Pergamon

Pattern Recognition, Vol. 28, No. 6, pp. 879–899, 1995
Elsevier Science Ltd
Pattern Recognition Society
Printed in Great Britain.
0031–3203/95 \$9.50 + .00

0031–3203(94)00161–8

MIN–MAX CLASSIFIERS: LEARNABILITY, DESIGN AND APPLICATION*

PING-FAI YANG†‡ and PETROS MARAGOS†§
‡ AT&T Bell Laboratories, 600 Mountain Ave., Murray Hill, NJ 07974, U.S.A.
§ School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, U.S.A.

(*Received* 9 *February* 1994; *in revised form* 27 *October* 1994; *received for publication* 14 *December* 1994)

Abstract—This paper introduces the class of min–max classifiers. These are binary-valued functions that can be used as pattern classifiers of both real-valued and binary-valued feature vectors. They are also lattice-theoretic generalization of Boolean functions and are also related to feed-forward neural networks and morphological signal operators.

We studied supervised learning of these classifiers under the Probably Approximately Correct (PAC) model proposed by Valiant. Several subclasses of thresholded min–max functions are shown to be learnable, generalizing the learnability results for the corresponding classes of Boolean functions.

We also propose a LMS algorithm for the practical training of these pattern classifiers. Experimental results using the LMS algorithm for handwritten character recognition are promising. For example, in our experiments the min–max classifiers were able to achieve error rates that are comparable or better than those generated using neural networks. The major advantage of min–max classifiers compared to neural networks is their simplicity and the faster convergence of their training algorithm.

Pattern classification Machine learning Mathematical morphology Image processing Character recognition

1. INTRODUCTION

There is much interest in the field of pattern recognition on trainable pattern classifiers, as seen, for example, in the growth in the area of neural networks. Parallel to this development, in the field of machine learning there have been many theoretical advances on distribution-free learning of Boolean functions. This learning framework is known as the *probably approximately correct* (PAC) model, pioneered by Valiant(1) and further developed by him and other researchers. There is already a wealth of literature about the PAC learning model; examples include Valiant,(1,2) Blumer *et al.*,(3,4) Haussler,(5) Kearns *et al.*(6) Kearns,(7) Rivest(8) and Schapire.(9) Most of the results in PAC learning deal with Boolean functions. If such functions are used as (Boolean) pattern classifiers, then the input features must be binary-valued. Although this may be sufficient for classifying high-level predicate-like features, most of the pattern recognition applications, such as computer speech and object recognition, involve real-valued feature vectors.

In this paper, we present the class of *min–max classifiers*‖ and study methods of their automatic design. These classifiers can accept as inputs both real-valued and binary-valued feature vectors. Each input variable to these functions is in the range $[0, 1]$, in contrast to $\{0, 1\}$ for the Boolean classifiers. Moreover, these min–max classifiers are natural generalizations of the Boolean functions, because they are based on MIN/MAX operations which are the lattice-theoretic counterparts of Boolean AND/OR operations on real numbers. Therefore, the learnability results on min–max classifiers we present in this paper generalize the Boolean counterparts expounded in previous works.

Another motivation for working with the min–max classifiers is their close relation to a large class of nonlinear signal/image operators known as morphological filters, which are defined via min–max operations on their inputs. As discussed in Serra(10) and Maragos and Schafer,(11) these min–max morphological operators can be applied to a broad variety of feature extraction and shape analysis/detection tasks in images or arbitrary geometrical objects. Hence, learning of the min–max classifiers provides an ability for automated training of the above feature extraction and shape analysis/detection signal operators.

*This research was supported by the National Science Foundation under Grant MIPS-86-58150 with matching funds from Xerox, and in part by the ARO Grant DAAL03-86-K-0171 to the Brown–Harvard-MIT Center for Intelligent Control Systems.

† The authors were with the Division of Applied Sciences at Harvard University when this research was done.

‖ We shall use the terminologies "min–max classifier" and "thresholded min–max function" interchangeably.

In this paper we show that three subclasses of thresholded min–max functions are learnable in the PAC model. We achieved these results by providing both polynomial bounds on the number of training samples and devising polynomial time learning algorithms. The subclasses are direct generalizations of the Boolean positive term, positive clause, and k-DNF functions. This last subclass of min–max classifiers is the most interesting since it is related to morphological filters which are maximum of minimum over a local moving window.

Besides the theoretical learnability results, we also propose a *Least Mean Square* (LMS) algorithm for actual design of min–max classifiers. Salembier(12) used an LMS algorithm to train morphological filters that are defined via min–max operations. We adapted his approach to the design of min–max classifiers and also improved it by refining the computation of the partial derivatives involved. The improvement translates to lower training error rate. We then applied the LMS algorithm to the problem of handwritten character recognition. Our simulations show that min–max classifiers are desirable alternatives to more traditional classifiers such as feed-forward neural networks. Their main advantages are simplicity and generally faster convergence of the training algorithm.

This paper is organized as follows: in Section 2, we define the min–max functions and their thresholded counterparts. A discussion of their relations with other classifiers is included. We also investigate aspects of their representation power using techniques from mathematical morphology. The PAC learnability results are presented in Section 3, where we show that several subclasses of threshold min–max functions are learnable under the PAC learning model. Then in Section 4 we turn to practical training of thresholded min–max functions using the LMS algorithm. The final section (Section 5) describes the application of the LMS algorithm to handwritten character recognition, which contains a discussion of the feature extraction procedures and the experimental results.

2. MIN–MAX FUNCTIONS

Here we start from general definitions about min–max functions and then use tools from mathematical morphology to explore some of their properties and relations to Boolean functions and perceptrons.

2.1. *Definitions*

A Boolean function $B(\vec{b})$, $\vec{b} = (b_1, \ldots, b_d) \in \{0,1\}^d$, in disjunctive normal form (DNF) is a finite disjunction (i.e., Boolean OR) of *terms*. A term is a conjunction (i.e., Boolean AND) of *literals*. A literal is either a Boolean variable $b_i \in \{0,1\}$ or its complement $\bar{b}_i$. To generate a *min–max function* from a DNF Boolean function, we replace the uncomplemented Boolean inputs b_i with real-valued variables $x_i \in [0,1]$, complemented variables $\bar{b}_i$ with real complements $x'_i \triangleq 1 - x_i$, and the Boolean AND/OR with MIN/MAX (denoted by $\wedge / \vee$).*

Formally, let $\vec{x} = (x_1, x_2, \ldots, x_d)$ be a real-valued vector in the d-dimensional unit cube $[0,1]^d$. We define a *min–max* function $f: [0,1]^d \to [0,1]$ with input $\vec{x}$ as the function

$$f(x_1, x_2, \ldots, x_d) = \bigvee_j \bigwedge_{i \in I_j} l_i, \quad l_i \in \{x_i, 1 - x_i\} \tag{1}$$

where an arguement l_i is called a *literal*, equal either to a variable x_i or its complement x'_i. Each minimum function $\wedge_{i \in I_j} l_i$ is called a *min term*. Each I_j denotes the set of coordinates of the input vector $\vec{x}$ that appear in the argument of the j-th min term. The *size of a min term* is the number of literals in the minimum function. The maximum $\vee_j$ has a finite number of terms. Thus, a min–max function is a finite maximum of min terms. Note that the restriction of a min–max function on the finite discrete space $\{0,1\}^d$ is a Boolean function.

A Boolean function B is called *monotone* (or positive) if $B(\vec{a}) \le B(\vec{b})$ whenever $\vec{a} \le \vec{b}$, where $\vec{a} \le \vec{b}$ means $a_i \le b_i$ for all i. Gilbert(13) showed that B is monotone if and only if all its variables appear uncomplemented. Similarly we call a function $g: [0,1]^d \to [0,1]$ *monotone* if

$$\vec{x} \le \vec{y} \Rightarrow g(\vec{x}) \le g(\vec{y}), \quad \forall \vec{x}, \vec{y} \tag{2}$$

It can be shown that a min–max function is monotone if and only if it admits an expression that does not contain any complemented variables.

To use a min–max function f as a classifier performing binary decisions we need to threshold f at some arbitrary value $\theta \in [0,1]$. This creates a *thresholded min*–max function (*min–max classifier*) $f_\theta: [0,1]^d \to \{0,1\}$ defined by

$$f_\theta(\vec{x}) = P[f(\vec{x}) \ge \theta] = \begin{cases} 1 & \text{if } f(\vec{x}) \ge \theta, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

$P(\cdot)$ is called the predicate function. An example of a thresholded min–max function is $P\{((x_1 \wedge x_4) \vee (x_2 \wedge (1 - x_3) \wedge x_5)) \ge 0.6\}$. It is generalized from the Boolean function $(b_1 \cdot b_4 + b_2 \cdot \bar{b}_3 \cdot b_5)$. The min term $(x_1 \wedge x_4)$ is size two while $(x_2 \wedge (1 - x_3) \wedge x_5)$ is size three. In the second min term, the variable x_3 is complemented. Note that there are an infinite number of thresholded min–max functions corresponding to a Boolean function. This is due to the freedom in the choice of the threshold value θ, which in our work will be a free parameter to be learned. A thresholded min–max function is monotone if its corresponding min–max function is monotone.

Exchanging the roles of AND and OR in a DNF Boolean function transforms the latter into a conjunc-

*In this paper, Boolean AND is denoted by the product symbol '·', which may be left out occasionally. The Boolean OR is denoted by '+'. The symbol $\vee$ and $\wedge$ are defined as $\vee_{n \in I} x_n = \max_n\{x_n\}$ and $\wedge_{n \in I} x_n = \min_n\{x_n\}$ if the index set I is finite; if I is infinite, then the max and min should be replaced by supremum and infimum, respectively.

tive normal form (CNF). Similarly, exchanging the roles of MIN and MAX in a (thresholded) min–max function will yield a (thresholded) *max–min function*, i.e., a (thresholded) minimum of maxima. Due to the straightforward duality relationships between these two latter function classes, in this paper we focus on functions in the min–max form.

An input vector is classified as a *positive* or *negative* instance of a thresholded min–max function $c(\vec{x})$ accordingly to whether the output of $c(\vec{x})$ is 1 or 0, respectively. We shall also refer to this classification process as the *labeling* of the input vector $\vec{x}$ by $c(\vec{x})$, and call general 0-1-valued functions *classifier* functions to emphasize the possibility of their use as pattern classifiers. In the setting of our learning model, classifier functions are also referred to as *concept functions*, or simply *concepts*. We shall use the latter name more often in the rest of the paper. A collection of concepts is called a *concept class*, which is usually denoted by $\mathscr{C}$. The set of all thresholded min–max functions with d variables is denoted by $\mathscr{C}^d_{min-max}$.

2.2. *Morphological representations and relations to Boolean functions*

Here we establish some relationships between (thresholded) min–max functions and Boolean functions using concepts from morphological filtering as discussed in Maragos and Schafer.[14] First note the following three useful properties of thresholding, which can be easily proven. The minimum and maximum functions obey a *threshold homomorphism* property:

$$\begin{aligned} P(x \wedge y \geq \theta) &= P(x \geq \theta) \wedge P(y \geq \theta) \\ &= P(x \geq \theta) \cdot P(y \geq \theta), \end{aligned} \tag{4}$$

$$\begin{aligned} P(x \vee y \geq \theta) &= P(x \geq \theta) \vee P(y \geq \theta) \\ &= P(x \geq \theta) + P(y \geq \theta). \end{aligned} \tag{5}$$

In addition, we have the *threshold reconstruction* property:

$$x = \int_0^1 P(x \geq \theta)\, d\theta, \quad \forall x \in [0, 1] \tag{6}$$

From (1), (3) and the above properties it follows that

$$f_\theta(x_1, x_2, \ldots, x_d) = \bigvee_j \bigwedge_{i \in I_j} P(l_i \geq \theta), \quad l_i \in \{x_i, 1 - x_i\} \tag{7}$$

Thus, a thresholded min–max function is equal to the disjunction of terms containing Boolean variables formed by thresholding the input coordinates x_i or their complements. Turning to the thresholding of a complemented variable,

$$P(x' \geq \theta) = P(1 - x \geq \theta) = P(x \leq 1 - \theta). \tag{8}$$

The thresholding of $x' = 1 - x$ is not equal to the Boolean complement $\overline{P(x \geq \theta)} = P(x < \theta)$ in general. However, this particular definition of x' remains a reasonable choice because it is identical to the Boolean complement if x takes on only 0, 1 values. It also preserves the range of the variable; i.e., $x \in [0, 1] \Rightarrow x' \in [0, 1]$.

To understand the behavior of the thresholded min–max function, it is helpful to consider its geometrical properties. The simplest form of such functions is the thresholding of a single minimum or a single maximum. The decision regions of these for the special case of $d = 2$ is shown in Fig. 1.

