical Computer Science
Technische Universität Graz
Klosterwiesgasse 32/2
A–8010 Graz, Austria
E-mail: mschmitt@igi.tu-graz.ac.at

Abstract

Perceptron-like learning rules are known to require exponentially many correction steps in order to identify Boolean threshold functions exactly. We introduce criteria that are weaker than exact identification and investigate whether learning becomes significantly faster if exact identification is replaced by one of these criteria: PAC identification, order identification, and sign identification. PAC identification is based on the learning paradigm introduced by Valiant and known to be easier than exact identification. Order identification uses the fact that each threshold function induces an ordering relation on the input variables which can be represented by weights of linear size. Sign identification is based on a property of threshold functions known as unateness and requires only weights of constant size. We show that Perceptron-like learning rules cannot satisfy these criteria when the number of correction steps is to be bounded by a polynomial. We also present an exponential lower bound for order identification with the learning rules introduced by Littlestone. Our results show that efficiency imposes severe restrictions on what can be learned with local learning rules.

1 Introduction

Perceptron-like learning rules are among the best investigated learning algorithms for neural networks. Although they are known for quite a long time they have not lost much of their attraction. This may be due to several appealing features that make Perceptron learning appear advantageous—at least when training single neuron architectures: Perceptron-like rules are easy to implement because the weight updates are based on simple operations, i.e. addition or subtraction of a constant value. The rules are local to the extent that the only information that is required at each weight is the input at this weight and the sign of the error. Further, Perceptron-like rules satisfy the definition of on-line learning in a very strong

sense: The corrections are based on single, misclassified examples and learning may start from an arbitrary initial weight vector. The latter property makes Perceptron learning even tolerate temporary classification errors or follow changing targets. Finally, the famous Perceptron convergence theorem states that any set of training examples that has a solution vector will be classified correctly after a finite number of correction steps.

These advantages, however, are contrasted by results concerning the efficiency of learning. Complexity considerations make Perceptron learning appear less favorable. The number of correction steps is known to grow in the worst case exponentially in the number of weights. On the other hand, there are training procedures based on algorithms for linear programming that accomplish learning in polynomial time. Thus, the only way to prevent learning from being slow is to abandon Perceptron rules and turn towards those more advanced training procedures. As far as we know, however, there is no single efficient training procedure that includes all the positive characteristics shared by Perceptron-like rules.

The use of simpler learning procedures seems to be possible if one weakens the demands on the learning process. In this paper we explore this possibility by investigating three criteria that do not require that learning generates weights that exactly represent the target function. One is rather content with a result that is somehow close to it.

The first criterion, which is called PAC identification, is based on Valiant's PAC learning paradigm. Exact identifiability in polynomial time implies PAC identifiability, but the converse is not true. It is known that there are classes of functions that can be PAC identified in polynomial time but require exponential time for exact identification. Therefore, it is a good starting point to investigate if the PAC criterion can be met by Perceptron-like rules. In PAC learning, one assumes that the training examples are drawn according to a fixed but unknown probability distribution. After completion of the learning process the result is tested on a further example drawn according to the same distribution. It is required that for all distributions with high probability (with regard to the training examples) the probability of an error (with regard to the test example) is low. We will show in Section 3 that Perceptron-like rules fail to meet this criterion in polynomial time by exhibiting small sets of training examples that require large weights.

With order and sign identification we avoid the necessity to make use of large weights. Order identification just requires to represent the relationship between the magnitudes of the weights. Each Boolean threshold function induces an ordering on the input variables that corresponds to the ordering of the absolute values of its weights. The learning process is successful if it yields a function with the same ordering. Thus, learning is not forced to produce large weights because such an ordering can be represented by weights with an absolute value not larger than the number of weights. Nevertheless, in Section 4 we present exponential lower bounds for order identification with Perceptron-like learning rules.

Finally, the criterion of sign identification is based on a monotonicity property of Boolean threshold functions. The learning process is required to produce a weight vector that has the same signs as the weight vector of the target function. In order

for the signs to be unique however, we have to restrict the possible targets to non-degenerate functions, i.e. those that do not allow zero weights. Sign identification can be considered a further weakening compared to order identification because sign identity can be represented by constant weights, say $\{-1, 1\}$, whereas order identification may require as many different values as there are weights. However, it turns out that the criterion is not weak enough. In Section 5 we show that Perceptron-like learning rules cannot sign identify non-degenerate threshold functions in polynomial time.

