Training of a Neural Network Classifier by Combining Hyperplane with Exemplar Approach

Hahn-Ming Lee and Weng-Tang Wang
Department of Electronic Engineering
National Taiwan Institute of Technology
Taipei, Taiwan. E-mail : HMLEE @ TWNNTIT.BITNET

Abstract—A neural network classifier which combines hyperplane with exemplar approach is presented. The network structure does not have to be specified before training and an appropriate network structure will be build during training. Perceptron-based algorithm is first applied to train a linear threshold unit (LTU). The LTU will build a hyperplane that classifies as many training instances as possible. Afterwards, HB nodes that represent hyperboxes will be generated to classify the training instances that can not be classified by the hyperplane. The proposed model also works well on both clustered and strip-distributed instances. Number of HB nodes generated depends on the overlapping degree of training instances. This classifier can classify continuous-valued and nonlinearly separable instances. In addition, on-line learning is supplied and learning speed is very fast. Furthermore, the parameters used are few and insensitive.

I. INTRODUCTION

The perceptron algorithm [1], one of the supervised learning algorithms, converges for linearly separable instances. The main drawback of the perceptron algorithm is that nonlinearly separable instances can not be classified. Although we can build a multi-layered perceptron [2] that forms any shape of decision region to solve this problem, unfortunately, it is quite complex and has not be proven that it will converge as with single layer perceptron. Moreover, there is no existing theory for determination of network structure. According to Lippmann89 [3], we know a training algorithm which can "match classifier complexity to training instances available" will obtain the best performance. In addition, hyperplane classifier [3] algorithms, such as perceptron and backpropagation, need low memory and have low computation load but training time may be long. So as to exemplar classifier [3] algorithms, they train rapidly but have high memory and computation requirements. We can see that, by combining these two approaches, the drawbacks of each approach could be relieved. According to the above description, we want to build a hyperplane and exemplar combined classifier algorithm with appropriate network structure generation ability for classifying continuous-valued and nonlinearly separable instances.

On the other hand, taking advantage of fuzzy theory could improve the learning ability of neural networks [4]-[6]. Therefore, our proposed classifier is concerned in combining hyperplane and exemplar classifier algorithm and utilizes fuzzy theory to classify continuous-valued and nonlinearly separable instances. Besides, if instances are strip-distributed, using the exemplar classifier algorithms is not a good choice. In the proposed classifier, most of instances are classified by the hyperplane and the hyperboxes are only responsible for classification of the overlapped instances. Therefore, number of hyperboxes needed is reduced and this makes the classifier also suitable for strip-distributed instances. The network structure of the classifier is automatically generated and the number of hidden nodes needed depends on the overlapping degree of training instances. In addition, this classifier also consists of many other features, for example, few nodes needed, on-line learning, and fast learning speed. Furthermore, the parameters needed are few and insensitive.

II. THE CLASSIFIER

This classifier is designed to generate a two-layered neural network with a single node at the output. The complete structure is illustrated in Fig.1. The first hidden layer is composed of a linear threshold unit (LTU) [1] and none or some HB nodes that represent hyperboxes. HB nodes are generated for classifying those training instances that can not be correctly classified by the hyperplane (defined by the LTU node). Therefore, the number of HB nodes generated depends on the overlapping degree of training instances. The second hidden layer contains only an output node. When an input vector X is presented to the classifier, this classifier will check whether X is contained in some hyperboxes. If it is, the innermost hyperbox will send a correct classification output to the output node. Otherwise, it will be classified

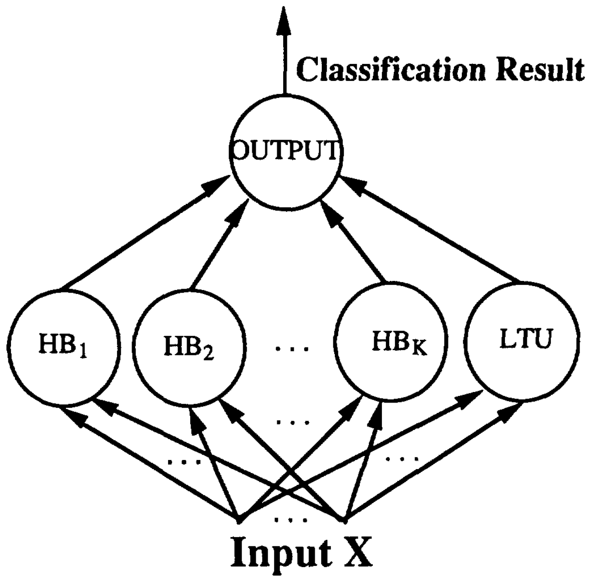

Fig. 1. The structure of the proposed calssifier.

by the hyperplane.