The positive region for the thresholded minimum function is the axes-parallel square whose sides intersect the axes at θ (the threshold) and with one vertex at (1, 1), while that of the thresholded maximum function is an L-shaped region formed by deleting the square in the lower left hand corner of the domain $[0, 1]$.[2] These conclusions were drawn using the thresholded homomorphism properties [equations (4) and (5)]. For

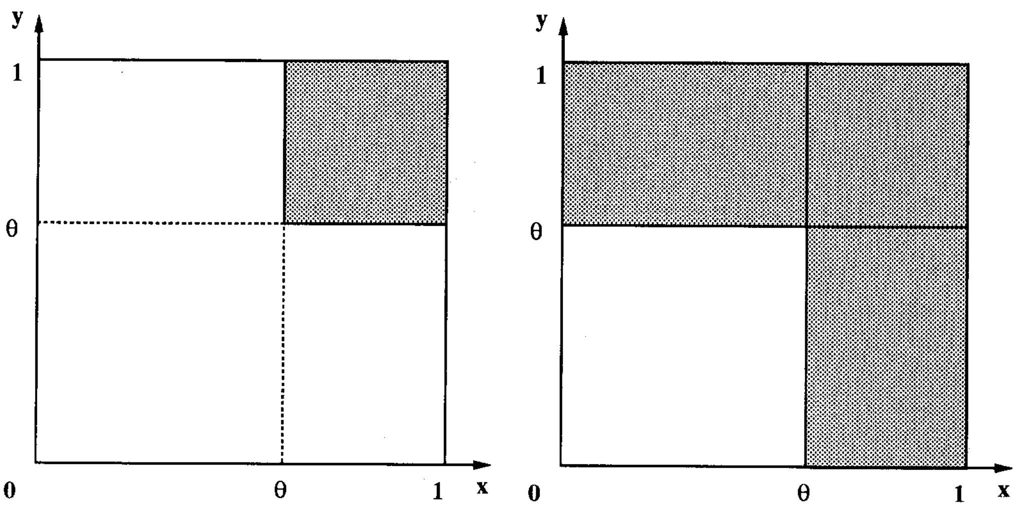


Fig. 1. In the special case of $d = 2$, the positive regions of the thresholded minimum function (shaded area in left figure) takes the form of an axes-parallel square with one vertex at (1, 1); while that of a thresholded maximum function (right) is an L-shaped region formed by cutting out the unshaded square.

example, the positive region of $P(x \wedge y \geq \theta)$ is equal to $\{x \geq \theta\} \cap \{y \geq \theta\}$ (set intersection because of the Boolean AND). For arbitrary dimensions, the positive region of the thresholded minimum function becomes a d-dimensional cube with one vertex at $(1, \ldots, 1)$. As for the maximum function, the *negative* region becomes a d-dimensional cube near the origin. We remark that this is related to the axes-parallel rectangles described in Blumer *et al.*[4] However, the axes-parallel rectangles cannot be represented using thresholded min–max functions, which is the primary interest in this work.

Next we present a result that indicates the representation power of monotone min–max functions. Some definitions are needed first: Consider arbitrary functions $f: [0,1]^d \to [0,1]$ that are *consistent* generalizations of Boolean functions, i.e., their value is binary whenever the input vector is binary; formally

$$f(\vec{x}_\theta) \in \{0,1\}, \quad \forall \theta \in [0,1] \tag{9}$$

where $\vec{x}_\theta \in \{0,1\}^d$ is a thresholded (and hence binary) input vector:

$$\vec{x}_\theta = (x_{1,\theta}, \ldots, x_{d,\theta}) \overset{\Delta}{=} (P(x_1 \geq \theta), \ldots, P(x_d \geq \theta)) \tag{10}$$

We also say that f commutes with thresholding if

$$f_\theta(\vec{x}) = f(\vec{x}_\theta), \quad \forall \vec{x}, \theta \tag{11}$$

Commuting with thresholding is an important property since it implies that the Boolean function $f(\vec{x}_\theta)$ obtained from f by thresholding the input vector at any θ (and hence restricting f on the finite discrete space $\{0,1\}^d$) gives identical values with thresholding the real-valued output of f at θ. It will be shown in the following theorem that if f commutes with thresholding, then it is monotone. Further, this theorem establishes that functions that commute with thresholding can be represented by monotone min–max functions.

Theorem 1. Let $f: [0,1]^d \to [0,1]$ be a function that obeys property (9). Then f commutes with thresholding if and only if it is monotone min–max function, or equivalently if and only if is a min–max function without any complemented variables.

Proof. Let f commute with thresholding. Consider binary vectors $\vec{a} \leq \vec{b} \in \{0,1\}^d$. We can always find some real vector $\vec{x}$ such that $\vec{a} = \vec{x}_{\theta_1}$ and $\vec{b} = \vec{x}_{\theta_2}$ with $\theta_1 \geq \theta_2$. Then, if B is the Boolean function corresponding to f, sine $f(\vec{x}_\theta) = B(\vec{x}_\theta)$ for each θ, we have

$$B(\vec{a}) = f(\vec{x}_{\theta_1}) = f_{\theta_1}(\vec{x}) \leq f_{\theta_2}(\vec{x}) = f(\vec{x}_{\theta_2}) = B(\vec{b}) \tag{12}$$

Hence B is monotone. Since the monotone B admits a DNF expression as a unique irreducible OR of position AND terms, it follows from (6) that:

$$f(\vec{x}) = \int_0^1 f_\theta(\vec{x})\,d\theta = \int_0^1 f(\vec{x}_\theta)\,d\theta$$

$$= \int_0^1 \bigvee_j \bigwedge_{i \in I_j} P(x_i \geq \theta)\,d\theta$$

$$= \bigvee_j \int_0^1 P\left(\bigwedge_{i \in I_j} x_i \geq \theta\right) d\theta$$

$$= \bigvee_j \bigwedge_{i \in I_j} \int_0^1 P(x_i \geq \theta)\,d\theta = \bigvee_j \bigwedge_{i \in I_j} x_i$$

Hence f is equal to a min–max function. Further f is monotone because its min–max representation contains no complements. Conversely, let f be a monotone min–max function. Then

$$f(\vec{x}) = \bigvee_j \bigwedge_{i \in I_j} x_i \Rightarrow f_\theta(\vec{x}) = \bigvee_j \bigwedge_{i \in I_j} P(x_i \geq \theta) = f(\vec{x}_\theta) \tag{13}$$

This follows by direct substitution and applying threshold homomorphism twice. Hence f commutes with thresholding, which completes the proof. □

The essence of the above theorem is that any monotone real-input real-output function that yields a binary output whenever the input vector is binary and commutes with thresholding can be represented as a min–max function (with no complements). Conversely, the class of thresholded monotone min–max functions is almost isomorphic to Boolean functions, except for the generally unknown parameter θ which is to be learned.

2.3. *Relations to other classifiers*

Beside the Boolean classifiers, another class of classifiers the thresholded min–max functions are related to three layer perceptrons. The link is provided by the thresholded homomorphism properties (4) and (5). We demonstrate this using an example. Consider the thresholded min–max function

$$P(x_1 \vee (x_2' \wedge x_4) \geq \theta). \tag{14}$$

Applying first (5) and then (4), we derive an equivalent function

$$P(x_1 \geq \theta) + (P(x_2' \geq \theta) \cdot P(x_3 \geq \theta)).$$

Finally, we use (8) to arrive at the desired form

$$P(x_1 \geq \theta) + (P(x_2 \leq 1 - \theta) \cdot P(x_3 \geq \theta)). \tag{15}$$

Observe that each predicate function in the above expression is the thresholding of a single variable, which can be implemented using a single layer perceptron. The Boolean conjunctions and disjunctions can also be implemented using single layer perceptrons. Therefore the original thresholded min–max function in (14) can be implemented using a three-layer perceptron. It is easy to see that any thresholded min–max function can be implemented using a three layer perceptron because the thresholded min–max function is formed by the composition of several minima and a maximum. It is not true that any three layer perceptron can be expressed as a thresholded min–max function. To see the reason, one simply has to look at (15). The first layer of perceptrons have the form $P(l_i \geq \theta)$. Their decision regions are parallel to the coordinate axes.

Since the second and third layers are Boolean AND and OR respectively, the positive region of the cascaded structure should have boundaries parallel to the axes. For a general three-layer perceptron, this condition is not necessarily true. Therefore, $\mathscr{C}^d_{min-max}$ is a *subclass* of the class of general three layer perceptrons.

Another type of classifier that is related to thresholded min–max functions are the axes-parallel rectangles described in Blumer *et al.*[4] However, they did not address the algebraic representation of axes-parallel regions using min–max functions. Finally, in reference (15) the authors mentioned decision trees that have axes-parallel decision nodes. It is obvious that the min–max classifiers can be constructed using them.

The structure of a min–max classifier can be envisioned as a feed forward network (See Fig. 2), the first layer consisting of minimum functions, the second a maximum and the third a thresholding. Networks with nodes as rank order operators have appeared in the literature previously. For example, Palmieri[16,17] considered networks with nodes that output the sorted values of their inputs, and trained the multiplicative weights connecting the successive layers. In our work, we train the subsets of vector components on which the nonlinear elements operate. Wilson[18] also used networks of "weighted rank order", which replaced the rank ordering of the inputs by ordering of them with additive weights. Training of his network is equivalent to finding these additive weights.

3. PAC LEARNING

The PAC model is an attempt at formalizing the study of machine learning. It was proposed by Valiant in reference (1), and since then there is much theoretical development related to PAC (for example, see the references at the beginning of Section 1). Attractive features of PAC include its emphasis on efficient algorithms and the lack of assumption on the probability distribution of the feature space. In this section, we shall show that the following three subclasses of $\mathscr{C}^d_{min-max}$ are learnable under the PAC model.

Thresholded monotone minimum functions. A thresholded monotone minimum function has the general form: $P(\wedge_{i\in I} x_i \geq \theta)$, where I is the set of coordinate indices of the input vector. This class of functions is denoted by the symbol $\mathscr{C}^d_{min}$.

Thresholded monotone maximum functions. These functions are dual forms of the thresholded minimum functions. The general form is $P(\vee_{i\in I} x_i \geq \theta)$, where I is again the set of coordinate indices of the input vector. The collection of all thresholded monotone maximum functions is denoted by the symbol $\mathscr{C}^d_{max}$.

Thresholded k-min–max functions. They are thresholded min–max functions with the restriction on the size of each min term to be $\leq k$. The class is denoted by the symbol $\mathscr{C}^d_{k\text{-}min-max}$.

These classes of functions are generalizations of the Boolean positive term, positive clause, and k-DNF functions, respectively.

3.1. *Background*

We now present some of the key ideas and results from PAC learning theory. For a more detailed description we refer the reader to, for example, Valiant,[1] Kearns[7] and Rivest.[8]

One of the assumptions in PAC is the *consistency* of the training data, which stipulates the following scenario for the generation of the training data: each time the learning algorithm requests a training example from an *oracle*, the latter outputs a vector $\vec{x}$ which is selected from the domain using an unknown probability measure; together with this it outputs $t(\vec{x})$ which is the value of the unknown *target concept*. In addition it is assumed that $t()$ belongs to the concept class $\mathscr{C}$. The goal in the PAC model is to construct a learning algorithm that inputs these samples of the target function (i.e. supervised learning), and outputs a concept function [the *hypothesis* $h()$] with small discrepancies from $t()$ [*true error rate* $\xi(h)$]. One possibility is to estimate the error rate from the sequence of training data. The estimate is called the *empirical error rate* [$\hat{\xi}(h)$]. Since the empirical error rate of the target concept must be equal to zero, one strategy for constructing learning algorithms is to return any concept $h(\vec{x})$ that has a zero empirical error rate. This type of algorithm is called *consistent* algorithm. In case the algorithm runs in polynomial time (in the number of training data), it has a special name: *poly-hy-fi* (for polynomial hypothesis finder). The following proposition shows that by using enough data, any consistent algorithm can be used as learning algorithm for $\mathscr{C}$ while meeting the PAC requirement.

Proposition 1. [Blumer *et al.*[4]] Let $\mathscr{C}$ be a nontrivial, well-behaved* concept class. If the Vapnik–Chervonenkis dimension of $\mathscr{C}$ is $VC(\mathscr{C}) < \infty$, then for $0 < \varepsilon, \delta < 1$, and training sample size at least

$$\max\left(\frac{4}{\varepsilon}\log_2\frac{2}{\delta}, \frac{8VC(\mathscr{C})}{\varepsilon}\log_2\frac{13}{\varepsilon}\right), \tag{16}$$

then with probability $\geq 1-\delta$, any consistent algorithm will return a hypothesis $h(\vec{x})$ with true error rate $\xi(h) \leq \varepsilon$.

