A neural-network-based fuzzy classifier

Abe, Shigeo
Ming-Shong Lan
Uebele, Folkmer

[19] C. O'Dunlaing and C. K. Yap, "A 'Retraction' Method for Planning the Motion of a Disc," *J. of Algorithms*, vol. 6, pp. 104–111, 1985.
[20] J. T. Schwartz and M. Sharir, "A Survey of Motion Planning and Related Geometric Algorithms," *Artificial Intelligence*, vol. 37, Nos. 1–3, pp. 157–169, Dec. 1988.
[21] J. T. Schwartz, M. Sharir and J. Hopcroft (Eds.), *Planning, Geometry, and Complexity of Robot Motion*, Ablex Pub. Corp., Norwood, NJ, 1987.
[22] M. Sharir and A. Schorr, "On Shortest Paths in Polyhedral Spaces," *SIAM J. of Computing*, vol. 15, no. 1, pp. 193–215, Feb. 1986.
[23] R. A. Singer, "Estimating Optimal Tracking Filter Performance for Manned Maneuvering Targets," *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-6, no. 4, pp. 473–483, July 1970.
[24] S. H. Suh and K. G. Shin, "A Variational Dynamic Programming Approach to Robot-Path Planning With a Distance-Safety Criterion," *IEEE Trans. Robotics Automat.*, vol. 4, no. 3, pp. 334–349, June 1988.
[25] D. D. Sworder and R. G. Hutchins, "Image-Enhanced Tracking," *IEEE Trans. Aerosp Electron. Syst.*, vol. AES-25, no. 5, pp. 701–710, Sept. 1989.
[26] ——, "Maneuver Estimation Using Measurements of Orientation," *IEEE Trans. on Aerosp. Electron. Syst.*, vol. AES-26, no. 4, pp. 625–637, July 1990.
[27] Y. A. Teng, D. DeMenthon and L. S. Davis, "Stealth Terrain Navigation," *IEEE Trans. Syst. Man Cyber.*, vol. 23, no. 1, pp. 96–110, Jan./Feb. 1993.
[28] J. S. Thorp, "Optimal Tracking of Maneuvering Targets," *IEEE Trans. on Aerosp. Electron. Syst.*, vol. AES-9, no. 4, pp. 512–519, July 1973.
[29] C. K. Yap, "Algorithmic Motion Planning," *Advances in Robotics*, J. T. Schwartz, and C. K. Yap, (Eds.), Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 95–143, 1987.

A Neural-Network-Based Fuzzy Classifier

Volkmar Uebele, Shigeo Abe, and Ming-Shong Lan

Abstract—**In this paper, a new technique for generating fuzzy rules for pattern classification is discussed. First, separation hyperplanes for classes are extracted from a trained neural network. Then, for each class, convex existence regions in the input space are approximated by shifting these hyperplanes in parallel using the training data set for the classes. Using fuzzy rules defined for each class, input data are directly classified without the use of the neural network. This method is applied to a number recognition system as well as to a blood cell classification system. Classifying performance is compared with that obtained with neural networks.**

I. Introduction

In the past decade, fuzzy logic [1] has proved to be a powerful tool for decision-making systems, such as expert systems and pattern classification systems. However, knowledge acquisition to create fuzzy rule bases is often difficult and time-consuming. Furthermore it requires deep insight into the system. Fuzzy rules are commonly expressed by means of linguistic variables that can be interpreted as fuzzy regions in the input and output spaces. The degree of membership of an input datum associated to a particular data group or class is determined by the degrees of membership of the fuzzy rules for that class.

Manuscript received May 29, 1993; revised March 24, 1994.
V. Uebele and S. Abe are with the Hitachi Research Laboratory, Hitachi Ltd., 7-1-1 Omika-cho, Hitachi-shi, Ibaraki-ken, 319-12 Japan.
M.-S. Lan is with the Science Center, Rockwell International Corporation, 1049 Camino Dos Rios, Thousand Oaks, California 91358 USA.
IEEE Log Number 9405712.

To automate the knowledge acquisition process, many attempts [2]–[4] have been made to transform numerical input-output data into fuzzy rules. Since neural networks and fuzzy systems are convertible [5], it is reasonable to combine the merits of both paradigms, namely the learning ability of the neural network [6] and the easy implementation of fuzzy logic. Based on this linkage, in [2] a classifier was constructed using a hybrid form of a feedforward neural network that imitated fuzzy reasoning, in which fuzzy rules were created using a multi-stage learning process and after learning, the final set of rules for the rule base was selected according to the connection strength between the neurons of two consecutive layers. A similar approach was also proposed in [3], where supervised learning of the network was carried out using fuzzy input neurons. After learning, fuzzy rules were selected according to the strength of a relationship, called the causal index, between the neurons of the input layer and those of the output layer. Another approach as in [4], obtained fuzzy rules by applying a hybrid scheme which used self-organizing and supervised learning techniques. All the techniques mentioned above used conventional fuzzy logic where the boundaries of fuzzy regions were formed by hyperplanes parallel to the input variables. Thus it is difficult to express complicated fuzzy regions with only a small number of rules.

To overcome this, in [7] the original classification power of a trained neural network was maintained by using separation hyperplanes [8], [9], extracted from the weights between input and hidden layers, as crisp boundaries for class existence regions. Input data were classified according to which side of the hyperplane they are on, which reduced classification to a decision-tree algorithm. The advantages of this approach are: 1) no hybrid form of the network structure or learning process for the neural network is necessary; and 2) the separation boundaries between class regions are not restricted to being parallel to the input variables. However, this method allows only one existence region per class, restricting it to problems with singly separable classes. Namely, a connected class region must be separable from all other class regions by means of a specific number of separation hyperplanes. Moreover, to obtain adequate class boundaries, many training data are usually required which slows down the learning process. Further, the generalization ability can be controlled only indirectly by applying a tuning step after the network is trained.

The neural-network-based fuzzy classifier discussed in this paper allows any number of existence regions for each class and also reduces the number by clustering regions of the same class. Furthermore, using fuzzy regions, the generalization ability can be directly controlled. The proposed algorithm is relatively easy to implement compared to the methods discussed in [2]–[4].

In the following, we first describe a method that generates existence regions of each class, using the separation hyperplanes extracted from a trained neural network. Then, we cluster the regions of the same classes and adjust the limits of the region by shifting the separation hyperplanes in parallel. Furthermore, the crisp boundaries of the regions are replaced by fuzzy membership functions which realize extended and more generalized fuzzy regions. Using the resultant fuzzy rule base, numerical input data can be directly classified without the need of the neural network. Finally, we apply this fuzzy classification approach to a number recognition system as well as to a blood cell classification system. Classification performance is compared to that gotten with neural networks.

II. Structure of Neural Networks for Pattern Classification

A. A Neural Network Architecture for Pattern Classification

Consider a usual multi-layered feed-forward neural network in which biased terms, represented by neurons with no input and a fixed output of 1, are added to the input and the hidden layers, respectively. For each layer, without considering the bias neuron, the number of neurons is $N(i)$, where $i = 1,\ldots,L$ and L is the number of layers. The neurons of the hidden layer and the output layer process incoming signals by a non-linear sigmoid function with saturation, which is expressed by $f(y) = 1/(1+e^{-y/T_c})$, where y is the input of the neuron and T_c is a constant. The neurons of the input layer are linear and are used to distribute input signals to the neurons of the first hidden layer. The neurons of two consecutive layers are completely connected by synapses represented by the weight matrix between these two layers.