A . The LTU node

The hyperplane classifier algorithm used here is the pocket algorithm [7]. The pocket-run-length [7] was used as a stopping criterion. The LTU node shown in Fig.2 is first trained to build a hyperplane. Where G_i in Fig.2 is the only weight needed to be adjusted on ith connection. Another weight on the ith connection is zero. Since a hyperplane can be viewed as a special kind of a hyperbox, we use this formulation to make it consistent with the representation of the hyperboxes. The pocket algorithm will classify as many training instances as possible.

B. HB nodes

A fuzzy set can be viewed as a template of a class [8] with fuzzy boundary. The membership value for instances located in fuzzy boundary is less than one and greater than zero. Boundary of kernel part of a class (instances whose membership value is equal to one) can be quickly defined by nested hyperboxes. Moreover, nested hyperboxes can be used to classify nonlinearly separable instances. Fig.3 illustrates two cases that instances can not be separated by a hyperplane. By using two(A and B)and three(A_1, A_2, and B) nested hyperboxes respectively, instances can be easily classified.

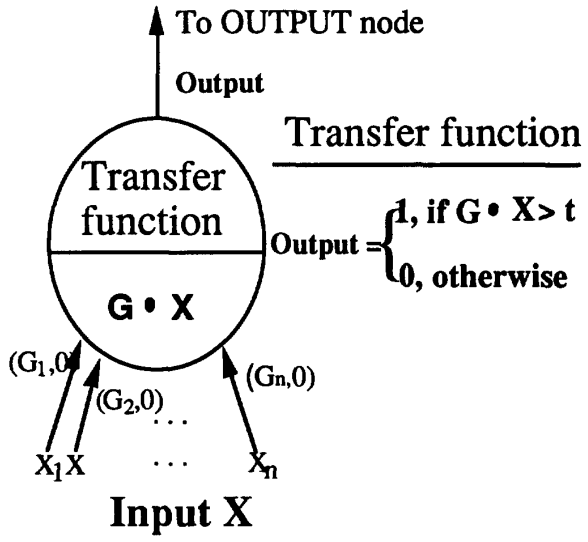

Fig.2. The LTU node.

Exemplars used here are hyperboxes. Each HB node represents a hyperbox. The minimun and maximun values of feature i of a hyperbox are the two weights, L_i and G_i respectively, on ith connection. Since we take fuzzy set as a template of a class, using the fuzzy subsethood degree [13] as the membership value makes the system work well. We use Kosko's fuzzy subsethood equation [13] (described in Section 3) as the activation function of HB nodes. A HB node's structure and its transfer function is illustrated in fig. 4. Simpson [9] used the same representation, but the activation function, transfer function, and the way hyperboxes used are different. The activation function used here is more neat and easily understood. When an input vector X is presented to

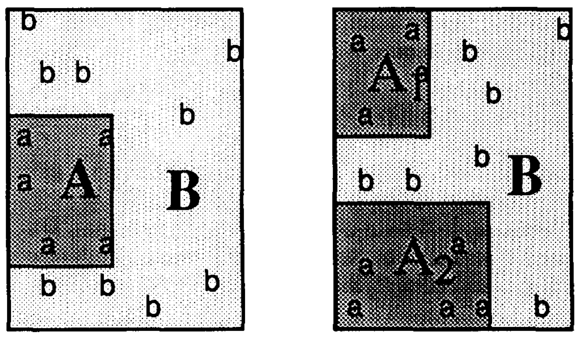

Fig.3. Nested hyperboxes which can calssify nonlinearly separable instances.