The Vapnik–Chervonenkis (VC) dimension of a concept class $\mathscr{C}$ is the size of the largest finite subset S of the domain X which can be labeled in all possible ways

* A concept class is trivial if it has only one concept or it has two disjoint concepts such that $c_1 \cup c_2 = X$. The well-behavedness conditions are some measurability conditions on the functions, it is detailed in Appendix A of Blumer *et al.*[4]

using concepts in $\mathscr{C}$ (S is then said to be *shattered* by $\mathscr{C}$). A formal definition of this parameter is discussed in the same paper by Blumer *et al.*[4] From (16), the number of training data required is polynomial in d, $1/\varepsilon$, and $1/\delta$ if the VC dimension is polynomial in the dimension d. Hence, using this technique, the proof of learnability of $\mathscr{C}$ can be divided into two steps:

(1) show that the VC dimension of $\mathscr{C}$ is polynomial in d;
(2) find a poly-hy-fi for $\mathscr{C}$.

We follow this approach in this paper to show the learnability of the classes $\mathscr{C}^d_{min}, \mathscr{C}^d_{max}$ and $\mathscr{C}^d_{k-min-max}$.

3.2. *Learnability of thresholded monotone minimum functions*

It turns out the learnability of $\mathscr{C}^d_{max}$ and $\mathscr{C}^d_{k-min-max}$ depend on that of the thresholded monotone minimum functions. Therefore in this section we shall first discuss the results for $\mathscr{C}^d_{min}$ in more detail. The theorems are motivated by the geometry of the thresholded monotone minimum function which we discussed in Section 2.2.

3.2.1. *VC-dimension of the thresholded monotone minimum functions.* In this section, we show that $VC(\mathscr{C}^d_{min}) = d$ by providing upper and lower bound of $VC(\mathscr{C}^d_{min})$.

To find the upper bound, we shall prove that no sets of $>d$ elements in $[0, 1]^d$ can be shattered by $\mathscr{C}^d_{min}$. In other words, if $|S| > d$, then we cannot label the members of S in all $2^{|S|}$ possible ways by using solely thresholded monotone minimum functions. This result depends on Lemma 1, which restricts the possible choices of S that can be shattered by $\mathscr{C}^d_{min}$. Throughout this paper, we use the symbol x_k to denote the k-th coordinate of a general vector $\vec{x}$.

Lemma 1. Let S be a set of points in $[0, 1]^d$.

$\forall \vec{y} \in S, \exists c() \in \mathscr{C}^d_{min}$ such that $\{\vec{y}\}$ is labeled negative and $S \backslash \{\vec{y}\}$ is labeled positive

if and only if

$\forall \vec{y} \in S, \exists$ a coordinate index k such that $y_k < \bigwedge_{\vec{w} \in S\backslash\{\vec{y}\}} w_k$.

Proof. $\Leftarrow$: Assuming the second condition holds, we only have to demonstrate $|S|$ thresholded minimum functions that perform the $|S|$ partitionings as specified. For each point $\vec{y} \in S$, one possibility is:

$$P\left(\bigwedge_{i \in \{k\}} x_i \geq \theta\right)$$

where $\theta = \frac{1}{2}\{\bigwedge_{\vec{w} \in S\backslash\{\vec{y}\}} (w_k) + y_k\}$.

$\Rightarrow$: Consider the point $\vec{y}$. If $\vec{y}$ is labeled negative by some monotone thresholded minimum function $P(\bigwedge_{i \in I} x_i \geq \theta)$ while $S\backslash\{\vec{y}\}$ is labeled positive, we must have the following inequalities:

$$\bigwedge_{i \in I} y_i < \theta \leq \bigwedge_{\vec{w} \in S\backslash\{\vec{y}\}, i \in I} w_i$$

Suppose $y_k = \bigwedge_{i \in I} y_i$ then,

$$y_k = \bigwedge_{i \in I} y_i < \theta \leq \bigwedge_{\vec{w} \in S\backslash\{\vec{x}\}, i \in I} w_i \leq \bigwedge_{\vec{w} \in S\backslash\{\vec{x}\}} w_k$$

which proves the required condition. □

Using this lemma, we can show that no sets of size $> d$ can be shattered. This fact relies on a proof by contradiction presented in Theorem 2.

Theorem 2. No set of $m \geq d + 1$ points in $[0, 1]^d$ can be shattered by $\mathscr{C}^d_{min}$, i.e. $VC(\mathscr{C}^d_{min}) \leq d$.

Proof. Let S be a set of $m \geq d + 1$ points in the d-dimensional unit hypercube. Suppose the set can be shattered, there must be m concept functions, each labeling only one of the m points negative.

Since $m \geq d + 1$, and each of the point in S must satisfy the condition stated in Lemma 1, there must be two distinct vectors $\vec{x}, \vec{y} \in S$ and a coordinate axis k such that

$$x_k < \bigwedge_{\vec{w} \in S\backslash\{\vec{x}\}} w_k \leq y_k$$

The first inequality follows from Lemma 1 while the second is a consequence of $\vec{y} \in S\backslash\{\vec{x}\}$. By exchanging the role of $\vec{x}$ and $\vec{y}$ in the above derivation, we get both $x_k < y_k$ and $y_k < x_k$, a contradiction. □

Turning to the lower bound, we construct sets of d elements in $[0, 1]^d$ that can be shattered by $\mathscr{C}^d_{min}$.

Theorem 3. There exists a set of d points in $[0, 1]^d$ that is shattered by $\mathscr{C}^d_{min}$, i.e. $VC(\mathscr{C}^d_{min}) \geq d$.

Proof. The following set is shattered by $\mathscr{C}^d_{min}$:

$$S = \{\vec{x}^1, \ldots, \vec{x}^d\}$$

$$x^i_k = \begin{cases} \frac{3}{4}, & \text{if } i \neq k, \\ \frac{1}{4}, & \text{if } i = k, \end{cases}$$

where x^i_k denotes the k-th coordinate of the i-th vector in S. In fact, to label only $\{x^j | j \in T\}$, $T \subset \{1, \ldots, d\}$ negative, use

$$P\left(\bigwedge_{j \in T} x_j \geq \tfrac{1}{2}\right)$$

To label all of them positive, use

$$P\left(\bigwedge_{j=1}^{d} x_j \geq \tfrac{1}{8}\right)$$

It follows immediately from Theorems 2 and 3 that the VC dimension of $\mathscr{C}^d_{min}$ is d.

3.3. *Poly-hy-fi for* $\mathscr{C}^d_{min}$

The symbols used in this section are listed below:

$[\vec{x}(n), l(n)]$ = a training sample, with $\vec{x}(n)$ being the input vector and $l(n)$ the label,
$(\vec{x}^+, 1)$ = a general positive training data,
$(\vec{x}^-, 0)$ = a general negative training data,

$x_c(n) =$ the c-th coordinate of the vector $\vec{x}(n)$
$x_c^+ =$ the c-th coordinate of the vector $\vec{x}^+$,
$x_c^- =$ the c-th coordinate of the vector $\vec{x}^-$,
$N =$ total number of training data,

Under the PAC learning model, the labels are assumed to be generated by a target function in the concept class, i.e. $l(n) = t[\vec{x}(n)], t() \in \mathscr{C}^d_{min}$.

In the first step of the algorithm, d threshold estimates $\hat{\theta}_l$, $1 \le l \le d$ are produced. The l-th estimate is calculated by taking the minimum of the l-th coordinate of all the positive training vectors. Since all the positive training data are labeled positive by the unknown target function $t(\vec{x}) = P(\wedge_{i \in I} x_i \ge \theta)$, the value $\wedge_{l \in I} \hat{\theta}_l$ is a good estimate for the actual threshold θ. The immediate question is how to find the coordinate indices I.

In the second step, the algorithm uses these d estimates to generate d thresholded monotone minimum functions $h_l(\vec{x}) = P(\wedge_{i \in I_l} x_i \ge \hat{\theta}_l)$. The coordinate list I_l is initialized to be $\{1, \ldots, d\}$ for all values of l. Denote $\mathscr{I} = \{I_l : 1 \le l \le d\}$. Then, the algorithm eliminates from I_l all coordinates c such that $\exists \vec{x}^+, x_c^+ < \hat{\theta}_l$. The rationale for this step comes from the threshold homomorphism property [equation (4)]

$$P\left(\bigwedge_{i \in I_l} x_i \ge \hat{\theta}_l\right) = \prod_{i \in I_l} P(x_i \ge \hat{\theta}_l).$$

If there is an index $c \in I_l$ such that $x_c^+ < \hat{\theta}_l$, the positive data will be labeled negative. Therefore, the variable c should be removed from I_l. If an I_l becomes empty after this step, it is removed $\mathscr{I}$ and therefore the corresponding concept h_l is removed from further consideration.

In the final step, $h_l()$ is eliminated if it is inconsistent with any negative training data. Therefore, the remaining thresholded monotone minimum functions are consistent with all the training data. One of them is returned as the hypothesis.

Before we present the correctness proof, we shall show that the algorithm is indeed polynomial time. We assume that comparison requires unit time. The first step takes at most Nd comparisons (there are at most N positive training data, each one requires d comparisons). The next one requires at most Nd^2 comparisons (there are at most d indices lists I_l, and each generates at most d comparisons for each positive training data.) The final one also takes Nd^2 comparisons with the same reasoning. Total number of comparisons is

$$Nd^2 + Nd^2 + Nd = O(Nd^2),$$

which is polynomial in the number of training data and the dimension d. Since N is polynomial in d, $1/\varepsilon$ and $1/\delta$, this consistent hypothesis runs in time polynomial in these variables too. Therefore, the PAC condition is met.

Theorem 4 asserts the correctness of the algorithm.

Theorem 4. The algorithm presented in this section always returns a thresholded monotone minimum function that is consistent with all the training data.

Proof. It is obvious that the hypothesis returned by this algorithm is consistent with all the training data. We only have to show that after the last step, $\mathscr{I}$ is not empty.

Suppose the target function has the functional form

$$t(\vec{x}) = P\left(\bigwedge_{i \in I} x_i \ge \theta\right).$$

In the first step, one of the estimates $\hat{\theta}_l$ must be equal to

$$\hat{\theta} = \bigwedge_{\vec{x}^+} \bigwedge_{i \in I} x_i^+ = \bigwedge_{i \in I}\left(\bigwedge_{\vec{x}} x_i^+\right) = \bigwedge_{i \in I} \hat{\theta}_i$$

Denote the coordinate list corresponding to this threshold value as $\hat{I}$. After the second step $\hat{I}$ is not empty, and will not be eliminated from $\mathscr{I}$. Moreover, the variables present in I are not removed, i.e. $I \subseteq \hat{I}$. This fact is easy to show using the definition of $\hat{\theta}$. Finally, $\hat{I}$ is not removed from $\mathscr{I}$ in the last step. To show this, observe that

$$\hat{\theta} \ge \theta$$

and hence for any negative training data $\vec{x}^-$ there is a coordinate $c \in I$ such that

$$x_c^- < \theta \le \hat{\theta}$$

because the $\vec{x}^-$ must be labeled negative by the target concept. Using the fact that $I \subseteq \hat{I}$, it is obvious that the hypothesis corresponding to $\hat{\theta}, \hat{I}$ must be labeled negative for any negative training data. □

It is important to note that we have not assumed the independence of the input variables $\{x_1, \ldots, x_d\}$. They can in fact be functions of each other. The only assumption made is that the target function takes the form of a thresholded monotone minimum function in these variables. This observation is important especially in Section 3.5, where the input variables for the poly-hy-fi are functionally related.

3.4. *Learnability of thresholded monotone maximum functions*

All the results in the previous section can be transcribed to apply to $\mathscr{C}^d_{max}$ by using the duality relation between thresholded monotone maximum and thresholded monotone minimum functions:

$$\overline{P\left(\bigwedge_{i \in I} (1 - x_i) > 1 - \theta\right)} = \overline{P\left(\bigvee_{i \in I} x_i < \theta\right)}$$
$$= P\left(\bigvee_{i \in I} x_i \ge \theta\right). \quad (17)$$

The first equality is derived from the relation $\vee_i x_i = 1 - \wedge_i (1 - x_i)$. The leftmost function resembles a complemented thresholded monotone minimum function of complemented inputs, with the exception of the definition of the thresholding ($> \theta$ instead of $\ge \theta$). It is easy to show that the results in Section 3.2 for $P(\wedge_i x_i \ge \theta)$ also hold for $P(\wedge_i x_i > \theta)$ upon the replacement of $\geq$ by $>$. Therefore to convert the results for $\mathscr{C}^d_{min}$ to apply to $\mathscr{C}^d_{max}$, we use the following substitutions:

- Replace minimum by maximum and vice versa;
- Replace "positive" by "negative" and vice versa.
- Replace inputs with complemented values.

