CHARACTER RECOGNITION USING MIN-MAX CLASSIFIERS DESIGNED VIA AN LMS ALGORITHM

Ping-Fai Yang and Petros Maragos

Division of Applied Sciences, Harvard University, Cambridge, MA 02138, USA

Abstract

In this paper we propose a *Least Mean Square* (LMS) algorithm for the practical training of the class of *min-max classifiers*. These are lattice-theoretic generalization of Boolean functions and are also related to feed-forward neural networks and morphological signal operators.

We applied the LMS algorithm to the problem of handwritten character recognition. The database consists of segmented and cleaned digits. Features that were extracted from the digits include Fourier descriptors and morphological shape-size histograms.

Experimental results using the LMS algorithm for handwritten character recognition are promising. In our initial experimentation, we applied the min-max classifier to binary classification of "0" and "1" digits. By preprocessing the feature vectors, we were able to achieve an error rate of 1.75% for a training set of size 1200 (600 of each digit); and an error rate of 4.5% on a test set of size 400 (200 of each). These figures are comparable to those obtained by 2-layer neural nets trained using back propagation. The major advantage of min-max classifiers compared to neural networks is their simplicity and the faster convergence of their training algorithm.

1 Introduction

Character recognition is a very important task in electronic document image analysis. In a character recognition system, two distinguished stages are usually recognized. The first one extracts features from the input character image which are fed into the second classifier stage. In this paper we shall describe the application of morphological systems to both of these operations.

In [1] we reported theoretical results on the automatic design of several subclasses of min-max pattern classifiers. There we provided polynomial time efficient search algorithms under the *Probably Approximately Correct* (PAC) model of machine learning [2]. These algorithms work, however, under the provision that a consistent classifier exists (i.e., there exists a min-max classifier that can correctly classify all the training data). Such assumption is rarely fulfilled in practical applications, which led us to investigate the possibility of obtaining time efficient training algorithms in the absence of the PAC condition. Unfortunately, we found that by loosening the PAC condition, the problem becomes difficult. More specifically, we found that the problem of designing a thresholded monotone minimum function which minimizes the error rate for an arbitrary training data set is NP-complete. Since the thresholded monotone minimum function is the simplest form of min-max classifier, we suspect the general question is also hard.

This discovery led us to investigate other approaches for designing the learning algorithm. In this paper, we report a *Least Mean Square* (LMS) type algorithm [3] and apply it to the task of handwritten character recognition. The LMS algorithm is a gradient based method and requires the calculation of derivatives of the classifier output with respect to some internal parameters. For the min-max classifiers, it is not immediately obvious how to define the internal parameters, and to write down the input-output relation in an easily differentiable form. These two problems are addressed in [4], in which the author dealt with the problem of designing morphological filters with flat structuring elements. To calculate the derivatives, he employed an *implicit* formulation of the maximum and minimum functions. Due to the close relationship between min-max classifiers and morphological filters, in our work we have adapted its formulation to the design of the min-max classifiers.

Turning to feature extraction, we note that morphological features have been used in character recognition before [5]. For our work we employed the morphological shape-size histogram (called "Pattern