Thus, the output vector $\mathbf{z}^{i+1}$ of the $(i+1)$-st layer is given by

$$\begin{aligned} \mathbf{z}^{i+1} &= \mathbf{f}(\mathbf{y}^{i+1}) && \text{and} \\ \mathbf{y}^{i+1} &= \mathbf{W}^i\mathbf{x}^i && \text{for } i = 1,\ldots,L-1, \\ \mathbf{x}^{i+1} &= \mathbf{z}^i && \text{for } i = 2,\ldots,L-1, \end{aligned} \tag{1}$$

where

$$\begin{aligned} \mathbf{x}^i &= [x_1^i,\ldots,x_{N(i)}^i,1]^T, \\ \mathbf{y}^{i+1} &= [y_1^{i+1},\ldots,y_{N(i+1)}^{i+1}]^T, \\ \mathbf{z}^{i+1} &= [z_1^{i+1},\ldots,z_{N(i+1)}^{i+1}]^T, \\ \mathbf{f}(\mathbf{y}^{i+1}) &= [f(y_1^{i+1}),\ldots,f(y_{N(i+1)}^{i+1})]^T, \end{aligned}$$

$\mathbf{x}^i$ is the i-th layer input vector and

$$\mathbf{W}^i = \begin{bmatrix} w_{11}^i & \cdots & w_{1N(i)+1}^i \\ \vdots & w_{pq}^i & \vdots \\ w_{N(i+1)1}^i & \cdots & w_{N(i+1)N(i)+1}^i \end{bmatrix} \tag{2}$$

is the weight matrix between the i-th and the $(i+1)$-st layer, w_{pq}^i is the weight between the p-th neuron of the $(i+1)$-st layer and the q-th neuron of the i-th layer. Weights w_{pq}^i are determined by the Backpropagation Algorithm (BP) [6] using a training data set given by M pairs of $N(1)$-dimensional inputs and the associated $N(L)$-dimensional target outputs as follows:

$$\{(x_i^m, t_j^m)\} \text{ for } i = 1,\ldots,N(1), j = 1,\ldots,N(L) \text{ and } m = 1,\ldots,M. \tag{3}$$

When applying a neural network to pattern classification, the number of output neurons is the same as the number of classes and the j-th neuron corresponds to the j-th class. The target vectors $\mathbf{t}^m = [t_1^m,\ldots,t_{N(L)}^m]^T$ are determined so that for class c the value t_c^m is 1 and all other t_j^m, with $c \neq j$, are 0. Thus, an input vector which produces the highest value at the c-th output neuron of the neural network is classified as class c. Fig. 1 shows a three-layered case of the network architecture used in this paper.

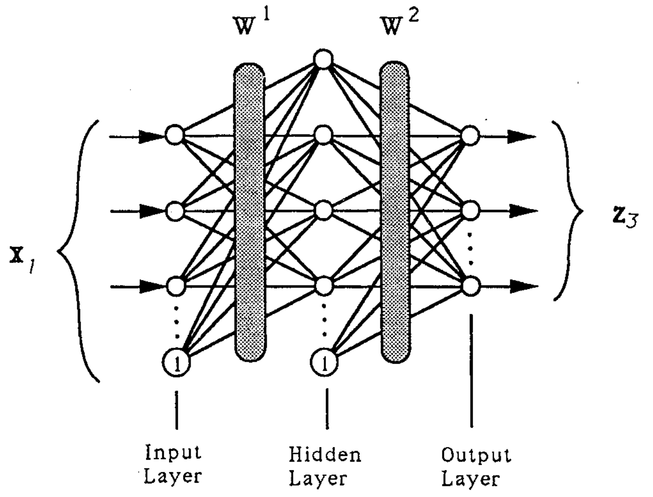


Fig. 1. Architecture of a Three-layered Pattern Classification Neural Network.

B. Interpreting Weight Matrices as Hyperplanes

The rows of the weight matrices of a neural network can be interpreted as the coefficients of hyperplanes [7]–[9]. Assuming $y_j^{i+1} = 0$ in (1), thus

$$\mathbf{w}_j^i\mathbf{x}^i = 0 \tag{4}$$

where $\mathbf{w}_j^i$ is the j-th row vector of $\mathbf{W}^i$, which represents a hyperplane in the $N(i)$-dimensional space. When $x_{N(i)+1}^i = 1$, a change of weight $w_{jN(i)+1}^i$ causes a parallel displacement of the j-th hyperplane. From (1), the value of z_j^{i+1} corresponding to y_j^{i+1} satisfying (4), which is a point on the hyperplane, is $1/2$. We say that the $N(i)$-dimensional point is on the positive side of the hyperplane if

$$y_j^{i+1} \geq 0 \text{ or } z_j^{i+1} \geq 1/2 \tag{5}$$

and on the negative side if

$$y_j^{i+1} < 0 \text{ or } z_j^{i+1} < 1/2. \tag{6}$$

In this sense, we can interpret the weight matrix between the i-th and the $(i+1)$-st layers of the neural network as a set of $N(i+1)$ hyperplanes, which partition the universe of discourse of the $N(i)$-dimensional space. Therefore, we call these hyperplanes separation hyperplanes.

III. Class Existence Regions in the Input Space

In this section we show how existence regions of classes in the input space can be defined by extracted hyperplanes of the weight matrix between the input and hidden layers.

A. Class Existence Regions Defined by Extracted Separation Hyperplanes

Since the $N(1)$-dimensional input space of the neural network is divided by $N(2)$ hyperplanes, we can define a maximum $2^{N(2)}$ disjoint regions R^k, by

$$\begin{aligned} R^1 &= \{\mathbf{x} \mid \mathbf{w}_1\mathbf{x} < 0 \cap \mathbf{w}_2\mathbf{x} < 0 \cap \cdots \cap \mathbf{w}_{N(2)}\mathbf{x} < 0\}, \\ R^2 &= \{\mathbf{x} \mid \mathbf{w}_1\mathbf{x} \geq 0 \cap \mathbf{w}_2\mathbf{x} < 0 \cap \cdots \cap \mathbf{w}_{N(2)}\mathbf{x} < 0\}, \\ &\vdots \\ R^{2^{N(2)}} &= \{\mathbf{x} \mid \mathbf{w}_1\mathbf{x} \geq 0 \cap \mathbf{w}_2\mathbf{x} \geq 0 \cap \cdots \cap \mathbf{w}_{N(2)}\mathbf{x} \geq 0\}, \end{aligned} \tag{7}$$

where some regions may be empty, i.e., $R^k = \emptyset$. The conjunction of all regions $\{R^1 \cup R^2 \cup \cdots \cup R^{2^{N(2)}}\}$ constitutes the entire input space and any $N(i)$-dimensional vector $\mathbf{x}$ is included in one region R^k.

The region R^k can be specified by the set of separation hyperplanes and information about which side of the hyperplanes it resides on. Thus, using a given set of hyperplanes, we can define a region R^k with a vector $\mathbf{p}^k = [p_1^k,\ldots,p_{N(2)}^k]^T$, whose j-th element indicates on which side of the j-th hyperplane the region resides. To designate that datum $\mathbf{x}$ is on the negative side of the hyperplane $\mathbf{w}_j\mathbf{x} = 0$, with $j = 1,\ldots,N(2)$, the corresponding value p_j^k in $\mathbf{p}^k$ is set to 0,

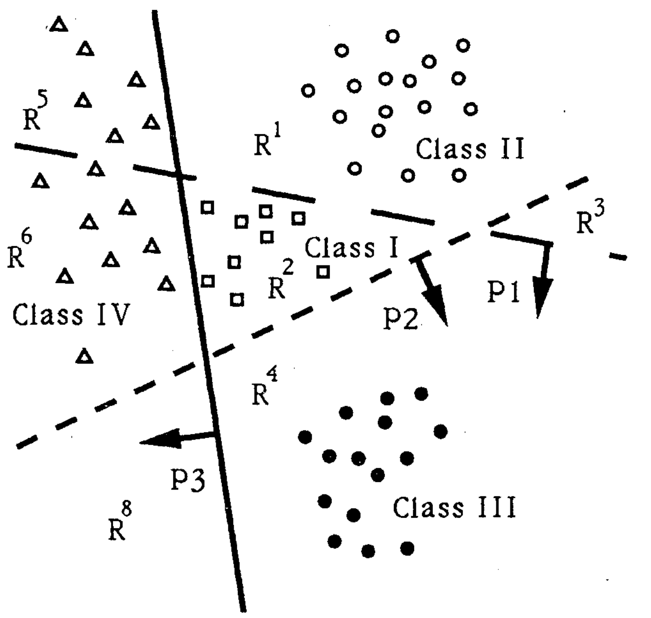


Fig. 2. Class Regions and Separation Hyperplanes in a Two-dimensional Input Space.

