Learning Multivalued Multithreshold Functions

Martin Anthony
Department of Mathematics
and Centre for Discrete and Applicable Mathematics
London School of Economics, London WC2A 2AE, UK
m.anthony@lse.ac.uk

CDAM Research Report LSE-CDAM-2003-03, January 2003

Abstract

This paper concerns *multivalued multithreshold* functions, $\{0, 1, \ldots, k\}$-valued functions on $\mathbb{R}^n$ that may be considered as generalizations of (linear) threshold functions, or as discretized versions of artificial neurons. Such functions have arisen in the context of multiple-valued logic and artificial neural networks. For any fixed k, we present two procedures which, given a set of points labelled with the values of some (unknown) multivalued multithreshold function, will produce such a function that achieves the same classifications on the points. (That is, we present 'consistent hypothesis finders'.) One of these is based on linear programming, and the other on an 'incremental' procedure suggested by Obradović and Parberry. In standard probabilistic models of learning, it is natural to ask for some information about how many labelled data points should be used as the basis for valid inference about the function that is labelling the data. We investigate this question for the class of multivalued multithreshold functions. Finally, we examine *multithreshold functions*, a class of $\{0, 1\}$-valued functions related to the multivalued multithreshold functions. We give a simple description of an algorithm based on a procedure suggested by Takiyama, and we raise some open questions on the effectiveness of this algorithm, and, generally, on the complexity of finding consistent hypotheses for samples of multithreshold functions.

1 Introduction

This paper concerns *multivalued multithreshold* functions, $\{0, 1, \ldots, k\}$-valued functions on $\mathbb{R}^n$ that may be considered as generalizations of the (linear) threshold functions, or as discretized versions of artificial neurons. (Such functions have arisen in the context of multiple-valued logic and artificial neural networks; see [22, 17], for example.)

A *consistent hypothesis finder* for a set of functions is an algorithm that, on being presented with a set of points, each labelled with the value of some (unknown) function from the class, will produce (a representation of) some function in the class that achieves the same classifications on the points. For the class of multivalued multithreshold functions (for any fixed k), we present two consistent hypothesis finders: one is based on linear programming, and the other is an ‘incremental’ method arising from a procedure suggested by Obradović and Parberry [22].

In standard probabilistic models of learning, it is useful to have bounds on the *sample complexity* of learning a set H of functions: that is, to know something about how many labelled data points should be used as the basis for valid inference about an unknown function $t \in H$ that is labelling the data. We obtain an upper bound on the sample complexity of learning multivalued multithreshold functions.

Finally, we examine *multithreshold functions*, a class of $\{0, 1\}$-valued functions related to the multivalued multithreshold functions. We give a simple description of an algorithm based on a procedure suggested by Takiyama [26], and we raise some open questions on the effectiveness of this algorithm, and, generally, on the complexity of finding a multithreshold function consistent with a sample of such a function.

2 Multivalued multithreshold functions

For $k \in \mathbb{N}$, let $[k]$ denote the set $\{0, 1, \ldots, k\}$. A $[k]$-valued multithreshold function (or, briefly, k-MTF) is a function $f : \mathbb{R}^n \to [k]$ defined as follows: there are *weights* $w_1, w_2, \ldots, w_n$ and *thresholds* $\theta_1 \leq \theta_2 \leq \ldots \leq \theta_k$ such that, for $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, if we define $S = \{0\} \cup \{r : \sum_{i=1}^{n} w_i x_i \geq \theta_r\}$, then $f(x) = \min S$, the least element of S.

It is useful to think of these functions geometrically. Any such function corresponds to k parallel hyperplanes dividing $\mathbb{R}^n$ into $k+1$ regions. If we traverse the regions in one of the two directions normal to the hyperplanes, the classifications given to points in the regions between the hyperplanes increase successively from 0 to k (with classification 0 to those below the first hyperplane, and classification k to those above the kth hyperplane, and with points lying on the hyperplanes being given the classifications of the region just being entered).

One reason for being interested in these functions is that they can be regarded as discretized (or finite-precision) versions of monotonic neural network activation functions. They have also proven to be of interest in multiple-valued logic. Obradović and Parberry [22], and Ngom *et al.* [17, 18, 19] examined special types of multivalued multithreshold functions, where the domain of the functions was taken to be $[k]^n$, rather than $\mathbb{R}^n$ as here. That is, they considered the $(k+1)$-ary functions [22] or $(k+1)$-*valued logic functions* [17] corresponding to k-MTFs.

3 Finding a consistent multivalued multithreshold function

Suppose we are given a sequence $(x_1, x_2, \ldots, x_m)$ of m points of $\mathbb{R}^n$, each of which has been labelled with the corresponding values $t(x_i)$ of some $[k]$-valued multithreshold function t (giving us a *sample* $((x_1, t(x_1)), \ldots, (x_m, t(x_m))$ of t). Without knowing precisely the *target function* t, we might want to construct a k-MTF *consistent* with t on the sample; that is, to produce a k-MTF h such that $h(x_i) = t(x_i)$ for $i = 1, 2, \ldots, m$. A procedure achieving this will be referred to as a *consistent hypothesis finder*. This is a natural and central problem in machine learning and data-mining, where one seeks a *hypothesis* that is an *explanation* of the classified data set, and potentially a good predictor of the classifications of other, as yet unseen, points from the same corpus of data.