In Section 6 we investigate the learning rules introduced by Littlestone. In contrast to the rules considered so far, they are based on a multiplicative update of the weights but they have some features in common with Perceptron-like rules. In particular, exponential lower bounds for exact identification are known for Littlestone's rules as well. We show that they also do not satisfy the criterion of order identification.

The basic concepts of this paper are introduced in the following section. In it we also review some results concerning McCulloch-Pitts neurons and Perceptron-like learning rules and give references to related work.

2 Preliminaries

A *McCulloch-Pitts neuron* with n inputs is characterized by a vector of $n+1$ real numbers consisting of weights $w_1, \ldots, w_n$ and threshold t (McCulloch and Pitts 1943). On binary inputs it computes a *Boolean threshold function* $f : \{0,1\}^n \to \{0,1\}$ defined as

$$f(x_1, \ldots, x_n) = 1 \quad \text{iff} \quad w_1 x_1 + \cdots + w_n x_n \geq t.$$

It is well known that any Boolean threshold function can be represented by integer weights and threshold (see e.g. Muroga 1971; Minsky and Papert 1988; Parberry 1994).

In order to investigate the complexity of learning with Perceptron-like rules we introduce the weight complexity of a threshold function.

Definition 1 *The* weight complexity *of a Boolean threshold function is defined as the smallest natural number such that the function can be represented by a weight vector with integer weights and threshold such that their absolute value is not larger than this number.*

The weight complexity of Boolean threshold functions where the truth values are represented as $\{0,1\}$ is known to be at most $2^{-(n+1)}(n+1)^{(n+3)/2} + 1/2$ (Schmitt 1994). For bipolar inputs $\{-1,1\}$ the more succinct bound $2^{-n}(n+1)^{(n+1)/2}$ has been shown (see also Håstad 1994; Parberry 1994). Several functions of weight complexity at least $2^{\Omega(n)}$ have been constructed (Muroga 1971; Minsky and Papert 1988; Parberry 1994). By a more involved proof, Håstad (1994) has defined a function that requires weights at least as large as $2^{\Omega(n \log n)}$ (see also Parberry 1994). Further,

it has been shown that a Boolean threshold function has weight complexity at least 1.4^n in the average (Hampson and Volper 1986).

To simplify notation we shall assume from now on that all threshold functions have threshold 0. This can be done without loss of generality by introducing a weight w_{n+1} with value $-t$ and constant input 1.

A *Perceptron-like learning rule* proposes how to modify a weight vector w that represents a *hypothesis function* g when given an *example* $\langle x, f(x)\rangle$ of a *target function* f with $f(x) \neq g(x)$. Let w' be the vector after the correction step. Then the rule is written as

$$w' \quad = \quad \begin{cases} w + c \cdot x & \text{if } g(x) = 0 \\ w - c \cdot x & \text{if } g(x) = 1. \end{cases}$$

Perceptrons trained by this method were originally known as α-Perceptrons (Rosenblatt 1958, for a historical perspective see Nagy 1991). The parameter c is an arbitrary scalar value greater than zero called *correction increment*. The initial weight vector may be chosen arbitrarily. The most popular variant of the Perceptron rule uses $w = (0, \ldots, 0)$ as initialization and $c = 1$ as correction increment.

The Perceptron convergence theorem guarantees that any set of training examples taken from a Boolean threshold function leads after finitely many correction steps to a weight vector that produces no further errors on the training set. A considerable number of proofs with subtle differences has been published (e.g. Novikoff 1962; Rosenblatt 1962; Lewis II 1966; Minsky and Papert 1988; Parberry 1994). Maass (1994) has derived an explicit upper bound in terms of the number of weights: At most $(n+1)^2 \cdot 2^{(n+1)\log(n+1)}$ corrections may occur.

Concerning lower bounds one easily concludes that the number of corrections cannot be smaller than the weight complexity. Lewis II (1966) has shown that at least $|\widetilde{w}|^2/(n+1)$ corrections are necessary, where $\widetilde{w}$ is an integer vector representing the target function such that $|\widetilde{w}|$ is minimal. Now $|\widetilde{w}|$, the Euclidean length of $\widetilde{w}$, is at least as large as the weight complexity of the function. Therefore, we obtain the following lower bound that slightly improves previous results.

Proposition 1 *To identify a Boolean threshold function with n inputs and weight complexity ω exactly requires at least $\omega^2/(n+1)$ many Perceptron correction steps.*

The above mentioned upper and lower bounds are valid for any correction increment $c > 0$, but they hold strictly speaking only if one chooses the zero vector as initialization. However, starting from any other fixed weight vector cannot improve the worst-case behavior of the learning process. Because we are interested in the coarse distinction between polynomial and exponential behavior we assume throughout the paper that Perceptron learning starts from the zero vector.

