# Supervised Fuzzy ART: Training of a Neural Network for Pattern Classification via Combining Supervised and Unsupervised Learning

Hahn-Ming Lee and Chia-Shing Lai
Department of Electronic Engineering
National Taiwan Institute of Technology
Taipei, Taiwan
E-mail: HMLEE@TWNNTIT.BITNET

**Abstract— A neural network model that incorporates a supervised mechanism into a fuzzy ART is presented. In any time, the training instances may or may not have desired outputs, that is, this model can handle supervised learning and unsupervised learning simultaneously. The unsupervised component finds the cluster relations of instances, then the supervised component learns the desired associations between clusters and categories. In addition, this model has the ability of incremental learning. It works equally well when instances in a cluster belong to different categories. This situation can not be handled well in unsupervised learning models, since they are unable to distinguish those instances in different categories with only the superficial input information. Moreover, multi-category and nonconvex classifications can also be dealt with.**

## I. Introduction

Artificial neural network training strategies can be identified as supervised and unsupervised learning [1]. In supervised learning, a target output pattern is associated with each training input pattern. On the other hand, the information to the network during unsupervised learning is just the input pattern. This two learning strategies have their specific environments and applications [2]. In this paper, we try to integrate these two approaches.

A neural network model that incorporates a supervised mechanism into the fuzzy ART [3] is presented here. It utilizes a learning theory called Nested Generalized Exemplar (NGE) theory [4]. The modified fuzzy ART subsystem quantizes the input vectors. A category node in this subsystem represents a cluster of the same category. It has a geometric representation of hyper-rectangle in feature space. Instances in a hyper-rectangle belong to the same category. Instances, belonging to different categories, in a cluster can be distinguished by nested hyper-rectangles [4]. A layer of Grossberg's Outstars [5] are used to learn the desired associations between categories and output patterns. In any time, the training instances may or may not have desired outputs. When the desired output is present, the model learns the desired association. When the desired output is absent, the model makes a prediction and generalization based on its learned associations. It also has the ability of incremental learning. In addition, multi-category and nonconvex classifications can be solved.

Though we derive our net from the fuzzy ART, there are some differences between them. The fuzzy ART is essentially a vector quantizer, forming categories according to the clusters of input vectors. In addition to the vector quantization, the supervised fuzzy ART can take supervised learning of categorical maps between analog vector pairs. In addition, the fuzzy ART stores only the concept (represented by the hyper-rectangle) in each category. In contrast, the supervised fuzzy ART stores both the concept and the prototype in each category. By including the statistic character (prototype) of clusters, the predictions for novel inputs are more reliable. On the other hand, the fuzzy ART recognizes only the regularities in input data. In contrast, the supervised fuzzy ART detects both the regularities and the exceptions. Exceptions are represented by nested hyper-rectangles. Thus, it can relieve the overgeneralization problems of neural net [6].

## II. System Architecture

The network architecture is shown in Fig. 1. The two lower layers are essentially fuzzy ART with complement coding [3], but with different bottom-up weights and match control. The input vector is transmitted through the bottom-up connections from $F1$ input field to $F2$ category field. Then, a node in $F2$ is chosen as a candidate category for the input vector according to the choice function. Next, the concept contained in the category is read out and transmitted through the top-down connections from $F2$ to $F1$. If the category is accepted by the match control, the concept will be generalized via including the input vector into the original concept.

A layer of Grossberg's Outstars is superimposed on the

0-7803-0999-5/93/$03.00 ©1993 IEEE

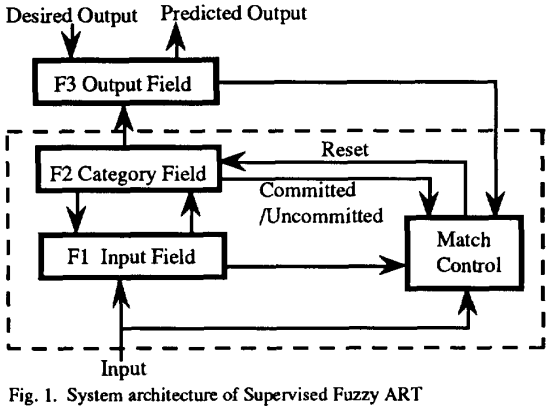

Fig. 1. System architecture of Supervised Fuzzy ART

top of $F2$ layer. During learning, the Outstar connection weights learn the desired output for the category chosen on $F2$. Initially, all categories are said to be uncommitted. After a category is selected and meets the match criteria, it becomes committed. If the chosen category is committed, $F3$ layer will read out the predicted output corresponding to the category. During learning, match signals that reflect the match degree between desired output and predicted one are computed in $F3$. The point is that another category must be chosen for the input if the desired output is different from the predicted one. The match control unit takes care of this. In what follows, we will detail the system architecture.

