Kobe University Repository : Kernel

PDF issue: 2025-12-05

Dynamic cluster generation for a fuzzy classifier with ellipsoidal regions

Abe, Shigeo

(Citation)
IEEE transactions on systems, man and cybernetics. Part B, Cybernetics,28(6):869-876

(Issue Date)
1998-12

(Resource Type)
journal article

(Version)
Version of Record

(URL)
https://hdl.handle.net/20.500.14094/90000219

KOBE UNIVERSITY

Dynamic Cluster Generation for a Fuzzy Classifier with Ellipsoidal Regions

Shigeo Abe

Abstract—**In this paper, we discuss a fuzzy classifier with ellipsoidal regions that dynamically generates clusters. First, for the data belonging to a class we define a fuzzy rule with an ellipsoidal region. Namely, using the training data for each class, we calculate the center and the covariance matrix of the ellipsoidal region for the class. Then we tune the fuzzy rules, i.e., the slopes of the membership functions, successively until there is no improvement in the recognition rate of the training data. Then if the number of the data belonging to a class that are misclassified into another class exceeds a prescribed number, we define a new cluster to which those data belong and the associated fuzzy rule. Then we tune the newly defined fuzzy rules in the similar way as stated above, fixing the already obtained fuzzy rules. We iterate generation of clusters and tuning of the newly generated fuzzy rules until the number of the data belonging to a class that are misclassified into another class does not exceed the prescribed number. We evaluate our method using thyroid data, Japanese Hiragana data of vehicle license plates, and blood cell data. By dynamic cluster generation, the generalization ability of the classifier is improved and the recognition rate of the fuzzy classifier for the test data is the best among the neural network classifiers and other fuzzy classifiers if there are no discrete input variables.**

Index Terms— **Blood cell data, cluster generation, fuzzy classifiers, license plate recognition, membership function, neural networks, rule extraction, thyroid data, tuning.**

I. Introduction

To facilitate analysis of classifiers as well as to accelerate training of classifiers, many types of fuzzy classifiers with a learning capability have been proposed [1]–[6]. We have developed three types of fuzzy classifiers: 1) a fuzzy classifier with hyperbox regions whose surfaces are parallel to one of the input variables [2]; 2) a fuzzy classifier with polyhedron regions whose surfaces are expressed by a linear combination of input variables [3]; and 3) a fuzzy classifier with ellipsoidal regions [6].

Training time of the fuzzy classifier with hyperbox regions is very short because we need only to calculate the minimum and maximum values of the training data in each input variable, and if the principal axes of the training data distribution for each class are parallel to input variables, we can obtain a good generalization ability. But if the principal axes are not parallel, the generalization ability of the classifier is not so good compared with that of neural network classifiers. The fuzzy classifier with polyhedron regions does not have this problem since the polyhedron regions are approximated by shifting the hyperplanes extracted from the trained neural network classifier. But since the fuzzy classifier is based on the neural network classifier, training is slow. To overcome this we have developed a fuzzy classifier with ellipsoidal regions. According to performance evaluation of four benchmark data sets, i.e., an iris data set, a numeral data set, a blood cell data set, and a thyroid data set, the recognition rates of the test data for the first three data sets were comparable to or better than the maximum recognition rate of the neural network classifier and the fuzzy classifiers with hyperbox regions and polyhedron regions, although only one fuzzy rule was defined for each class. But for the thyroid data, the fuzzy classifier with ellipsoidal regions performed poorly even if more than one fuzzy rule were defined for each class. The reasons of poor performance were that most of the input variables were discrete and class data overlapped heavily and appropriate clustering was not possible before training.

In this paper, to overcome the difficulty of clustering, we discuss a fuzzy classifier with ellipsoidal regions that dynamically generates clusters. In [6], to improve the recognition rate, the training data were clustered in advance. Here, instead, clustering is postponed after the training is completed. Namely, first for the data belonging to a class we define one fuzzy rule with an ellipsoidal region, by calculating the center and the covariance matrix of the ellipsoidal region for the class. Then we tune the fuzzy rules, i.e., the slopes of the membership functions, successively until there is no improvement in the recognition rate of the training data. Then, if the number of the data belonging to a class that are misclassified into another class exceeds a prescribed number, we define a new cluster to which those data belong. Then we tune the newly defined fuzzy rules in the similar way as stated above, fixing the already obtained fuzzy rules. We iterate cluster generation and tuning of newly generated fuzzy rules until the number of the data belonging to a class that are misclassified into another class does not exceed the prescribed number.

In Section II, we describe the classifier architecture which consists of two layers. In Section III, we describe dynamic cluster generation in which a cluster is generated during training if the number of the data belonging to a class that are misclassified into another class exceeds a prescribed number. In Section III, using the thyroid data, Japanese Hiragana data for vehicle license plate recognition, and the blood cell data, we compare the performance of the proposed classifier with that of other fuzzy classifiers and the neural network classifier.

II. Classifier Architecture

Consider classification of an m dimensional input vector $\mathbf{x}$ into n classes. Assume that the training data for class i $(i = 1, \cdots, n)$ are divided into several clusters ij $(j = 1, \cdots)$ where cluster ij denotes the jth cluster for class i. For each cluster ij, we define the following fuzzy rule:

$$R_{ij}: \quad \text{if } \mathbf{x} \text{ is } \mathbf{c}_{ij} \text{ then } \mathbf{x} \text{ belongs to class } i \tag{1}$$

where $\mathbf{c}_{ij}$ is the center of cluster ij and is calculated by using the training data belonging to cluster ij:

$$c_{ij,k} = \frac{1}{N_{ij}} \sum_{\mathbf{x} \in \text{ cluster } ij} x_k \tag{2}$$

where N_{ij} is the number of the data belonging to cluster ij. The membership function $m_{ij}(\mathbf{x})$ of (1) for input $\mathbf{x}$ is given by

$$m_{ij}(\mathbf{x}) = \exp(-h_{ij}^2(\mathbf{x})) \tag{3}$$

$$h_{ij}^2(\mathbf{x}) = \frac{d_{ij}^2(\mathbf{x})}{\alpha_{ij}} \tag{4}$$

$$d_{ij}^2(\mathbf{x}) = (\mathbf{x} - \mathbf{c}_{ij})^t Q_{ij}^{-1} (\mathbf{x} - \mathbf{c}_{ij}) \tag{5}$$

where $d_{ij}(\mathbf{x})$ is the weighted distance between $\mathbf{x}$ and $\mathbf{c}_{ij} = (c_{ij,1}, \cdots, c_{ij,m})^t$, $h_{ij}(\mathbf{x})$ is the tuned distance, $\alpha_{ij}(>0)$ is a tuning parameter for cluster ij, Q_{ij} is the $m \times m$ covariance matrix of cluster ij, the superscript t denotes the transpose of a matrix,

Manuscript received January 6, 1997; revised April 10, 1998.