3 PAC Identification

It is obvious from the lower bounds on the weight complexity mentioned in the previous section that for Perceptron-like learning rules exact identification is too

strong a requirement to be accomplished in polynomial time. As a first step in search for a weaker criterion that can be met by Perceptron-like rules, we define the so-called PAC identification inspired by the learning paradigm of Valiant (1984). In this model the training examples are chosen from a target function independently according to an unknown probability distribution. The learning process must with probability at least $1 - \delta$ generate a hypothesis that differs from the target by at most ε where the error is determined according to the same distribution.

A class of target functions is said to be PAC learnable in polynomial time if the number of examples required is bounded by a polynomial in the relevant parameters (in the case of Boolean threshold functions in $n, 1/\epsilon$, and $1/\delta$) and if the learning algorithm runs in polynomial time (in the case of Perceptron rules at most polynomially in $n, 1/\epsilon$, and $1/\delta$ many correction steps). PAC learning is known to be easier than exact identification. Angluin (1988) has shown that any method for exact identification can be turned into a method for PAC identification under preservation of polynomial time. She has also presented an example of a target class that can be PAC identified in polynomial time but needs exponentially many examples for exact identification in the worst case. On the other hand, it is known that Boolean threshold functions *are* polynomially PAC learnable by methods of linear programming (Blumer *et al.* 1989). Therefore, it is worth considering if the algorithms for linear programming, which are quite complicated, can be replaced by the much simpler Perceptron-like rules such that PAC identification can still be achieved in polynomial time.

Definition 2 *Let a learning rule for the modification of weight vectors be given. The learning rule is said to* PAC identify *the class of Boolean threshold functions* in polynomial time *if there exist polynomials $p(n, r, s)$ and $q(n, r, s)$ such that for all $n \geq 1$, $0 < \varepsilon, \delta < 1$, and for all probability distributions P on $\{0,1\}^n$ the following holds: If $p(n, 1/\varepsilon, 1/\delta)$ examples are drawn from an arbitrary Boolean threshold function f independently according to P then with probability at least $1 - \delta$ the repeated application of the learning rule yields after at most $q(n, 1/\varepsilon, 1/\delta)$ corrections the weight vector of a function g such that $P[\{x : f(x) \neq g(x)\}] \leq \varepsilon$.*

In the definition we use the polynomial p to bound the number of examples and the polynomial q to bound the number of computation steps. For Boolean threshold functions it is known that a polynomial number of examples is sufficient for PAC learning (Blumer *et al.* 1989). Thus the existence of p is guaranteed. Consequently, in order to show that PAC identification cannot be achieved by a given algorithm, one has to prove that the number of computation steps cannot be bounded by a polynomial. In the proof of the following theorem we show that the number of correction steps for Perceptron rules does not decrease substantially if exact identification is replaced by PAC identification.

Theorem 1 *Perceptron-like learning rules cannot PAC identify the class of Boolean threshold functions in polynomial time.*

Proof. We consider the Boolean threshold function f represented by the vector $(a_1, \ldots, a_n, -a_{n+1})$ with weights $a_1, \ldots, a_n$ and threshold a_{n+1} where n is odd and a_i denotes the i-th Fibonacci number

$$a_i \;=\; a_{i-1} + a_{i-2}, \quad a_1 = a_2 = 1. \tag{1}$$

According to Hampson and Volper (1986) this function has weight complexity at least $\Omega(1.6^n)$, more precisely $\Omega(\phi^n)$ where ϕ is the golden ratio $\phi = (\sqrt{5}+1)/2$ (see also Parberry 1994). As the function can be written in the form

$$(\cdots(((x_1 \wedge x_2) \vee x_3) \wedge x_4)\cdots) \vee x_n$$

it belongs to the class of so-called nested functions defined by Anthony *et al.* (1995). It has been shown that any nested Boolean function f on n inputs has specification number $n+1$ (Anthony *et al.* 1995) which means that there exists a set $A \subseteq \{0,1\}^n$ of cardinality $n+1$ such that any Boolean function g that agrees with f on A is identical with f.