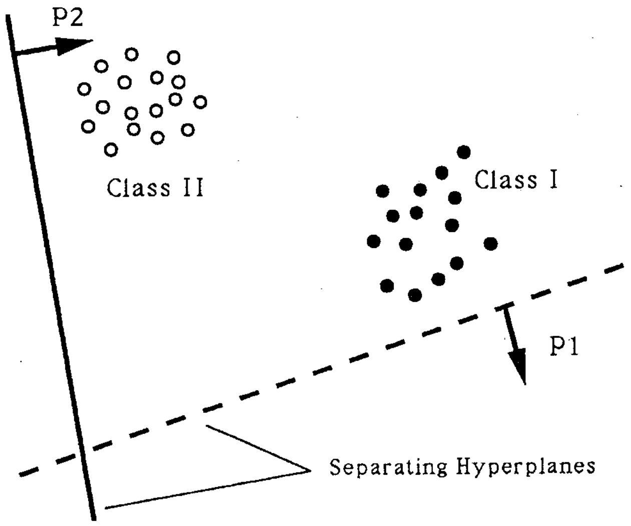


Fig. 3. Existence of Two Dependent Classes in a Two-dimensional Input Space.

while to designate it is on the positive side of the hyperplane, p_j^k is set to 1. The vector $\mathbf{p}^k$ is considered as the pattern of region R^k. All patterns $\mathbf{p}^k$ are disjoint.

For illustration, let us consider classification of four classes in a two-dimensional input space using a three-layered neural network with three hidden neurons, as shown in Fig. 2. We assume that a set of suitable separation hyperplanes has been obtained by successfully training a network. The arrows attached to the three hyperplanes $P1, P2$ and $P3$ show the positive side of the hyperplanes, and each symbol in the figure denotes a datum in the input space, belonging to the tagged class. As an input datum can lay on either side of the hyperplanes, the existence regions of the classes in the input space can be expressed in terms of the regions $R^1, \ldots, R^8$, formed by the separation hyperplanes $P1, P2$ and $P3$. Note that in the example R^7 is empty. Thus, each class with training data holds at least one existence region and pattern.

To obtain the existence regions of classes in terms of regions R^k, we check the output of the first hidden neurons for all M training input vectors $\mathbf{x}^m$, if either (5) or (6) holds, to decide on which side of the hyperplanes $\mathbf{x}^m$ is found. Then, we generate an associated pattern vector $\mathbf{p}^m$ in the same way as described above. We call this procedure digitizing, the resulting vector $\mathbf{p}^m$ is the digitized output or the pattern of the datum $\mathbf{x}^m$, and the value p_j^m is the j-th digit of pattern $\mathbf{p}^m$. All different patterns of all classes are stored, so that after digitizing all M training data, we can obtain a set of patterns $\mathbf{P}_c$ for each class c as follows:

$$\mathbf{P}_c = \{\mathbf{p}^m \mid m = 1, \ldots, M, \mathbf{x}^m \in \text{class } c\}. \quad (8)$$

Thus all the patterns in class $\mathbf{P}_c$ form the existence region

$$\bigcup_{m \text{ for } \mathbf{p}^m \in \mathbf{P}_c} R^m \quad (9)$$

of class c. We call this region an unclustered existence region in contrast to the clustered regions which are defined in the following subsection.

If all regions R^m belonging to one class are different from those of other classes, in other words, no data of one class exist in a region of another class, we consider these two independent. Otherwise, they are dependent. If two classes are dependent, they share the same pattern $\mathbf{p}^m$. As an example Fig. 3 shows two dependent classes in a two-dimensional input space.

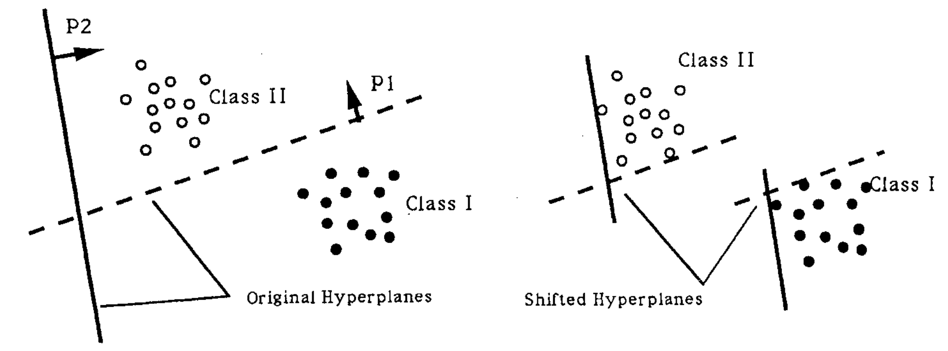
If the data of one class, like Class IV in Fig. 2, exist on both sides of a hyperplane $\mathbf{w}_j\mathbf{x} = 0$, i.e. $P1$, we can combine the two original regions on both sides of that hyperplane into one hyper region. In this case, we allow the digitized value p_j^m to be indefinite, and denote it as dc. As is discussed in the following, hyper existence regions can be obtained by clustering connected regions of one class using dc's.

B. Clustering Class Existence Regions

We show two methods of clustering existence regions, which reduce the number of patterns of a class. The first method defines a single region for each class where most of the class data exist, while the second method combines neighboring regions of a class.

Heuristic Approximation of Class Regions by A Single Pattern: A heuristic method for clustering has been proposed in [7]. It assumes that a class is singly separable, namely, it covers only one connected region in the input space, that can be separated by the $N(2)$ hyperplanes of the input-to-hidden weights [10]. Thus, for every class a single region and a single pattern can be created. First, all digits p_j^k of all patterns $\mathbf{p}^k$ of a class are checked successively. If the values of the j-th digit of p_j^k for all the patterns of that class are identical, the digit of the clustered pattern then has the same value. Otherwise, the digit of the clustered pattern is set to dc. In the second step, some digits with values of dc's of the clustered patterns are changed to either 0 or 1 until each pattern has at least one different digit for all clustered patterns of the remaining classes that is not dc. Digits are considered different if their values are different, e.g. dc and 1 are different. In this case, the digit of the class with the highest number of occurrences of the same value, either 0 or 1, is changed. Thus, this method generates regions, where most of the training data of classes reside. The resultant regions are called heuristically clustered regions.

Neighboring Regions Clustering: The premise of the second method for clustering patterns is that only neighbored regions containing the data of one class can be combined into a hyper region. An advantage over the heuristic method is that all regions of classes, obtained by digitizing the training data set, are considered. Thus, regions clustered with this method have the ability to approximate



(a) (b)

Fig. 4. (a) Class Regions in a Two-dimensional Input Space. (b) Single-sided Shifting of Separation Hyperplanes in a Two-dimensional Input Space.

complex shapes of class existence regions and allow plurally separable classes. To generate the combined patterns, we repeat the following algorithm for every class:

i) For class c check all the combinations of patterns $\mathbf{p}^i \in \mathbf{P}_c$ successively. If two patterns $\mathbf{p}^j$ and $\mathbf{p}^k$ of $\mathbf{P}_c$ differ in only one digit, namely $p_l^j \neq p_l^k$, for $l \in 1 \cdots N(1)$ and $p_i^j = p_i^k$, for $i = 1, \ldots, N(1), i \neq l$, create a new pattern $\mathbf{p}^r$ whose value for p_l^j is dc and the values for all other p_i^j are the same as those of the patterns $\mathbf{p}^j$ and $\mathbf{p}^k$. The region of the new associated pattern then is $C_c^r = R^j \cup R^k$. After all combinations are checked move to step (ii).

ii) Check all combinations of patterns in $\mathbf{P}_c$, including the newly created patterns $\mathbf{p}^r$. If two patterns $\mathbf{p}^j$ and $\mathbf{p}^k$ differ in one digit l only, with $p_l^j \neq p_l^k$ and either $p_l^j = dc$ or $p_l^k = dc$, delete the pattern which has the smaller number of dc's. After all combinations are checked, return to step (i) until no two patterns of this class can be further clustered.