The author is with the Department of Electrical and Electronics Engineering, Kobe University, Kobe, Japan (e-mail: abe@eedept.kobe-u.ac.jp).

Publisher Item Identifier S 1083-4419(98)09249-8.

1083–4419/98$10.00 © 1998 IEEE

and the superscript -1 denotes the inverse of a matrix. Then we calculate the covariance matrix Q_{ij} by

$$Q_{ij} = \frac{1}{N_{ij}} \sum_{\mathbf{x} \in \text{ cluster } ij} (\mathbf{x} - \mathbf{c}_{ij})(\mathbf{x} - \mathbf{c}_{ij})^t. \quad (6)$$

If the covariance matrix Q_{ij} is singular, we set all the off diagonal elements of Q_{ij} to zero so that Q_{ij} becomes regular. By making the covariance matrix diagonal, the principal axes of the associated ellipsoidal region are parallel to the input axes.

For input $\mathbf{x}$ if the membership function $m_{kl}(\mathbf{x})$ is the largest, input $\mathbf{x}$ is classified into class k. The exponential function in (3) makes the output range of (3) lie in [0, 1]. Thus, if we classify input $\mathbf{x}$ using the input of the exponential function in (3), we need to find the smallest $h_{ij}(\mathbf{x})$. This is the simplest architecture that is conceivable. If we add a layer to the output of the above architecture, we can obtain the radial basis function network [4]. The centers and the covariance matrices are determined by unsupervised competitive learning [7], by the steepest descent method [8], or estimated by the Gram–Schmidt orthogonalization [4]. Here, we determine them using (2) and (6). These are good estimates if the training data obey the Gaussian distributions. To improve generalization ability when the training data do not obey the Gaussian distributions, we introduce the tuning parameter α_{ij} that resolves overlaps of ellipsoidal regions belonging to different classes [6]. An increase of α_{ij} decreases the slope of the membership function $m_{ij}(\mathbf{x})$ or increases the value of $m_{ij}(\mathbf{x})$. And a decrease of α_{ij} increases the slope of $m_{ij}(\mathbf{x})$ or decreases the value of $m_{ij}(\mathbf{x})$. The sophisticated tuning algorithm of the tuning parameter α_{ij} developed in [6] is summarized in Section III and the Appendix.

The other factor that affects the generalization ability of the fuzzy classifier with ellipsoidal regions is how to divide the training data into clusters. If the training data are divided into clusters with small numbers of data, good generalization ability of the classifier is not expected because of overfitting. In [6], we discussed a simplified algorithm to divide the training data, but we found that the trial and error was necessary to optimize the division. Also, the four benchmark data set evaluated in [6], clustering was necessary only for the thyroid data set. This means that whether clustering is necessary or not is judged best after evaluating the recognition rate of the training data for a fuzzy classifier with one fuzzy rule for each class. This leads to dynamic cluster generation, in which first the classifier is generated without clustering the training date and then if the number of the data belonging to a class that are misclassified into another class exceeds the specified number, a cluster is defined for those misclassified data. In the following section, we discuss dynamic cluster generation.

III. Dynamic Cluster Generation

A. Concept

Using two-dimensional training data for classes i and j shown in Fig. 1, we explain the concept of dynamic cluster generation. First, we approximate each class region by an ellipsoid with the center and the covariance matrix calculated using (2) and (6), respectively. Let Fig. 1(a) show the result and the ellipsoids in the figure have the same degree of membership. Then the class boundary of classes i and j becomes as shown in the figure and the seven data in the overlapping region of the two ellipsoids are misclassified.

Since we determine the centers and the covariance matrices without considering the overlapping of class regions, we can improve the recognition rate of the training data by tuning the tuning parameters

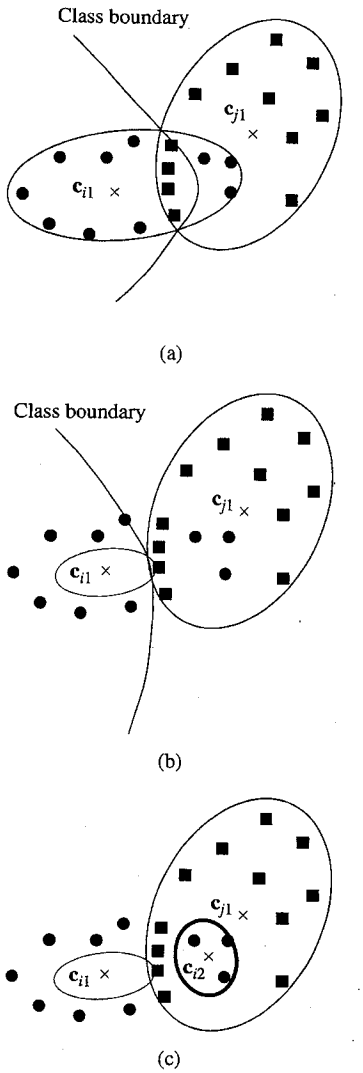

Fig. 1. Concept of dynamic cluster generation. (a) Define a rule for each class, (b) decrease α_{il} to improve the recognition rate, and (c) generate a cluster for class j.

α_{i1} and α_{j1}. If we increase α_{i1}, the degree of membership for class i increases. Thus we can make the three data belonging to class i in the overlapping region be correctly classified, while the four misclassified data remain misclassified. Or if we decrease α_{i1}, the degrees of membership for class i decreases. Thus we can make the four data belonging to class j in the overlapping region be correctly classified, while the three misclassified data remain misclassified. We can do the same thing by increasing or decreasing α_{j1}. Thus, the maximum recognition rate is obtained if we decrease α_{i1} or increase α_{j1}. Fig. 1(b) shows the case when α_{i1} is decreased.

After tuning, the three data belonging to class i that are in the ellipsoid for class j are misclassified. These data can be correctly classified if we define a cluster for these data as shown in Fig. 1(c). But if we define a cluster for a small number of training data, the generalization ability becomes poor because of overfitting. Therefore, we introduce a minimum number of the training data belonging to a class that are misclassified into another class, N_c. If the number of the data belonging to a class that are misclassified into another class is larger than or equal to N_c, we define a cluster and calculate the center and the covariance matrix using (2) and (6), respectively. In the following we describe the general flow of dynamic cluster generation and fuzzy rule tuning.

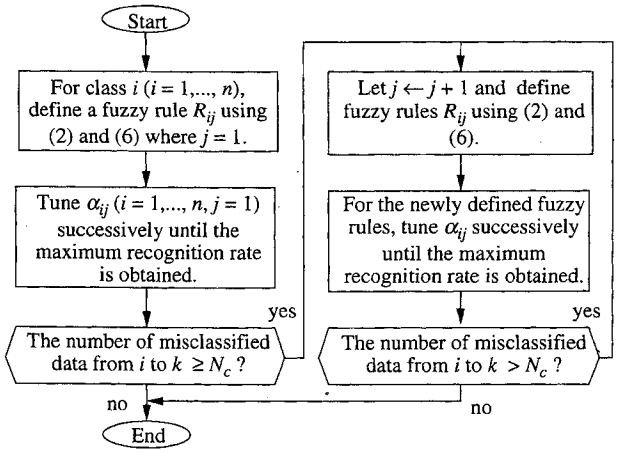

Fig. 2. Flow chart of dynamic cluster generation.