Assume that a Perceptron-like learning rule PAC identifies the class of Boolean threshold functions in polynomial time and let p and q be polynomials according to Definition 2. We define the distribution P on $\{0,1\}^n$ by $P(x) = 1/(n+1)$ if $x \in A$, and 0 otherwise. Further we let $\varepsilon = 1/(n+2)$ and choose an arbitrary, but fixed $\delta > 0$. We randomly draw $p(n, n+2, 2)$ examples and apply $q(n, n+2, 2)$ correction steps. Because $P(x) > \varepsilon$ for all $x \in A$ the resulting threshold function g must agree with f on A with probability at least δ. Thus, because f is nested, g must agree with f completely with the same probability. However, the largest weight produced can have absolute value at most $q(n, n+2, 2)$. This contradicts the fact that f has weight complexity at least $\Omega(\phi^n)$. □

We have chosen the distribution and the values for the parameters ε and δ such that the algorithm is forced to produce hypotheses that agree with all examples. This method is due to Pitt and Valiant (1988). The nested function f we have used is different from that in Anthony and Shawe-Taylor (1993). They show that the largest weight of their function has to be at least $(\sqrt{3})^{n/2-1}$, whereas for f the slightly better bound a_n is known (Parberry 1994).

4 Order Identification

The result in the previous section shows that not only exact but also PAC identification requires to generate large weights. Therefore, to be able to learn with Perceptron-like rules in polynomial time one has to look for identification criteria that avoid this obstacle. In this and the following section we introduce and analyze two criteria based on equivalence relations on Boolean threshold functions. The learning task is to identify the class of the target function by producing a representative of this class. The relation considered in this section is based on an ordering of the input variables and can be represented by weights of size at most n.

Definition 3 *Let $\beta : \{x_i, x_j\} \to \{0,1\}$ be a truth assignment of the variables x_i, x_j and $f : \{0,1\}^n \to \{0,1\}$ be a threshold function. Let $f_\beta : \{0,1\}^{n-2} \to \{0,1\}$ be the function obtained from f by fixing variable x_i to $\beta(x_i)$ and variable x_j to $\beta(x_j)$. The complementary assignment $\overline{\beta} : \{x_i, x_j\} \to \{0,1\}$ is defined as $\overline{\beta}(x) = 1 - \beta(x)$. Now let $\beta(x_i) = 1$ and $\beta(x_j) = 0$. We say that*

$$x_i \overset{f}{\succsim} x_j$$

is satisfied if and only if $f_\beta \geq f_{\overline{\beta}}$ holds (i.e. $f_\beta(x) \geq f_{\overline{\beta}}(x)$ for all x). The relations

$$x_i \overset{f}{\succ} x_j, \quad x_i \overset{f}{\approx} x_j$$

are defined correspondingly. Finally, we let $x_i \overset{f}{\approx} x_i$ always.

For any threshold function the relation defined above provides a total ordering of the variables. This is stated in the following lemma by Muroga (1971). It also establishes a relationship between the ordering of variables and the ordering of weights. We quote it without proof.

Lemma 1 (Muroga 1971, Theorem 5.1.4) *Let $(w_1, \ldots, w_n, -t)$ be a weight vector for the threshold function $f : \{0,1\}^n \to \{0,1\}$.*

(i) The variables of f are totally ordered with respect to $\overset{f}{\succsim}$.

(ii) If $x_i \overset{f}{\succ} x_j$, then $w_i > w_j$.

(iii) $x_i \overset{f}{\approx} x_j$ if and only if $(w_i + w_j)/2$ can be used in both w_i and w_j to give weights for f.

(iv) Conversely, if $w_i > w_j$ then $x_i \overset{f}{\succsim} x_j$, and if $w_i = w_j$ then $x_i \overset{f}{\approx} x_j$.

Now we define two functions to be equivalent if the relationships among their weights are identical. This implies that each equivalence class has a representative with small weights. Then we use the equivalence relation to define the criterion of order identification.

Definition 4 *Let $f, g : \{0,1\}^n \to \{0,1\}$ be threshold functions. Then f and g have the same variable order if for all $1 \leq i, j \leq n$*

$$x_i \overset{f}{\succ} x_j \quad \textit{iff} \quad x_i \overset{g}{\succ} x_j.$$

Corollary 1 *For each Boolean threshold function f there exists a Boolean threshold function g that has the same variable order as f and weight complexity at most n.*

Definition 5 *A learning rule for the modification of weight vectors is said to* order identify *the class of Boolean threshold functions* in polynomial time *if there exist polynomials $p(n)$ and $q(n)$ such that for all $n \geq 1$ and all threshold functions $f : \{0,1\}^n \to \{0,1\}$ the following holds: On all sequences of training examples for f of length at most $p(n)$ the repeated application of the learning rule yields after at most $q(n)$ corrections the weight vector of a function g that has the same variable order as f.*

The main result of this section shows that order identification requires exponentially many correction steps although weights of polynomial size are sufficient.