 0-8194-1018-7/92/$4.00

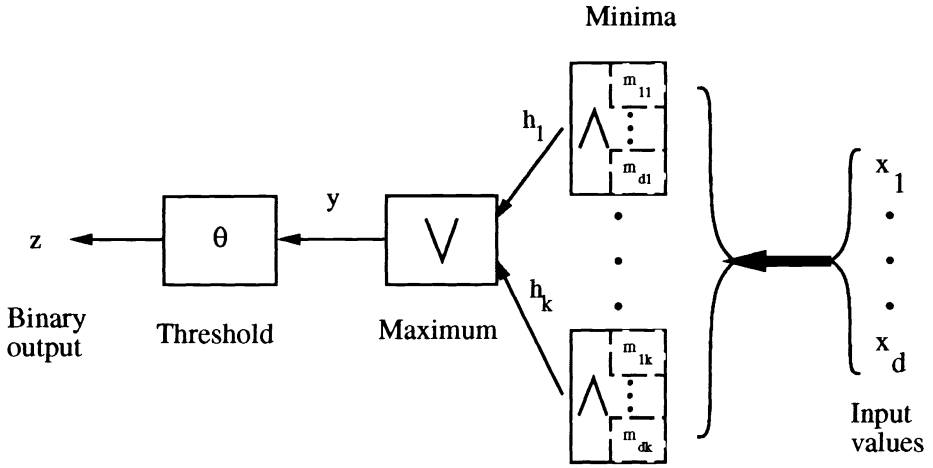


Figure 1: Signal flow diagram of the min-max classifier

Spectrum" in [6]), which is constructed from morphological size distributions (granulometries) [7, 8]. It provides a quantitative multiscale description of shape based on morphological operations.

This paper is organized as follows: Section 2 is a very brief overview of the definition of min-max classifier. The LMS algorithm is described in Section 3. Finally, the LMS algorithm is applied to handwritten character recognition. The experimental results are presented in Section 4, which also contains a discussion of the feature extraction procedures.

2 Min-Max Functions

We shall only provide the basic definitions to facilitate discussion in the following sections. A more detailed exposition of the properties of the min-max classifiers can be found in [1]. The min-max functions are an extension of Boolean functions to real variables based on the theory of fuzzy sets [9], where the Boolean AND/OR corresponds to a MIN/MAX.

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

$$y = f(x_1, x_2, ..., x_d) = \max_{j=1}^{k} \min_{i \in I_j} \ell_i \quad , \qquad \ell_i \in \{x_i, 1 - x_i\} \tag{1}$$

where an argument ℓ_i is called a *literal*, equaling either a variable x_i or its complement $1 - x_i$. Each minimum function $h_j = \min_{i \in I_j} \ell_i$ is a called a *min term*. The *input mask* 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 maximum is calculated over the output of the minimum functions. In this paper we shall restrict the total number of minima k to be a prespecified constant.

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 *min-max classifier* $f_\theta : [0,1]^d \rightarrow \{0,1\}$ defined by

$$z = f_\theta(\vec{x}) = \begin{cases} 1 & \text{if } f(\vec{x}) \geq \theta, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

A signal flow representation of this equation is shown in Figure 1.

3 LMS Algorithm for Training Min-Max Classifiers

We start by describing the LMS algorithm [3] 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 interested in minimizing the mean-square error (MSE) of the output $z(t)$ and a desired process $d(t)$,

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

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 $\vec{p}(t)$ is changed along the direction so as to produce the largest decrease in the MSE. This is achieved using the iterative equation

$$\vec{p}(t+1) = \vec{p}(t) - \mu \nabla_{\vec{p}} \mathcal{E}(t) \tag{3}$$

The *convergence factor* μ 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, resulting in the following approximate update equation:

$$\vec{p}(t+1) = \vec{p}(t) - 2\mu(z(t) - d(t))\mu \nabla_{\vec{p}} z(t) \tag{4}$$

In this paper this update rule is used for supervised training of min-max classifiers. The members of the training set are formed into a sequence $d(t)$. The step variable t is the sample number within the training data set.

Specializing to the min-max classifier, the parameter $p(t)$ is a $k \times 2d+1$ dimensional vector. They include the threshold θ, and $k \times 2d$ parameters for specifying the I_j — there are k input masks, each requiring $2d$ parameters to determine whether the variable or its complement is to be included. In the discussion that follows, it is more convenient to restrict the minimum functions to be *monotone*, i.e. their input variables are not complemented. Complemented variables are introduced by remapping d-dimensional input vector $\vec{x}$ into a $2d$-dimensional one $\vec{X}$ composed of pairs of uncomplemented and complemented variables.

Now, we define the parameters that specify the input masks $I_j, j = 1, \ldots, k$. Since the learning algorithm employs gradients of the internal states, the states should be continuous variables. We adopt the approach suggested in [4]: for each I_j we introduce $2d$ continuous real variables m_{ij}, $i = 1, \ldots, 2d$ corresponding to the remapped feature vector $\vec{X}$. 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 $m_{ij} < 0$.