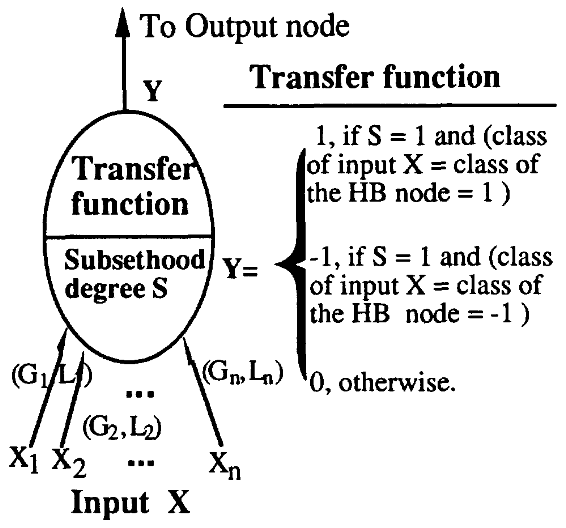
To Output node
Y
Transfer function
Transfer function
Subsethood degree S
Y=
1, if S = 1 and (class of input X = class of the HB node = 1)
-1, if S = 1 and (class of input X = class of the HB node = -1)
0, otherwise.
(G1,L1)
(G2,L2)
(Gn,Ln)
X1 X2 ... Xn
Input X

Fig.4. The HB node.

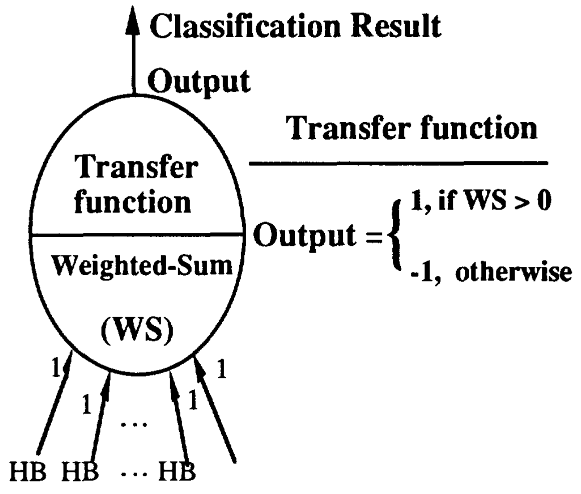

Fig.5. The Output node.

the proposed classifier, at most one HB node is allowed to be active (explained in Section 3).

C. The Output node

Fig.5 shows the structure and the function of the output node. All connection weights between the first hidden layer and the output node are 1. The output node receives the information from both the LTU node and the HB nodes. The activation function of the output node is weighted-Sum. If input X is contained in some hyperboxes, only the innermost hyperbox is allowed to be active (its output will be either -1 or 1), then the weighted-sum operation will correct the misclassification output coming from the LTU node.

III. LEARNING

Before detailing the learning procedure, we will explain fuzzy subsethood theorem first.

A. Fuzzy Subsethood Theorem

The membership value of a training instance belonging to a hyperbox is measured by Kosko's fuzzy subsethood equation. The membership value of a training instance belonging to a hyperbox is measured by Kosko's fuzzy subsethood equation [10]. It is described as follows :

Set A is a subset of set B if and only if A belongs to B's power set.

For crisp set : $A \subset B$ iff $\forall\, x \in A \rightarrow x \in B$.

For fuzzy set : $A \subset B$ iff $\forall\ x_i \in X$, $m_a(x_i) \leq m_b(x_i)$. Where $m_a(x)$ and $m_b(x)$ are the membership values of element x belonging to fuzzy set A and B, respectively. The degree $S(A,B)$ which represents the degree fuzzy set A is a subset of fuzzy set B is calculated by the equation :

$$S(A,B) = M\,(A \cap B)\ /\ M\,(A),\ 0 \leq S(A,B) \leq 1 \qquad (1)$$

, which $M\,(A) = \sum_{i=1}^{n} m_a(x_i)$, $m_{A\cap B} = \mathrm{Min}(m_a(X), m_b(X))$.

Fuzzy Set A is a subset of Fuzzy set B ($S(A,B) = 1$) if and only if A is within the hyperbox $F(2^B)$, power set of fuzzy set B. That is because $M\,(A \cap B) = M\,(A)$, and $S(A,B) = M(A)\ /\ M(A) = 1$. If fuzzy set A is outside the hyperbox $F(2^B)$ as illustrated in Fig.6, then $B^* = A \cap B$ and,

$$S(A,B) = M\,(A \cap B)\ /\ M\,(A) = M(B^*)/\ M\,(A). \qquad (2)$$

B. The learning procedure

First of all, we use pocket algorithm and stopping criterion mentioned in Section 2 to build a hyperplane which classifies most of training instances. Afterwards, hyperboxes will be generated to classify the training instances that can not be classified by the hyperplane.