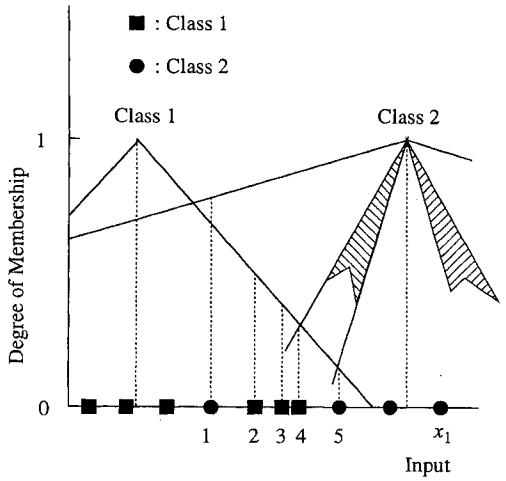

Fig. 3. Concept of tuning.

B. General Flow

The general flow of dynamic cluster generation is as follows (see Fig. 2).

1) Assume that each class consists of one cluster, calculate the centers and the covariance matrixes for classes i ($i = 1, \cdots, n$) using (2) and (6), respectively, and define fuzzy rules given by (1).
2) Let the maximum allowable number of misclassified data that are previously correctly classified be $l_M - 1$. Increase or decrease α_{i1} ($i = 1, \cdots, n$) successively so that the recognition rate of the training data is maximized, allowing $l_M - 1$ correctly classified data to be misclassified.
3) If the number of the data belonging to a class that are misclassified into another class is larger than or equal to N_c, and these misclassified data are not previously defined as a cluster, define a new cluster that includes only these misclassified data. Then calculate the centers and the covariance matrixes of the newly defined clusters using (2) and (6), respectively, and define fuzzy rules given by (1). Otherwise, terminate the algorithm.
4) Tune the tuning parameters of the newly defined fuzzy rules successively, fixing the already tuned tuning parameters. The tuning procedure is the same as in Step 2. Go to Step 3.

In Step 3, we check whether the training data belonging to a class that are misclassified into another class are previously defined as a cluster. The reason is as follows. If the recognition rate is not improved by the newly defined cluster, the same misclassified data remain. Thus, if we do not check this situation, a new cluster will be defined indefinitely.

Tuning of α_{ij} in Steps 2 and 4 is done if the recognition rate of the training data is improved. In addition, in Step 4, we fix the tuning parameters that are already tuned. Thus by our dynamic cluster generation, the recognition rate of the training data is monotonically improved. But this does not guarantee that the recognition rate of the test data is also monotonically improved.

C. Fuzzy Rule Tuning

Tuning of the tuning parameters α_{ij} is discussed in [6]. Here, we summarize the procedure of tuning and we discuss the detailed procedure in the Appendix. To explain the concept of tuning, we consider a two-class case with one rule for each class as shown in Fig. 3. (In the figure, instead of the Gaussian function, we use the triangular function as the membership function.) Datum 1 is correctly classified into class 2, while data 2–4 are misclassified into class 2. If we increase α_{11} or decrease α_{21}, datum 1 is first misclassified, but

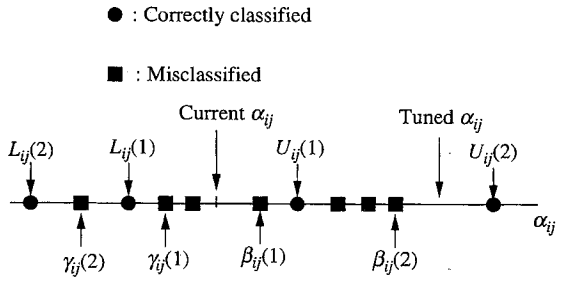

Fig. 4. Determination of tuned α_{ij}. If the current α_{ij} is modified to the tuned α_{ij} in $(\beta_{ij}(2), U_{ij}(2))$, one correctly classified datum is misclassified but four misclassified data are correctly classified.

if we allow datum 1 to be misclassified we can make data 2–4 be correctly classified. Fig. 3 shows this when α_{21} is decreased so that the degree of membership for class 2 lies between the shaded regions. Then by allowing one datum to be misclassified, three data are correctly classified, i.e., the recognition rate is improved by two data.

Now suppose we tune the tuning parameter α_{ij}. Up to some value we can increase or decrease α_{ij} without making the correctly classified data belonging to class i be misclassified. Now let $U_{ij}(1)$ and $L_{ij}(1)$ denote the upper and lower bounds that do not make the correctly classified data be misclassified, respectively. Likewise, $U_{ij}(l)$ and $L_{ij}(l)$ denote the upper and lower bounds in which $l-1$ correctly classified data are misclassified, respectively. Then, for instance, if we set a value in the interval $[U_{ij}(1), U_{ij}(2))$ to α_{ij}, one correctly classified datum belonging to class i is misclassified, where $[a, b]$ and (a, b) denote the closed and open intervals, respectively.

Similarly, if we increase or decrease α_{ij}, misclassified data may be correctly classified. Let $\beta_{ij}(l)$ denote the upper bound of α_{ij} that is smaller than $U_{ij}(l)$ and that makes the previously misclassified data be correctly classified. And $\gamma_{ij}(l)$ denotes the lower bound of α_{ij} that is larger than $L_{ij}(l)$ and that makes the previously misclassified data be correctly classified. Fig. 4 shows an example. If we change the current α_{ij} to the tuned α_{ij} in $(\beta_{ij}(2), U_{ij}(2))$, one correctly classified datum is misclassified but four misclassified data are correctly classified.

Then, the next task is to find which interval among $(L_{ij}(l), \gamma_{ij}(l))$ and $(\beta_{ij}(l), U_{ij}(l))$ ($l = 1, \cdots$) gives the maximum recognition rate. To limit the search space, we introduce the maximum l, i.e., l_M. Let $(L_{ij}(l), \gamma_{ij}(l))$ be the interval that gives the maximum recognition rate of the training data among $(L_{ij}(k), \gamma_{ij}(k))$ and $(\beta_{ij}(k), U_{ij}(k))$

TABLE I
BENCHMARK DATA SPECIFICATIONS AND TRAINING CONDITIONS OF THE THREE-LAYERED NEURAL NETWORK CLASSIFIER

	Thyroid	Hiragana	Blood Cell
No. Inputs	21	13	13
No. Classes	3	38	12
No. Training Data	3772	8375	3097
No. Test Data	3428	8356	3100
No. Hidden Units	3	25	15
No. Epochs	10000	10000	15000
No. Runs	10	1	25

for $k = 1, \cdots, l_M$. Then even if we set any value in the interval to α_{ij}, the recognition rate of the training data does not change but the recognition rate of the test data may change. To control the generalization ability, we set α_{ij} as follows:

$$\alpha_{ij} = \beta_{ij}(l) + \delta(U_{ij}(l) - \beta_{ij}(l)) \tag{7}$$

for $(\beta_{ij}(l), U_{ij}(l))$, where δ satisfies $0 < \delta < 1$ and

$$\alpha_{ij} = \gamma_{ij}(l) - \delta(\gamma_{ij}(l) - L_{ij}(l)) \tag{8}$$

for $(L_{ij}(l), \gamma_{ij}(l))$.