We are now in a position to find the gradients required in Equation (4). It follows from the definition of the threshold function

$$z = \frac{\text{sgn}(y - \theta) + 1}{2} \tag{5}$$

where sgn() is the signum function defined as

$$\text{sgn}(x) = \begin{cases} 1 & \text{if } x \geq 0, \\ 0 & \text{if } x = 0, \\ -1 & \text{otherwise.} \end{cases}$$

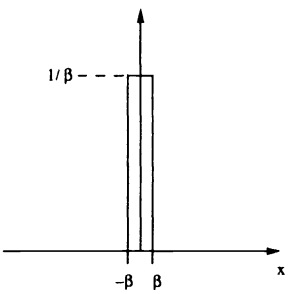


Figure 2: The approximation to the $\delta(x)$. β is a training parameter to be specified.

A strict forward calculation yields the derivative,

$$\frac{\partial z}{\partial \theta} = -\delta(y - \theta). \tag{6}$$

The delta function in the above equation is approximated using a finite impulse shown in Figure 2. The parameter β controls the width of the pulse. Using this approximation, one derives the approximate derivative

$$\frac{\partial z}{\partial \theta} = \begin{cases} -\frac{1}{\beta} & \text{if } |y - \theta| \leq \beta \ , \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

To calculate the derivatives with respect to the parameters m_{ij}, one employs the chain rule:

$$\frac{\partial z}{\partial m_{ij}} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial h_j} \frac{\partial h_j}{\partial m_{ij}} \tag{8}$$

3.1 Derivative of Maximum

The second term in the right hand side of Equation (8) 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

$$y = \max\{h_1, \ldots, h_k\}.$$

One can use instead an *implicit* definition of the maximum function [4] :

$$G(y, h_1, \ldots, h_k) = \sum_{j=1}^{k} \{\mathrm{sgn}(y - h_j) - 1\} + G_e = 0. \tag{9}$$

The quantity G_e is the number of inputs h_j that are equal to the output y. The total derivative of $G()$ with respect to m_{ij} is identically zero. Following the derivation in [4], one obtains

$$\frac{\partial y}{\partial h_j} \approx \begin{cases} \frac{1}{G_e} & \text{if } y = h_i \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

3.2 Derivative of Minimum

The third derivative in Equation (8) now poses no special difficulties. The output of the j-th minimum function is

$$h_j = \min\{X_i : i = 1, \ldots, 2d \text{ and } m_{ij} \geq 0\}$$

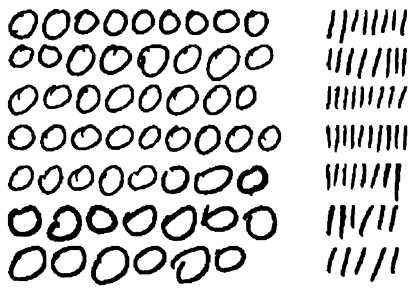


Figure 3: Sample data from the handwritten database.

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

$$L(h_j, X_1, \ldots, X_{2d}, m_{1j}, \ldots, m_{2dj}) = \sum_{i=1}^{2d} \left\{ \frac{\text{sgn}(m_{ij}) + 1}{2} \text{sgn}(X_i - h_j) - 1 \right\} + L_e = 0, \tag{11}$$

where L_e denotes the number of inputs X_i whose $m_{ij} \geq 0$ and are equal to the output value h_j. Following calculations similar to the previous section one finds the approximate gradient:

$$\frac{\partial h_j}{\partial m_{ij}} \approx \begin{cases} -\frac{1}{2L_e} & \text{if } |m_{ij}| \leq \beta \text{ and } h_j = X_i \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

4 Application to Handwritten Character Recognition

In this section we shall apply the LMS algorithm to hand-written character recognition. Due to the binary nature of the min-max classifier we shall use it for distinguishing between 0's and 1's. Our database consists of segmented and cleaned handwritten digits.[1] Sample 0 and 1 digits from the database are shown in Figure 3. The size of each digit is around 50×50 pixels. For our experiments, we selected 600 samples of 0's and 1's to form a training set of 1200 digits. The test set consists of 200 samples of each of 0's and 1's for a total of 400 test digits.