*Input Vector:* The input vector can be binary or analog. The components of analog input are in the interval [0,1]. The $N$-dimensional input vectors are preprocessed by complement coding [3]. The complement of the original input vector $\boldsymbol{I}$ is denoted by $\boldsymbol{I}^c$, where

$$\boldsymbol{I} = ( i_1, ..., i_i, ..., i_N ),$$
$$\boldsymbol{I}^c = ( i_1^c, \cdots, i_i^c, \cdots, i_N^c ),$$
$$\text{and } i_i^c = 1 - i_i . \quad (1)$$

*Bottom-Up Connection Weights:* There are bottom-up connections from those $F1$ nodes corresponding to the original input vector to every $F2$ node. The weight on the connection between the $i$th $F1$ node and the $j$th $F2$ node is denoted by $b_{ji}$. The weight vector $\boldsymbol{b}_j$ represents a prototype vector in the $j$th category. It supports the similarity measure between categories and the input vector (described in the choice function subsection). Initially, all bottom-up connection weights are set to 1. They are updated during learning (outlined below).

*Top-Down Connection Weights:* Unlike bottom-up connections, top-down connections connect every $F2$ node to every $F1$ node. The weight on the connection between the $j$th $F2$ node and the $i$th $F1$ node in $\boldsymbol{I}$ part is denoted by $u_{ji}$. Let $v_{ji}^c$ denote the weight on the connection between the $j$th $F2$ node and the $i$th $F1$ node in $\boldsymbol{I}^c$ part. Each node $j$ in $F2$ corresponds to a weight vector [3]

$$w_j = ( u_j , v_j^c ), \text{ where}$$
$$u_j = ( u_{j1} , \cdots, u_{ji} , \cdots, u_{jN} ),$$
$$v_j^c = ( v_{j1}^c , \cdots, v_{ji}^c , \cdots, v_{jN}^c ). \quad (2)$$

Each weight vector $\boldsymbol{w}_j$ corresponds to a $N$-dimensional hyper-rectangle [3]. The vectors $\boldsymbol{u}_j$ and $\boldsymbol{v}_j$ are two vertices of the hyper-rectangle with the minimum and the maximum values in every dimension, respectively. For the specific two-dimensional case, the rectangle associated with a weight vector is shown in Fig. 2. The same hyper-rectangle corresponds to the same cluster in the feature space. Nevertheless, input vectors in one cluster may not be associated with the same output vector. Nested hyper-rectangles are allowed to deal with this situation [4]. The inner hyper-rectangle represents an "exception" to the outer one. Within a hyper-rectangle, another hyper-rectangle is created for the input vectors mapped to different outputs (outlined in the learning subsection). Initially,

$$u_{ji} = v_{ji}^c = 1, \text{ for } 1 \le i \le N, 1 \le j \le M. \quad (3)$$

*Outstar Connection Weights:* For each $F2$ node, there are Grossberg's Outstar connections to $F3$ nodes. The weight vector $\boldsymbol{t}_j$ corresponds to the $j$th $F2$ node. During learning, the weight vector learns the activation pattern in $F3$ layer. In this way, input vectors in one hyper-rectangle are mapped to the desired output by the Outstar connections. Initially,

$$t_{jk} = 0, \text{ for } 1 \le j \le \mathrm{M}, 1 \le k \le P. \quad (4)$$

*Choice Function:* For each input vector $\boldsymbol{I}$, $F2$ node $J$ is chosen for candidate category according to the choice function

$$T_j = \beta * \sqrt{\sum_{i=1}^{N} ( dif_{ji} )^2} + ( 1 - \beta ) * \sqrt{\sum_{i=1}^{N} ( i_i - b_{ji} )^2}$$

where

$$dif_{ji} = \begin{cases} i_i - v_{ji} & \text{,when } i_i > v_{ji} \\ u_{ji} - i_i & \text{,when } i_i < u_{ji} \\ 0 & \text{, others} \end{cases} \quad (5)$$

The parameter $\beta$ is set in [0,1]. In terms of geometric interpretation, the first term of the choice function calculates the Euclidean distance between the input vector $\boldsymbol{I}$ and the hyper-rectangle corresponding to the $j$th node in $F2$.

