

The Handling of Disjunctive Fuzzy Information via Neural Network Approach

Hahn-Ming Lee, Kuo-Hsiu Chen and I-Feng Jiang

Department of Electronic Engineering
National Taiwan University of Science and Technology
43, Keelung Rd., Sec. 4, Taipei, Taiwan
hmlee@et.ntust.edu.tw

Abstract

This paper proposes a neural network approach for the handling of disjunctive fuzzy information in the feature space. This neural network model consists of two types of nodes in the hidden layer. The prototype nodes and the exemplar nodes represent cluster centroids and exceptions in the feature space, respectively. This classifier can automatically generate and refine prototypes for distinct clusters in the feature space. The prototypes will form near-optimal decision regions to meet the distribution of input patterns and classify as many input patterns as possible. Next, exemplars will be created and expanded to learn the patterns that cannot be classified by the prototypes. Such a training strategy can reduce the memory requirement and speed up the process of nonlinear classification. In addition, on-line learning is supplied in this classifier and the computational load is lightened.

Introduction

Neural networks and fuzzy set theory are frequently used to solve pattern classification problems (Lippmann 1989; Zimmermann 1991). The objective of pattern classification is to partition the pattern space into decision regions, one region for each class (Duda and Hart 1973). However, the decision region of each class is not usually continuous but consists of disjunctive subsets of the pattern space. Such disjunctive decision regions may be caused by improper choice of features, noises, or be the nature of pattern classes. In this paper, a neural network classifier that learns disjunctive fuzzy information in the pattern space is proposed. Also this classifier can rapidly form complex decision regions with efficient memory utilization.

The proposed classifier consists of two types of nodes in the hidden layer. The prototype nodes and the exemplar nodes (Lippmann 1989) represent cluster centroids and exceptions in the feature space, respectively. The classifier generates and refines prototypes for distinct clusters in the feature space. The number and sizes of these prototypes are not restricted, so the prototypes will form near-optimal decision regions to meet the distribution of input patterns. Then these clusters learn the mappings to the responding classes. It is obviously that the decision regions may be

complex or even overlap one another so that the prototypes could not classify all patterns correctly. To solve this problem, the proposed classifier places and adjusts exemplars to learn the patterns that cannot be classified by the prototypes. The number of the exemplars generated depends on the shapes of decision regions. The more complex the decision regions are, the more exemplars will be generated. This strategy reduces the memory requirement while minimizes the amount of misclassification. The exemplars can be nested inside one another. It provides this classifier the ability to handle the exceptions inside exemplars. Each time when a new information is received, on-line learning (Simpson 1992) is realized with the aid of the generating, refining and nesting of exemplars. The inputs and weights of the proposed model are trapezoidal fuzzy intervals in *LR*-type (Dubois and Prade 1980), so numerical information and fuzzy information can be learned well. Also the computational load of the network is not heavy.

Input Representation

Each feature of all the input data, the network's prototypes and exemplars is represented as a trapezoidal fuzzy interval (Zimmermann 1991) in *LR*-type (Dubois and Prade 1980). This is because the *LR*-representation of fuzzy sets can increase computational efficiency without limiting the generality beyond acceptable limits, and the fuzzy interval can represent various types of information (Zimmermann 1991). In this way, fuzzy information and numerical information can be handled well. The membership function for a trapezoidal fuzzy interval $\tilde{W} = (w1, w2, a, b)$ is

$$\mu_{\tilde{W}}(x) = \begin{cases} L\left(\frac{w1-x}{a}\right) & \text{for } x \leq w1 \\ 1 & \text{for } w1 \leq x \leq w2 \\ R\left(\frac{x-w2}{b}\right) & \text{for } x \geq w2 \end{cases} \quad (1)$$

where $L(x) = R(x) = \max(0, 1 - x)$ (Zimmermann 1991).