3.1 Using linear programming

One approach to finding a k-MTF consistent with a sample $\mathbf{s}$ of points labelled by a k-MTF is to use linear programming. Suppose that, in the sample, m_i points have classification i (for $0 \leq i \leq k$) and denote these points by $x_1^{(i)}, x_2^{(i)}, \ldots, x_{m_i}^{(i)}$. Consider the following linear program, in which there are $n+k+1$ real variables: w_i for $1 \leq i \leq n$, θ_j for $1 \leq j \leq k$, and y. Here,

$w = (w_1, w_2, \ldots, w_n)$ and, for $a, b \in \mathbb{R}^n$, $\langle a, b\rangle$ denotes the inner product $a^T b = \sum_{i=1}^n a_i b_i$.

$$
\begin{array}{rcl}
\multicolumn{3}{c}{\text{Maximize } y \text{ subject to the constraints}} \\
\langle w, x_j^{(i)}\rangle - \theta_i - y & \geq & 0 \\
\theta_{i+1} - \langle w, x_j^{(i)}\rangle - y & \geq & 0 \\
\theta_1 - \langle w, x_j^{(0)}\rangle - y & \geq & 0 \\
\langle w, x_j^{(k)}\rangle - y & \geq & 0 \\
w_l & \geq & -1 \\
-w_l & \geq & -1 \\
\theta_l & \geq & -1 \\
-\theta_l & \geq & -1 \\
y & \geq & 0,
\end{array}
$$

where l ranges from 1 to k, i ranges from 1 to $k-1$ and, for each fixed i (for the inequalities involving $x_j^{(i)}$), j ranges from 1 to m_i. There are $m + 2n + 2k$ constraints in total. Given that the sample is labelled according to the values of some k-MTF t, the program is feasible and has a positive solution. Note that the first four sets of inequalities require a weight vector w and threshold vector θ such that the resulting k-MTF correctly classifies the x_i and such that, additionally, the inner products 'clear' the required threshold by at least the amount y. Now, all of this is possible for some positive y, given the existence of t and the finiteness of s. Furthermore, the next four inequalities make the feasible region bounded, and, by scaling weight and threshold vectors, if necessary, it can be seen that t has a realizable weight vector and threshold vector satisfying these bounds. By solving this linear program, a consistent k-MTF can therefore be obtained, and so we have a consistent hypothesis finder. This method can be made to run in polynomial time in the logarithmic cost model by using, for instance, Karmarkar's algorithm [11]. In particular, if the sample points x_i are restricted to domain $[k]^n$, then the running time of the algorithm is bounded by a polynomial in $m(n+1)$, the size of the sample.

3.2 An incremental procedure

Obradović and Parberry [22] proposed an incremental algorithm for 'learning' k-MTFs on the basis of a given sample of such a function. A slightly modified version of this algorithm was presented in [17]. These algorithms are generalizations of the well-known and well-studied perceptron learning algorithm (details of which may be found in [25, 2]), which corresponds to the special case in which $k = 1$.

The algorithm in [22] maintains a *current weight vector* $w = (w_1, w_2, \ldots, w_n)$ and *threshold vector* $\theta = (\theta_1, \theta_2, \ldots, \theta_k)$, where $\theta_1 \leq \theta_2 \leq \cdots \leq \theta_k$. Together, these represent a *current hypothesis* MTF h . On presentation of an example $x \in [k]^n$ together with its classification $t(x)$, if $h(x) = t(x)$ the algorithm does nothing, while if $h(x) \neq t(x)$ then the algorithm slighly alters w and one of the θ_i. The algorithm is, in this sense, *incremental*. Obradović and Parberry established a result along the lines of the classical 'perceptron convergence theorem', by proving that on any (possibly infinite) sequence of examples from $[k]^n$, each classified by some k-MTF, t, there is an absolute bound on the number of mistakes (and hence updates) the algorithm can make (this bound depending on t). To prove this, they invoked the classical result for the perceptron.

As a consequence of the finiteness result of Obradović and Parberry, the incremental procedure can be used to construct a consistent hypothesis finder in the case where all the examples belong to $[k]^n$. For, given a finite sample $\mathrm{s} = ((x_1, t(x_1), \ldots, (x_m, t(x_m)))$, we can cycle through these labelled examples repeatedly until no further updates will occur, at which point the current hypothesis must be consistent with the sample. We will give a direct proof that, more generally, this procedure for finding a consistent k-MTF works when the examples can be in $\mathbb{R}^n$ and are not restricted to be in $[k]^n$ (which, as already mentioned, was the focus in [22, 17]). First, we describe the consistent hypothesis finder in pseudo-code.