Theorem 2 *Perceptron-like learning rules cannot order identify the class of Boolean threshold functions in polynomial time.*

Proof. Let $f : \{0,1\}^{n+2} \to \{0,1\}$ be a threshold function represented by the weight vector $(1, 2, a_1, a_2, \ldots, a_n, -a_{n+1})$ where n is odd and a_i denotes the i-th Fibonacci number as defined in (1). Thus f is the function from the proof of Theorem 1 extended by two additional inputs. First we show that

$$x_1 \overset{f}{\prec} x_2 \tag{2}$$

holds. Let $\beta : \{x_1, x_2\} \to \{0,1\}$ be defined by $\beta(x_1) = 1$, $\beta(x_2) = 0$. Then f_β is represented by $w = (a_1, a_2, \ldots, a_n, 1 - a_{n+1})$ and $f_{\overline{\beta}}$ by $\overline{w} = (a_1, a_2, \ldots, a_n, 2 - a_{n+1})$. Obviously we have $f_\beta \leq f_{\overline{\beta}}$. To show $f_\beta \neq f_{\overline{\beta}}$ we consider the element $x \in \{0,1\}^n$ where $x_i = 1$ iff $i \in \{2, \ldots, n-1\}$. We then have

$$w_1 x_1 + \cdots + w_n x_n \;=\; \overline{w}_1 x_1 + \cdots + \overline{w}_n x_n .$$

and

$$w_1 x_1 + \cdots + w_n x_n \;=\; \sum_{i=2}^{n-1} a_i \;=\; (\sum_{i=1}^{n-1} a_i) - 1 \;=\; a_{n+1} - 2.$$

Thus $f_\beta(x) = 0$ and $f_{\overline{\beta}}(x) = 1$ are shown. For the last equality we have used the property $a_{n+1} = 1 + \sum_{i=1}^{n-1} a_i$ of the Fibonacci series.

Given a polynomial q, assume that a Perceptron-like learning rule order identifies f in at most $q(n+2)$ correction steps. We choose the set A of cardinality $n+1$ from the proof of Theorem 1 and add to each element a first and a second component, both containing 0. Adding inputs to a threshold function does not decrease its weight complexity. Hence, the construction of f implies that it has weight complexity at least $\Omega(\phi^n)$. Therefore, we can choose for each sufficiently large n a sequence of training examples from A such that the rule makes $q(n+2)$ corrections. Starting with the zero vector, the resulting weight vector $(w'_1, \ldots, w'_{n+3})$ satisfies $w'_1 = w'_2 = 0$ because the first two weights are not modified when using examples from A. According to Lemma 1(iii) the represented function g satisfies $x_1 \overset{g}{\approx} x_2$. Together with (2) this implies that the variable orders of f and g are different. □

The corrections made by the learning rule satisfy $x_1 \overset{g}{\approx} x_2$ for an exponential number of steps. Choosing a suitable initial vector one can also achieve that $x_1 \overset{g}{\succ} x_2$ holds during these steps.

5 Sign Identification

As a means to investigate the power of Perceptron-like learning rules we finally introduce sign identification. It is weaker than order identification in the sense that the learning process is just required to produce a weight vector that agrees with the target function on the signs of the weights. Thus the result can be represented by weights of constant size, e.g. $\{-1, 1\}$. For this criterion to make sense, however, we have to restrict the class of target functions to those that cannot be represented by vectors containing 0's as weights. We call such a function according to Parberry (1994) *non-degenerate*. The threshold is excluded from this restriction. A threshold function that has 0 as a weight does not depend on this input, hence the weight can also be an arbitrary small positive or an arbitrary small negative real number. But then the sign of the weight is not unique. The uniqueness of the signs of non-degenerate threshold functions is guaranteed by a property known as unateness.

Definition 6 *A Boolean function is said to be* positive in the variable x_i *if it has a representation in disjunctive normal form where x_i does not occur negated. It is called* negative in x_i *if it has a disjunctive normal form where x_i occurs only negated. A function f is* unate *if for each $1 \leq i \leq n$ f is positive or negative in x_i.*

Paull and McCluskey (1960) have shown that all threshold functions are unate. For non-degenerate threshold functions they have also established a relationship between the property of being positive, resp. negative, in x_i and the sign of weight w_i. We quote the result, which can also be found in McNaughton (1961), without proof.