In the following, we provide a summary of the theoretical results for $\mathscr{C}^d_{max}$. The proofs are omitted and we refer the reader to Yang and Maragos[19] for details.

Lemma 2. Let S be a set of points in $[0, 1]^d$.

$$\forall \vec{y} \in S, \exists c \in \mathscr{C}^d_{max} \text{ such that } \{\vec{y}\} \text{ is labeled positive and } S \backslash \{\vec{y}\} \text{ is labeled negative}$$

if and only if

$$\forall \vec{y} \in S, \exists \text{ a coordinate index } k \text{ such that } y_k > \bigvee_{w \in S \backslash \{\vec{y}\}} w_k$$

Theorem 5. No set of $n \geq d+1$ points in $[0, 1]^d$ can be shattered by $\mathscr{C}^d_{max}$, i.e. $VC(\mathscr{C}^d_{max}) \leq d$.

Theorem 6. There exists a set of d elements which is shattered by $\mathscr{C}^d_{max}$, i.e. $VC(\mathscr{C}^d_{max}) \geq d$.

Combining the two theorems, we immediately find that $VC(\mathscr{C}^d_{max}) = d$.

Turning to the poly-hy-fi, we can use the threshold homomorphism and the duality between Boolean AND/OR to transform the poly-hy-fi for $\mathscr{C}^d_{min}$ to one for $\mathscr{C}^d_{max}$. In the general discussion in Section 3.3, swap the words "positive"/"negative" and "minimum"/"maximum", and replace "conjunction" by "disjunction" and "$x_c^+ < \theta_k$" with "$x_c^- \geq \theta_k$". The basic operations in this algorithm are the same as its $\mathscr{C}^d_{min}$ counterpart, so the computational complexity remains polynomial $[O(nd^2)]$. The correctness proof in Section 3.3 can also be adapted using duality.

3.5. *Learnability of thresholded k-min–max functions*

A general thresholded k-min–max function has the form $P(\bigvee_n T_n \geq \theta)$, with T_n denoting a min term of size at most k (i.e. a minimum function with at most k literals in its argument). This form is very suggestive of the connection between thresholded k-min–max function and thresholded monotone maximum function: the k-min–max function $(\bigvee_n T_n)$ is a maximum of the uncomplemented min terms T_n. Using this observation, the evaluation of a thresholded k-min–max function can be broken up into two parts—the first step calculates the values of *all* min terms with size $\leq k$. This can be considered a *remapping* of the input variables $\vec{x}$ into the set of min terms $R = \{r_n = \bigwedge_{i \in I, |I| \leq k} l_i\}$. The min terms T_n will be elements of R. These are the dependent variables of the thresholded monotone maximum function that is evaluated in the second step: $P(\bigvee_{\{n|r_n = T_n\}} r_n \geq \theta)$. Therefore, any thresholded k-min–max function is equivalent to a thresholded monotone maximum function in a higher dimensional space. In other words, we establish a mapping between the class $\mathscr{C}^d_{k\text{-}min\text{-}max}$ and a class of thresholded maximum functions with a larger number of input variables $(\mathscr{C}^{|R|}_{max})$.

To illustrate the remapping idea, we shall use the following thresholded min–max function

$$P(x_1 \vee (x'_2 \wedge x_3) \geq \theta). \tag{18}$$

This function belongs to $\mathscr{C}^3_{2\text{-}min\text{-}max}$, i.e. the class of thresholded min–max functions with three input variables and at most two literals in each of the min term. Following the remapping scheme, we introduce a set of new variables r_n which are min terms of x_1, x_2, x_3 with at most two literals. The new variables are listed below.

$$\begin{array}{lll}
r_1 = x_1, & r_2 = x_2, & r_3 = x_3, \\
r_4 = x'_1, & r_5 = x'_2, & r_6 = x'_3, \\
r_7 = x_1 \wedge x_2, & r_8 = x_1 \wedge x_3, & r_9 = x_2 \wedge x_3, \\
r_{10} = x'_1 \wedge x_2, & r_{11} = x_1 \wedge x'_2, & r_{12} = x'_1 \wedge x'_2, \\
r_{13} = x'_1 \wedge x_3, & r_{14} = x_1 \wedge x'_3, & r_{15} = x'_1 \wedge x'_3, \\
r_{16} = x'_2 \wedge x_3, & r_{17} = x_2 \wedge x'_3, & r_{18} = x'_2 \wedge x'_3, \\
r_{19} = x_1 \wedge x'_1, & r_{20} = x_2 \wedge x'_2, & r_{21} = x_3 \wedge x'_3.
\end{array}$$

Exact numbering of the new variables is irrelevant as long as all the possible min terms of size ≤ 2 are present. Note that the variables r_{19} through r_{21} are formed by taking the minimum of a variable with its complement. This is because the expression $P(x_i \wedge x'_i \geq \theta)$ is not always equal to 0. The second step in the process entails the introduction of a thresholded monotone maximum function that uses the r_n's as input. For the thresholded function in (18), the new function is $P(\bigvee_{i \in \{1,16\}} r_i \geq \theta)$. Other functions in $\mathscr{C}^3_{2\text{-}min\text{-}max}$ can be expressed as a thresholded monotone maximum function of $\vec{r}$. For example, $P(((x'_1 \wedge x_2) \vee (x'_1 \wedge x'_3)) \geq \theta)$ can be expressed as $P(r_{10} \vee r_{15} \geq \theta)$.

Denote the number of variables in the remapped vector $\vec{r}$ by $d' = |R|$. (In the example the dimension of the new vector is $d'' = 18$.) The parameter d' is a function of d and k. By a simple combinatorial argument, one can easily show that the functional form is:

$$d' = \binom{2d}{1} + \cdots + \binom{2d}{k} \leq k(2d)^k \leq (2d)^{(k+1)} \tag{19}$$

where $\binom{p}{q} = \frac{p!}{q!(p-q)!}$ is the combination. The upper bound on d' is polynomial in the parameter d when k is fixed.

Using the remapping idea, the domain of the k-min-max functions X can be mapped to a subset of a d' dimensional space X'. Also, any set of points $S \subset X$ can be mapped to $S' \subset X'$. From this observation, we can easily prove the following theorem.

Theorem 7. $VC(\mathscr{C}^d_{k\text{-}min\text{-}max}) \leq VC(\mathscr{C}^{d'}_{max})$.

Proof. Consider a set $S \subset X$ that is shattered by $\mathscr{C}^d_{k\text{-}min\text{-}max}$. The size of S is $VC(\mathscr{C}^d_{k\text{-}min\text{-}max})$. Using the remapping procedure, this set can be mapped to $S' \subset X'$ which has the same number of elements. Moreover, the output value of thresholded k-min–max function $c(\vec{x})$ is the same as that of the thresholded monotone maximum function $c'(\vec{r})$ after the remapping. Therefore, if the set S is shattered by a collection of concepts $\{c(\vec{x}) \in \mathscr{C}^d_{k\text{-}min\text{-}max}\}$, the set S' will also be shattered by the remapped functions $\{c'(\vec{r}) \in \mathscr{C}^d_{max}\}$. Since the set S' is shattered by $\mathscr{C}^{d'}_{max}$, the VC dimension of this concept class is bounded by the inequality

$$VC(\mathscr{C}^{d'}_{max}) \geq |S'| = VC(\mathscr{C}^d_{k\text{-}min\text{-}max})$$

$\square$

From Section 3.4, the VC dimension of $\mathscr{C}^d_{max}$ is d. Also, using the upper bound on d' in equation (19), we get the following bound on $VC(\mathscr{C}^d_{k\text{-}min\text{-}max})$.

Corollary 1. $VC(\mathscr{C}^d_{k\text{-}min\text{-}max}) \leq (2d)^{(k+1)}$.

Turning to the learning algorithm for $\mathscr{C}^d_{k\text{-}min\text{-}max}$ we found that any thresholded k-min–max function becomes a thresholded monotone maximum function in the new variables r_i. Moreover, there are only a polynomial number of remapped variables r_i. Therefore, the sequence of training data $(\vec{x}^j, l^j)$ can first be mapped into the new coordinates $(\vec{r}^j, l^j)$. The result is fed to the poly-hy-fi for $\mathscr{C}^d_{max}$. It will return a hypothesis in the remapped variable, which can be converted back to a thresholded k-min–max function easily by replacing the coordinates r_i by the corresponding minimum function on $\vec{x}$. Assuming unit time for computing a comparison, the amount of time required by the remapping step is at most $ndd' \leq nd(2d)^{(k+1)}$. Therefore the total time required by the algorithm is

$$ndd' + (nd' + nd'^2) = (d+1)nd' + nd'^2 \leq n\{(d+1)(2d)^{k+1} + (2d)^{2(k+1)}\} = O(n(2d)^{2(k+1)}).$$

3.6. *Difficulty of learning in the presence of noise*

The proofs for learnability in the previous sections hinges on the assumption that the training data can be labeled without error by at least one function within the concept class. Despite the intellectual clarity of the PAC model, for practical applications we think this condition to be too stringent, which led us to investigate the possibility of finding polynomial time efficient learning algorithms that can operate without this assumption of consistency. In this more general setting, it is still possible to obtain results that resemble the PAC ones. Indeed, learnability of a concept class still depends on the VC-dimension being polynomial Blumer *et al.*[20] This will provide a polynomial bound on the number of training vectors. As for the learning algorithm, instead of returning a hypothesis that has zero training error rate, it should find a concept that minimizes the training error.

Therefore, to show that $\mathscr{C}^d_{min}$ is learnable under this general learning model, the natural step is to devise minimum-error algorithm for the thresholded min–max functions. Unfortunately, it turns out that for the thresholded monotone minimum functions this problem is NP-complete: one can show that the Minimum Disjunctive Normal Form problem[21] is a restriction of minimum-error algorithm for $\mathscr{C}^d_{k\text{-}min\text{-}max}$.

In the rest of the paper, we shall describe a LMS (Least Mean Square) algorithm for training thresholded min–max functions. The algorithm is a gradient descent based approach and therefore not guaranteed to find the best classifier. Nonetheless experience shows that it performs quite well.

4. LMS ALGORITHM FOR TRAINING MIN–MAX CLASSIFIERS

We start describing the LMS algorithm[22] on a general function. If $\vec{x}(t)$ is the input vector to the classifier at discrete time $t = 1, 2, 3, \ldots$, then the output is $z(t) = F(\vec{x}(t); \vec{p}(t))$ where $\vec{p}(t)$ is a vector of parameters that together with $F(\cdot)$ determine the input–output rule of the classifier. One is usually interested in minimizing the mean-square error (MSE) of the output $z(n)$ and a desired process $d(n)$,

$$\mathscr{E}(t) = E[(z(t) - d(t))^2].$$

Since we are dealing with binary functions $z(t)$, $d(t) \in \{0, 1\}$, the mean square error is identical to the probability of error of the classifier. The mean-square error is minimized using a gradient descent approach. At each iterative step, the internal parameters $\vec{p}(t)$ are changed along the direction so as to produce the largest decrease in the MSE. This is achieved using the iterative equation

$$\vec{p}(t+1) = \vec{p}(t) - \mu \nabla_{\vec{p}} \mathscr{E}(t). \qquad (20)$$

The gradient symbol $\nabla_{\vec{p}} \mathscr{E}(t)$ is equal to $(\partial\mathscr{E}/\partial p_i, \ldots, \partial\mathscr{E}/\partial p_d)$ if $\vec{p} = (p_1, p_2, \ldots, p_d)$. The parameter μ is the convergence factor and is used to control the convergence rate of the system. The main simplification of the LMS approach is the replacement of the mean-square error by the instantaneous error $\varepsilon(n)$. Therefore Equation (20) is approximated by

$$\vec{p}(t+1) = \vec{p}(t) - 2\mu(z(t) - d(t))\mu\nabla_{\vec{p}} z(t) \qquad (21)$$

In this paper we shall be interested in using LMS for supervised training of binary valued functions (i.e., functions whose output values are 0 or 1). Typically a set of labeled examples are used to train the classifier. The members of this training set are formed into a sequence and presented to the LMS algorithm one by one. The step variable n is the sample number within the training data set. The desired process $d(n)$ is the actual label of the samples.

We shall state a few technical details before proceeding. In Section 3.5, we proved the learnability of the class of thresholded k-min–max functions. For the LMS training scheme, it is more convenient to restrict instead the total number of minima. Therefore in the rest of the paper we shall use the class of *thresholded* k-term-min–max functions. Another detail involves introducing the complemented variable $x' = 1 - x$. In the discussion that follows, the minimum functions are assumed to be *monotone*, i.e. the input variables are not complemented. This presents no difficulty because we can always remap a feature vector of dimension d into one of dimension $2d$ which are composed of pairs of uncomplemented and complemented variables. Therefore from now on we shall assume this remapping is performed and the input variables may be complemented.