Algorithm L: Incremental consistent hypothesis finder for k-MLTs.

```
Input:    Sample $\mathrm{s} = ((x_1, t(x_1), \ldots, (x_m, t(x_m)))$ of $k$-MLT $t$.
Output:   Weights $w_1, \ldots, w_n$ and thresholds $\theta_1 \leq \cdots \leq \theta_k$

for all $i$, set $w_i := 0$
for all $l$, set $\theta_l := 0$
repeat until no updates needed in a complete cycle through s
for $i := 1$ to $m$ do
    let $h$ be the current hypothesis, represented by $w$ and $\theta$
    if $v = h(x_i) \neq t(x_i)$ then
        let $\Delta = t(x_i) - h(x_i) = t(x_i) - v$
        if $\Delta < 0$ then
```

```
            update the weights and thresholds as follows
            $\theta_v \leftarrow \theta_v + 1$
            $\theta_l \leftarrow \theta_l$ for $l \neq v$ (i.e., no change)
            $w \leftarrow w - x_i$
        **if** $\Delta > 0$ then
            update the weights and thresholds as follows
            $\theta_{v+1} \leftarrow \theta_{v+1} - 1$
            $\theta_l \leftarrow \theta_l$ for $l \neq v+1$ (i.e., no change)
            $w \leftarrow w + x_i$
**return** $w_1, w_2, \ldots, w_n$ and $\theta_1, \ldots, \theta_k$.
**end**
```

The k-MTF corresponding to the weights and thresholds output by the algorithm is called the *output hypothesis* of the algorithm, and is denoted $L(\mathbf{s})$.

Ngom *et al.* [17] considered a slight variant of the procedure suggested by Obradović and Parberry. (The problem they considered was slightly more general too: they were interested in incrementally learning 'permutably homogeneous perceptrons', of which k-MTFs are a special type.) Following their variation of [22], an alternative consistent hypothesis finder can be devised that has the following update rule:

```
**if** $v = h(x_i) \neq t(x_i)$ **then**
    **let** $\Delta = t(x_i) - h(x_i) = t(x_i) - v$
    **if** $\Delta < 0$ then
        update the weights and thresholds as follows
        $\theta_v \leftarrow \theta_v + \Delta = \theta_v - |\Delta|$
        $\theta_l \leftarrow \theta_l$ for $l \neq v$ (i.e., no change)
        $w \leftarrow w - \Delta x_i = w + |\Delta| x_i$
    **if** $\Delta > 0$ then
        update the weights and thresholds as follows
        $\theta_{v+1} \leftarrow \theta_{v+1} - \Delta$
        $\theta_l \leftarrow \theta_l$ for $l \neq v+1$ (i.e., no change)
        $w \leftarrow w + \Delta x_i$
```

Thus, in this case, the extent by which the weights and thresholds are changed depends on how far $h(x_i)$ is from $t(x_i)$ and not merely on the 'sign' of the difference. A further possible modification is to have a ‘learning rate’ (possibly changing in time) multiplying the additive changes.

Obradović and Parberry [22] also investigated the performance of an alternative procedure in which weights and thresholds are updated *multiplicatively* rather than *additively*, following Littlestone’s ‘Winnow’ generalization of the standard perceptron learning algorithm [14]. (Indeed, this is the primary focus of their paper.) They show that this multiplicative algorithm is in many cases better than the additive one, in that the bound on the number of updates required can be significantly smaller. This multiplicative algorithm can also, in an analogous way, be turned into a consistent hypothesis finder.

Theorem 3.1 *Given any sample* $\mathbf{s}$ *of a* k*-MTF, the incremental consistent hypothesis finder for* k*-MLTs will terminate to produce an output hypothesis* $L(\mathbf{s})$ *consistent with* $\mathbf{s}$*. Furthermore, if the examples* x_i *in the sample satisfy* $\|x_i\| \leq R$ *and if the* k*-MTF* t *by which the sample points are labelled is represented by weight vector* W *and threshold vector* Θ *with the property that* $\|W\|^2 + \|\Theta\|^2 = 1$ *and no* x_i *lies on any of the* k *hyperplanes defined by* w *and* θ*, then the total number of updates (and hence cycles) required by* L *is at most* $(R^2+1)/\gamma^2$ *where*

$$\gamma = \min\{|\langle W, x_i\rangle - \Theta_l| : 1 \leq i \leq m,\ 1 \leq l \leq k\} > 0.$$