Lemma 2 (Paull and McCluskey, Jr. 1960, Theorem 1) *Every Boolean threshold function is unate. Let $f : \{0,1\}^n \to \{0,1\}$ be non-degenerate and $(w_1, \ldots, w_{n+1})$ a weight vector representing f. Then we have for $1 \leq i \leq n$: f is positive in x_i iff $w_i > 0$, and f is negative in x_i iff $w_i < 0$.*

Unateness implies that the signs of the weights of a non-degenerate threshold function are unique. For sign identification it is sufficient to achieve agreement of signs.

Definition 7 *A learning rule for the modification of weight vectors is said to* sign identify *the class of non-degenerate Boolean threshold functions* in polynomial time *if there exist polynomials $p(n)$ and $q(n)$ such that for all $n \geq 1$ and all threshold functions $f : \{0,1\}^n \to \{0,1\}$ the following holds: On all sequences of training examples for f of length at most $p(n)$ the repeated application of the learning rule yields after at most $q(n)$ corrections the weight vector $(u_1, \ldots, u_{n+1})$ of a function g such that there exists a weight vector $(w_1, \ldots, w_{n+1})$ for f that satisfies*

$$u_i \cdot w_i > 0 \quad \text{for } i = 1, \ldots, n. \tag{3}$$

We remark that if (3) holds for a weight vector $(w_1, \ldots, w_{n+1})$ then it holds for all weight vectors representing f due to Lemma 2.

Theorem 3 *Perceptron-like learning rules cannot sign identify the class of non-degenerate Boolean threshold functions in polynomial time.*

Proof. As in the proof of Theorem 2 we use the principle of leaving certain weights untouched as long as possible. Let $f : \{0,1\}^{n+1} \to \{0,1\}$ be a threshold function represented by the weight vector $w = (1, a_1, a_2, \ldots, a_n, -a_{n+1})$ where n is odd and a_i denotes the i-th Fibonacci number as in (1). Thus f is the function from the proof of Theorem 1 with one additional input. First we show that f is non-degenerate. Let $x \in \{0,1\}^{n+1}$ denote the element where $x_i = 1$ iff $i \leq n$, and for $j = 1, \ldots, n$ let $x^{(j)}$ denote the element that differs from x exactly in the j-th component. Then we have

$$w_1 x_1 + \cdots + w_{n+1} x_{n+1} = 1 + \sum_{i=1}^{n-1} a_i = a_{n+1}.$$

Therefore $f(x) = 1$. On the other hand for $j = 1, \ldots, n$

$$w_1 x_1^{(j)} + \cdots + w_{n+1} x_{n+1}^{(j)} < a_{n+1},$$

which implies $f(x^{(j)}) = 0$. Thus f depends on inputs $x_1, \ldots, x_n$. Further, consider the element $y \in \{0,1\}^{n+1}$ where $y_i = 1$ iff $n \leq i \leq n+1$ and let z denote the element that differs from y exactly in component $n+1$. We obtain $f(y) = 1$ and $f(z) = 0$ in a similar way. Thus f also depends on x_{n+1} and is therefore non-degenerate.

Assume that a Perceptron-like learning rule sign identifies f in at most $q(n+1)$ correction steps where q is a polynomial. Starting from the zero vector we give $\langle (1, 0, \ldots, 0), 0 \rangle$ as first example for f. This leads to a correction step resulting in the weight vector $(-c, 0, \ldots, 0, -c)$ where c is the correction increment. (We recall that we have constant input 1 for component $n+2$ representing the negative of the threshold.) Then we take the set A from the proof of Theorem 1 and add to each element 0 as first component. Because f has weight complexity at least $\Omega(\phi^n)$, we can choose for each sufficiently large n a sequence of training examples from A such that the rule continuing with $(-c, 0, \ldots, 0, -c)$ makes $q(n+1)$ corrections. The resulting weight vector $(u_1, \ldots, u_{n+2})$ obviously satisfies $u_1 = -c$ because the first weight does not change when the extended examples from A are used. Thus we have $u_1 \cdot w_1 = -c < 0$ contradicting (3). □

Although we have restricted sign identification to non-degenerated functions, the learning process was started with the zero vector which represents the function that is 1 everywhere. The proof can easily be adapted to the case where an initial vector is chosen that represents a non-degenerate function.