The resulting patterns $\mathbf{p}^u$ and regions C_c^u, where $u = 1, \ldots, U$ and $c \in \{1, \ldots, N(L)\}$, incorporate all the patterns of the classes, which are obtained by digitizing the training data set, where U is the number of resulting patterns of all classes after combination. If overlaps among different classes exist before clustering, these overlaps still exist after clustering. However, they may be resolved by shifting the separation hyperplanes as is described next.

C. Class Representation with Shifted Hyperplanes

Based on the knowledge of the existence regions of all classes in the input space, we can construct a decision-tree-like classifier like that described in [7] without using the neural network. The existence regions are expressed in terms of regions C_c^u, whose boundaries are the separation hyperplanes extracted from input-to-hidden weights of the neural network. Classification is performed by digitizing a test datum $\mathbf{x}$ using the extracted separation hyperplanes and by comparing the digits of the resulting pattern, one at a time through the decision tree, with the stored pattern set of the classes. However, performance of this classifier will be poor, if training and test data differ, because test data may produce patterns that are not stored in the decision tree and thus cannot be classified.

Another problem arises, if the patterns of two classes are dependent, namely that two different classes exist in the same region, i.e., for $u \neq \tilde{u}$ and $c \neq \tilde{c}, C_{\tilde{c}}^u = C_{\tilde{c}}^{\tilde{u}}$. In this case, a test datum can belong to either of the dependent classes and hence no data in this region can be classified.

Since class boundaries are given in terms of the artificial limits of separation hyperplanes, they are only rough estimates of the regions a class occupies. Using the existence regions C_c^u, we lose valuable information about which part of region C_c^u the data of a class reside in. In order to improve the performance and reduce or resolve overlapping regions between classes, we need to define the existence regions more precisely.

To adjust the boundaries of existence regions, the $N(2)$ separation hyperplanes, obtained through a trained network, are shifted in parallel to the limits of the training data set, as illustrated in Figs. 4(a)–4(c). Fig. 4(a) shows the training data of two classes in a two-dimensional input space and two extracted separation hyperplanes. If we shift a separation hyperplane to the datum of the considered region in the training data set which is closest to the extracted hyperplane, this type of shifting, as shown in Fig. 4(b), is called single-sided shifting. This method requires knowing on which side of a hyperplane a region resides. If data of the region exist on either side of the considered hyperplane, namely the corresponding pattern value is dc, we do not shift the hyperplane for this region.

Further we can shift the hyperplanes in two directions to the closest and the farthest data points of the training data set of the considered existence region, as shown in Fig. 4(c). This type of shifting is called double-sided shifting. If class data reside on either side of the considered hyperplane, namely the corresponding pattern value is dc, we shift the hyperplane to the points of the training data set for that existence region, with the longest distance in the positive and the negative direction of the hyperplane vector $\mathbf{w}$, as illustrated in Fig. 4(d). Double-sided shifting limits the regions in two directions and thus reduces overlapping.

To apply shifting, we vary the weights $w_{jN(1)+1}^1, j = 1, \ldots, N(2)$, which correspond to the bias terms in the input-to-hidden matrix $\mathbf{W}^1$. The resulting column weight vectors are $\tilde{\mathbf{w}}_{N(1)+1}^u = [\tilde{w}_{1N(1)+1}^u, \cdots, \tilde{w}_{N(2)N(1)+1}^u]^T$ for $u = 1, \ldots, U$ and U is the number of class regions. Thus, we obtain a set of U class existence regions $\tilde{C}_c^u$, whose boundaries, parallel to the original separation hyperplanes, define the limits of the training data set within the specified class regions C_c^u.

According to the discussion in section 2.2, the new weights $\tilde{w}_{jN(1)+1}^u$ of the existence regions $\tilde{C}_c^u$ can be expressed by:

$$\tilde{w}_{jN(1)+1}^u = w_{jN(1)+1}^1 - s_j^u \qquad \text{for } u = 1, \ldots, U \text{ and } j = 1, \ldots, N(2), \tag{10}$$

where s_j^u is a shifting factor. In the following, we describe the algorithms for single-sided shifting and for double-sided shifting.

Single-Sided Shifting: Check all the patterns $\mathbf{p}^u$ of all classes c successively. For each pattern $\mathbf{p}^u$, check the values p_j^u, for $j = 1, \ldots, N(2)$. If p_j^u is not dc, the shifting factor is determined by

$$s_j^u = \begin{cases} \max\limits_{\mathbf{x}^m \text{in} C_c^u, m=1,\ldots,M} (\mathbf{w}_j^1 \mathbf{x}^m) & \text{for } p_j^u = 0 \\ \min\limits_{\mathbf{x}^m \text{in} C_c^u, m=1,\ldots,M} (\mathbf{w}_j^1 \mathbf{x}^m) & \text{for } p_j^u = 1 \end{cases} \tag{11}$$

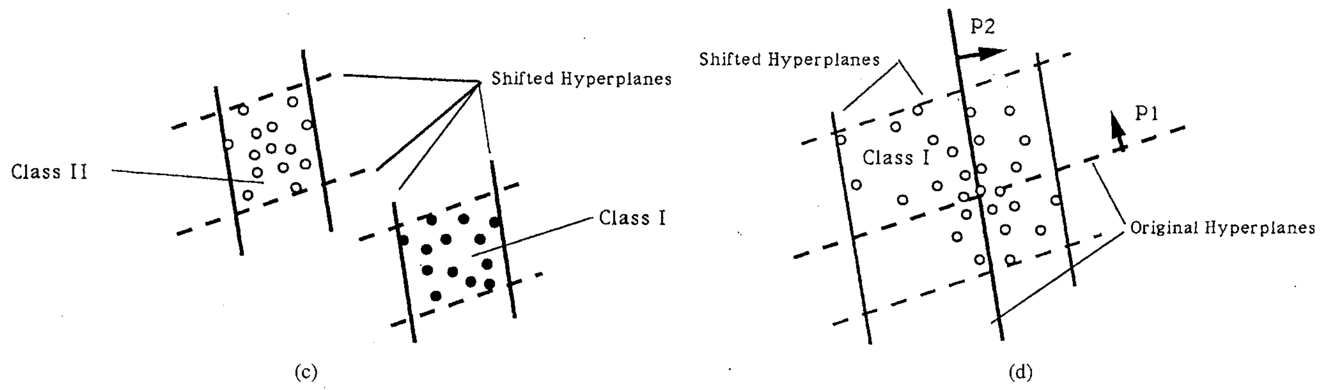


Fig. 4. (c) Double-sided Shifting of Separation Hyperplanes in a Two-dimensional Input Space ($p_j^u = 0, 1$). (d) Double-sided Shifting of Separation Hyperplanes in a Two-dimensional Input Space ($p_j^u = dc$).

where $\mathbf{w}_j^1\mathbf{x}^m$ is the scalar product between the j-th row of the weight matrix $\mathbf{W}^1$ and a training data vector $\mathbf{x}^m$ belonging to the region C_c^u. Thus, we obtain a shifting vector $\mathbf{s}^u = [s_1^u, \ldots, s_{N(2)}^u]^T$ for each pattern $\mathbf{p}^u$. The new weights $\tilde{w}_{jN(1)+1}^u$ of the shifted hyperplanes are calculated by (10). After single-sided shifting, the class regions are approximated more precisely by a set of shifted hyperplanes.