4.1 Feature Extraction

4.1.1 Fourier Descriptors

The first variety of feature is based on the chain code description of the boundary of the characters [10]. The algorithm is similar to Algorithm 7.1 in [10] with the modification that we used a four-connectivity chain instead. Only the outer boundary of the character is traced.

The first one is related to the Fourier descriptors discussed in [11]. Given the initial point, the boundary is fully characterized by the direction of the transversal at each step. Corresponding to each direction is the angle it makes with the x axis. Using this mapping scheme, one obtains a periodic functions $\phi(n), 0 \leq n \leq L-1$, where L is the total length of the chain code. The Discrete Fourier Transform (DFT) of this sequence is

$$\hat{\phi}(k) = \frac{1}{L} \sum_{n=0}^{L-1} \phi(n) \exp^{-jnk(\frac{2\pi}{L})}, \text{ for } k \geq 0.$$

[1] This database was supplied by Dan Bloomberg of Xerox PARC.

Since the inputs to the min-max classifier have to be restricted to a range of $[0, 1]$, the actual features are computed using the normalized magnitude of the DFT:

$$\Phi(k) = \frac{|\hat{\phi}(k)|}{|\hat{\phi}(0)|}. \tag{13}$$

We shall call this the angular Fourier descriptor. It is within the required range because $|\hat{\phi}(k)| \leq |\hat{\phi}(0)|$ as can be proved easily using the triangle inequality and the fact that $\phi(n)$ are nonnegative values.

Another description of the boundary curve is the sequence of position vectors $(x(n), y(n))$ of its pixels. Screen coordinates were used (x increases moving right and y increases moving down). Two Fourier descriptors are defined using the magnitude of the Discrete Fourier Transforms of $x(n)$ and $y(n)$ as

$$X(k) = \frac{|\hat{x}(k)|}{|\hat{x}(0)|}, \tag{14}$$

$$Y(k) = \frac{|\hat{y}(k)|}{|\hat{y}(0)|} \tag{15}$$

where $\hat{x}$ and $\hat{y}$ denotes respectively the DFT of x and y. The same normalization as in Equation (13) is used.

For our experiments, values of the Fourier descriptors for $1 \leq k \leq 10$ were used. Only low frequency components were used so as to increase noise tolerance. Samples of the features are shown in Figure 5 (left).

4.1.2 Morphological Shape-Size Histogram

We shall only describe the closing part of the shape-size histogram since this is the one we employed. To generate the shape-size histogram, the image is nonlinearly smoothed using the morphological closing filter. Intuitively, the closing smoothes the background by removing parts of it that the *structuring element* B doesn't fit into.

In generating the shape-size histogram, structuring elements of increasing sizes are used. A structuring element of size n (denoted nB) is given by:

$$nB = \underbrace{B \oplus \ldots \oplus B}_{n} \quad \text{for } n \geq 1, \text{ and } \quad 0B = \{0\}.$$

By taking the difference in the area of images smoothed at consecutive sizes, one obtains the shape-size histogram. More formally, we have the following definition:

$$SH_X(n, B) = A[X \bullet (n+1)B] - A[X \bullet nB] \text{ for } n = 0, 1, 2, \ldots$$

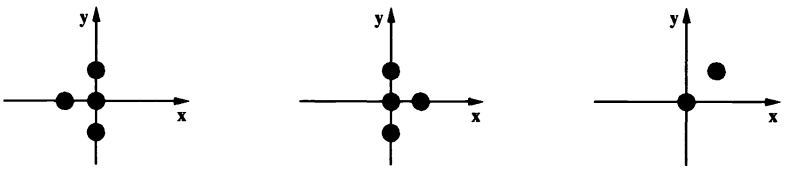
where $A[\,]$ denotes the area of the foreground of the image (for discretized image this is equated to the pixel count). The shape-size histogram is always nonnegative [6].