According to the above discussion, the tuning algorithm becomes as follows.

1) Set a positive number to parameter l_M, where $l_M - 1$ is the maximum number of misclassifications allowed for tuning α_{ij}, set a value in (0, 1) to δ in (7) and (8), and set the same positive initial value (usually 1) to α_{ij}.
2) For α_{ij} $(i = 1, \cdots, n, j = 1, \cdots)$, calculate $L_{ij}(l)$, $U_{ij}(l)$, $\beta_{ij}(l)$, and $\gamma_{ij}(l)$ for $l = 1, \cdots, l_M$. Find the interval $(L_{ij}(l), \gamma_{ij}(l))$ or $(\beta_{ij}(l), U_{ij}(l))$ that realizes the maximum recognition rate of the training data, and change α_{ij} using (7) or (8).
3) Iterate Step 2 until there is no improvement in the recognition rate of the training data.

Usually $l_M = 10$ is sufficient [6]. According to our experiments [6], the value of δ did not affect the recognition rate of the test data significantly, but a small value of δ sometimes gave a better recognition rate of the test data. Thus in the experiments in Section IV, we use 0.1. The detailed tuning algorithm is discussed in the Appendix.

We call the update of all α_{ij} $(i = 1, \cdots, n, j = 1, \cdots)$ one iteration of tuning, and if there is no improvement in the recognition rate for the two consecutive iterations, or the recognition rate of the training data reaches 100%, we stop tuning. Our tuning algorithm determines, for each fuzzy rule R_{ij}, the optimum tuning parameter α_{ij}, allowing the data that are correctly classified before tuning R_{ij} to become misclassified after tuning R_{ij} as long as the recognition rate of the training data is improved. To allow the data that are correctly classified before tuning some fuzzy rule to be misclassified after tuning that fuzzy rule is, so to speak, to prevent the tuning process from leading to convergence to a local minimum. But of course, since the tuning process is nonlinear, we cannot guarantee that this method always gives the optimal solution.

IV. PERFORMANCE EVALUATION

We evaluated the performance of the fuzzy classifier with ellipsoidal regions using the thyroid data [9], the hiragana data for license plate recognition [10], and the blood cell data [11], and compared the performance with that of the multilayered neural network classifier, the fuzzy classifier with hyperbox regions [2], and the fuzzy classifier with polyhedron regions [3]. The upper part of Table I shows the benchmark data specifications. And the lower part of the table shows the training conditions of the neural network classifier. We used the backpropagation algorithm to train the three-layered neural network classifier with the learning rate of 1.0 and the momentum coefficient of 0. For the thyroid data and the blood cell data, we trained the neural network classifier ten and 25 times, respectively, changing the initial weights and calculated the average recognition rates. But since it took eight hours to train the neural network classifier for the Hiragana data, we trained the network only once.

TABLE II
PERFORMANCE FOR THE THYROID DATA

Classifier	Rate (%)	No. Rules	Time (s)
N.N.	98.00 (99.02)	3 units	61 min
Hyperbox	99.15 (100)	10	0.7
Ellipsoid	86.41 (86.77) → 97.29 (99.02)	13	133

(): Recognition rate of the training data

TABLE III
PERFORMANCE OF THE FUZZY CLASSIFIER WITH ELLIPSOIDAL REGIONS FOR THE THYROID DATA BY CHANGING N_c

N_c	Rate (%)	No. Rules	Time (s)
–	95.60 (96.02)	3	25
40	97.00 (98.12)	5	36
30	97.00 (98.12)	5	36
20	97.17 (98.67)	10	95
10	97.29 (99.02)	13	133

(): Recognition rate of the training data

Unless otherwise stated, for the fuzzy classifier with ellipsoidal regions, we set $\alpha_{ij} = 1$, $\delta = 0.1$, and $l_M = 10$ [6].

For evaluation of the fuzzy classifier with hyperbox regions, we used a 16 MIPS workstation and for evaluation of the fuzzy classifier with polyhedron regions we used a 70 MIPS workstation. For all other evaluations, we used a 60-MIPS mainframe computer, and the calculation times listed in the following tables are the CPU times.

A. Thyroid Data

The thyroid data include 15 digital features and are not suited for the fuzzy classifier with ellipsoidal regions [6]. Table II shows the results for the thyroid data by the neural network classifier and the fuzzy classifiers with hyperbox regions and ellipsoidal regions. The recognition rates of the neural network classifier are the average values of ten runs. The maximum and minimum recognition rates of the test data (training data) were 98.48% (99.36%) and 97.78% (98.54%), respectively. For the "no. Rules" column of the neural network classifier, we list the number of hidden units. For the fuzzy classifier with hyperbox regions the expansion parameter, which controls the generalization ability, was set to 0.01. The recognition rates of both the training data and the test data were the best. The initial recognition rates of the fuzzy classifier with ellipsoidal regions is listed on the left hand side of the arrow and the final recognition rate is listed on the right hand side of the arrow. The parameter N_c was set to ten. The recognition rates were drastically improved by tuning and cluster generation. Initially, the number of clusters was three and during training ten clusters were dynamically generated. The final recognition rate of the training data was comparable to those of the neural network classifier and the fuzzy classifier with hyperbox regions. The final recognition rate of the test data was comparable to that of the neural network classifier. The training time of the fuzzy classifier with hyperbox regions was the shortest. Training time of the fuzzy classifier with ellipsoidal regions was about 2 min but it was much shorter than that of the neural network classifier.