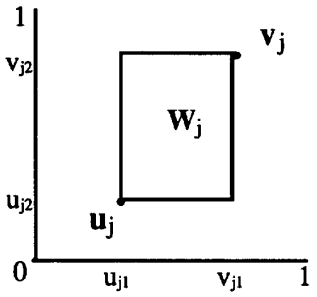

Fig. 2. Rectangle corresponding to a weight vector in two-dimensional feature space

Whenever the input vector $I$ locates in this hyper-rectangle, the distance is zero. In addition, the second term of the choice function calculates the distance between the input vector and the prototype vector $b_j$. Thus, the choice function takes account of the learned concept and the statistic character in the hyper-rectangle. The chosen category is indexed by $J$, where

$$T_J = \min \{ T_j : j = 1 \ldots M \} \tag{6}$$

All uncommitted $F2$ nodes have the maximum value for this choice function. If the input vector $I$ locates in nested hyper-rectangles, the node $J$ corresponding to the innermost hyper-rectangle is selected unless its prototype is farther from $I$. The selected node has activation value of one, and the other nodes are set to be zero.

*Matching Criteria:* If the chosen node is uncommitted, it is not necessary to check the matching criteria. Since the search for committed node to include input $I$ fails, the first chosen uncommitted node is the right choice beyond question. There are three criteria the candidate category must satisfy. The first is that the chosen category must meet the vigilance criterion [3]; that is,

$$|(I \vee v_J) - (I \wedge u_J)| \leq N * (1 - \rho) \tag{7}$$

where the fuzzy OR operator $\vee$ is defined as

$$(x \vee y)_i \equiv \max (x_i, y_i), \tag{8}$$

the fuzzy AND operator $\wedge$ is defined as

$$(x \wedge y)_i \equiv \min (x_i, y_i), \tag{9}$$

the norm is defined as

$$|x| \equiv \sum_{i=1}^{N} |x_i| \tag{10}$$

and the vigilance parameter $\rho \in [0,1]$.

If the candidate category is accepted, the corresponding hyper-rectangle will be expanded during learning to include the input vector $I$ (described below). The left term in the above vigilance criterion reflects the size of the expanded hyper-rectangle. This criterion guards the size of every hyper-rectangle below some threshold and defines the similarity between the learned concept and the input vector [3]. If the chosen category represents a inner nested hyper-rectangle, the following condition must also be met.

$$|(I \vee v_J) - (I \wedge u_J)| \leq \alpha |v_J^{(outer)} - u_J^{(outer)}| \tag{11}$$

,where $v_J^{(outer)}$ and $u_J^{(outer)}$ are vertices of the outer hyper-rectangle.

The constant $\alpha$ (nesting parameter) is suggested to be not greater than 0.25. This condition means that the inner hyper-rectangle can not be expanded to cover too much of the outer one. The reason is given in the learning subsection below.

The second criterion is that the expanded hyper-rectangle can not overlap with other hyper-rectangles; that is, the expanded hyper-rectangle must meet some conditions for all committed node $k$. The entire algorithm is attached in the appendix for detail. The checking equation shown in [7] is not sufficient. Some nonoverlap situations would be recognized as overlaps. We derive the conditions from the point of mathematics to avoid neglect. This criterion avoids confusion in recognition phase. If this condition fails, the system will search another nearby hyper-rectangle or create a new category (select another uncommitted $F2$ node) if necessary.

The last criterion is that the candidate category must make a correct prediction. The chosen node makes a prediction on $F3$ output pattern through Grossberg's Outstar connections. This prediction must match the desired output of input vector $I$. This criterion is implemented by incorporating a matching mechanism into each $F3$ node. Fig. 3 shows the processing. The output of the equivalence logic is 1 if the predicted output matches the desired output within some small threshold. The threshold equals 0 for binary inputs. This matching criterion is to check

$$\sum_{k=1}^{P} m_k = P \tag{12}$$

It guards that inputs within the same hyper-rectangle are mapped to the same output pattern.

If any of the matching criteria described above is violated, the chosen node is reset and the search process continues. In other words, if all criteria are met or one uncommitted node is selected, the system enters resonance state.