As in the case of the Fourier descriptors, the features should be normalized to within the range of 0 to 1. A natural normalization to reduce the effect of differing image sizes is by dividing $SH_X(n, B)$ by the area of the *maximally closed* image (M) defined as

$$M = \lim_{n \to \infty} X \bullet nB = X \bullet KB$$

where the limit is attained for a finite $n = K$ due to the boundedness of X (see [6]). Obviously, $SH_X(n, B) \leq A[M]$. Hence, the actual features used were the *normalized* shape-size histogram:

$$NSH_X(n, B) = \frac{SH_X(n, B)}{A[M]}, \text{ for } n \geq 0.$$



(a) (b) (c)

Figure 4: Structuring elements used in calculating the shape-size histograms. (a) Left pointing triangle. (b) Right pointing triangle. (c) 45° vector.

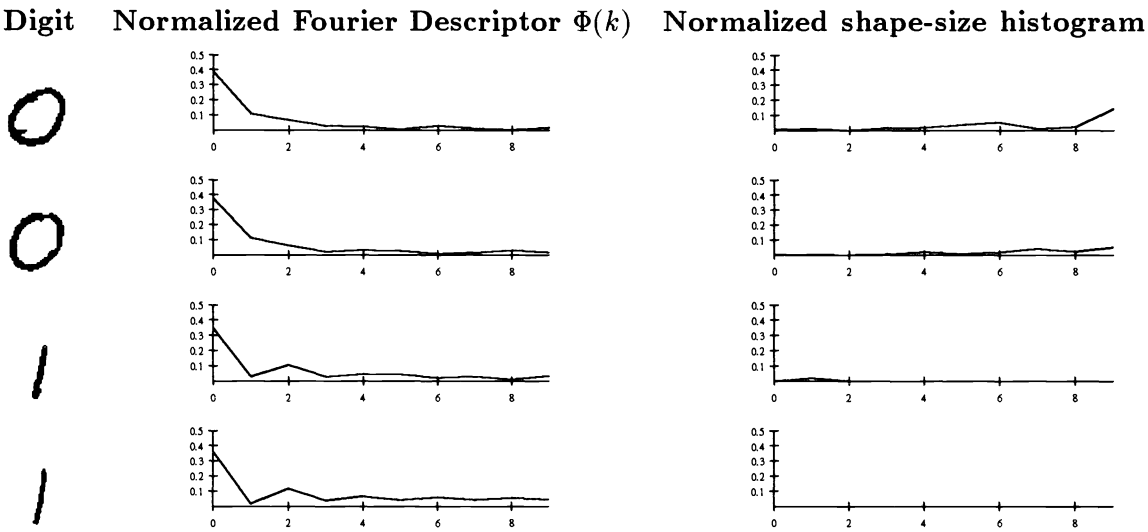


Figure 5: Examples of features employed in the experiments. The center column shows the normalized Fourier descriptor $\Phi(k)$, while the third one shows the normalized shape-size histogram calculated with the left pointing triangle in Figure 4(a).

In our experiments we computed closings of size up to $n = 39$. For our images and choice of structuring elements, this value is greater than K. Of the forty components that were computed, only ten of them ($0 \leq n \leq 9$) were used.

Turning to the structuring elements B, the three designs are shown in Figure 4. The left pointing and right pointing triangles are used to detect left and right pointing cavities respectively. The 45 degree vector is useful for finding long strokes, such as those in characters 1 and 7. For samples of the feature see Figure 5 (right).

4.1.3 Rotating the Feature Vectors

It turns out the boundary of the decision regions of min-max classifiers are axis-parallel (see [1] for proof). If the boundary between different clusters in the data conform to this geometry, a low error rate could be achieved. A simple preprocessing scheme that we tried is rotation of the feature so as to diagonalize the sample covariance matrix.