To avoid one feature overwhelms the others, input value for each feature must be normalized. Normalized value of each feature is between 0 and 1. The normalization equation used is :

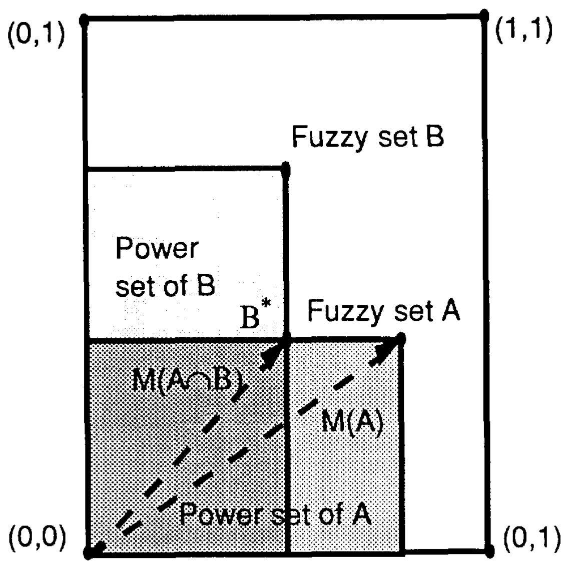

Fig.6. Fuzzy subsethood theorem[10]. S(A,B) = M(B^*)/ M (A).

$$x_i = (f_{i(max)} - x_i) / (f_{i(max)} - f_{i(min)}), \quad (3)$$

where $f_{i(max)}$ and $f_{i(min)}$ are the maximun and minimun values of feature i, respectively.

After a training instance X is normalized and presented to the proposed classifier, Kosko's fuzzy subsethood equation is used as membership function indicating the degree to which training instance X belongs to a hyperbox. Because the hyperboxes generated in the proposed classifier may be located in any position in $[0,1]^n$ instead of in position [0,0], coordinate transformation for training instance X and each hyperbox is needed before applying Kosko's fuzzy subsethood equation. The transformation procedure is described as follows :

[Step 1] : Find the farest corner U_j and nearest corner V_j of each hyperbox j to the training instance X.

[Step 2] : For each hyperbox j, let W_j be the coordinate of U_j, and Z be the dummy coordinate of X. For each value of feature i,

$$W_{ji} = | V_{ji} - U_{ji} | \text{ and } z_i = | x_i - U_{ji} |, \quad (4)$$

where $W_j = \{w_{j1}, w_{j2}, \cdots, w_{jn}\}$, $U_j = \{u_{j1}, u_{j2}, \cdots, u_{jn}\}$, $V_j = \{v_{j1}, v_{j2}, \cdots, v_{jn}\}$ and $Z = \{z_1, z_2, \cdots, z_n\}$.

The degree of a training instance belonging to a expanded hyperbox j is then calculated by the equation :

$$S(X, \text{hyperbox } j) = M(W_j \cap Z) / M(Z). \quad (5)$$

If hyperbox j is just a single point, the degree of a training instance belonging to it is calculated by the equation :

$$S(X, \text{hyperbox } j) = 1 / (1 + M(Z)). \quad (6)$$

If training instance X is not contained in any existing hyperboxes (for each hyperbox j, $S(X,\text{Hyperbox } j) < 1$), the best matched hyperbox will be expanded to contain it . The expansion criterion is : the expanded hyperbox will never overlap with other hyperboxes that represent different classes.

If expansion criterion can not be satisfied, then a new hyperbox will be generated to contain training instance X. At this time, this new hyperbox is just a point . Because hyperboxes are only generated for training instances that can not be classified by the hyperplane, hyperboxes needed are few and the chance that cross-overlapped hyperboxes occurs is reduced. Thus, if the expansion criterion can not be satisfied, we would like to create a new hyperbox. A hyperbox is expanded by the Fuzzy-AND and Fuzzy-OR operation. That is,

$$L_i = \text{Fuzzy-AND}(L_i, x_i) = \text{Min } (L_i, x_i), \quad (7)$$
$$G_i = \text{Fuzzy-OR}(G_i, x_i) = \text{Max } (G_i, x_i) \quad (8)$$

If the size of a hyperbox exceeds the maximun allowed size then a new hyperbox will be created. It is easy to see that, on-line learning can be supplied if we allow the hyperboxes to be expanded or generated during operation. Fig.7 illustrates expanding situation of a hyperbox.