Double-Sided Shifting: Check all the patterns of all classes successively. For each pattern $\mathbf{p}^u$ calculate two shifting vectors $\mathbf{s}_{\min}^u = [s_{1,\min}^u, \ldots, s_{N(2),\min}^u]^T$ and $\mathbf{s}_{\max}^u = [s_{1,\max}^u, \ldots, s_{N(2),\max}^u]^T$, with the shifting factors

$$s_{j,\min}^u = \min_{\mathbf{x}^m \text{in} \mathrm{C}_c^u, m=1,\ldots,M} (\mathbf{w}_j^1\mathbf{x}^m) \tag{12}$$

and

$$s_{j,\max}^u = \max_{\mathbf{x}^m \text{in} \mathrm{C}_c^u, m=1,\ldots,M} (\mathbf{w}_j^1\mathbf{x}^m) \tag{13}$$

where $\mathbf{w}_j^1\mathbf{x}^m$ is the scalar product between the j-th row of the weight matrix $\mathbf{W}^1$ and a training data vector $\mathbf{x}^m$ belonging to region C_c^u. Thus, for each class region $\mathbf{p}^u$, we obtain two shifting vectors and two sets of shifted hyperplanes. Each class region is limited by a set of $N(2)$ hyperplanes in both directions of these hyperplanes. Using (10), (12) and (13) the weights $\tilde{w}_{j,\min}^u$ and $\tilde{w}_{j,\max}^u$, which correspond to $s_{j,\min}^u$ and $s_{j,\max}^u$, respectively, are calculated.

Because the generated regions are limited on both sides, double-sided shifting reduces overlapping. In general, resolving existing overlaps between different classes is difficult, because we do not have enough knowledge about the shifted hyperplanes and regions $\tilde{C}_c^u$.

IV. Fuzzy Rules Generation and Fuzzy Classification

The existence region $\tilde{C}_c^u$ for class c is defined as

$$\tilde{C}_c^u = \{\mathbf{x} \mid \tilde{\mathbf{w}}_j^u\mathbf{x} \geq 0 \text{ if } p_j^u = 1, \text{ and } \tilde{\mathbf{w}}_j^u\mathbf{x} \leq 0 \text{ if } p_j^u = 0, \\ \text{for } j = 1, \ldots, N(2) \text{ and } p_j^u \neq dc\}, \tag{14}$$

for single-sided shifting and

$$\tilde{C}_c^u = \{\mathbf{x} \mid \tilde{\mathbf{w}}_{j,\min}^u\mathbf{x} \geq 0 \cap \tilde{\mathbf{w}}_{j,\max}^u\mathbf{x} \leq 0 \text{ for } j = 1, \ldots, N(2)\} \tag{15}$$

for double-sided shifting, and $\tilde{\mathbf{w}}_j^u$ represents the j-th shifted hyperplane. Then we can define fuzzy rules FR_c^u as follows:

$$\mathrm{FR}_c^u : \text{ If } \mathbf{x} \text{ is in } \tilde{C}_c^u \text{ then } \mathbf{x} \text{ belongs to class } c, \tag{16}$$

where $u = 1, \ldots, U$.

Using membership functions, we can create more generalized regions in the input space by replacing the crisp boundaries of existence regions with fuzzy boundaries. Thus for every existence region $\tilde{C}_c^u$, we define a set of membership functions in the direction of the hyperplanes vectors. Membership functions for single- and double-sided shifting are different. For fuzzy reasoning, we calculate the degree of membership of the test datum for each rule, and the datum is classified as belonging to the class which has the highest degree of membership.

A. Definitions of Membership Functions and Fuzzy Regions

An intuitive assumption is that data, which reside within the crisp boundaries of the existence region $\tilde{C}_c^u$, should belong to the same class c with the degree of membership 1. As the datum location becomes farther away from the boundaries of the original class region, the degree of membership decreases and eventually reaches the minimum value of 0, where the distance between the test datum and the considered class region becomes so large that the datum becomes unlikely to belong to that class. Hence we can define membership functions $\mu(\mathbf{x}, p_j^u)$ of a region $\tilde{C}_c^u$ with the corresponding pattern $\mathbf{p}^u$ in the direction of the hyperplane vectors $\mathbf{w}_j$ by

$$\mu(\mathbf{x}, p_j^u) = \begin{cases} \min(1, \max(0, -\gamma_j^u\tilde{\mathbf{w}}_j^u\mathbf{x})) & \text{for } p_j^u = 0 \\ \min(1, \max(0, \gamma_j^u\tilde{\mathbf{w}}_j^u\mathbf{x})) & \text{for } p_j^u = 1 \\ 1 & \text{for } p_j^u = dc \end{cases} \tag{17}$$

for single-sided shifting, and

$$\mu(\mathbf{x}, p_j^u) = \min(1, \max(0, \gamma_{j,\min}^u\tilde{\mathbf{w}}_{j,\min}^u\mathbf{x})) \\ \times \min(1, \max(0, -\gamma_{j,\max}^u\tilde{\mathbf{w}}_{j,\max}^u\mathbf{x})) \tag{18}$$

for double-sided shifting, where γ_j^u, $\gamma_{j,\min}^u$ and $\gamma_{j,\max}^u$ are sensitivity parameters that control the steepness of the membership functions and thus, control the generalization ability of the fuzzy regions. Figs. 5(a) and 5(b) show membership functions for single- and double-sided shifting in the direction of the considered hyperplane vector $\mathbf{w}_j$, respectively.

B. Fuzzy Inference Operators

To calculate the degree of membership of a fuzzy rule we propose the minimum operator and the summation operator which are defined in the following. The final classification of a datum $\mathbf{x}$ is carried out by using the maximum operator, which selects the class, whose fuzzy rule has the highest degree of membership among all rules. We denote the combination of minimum and maximum operators for fuzzy reasoning as Min-Max inference and the combination of summation and maximum operators as Sum-Max inference.

Minimum Operator: The minimum operator [11], which takes the minimum value of all one-dimensional membership functions for the fuzzy rule FR_c^u, is given by:

$$\mu(\mathbf{x}, \mathbf{p}^u) = \min_{j=1,\ldots,N(2)} \mu(\mathbf{x}, p_j^u). \tag{19}$$

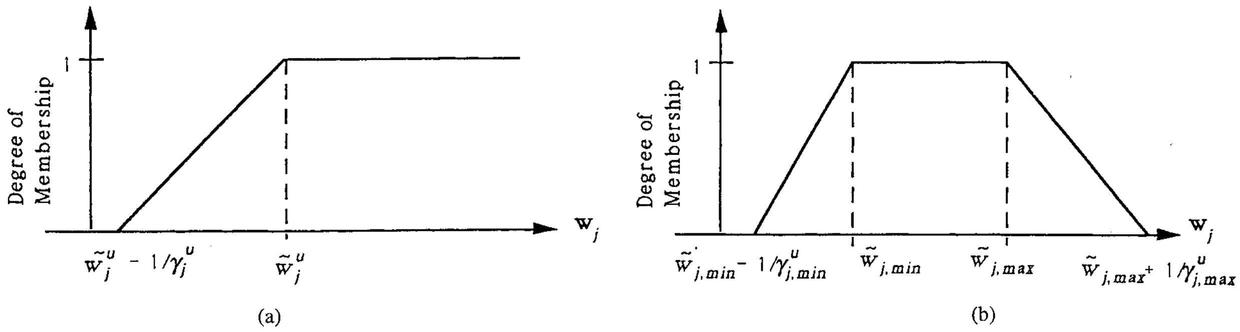


Fig. 5 (a) Membership Function in the Direction of the Hyperplane Vector $\mathbf{w}_j$ for Single-sided Shifting ($p_j^u = 1$). (b) Membership Function in the Direction of the Hyperplane Vector $\mathbf{w}_j$ for Double-sided Shifting.

The minimum value is taken so that the degree of membership within the class boundaries of the training data set becomes 1. The minimum operator selects the smallest degree of membership or, in other words, the largest distance of the test datum from any of the boundaries of the considered region $\tilde{C}_c^u$. When selecting the minimum degree of membership, we lose information on the distance of the test datum from the other boundaries of the considered region.