Equation (21) requires the gradient of the function output with respect to the internal parameters, which immediately raises two questions:

(1) How do we define the parameters for the thresholded min–max function such that we can perform differentiation?

(2) Since the output of the min–max function involves nondifferentiable functions such as thresholding and minimum, how do we find (or approximate) the gradient of $z(n)$ with respect to $\vec{p}(n)$?

To answer the first question, we recall the functional form of a monotone thresholded k-term min–max function. If we denote the min-terms by h_j and the min–max function by y, then (see Fig. 2)

$$h_j = \bigwedge_{i\in I_j} x_i, \quad j = 1, 2, \ldots, k$$

$$y = \bigvee_{j=1}^{k} h_j$$

$$z = \begin{cases} 1 & y \geq \theta, \\ 0 & y < \theta. \end{cases}$$

The defining parameters include the threshold θ and the k coordinate sets I_j over which the minima are evaluated. Since the learning algorithm employs gradient of the internal states, the states should be continuous variables. This poses no problem for the threshold θ. As for the coordinate sets I_j, which are inherently discrete variables, we adopt the approach suggested in Salembier.[12] For each I_j we introduce d continuous real variables $m_{ij}, i = 1, \ldots, d$. They control the coordinate list in the following manner:

- x_i is included in I_j if $m_{ij} \geq 0$,
- x_i is excluded from I_j if $m_{ij} < 0$.

The parameters θ and m_{ij} are the only ones required to describe a thresholded monotone k-term min–max function. Hence, $\vec{p}(t) = (\theta(t), m_{11}(t), \ldots, m_{d1}(t), \ldots, m_{dk}(t))$. These parameters are also shown in Fig. 2. Turning to the second question,[12] provides *implicit* formulae for the minimum and maximum functions. Using these one can find a reasonable approximation to the gradients.

We are now in a position to find the gradients required in Equation (21). The derivative with respect to θ is easy. It follows from the definition of the threshold function

$$z = U_2(y - \theta) \tag{22}$$

where $U_2()$ is the unit step function with *two* output values defined as

$$U_2(x) = \begin{cases} 1 & \text{if } x \geq 0, \\ 0 & \text{if } x < 0. \end{cases}$$

A strict forward calculation yields the derivative,

$$\frac{\partial z}{\partial \theta} = -\delta(y - \theta). \tag{23}$$

For practical purposes the delta function in the above equation is approximated using a finite impulse shown in Fig. 3.

The parameter β controls the width of the pulse. Using this approximation, one derives the approximate derivatives

$$\frac{\partial z}{\partial \theta} = \begin{cases} -\dfrac{1}{2\beta} & \text{if } |y - \theta| \leq \beta, \\ 0 & \text{otherwise.} \end{cases} \tag{24}$$

To calculate the derivatives with respect to the para-

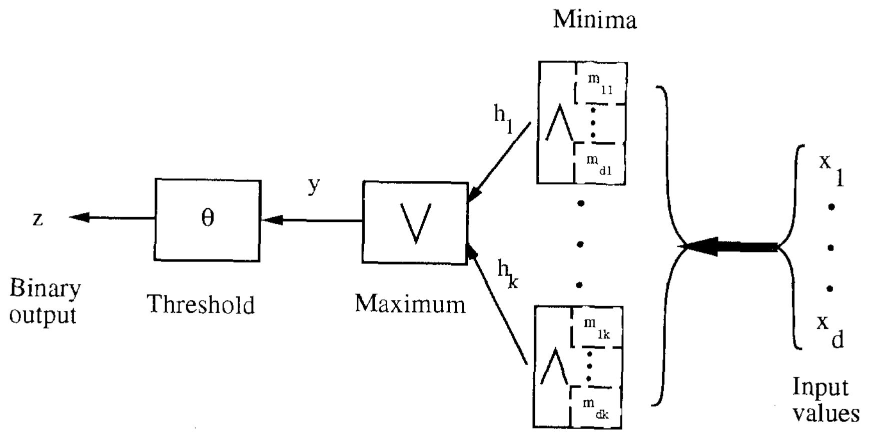


Fig. 2. The connections between the different modules within a thresholded min–max function.

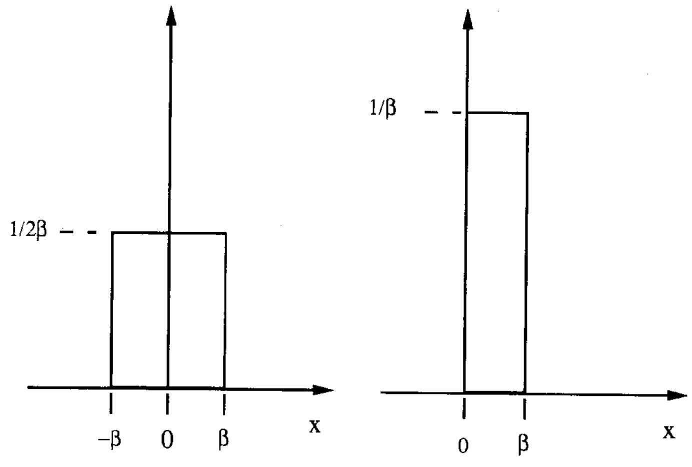


Fig. 3. Approximations to the $\delta(x)$. The one on the left is used in the update equation for θ while the other is used for the maximum and minimum functions. β is a parameter to be specified.

meters m_{ij}, one employs the chain rule:

$$\frac{\partial z}{\partial m_{ij}} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial h_j}\frac{\partial h_j}{\partial m_{ij}} \tag{25}$$

4.1. *Derivative of maximum*

The second term in equation (25) is the derivative of the output of a maximum function with respect to one of its inputs. It cannot be calculated using the explicit form of the maximum function. To overcome this difficulty, we used an *implicit* definition of the maximum function

$$G(y, h_1, \ldots, h_k) = \sum_{j=1}^{k} \{U_3(y - h_j) - 1\} + \frac{G_e}{2} = 0. \tag{26}$$

In this equation $U_3()$ denotes the unit step function with *three* output values:

$$U_3(x) = \begin{cases} 1 & \text{if } x > 0, \\ \frac{1}{2} & \text{if } x = 0, \\ 0 & \text{if } x < 0. \end{cases}$$

If y is equal to the maximum value of $h_1, \ldots, h_k$, the summation in (26) evaluates to $G_e/2$, where G_e is the number of inputs h_j that are equal to the output y. For mathematical tractability this parameter G_e is assumed to be constant and independent of y and h_j. To show (26) implies $y = \bigvee_{l=1}^{k} h_l$, we note that the summation term increases monotonically as y increases. So there is only one unique value of y that satisfies the equality, and that value is the maximum of the inputs.

To calculate the partial derivative $\partial y/\partial h_j$, we use the fact that the total derivative of $G()$ with respect to h_j is identically zero. Using the implicit functions theorem and after some simplifications, one obtains

$$\frac{\partial y}{\partial h_j} = -\frac{\dfrac{\partial G}{\partial h_j}}{\dfrac{\partial G}{\partial y}} \tag{27}$$

Using (26) the partial derivatives $\partial G/\partial h_j$ and $\partial G/\partial y$ can be calculated as follows:

$$\frac{\partial G}{\partial h_j} = -\delta(y - h_j) \tag{28}$$

Turning to the other derivative, we have

$$\frac{\partial G}{\partial y} = \sum_{l=1}^{k} \delta(y - h_l) \tag{29}$$

These two equations involve the delta function. For our practical purposes we shall approximate them using the finite pulse shown on the right of Fig. 3. Whereupon after combining equations (27), (28) and (29), we get the simple update equation

$$\frac{\partial y}{\partial h_j} \approx \begin{cases} \dfrac{1}{N_{max}} & \text{if } 0 \le y - h_j \le \beta \\ 0 & \text{otherwise.} \end{cases} \tag{30}$$

The parameter N_{max} is defined as

$$N_{max} \overset{\Delta}{=} \text{number of } h_j\text{'s such that } y - h_j \le \beta \tag{31}$$

$$= \sum_{j=1}^{k} U_2(\beta - (y - h_j)). \tag{32}$$

To conclude this section, we remark that the implicit formulation of maximum was also used in Salembier(12) for adaptive morphological filtering. There are some differences between the formulation used in the reference from ours. First of all, we did not assume that $G_e = 1$, which is the case in.(12) Another difference regards equation (30) (the update equation): in,(12) N_{max} is taken to be 1 (a direct result of assuming $G_e = 1$); while in our case $N_{max} \ge 1$. A direct consequence is generally slower convergence for our algorithm compared to.(12) We also observed about a 5% decrease in error rates by using our updated training equations.

4.2. *Derivative of minimum*

The third derivative in equation (25) now poses no special difficulties. The output of the j-th minimum function is

$$h_j = \min_{i, m_{ij} \geq 0} \{x_i\}$$

Using the implicit formulation again, this equation can be reexpressed as

$$L(h_j, x_1, \ldots, x_d, m_{1j}, \ldots, m_{dj}) = \sum_{i=1}^{d} \{U_2(m_{ij})[U_3(x_i - h_j) - 1]\} + \frac{L_e}{2} = 0, \quad (33)$$

where L_e denotes the number of inputs x_i that are equal to the output value h_j and is assumed to be constant. Again the update equation is independent of this constant. Using the total derivative of L with respect to m_{ij} we immediately find the gradient of the output h_j with respect to m_{ij}

$$\frac{\partial h_j}{\partial m_{ij}} = -\frac{\dfrac{\partial L}{\partial m_{ij}}}{\dfrac{\partial L}{\partial h_j}} = \frac{\delta(m_{ij})[U_3(x_i - h_j) - 1]}{\sum_{l=1}^{d} U_2(m_{lj})\delta(x_l - h_j)} \approx \begin{cases} -\dfrac{1}{2N_{min}} & \text{if } |m_{ij}| \leq \beta \text{ and } h_j = x_i \\ 0 & \text{otherwise.} \end{cases} \quad (34)$$

and

$$N_{min} \overset{\Delta}{=} \text{number of inputs such that } m_{lj} \geq 0 \text{ and } |x_l - h_j| \leq \beta = \sum_{l=1}^{d} U_2(m_{lj}) U_2(\beta - |x_l - h_j|)$$

We shall denote the training parameters for the minimum modules by μ_{min} [convergence rate in equation (21)] and β_{min} [approximate delta function width in equation (34).]*

5. APPLICATION TO HANDWRITTEN CHARACTER RECOGNITION

In this section, we shall describe the experimentation we did on handwritten character recognition using the LMS algorithm described in Section 4.

As the outputs of thresholded min–max functions are binary valued, we applied them to the task of distinguishing between two types of handwritten digits.

Our database consists of segmented and cleaned handwritten digits.† Samples of digits we used are shown in Fig. 4. The size of each digit is around 50 × 50 pixels. Three different sets of data were used in our experiments. In increasing order of difficulty: the first set consisted of 0's and 1's, the second set 0's and 6's, and the third 6's and 8's. The training set consists of 600 samples of each type giving a total of 1200 digits, while the test set consists of 200 samples of each for a total of 400 test digits.

5.1. *Feature extraction*

The two types of features that we used are the morphological shape-size histograms and Fourier descriptors, described respectively in sections 5.1.1 and 5.1.2. Fourier descriptors have been used in character recognition before [for example, see[23,24]]. Ideas similar to the shape-size histogram was employed in Trenkle *et al.*[25] The feature they employed is based on size distribution of the x or y border signals of the character. However, our feature is based on multiscale smoothing of the two dimensional image.

5.1.1. *Morphological shape-size histogram.* A central theme in object recognition is the problem of multiscale shape representation. In morphological image analysis, an important tool for multiscale shape representation is the shape-size histogram[10,26] [also called "pattern spectrum"[26]]. The direct application of this shape descriptor to character images is not, however, desirable: experience shows that the shape-size histogram is sensitive to the thickness of the stroke, the actual size of the image, and also to rotation. Therefore, the normalization procedure described in references (27) and (28) was employed to reduce the fluctuation in feature values due to these factors. It includes the following steps:

(1) The character image is dilated‡ by a 3 × 3 pixel square to fill up small gaps. The image is then thinned using a procedure described in Bloomberg.[29] The basic *thinning* operation in the algorithm is

$$I \mapsto I \setminus \bigcup_i ((I \ominus A_i) \cap (I^c \ominus B_i)).$$

Basically, the set difference (\) removes the boundary of the character while still preserving four connectivity of the foreground. For details about the structuring elements $S_i = (A_i, B_i)$ we employed.[27] In each step of the thinning algorithm, the image is thinned using S_i and their rotated versions. This is repeated under convergence (i.e. no change in the image).