Because we allow hyperboxes to be nested, inner hyperboxes represent the exceptions to the surrounding hyperboxes. It may happen that a training instance X is contained in many hyperboxes (these hyperboxes is nested). For this situation, only the innermost hyperbox will be chosen. An example is illustrates in Figure 8. In this case,

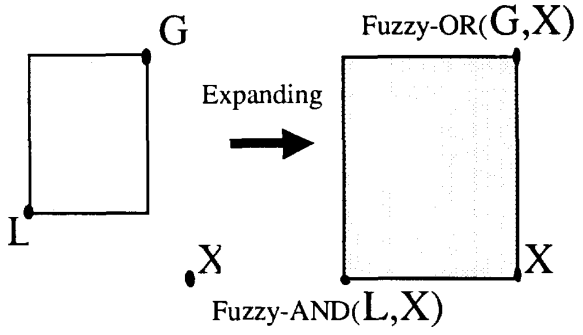

Fig. 7. The hyperbox expansion.

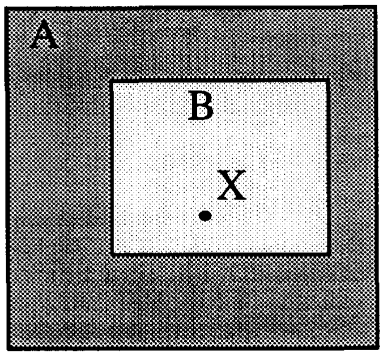

Fig. 8. The nested hyperboxes.

training instance *X* will be classified as the class the innermost hyperbox *B* represents.

IV. Experimental results

In this section, we present some experiment results produced by the simulation of proposed training algorithm. The experiment was simulated under PDP software environment[11] on SUN/ SPARC II. In order to evaluate the effectiveness of the proposed model, experimental results [4] of fuzzy perceptron and crisp perceptron running on IRIS [4] data were taken and compared to the results of the proposed model. The IRIS data set contains three classes, each with 50 instances, of four continuous features. Since only the Virginca, Versicolor of IRIS data are overlapped distributed, and Setosa is linearly separable from the above two classes, only Virginca and Versicolor were used in this experiment.

Table I summarizes the comparison results of crisp perceptron, fuzzy perceptron, and the proposed model. The data of the first two rows is obtained from fuzzy perceptron, and the data of the proposed model is the average of ten trials. Pocket-run-length [7] used here is 12 and maximun allowed size of a hyperbox was set to 0.3. Inner hyperbox's size allowed was 0.25 times of outer hyperbox's. Notice that crisp perceptron is unstable because of its decision boundary deteriorates substantially as number of iteration changes.

Leaving-one-out [12] is another approach used here for estimating classifier error rates. Maximun allowed size of a hyperbox was set to 0.3 and the pocket-run-length was set to 12. leaving-one-out error rate of our proposed model is 0.076. It is the average of ten trials.

Fig.9 shows the relationship between the maximun allowed size of a hyperbox and the number of hyperboxes generated. We can see that the number of hyperboxes generated is not absolutely increased as the maximun allowed size of a hyper size of a hyperbox is decreased. The number of hyperboxes

TABLE I
Comparison of Techniques for Estimating Speed and Error Rate.

Item / classifier	Iteration	Number of misclassification out of 100 instances
Crisp perceptron	100	3
Fuzzy perceptron	120	2
Our proposed model	4.2	0.2

generated partly depends on the input sequence of training instances and the position of generated hyperplane. Generally speaking, the higher overlapping degree of training instances is, the more hyperboxes are generated in the same condition. Moreover, pocket-run-length and maximun size of a hyperbox allowed are the two parameters needed to be adjusted and they only affects the number of HB nodes created.

