In: R. Trappl (ed.), *Proceedings of the European Meeting on Cybernetics and Systems Research EMCSR'96*, volume 2, pages 1049–1054, Vienna, 1996.

Lower Bounds on Identification Criteria for Perceptron-like Learning Rules

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

Abstract

We investigate the computational complexity of identifying neural weights using Perceptron-like learning rules. These are understood as instructions to change weights by a fixed amount after occurrence of an error. We are considering worst-case bounds on the number of correction steps when the training examples are taken from Boolean functions computable by McCulloch-Pitts neurons. In order to avoid known exponential lower bounds for exact identification we define and analyze three criteria that do not require that the learning process generates weights that precisely represent the target function: PAC identification, order identification, and sign identification. Our results show that Perceptron-like learning rules cannot satisfy any of these criteria when the number of correction steps is to be bounded by a polynomial. This indicates that even by considerably lowering one's demands one cannot prevent Perceptron-like rules from being computationally infeasible.

1 Introduction

Although the Perceptron learning rule is now almost 40 years old it does not seem to have lost much of its attraction. On the contrary, there are several appealing features that make it appear advantageous—at least when training single neuron architectures: The Perceptron rule is easy to implement, because the hypothesis that has to be maintained during the learning process is a single weight vector, and the corrections are simple additions or subtractions. It can even be implemented easily in parallel due to the fact that the only information required for the correction is the sign of the error and communication between weights is therefore quite low. The Perceptron rule is a classic example for on-line learning: the updates are based on single, misclassified examples and learning may start from any initial weight vector. The latter fact makes the rule even tolerate temporary classification errors by the teacher or allows it to 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.

Complexity considerations, however, make the Perceptron rule appear in a less favorable light. Boolean functions are known that make the number of corrections grow exponentially in the number of weights—even independently of the order of the training examples. One way out of this fairly unacceptable situation is to abandon Perceptron rules and turn towards more advanced training procedures based on polynomial-time algorithms for linear programming. On the other hand, there seems to be a chance to stick to simpler learning procedures if one weakens the demands on the learning process and is content with a result that does not completely agree with the learning target but is somehow close to it.

In this paper we explore the latter road by defining three criteria for neural weight vectors to be close to each other. A learning process which is based on one of these criteria does not have to meet the requirement of exact identification.

The first one, which is called PAC identification, is derived from Valiant's PAC learning paradigm. We assume 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. We require that with high probability (with respect to the training examples) the probability of an error (with respect to the test example) is low. The definition is motivated by the fact that learning in the PAC paradigm is known to be easier than exact identification.

The second criterion, order identification, is based on the relationship between the magnitudes of the weights. For each Boolean threshold function we define an ordering of the weights that corresponds to their order of magnitude and require that the learning process yields a function with the same ordering. Thus, learning is not forced to produce large weights because any ordering can be represented by weights with an absolute value that is not larger than the number of weights.

Finally, sign identification is based on a monotonicity property of Boolean threshold functions and states that the learning process is successful if it produces a weight vector where the corresponding weights in the target function have the same sign. For this criterion to make sense, however, we have to restrict the possible target functions to 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 two different weights, say $\{-1, 1\}$, whereas order identification requires in the worst case a linear number of different values.

In the following we ask if Perceptron-like learning rules can meet these criteria in a polynomial number of correction steps. The answer will in all three cases be to the negative. In Section 2 we introduce the basic concepts and review some properties of McCulloch-Pitts neurons and Perceptron rules. This section also contains references to related work. Each of the following sections, then, addresses the definition and investigation of one specific identification criterion. In the final section we conclude with some remarks and open questions.

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 by

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

It is well known that integer weights are sufficient to represent any Boolean threshold function, see e.g. [Minsky and Papert, 1988; Muroga, 1971; Parberry, 1994].

As a useful tool for the investigation of the learning complexity of Perceptron-like rules we introduce the *weight complexity* of a threshold function. It is defined as the smallest natural number such that the function can be represented by a weight vector where the weights and threshold are integers and their absolute value is not larger than this number. The weight complexity of Boolean threshold functions 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 $2^{-n}(n+1)^{(n+1)/2}$ is known (see also [Håstad, 1994; Parberry, 1994]). By different methods several functions of weight complexity at least $2^{\Omega(n)}$ have been constructed [Minsky and Papert, 1988; Muroga, 1971; Parberry, 1994]. By a more involved construction [Håstad, 1994] has recently defined a function that requires weights at least as large as $2^{\Omega(n \log n)}$ (see also [Parberry, 1994]). Using a counting argument [Hampson and Volper, 1986, Section 3.3.3] proved that the average weight complexity of a Boolean threshold function is at least 1.4^n.