Summation Operator As a second operator to calculate the degree of membership of a fuzzy rule, summation of the one-dimensional degrees of membership is proposed. It imitates neural network based inference. Since a datum $\mathbf{x}$ is more likely to belong to a fuzzy region $\tilde{C}_c^u$ if its average distance is closer to all shifted hyperplanes of $\tilde{C}_c^u$, we define the summation operator by:

$$\mu(\mathbf{x}, \mathbf{p}^u) = \frac{1}{N(2)} \sum_{j=1}^{N(2)} \mu(\mathbf{x}, p_j^u). \qquad (20)$$

Maximum Operator With the maximum operator we select the fuzzy rule $\mathrm{FR}_c^{u_{\max}}$, whose degree of membership is the highest among all fuzzy rules for an input vector $\mathbf{x}$ according to

$$u_{\max} = \arg \max_{u=1,\ldots,U} \mu(\mathbf{x}, \mathbf{p}^u), \qquad (21)$$

where $\mu(\mathbf{x}, \mathbf{p}^u)$ represent the degrees of membership obtained by (19) or (20) and U is the number of fuzzy rules for all classes. Thus a datum $\mathbf{x}$ is classified as class c if the fuzzy rule $\mathrm{FR}_c^{u_{\max}}$ corresponds to that class. However, if two classes overlap and the test data resides in the overlapping region, it is impossible to classify this datum, since more than one class has the degree of membership 1. The same thing happens if two or more fuzzy rules have the highest membership or if all the fuzzy rules have the degree of membership 0. When the degree of membership is zero, the generalization ability may be enhanced by reducing the sensitivity parameters, which might help to classify this datum.

V. Performance Evaluation

We chose two classification systems to evaluate the performance of our method. The first system was a number recognition system which had very well-defined class regions. In contrast to that, the second system for blood cell classification had complicated regions with severe overlapping.

A. Number Recognition System

The original system for number recognition [7] of license plates used a decision-tree algorithm and recognized 10 numbers using 12 input features extracted from video images of moving cars. The numbers were printed, but distorted and covered with dirt. In our study a total of 1630 data were divided into a combination of 200 training data and 1430 test data. For performance comparison, we used a three-layered neural network with 12 input neurons plus a bias neuron and 10 output neurons. Since the performance of the network varies according to the initial weights, we trained 100 networks, using initial weights randomly assigned between -1 and 1. To train the networks a learning parameter of 0.3 and a momentum term [9] of 0.5 were used while learning. After learning, the separation hyperplanes were extracted from each network and fuzzy rules were created by applying single- and double-sided shifting to unclustered existence regions and regions clustered by the heuristic method and the neighboring regions clustering method. The recognition rates for the training and test data were measured for the original network obtained by the BP and the fuzzy-rule bases for the above-mentioned combinations of shifting and clustering for both Min-Max and Sum-Max reasoning. Since the minimum number of hyperplanes for separating 10 classes was 4, we varied the number of hidden neurons from 4 to 10. All measures were obtained with a training convergence criterion ε between 0.01 and 0.3. For each network we started the learning process with $\varepsilon = 0.3$. After the training converged, the fuzzy rules were created and the performance of the neural network and the fuzzy classifiers were evaluated. With the same network the process was reiterated, successively lowering ε from 0.3 to 0.01. The above-described procedure was carried out for 100 different initial networks with 8 hidden neurons using a 70 MIPS workstation. The work was performed for about 8 hours of CPU time, which resulted in an average of approximately 4.8 minutes for each network. The time required for generating fuzzy rules was less than 1 second on average.

The sensitivity parameter γ that determines the slope steepness of the membership functions was set to 0.2. With this value the degree of membership for all data scattered between 0.1 and 1. Trials with smaller γ-values showed similar results.

On average, for all the 6, 8 and 10 hidden-neuron cases, training converged after 41, 105, 212 and 467 epochs for $\varepsilon = 0.3$, $\varepsilon = 0.15$, $\varepsilon = 0.05$ and $\varepsilon = 0.01$, respectively; while for 4 hidden-neuron cases convergence was reached after 67, 213, 442 and 1021 epochs. For all the cases that we tried, the recognition rates for the neural networks and all fuzzy classifiers with double-sided shifting were 100% for the 200 training data, while for fuzzy classifiers with single-sided shifting, the recognition rate dropped due to overlapping of regions. For example, for 4 hidden-neuron networks we obtained three networks whose fuzzy classifiers could not recognize an entire class and thus the overall recognition rate for the training data set was 99.2%. In all other cases, the recognition rate was 99.9%.

Fig. 6 shows the average recognition rate for 1430 test data with $\varepsilon = 0.01$ applied to different numbers of hidden neurons. (In the following Figs., the term 's-shi' indicates single-sided shifting; 'd-

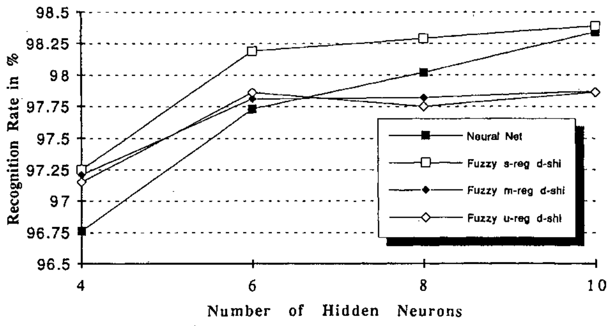


Fig. 6. Average Recognition Rate for 1430 Test Data vs. the Number of Hidden Neurons (Convergence Criterion ε = 0.01, Sum-Max Inference, Double-sided Shifting).

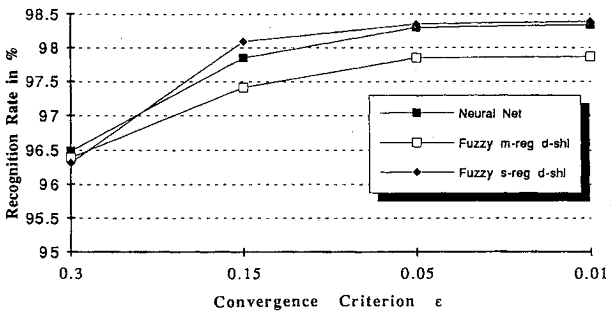


Fig. 7. Average Recognition Rate for 1430 Test Data vs. the Convergence Criterion (10 Hidden Neurons, Sum-Max Inference, Double-sided Shifting).

shi', double-sided shifting; 'u-reg', unclustered regions; 'm-reg', neighbored region clustering; and 's-reg' indicates the use of a single hyper region for every class, generated by the heuristic clustering method.) The fuzzy classifiers used Sum-Max inference. The advantage of using a larger number of hidden neurons for the neural networks and the fuzzy classifiers with regions clustered by the heuristic method can be seen in the figure. The performance of unclustered and neighboring regions clustered fuzzy classifiers reached a plateau at 6 hidden neurons and for higher numbers of hidden neurons their performances were inferior to the neural networks. The performance gap between the neural networks and the fuzzy classifiers dropped from 0.5% to 0.07% with increasing numbers of hidden neurons, although the number of fuzzy classifiers that performed better than the neural networks dropped only from a ratio of 60:40 to 57:43 for the regions clustered by the heuristic method. The total number of fuzzy rules for neighboring regions clustering increased steadily with the number of hidden neurons, e.g., from 10 to 24 for 4 to 10 hidden neurons. Since the performance of the classifiers with clustered neighboring regions was better than that of the classifiers with unclustered regions, in the following we show only the clustered neighboring regions.

Fig. 7 shows the influence of the convergence criteria on the performance using 1430 test data and the networks with 10 hidden neurons. The fuzzy classifiers used Sum-Max inference. In all cases smaller ε led to a better average performance. The trends for all the cases were very similar. Fuzzy classifiers using the heuristic clustering method outperformed the neural networks and the fuzzy classifiers using neighboring regions clustering. The total numbers of fuzzy rules decreased as the convergence criterion ε decreased. For example, when ε varied from 0.3 to 0.01, for 6 hidden-neuron cases, the number of rules varied from 30 to 23 for unclustered regions, while for neighboring regions clustering, it varied from 19 to 14.