Table III shows the performance of the fuzzy classifier with ellipsoidal regions by changing N_c. Without dynamic cluster generation,

TABLE IV
PERFORMANCE FOR THE THYROID DATA USING SIX CONTINUOUS INPUTS

Classifier	Rate (%)	No. Rules	Time (s)
N.N.	96.42 (97.64)	3 units	23 min
Hyperbox	96.79 (100)	61	2
Ellipsoid	94.31 (94.88) → 97.32 (98.25)	13	11

(): Recognition rate of the training data

TABLE V
PERFORMANCE OF THE FUZZY CLASSIFIER WITH ELLIPSOIDAL REGIONS FOR THE THYROID DATA BY CHANGING N_c

N_c	Rate (%)	No. Rules	Time (s)
–	96.65 (97.06)	3	4
40	96.56 (97.19)	4	5
30	97.00 (97.72)	8	9
20	97.20 (98.01)	11	11
10	97.32 (98.25)	13	11

(): Recognition rate of the training data

TABLE VI
PERFORMANCE FOR THE HIRAGANA DATA

Classifier	Rate (%)	No. Rules	Time (s)
N.N.	99.20 (99.63)	25 units	8.6 h
Hyperbox	96.73 (100)	1097	43
Ellipsoid	99.66 (99.84) → 99.78 (100)	39	127

(): Recognition rate of the training data

TABLE VII
PERFORMANCE FOR THE BLOOD CELL DATA

Classifier	Rate (%)	No. Rules	Time (s)
N.N.	87.44 (90.46)[1]	15 units	2.15 h
Polyhedron	90.58 (91.68)[1]	302	2.15 h
Hyperbox	86.52(100)[2]	217	7
Ellipsoid	87.45 (92.64)[2] → 92.13 (95.96)[2]	13	35

()[1]: Maximum recognition rate
()[2]: Recognition rate of the training data

after tuning, 118 data belonging to class 2 were misclassified into class 3. Thus if N_c was set to be larger than 118, clusters were not generated.

For $N_c = 40$, two additional clusters were generated, and the recognition rates of both the test data and the training data were improved greatly and the improvement for N_c smaller than 40 was small.

By dynamic cluster generation the recognition rate of the test data was improved but still a little lower than the average recognition rate of the neural network classifier. The main reason for the lower recognition rate was that most of the input variable were discrete and they did not obey the Gaussian distributions. To eliminate the effect of discrete input variables, we compared the performance of the neural network classifier, the fuzzy classifiers with hyperbox regions and ellipsoidal regions only using the six continuous inputs that were the first and the seventeenth to the twenty first inputs. Table IV shows the results for the neural network classifier and the fuzzy classifiers with hyperbox regions and ellipsoidal regions. The training conditions of the neural network classifier were the same as those with the 21 inputs. The maximum and minimum recognition rates of the test data (training data) were 96.53% (97.77%) and 96.38% (97.56%), respectively. The recognition rate of the fuzzy classifier with ellipsoidal regions was the best for the test data.

Table V shows the performance of the fuzzy classifier with ellipsoidal regions when the six continuous inputs were used and N_c was changed. Comparing Tables III and V, the corresponding recognition rates were almost the same. This meant that the discrete input variables did not contribute to improving the recognition rate of the fuzzy classifier with ellipsoidal regions.

B. License Plate Recognition System

The data used in this study were Japanese Hiragana characters collected to develop a license plate recognition system [10]. The original gray-scale images of 38 Hiragana characters were transformed into 7×15 pixels, the value of each pixel varied from 0 to 255. Then by performing gray scale shift, position shift, and random noise addition to the 7×15 images, the training and test data were generated. Then to reduce the number of input variables, i.e., $7 \times 15 = 105$, we calculated the 13 central moments for the 7×15 images [10], [12].

Table VI shows the results for the three-layered neural network classifier and the fuzzy classifiers with hyperbox regions and ellipsoidal regions. It took 8.6 h to train the neural network classifier with 25 hidden units. When we changed the number of hidden units to 20 and 30, the recognition rates of the test data were 98.76 and 99.16%, respectively. The expansion parameter of the fuzzy classifier with hyperbox regions was set to 0.3 and the number of fuzzy rules generated was 1097; in average 29 rules were generated for each Hiragana character. Although the recognition rate of the test data was not so good, the training time was extremely fast. When the expansion parameter was 0.01, the recognition rate of the test data was 93.93% and the number of rules generated was 612. The fuzzy classifier with ellipsoidal regions realized the best recognition rate of the test data with 39 fuzzy rules. Without dynamic cluster generation, the recognition rate of the training data was 99.99%. Namely, one datum failed to be correctly classified. When we set $N_c = 1$, one additional cluster was generated. The recognition rate of the training data reached 100% but the recognition rate of the test data was the same. The training time was about 2 min and the superiority over the neural network classifier was evident.

C. Blood Cell Data

The blood cell classification involves classifying optically screened white blood cells into 12 classes using 13 features. This is a very difficult problem; class boundaries for some classes are ambiguous because the classes are defined according to the growth stages of blood white cells.

Table VII shows the results for the neural network classifier, the fuzzy classifier with polyhedron regions, the fuzzy classifier with hyperbox regions, and the fuzzy classifier with ellipsoidal regions. The average recognition rates of the neural network classifier and the fuzzy classifier with polyhedron regions for the training data were 92.4 and 95.7%, respectively [3]. We set the expansion parameter to 0.2 for the fuzzy classifier with hyperbox regions. The recognition rate of the test data was comparable to the average performance of the neural network classifier and the initial recognition rate of the fuzzy classifier with ellipsoidal regions. The parameter N_c was set to 25 for the fuzzy classifier with ellipsoidal regions. The recognition rate of the test data was better than the maximum recognition of the fuzzy classifier with polyhedron regions.

Table VIII shows the performance of the fuzzy classifier with ellipsoidal regions when N_c was changed. When clusters were dynamically generated, the recognition rates of the test data were better than that without dynamic clustering and the best recognition rate was achieved for $N_c = 25$, in which one additional cluster was generated.

V. DISCUSSION

The fuzzy classifier with ellipsoidal regions is suited for applications whose training data belonging to a class obey the Gaussian

TABLE VIII
PERFORMANCE OF THE FUZZY CLASSIFIER WITH ELLIPSOIDAL REGIONS FOR THE BLOOD CELL DATA BY CHANGING N_c

N_c	Rate(%)	No. Rules	Time (s)
–	91.65 (95.41)	12	29
25	92.13 (95.96)	13	35
20	92.10 (96.32)	14	42
15	92.03 (96.29)	14	36
10	92.03 (96.29)	15	43
5	91.81 (97.03)	29	77

(): Recognition rate of the training data

distribution. Thus if the input variables include discrete variables such as the thyroid data, the recognition rate of the test data is lower than that of the neural network classifier and other fuzzy classifiers, even by dynamic cluster generation.

By introducing dynamic cluster generation, the recognition rates of the fuzzy classifier with ellipsoidal regions improved remarkably for the thyroid data. In [6], to improve the recognition rate, the training data were divided in advance. The best recognition rates of the thyroid test data were 96.79% with the 21 input variables and 96.47% with the six continuous input variables. These recognition rates were obtained by trial and error. By the dynamic clustering the best recognition rates of the thyroid data were 97.29% with the 21 input variables and 97.32% with the six continuous input variables. These recognition rates were obtained without much effort. We need to set a proper value to N_c. The optimal value may change according to applications. One way to avoid trial and error is to decrease N_c during training. First, we set a large value to N_c and if the recognition rate of the training data is poor for a given N_c, we decrease N_c. To avoid overfitting, the training is terminated when the recognition rate of the test data begins to increase.