6 Identification with Littlestone's Rules

Littlestone (1988) has proposed two learning rules that are quite similar to Perceptron-like rules. Instead of changing the weights additively they use a multiplicative update. These so-called Winnow algorithms share with Perceptron-like rules the property that the weights are updated only if an error has occured and if so, that

only those weights are changed that received a non-zero input. If the output of the current hypothesis is supposed to be 1 then w_i is changed to αw_i for some fixed constant $\alpha > 1$. If the output is supposed to be 0 then the two Winnow rules work differently. Winnow1 changes w_i to 0, whereas Winnow2 changes it to w_i/α. Thus, the Winnow algorithms are local rules and fit into the model of on-line learning in the same way as Perceptron-like rules.

The complexity of learning with Winnow rules has been investigated by Maass and Turán (1994). For the more general class of so-called k-bounded learning algorithms, which comprises the Winnow rules for a sufficiently large but fixed k, they have established a lower bound of $2^{\Omega(n/k)}$ for exact identification. For the Winnow rules in particular, this implies that there are target functions and sequences of training examples where they make $2^{\Omega(n)}$ corrections. As pointed out in the previous paragraph, Winnow rules leave weights with input 0 unchanged. We can now make use of this fact as we did in the proof of the lower bound for order identification with Perceptron-like learning rules. We extend the training examples by two components containing 0. We also add two inputs to the initial hypothesis and the target function such that their variable order with respect to these inputs is different. Reasoning in the same way as we did in the proof of Theorem 2 we obtain a lower bound for order identification with Littlestone's rules.

Theorem 4 *Winnow rules cannot order identify the class of Boolean threshold functions with non-negative weights in polynomial time.*

Winnow rules do not change the signs of the weights. Therefore, sign identification is out of question for these rules. However, Littlestone (1988) has presented a transformation that allows to learn Boolean threshold functions with varying signs. This transformation maps a threshold function with n inputs to a threshold function with $2n$ inputs and only non-negative weights in such a way that each weight is replaced by two weights. One of them is used in case of a positive, the other in case of a negative sign. Learning a function with negative weights is then reduced to learning this extended function.

One can then define the criterion of sign identification also for Winnow rules. The learning task would be to identify the sign of a weight by making the corresponding weight larger than its counterpart. Thus, weights of constant size are sufficient. Using the result of Maass and Turán (1994) mentioned above and the constructions presented here one obtains an exponential lower bound also for this kind of sign identification.

7 Concluding Remarks

We have taken a closer look at the capabilities of Perceptron-like learning rules. We have lowered the demand of exact identification introducing three weaker criteria: PAC identification is a probabilistic criterion that requires to find with high probability hypotheses with small error. Order and sign identification are criteria related to partial knowledge about the representation of the target function. We have shown

how these criteria can be used in an on-line learning task and investigated how many correction steps are required. It has turned out that none of the criteria can be met by Perceptron-like learning rules in polynomial time. This raises the question what Perceptron-like rules can accomplish in polynomial time at all. We mention some results.

Maass and Turán (1994) have considered subclasses of Boolean threshold functions which have weights bounded by a polynomial $p(n)$. They have shown that Perceptron-like rules can identify such a class in at most $O(n^2p^2(n))$ correction steps. Furthermore, they concluded that this bound is optimal for all k-bounded rules, i.e. rules where each weight can be modified using one of k possible instructions and where the weight change operations are commutative in each weight. However, as we have already mentioned in Section 2, the average weight complexity of a Boolean threshold function is at least 1.4^n (Hampson and Volper 1986). Thus, the polynomial bound on the weights involves a considerable loss of the computational power of single neurons.

The lower bound for PAC identification is based on the distribution independence assumption. Bartlett and Williamson (1991) have shown that for the subclass of so-called λ-reasonable distributions halfspaces on the surface of the unit hypersphere can be PAC learned in time polynomial in n, $1/\varepsilon$, $1/\delta$, and $1/\lambda$. It is not known if this result also holds for Boolean threshold functions. Also not known is whether Littlestone's rules can PAC identify in polynomial time.

Kivinen and Warmuth (1995) have compared Perceptron and Winnow rules when learning k-literal disjunctions. They have constructed training sequences where Perceptron rules make $(n-k+1)/2$ corrections whereas Winnow2 manages with $O(k \log n)$ corrections. It would be interesting to know if there is also a class of threshold functions where Perceptron rules perform considerably better than Winnow rules.

A further question concerns the separability of the identification criteria. All criteria considered here are met by polynomial-time algorithms for linear programming, some of which can even be implemented as on-line learning rules (Maass and Turán 1994). Of course, these algorithms achieve exact identification which is the strongest criterion one can imagine. May-be there are learning rules somewhat more complicated than Perceptron or Winnow rules but simpler than algorithms for linear programming which may not be able to exactly identify threshold functions but meet one of the weaker criteria.