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 adding a weight w_{n+1} with value $-t$ and constant input 1.

A *Perceptron-like learning rule* prescribes 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

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

Perceptrons trained by this method were originally known as α-Perceptrons [Rosenblatt, 1958] (see also [Nagy, 1991] for a historical perspective). The value c is an arbitrary scalar constant 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. An endless number of proofs with subtle differences has been published. [Lewis II, 1966; Minsky and Papert, 1988; Novikoff, 1962; Parberry, 1994; Rosenblatt, 1962] are just to mention a few. Using the weight complexity of Boolean threshold functions, [Maass, 1994, Theorem 8.2] 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 can occur.

Concerning lower bounds one can easily conclude 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 $\widetilde{w}$, cannot be smaller than the weight complexity of the function. Therefore, we obtain the following slightly improved lower bound.

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 time bounds hold strictly speaking only if one chooses the zero vector as initialization. Because we are here interested in the coarse distinction between polynomial and exponential behavior we shall make this simplifying assumption throughout the paper. The starting from any other weight vector cannot improve the worst-case behavior of the learning process.

3 PAC identification

It is obvious from the lower bounds mentioned in the previous section that for Perceptron-like learning rules exact identification is too strong a requirement to be achieved in polynomial time. In the following, 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 by the same distribution.

A class of target functions is said to be PAC learnable in polynomial time if the number of examples necessary 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 many correction steps). PAC learning is known to be easier than exact identification. [Angluin, 1988, Section 2.4] shows that any method for exact identification can be turned into a method for PAC identification under preservation of polynomial time. She also gives an example of a target class that can be PAC identified in polynomial time but needs exponentially many examples in the worst case for exact identification. This and the fact that Boolean threshold functions are polynomially PAC learnable by methods of linear programming [Blumer *et al.*, 1989, Example 3.1.2] motivates us to ask if Perceptron-like rules can achieve PAC identification in polynomial time.

Definition 1 *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 $P[\{x : f(x) \neq g(x)\}] \leq \varepsilon$.*

The definition of PAC identification uses polynomial p to bound the number of examples and polynomial q to bound the number of computation steps. It is not a restriction if one expresses both bounds by a single polynomial. For Boolean threshold functions it is known that a polynomial number of examples is sufficient for PAC learning [Blumer *et al.*, 1989, Example 2.4]. The crucial step in the following proof of the lower bound will be to show that compared to exact identification the computation time for Perceptron rules cannot decrease substantially.

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 weight vector $(a_1, \ldots, a_n, -a_{n+1})$ where n is odd and a_i denotes the i-th Fibonacci number $a_i = a_{i-1} + a_{i-2}, a_1 = a_2 = 1$. (Recall that a_{n+1} represents the threshold.) [Hampson and Volper, 1986, Section 3.3.1] remark that this function has weight complexity $\Omega(1.6^n)$ (more precisely $\Omega(\phi^n)$ where ϕ is the golden ratio $\phi = (\sqrt{5}+1)/2$, see also [Parberry, 1994, Theorem 4.2.6]) and that it can be written as $(\cdots(((x_1 \wedge x_2) \vee x_3) \wedge x_4)\cdots) \vee x_n$. Therefore it belongs to the class of nested functions defined by [Anthony *et al.*, 1992, Section 2.3]. The latter have shown that any nested Boolean function f on n inputs has specification number $n+1$ [Anthony *et al.*, 1992, Theorem 2.9] 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. Then there exist polynomials p and q according to Definition 1. 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 $\delta = 1/2$. 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 must agree with f on A with probability at least 1/2. Thus, because f is nested, it 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 $\Omega(\phi^n)$. $\square$

In the proof we have chosen $\delta = 1/2$ but any fixed value $0 < \delta < 1$ is sufficient. The method of concentrating a distribution on a set of examples that is hard to be learned goes back to [Pitt and Valiant, 1988].