VI. CONCLUSIONS

We discussed a fuzzy classifier with ellipsoidal regions that dynamically generates clusters. First, for the data belonging to a class we define a fuzzy rule with an ellipsoidal region. Namely, using the training data for each class, we calculate the center and the covariance matrix of the ellipsoidal region for the class. Then we tune the fuzzy rules, i.e., the slopes of the membership functions, successively until there is no improvement in the recognition rate of the training data. Then, if the number of the data belonging to a class that are misclassified into another class exceeds a prescribed number, we define a new cluster to which those data belong. Then we tune the newly defined fuzzy rules in the similar way as stated above, fixing the already obtained fuzzy rules. We iterate generation of clusters and tuning of newly generated fuzzy rules until the number of misclassified data does not exceed the prescribed number. We evaluated our method using the thyroid data, the Hiragana data of vehicle license plates, and the blood cell data. By dynamic cluster generation, the generalization ability of the classifier was improved and the recognition rate of the fuzzy classifier for the test data was best among the neural network classifiers and other fuzzy classifiers if there were no discrete input variables.

APPENDIX

In Appendix A, we calculate the upper bound and the lower bound of α_{ij} that allow the $l-1(\geq 0)$ data that are correctly classified to become misclassified. And in Appendix B, we check how many data that are misclassified are correctly classified if α_{ij} is changed within the bounds calculated in Appendix A. Then in Appendix C, α_{ij} is determined so that the recognition rate of the training data is maximized.

A. Upper and Lower Bounds of α_{ij}

We calculate the upper bound $U_{ij}(l)$ and the lower bound $L_{ij}(l)$ of α_{ij} allowing the $l-1(\geq 0)$ data that are correctly classified to be misclassified. We divide a set of input data into X and Y, where X consists of the data correctly classified using the set of fuzzy rules $\{R_{ij}\}$ and Y consists of the misclassified data. Then, we choose $\mathbf{x}(\in X)$ that belongs to class i, and that satisfies

$$h_{ij}(\mathbf{x}) \leq \min_{k\neq j} h_{ik}(\mathbf{x}). \tag{9}$$

If (9) does not hold, $\mathbf{x}$ remains to be correctly classified even if we change α_{ij}. If $\mathbf{x}$ further satisfies

$$h_{ij}^2(\mathbf{x}) = \frac{d_{ij}^2(\mathbf{x})}{\alpha_{ij}} < \min_{o\neq i,\, p=1,\cdots} h_{op}^2(\mathbf{x}) < \min_{k\neq j} h_{ik}^2(\mathbf{x}) \tag{10}$$

there is a lower bound $L_{ij}(\mathbf{x})$ to keep $\mathbf{x}$ correctly classified.

$$L_{ij}(\mathbf{x}) = \frac{d_{ij}^2(\mathbf{x})}{\min\limits_{o\neq i,\, p=1,\cdots} h_{op}^2(\mathbf{x})} < \alpha_{ij}. \tag{11}$$

If (10) is not satisfied, namely

$$h_{ij}^2(\mathbf{x}) = \frac{d_{ij}^2(\mathbf{x})}{\alpha_{ij}} < \min_{k\neq j} h_{ik}^2(\mathbf{x}) < \min_{o\neq i,\, p=1,\cdots} h_{op}^2(\mathbf{x}), \tag{12}$$

α_{ij} can be decreased without making $\mathbf{x}$ become misclassified.

Now the lower bound $L_{ij}(1)$, which is defined as the lower bound that does not make any correctly classified data become misclassified, is

$$L_{ij}(1) = \max_{\mathbf{x}\in X} L_{ij}(\mathbf{x}). \tag{13}$$

To clarify the discussion, we assume that $L_{ij}(\mathbf{x})$ is different for different $\mathbf{x}$. Then (13) is satisfied by one $\mathbf{x}$. Similarly, $L_{ij}(2)$, which is defined as the lower bound that allows one correctly classified datum to be misclassified, is the second maximum among $L_{ij}(\mathbf{x})$ and is given by

$$L_{ij}(2) = \max_{\mathbf{x}\in X,\, L_{ij}(\mathbf{x})\neq L_{ij}(1)} L_{ij}(\mathbf{x}). \tag{14}$$

In general,

$$L_{ij}(l) = \max_{\mathbf{x}\in X,\, L_{ij}(\mathbf{x})\neq L_{ij}(1),\cdots,L_{ij}(l-1)} L_{ij}(\mathbf{x}). \tag{15}$$

In the similar manner that we determined the lower bound $L_{ij}(l)$, we can determine the upper bound $U_{ij}(l)$. We choose $\mathbf{x}(\in X)$ which belongs to class $o(\neq i)$. Let cluster op have the minimum tuned distance $h_{op}(\mathbf{x})$

$$h_{op} = \min_{q} h_{oq}(\mathbf{x}). \tag{16}$$

Since the tuned distance $h_{ij}(\mathbf{x})$ is larger than $h_{op}(\mathbf{x})$, the upper bound $U_{ij}(\mathbf{x})$ of α_{ij} in which $\mathbf{x}$ remains correctly classified is

$$U_{ij}(\mathbf{x}) = \frac{d_{ij}^2(\mathbf{x})}{\min\limits_{q} h_{oq}^2(\mathbf{x})}. \tag{17}$$

Now the upper bound $U_{ij}(1)$, which is defined as the upper bound that does not make any correctly classified data be misclassified, is

$$U_{ij}(1) = \min_{\mathbf{x}\in X} U_{ij}(\mathbf{x}). \tag{18}$$

Here we also assume that $U_{ij}(\mathbf{x})$ is different for different $\mathbf{x}$. Then (18) is satisfied by one $\mathbf{x}$. Similarly, $U_{ij}(2)$, which is defined as the upper bound that allows one correctly classified datum to be misclassified, is the second minimum among $U_{ij}(\mathbf{x})$ and is given by

$$U_{ij}(2) = \min_{\mathbf{x}\in X,\, U_{ij}(\mathbf{x})\neq U_{ij}(1)} U_{ij}(\mathbf{x}). \tag{19}$$

In general

$$U_{ij}(l) = \min_{\mathbf{x}\in X,\, U_{ij}(\mathbf{x})\neq U_{ij}(1),\cdots,U_{ij}(l-1)} U_{ij}(\mathbf{x}). \tag{20}$$

Thus α_{ij} is bounded by

$$\cdots < L_{ij}(l) < L_{ij}(l-1) < \cdots < L_{ij}(1) < \alpha_{ij} < U_{ij}(1) < \cdots < U_{ij}(l-1) < U_{ij}(l) < \cdots. \tag{21}$$

If we change α_{ij} in the range of $(L_{ij}(1),\ U_{ij}(1))$, the correctly classified data remain to be correctly classified where $(a,\ b)$ denotes the open interval. And if we change α_{ij} in the range of $[U_{ij}(l-1),\ U_{ij}(l))$, or $(L_{ij}(l),\ L_{ij}(l-1)]$, the $l-1$ correctly classified data are misclassified where $[a,\ b]$ denotes the closed interval.