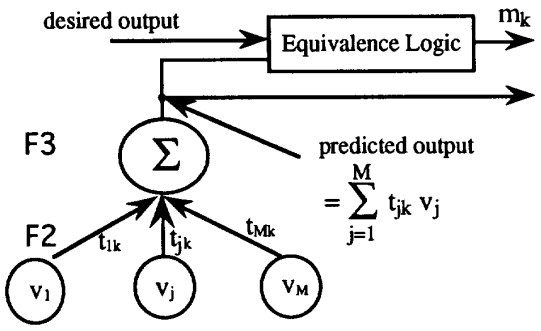


Fig. 3. The processing within each F3 node

*Learning:* Whenever resonance occurs, learning takes place. There are three kinds of weights to be updated during learning. In the bottom-up connections from $F1$ to $F2$, the weights are updated by the learning rule of the Cluster Euclidean algorithm [8],

$$b_{Ji} = (1 - \lambda)\ b_{Ji} + \lambda\, i_i, \tag{13}$$

where $\lambda$ is the learning rate.

The weight vector $b_j$ is moved toward the input vector. Eventually, the weight vector will be the prototype vector in the $J$th category. It participates in the similarity measure between the input vector and categories. By including the statistic character of clusters, the choice function could compensate for the overgeneralization by the hyper-rectangle.

In the top-down connections from $F2$ to $F1$, the weight vector is updated by the learning rule [3]:

$$u_i^{(new)} = I \wedge u_J\ ,\ v_i^{(new)} = I \vee v_J \tag{14}$$

The hyper-rectangle is expanded by this learning rule to encapsulate the input vector $I$. The vectors, $v_i^{(new)}$ and $u_i^{(new)}$, are new vertices of the expanded hyper-rectangle. A two-dimensional case is shown in Fig. 4. If the chosen node is uncommitted, the expanded hyper-rectangle is a point in feature space located by $I$. This is obvious since $I \wedge 1 = I$ and $I \vee 0 = I$.

According to the matching criteria, if the input vector $I$ is mapped to different desired output from that of the hyper-rectangle containing $I$, a new category is created. Initially, the new category represents an exception point in the original hyper-rectangle. The point will expand to a hyper-rectangle if another such input vector near the point exists. Thus, nested hyper-rectangles are constructed. Fig. 5 shows the two-dimensional case. The depth of nest is unlimited. With nested hyper-rectangles, any noncovex classification can be solved.

In the Outstar connections, the weight vector $t_J$ corresponding to the chosen node $J$ learns the desired output pattern $O^*$. The learning rule,

$$t_{Jk} = O_k^*, \tag{15}$$

effects if the chosen node $J$ is uncommitted. Because all inputs within one category are mapped to the same output pattern, the desired output is learned only when the new category is created. In any time, the training instances may or may not have desired outputs. When the desired output is present, the model learns the desired association (supervised learning). One candidate category is chosen for the input vector. If the output pattern of the chosen category is different from the desired one, the category is reset. Searching process continues until a category with the same output pattern is selected or a new category (uncommitted node) is created. The hyper-rectangle of the selected category may be expanded by the input vector. On the other hand, when the desired output is absent, the model makes a prediction and generalization based on its learned associations (unsupervised learning). If the chosen category is not reset by the matching criteria, the output pattern of the category is read out and the hyper-rectangle may be expanded by the input vector. If a new category is created, the system will request the user to support the desired output for this novel input vector. This new association will be made another training example to train the system. Therefore, this system has the ability of incremental learning.

One problem may occur in the expanded inner hyper-rectangle. As shown in Fig. 6, the inner rectangle will be expanded to cover region $A$. Some instances, belonging to the category of outer rectangle, in region $A$ may have previously been learned. If one of these instances is present again, the model will predict incorrectly. Therefore, the size of the inner hyper-rectangle is limited by matching criteria

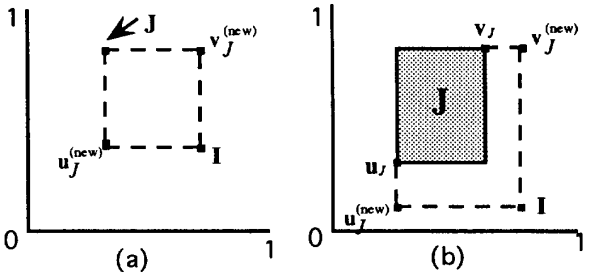


Fig. 4. During learning, the rectangle J expands to the smallest rectangle that includes the input vector I. (a) Rectangle is a point initially. (b) Rectangle represents a category.