Proof: The proof is a variant of the proof of the perceptron convergence theorem [25, 20, 21, 2]. Clearly, since the sample is finite, there will be a weight vector W and a threshold vector Θ representing t such that no point of the sample lies on any of the k hyperplanes (because there is flexibility to perturb the thresholds). By scaling the weights and thresholds if necessary, we can further assume that $\|W\|^2 + \|\Theta\|^2 = 1$. Let W and Θ be a fixed choice of such vectors. Denote by $w(u)$ and $\theta(u)$ the weight and threshold vectors after u updates have been made, and let the components of these be $w_i(u)$ and $\theta_l(u)$. (Note that $w(0)$ and $\theta(0)$ are the all-zero vectors.) For some x in the sample, the uth update rule takes the form

$$\begin{aligned} w(u) &= w(u-1) + \delta x \\ \theta_l(u) &= \theta_l(u-1) - \delta. \end{aligned}$$

Here, δ is 1 or -1, according to whether $t(x) > h(x)$ or $t(x) < h(x)$, respectively; and l is, correspondingly, $v+1$ or v, where $v = h(x)$ and h is the current hypothesis (represented by $w(u-1)$ and $\theta(u-1)$). Let $N(u)$ be defined as

$$N(u) = \langle (W, \Theta), (w(u), \theta(u))\rangle = \langle W, w(u)\rangle + \langle \Theta, \theta(u)\rangle.$$

Then,

$$N(u) - N(u-1) = \langle W, \delta x\rangle - \Theta_l \delta = \delta(\langle W, x\rangle - \Theta_l).$$

Now, if $\delta = 1$, then $l = v + 1$ and, since $v = h(x) < t(x)$, we have $t(x) \geq v + 1$ and so $\langle W, x\rangle \geq \Theta_{v+1} + \gamma$, as a consequence of which $\delta(\langle W, x\rangle - \Theta_{v+1}) \geq \gamma$. If, however, $\delta = -1$, then $l = v$ and $t(x) < h(x) = v$, so that $\langle W, x\rangle < \Theta_v - \gamma$ and hence $\delta(\langle W, x\rangle - \Theta_{v+1})$. In both cases, therefore, $N(u) - N(u-1) \geq \gamma$. It follows that $N(u) \geq N(0) + \gamma u = \gamma u$.

Now let

$$L(u) = \|(w(u), \theta(u))\|^2 = \|w(u)\|^2 + \|\theta(u)\|^2.$$

From the fact that $N(u) = \langle w, w(u)\rangle + \langle \theta, \theta(u)\rangle \geq u\gamma$, together with the Cauchy-Schwarz inequality and the fact that $\|(W, \Theta)\| = 1$, we have

$$\begin{aligned} L(u) &= \|w(u)\|^2 + \|\theta(u)\|^2 \\ &= \|(w(u), \theta(u))\|^2 \\ &= \|(w(u), \theta(u))\|^2 \|(W, \Theta)\|^2 \\ &\geq (\langle (w(u), \theta(u)), (W, \Theta)\rangle)^2 \\ &= (N(u))^2 \\ &\geq (\gamma u)^2. \end{aligned}$$

But, if e_l denotes the vector with lth entry equal to 1 and all other entries 0, then

$$\begin{aligned} L(u) &= \|w(u)\|^2 + \|\theta(u)\|^2 \\ &= \|w(u-1) + \delta x\|^2 + \|\theta(u-1) - \delta e_l\|^2 \\ &= \|w(u-1)\|^2 + \|\theta(u-1)\|^2 + \delta^2\|x\|^2 + \delta^2 + 2\delta\langle w(u-1), x\rangle - 2\delta\langle \theta(u-1), e_l\rangle \\ &\leq L(u-1) + (R^2 + 1) + 2\delta\left(\langle w(u-1), x\rangle - \theta_l(u-1)\right). \end{aligned}$$

Because of the update rule, either $v = h(x) > t(x)$, in which case

$$\delta < 0, \;\; l = v, \;\text{ and } \langle w(u-1), x\rangle > \theta_v(u-1);$$

or $v = h(x) < t(x)$, and

$$\delta > 0, \;\; l = v + 1, \;\text{ and } \langle w(u-1), x\rangle < \theta_{v+1}(u-1).$$

So, in both cases, $\delta\left(\langle w(u-1), x\rangle - \theta_l(u-1)\right) < 0$, and hence $L(u) \leq L(u-1) + (R^2 + 1)$ and so $L(u) \leq L(0) + (R^2 + 1)u = (R^2 + 1)u$. It follows that

$$(\gamma u)^2 \leq L(u) \leq (R^2 + 1)u$$

and so $u \le (R^2 + 1)/\gamma^2$, completing the proof. $\square$

Note that the upper bound given in Theorem 3.1 on the number of updates depends both on t and on the precise points in the sample (through the dependence on γ). For the standard perceptron (the case $k = 1$), the case in which only Boolean points (that is, points of $\{0, 1\}^n$) have been considered has been of particular interest historically. A counterpart to this in the case $k > 1$ is to consider only points of $[k]^n$ (as in [22, 17]). With this restriction to a finite domain, for a given t, the parameter γ can of course be bounded below independently of the sample. Thus, one can bound the number of updates (and hence cycles) independently of the sample. It is, however, well-known (in the case $k = 1$) that this bound can be exponential in n; see [15, 5], for instance.