Fig. 8 shows the recognition performance of the fuzzy classifiers with single- and double-side shifting for 1430 test data versus

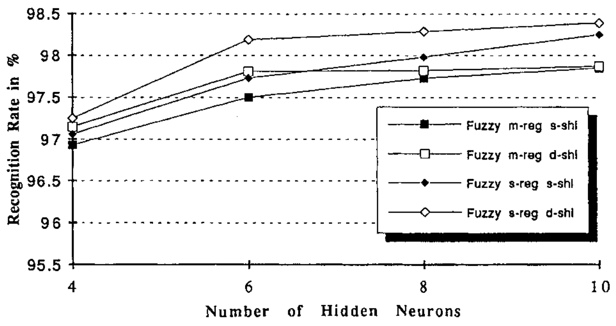


Fig. 8. Average Recognition Rate for 1430 Test Data vs. the Number of Hidden Neurons for Single- and Double-sided Shifting (Convergence Criterion $\varepsilon = 0.01$, Sum-Max Inference).

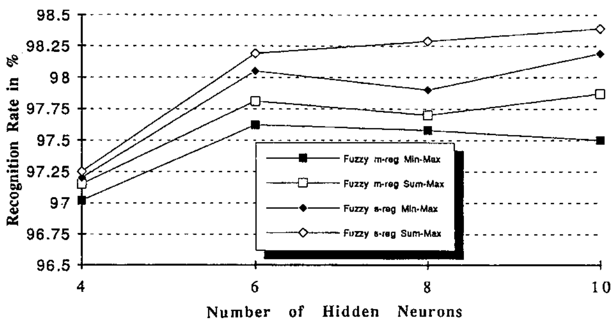


Fig. 9. Average Recognition Rate for 1430 Test Data vs. the Number of Hidden Neurons for Min-Max and Sum-Max Inference (Convergence Criterion $\varepsilon = 0.01$, Double-sided Shifting).

different numbers of hidden neurons. The classifiers used Sum-Max inference and were generated with a convergence criterion of 0.01. The classifiers with double-side shifting performed better than the ones with single-sided shifting, although the performance gap decreased with decreasing ε. The percentage of unclassified data due to overlapping for the single-sided shifting using networks with 6 hidden neurons was 0.019%, while that for double-sided shifting was 0%. In this case, the performance gap between the classifiers with single- and double-sided shifting was approximately 0.25% and thus overlapping could not be the main reason for the poor performance of the single-side shifted cases. Since the separation hyperplanes are defined during training, they are only rough estimates of regions where training data reside. Single-sided shifting extrapolates the class regions not covered by training data which results in higher misclassification rates. This can be understood by the fact that for single-sided shifting the degree of membership for the misclassified data is higher than that for double-sided shifting. For the number recognition system, the performance of the classifiers using the heuristic clustering method is superior to that of neighboring-region-clustered ones, because the heuristic clustering creates larger convex regions that also include regions where no training data reside and thus it has a greater generalization ability.

Fig. 9 shows the performance of classifiers using Min-Max and Sum-Max inference, respectively, for different numbers of hidden neurons. The fuzzy rules were generated using a convergence criterion of $\varepsilon = 0.01$. The results show that the Summation operator performs better than the minimum operator.

B. Classification of Blood Cells

The second application was a blood cell classification system [12], used to classify optically screened white blood cells into 12 categories of mature and immature cells using 13 features such as area and

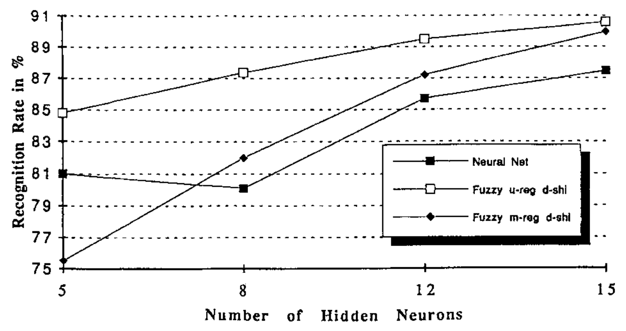


Fig. 10. Average Recognition Rate for 3100 Test Data vs. the Number of Hidden Neurons (15,000 Epochs,Sum-Max Inference).

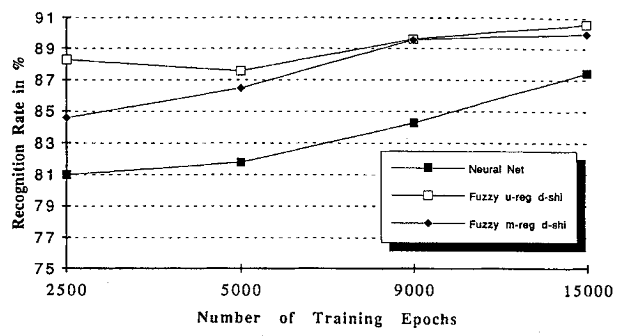


Fig. 11. Average Recognition Rate for 3100 Test Data vs. Number Training Epochs (15 Hidden Neurons, Sum-Max Inference).

perimeter of a kernel. For evaluation, we divided a set of 6197 input-target data into a learning data set of 3097 input-target pairs and a test data set of 3100 input-target pairs. Meanwhile we used three-layered neural networks with 13 input neurons plus a bias neuron, different numbers of hidden neurons and 12 output neurons. We varied the numbers of hidden neurons between 5 and 15, plus a bias neuron, and trained each type for 25 times using initial weights randomly assigned between -1 and 1. The learning rate and the momentum term were initially set to 0.3 and 0.5, respectively, and the learning rate was lowered in three steps to 0.05. Training was difficult and tests showed that even for 50,000 epochs and $\varepsilon = 0.3$ we could not obtain convergence, although the summed square error decreased. So we set ε to 0.3 for all further trials. Due to the difficult training, the recognition rate for the training data set after learning scattered between 80.7% and 92.4% for all trials of the neural networks. Because of severe overlapping, the performance of the fuzzy classifiers ranged between 86.2% and 95.7%. We set γ to 0.1 for the fuzzy classifiers.

First we trained the networks with 2,500 training epochs and created fuzzy rules. Then we evaluated the performance of the classifiers. The already trained networks were used to reiterate this procedure three times with 5,000, 9,000 and 15,000 training epochs. The whole procedure took 53 hours of CPU time for networks with 15 hidden neurons on a 70 MIPS workstation and thus about 2.15 hours for each network. The time to create fuzzy rules was 7 seconds on average.

The terms used in the Fig. legends are the same as those used for the number recognition system. The performance of the fuzzy classifiers using the heuristic clustered regions is not shown, since their performance was very poor (around 75% recognition rate) due to severe overlapping.

Fig. 10 shows the average recognition rate for 3100 test data for networks of different numbers of hidden neurons trained for 15,000 epochs. Sum-Max inference was used. The average number of rules created was between 122 and 302 for unclustered regions and between 53 and 152 for neighboring region clustered cases. The number of rules increased with the number of hidden neurons and with the number of training epochs. In all cases, the fuzzy classifiers outperformed the neural network by an average of 2% to 7.5%. The best neural network had a recognition rate of 90.46%, while the recognition rate of the best fuzzy classifier was 91.68%. Due to overlappping, the performance of the classifier using clustered regions was worse than that using unclustered regions. For higher numbers of training epochs and higher numbers of hidden neurons the class regions could be defined more precisely; this was indicated by the increasing numbers of created rules, and thus the performance of the clustered regions approached that of the unclustered regions. For example, using 15 hidden neurons and 15,000 training epochs, on

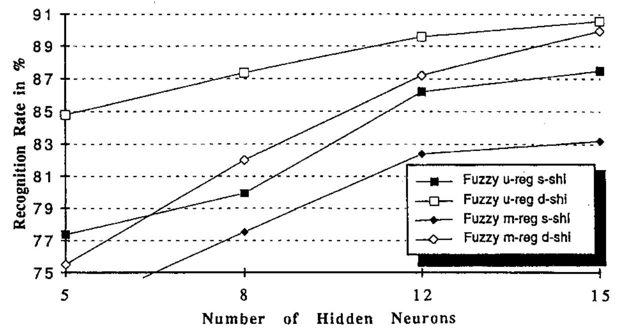


Fig. 12. Average Recognition Rate for 3100 Test Data vs. the Number of Hidden Neurons for Single- and Double-sided Shifting (15,000 Epochs, Sum-Max Inference).