It is worthwhile to notice that if instances are strip-distributed, using exemplars like hyperboxes as an approach

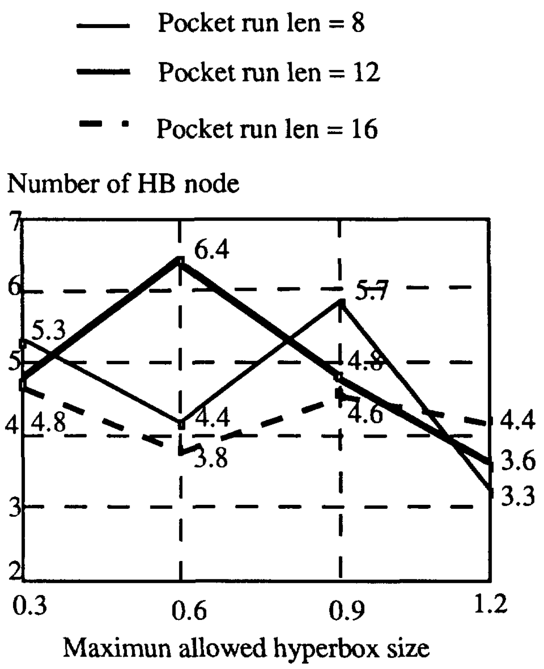

Fig 9. Relationship between hyperbox size and number of HB nodes.

is not a good choice. That is because number of hyperboxes needed is large and the frequency that overlapping hyperboxes occur may quite often in such situation. In our model, since most of instances are classified by a hyperplane, number of hyperboxes needed is reduced quite a lot. So, it works well on both clustered and strip-distributed instances.

V. Conclusion

We have proposed a neural network classifier which combines hyperplane with exemplar approach. The network structure does not have to be specified before training and the number of hidden nodes needed depends on the overlapping degree of training instances. The proposed classifier utilizes the pocket algorithm to classify most of instances. Nested hyperboxes are then generated to classify these training instances that can not be classified by the hyperplane. The hyperplane can be viewed as a special kind of a hyperbox. Kosko's fuzzy subsethood equation is utilized as the membership function of a training instance belonging to a hyperbox. This classifier also consists of many other features, for example, it can classify continuous-valued and nonlinearly separable instances , nodes needed are quite few, on-line learning is supplied, learning parameters are few and insensitive, and its learning speed is fast. Furthermore, the proposed classifier is suitable for both clustered and strip-distributed instances.

Reference

[1] Richard P. Lippman, "An introduction to computing with neural net," *IEEE ASSP magazine*, pp. 4-22, April 1987.

[2] D. E. Rumelhart, G. E. Hinton, and R.J. Williams, "Learning internal representations by error propagation," Parallel Distributed Processing : Explorations in the Microstructure of cognition, Vol. 1, Cambridge : Bradford Books / Mit press, 1986.

[3] Richard P.Lippmann, "Pattern classification using neural networks,", *IEEE Communications magazine*, pp. 47-64, November 1989.

[4] James M. Keller and Douglas J. Hunt, "Incorporating fuzzy membership function into perceptron algorithm," *IEEE Transactions. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-7, No. 6, pp. 693-699, November 1985.

[5] Gail A. Carpenter, Stephen Grossberg, and David B. Rosen, "Fuzzy ART : fast stable learning and categorization of analog patterns by an adaptive resonance system,",*Neural Networks*, Vol. 4, pp. 759-771, 1991.

[6] W. Pedrycz, "Fuzzy logic in development of fundamentals of pattern recognition," *International Journal of Approximate Reasoning*, Vol. 5, pp. 251-264, 1991.

[7] Stephen I. Gallant, "Connectionist expert system," *Communication. ACM*, Vol. 31, No. 2, pp. 152-169, 1988.

[8] Bezdek, J. *Pattern Rrecognition with Fuzzy Objective Algorithms*, Plenum Press: New York, NY, 1981.

[9] Patric K. Simpson, "Fuzzy min-max neural networks," *IJCNN*, Singapore, 1991, pp. 1658-1669.

[10] Bart Kosok, *Neural networks and fuzzy systems*, Prentice-Hall, 992.

[11] J. McClelland and D. Rumelhard, *Explorations in Parallel Distributed Processing*, Cambridge : Bradford Books/Mit Press, 1988.

[12] Sholom M. Weiss and Ioannis Kapouleas, "An empirical comparison of pattern recognition, neural nets, and machine learning classification method," *Proceeding of Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI, 1989, pp. 781-787.