The Classifier Structure

The proposed classifier is a three-layer neural network model. Figure 1 illustrates the complete structure of this classifier. This model consists of L input nodes, M exemplar nodes, R prototype nodes, and N output nodes. The input layer contains one node for each input feature. Each input feature is represented as a trapezoidal fuzzy interval in LR -type (Zimmermann 1991). These nodes are used to hold input values and to distribute these values to all nodes in the hidden layer.

The hidden layer consists of two sets of nodes: prototype nodes and exemplar nodes. Each of these nodes represents a fuzzy vector. Each class instances may generate several prototype nodes and these prototype nodes will classify as many input instances as possible. The exemplar nodes represent the exceptions that cannot be correctly classified by the prototypes. Exemplars used here are nested fuzzy hyperboxes (Salzberg 1991; Simpson 1992). Each exemplar node represents a fuzzy hyperbox. Nested fuzzy hyperboxes are used to classify nonlinearly separable instances that cannot be separated by hypersurfaces.

Let input fuzzy pattern $\tilde{X} = (\tilde{X}_1, \tilde{X}_2, \dots, \tilde{X}_L)$, where $\tilde{X}_1, \tilde{X}_2, \dots, \tilde{X}_L$ are trapezoidal fuzzy intervals in LR -type and $\tilde{X}_i = (x1_i, x2_i, a_i, b_i)$. The connection weight \tilde{W}_{ij} between input feature \tilde{X}_i and prototype node P_j is also a fuzzy interval. We use the similarity degree to indicate which prototype node is the nearest prototype to the input pattern. The similarity degree s_j between input \tilde{X} and the j th prototype is measured by

$$s_j = 1 - \sqrt{\frac{1}{L} \sum_{i=1}^L (COA(\tilde{X}_i) - COA(\tilde{W}_{ij}))^2} \quad (2)$$

where $COA(\cdot)$ is the center of the trapezoidal fuzzy interval. It should be mentioned that the feature space is rescaled into $[0,1]$. This similarity degree takes value in $[0,1]$ and it is the output of the j th prototype node. The shorter the distance between $COA(\tilde{X}_i)$ and $COA(\tilde{W}_{ij})$ is, the higher the similarity degree s_j will be.

The connection weight \tilde{V}_{ij} between input feature \tilde{X}_i and exemplar node E_j is also a trapezoidal fuzzy interval. Let $\tilde{V}_{ij} = (v1_{ij}, v2_{ij}, c_{ij}, d_{ij})_{LR}$, the subset degree r_j between input \tilde{X} and the E_j is defined as

$$r_j = \begin{cases} 2 - \sqrt{\frac{1}{L} \sum_{i=1}^L (x1_i - v1_{ij})^2 + (v2_{ij} - x2_i)^2}, & \text{if } v1_{ij} \leq x1_i \text{ and } x2_i \leq v2_{ij}, \forall \tilde{X}_i \in \tilde{X} \\ 1 - \sqrt{\frac{1}{L} \sum_{i=1}^L (COA(\tilde{X}_i) - COA(\tilde{V}_{ij}))^2}, & \text{otherwise.} \end{cases} \quad (3)$$

According to this subset degree, when the input pattern \tilde{X} falls inside the fuzzy hyperbox, the subset degree r_j will take value in $[1,2]$. It is higher than the output value (between zero and one) of any prototype node and thus this exemplar has privilege over the prototypes to be a winner.