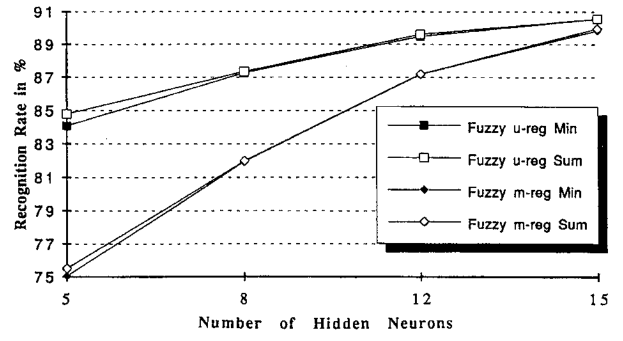


Fig. 13. Average Recognition Rate for 3100 Test Data vs. the Number of Hidden Neurons for Min-Max and Sum-Max Inference (15,000 Epochs, Double-sided Shifting).

average 302 unclustered rules were created, while the number was reduced to 179 when clustering was applied.

Fig. 11 shows the influence of the number of training epochs on the performance of the classifiers for 3100 test data. At epoch 2500, both fuzzy classifiers achieved the performance of the neural network which was trained for 15,000 epochs. Thus we concluded that if we use the fuzzy classifiers described in this paper, we can shorten training time for the neural network.

Fig. 12 shows fuzzy classifiers with unclustered and neighboring-clustered regions for single- and double-sided shifting based on networks trained 15,000 epochs. The performance of classifiers using single-sided shifting was worse than that of the classifiers using double-sided shifting due to severe overlapping.

Fig. 13 shows the performance of the fuzzy classifiers which used Min-Max and Sum-Max inference for 3100 test data. Although the

TABLE I
PERFORMANCE OF THE BEST NEURAL NETWORK AND BEST FUZZY CLASSIFIER

	Neural Network	Fuzzy Classifier
overall performance	89.99%	91.97%
correct mature cell class	99.20%	99.49%
correct immature cell class	95.47%	96.00%

recognition rates of the Sum-Max method were slightly higher, the recognition rates of both methods did not differ significantly.

For this system, besides the overall performance, two characteristic values were also compared: the percentage of correctly classified mature cells against detected mature cells and the percentage of detected immature cells against total immature cells. From all trials, we found that a three-layered neural network using 10 hidden neurons and trained for 6000 epochs performed the best. Table I summarizes the performance of this network and that of the fuzzy classifier derived from it.

VI. DISCUSSION

The separation hyperplanes extracted from the pattern classification neural network can be used as boundaries of the class regions, and the fuzzy classifiers have the potential to outperform the original neural networks. The performance of fuzzy classifiers based on double-sided shifting is superior to that of those based on single-sided shifting due to reduced overlapping. Both Sum-Max and Min-Max inference methods are applicable, while Sum-Max inference shows better results on average. The choice of a suitable clustering method is essential to the performance of the fuzzy classifiers.

The advantages of the proposed fuzzy classification system over the neural networks can be summarized as follows.

- The implementation of the fuzzy classifier is easy.
- The fuzzy classifier outperforms the neural network.
- The training time can be shortened while maintaining the same performance.
- Misclassification can be analyzed by checking the degree of membership of the fuzzy rules.
- The generalization ability can be easily controlled by modifying the sensitivity parameters γ. If a test datum is in a region not covered by class existence regions, the system can determine that this datum cannot be classified.
- Class boundaries can easily be adjusted, without retraining the neural network.
- Additional fuzzy rules can be added easily, without retraining.

A disadvantage of the fuzzy classifier over the neural network is the following.

- For complicated shapes of class regions a lot of fuzzy rules are generated, slowing down the inference speed.

The advantages over conventional fuzzy systems can be summarized as follows.

- Fuzzy rules are generated automatically using the learning ability of neural networks.
- Since the boundaries of the fuzzy regions need not be parallel to the input variables, separation of two classes can be defined more precisely and that also requires fewer rules.

A disadvantage over conventional fuzzy systems is the following.

- Analyzing and modifying the existence regions are difficult tasks, because the boundaries between the class regions are not parallel to the input variables.

VII. CONCLUSION

In this paper, we developed a method for extracting fuzzy rules from trained pattern classification neural networks. The fuzzy rules are represented by variable fuzzy existence regions of the classes in the input space, of which boundaries are formed by separation hyperplanes extracted from the input-to-hidden weights of the neural networks. The existence regions are clustered and the resulting boundaries are shifted in parallel to define the class regions more precisely. By replacing the crisp boundaries by membership functions, we obtained fuzzy regions in the input space that represent fuzzy rules, which can be used with common fuzzy inference techniques to classify input data without the use of the neural network. The method described in this paper was compared with use of neural networks for a license plate recognition system and a blood cell classification task.

ACKNOWLEDGMENT

We are very grateful to Professor N. Matsuda of Kawasaki Medical School for providing the blood cell data used in our study.

REFERENCES

[1] L. A. Zadeh, "Fuzzy sets," *Information Control*, vol. 8 pp. 338–353, 1965.
[2] S. K. Halgamuge and M. Glesner, "A fuzzy-neural approach for pattern classification with generation of rules based on supervised learning," *Proc. Neuro Nimes 92*, pp. 165–173, Nimes, 1992.
[3] I. Enbutsu, K. Baba, and N. Hara, "Fuzzy rule extraction from multi-layered neural networks," *Proc. IJCNN-91*, Seattle, vol. 2, pp. 691–695, 1991.
[4] C.-T. Lin and C. S. G. Lee, "Neural-network-based fuzzy logic control and decision system," *IEEE Trans. Computers*, vol. 40, pp. 1320–1336, 1991.
[5] J. J. Buckley, Y. Hayashi, and E. Czogala, "On the equivalence of neural networks and fuzzy expert systems," *Proc. IJCNN-92*, Baltimore, vol. 2, pp. 691–695, 1992.
[6] D. E. Rumelhart *et al.*, *Learning Internal Representations by Error Backpropagation, Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, Vol. 1: Foundations. Cambridge, MA: MIT Press, 1986, pp. 318–362.
[7] S. Abe, M. Kayama, H. Takenaga, and T. Kitamura, "Extracting algorithms from pattern classification neural networks," *Neural Networks*, vol. 6, no. 5, pp. 729–735, 1993.
[8] R. P. Lippmann, "An introduction to computing with neural nets," *Proc. IEEE ASSP Magazine*, vol. 4, pp. 4–22, 1987.
[9] J. Hertz, A. Krogh, and R. Palmer, *Introduction to the Theory of Neural Computation*. Addison-Wesley Publishing Company, 1991.
[10] S. Abe, M. Kayama, and H. Takenaga, "How neural networks for pattern recognition can be synthesized," *J. Information Processing*, vol. 14, no. 3 pp. 344–350, 1991.
[11] J. C. Bezdek and S. K. Pal, *Fuzzy Models for Pattern Recognition*. New York: IEEE Press, 1992.
[12] A. Hashizume, J. Motoika, and R. Yabe, "Fully automated blood cell differential system and its application," *Proc. IUPAC 3rd Int. Cong. Automat. and New Technol. in the Clinical Laboratory*, pp. 297–302, Sept. 1988.