B. Resolution of Misclassification by Changing α_{ij}

For $\mathbf{x}\ (\in Y)$ which is misclassified into class i or which belongs to class i but is misclassified into class $o\ (\neq i)$, we check whether it can be correctly classified by changing α_{ij}. First, we consider increasing α_{ij}. Let $\mathbf{x}$, which belongs to class i, be misclassified into class o. This datum can be correctly classified if

$$\alpha_{ij} > V_{ij}(\mathbf{x}) = \frac{d_{ij}^2(\mathbf{x})}{\min_p h_{op}^2(\mathbf{x})} \tag{22}$$

irrespective of the values of $h_{ik}(\mathbf{x})\ (k \neq i)$ where $V_{ij}(\mathbf{x})$ is the lower bound of α_{ij} that makes the misclassified $\mathbf{x}$ correctly classified.

Let $\mathrm{Inc}(l)$ denote the number of the misclassified data that are correctly classified if we set the value of α_{ij} in $[U_{ij}(l-1),\ U_{ij}(l))$. We increase $\mathrm{Inc}(l)$ by one if $V_{ij}(\mathbf{x})$ is included in $(\alpha_{ij},\ U_{ij}(l))$ and we define

$$\beta_{ij} = \max_{V_{ij}(\mathbf{x})<U_{ij}(l)} V_{ij}(\mathbf{x}). \tag{23}$$

If α_{ij} is set to be larger than $\max(\beta_{ij}(l),\ U_{ij}(l-1))$, $\mathrm{Inc}(l)$ data are correctly classified although the $l-1$ correctly classified data are misclassified.

Let $\mathbf{x}$, which belongs to class o, be misclassified into class i. Then similar to the above discussions, we check whether $\mathbf{x}$ can be correctly classified by decreasing α_{ij}. First, the minimum tuned distance for class o should be the second minimum among n classes, namely q in the following equation needs to be o:

$$\min_k h_{ik}(\mathbf{x}) < \min_{q\neq i,\, r=1,\cdots} h_{qr}(\mathbf{x}). \tag{24}$$

Second, $h_{ij}(\mathbf{x})$ needs to be the minimum in class i, and the second minimum in class i is larger than the minimum tuned distance in class o

$$h_{ij}(\mathbf{x}) < \min_p h_{op}(\mathbf{x}) < \min_{k\neq j} h_{ik}(\mathbf{x}). \tag{25}$$

Then, the datum can be correctly classified if

$$\alpha_{ij} < K_{ij}(\mathbf{x}) = \frac{d_{ij}^2(\mathbf{x})}{\min_p h_{op}^2(\mathbf{x})} \tag{26}$$

where $K_{ij}(\mathbf{x})$ is the upper bound of α_{ij} that makes misclassified $\mathbf{x}$ become correctly classified.

Let $\mathrm{Dec}(l)$ denote the number of the misclassified data that are correctly classified if we set the value of α_{ij} in $(L_{ij}(l),\ L_{ij}(l-1)]$. We increase $\mathrm{Dec}(l)$ by one if $K_{ij}(\mathbf{x})$ is included in $(L_{ij}(l),\ \alpha_{ij})$. We define

$$\gamma_{ij}(l) = \min_{K_{ij}(\mathbf{x})>L_{ij}(l)} K_{ij}(\mathbf{x}). \tag{27}$$

If α_{ij} is set to be smaller than $\min(\gamma_{ij}(l),\ L_{ij}(l-1))$, $\mathrm{Dec}(l)$ data are correctly classified although the $l-1$ correctly classified data are misclassified.

C. Modification of α_{ij}

For $\mathrm{Inc}(l)$, $l = 1, \cdots, l_M$, where l_M is a positive integer, we find l that satisfies

$$\max_l\ (\mathrm{Inc}(l) - l + 1). \tag{28}$$

Similarly, for $\mathrm{Dec}(l)$, $l = 1, \cdots, l_M$, we find l that satisfies

$$\max_l(\mathrm{Dec}(l) - l + 1). \tag{29}$$

If there are plural l's that satisfy (28) or (29), we chose the smallest l. First, we consider the case where (28) is larger than or equal to (29). If we increase α_{ij} so that it is larger than $\beta_{ij}(l)$ in $(\alpha_{ij},\ U_{ij}(l))$, the net increase of the correctly classified data is $\mathrm{Inc}(l) - l + 1$. Thus we set α_{ij} in $[\beta_{ij}(l),\ U_{ij}(l))$ as follows:

$$\alpha_{ij} = \beta_{ij}(l) + \delta(U_{ij}(l) - \beta_{ij}(l)), \tag{30}$$

where δ satisfies $0 < \delta < 1$. Here, $\beta_{ij}(l) \geq U_{ij}(l-1)$ holds, otherwise l cannot satisfy (28).

Likewise, if (28) is smaller than (29), we decrease α_{ij} so that it is smaller than $\gamma_{ij}(l)$ in $(L_{ij}(l),\ \gamma_{ij}(l)]$ as follows:

$$\alpha_{ij} = \gamma_{ij}(l) - \delta(\gamma_{ij}(l) - L_{ij}(l)). \tag{31}$$

Equations (30) and (31) are the same as (7) and (8), respectively. The parameter δ is used to control the recognition rate of the test data (the recognition rate of the training data is the same irrespective of the value of δ).

ACKNOWLEDGMENT

The authors are grateful to Prof. N. Matsuda, Kawasaki Medical School, for providing the blood cell data and to P. M. Murphy and D. W. Aha, University of California at Irvine, for organizing the data bases including the thyroid data (http://ics.uci.edu/pub/machine-learning-databases). Thanks are also due to the anonymous reviewers for their constructive comments.

REFERENCES

[1] P. K. Simpson, "Fuzzy min–max neural networks—Part 1: Classification," *IEEE Trans. Neural Networks,* vol. 3, pp. 776–786, 1992.

[2] S. Abe and M.-S. Lan, "A method for fuzzy rules extraction directly from numerical data and its application to pattern classification," *IEEE Trans. Fuzzy Syst.,* vol. 3, pp. 18–28, 1995.

[3] F. Uebele, S. Abe, and M.-S. Lan, "A neural network-based fuzzy classifier," *IEEE Trans. Syst., Man, Cybern.,* vol. 25, pp. 353–361, 1995.

[4] M. T. Musavi, W. Ahmed, K. H. Chan, K. B. Faris, and D. M. Hummels, "On the training of radial basis function classifiers," *Neural Networks,* vol. 5, no. 4, pp. 595–603, 1992.

[5] S. Abe, *Neural Networks and Fuzzy Systems: Theory and Applications.* Boston, MA: Kluwer, 1996.

[6] S. Abe and R. Thawonmas, "A fuzzy classifier with ellipsoidal regions," *IEEE Trans. Fuzzy Syst.,* vol. 5, pp. 358–368, 1997.