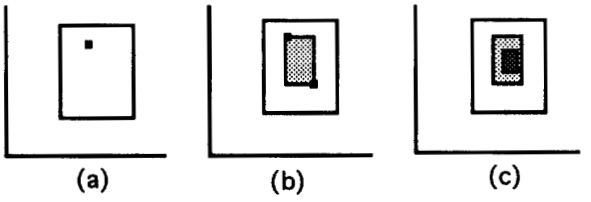


Fig. 5. Nested rectangles (a) One exception occurs within a rectangle. (b) The exception point expands to a rectangle with another exception. (c) In the inner rectangle, some exceptions occur.

to decrease the chance of misclassifications. The second term in the choice function has the same effect. Similar problem also exists in [4] and [7].

## III. Experimental Results

The example given here is the IRIS plant classification [4]. This problem is to classify a set of IRIS flowers into three classes: Virginica, Setosa, and Versicolor. The data set consists of 150 instances, 50 for each class. One class is linearly separable from the other two, but the other two are not linearly separable from each other. Therefore, it is suitable for evaluating the performance of the supervised fuzzy ART.

The simulation program was coded in C under the software environment of PDP programs [9]. It was run on the IBM PC and the Sun Workstation. In the following experiments, the net parameters: learning rate, vigilance parameter, and nest parameter, were set to 0.1, 0.2, and 0.25, respectively. The data set is obtained from the UCI Repository of Machine Learning Databases and Domain Theories, maintained by Murphy (e-mail: ml-repository @ics.uci.edu). In order to evaluate the generalization ability, the data set was partitioned into a training set and a test set, 75 instances for each set. The net was trained using the training set for only one epoch and then tested on the test set. The error rate obtained is shown in Table I for two cases -- the training instance is taken from the training set in sequence and at random, respectively. The result for random order case is the average of 30 trials. After one single pass through the training set, a 100% success rate was obtained for the training set and a 94.7% success rate was gotten for the novel test set. From these indications, we have confidence in its ability of generalization.

In addition, we made a few experiments to compare the performance of the supervised fuzzy ART with Backpropagation. Two error rates [10] were measured. The first one is the apparent error rate that is measured by testing the same 150 instances as those of training set. The second one is the leaving-one-out error rate. In this case, the net is trained using 149 instances and then tested on the remaining instance. This process is repeated 150 times, one time for each instance as the testing data. The error rate is the number of errors divided by 150. Table II shows the apparent error rate. The result for random order case is the average of 20 trials. The corresponding error rate of Backpropagation in [11] was attached for comparison. Apparently, supervised fuzzy ART is superior to Backpropagation in this respect. Table III shows the leaving-one-out error rate resulted from the average of 5 complete leaving-one-out trials. In the leaving-one-out trials, an average of 5 errors occurred in Backpropagation and an average of 1.5 errors took place in supervised fuzzy ART. By the comparisons, supervised fuzzy ART took only one epoch to get superior performance than that of Backpropagation trained for 1000 epochs.

Lastly, we made experiments to evaluate the incremental learning ability of supervised fuzzy ART. In these experiments, each of the 150 instances was utilized to train the net one by one. After the incoming instance had been learned, each of the previously learned instances was utilized to test the net. The number of recognized instances at each iteration was shown in Fig. 7. The learning of newcome instance did not corrupt the learned memory. Thus, its ability of incremental learning was demonstrated.

## IV. Conclusion

The combined learning model presented in this paper is superior to purely supervised or unsupervised models in generalization ability and performance. It can deal with the classification problems, such as the above example, which unsupervised models can not do well. In addition, since it learns the cluster relations between instances, its ability of abstraction and generalization is superior to that of supervised models. According to its operation environment, it can take both supervised learning and unsupervised learning. Since the desired output for each instance may or may not be known during operation, the proposed model is of great flexibility. It also has the ability of incremental learning. In order to correctly classify all the input vectors, additional training passes may be required, depending on the distribution of categories.

Although we are confident that our method can be used for many applications well, there are a number of issues which need to be further investigated. The first one is that the overlap checking of expanding hyper-rectangle is a tedious work when the number of hyper-rectangles increases. Efficient match control methods will be studied. The other one is that the number of hyper-rectangles created depends on the sequence of training data. The optimization of the number of the hyper-rectangles is left a future work. Nowadays we are planning to do the handwritten Chinese character recognition using this model.