A *principal direction* is an eigenvector of the sample covariance matrix, defined as

$$\hat{\Lambda}_{ij} = \frac{1}{N} \sum (X_i - a_i)(X_j - a_j)$$

where X_i and a_i denotes the i-th component of a feature vector and the mean vector $\vec{a}$ respectively. The summation is carried over the entire set of N feature vectors from the training set. If clusters of different classes are fairly localized, one expects a principal direction to point perpendicular to the decision plane. By rotating the eigenvectors to be axis-parallel, hopefully the decision regions will become axis-parallel too. Using elementary linear algebra, one derives the transformation on the feature vectors

$$\vec{X}' = O^T(\vec{X} - \vec{a}) \tag{16}$$

where O is the matrix whose columns are the eigenvectors of $\hat{\Lambda}$. After the rotation, a second step is performed in which the feature vectors are shrinked uniformly in all directions and then translated so that an axis-parallel hypercube transformed in the same way would just fit within its original position. This is to ensure that the transformed feature vector remains in the valid domain $[0,1]^d$.

4.2 Experimental Results

The experiments are centered around three ideas:

Comparison with neural network The misclassification rates of min-max classifiers trained via the LMS algorithm and feed forward neural networks trained using the back propagation algorithm [12] are compared.

Importance of each type of feature Two types of features we used were Fourier descriptors and morphological shape-size histograms. Different combinations of these two features were used as inputs and the effect on the error rate examined.

Rotating the data We explored the possibility of reducing the error rates by preprocessing both the training and test data using the ideas in Section 4.1.3.

The neural network we used has one input layer, one hidden layer and one output node. This architecture was used in order to provide fair comparison with the min-max classifiers — the maximum gate corresponds to the output node and the minimum gates to the hidden nodes. The number of nodes in the hidden layer was chosen to be equal to the number of minima in the min-max function. The second consideration in maintaining fairness is to allow the LMS and back propagation programs consume the same amount of CPU time. We approximated this requirement by limiting the number of different initial conditions that the back propagation to five and report the average of the best four; while the LMS was run with fifteen initial conditions and we report the average of the best five. In the back propagation algorithm we also included a momentum term in the update equation.

To explore the importance of different features, three different sets of feature vectors were extracted from the input images:

1. *Mixed set (Set I)* Each feature vector is a concatenation of 10 components of angular Fourier descriptor $\Phi(k)$, 10 components of normalized shape-size histograms calculated using the right pointing triangle (Figure 4(b)), and 10 components of shape-size histogram for the 45 degree vector (Figure 4(c)).

2. *All Fourier descriptors (Set II)* 10 components from the three different types of Fourier descriptors ($\Phi(k), X(k)$, and $Y(k)$).

3. *All shape-size histograms (Set III)* 10 components of normalized shape-size histograms calculated using each of the structuring elements shown in Figure 4.

All feature vector has a total of 30 components so that the comparison remains fair.

Finally, for each of these data sets we ran the learning algorithms (LMS and back propagation) both with and without preprocessing, making a total of six different data sets.

In our experiments, we used the LMS algorithm in a "batch" mode operation. This means that training and testing are separated into two distinct stages. During each iteration of training stage, the min-max function is changed by running the LMS procedure for all the data in the training set. We shall refer each iteration step as a *scan*. After each scan, the error rate of the current classifier is calculated using the training data set. The m_{ij} parameters for the minimum gates were initialized using a random number generator. However, we found that the convergence depends quite critically on the initial value of the threshold θ and therefore it is not randomly generated. In our particular implementation, the training program performs a fixed number of scans (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 set. We only report the setting that resulted in the lowest observed training error rate. Typical values for the convergence rate is $\mu_{\min} = 1 \times 10^{-3}, \beta_{\min} = 1, \mu_\theta = 1 \times 10^{-4}$, and $\beta_\theta = 0.1$. A similar procedure was also used for the back propagation where we used 1000 scans.

Table 1 shows the results for the mixed data set (both Fourier descriptors and shape-size histogram). Typical error rates for the min-max classifiers are 1.75% on the training set and 4.5% on the test set. These figures are very encouraging. For the back propagation algorithm, the error rates dropped to 0.4% on the training set, which is one fourth of the value for the min-max classifiers. The error rate on the test set dropped by 0.5% to 4%. A smaller error rate for the neural networks is expected because they are more general classifiers (i.e. their decision regions are more flexible than the min-max functions).

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 Figure 6. 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. Judging 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 20 scans is required before 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 functions requires only binary comparisons; while neural networks require the calculating the computational costly exponential functions.