[7] J. A. Dickerson and B. Kosko, "Fuzzy function approximation with ellipsoidal rules," *IEEE Trans. Syst., Man, Cybern.,* vol. 26, no. 4, pp. 542–560, 1996.

[8] R. Katayama, M. Watanabe, K. Kuwata, Y. Kajitani, and Y. Nishida, "Performance evaluation of self generating radial basis function for function approximation," in *Proc. Int. Joint Conf. Neural Networks,* Nagoya, Japan, Oct. 1993, vol. 1, pp. 471–474.

[9] S. M. Weiss and I. Kapouleas, "An empirical comparison of pattern recognition, neural nets, and machine learning classification methods," in *Proc. Int. Joint Conf. Artificial Intelligence,* Aug. 1989, pp. 781–787.

[10] M.-S. Lan, H. Takenaga, and S. Abe, "Character recognition using fuzzy rules extracted from data," in *Proc. 3rd IEEE Int. Conf. Fuzzy Systems,* Orlando, FL, June 1994, vol. 1, pp. 415–420.

[11] A. Hashizume, J. Motoike, and R. Yabe, "Fully automated blood cell differential system and its application," in *Proc. IUPAC 3rd Int. Congr. Automation New Technology Clinical Laboratory,* Sept. 1988, pp. 297–302.

[12] G. L. Cash and M. Hatamian, "Optical character recognition by the method of moments," *Comput. Vis., Graph., Image Processing,* vol. 39, pp. 291–310, 1989.

Functional Graph Model of a Neural Network

Igor T. Podolak

***Abstract*—A model representing neural networks is proposed. It uses the functional graphs notion defined by Jakubowski [6]. This is a system of nodes connected with functional edges between which binary relations can be defined. Multilayer artificial neural networks can easily be defined using functional edges to model neurons, and parametrized binary relations to model synaptic connections. Learning is also defined in terms of functional graphs. The proposed description can produce descriptions of whole classes of networks.**

***Index Terms*—Artificial neural networks, functional graphs.**

I. Introduction

As neural networks are being more and more widely used in recent years, the need for their more formal definition becomes increasingly apparent.

Haykin [5] gives the following definition of *artificial neural network* (ANN) based on *signal–flow graphs* which are networks of directed links interconnected at nodes (see Fig. 1).

Definition 1: A neural network is a directed graph consisting of nodes with interconnecting synaptic and activation links, and is characterized by four properties:

- each neuron is represented by a set of linear synaptic links, an externally applied threshold, and a nonlinear activation link; the threshold is represented by a synaptic link with an input signal fixed at a value of -1;
- the synaptic links of a neuron weight their respective input signals;
- the weighted sum of the input signals defines the total internal activity level of the neuron in question;
- the activation link squashes the internal activity level of the neuron to produce an output that represents the state variable of the neuron.

This definition describes the *architecture* of an ANN, the activation function used, and the modes of computation at nodes.

For the purpose of checking the uniqueness of ANN weights Albertini *et al.* [2] define a single hidden layer feedforward ANN with m inputs, n hidden nodes, and p outputs as a 5-tuple

$$\Sigma = \Sigma(B, C, \beta, c_0, \sigma) \tag{1}$$

where B and C are real matrices of sizes $n \times m$ and $p \times n$; β and c_0 are vectors of sizes m and p; σ: $\mathbb{R} \to \mathbb{R}$ is any function. These authors define a *behavior* function beh_Σ

$$beh_\Sigma: \mathbb{R}^m \to \mathbb{R}: u \mapsto C\vec{\sigma}(Bu + \beta) + c_0 \tag{2}$$

Manuscript received January 6, 1997; revised October 22, 1997.

The author is with the Institute of Computer Science, Jagiellonian University, Kraków, Poland (e-mail: uipodola@if.uj.edu.pl).

Publisher Item Identifier S 1083-4419(98)09264-4.

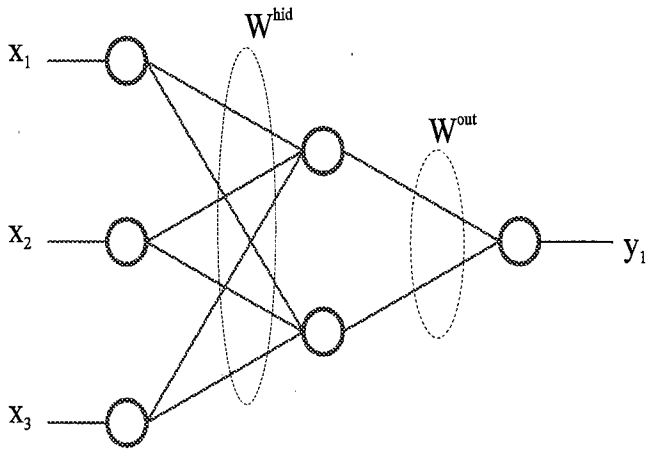

Fig. 1. A typical feed forward neural network architecture.

where

$$\vec{\sigma}(x_1, \cdots, x_n) = (\sigma(x_1), \cdots, \sigma(x_n)). \tag{3}$$

The authors then say that two networks Σ and $\hat{\Sigma}$ are *equivalent* when $beh_\Sigma = beh_{\hat{\Sigma}}$.

There is a number of graphical ANN models where numerical signals are propagated along the links of the graph: Ahson in [1] gives a model based on Petri nets, Haykin [5] as a signal flow graph, Saul and Jordan [8] represent ANN's as a hidden Markov model for the case of Bolztmann machines.

All these models define ANN's only as static objects where no change in parameters is done and the learning phase is only said to have taken place previously, or it is defined using a different formalism than the one describing the ANN's architecture. There seems to be a need for an ANN model, where all its phases (i.e., both learning and knowledge extraction) as well as ANN's architecture, are defined using the same formalism. Functional graphs seem to be perfect for this purpose.

The rest of the paper is organized as follows. In Section II the basic definition of functional graphs is given; in sections III the architecture and in Section IV learning models of a feedforward ANN are given. Concluding remarks are given in Section V. A cascade correlation ANN model is given in the Appendix.

II. Functional Graphs

In [6], Jakubowski gives a definition of a *functional graph.*

Definition 2: Functional graph is a system $S = (X, Y, F, R)$ where X and Y are sets of inputs and outputs of the system elements

$$\begin{aligned} X &= \cup_{i=1}^{m} X_i \\ Y &= \cup_{i=1}^{m} Y_i \\ X_i &= \{x_{i1}, \cdots, x_{i\alpha_i}\} \\ Y_i &= \{y_{i1}, \cdots, y_{i\alpha_i}\} \end{aligned} \tag{4}$$

where F is a set of elements f_i called *functional edges* which are pairs $f_i = (X_i, Y_i)$ and $F = \{f_i: i = 1, \cdots, m\}$, R is a set of binary relations $R \subset Y \times X$. The relation

$$R_i = \{(y, x): (y, x) \in R \wedge x \in X_i\}$$

is called a *connecting edge.*

1083–4419/98$10.00 © 1998 IEEE