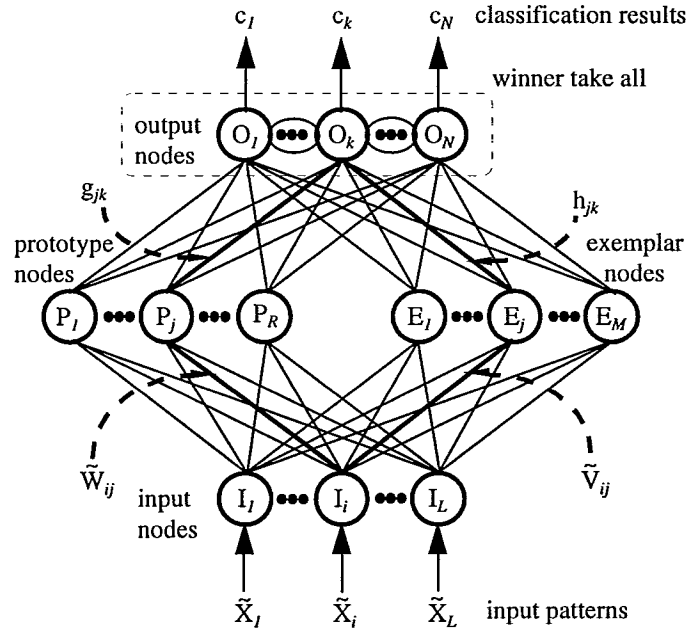


Figure 1. The structure of proposed neural network classifier.

In addition, the innermost and closest fuzzy hyperbox will take the highest value of subset-degree. If the input pattern \tilde{X} does not fall inside the fuzzy hyperbox, the subset degree r_j will take value in $[0,1]$. In this way, the exemplar nodes can handle the exceptions more efficiently.

The output layer contains a node for each class. The weight vectors from prototype nodes and exemplar nodes to each output node are adjusted such that when an input vector is presented to the classifier, this classifier's behavior will be like to check whether the input vector can be classified correctly by some exemplar. If so, the winner exemplar (i.e., the closest or the innermost exemplar) will send a correct classification output to the output node. Otherwise, the input vector is classified by the prototype nodes. The connection weight g_{jk} between prototype node P_j and output node O_k is either 1 or 0. The equation for assigning the value to the connection is given below:

$$g_{jk} = \begin{cases} 1 & \text{if prototype node } P_j \text{ is class } k \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

The connection weight h_{jk} between exemplar node E_j and output node O_k is also a binary value and is fixed. The value of the connection weight is assigned by the following equation:

$$h_{jk} = \begin{cases} 1 & \text{if exemplar node } E_j \text{ is class } k \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

The output c_k of the output node O_k indicates the degree to which the input vector \tilde{X} fits within the class k . The transfer function for the output nodes is the fuzzy union of the appropriate prototype similarity degree and fuzzy hyperbox subset degree. The output for the class node O_k is defined as

$$c_k = \max \left(\max_{j=1}^R s_j g_{jk}, \max_{j=1}^M r_j h_{jk} \right) \quad (6)$$

where s_j is the similarity degree between input \tilde{X} and prototype P_j , r_j is the subset degree between input \tilde{X} and exemplar E_j .

The Training Algorithm

In order to handle the disjunctive fuzzy information and exceptions, the training algorithm of this classifier consists of two separate passes. Such a 2-pass training strategy can speed up the process of nonlinear classification while reduce the memory requirement. In the pattern space that some classes contain several pattern clusters, the clusters needed will be determined first. The pass 1 training algorithm, named DYNamic numbers of PROtotypes (DYNPRO), is used to find the appropriate number of

prototypes according to the distribution of training patterns. In the pass 2, an on-line adaptation method based on NGE theory (Salzberg 1991), named Generalized FENCE (GFENCE), is used to generate the exemplar nodes. The GFENCE algorithm extended from our previous work FENCE (Fuzzy Exemplars Nested Creation and Expansion) (Chen et al. 1997) is applied to handle those exceptions distributed over pattern clusters.

The Pass 1 Training Algorithm—DYNPRO

The DYNPRO combines part of the Kohonen learning (Kohonen 1989) and the Grossberg's outstar network (Grossberg 1982). It is similar to Counter-propagation neural network (CPN) developed by Hecht-Nielsen (1987) and consists of two separate training subpasses. The first subpass extends the Kohonen learning to a fuzzy version. The fuzzy weights between the input nodes and winner prototype node are updated by the winner-take-all learning rule (Kohonen 1989). The goal of such a training strategy is to cluster similar training patterns by a prototype node. During few training epochs, the first subpass will automatically generate the appropriate number of prototype nodes according to the distribution of fuzzy input instances in the pattern space. After the first subpass ends, the second subpass is used to determine which class the prototype node belongs to.