* Salembier[12] assumed $L_e = 1$ and hence also $N_{min} = 1$, similar to the situation for maximum. Since $N_{min} \geq 1$, our update equation will generally converge slower. We also found that our equations produced about a 5% decrease in error rates.

† This database was supplied by Dan Bloomberg of Xerox PARC.

‡ Representing the foreground of a binary image by a set I (and its background by I^c), then three basic morphological operators are:

erosion: $I \ominus B = \{z : z + b \in I \text{ for all } b \in B\}$,

dilation: $I \oplus B = \{x + b : x \in I \text{ and } b \in B\}$,

closing: $I \bullet B = (I \oplus B) \ominus B$.

The set B in the above expressions is called the *structuring element*.

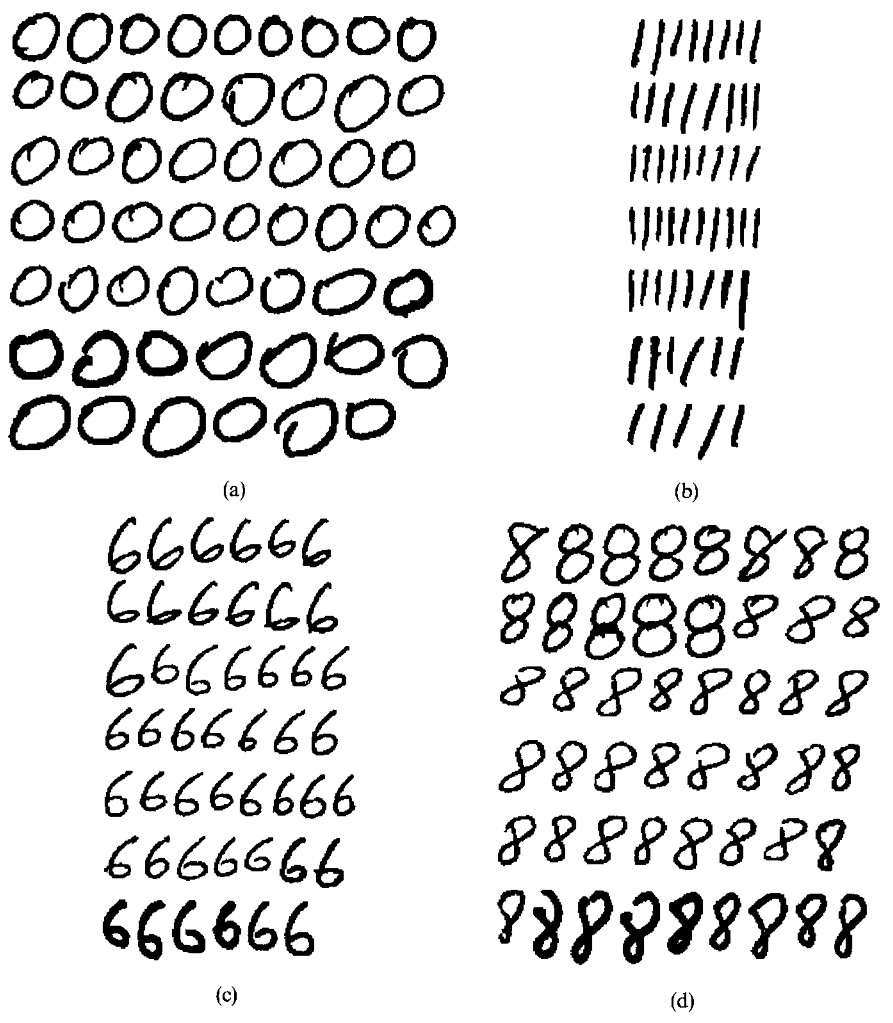


Fig. 4. Sample data from the handwritten database. (a) A collection of 0's. (b) A collection of 1's. (c) A collection of 6's. (d) A collection of 8's.

(2) Next we calculate the area functions $A[I \bullet nB]$ for $n = 0, \ldots, N_b$, where

$$nB = \underbrace{B \oplus \cdots \oplus B}_{n} \text{ for } n \geq 1; \quad 0B = \{(0,0)\}$$

are multiscale dilations of a unit-size convex set B. For discretized images, the area count is equated to the number of pixels. The maximum size N_b is determined by the relative size of the image with respect to the structuring element B: suppose the dimension of the bounding box of the image I is $i_w \times i_h$, and that for the structuring element $b_w \times b_h$; then

$$N_b = \max(i_w - b_w, i_h - b_h).$$

In order to fit this information into a feature vector of size M, we *resampled* the area function:

$$A'(m) = A\left(I \bullet \left\lfloor \frac{mN_b}{M} \right\rfloor B\right), \quad \text{for } m = 0, \ldots, M \quad (35)$$

where the floor function ($\lfloor \bullet \rfloor$) equals the largest integer $\leq$ its argument.

(3) Finally, the normalized shape-size histogram is calculated via the following equations:

$$NSH'_I(m, B) = \frac{1}{\mathcal{M}}(A'(m+1) - A'(m)),$$

$$\text{for } m = 0, \ldots, M - 1 \quad (36)$$

The quantity $\mathcal{M}$ is the area of the maximally closed image.

Examples of normalized shape-size histograms are shown in Fig. 5.

If insensitivity to rotation is also desired, one can use the *radial size histogram* (called "Oriented Pattern Spectrum").[26] It replaces the closing by a 2-D element B with an intersection of closings by the four directed vectors $\{\rightarrow, \nearrow, \uparrow, \nwarrow\}$. In finding the parameter N_b (the maximum closing size), the bounding box is allowed a 45° shear in either directions. This feature was also employed in our experiments. Examples of this feature are also shown in Fig. 5.

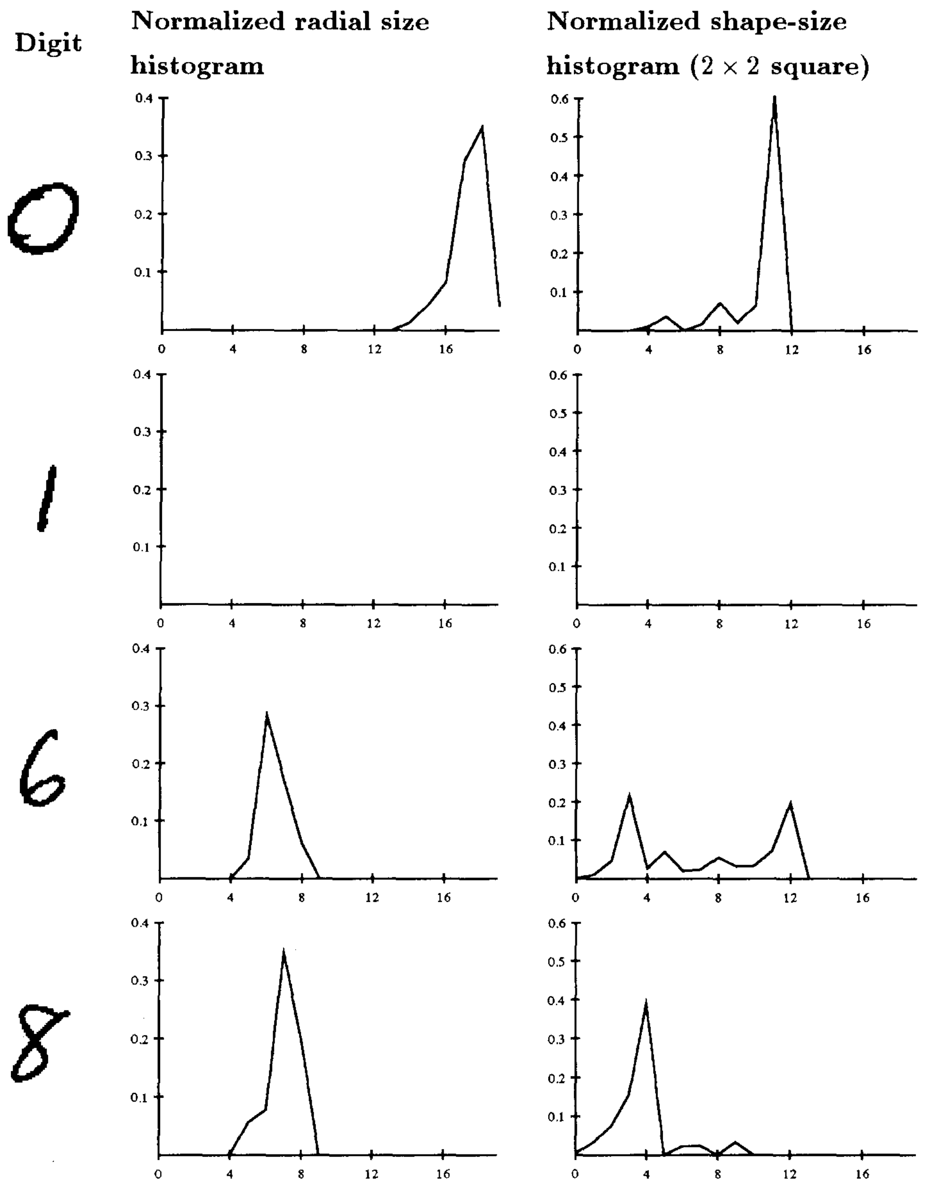


Fig. 5. Samples of handwritten digits together with morphological shape-size histograms features extracted from them. The center column are normalized radial size histograms, while the rightmost column are normalized shape-size histograms calculated using a 2 × 2 square.

5.1.2. *Fourier descriptors.* We employed the Fourier descriptor discussed in reference (30). It provides a representation for the outer boundary of the character image. This shape descriptor is independent of translation and rotation of the curve. To calculate the Fourier descriptor we first preprocessed the image using the same procedure (gap filling and thinning) as in Section 5.1.1. Then the boundary of the discretized image is traced using a standard eight-connectivity chain code algorithm.(31) The output of this algorithm describes an m-sided polygon with sides of length 1 or $\sqrt{2}$, from which the following Fourier coefficients are calculated:

$$|c_n| = \frac{1}{2\pi n}\left|\sum_{j=1}^{m} \Delta\phi_j \exp\left(i\frac{2\pi n l_j}{L}\right)\right|, \quad \text{for } n = 1, 2, \ldots \tag{37}$$

$$l_j = \sum_{k=1}^{j} \Delta l_k \tag{38}$$

In the above equations $\Delta\phi_j$ denotes the amount of change in direction at vertex j as we transverse the chain code; and Δl_k denotes the length of the edge between vertices $k-1$ and k (vertex m is the same as vertex 0). The total length of the polygon is L. We remark that the amplitude $|c_n|$ is also independent of the initial vertex of the polygon.

Since the input to the threshold min–max functions has to be restricted to a range of [0, 1], the coefficients $|c_n|$ are divided by the factor $(\sum_{j=1}^{m}|\Delta\phi_j|/2\pi)$. It is easy to see that the normalized coefficients $FD(n)$ are within the desired range:

$$0 \le FD(n) = \frac{\left|\sum_{j=1}^{m}\Delta\phi_j \exp\left(-i\frac{2\pi n l_j}{L}\right)\right|}{n\sum_{j=1}^{m}|\Delta\phi_j|} \le 1.$$

Examples of the Fourier descriptors are shown in Fig. 6.

5.2. *Experimental results*

In this section we shall provide the results of applying the LMS training algorithm described in Section 4.