The latter issue seems to be closely related to the question of locality or distributedness of learning rules. Although there is no clear definition of local[1] there seems to be a wide agreement that learning rules can be lined up according to their degree of locality. As Maass and Turán (1994) have argued, Perceptron and Winnow rules belong more to the local ones whereas rules based on linear programming are considered more global. It appears from the results of this paper that there is a trade-off between locality and efficiency of learning rules. Giving up locality

[1] Palm (1991) further distinguishes local two-term rules, like the Hebb rule, from local three-term rules, like the Perceptron rule.

may increase efficiency, on the other hand, all known rules that learn fast lack the simplicity and lightness of local learning rules.

References

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

Anthony, M., Brightwell, G., and Shawe-Taylor, J. 1995. On specifying Boolean functions by labelled examples. *Discrete Applied Mathematics* **61**, 1–25.

Anthony, M., and Shawe-Taylor, J. 1993. Using the Perceptron algorithm to find consistent hypotheses. *Combinatorics, Probability and Computing* **2**, 385–387.

Bartlett, P. L., and Williamson, R. C. 1991. Investigating the distribution assumptions in the pac learning model. In Valiant, L. G., and Warmuth, M. K., editors, *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pp. 24–32. Morgan Kaufmann, San Mateo, CA.

Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K. 1989. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association for Computing Machinery* **36**, 929–965.

Hampson, S.E., and Volper, D. J. 1986. Linear function neurons: Structure and training. *Biological Cybernetics* **53**, 203–217.

Håstad, J. 1994. On the size of weights for threshold gates. *SIAM Journal on Discrete Mathematics* **7**, 484–492.

Kivinen, J., and Warmuth, M. K. 1995. The Perceptron algorithm vs. Winnow: linear vs. logarithmic mistake bounds when few input variables are relevant. In *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, pp. 289–296. ACM Press, New York.

Lewis II, P. M. 1966. A lower bound on the number of corrections required for convergence of the single threshold gate adaptive procedure. *IEEE Transactions on Electronic Computers* **15**, 933–935.

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

Maass, W. 1994. Perspectives of current research about the complexity of learning on neural nets. In Roychowdhury, V., Siu, K.-Y., and Orlitsky, A., editors, *Theoretical Advances in Neural Computation and Learning*, pp. 295–336. Kluwer Academic Publishers, Boston, MA.

Maass, W., and Turán, G. 1994. How fast can a threshold gate learn? In Hanson, S. J., Drastal, G., and Rivest, R., editors, *Computational Learning Theory and Natural Learning Systems: Constraints and Prospects*, pp. 381–414. MIT Press, Cambridge, MA.

McCulloch, W. S., and Pitts, W. 1943. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* **5**, 115–133.

McNaughton, R. 1961. Unate truth functions. *IRE Transactions on Electronic Computers* **10**, 1–6.

Minsky, M. L., and Papert, S. A. 1988. *Perceptrons: An Introduction to Computational Geometry.* MIT Press, Cambridge, MA, expanded edition.

Muroga, S. 1971. *Threshold Logic and Its Applications.* John Wiley & Sons, New York.

Nagy, G. 1991. Neural networks—then and now. *IEEE Transactions on Neural Networks* **2**, 316–318.

Novikoff, A. 1962. On convergence proofs for Perceptrons. In *Symposium on Mathematical Theory of Automata*, pp. 615–622. Polytechnic Institute of Brooklyn.

Palm, G. 1991. Memory capacities of local rules for synaptic modification: A comparative review. *Concepts in Neuroscience* **2**, 97–128.

Parberry, I. 1994. *Circuit Complexity and Neural Networks.* Foundations of Computing Series, MIT Press, Cambridge, MA.

Paull, M. C., and McCluskey, Jr., E. J. 1960. Boolean functions realizable with single threshold devices. *Proceedings of the IRE* **48**, 1335–1337.

Pitt, L., and Valiant, L. G. 1988. Computational limitations on learning from examples. *Journal of the Association for Computing Machinery* **35**, 965–984.

Rosenblatt, F. 1958. The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* **65**, 386–408.

Rosenblatt, F. 1962. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.* Spartan Books, Washington, DC.

Schmitt, M., 1994. On the size of weights for McCulloch-Pitts neurons. In Caianiello, E. R., editor, *Proceedings of the Sixth Italian Workshop on Neural Nets WIRN VIETRI-93*, pp. 241–246. World Scientific, Singapore.

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