This consistent hypothesis finder has an appealing on-line, incremental character, but (unlike the method based on linear programming) it is not efficient. Even when the sample points are restricted to $\{0, 1\}^n$, the time taken to produce a hypothesis finder will not generally be polynomial in $m(n + 1)$, the size of the input. For, when $k = 1$, the algorithm is equivalent to the consistent hypothesis finder based on the standard perceptron learning algorithm, and this is known not to be efficient [5]. (There is a Boolean threshold function t and a set S of $n + 1$ examples with the property that the only threshold function consistent with t on S is t itself and, moreover, the ratio of the largest to the smallest weight in any weight vector representing t is exponential in n. On presentation of the sample corresponding to S and t, the algorithm will necessarily make an exponential number of updates to achieve the exponential separation between the largest and smallest weights for most choices of initial weights and thresholds.)

4 Learning multivalued multithreshold functions

We now discuss the *sample complexity* of learning k-MTFs in a $[k]$-valued version of the basic PAC model of learning [28, 7, 3, 13, 6]. (Extensions to more general models such as the $[k]$-valued versions of ‘agnostic PAC learning’ [12, 2] could also be given, but for brevity we consider only the case corresponding to the basic PAC model.)

Let H be a set of functions from $\mathbb{R}^n$ to $[k]$ and suppose that there is some probability distribution P on the domain $\mathbb{R}^n$. In the ‘PAC’ model of learning [28, 7], it is assumed that a *learning algorithm* L receives a sequence of m points of $\mathbb{R}^n$, each drawn independently according to P, and is also given the values of some *target function* $t \in H$ on these points. Thus, the input to the learning algorithm is a *sample* $\mathbf{s} = ((x_1, t(x_1)), \ldots, (x_m, t(x_m)))$, for some m; and this sample is randomly drawn, in the sense that $(x_1, \ldots, x_m)$ is distributed according to the product probability distribution P^m. On the basis of a sample, the algorithm outputs a (representation of a) *hypothesis* $h = L(\mathbf{s}) \in H$. Loosely speaking, the learning algorithm is regarded as being successful (that is, it is a *PAC learning algorithm*) if, without knowing the target function or the distribution, a guarantee can be given that, provided the sample is long enough, then, with high probability, the output hypothesis $h = L(\mathbf{s})$ closely approximates to t. Formally, the *error* of h with respect to t and P is $\mathrm{er}_P(h, t) = P(\{x : h(x) \neq t(x)\})$, and we say that L is a PAC learning algorithm if there is a function $m_0 : (0,1) \times (0,1) \to \mathbb{R}$ such that for any $t \in H$ and any probability distribution P on $\mathbb{R}^n$, if $m > m_0(\delta, \epsilon)$ then, with probability at least $1 - \delta$ (with respect to the product distribution P^m that governs $x = (x_1, x_2, \ldots, x_m)$), a sample $\mathbf{s}$ is such that $\mathrm{er}_P(L(\mathbf{s}), t) < \epsilon$. That is,

$$m > m_0(\delta, \epsilon) \Longrightarrow \mathrm{Prob}\left(\mathrm{er}_P(L(\mathbf{s}), t) \geq \epsilon\right) < \delta.$$

Note that m_0 is independent of t and P. A bound on the function m_0 is known as a *sample complexity bound*.

We now use results from computational (or statistical) learning theory. For this, we define the *growth function* of a set of functions H mapping from $\mathbb{R}^n$ to $\{0,1\}$. Let $\Pi_H : \mathbb{N} \to \mathbb{N}$ be given by

$$\Pi_H(m) = \max\{|H|_S| : S \subseteq X, |S| = m\},$$

where $H|_S$ denotes H restricted to domain S. Note that $\Pi_H(m) \leq 2^m$ for all m. Following [30, 29], Blumer *et al.* [7] proved the following bound for the case in which H maps into $\{0,1\}$ (that is, $k = 1$), and in which L is a consistent hypothesis finder:

$$\mathrm{Prob}\left(\mathrm{er}_P(L(\mathbf{s}), t) \geq \epsilon\right) < 2\,\Pi_H(2m)\,2^{-\epsilon m/2}.$$

To obtain a sample complexity bound for the case in which the functions map into $[k]$ for $k \geq 2$, we use the *graphs* of the functions [16, 7]. For $h \in H$, let $\mathcal{G}h$, the *graph of* h, be the function from $\mathbb{R}^n \times [k]$ to $\{0,1\}$ defined by $\mathcal{G}h(x,y) = 1 \Leftrightarrow h(x) = y$ and let $\mathcal{G}H = \{\mathcal{G}h : h \in H\}$, the *graph space* of H. It follows from the result of Blumer *et al.* [7] that if L is a consistent hypothesis finder for H, then

$$\mathrm{Prob}\left(\mathrm{er}_P(L(\mathbf{s}), t) \geq \epsilon\right) < 2\,\Pi_{\mathcal{G}H}(2m)\,2^{-\epsilon m/2}.$$

It is known [1] that the number of ways in which m points can be partitioned into $k+1$ sets by k parallel hyperplanes is no more than

$$\sum_{i=0}^{n+k-1} \binom{mk-1}{i},$$

from which it follows that the number of ways in which a given set of m points can be classified by the set H of k-MTFs is no more than $2\sum_{i=0}^{n+k-1} \binom{mk-1}{i}$. This quantity is therefore also an upper bound on the growth function $\Pi_{\mathcal{G}H}(m)$ of the graph class. Using these observations, we obtain the following sample complexity bound for learning k-MTFs.