4 Order identification

From the result in the previous section we see that the requirement of producing large weights constitutes a serious obstacle for Perceptron-like learning rules not only when exactly but also when PAC identifying Boolean threshold functions. In this section we try to circumvent this obstacle by defining an equivalence relation that allows to identify threshold functions using only weights of size at most n. To this end we introduce an ordering relation on the input variables of a threshold function.

Definition 2 *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. The* restriction $f_\beta : \{0,1\}^{n-2} \to \{0,1\}$ *is defined by*

$$f_\beta(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n) = f(x_1, \ldots, x_{i-1}, \beta(x_i), x_{i+1}, \ldots \ldots, x_{j-1}, \beta(x_j), x_{j+1}, \ldots, x_n).$$

The complementary assignment $\overline{\beta} : \{x_i, x_j\} \to \{0,1\}$ is defined by $\overline{\beta}(x) = 1 - \beta(x)$. Now let $\beta(x_i) = 1$ and $\beta(x_j) = 0$. We say that

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

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

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

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

It is not hard to see that for any threshold function the above defined relation provides a total ordering of the variables. The following lemma by Muroga also establishes a relationship between the ordering of variables and the ordering of weights. We state it without proof.

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

(i) The input 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 $x_i \overset{f}{\approx} x_j$.

Definition 3 *Let $f, g : \{0,1\}^n \to \{0,1\}$ be threshold functions. They are said to have the same variable order if for all $1 \leq i, j \leq n$*

$$x_i \overset{f}{\succsim} x_j \quad \text{iff} \quad x_i \overset{g}{\succsim} x_j \qquad \text{and}$$

$$x_i \overset{f}{\approx} x_j \quad \text{iff} \quad x_i \overset{g}{\approx} x_j.$$

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

We now use the variable ordering to define the criterion of order identification considering functions that have the same order as indistinguishable.

Definition 4 *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.*

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 represented by $(1, 2, a_1, a_2, \ldots, a_n, -a_{n+1})$ where n is odd and a_i denotes the i-th Fibonacci number. Thus f is the function from the proof of Theorem 1 enlarged by two additional inputs. First we show that

$$x_1 \overset{f}{\prec} x_2 \tag{1}$$

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

$$\begin{aligned} & w_1x_1 + \cdots + w_nx_n = \overline{w}_1x_1 + \cdots + \overline{w}_nx_n \\ &= \textstyle\sum_{i=2}^{n-1} a_i \;=\; (\sum_{i=1}^{n-1} a_i) - 1 \;=\; a_{n+1} - 2. \end{aligned}$$

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

Now assume that a Perceptron-like learning rule order identifies f in at most $q(n+2)$ correction steps. We take the set A from the proof of Theorem 1 and add to each element a first and a second component, both containing 0. 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 makes $q(n+2)$ corrections. 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 induces $x_1 \overset{g}{\approx} x_2$. Together with (1) this implies that f and g have different variable order. □

5 Sign identification

As a means to investigate the power of Perceptron-like learning rules we finally introduce the sign identification criterion. It is weaker than order identification in the sense that it requires that the learning process produces a weight vector that agrees with the target function only in 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. (The threshold is excluded from this restriction.) Each function in this class depends on all its inputs and we call such a function, following [Parberry, 1994, p. 53], *non-degenerate.*

Definition 5 *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.*

A proof that every threshold function is unate has been given by [Paull and McCluskey, Jr., 1960]. 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, Theorem 3.2], 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 represented by $(w_1, \dots, w_{n+1})$. Then f is positive (negative) in x_i iff $w_i > 0$ ($w_i < 0$).*

Unateness implies that each non-degenerate threshold function has a unique sign vector. Based on this fact is the definition of sign identification.