In the first subpass of the DYNPRO, the number R of prototype nodes in the hidden layer is set to the number N of pattern classes first. In addition, the initial weights from input nodes to prototype nodes are set as random values in $[0.45, 0.55]$. The similarity degrees of a fuzzy input vector \tilde{X} and the prototype nodes are computed and the prototype node that has highest degree is declared to be the winner. This winning prototype node is then moved toward to the input pattern \tilde{X} . The fuzzy weight \tilde{W}_{ij} from the input feature \tilde{X}_i to the winning prototype node P_j is updated as

$$\tilde{W}'_{ij} = \tilde{W}_{ij} + \alpha(\tilde{X}_i - \tilde{W}_{ij}) \quad (7)$$

where α is the learning rate initialized in $[0, 1]$ and $i = 1, 2, \dots, L$. L is the number of input features.

After a training epoch, the terminal condition will be checked to determine whether the first subpass should be terminated. If this condition does not meet, the current weights of these existing prototype nodes will be saved and a new prototype node will be added in the hidden layer. The initial weight of this new prototype node is also set as random values in $[0.45, 0.55]$. The above training process is repeated until the terminal condition meets. When the terminal condition meets, the fuzzy weights of the prototype nodes saved in the last epoch are restored and the prototype node added in the last epoch is removed. In order to determine whether the first subpass should be

terminated, one terminal condition is proposed. An equation is adopted and defined as

$$\text{total_distance} = \sum_p (\min_j d_{pj}) \quad (8)$$

where $d_{pj} = 1 - s_{pj}$ the similarity degree between the p th input pattern and the j th prototype node. According to this equation, when the total_distance is less, the centers of existing prototypes are closer to the centers of corresponding clusters distributed over the pattern space. Therefore, training in the first subpass stops if the current total_distance is greater than the total_distance calculated in the last training epoch.

After the prototype nodes are generated by the first subpass, the second subpass is used to determine the corresponding classes of the prototype nodes. Outstar learning rule (Grossberg 1982) is used to learning the mapping. Before training, the initial weight g_{jk} between prototype node P_j and output node O_k is set to 0.5. The prototype node with the highest degree is declared to be the winner. The weight g_{jk} between the winner node P_j and the output node O_k is adjusted by the outstar learning rule (Grossberg 1982):

$$g'_{jk} = g_{jk} + \alpha(y_k - g_{jk}) \quad (9)$$

where y_k is the desired output of the input instance for the k th output node and α is the learning rate. After all input patterns are processed once, we determine the class of the prototype node as

$$g_{jk} = \begin{cases} 1 & \text{if } g_{jk} = \max_{k=1}^N g_{jk} \\ 0 & \text{otherwise.} \end{cases} \quad (10)$$

where $j = 1, 2, \dots, R$. If the g_{jk} is the highest weight value of all connection weights, this g_{jk} is set to 1 (i.e., the P_j belongs to the k th output node), otherwise the g_{jk} is set to 0. Note that this subpass will be trained for just one epoch.

The Pass 2 Training Algorithm—GFENCE

To handle those fuzzy input instances that cannot be correctly classified by the prototypes, the GFENCE (Generalized Fuzzy Exemplars Nested Creation and Expansion) training algorithm is proposed. The GFENCE is used to generate and train the exemplar nodes. In the FENCE algorithm (Chen et al. 1997), the exemplar in the pattern space will not be used unless the input instance exactly falls inside it, i.e., the exemplar nodes have no generalization ability. Thus, we redefine the similarity function of the exemplar node and modify the training algorithm to improve such a condition.