Theorem 4.1 *Suppose that L is a consistent hypothesis finder for the class of $[k]$-valued multi-threshold functions. Then L is a PAC learning algorithm for the class, and its sample complexity is bounded above by*

$$m_0(\delta,\epsilon) = \frac{4}{\epsilon}\left((n+k-1)\log_2\left(\frac{12k}{\epsilon}\right) + \log_2\left(\frac{4}{\delta}\right)\right).$$

Proof: We have

$$\begin{aligned}
\text{Prob}\left(\text{er}_P(L(\mathbf{s}),t) \geq \epsilon\right) &< 2\,\Pi_{\mathcal{G}H}(2m)\,2^{-\epsilon m/2} \\
&\leq 4\sum_{i=0}^{n+k-1}\binom{2mk-1}{i}2^{-\epsilon m/2} \\
&< 4\left(\frac{2mk}{n+k-1}\right)^{n+k-1} 2^{-\epsilon m/2},
\end{aligned}$$

where (see [7]) the last inequality is valid for $m \geq n+k$. As in [4], using the fact that for all $0<\alpha<1$, $\ln x \leq \alpha x + \ln(1/\alpha) - 1$, choosing $\alpha = (\epsilon \ln 2)/(4\epsilon)$, and performing some easy manipulation, we see that the probability is therefore less than δ if

$$\frac{\epsilon m \ln 2}{4} \geq \ln\left(\frac{4}{\delta}\right) + (n+k-1)\ln\left(\frac{8k}{\epsilon\ln 2}\right),$$

from which the result follows. □

From the bound in the proof of this theorem, an equivalent statement can be made concerning the 'generalization error', for a fixed length m of sample. Namely, for any t and P, for $m \geq n+k$, and for any $\delta \in (0,1)$, the following holds: with probability at least $1-\delta$, a random sample $\mathbf{s}$ is such that

$$\mathrm{er}_P\left((L(\mathbf{s},t)\right) < \frac{2}{m}\left((n+k-1)\log_2\left(\frac{2mk}{n+k-1}\right)+\log_2\left(\frac{8}{\delta}\right)\right).$$

5 Multithreshold functions

5.1 Definitions

We now turn our attention to a class of $\{0,1\}$-valued functions defined by k parallel hyperplanes. We define a *k-threshold function f* as follows. There are k parallel hyperplanes, which divide $\mathbb{R}^n$ into $k+1$ regions. The function assigns points in the same region the same value, either 0 or 1. Without any loss, we may suppose that the classifications assigned to points in neighbouring regions are different (for, otherwise, at least one of the hyperplanes is redundant); thus, the classifications alternate as we traverse the regions in the direction of the normal vector common to the hyperplanes. Equivalently, f is a k-threshold function if the following holds: there is a weight-vector $w=(w_1,w_2,\ldots,w_n)$ and a threshold vector $\theta=(\theta_1,\theta_2,\ldots,\theta_k)$ (with $\theta_1 \leq \theta_2 \leq \cdots \leq \theta_k$) such that, if $I_0=(-\infty,\theta_1)$, $I_k=[\theta_k,\infty)$ and, for $1 \leq s \leq k-1$, $I_s=[\theta_s,\theta_{s+1})$, then *either* $f(x)=1$ if and only if $\langle w,x\rangle \in I_j$ for j even, *or* $f(x)=1$ if and only if $\langle w,x\rangle \in I_j$ for j odd.

These functions have been studied in a number of papers, such as [8, 23, 26, 27], for instance. This method of binary classification is reasonably powerful. In particular, restricting attention to the Boolean domain $\{0,1\}^n$, Bohossian and Bruck [8] observed that any Boolean function can be realized as a 2^n-threshold function. (For that reason, they paid particular attention to multithreshold functions where the number of thresholds is polynomial in n.)

5.2 Finding consistent hypotheses: some open questions

We now consider the problem of finding consistent hypotheses for the class of k-threshold functions. The fact that the classifications now alternate between 0 and 1, rather than take, successively, the values 0 to k, makes it difficult to adapt the linear programming and incremental approaches to finding consistent k-MTFs. We raise two open questions: first, is a procedure based on a technique proposed by Takiyama [26] effective; and, secondly, is there any *efficient* means of finding a consistent hypothesis, in the restricted problem where all examples are assumed to be binary vectors?