The advantage in error rate of the neural network over the min-max classifier diminishes after preprocessing was incorporated. Tables 2 contains the result generated from the preprocessed data. For the LMS algorithm preprocessing reduced the training error by a significant factor of 3 from the unprocessed value of 1.75% to around 0.6%. However, the training error dropped by a lower margin from a typical 4.5% to around 3.7%. These error figures are comparable to the error rate achieved by the neural networks. Turning to the back propagation algorithm, both the training and test error rates increased. This is probably due to the decrease in cluster separation due to the scaling operation in the preprocessing.

Table 3 summarizes the results for training the min-max classifiers with data set II (pure Fourier descriptors). The training and test error rates for the min-max functions is much higher than those obtained with the mixed data set (I), indicating that it is advantageous to include both types of data for the min-max classifiers. Interestingly, the training error rate for the back propagation dropped to an occasional 0%.

The corresponding results with preprocessing are shown in Table 4. By using preprocessing on the feature vectors, the training error rates for the min-max classifiers dropped by a factor of about four, with a corresponding decrease in the test error rate by a factor of 2. However, for the back propagation algorithm an increase in the training and test error rate was observed. Again this is possibly due to the

Min-Max			Neural network		
No. of minima	Error (train)	Error (test)	Network	Error (train)	Error (test)
1	1.75	4.5	1,1	0.396	3.813
3	1.75	4.5	3,1	0.417	3.75
5	1.75	4.5	5,1	0.333	3.813
7	1.75	4.5	7,1	0.333	3.5

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

Min-Max			Neural network		
No. of minima	Error (train)	Error (test)	Network	Error (train)	Error (test)
1	0.517	3.7	1,1	2.208	5.063
3	0.517	3.7	3,1	2.375	4.875
5	0.517	3.75	5,1	2.417	4.688
7	0.517	3.7	7,1	2.271	4.875

Table 2: Results for 0-1 classification problem using *preprocessed* Fourier descriptors plus shape-size histograms.

Min-Max			Neural network		
No. of minima	Error (train)	Error (test)	Network	Error (train)	Error (test)
1	11.133	12.15	1,1	0.375	4.313
3	11.85	10.8	3,1	0.0	3.94
5	11.6	10.8	5,1	0.0	4
7	12.083	10.75	7,1	0.0	4

Table 3: Results for the 0-1 classification problem using *only* Fourier descriptors.

Min-Max			Neural network		
No. of minima	Error (train)	Error (test)	Network	Error (train)	Error (test)
1	1.883	5	1,1	24.688	21.563
3	2.017	4.6	3,1	12.625	14
5	1.883	5.1	5,1	9.729	10.75
7	1.867	5.15	7,1	10.292	11.188

Table 4: Results generated using *preprocessed* Fourier descriptors.

Min-Max			Neural network		
No. of minima	Error (train)	Error (test)	Network	Error (train)	Error (test)
1	6.117	6.5	1,1	3.646	7.75
3	2.05	2.65	3,1	3.729	7.813
5	1.8	2.65	5,1	3.958	8.875
7	1.733	2.25	7,1	3.708	8

Table 5: Results using only shape-size histograms.

Min-Max			Neural network		
No. of minima	Error (train)	Error (test)	Network	Error (train)	Error (test)
1	6.917	7.15	1,1	31.521	38.686
3	6.95	7.55	3,1	24.458	29.5
5	6.967	7.7	5,1	21.375	24.688
7	6.333	8	7,1	22.021	25

Table 6: Results using *preprocessed* shape-size histograms.

reduction in cluster separation. For the LMS algorithm, even with data preprocessing the training and test error rates were still higher than those obtained with the mixed feature type data set.

The third data set that we used consisted only of shape-size histograms. Table 5 displays the results. Both the training and test error rate for the min-max function were higher than those obtained using data set I (both types of features), providing another indication that it is better to use both Fourier descriptors and shape-size histograms for the LMS algorithm. Preprocessing this set of feature vectors (Table 6), however, did not improve the performance min-max classifiers.