The GFENCE algorithm begins by giving an input

pattern \tilde{X} from the training set. If the prototypes constructed by DYNPRO misclassify the input pattern, GFENCE then finds an expandable exemplar of the same class that provides the highest subset degree. The found exemplar will then be expanded (if necessary) to include the input \tilde{X} . Only the exemplar that passes the size test and the overlap test can be expanded. If no expandable exemplar for the same class can be found, a new exemplar node will be created to represent the input \tilde{X} . The GFENCE algorithm can learn new classes and refine existing classes without retraining, made the classifier provide the ability of on-line learning (Simpson 1992). There are four operations in GFENCE. They are size test, overlap test, expansion, and creation. The details of the four operations are described as follows:

(1) Size test: When an exemplar E that provides the highest subset degree is chosen to be expanded, the size of the expanded exemplar E' is bounded by the size parameter $0 < \rho < 1$. Such an operation tries to avoid the overgeneralization of exemplars..

(2) Overlap test: When the exemplar that provides the highest subset degree is expanded, the expanded exemplar should not overlap other exemplars of the difference classes for all dimensions. This restriction can improve the predictive accuracy (Wettschereck and Dietterich 1995). Two exemplars with different classes are allowed to be nested. Besides, the exemplars with the same class are also allowed to overlap.

(3) Expansion: An expandable exemplar node must belong to the same class of input pattern, and pass the size test as well as the overlap test. Only the exemplar node E_j with the highest subset degree will be expanded to include the input pattern \tilde{X} .

(4) Creation: If no exemplar can be expanded to include the input instance \tilde{X} , a new exemplar node will be created to represent this input pattern. The weight vector from the input nodes to the new exemplar node will be set as the current input pattern. In addition, the weight between the new exemplar node and the output node of the corresponding class is set to one.

Experimental Results

In order to illustrate the workings of our methods, two databases Knowledge Base Evaluator (KBE) (Keller 1987) and IRIS data (Fisher 1936) are modified to include disjunctive fuzzy information for evaluating this model. Table 1 shows the 18 instances of the original KBE data. Inputs of each feature are linguistic terms. The output of the KBE is Suitability, which takes the terms: good, fair, or poor, indicating that the application of the expert system on a domain is Good, Fair or Poor. To simulate the KBE database to be a disjunctive database, we change the output

Table 1. 18 Instances of KBE

Instance	Feature						Output (Suitability)
	Worth	Employee Acceptance	Solution Available	Easier Solution	Teachability	Risk	
1	High	Positive	None	None	Frequent	Low	Good
2	Negative	**	**	**	**	**	Poor
3	Low	**	**	**	**	High	Poor
4	Moderate	Neutral	Adequate	Complete	Difficult	High	Poor
5	Low	Negative	None	Partial	Frequent	Low	Poor
6	High	Negative	Partial	None	Difficult	Moderate	Fair
7	High	Positive	Partial	Complete	Frequent	High	Poor
8	High	Positive	Partial	Partial	Possible	Low	Good
9	Low	Positive	Adequate	None	Frequent	Low	Fair
10	High	Negative	Partial	None	Frequent	High	Fair
11	Low	Positive	None	Complete	Difficult	Moderate	Poor
12	Low	Neutral	Adequate	Complete	Frequent	Low	Fair
13	Low	Neutral	None	None	Difficult	Low	Fair
14	Moderate	Positive	Adequate	None	Difficult	High	Poor
15	High	Negative	Adequate	Partial	Frequent	High	Poor
16	High	Negative	Partial	Complete	Possible	Low	Fair
17	Moderate	Negative	None	Partial	Difficult	High	Fair
18	Moderate	Neutral	Adequate	Partial	Difficult	Low	Poor

** : Don't care

(poor) of the third instance to the term-Fair. For instances 2 and 3, we generate each possible term for the don't-care condition. Thus, total 340 training instances are generated from the original 18 training instances. Two thirds of the 340 instances are randomly selected to train this model and the remaining instances are used as testing set. The learning rate for DYNPRO is set 0.015, and hyperbox size is set 0.85. The experiment results of twenty trials are shown in Table 2.