An incremental method based on a procedure of Takiyama

An incremental ‘learning’ algorithm was proposed by Takiyama [26]. Although he cites some experimental success with the method, no analysis of its effectiveness was undertaken. The presentation of the method in [26] is very complex, but the procedure can be described much more simply. We describe here the modification of Takiyama’s method that would be applicable to cycling through a finite sample (in the hope of creating a consistent hypothesis). Suppose that a sample s for some multithreshold function t is given. Then the procedure can be described as follows. The numbers $a(u)$ for $u \in \mathbb{N}$ constitute some prescribed sequence of positive ‘learning rates’, and the variable u indexes the updates made by the algorithm.

```
Input:    Sample $\mathrm{s} = ((x_1, t(x_1), \dots, (x_m, t(x_m)))$ of $k$-threshold function $t$.
Output:   Weights $w_1, \dots, w_n$ and thresholds $\theta_1 \le \theta_2 \le \cdots, \theta_k$

for all $i$, choose $w_i$ randomly
for all $l$, choose $\theta_l$ randomly
repeat until no updates needed in a complete cycle through s
for $i := 1$ to $m$ do
    let $h$ be the current hypothesis, represented by $w$ and $\theta$
    if $h(x_i) \neq t(x_i)$ then
        let $v$ be such that $\frac{\theta_{v-1}+\theta_v}{2} < \langle w, x \rangle \le \frac{\theta_v+\theta_{v+1}}{2}$
        if $\langle w, x \rangle \ge \theta_v$ then
        update the weights and thresholds as follows
            $\theta_v \leftarrow \theta_v + a(u)$
            $\theta_l \leftarrow \theta_l$ for $l \neq v$ (i.e., no change)
```

```
                $w \leftarrow w - a(u)x_i$
        **if** $\langle w, x \rangle < \theta_v$ **then**
                update the weights and thresholds as follows
                $\theta_v \leftarrow \theta_v - a(u)$
                $\theta_l \leftarrow \theta_l$ for $l \neq v$ (i.e., no change)
                $w \leftarrow w + a(u)x_i$
**return** $w_1, w_2, \dots, w_n$ and $\theta_1, \dots, \theta_k$.
**end**
```

Thus, given an example x_i misclassified by the current hypothesis (represented by w and θ), the procedure shifts the threshold θ_v nearest in value to $\langle w, x_i \rangle$ and alters the weight vector w, so as to make the quantity $\langle w, x_i \rangle - \theta_v$ decrease or increase, as appropriate.

For $k \geq 2$, the proof of Theorem 3.1 apparently cannot be adapted to show that this procedure terminates (because the fact that $h(x) \neq t(x)$ does not imply either that $\langle w, x \rangle$ is too large or that it is too small). We therefore raise the following open question:

Question: Does this procedure terminate for some choice of sequence $(a(u))_{u=1}^{\infty}$? That is, is it a consistent hypothesis finder?

Computational complexity of finding a consistent hypothesis

Let us suppose that the domain is restricted to $\{0, 1\}^n$, so that we are considering the *Boolean* k-threshold functions (simply referred to as Boolean threshold functions in the case $k = 1$). Even if the incremental method described above is indeed, in this case, a consistent hypothesis finder, it is not an *efficient* one, in the sense that the running time will not generally be polynomial in $m(n + 1)$, the size of the input. That this is the case follows from the observation that when $k = 1$, the procedure is equivalent to the consistent hypothesis finder based on the standard perceptron algorithm, and, as already noted, this is not a polynomial-time algorithm.

The question arises therefore whether there can be some other efficient consistent hypothesis finder for the class of Boolean k-threshold functions. In the case $k = 1$, there is. For, in this case the consistent hypothesis finder for 1-MTF based on linear programming is a consistent hypothesis finder for Boolean threshold functions, and is known to be efficient (by using, for example, Karmarkar's algorithm [11], as noted in [7].)

More specifically, consider the case $k = 2$ (and, again, Boolean domain $\{0,1\}^n$). Corresponding to the problem of finding a consistent hypothesis, we have the following 'consistency problem':

TWO PARALLEL PLANE SEPARABILITY
Instance: $S^+ \subseteq \{0,1\}^n$ and $S^- \subseteq \{0,1\}^n$.
Question: Are there two parallel hyperplanes such that all points of S^+ lie between the hyperplanes and all points of S^- do not?

If this problem is NP-complete then (assuming P $\neq$ NP), there can certainly be no efficient consistent hypothesis finder for 2-threshold functions. Furthermore, since a class of Boolean functions is learnable in the standard PAC model of learning if and only if the corresponding consistency problem is in RP (as shown in [24], for example), unless TWO PARALLEL PLANE SEPARABILITY is in RP, there can be no efficient PAC learning algorithm for Boolean 2-threshold functions (unless P $=$ RP). We therefore raise the following question:

Question: Is TWO PARALLEL PLANE SEPARABILITY NP-complete?