In our experiments, we used the LMS algorithm in a "per-sample" mode operation. This means that training and testing are separated into two distinct stages. During each iteration of training stage, the defining parameters in the min–max function are updated with the LMS rule after presentation of each training vector. After the entire training data set is stepped through, the error rate of the current classifier is calculated. In our particular implementation, the training program performs a fixed number of scans through the training

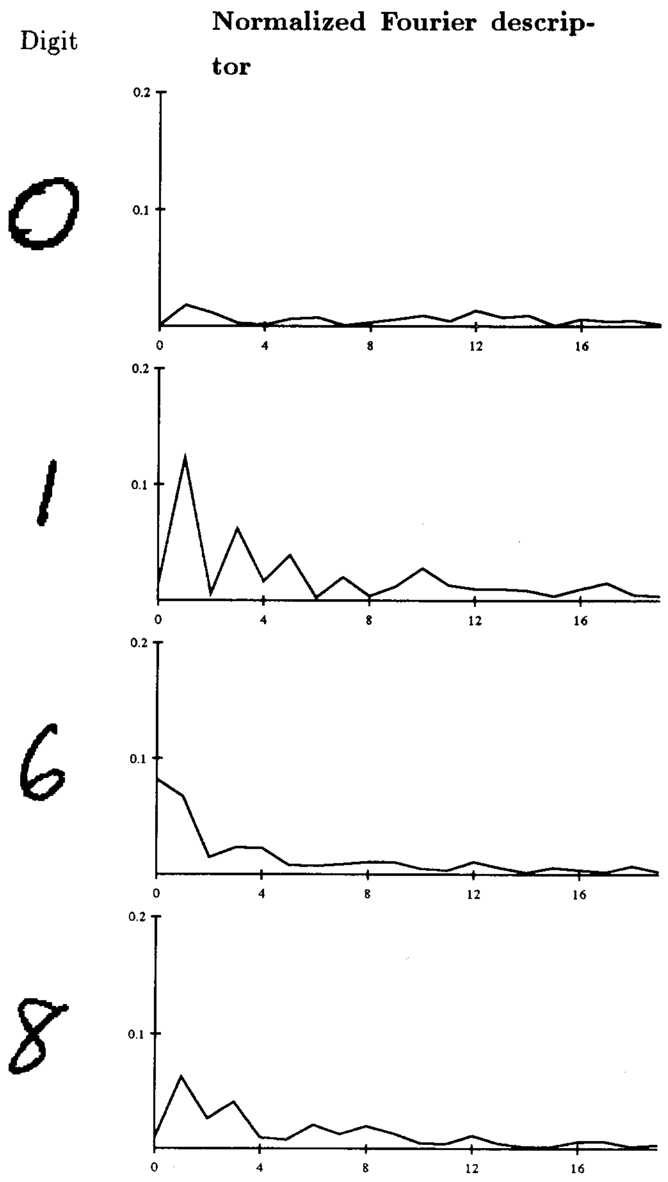


Fig. 6. Samples of handwritten digits together with Fourier descriptors calculated using the outermost boundary found by an eight connectivity chain code. See Section 5 for more details of these features.

set (200 is the value we used), and returns the classifier that achieved the lowest error rate within the 200 scans. This classifier is then used to calculate the error rate on the test rate. For each training parameter setting, the training program was run on 15 different initial settings of m_{ij} which are generated randomly. The error rates reported are averaged over settings that achieved the five lowest test error figures. The m_{ij} parameters for the minimum gates were initialized using a random number generator. We found that convergence depended quite critically on the initial value of the threshold θ and therefore it is not randomly generated. It is initially set to some value close to 0.5, since the means of the data are shifted to 0.5.

In the first set of experiments, we compared the performance of the min–max classifiers to feed-forward preceptron-based neural networks trained with back propagation.(32) Three different sets of data were used. In increasing order of difficulty: the first set consisted of 0's and 1's, the second set 0's and 6's, and the third 6's and 8's. For each digit we extracted 20 components of shape-size histogram and 20 components of normalized Fourier descriptors forming feature vectors of 40 components. Two types of shape-size histograms were used in the experiments. The first one is the normalized radial size histograms. The other kind is $NSH'_I(m, B)$ (see Section 5.1.1) with the structuring element B a 2×2 square. After the features were extracted we shifted the mean for each component to 0.5. This preprocessing step was needed because the values of the features tend to have small values < 0.5 and the complemented variables will not be used otherwise. The shifted values were then rescaled in order to preserve the range $[0, 1]$. Typical training parameter setting is $\beta_\theta = 0.1$, $\mu_\theta = 10^{-5}$, $\beta_{max} = 10^{-3}$, $\mu_{min} = 10^{-6}$ and $\beta_{max} = 10^{-4}$.

Turning to the neural networks; only two layer nets were used in order to provide fair comparison with the min–max classifiers (the maximum and minimum gates correspond to different layers in a neural network). There was one node in the output layer—an output value of ≥ 0.9 is treated as a positive sample while those ≤ 0.1 negatives.(32) Again we used fifteen initial conditions for the back propagation algorithm, and reported averages of errors from the five best runs. A "per-sample" update scheme was used (the weights and thresholds are updated after presentation of each training vector). Finally, a momentum term was included in the update equation. Typical training parameters are μ (convergence rate) $= 0.7$ and momentum 0.2. Section 5.2.1 contains the results for the first set of experiments.

In the second set of experiments, we investigated the importance of each type of feature for good classification. Three different sets of feature vectors were extracted from the database consisting of 0's and 6's. They are

(1) 20 components of normalized radial size histogram plus 20 components of normalized Fourier descriptors. (Same as previous experiments.)

(2) 40 components of normalized radial size histogram.

(3) 40 components of normalized Fourier descriptors.

These features were also preprocessed (shifted and scaled) as described previously. The results are detailed in Section 5.2.2.

5.2.1. *Comparison of min–max classifier with neural networks*. The results for 0's and 1's are shown in Table 1. The results generated using normalized radial size histogram with Fourier descriptors are shown in the top two tables. Typical error rates for the min–max classifiers are 0.083% on the training data and 0.25% on the test data, which amounts to 1 error in the training data set and 1 in the test set. The neural networks were able to achieve no error on the test data set. These results are very encouranging, the min–max classifiers achieved error rates which are essentially the same as that generated by the neural networks. The error rates deteriorated if the radial histograms are

Table 1. Results for 0–1 classification problem employing *both* shape-size histograms and Fourier descriptors

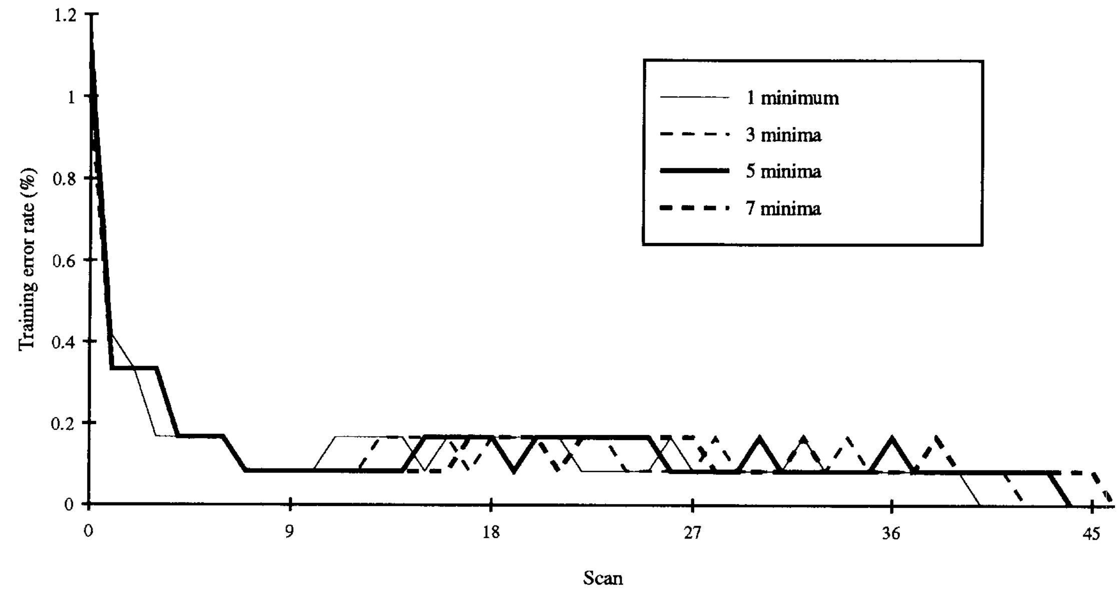
Distinguishing 0's and 1's
Normalized radial size histograms and Fourier descriptors

No. of minima	Min–max % error (train)	% error (test)	Network	Neural network % error (train)	% error (test)
1	0.083	0.25	1, 1	0.083	0
3	0.083	0.25	3, 1	0.083	0
5	0.1	0.25	5, 1	0.083	0
7	0.083	0.25	7, 1	0.083	0
	Normalized shape-size histograms with 2×2 square and Fourier descriptors				
1	3.867	2.6	1, 1	0.633	1.2
3	1.9	2.8	3, 1	0.633	0.85
5	1.083	3	5, 1	0.567	0.8
7	1.733	3	7, 1	0.533	0.55

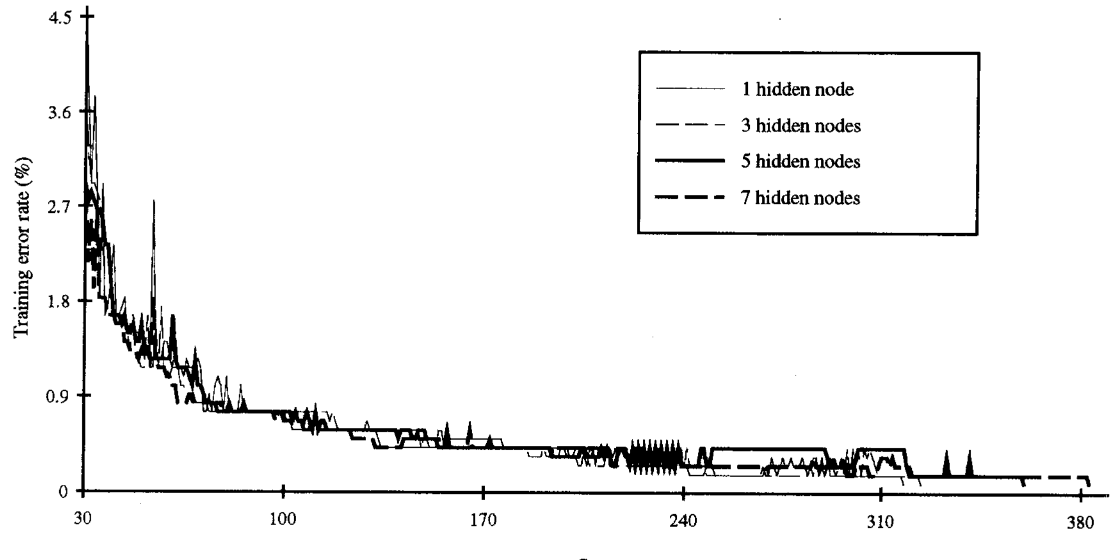
The top two tables are generated using normalized radial histograms and Fourier descriptors, while the lower two using normalized shape-size histogram with 2×2 square and Fourier descriptors.

replaced by the 2 × 2 square histograms. This is because the 2 × 2 square histogram is detecting the notch in the digits written like **1**.

Besides comparing the error rates, another attribute that is of interest is the speed of convergence of the training algorithms. Typical convergence plots of the LMS algorithm and the back propagation algorithm on data set (I) are shown in Fig. 7. The graphs show the error rate on the training data set as a function of the number of scans. (The error rate is computed after each scan of the entire training sequence.) The plots terminate after the minimum error rate is found. Judg-



(a)



(b)

Fig. 7. Comparison of the convergence rate of min–max classifiers trained via LMS algorithm versus neural networks trained via Back. Propagation algorithm. The feature vectors are normalized radial size histograms with Fourier descriptors, extracted from the training data set of 0's and 1's. The curves are terminated after the minimum error rate are achieved. (a) Typical convergence plots for the LMS algorithm. The minimum error rate (0%) is attained within 40 scans. (b) Convergence plots for the Back Propagation on the same data set. After 40 scans the typical error rate was 2%. The minimum error rate (0.083%) was achieved around 400 scans.

ing from these plots, we see that a clear advantage of the LMS algorithm over the back propagation algorithm is the speed of convergence. Typically only 40 scans is required befoɛe the minimum training error is reached whereas for the Back Propagation ten times more scans are required. Another advantage we found in running the programs is that the min–max classifier required less execution time: computing a min–max function requires only binary comparisons; while neural networks require the calculating the computational costly multiplications and the exponential functions.

The results for the second set of experiments (0's and 6's) are shown in Table 2. For the min–max classifiers the test error rates were around 1%, smaller by 0.5% of that attained by neural networks. In Table 2(b), the normalized radial size histograms were replaced by 2 × 2 square histograms. The error rates have increased by using the latter set of features. Notice also that the min–max classifiers have lower training and test error rates than the neural networks.

Finally, the results for the third set of experiments (6's and 8's) are shown in Table 3. In the top two tables the features are normalized radial size histograms with Fourier descriptors. The best test error rates for min–max classifiers are around 16%, smaller by 4% of the best value attained by neural networks. In Table 3(b) the radial histograms were replaced with 2 × 2 square histograms, resulting in about 50% drop in both training and test error rates.