Definition 6 *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 $(u_1, \dots, u_{n+1})$ of a function g such that there exists a weight vector $(w_1, \dots, w_{n+1})$ for f that satisfies*

$$u_i \cdot w_i > 0 \quad \text{for } i = 1, \dots, n. \tag{2}$$

We remark that if property (2) holds for a weight vector $(w_1, \dots, w_{n+1})$ then it must hold 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 letting certain weights untouched as long as possible. Let $f : \{0,1\}^{n+1} \to \{0,1\}$ be the threshold function represented by the weight vector $w = (1, a_1, a_2, \dots, a_n, -a_{n+1})$ where n is odd and a_i denotes the i-th Fibonacci number. Thus f is the function from the proof of Theorem 1 with one additional input as first component. 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 $j = 1, \dots, n$ let $x^{(j)}$ denote the element that differs from x exactly in the j-th component. Then

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

Therefore $f(x) = 1$. On the other hand

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

implies $f(x^{(j)}) = 0$. Thus f depends on inputs $x_1, \dots, x_n$. 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. We give $\langle (1, 0, \dots, 0), 0 \rangle$ as first example for f. Because we initialize with the zero vector it leads to a correction step resulting in the weight vector $(-c, 0, \dots, 0, -c)$ where c is the correction increment. (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 $(-c, 0, \dots, 0, -c)$ makes $q(n+1)$ corrections. The resulting weight vector $(u_1, \dots, u_{n+2})$ obviously satisfies $u_1 = -c$ because the first weight cannot be modified when using examples from A. Thus we have $u_1 \cdot w_1 = -c < 0$ contradicting (2). □

6 Conclusions and open questions

We have taken a closer look at the learning capabilities of Perceptron-like learning rules. Starting from the fact that exact identification requires exponential time we have lowered the demands introducing three weaker identification criteria: PAC identification is a probabilistic criterion that requires to produce with high probability hypotheses with small errors; order and sign identification are structural criteria containing some partial information about the representation of the target function. We have shown that the question whether any of these criteria can be met by Perceptron-like learning rules in polynomial time leads in all three cases to a negative answer.

[Maass and Turán, 1994, Proposition 6.3] have considered subclasses of Boolean threshold functions that have weights bounded by a polynomial $p(n)$. They have shown that Perceptron-like rules can identify such a class in at most $O(n^2 p^2(n))$ correction steps. However, due to the already mentioned result of [Hampson and Volper, 1986, Section 3.3.3], bounding the weights by a polynomial one loses much of the computational power of single neurons because the average weight complexity is at least 1.4^n.

The negative result for PAC identification is based on the fact that the distribution may be chosen arbitrarily. [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 can be extended to Boolean threshold functions.

[Maass and Turán, 1994] have shown that more complicated algorithms for linear programming can be implemented as learning rules for exact identification in polynomial time. Thus one can also PAC, order, and sign identify in polynomial time using these rules. An interesting question is therefore if there are learning rules that separate these identification criteria, i.e. they run in polynomial time for one or two criteria and need exponential time for the other. This would classify our identification criteria more distinctively according to their degree of difficulty.

References

[Angluin, 1988] Dana Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

[Anthony *et al.*, 1992] Martin Anthony, Graham Brightwell, Dave Cohen, and John Shawe-Taylor. On exact specification by examples. In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 311–318. ACM Press, New York, 1992.

[Bartlett and Williamson, 1991] Peter L. Bartlett and Robert C. Williamson. Investigating the distribution assumptions in the pac learning model. In Leslie G. Valiant and Manfred K. Warmuth, editors, *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 24–32. Morgan Kaufmann, San Mateo, CA, 1991.

[Blumer *et al.*, 1989] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association for Computing Machinery*, 36:929–965, 1989.

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

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

[Lewis II, 1966] P. M. Lewis II. 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, 1966.

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

[Maass, 1994] Wolfgang Maass. Perspectives of current research about the complexity of learning on neural nets. In Vwani Roychowdhury, Kai-Yeung Siu, and Alon Orlitsky, editors, *Theoretical Advances in Neural Computation and Learning*, pages 295–336. Kluwer Academic Publishers, Boston, MA, 1994.

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

[McNaughton, 1961] Robert McNaughton. Unate truth functions. *IRE Transactions on Electronic Computers*, 10:1–6, 1961.

[Minsky and Papert, 1988] Marvin L. Minsky and Seymour A. Papert. *Perceptrons: An Introduction to Computational Geometry.* The MIT Press, Cambridge, MA, expanded edition, 1988.

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

[Nagy, 1991] George Nagy. Neural networks—then and now. *IEEE Transactions on Neural Networks*, 2:316–318, 1991.

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

[Parberry, 1994] Ian Parberry. *Circuit Complexity and Neural Networks.* Foundations of Computing Series. The MIT Press, Cambridge, MA, 1994.

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

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

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

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

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

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