5 Conclusion

We reported on our preliminary experimentations on the practical training of min-max classifiers and applied it to handwritten character recognition. The results are encouraging. By using preprocessing on the feature vectors we were able to achieve error rates comparable to neural networks trained by back propagation. The main advantage of the min-max classifiers is their simplicity and faster convergence. We are currently working on refinements of the training equations and the features.

References

[1] P.-F. Yang and P. Maragos, "Learnability of min-max pattern classifiers," in *SPIE Vol. 1606 Visual Communications and Image Processing '91: Image Processing*, pp. 294–308, 1991.

[2] L. G. Valiant, "A theory of the learnable," *Communications of the ACM*, vol. 27, pp. 1134–1142, Nov. 1984.

[3] B. Widrow and S. D. Stearns, *Adaptive Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1985.

[4] P. Salembier, "Structuring element adaptation for morphological filters," *Journal of Visual Communication and Image Representations*, vol. 3, pp. 115–136, June 1992.

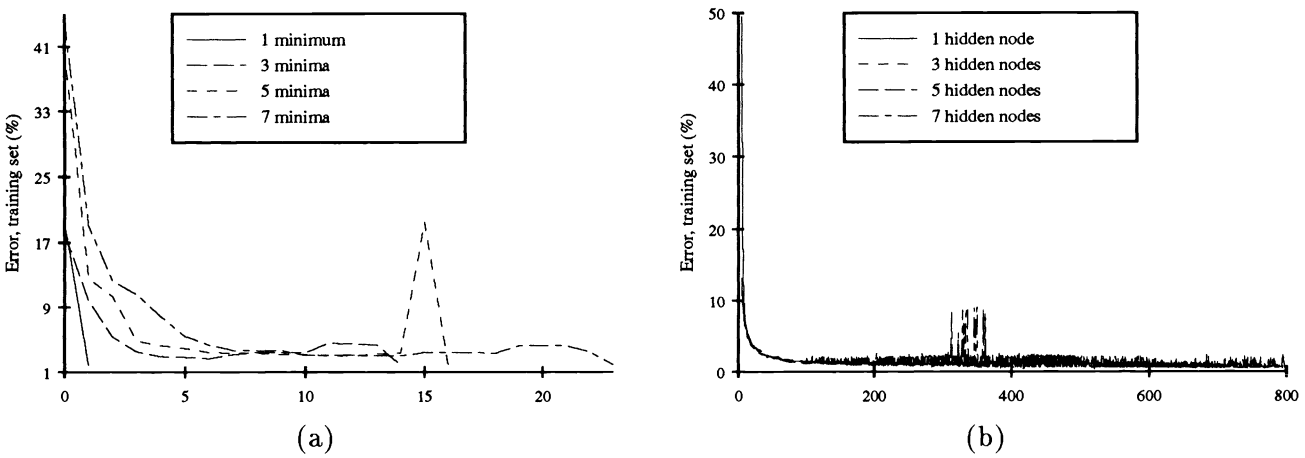


Figure 6: Plots comparing convergence speed of LMS algorithm with the back propagation. The lines are terminated after the minimum error rate over the entire scan is achieved. (a) Typical convergence plots for the LMS algorithm. The minimum error rate (1.75%) is attained within 25 scans. (b) Convergence plots for the back propagation on the same data set. The error rate decreases to within 1.5% after about 60 scans.

[5] 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*, (San Diego, CA), pp. 212–223, 1991.

[6] P. Maragos, "Pattern spectrum and multiscale shape representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 701–716, 1989.

[7] G. Matheron, *Random Sets and Integral Geometry.* New York: Wiley, 1974.

[8] J. Serra, *Image Analysis and Mathematical Morphology.* New York: Academic Press, 1982.

[9] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, pp. 338–353, 1965.

[10] T. Pavlidis, *Algorithms for Graphics and Image Processing.* Rockville, MD: Computer Science Press, 1982.

[11] C. T. Zahn and R. Z. Roskies, "Fourier descriptors for plane closed curves," *IEEE Computer*, vol. C-21, pp. 269–281, March 1972.

[12] D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, *Parallel distributed processing: explorations in the microstructure of cognition.* Cambridge, MA: MIT Press, 1986.