That the answer to this question could be 'yes' might be suggested by the fact that the consistency problem for the intersection of two halfspaces is NP-complete [9] (though, here, of course, the halfspaces need not necessarily be defined by parallel hyperplanes). On the other hand, for the case of single-plane separability, as already noted, the consistency problem can be solved in polynomial time.

References

[1] M. Anthony. *Partitioning points by parallel planes*. RUTCOR Research Report RRR-39-2002, Rutgers Center for Operations Research, 2002. (Also, CDAM research report LSE-CDAM-2002-10, Centre for Discrete and Applicable Mathematics, London School of Economics.)

[2] M. Anthony and P. L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

[3] M. Anthony and N. L. Biggs. *Computational Learning Theory: An Introduction*. Cambridge Tracts in Theoretical Computer Science, 30, 1992. Cambridge University Press, Cambridge, UK.

[4] M. Anthony, N. Biggs, and J. Shawe-Taylor. The learnability of formal concepts. In *Proceedings of the 3rd Annual Workshop on Computational Learning Theory*. Morgan Kaufmann, San Mateo, CA, 1990: 246–257

[5] M. Anthony and J. Shawe-Taylor. Using the perceptron algorithm to find consistent hypotheses. *Combinatorics, Probability and Computing*, 4(2), 1993: 385–387.

[6] S. Ben-David, N. Cesa-Bianchi, D. Haussler, and P. M. Long. Characterizations of learnability for classes of $\{0, \ldots, n\}$-valued functions. *Journal of Computer and System Sciences* 50(1), 1995: 74–86.

[7] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth: Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, **36**(4), 1989: 929–965.

[8] V. Bohossian and J. Bruck. Multiple threshold neural logic. In *Advances in Neural Information Processing, Volume 10: NIPS'1997*, (Michael Jordan, Michael Kearns, Sara Solla, eds), MIT Press, 1998.

[9] A. Blum and R. L. Rivest. Training a 3-node Neural Network is NP-complete. *Neural Networks*, 5(1), 1992: 117–127.

[10] J. Håstad. On the size of weights for threshold gates, *SIAM Journal on Discrete Mathematics*, 7(3), 1994: 484–492.

[11] N. Karmarkar, A new polynomial time algorithm for linear programming. *Combinatorica* 4, 1984: 373–395.

[12] M. J. Kearns, R. E. Schapire and L. M. Sellie. Toward efficient agnostic learning. *Machine Learning* 17(2/3), 1994: 115–142.

[13] M. J. Kearns and U. Vazirani. *Introduction to Computational Learning Theory*, MIT Press, Cambridge, MA, 1995.

[14] N. Littlestone, Learning quickly when irrelevant attributes abound: a new linear threshold learning algorithm. *Machine Learning*, 2(4), 1988: 245–318.

[15] M. Minsky and S. Papert, *Perceptrons*. MIT Press, Cambridge, MA., 1969. (Expanded edition 1988.)

[16] B. K. Natarajan. On learning sets and functions. *Machine Learning*, 4(1), 1989: 67–97.

[17] A. Ngom, C. Reischer, D. Simovici and I. Stojmenović. Learning with permutably homogeneous multiple-valued multiple-threshold perceptrons. In *Proceedings of the 28th IEEE International Symposium on Multiple-Valued Logic*, IEEE Press, 1998.

[18] A. Ngom, A. Obradović and I. Stojmenović. Minimization of multiple-valued multiple-threshold perceptrons using genetic algorithms. In *Proceedings of the 28th IEEE International Symposium on Multiple-Valued Logic*, IEEE Press, 1998.

[19] A. Ngom, I. Stojmenović and J. Žunić. On the number of multilinear partitions and the computing capacity of multiple-valued multiple-threshold perceptrons. In *Proceedings of 29th IEEE International Symposium on Multiple-Valued Logic*, IEEE Press, 1999.

[20] N. J. Nilsson. *Learning Machines*. McGraw-Hill, New York, 1965.

[21] A. B. Novikoff. On convergence proofs on perceptrons. In *Symposium on the Mathematical Theory of Automata*, 12. Polytechnic Institute of Brooklyn, 1962: 615-622.

[22] Z. Obradović and I. Parberry. Learning with discrete multivalued neurons. *Journal of Computer and System Sciences* 49, 1994: 375–390.

[23] S. Olafsson and Y. S. Abu-Mostafa. The capacity of multilevel threshold functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10 (2), 1988: 277–281.

[24] L. Pitt and L. Valiant. Computational limitations on learning from examples. *Journal of the ACM*, 35, 1988: 965–984.

[25] F. Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, 65, 1958: 386–407.

[26] R. Takiyama. Multiple threshold perceptron. *Pattern Recognition* 10, 1978: 27–30.

[27] R. Takiyama. The separating capacity of a multi-threshold element. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7, 1985: 112–116.

[28] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11), 1984: 1134–1142.

[29] V. N. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, New York, 1982.

[30] V.N. Vapnik and A.Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2), 1971: 264–280.