5.2.2. *Comparison of different features.* In this set of experiments we compared the error rates obtained

Table 2. Results for 0–6 classification problem employing *both* shape-size histograms and Fourier descriptors

Distinguishing 0's and 6's					
Normalized radial size histograms and Fourier descriptors					
No. of minima	Min–max % error (train)	% error (test)	Network	% error (train)	Neural network % error (test)
1	1.783	1.35	1, 1	1.517	1.5
3	2.283	1.15	3, 1	1.567	1.5
5	2.25	1	5, 1	1.517	1.5
7	2.267	0.85	7, 1	1.517	1.5
Normalized shape-size histograms with 2 × 2 square and Fourier descriptors					
1	3.35	2.7	1, 1	4.6	2.65
3	3.35	1.65	3, 1	4.9	2.65
5	3.233	1.65	5, 1	4.917	2.35
7	3.083	1.9	7, 1	4.65	2.1

Table 3. Results for 6–8 classification problem employing *both* shape-size histograms and Fourier descriptors

Distinguishing 6's and 8's					
Normalized radial size histograms and Fourier descriptors					
No. of minima	Min–max % error (train)	% error (test)	Network	% error (train)	Neural network % error (test)
1	16.767	17.7	1, 1	49.867	49.55
3	14.283	15.4	3, 1	22.517	20.5
5	15.35	15.5	5, 1	18.4	18.15
7	15.35	16.85	7, 1	19.55	18.2
Normalized shape-size histograms with 2 × 2 square and Fourier descriptors					
1	8.95	11.75	1, 1	19.95	17.85
3	8.55	11.6	3, 1	15.567	13.9
5	10.233	11.55	5, 1	15.217	14.25
7	11.35	11.85	7, 1	15.25	13.55

Table 4. Results for 0–6 classification problem using *only* normalized radial size histograms

Distinguishing 0's and 6's (only radial size histogram)					
No. of minima	Min–max % error (train)	% error (test)	Network	% error (train)	Neural network % error (test)
1	2.283	1	1, 1	2.167	1.65
3	2.1	2.1	3, 1	2.05	1.3
5	1.933	2	5, 1	1.967	1.2
7	2.067	2	7, 1	1.967	1.15

Table 5. Results for 0–6 classification problem using *only* normalized Fourier descriptors

Distinguishing 0's and 6's (only Fourier descriptors)					
	Min–max				Neural network
No. of minima	% error (train)	% error (test)	Network	% error (train)	% error (test)
1	4.75	4.85	1, 1	4.117	3.05
3	3.2	2.35	3, 1	3.567	3
5	3.133	2.25	5, 1	3.333	2.4
7	3.133	2.1	7, 1	3.267	2.45

using different sets of features. Table 4 shows the results for using only normalized radial size histograms while Table 5 with only Fourier descriptors. In both cases the test error rates for the min–max classifiers showed approximately a factor of 2 increase in the error rate for the min–max classifiers for using only one type of feature versus the using the mixed set. The error rates for using only shape-size histograms were about the same as using just Fourier descriptors. Increases in the error rates for the neural networks were also observed. Therefore it is beneficial to include both types of features instead of using just one.

6. CONCLUSION

In conclusion, we have introduced the class of min–max classifiers which are lattice-theoretic generalizations of the Boolean functions. The main theoretical results we proved was the learnability of three subclasses of these functions under the PAC model of machine learning. This was achieved by demonstrating both polynomial time learning algorithms and polynomial bound on the number of training samples required. For the practical training of min–max classifiers we introduced an LMS algorithm. This training algorithm was then applied to the problem of handwritten character recognition; which demonstrated very good performance in our experiments. Comparison with feed forward neural networks trained with back propagation highlighted the advantages of min–max classifiers trained via the LMS algorithm; which include the simplicity of the min–max classifiers, and the faster speed of convergence of the LMS training algorithm, while achieving similar (or a few times) smaller error rates. Both the LMS and back propagation algorithms are used in a "per-sample" update mode so that the comparison is made on common grounds. While it is possible to speed up the back propagation by various techniques (see, for example, Xu *et al.*,[33]) most of them do rely on improving the basic gradient descent scheme. The same techniques can therefore also be applied to speed up the basic LMS algorithm presented in this paper.

The experimental results on handwritten recognition provides evidence on the practical utility of the min–max classifier. Future work with min–max classifiers is their further development for practical use. First of all, practical character recognition systems employ rejection schemes to reduce error rate by rejecting uncertain input patterns. One way to incorporate rejection into the basic thresholded min–max function is to use two different thresholds θ_{upper} and θ_{lower}; an input vector is classified as a positive instance if the min–max function value ($f(\vec{x})$) is higher than θ_{upper} negative if lower than θ_{lower}. Otherwise it is rejected. These two values can be selected to be some value away from the trained threshold θ so that a desired error rate is achieved, i.e. set $\theta_{\text{upper}} = \theta + \Delta$ and $\theta_{\text{lower}} = \theta - \Delta$; choose Δ so that a desired error rate is achieved.

The next natural step for the development of a full fledged character recognition system is to generalize the min–max classifiers for arbitrary number of classes. Work in this direction is reported in Yang,[28] where a multiclass min–max classifier is constructed by associating with each class in the problem the following min–max function:

$$F(\vec{x}) = \bigvee_{j} \bigwedge_{i \in I_j} l_i + w_i, \quad l_i \in \{x_i, -x_i\}.$$

An input feature vector is classified according to which min–max function has the highest output. The additive weights (w_i) and the dependent parameter list (I_j) are trained using an LMS algorithm similar to the one described in Section 4. This architecture resembles a feedforward neural network for multiclass classification problems, in which one output node is associated with each output class and classification is done by comparing the output at these nodes. Moreover,with this "winner-take-all" scheme, it is more natural to incorporate a rejection criterion in the multiclass min–max classifier—the output of the min–max functions can be interpreted as the probability that the input belongs to the corresponding classes; if the top score is not significantly larger than the rest then the input data is rejected. We note that this is the usual procedure used in neural networks for rejecting input data. An LMS algorithm was also derived or training the multiclass min–max classifier. This was applied to the classification of handwritten digits (all 10 classes). The multiclass min–max classifier was found to be able to achieve error rates comparable to traditional feedforward neural networks.

REFERENCES

1. L. G. Valiant, A theory of the learnable, *Commun. ACM* **27**, 1134–1142 (1984).

2. L. G. Valiant, Learning disjunction of conjunctions, in *Proceedings of 9th International Joint Conference on Artificial Intelligence* 560–566 Los Angeles, CA (1985).
3. A Blumer, A. Ehrenfeucht, D. Haussler and M. K. Warmuth, Occam's Razor, *Inform. Process. Lett.* **24**, 377–380 (1987).
4. A. Blumer, A. Ehrenfeucht, D. Haussler and M K. Warmuth, Learnability and Vapnik–Chervonenkis dimension, *J. ACM* **36**(4), 929–965 (1989).
5. D. Haussler, Probably approximate correct learning, Tech. Rep. UCSC-CRL-90-16, University of California, Santa Cruz (1990).
6. M. Kearns, M. Li, L. Pitt and L. G. Valiant, On the learnability of Boolean formulae, in *Proceedings of the 19th Annual Symposium on Theory of Computing*, 285–295, The Association of Computing Machinery, New York (1987).
7. M. Kearns, *The Computational Complexity of Machine Learning.* MIT Press, Cambridge, Massachusetts (1990).
8. R. L. Rivest, Lecture notes in machine learning. Massachusetts Institute of Technology (1990).
9. R. E. Schapire, The design and analysis of efficient learning algorithms, Tech. Rep. CICS-TH-273, Brown- Harvard-MIT Center for Intelligence Control Systems, (1991).
10. J. Serra, *Image Analysis and Mathematical Morphology.* Academic Press, New York (1982).
11. P. Maragos and R. W. Schafer, Morphological systems for multidimensional signal processing, *Proc. IEEE* **78**(4), 690–710 (1990).
12. P. Salembier, Structuring element adaptation for morphological filters, *J. Visual Commun. Image Represent.* **3**, 115–136 (1992).
13. E. N. Gilbert, Lattice-theoretic properties of frontal switching functions, *J. Math. Phys.* **33**, 57–67 (1954).
14. P. Maragos and R. W. Schafer, Morphological filters (parts I and II), *IEEE Trans. Acoustics, Speech Signal Process.* **ASSP-35**, 1153–1184 (1987), *ibid.* **ASSP-37**, 597 (1989).
15. L. Breiman, J. H. Friedman, R. A. Olsen and C. J. Stone, *Classification and Regression Trees.* Wadsworth and Brooks, Pacific Grove, California (1984).
16. F. Palmieri, A backpropagation algorithm for multilayer hybrid order statistic filters, in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, 1179–1182, Glasgow, U.K. (1989).
17. F. Palmieri, Adaptive recursive order statistic filters, in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, 1229–1232, Albuquerque, New Mexico.
18. S. S. Wilson, Morphological networks, in *SPIE Vol.* 1199 *Visual Communication and Image Processing IV* 483–493 (1989).
19. P.-F. Yang and P. Maragos, Learnability of min–max pattern classifiers, in *SPIE Vol.* 1606 *Visual Communications and Image Processing* '91: *Image Processing*, 294–308 (1991).
20. A. Blumer, A. Ehrenfeucht, D. Haussler and M. Warmuth, Classifying learnable geometric concepts with the Vapnik-Chervonenkis dimension, in *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, 273–282, The Association of Computing Machinery, Berkeley, CA (1986).
21. M. R. Garey and D. S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness.* W. H. Freeman, San Francisco, California, (1979).
22. B. Widrow and S. D. Stearns, *Adaptive Signal Processing.* Prentice-Hall, Englewood Cliffs, New Jersey (1985).
23. C. C. Tappert, C. Y. Suen and T. Wakahara, The state of the art in one-line handwriting recognition, *IEEE Trans. Pattern Analy. Mach. Intell.* **12**, 787–808 (1990).
24. E. Persoon and K. S. Fu, Shape discrimination using Fourier descriptors, *IEEE Trans. Systems, Man Cybern.* **SMC-7**, 170–179 (1977).
25. J. M. Trenkle, S. G. Schlosser and R. C. Vogt, Morphological feature set optimization using genetic algorithm, in *SPIE Vol.* 1568 *Image Algebra and Morphological Image Processing II*, 212–223 San Diego, California (1991).
26. P. Maragos, Pattern spectrum and multiscale shape representation, *IEEE Trans. Pattern Analy. Mach. Intell.* **11**(7), 701–716 (1989).
27. P.-F. Yang and P. Maragos, Morphological systems for character image processing and recognition, in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, 97–100 Minneapolis, Minnesota, (1993).
28. P.-F. Yang, *Morphological Systems for Character Image Processing and Recognition.* PhD thesis, Harvard University (1993).
29. D. S. Bloomberg, Connectivity-preserving morphological image transformations, in *SPIE Vol.* 1606 *Visual Communications and Image Processing*'91: *Image Processing* (1991).
30. C. T. Zahn and R. Z. Roskies, Fourier descriptors for plane closed curves, *IEEE Comput.* **C-21**, 269–281 (1972).
31. T. Pavlidis, *Algorithms for Graphics and Image Processing.* Computer Science Press, Rockville, Maryland (1982).
32. D. E. Rumelhart, J. L. McClelland and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* MIT Press, Cambridge, Massachussets (1986).
33. L. Xu, S. Klasa and A. Yuille, Recent advances on techniques of static feedforward networks with supervised learning, Tech. Rep. 92-2, Harvard Robotics Laboratory (1992).

About the Author—PING-FAI YANG received a B.S. (summa cum laude) in electrical engineering from Rensselaer Polytechnic Institute, Troy, New York in 1988, and the S.M. and Ph.D. degrees from Harvard University, Cambridge, Massachusetts in 1989 and 1993, respectively. Currently he is with the Signal Processing Research Department at AT & T Bell Laboratories. His interests include pattern recognition, image analysis/processing, statistics and information theory.

About the Author—PETROS MARAGOS received the Diploma degree in electrical engineering from the National Technical University of Athens, Greece, in 1980, and the M.S.E.E. and Ph.D. degrees from the Georgia Institute of Technology, Atlanta, in 1982 and 1985. In 1985 he joined the faculty of the Division of Applied Sciences at Harvard University, where he worked as Assistant (1985–1989) and Associate Professor (1989–1993) of Electrical Engineering. In 1993 he joined the faculty at Georgia Institute of Technology where he is currently an Associate Professor of Electrical and Computer Engineering. His research activities have been in the general areas of signal processing, systems theory, communications, pattern recognition, and their applications to image processing and computer vision, and computer speech processing and recognition. He has been involved in several professional society activities, including: Associate Editor and Guest-Editor for two IEEE Transactions (on Signal and Image Processing); General Chairman for the 1992 SPIE Conference on Visual Communications and Image Processing, Boston; Member of the IEEE DSP and IMDSP Technical Committees; Vice-President of the International Society of Mathematical Morphology. In 1987 Dr Maragos received a National Science Foundation Presidential Young Investigator Award for his work in signal and image processing. He also received the 1988 IEEE Acoustics, Speech, and Signal Processing Society's Paper Award for his paper "Morphological Filters", and the 1994 IEEE Signal Processing Society's Senior Award for the paper "Energy Separation in Signal Modulations with Application to Speech Analysis".