The original IRIS data set includes three classes: Virginica, Setosa, and Versicolor. The data set consists of 150 instances, 50 for each class. The Setosa class is linearly separable from the other two, but the other two are overlapped each other. In order to have disjunctive information, these instances belonging to the Setosa class are changed to belong to the Virginica class. Therefore,

the modified IRIS databases have the disjunctive property, and it is suitable for evaluating the performance of the proposed algorithm. We randomly select 75 patterns as training patterns, and the remaining patterns are used as testing set. The learning rate for DYNPRO is set 0.1. The hyperbox size is set to be 0.09. On the average, 3.6 prototype nodes are generated in twenty trials. It shows that the DYNPRO can find the appropriate number of clusters distributed over the input pattern space. Table 2 provides the results of this experiment.

Conclusion

A neural network classifier with fuzzy disjunctive information is introduced. This on-line adaptive model combines the approaches of prototype-based classifiers and the exemplar-based classifiers. During the training, the

Table 2. Classification Results

	Number of Prototypes	Number of Exemplars	Number of Hidden Nodes	Testing Recognition Rate
Modified IRIS Data	3.6	8.15	11.75	94.3%
Modified KBE	3.45	12.6	16.05	94.8%

pass 1 training algorithm DYNPRO (DYNamic numbers of PROtypes) can automatically generate the prototype nodes needed to represent the pattern clusters distributed over the pattern space. It can reduce memory and time needed efficiently. In the second pass, an on-line adaptation method GFENCE (Generalized Fuzzy Exemplars Nested Creation and Expansion) is applied to generate the exemplar nodes, and it has a better recognition rate than our previous work (Chen et al. 1997). Also nonlinear separate and overlapping classes can be easily handled. Each time when the new information is received, on-line learning is realized with the aid of the generating, refining and nesting of exemplars. In addition, since the inputs and weights of the proposed network are fuzzy intervals in LR-type, numerical information as well as fuzzy information can be learned. Besides, the computational load of the network is not heavy.

Although our model can be applied to disjunctive fuzzy information well, there are some issues needed to be investigated: (a) A policy to merge or prune some prototype nodes that are activated by few patterns. Such a policy will speed up the learning and recalling of the classifier and will reduce the memory needed; (b) Finding a better similarity function for the prototype and exemplar nodes to improve the generalization ability; and (c) Constructing a classifier with special nodes in the hidden layer. These nodes can adapt to prototype nodes or exemplar nodes. That is, the classifier would be auto-configured as a prototype-based classifier or an exemplar-based classifier depending on different applications.

References

1. Chen, K. -H., Chen, H. -L., and Lee, H. -M. 1997. A multiclass neural network classifier with fuzzy teaching inputs. *Fuzzy Sets and Systems*, 91: 15-35.
2. Dubois, D., and Prade, H. 1980. *Fuzzy sets and systems: Theory and applications*. London: Academic Press.
3. Duda, R. O., and Hart, P. E. 1973. *Pattern classification and scene analysis*. New York: John Wiley & Sons.
4. Fisher, R. A. 1936. The use of multiple measurements in taxonomic problem. *Annals of Eugenics*, 7(2): 179-188.
5. Grossberg, S. 1982. *Studies of mind and brain: Neural principles of learning, perception, development, cognition, and motor control*. Boston: Reidel.
6. Hecht-Nielsen, R. 1987. *Counterpropagation networks*. *Applied Optics*, 26: 4979-4984.
7. Keller, R. 1987. *Expert system technology: Development and application*. Englewood Cliffs, NJ: Prentice-Hall.
8. Kohonen, T. 1989. *Self-organization and associative memory* (3rd ed.). Berlin: Springer-Verlag.
9. Lippmann, R. P. 1989, November. Pattern classification using neural networks. *IEEE Communications Magazine*, 27: 47-64.
10. Salzberg, S. 1991. A nearest hyperrectangle learning method. *Machine Learning*, 6: 251-276.
11. Simpson, P. K. 1992. Fuzzy min-max neural networks—part 1: Classification. *IEEE Transactions on Neural Networks*, 3: 776-786.
12. Wettschereck, D., and Dietterich, T. G. 1995. An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms. *Machine Learning*, 19: 5-27.
13. Zimmermann, H. -J. 1991. *Fuzzy set theory: and its applications* (2nd ed.). Boston: Kluwer Academic Publishers.