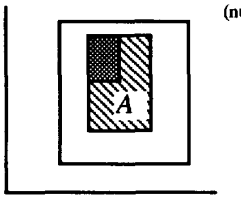


Fig. 6. The problem that may occurs in expanded inner rectangle

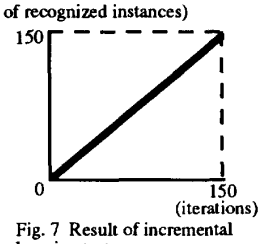


Fig. 7 Result of incremental learning test

## REFERENCES


[1] G. E. Hinton, "Connectionist learning procedures," *Artificial Intelligence*, vol. 40, pp. 185-234, 1989.

[2] Richard P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP MAGAZINE*, vol. 4, pp. 4-22, 1987.

[3] G. A. Carpenter, Stephen Grossberg, and D. 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.

[4] Steven Salzberg, "A nearest hyperrectangle learning method," *Machine Learning*, vol. 6, pp. 251-276, 1991.

[5] Gail A. Carpenter, "Neural network models for pattern recognition and associative memory," *Neural Networks*, vol. 2, pp. 243-257, 1989.

[6] Mohammad Bahrami, "Recognition of rules and exceptions by neural networks," *International Journal of Neural Systems*, vol. 2, pp. 341-344, 1992.

[7] P. K. Simpson, "Fuzzy min-max neural networks," *International Joint Conference on Neural Networks*, Singapore, 1991, pp. 1658-1669.

[8] B. Moore, "ART1 and pattern clustering," *Proceedings of the 1988 Connectionist Models Summer School*, San Mateo, CA, 1989, pp. 174-185.

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

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


TABLE I

GENERALIZATION ABILITY TEST OF SUPERVISED FUZZY ART

| order | number of category nodes | misclassifications training set | misclassifications test set | success rate |
|---|---|---|---|---|
| sequent | 4 | 0 | 4 | 94.7% |
| random | mean = 4.6<br>max. = 7<br>min. = 4 | mean = 0<br>max. = 0<br>min. = 0 | mean = 4.0<br>max. = 5<br>min. = 3 | 94.7% |

TABLE II

APPARENT ERROR TESTS OF SUPERVISED FUZZY ART AND BACKPROPAGATION

| order | number of category nodes | misclassifications | success rate |
|---|---|---|---|
| sequent | 9 | 0 | 100% |
| random | mean = 8.7<br>max. = 10<br>min. = 8 | mean = 0<br>max. = 0<br>min. = 0 | 100% |
| Backpropagation:<br>1 trial = 1000 epochs ave. of 5 trials | | | 98.3% |

TABLE III

LEAVING-ONE-OUT TESTS OF SUPERVISED FUZZY ART AND BACKPROPAGATION

| number of category nodes | misclassifications | success rate |
|---|---|---|
| mean =8.09<br>max. = 12<br>min. = 6 | mean = 1.5<br>max. = 3<br>min. = 0 | 99.0% |
| Backpropagation:<br>1 trial = 1000 epochs<br>1 leaving-one-out trial = 150 trial<br>ave. of 5 leaving-one-out trials | | 96.7% |

## APPENDIX

Hyper-rectangle Overlap Checking Algorithm

expanding hyper-rectangle J
other hyper-rectangle k
$\mathbf{v}_J^{(new)}$, $\mathbf{u}_J^{(new)}$: new vertices of hyper-rectangle J
$\mathbf{v}_k$, $\mathbf{u}_k$: vertices of hyper-rectangle k
**a**: any vector in hyper-rectangle J

To check if J overlaps with k, there are some conditions to be investigated, as following.

[*Condition*1] $\forall$ i, $u_{ki} \le v_{Ji}^{(new)} \le v_{ki}$

[*Condition*2] $\forall$ i, $u_{ki} \le u_{Ji}^{(new)} \le v_{ki}$

The following two rules can be derived.

[*Rule*1] If condition 1 and condition 2 are satisfied, hyper-rectangle J nests in the hyper-rectangle k.

[*Rule*2] If only one of the above two conditions is satisfied, overlap occurs.

On the other hand,

[*Condition*3] $\exists$ p, $u_{kp} \le v_{Jp}^{(new)} \le v_{kp}$

[*Condition*4] $\exists$ p, $u_{kp} \le u_{Jp}^{(new)} \le v_{kp}$

Because overlap $\Leftrightarrow \forall$ i, $\exists$ a, $u_{ki} \le a_i \le v_{ki}$

nonoverlap $\Leftrightarrow \exists$ i, $\forall$ a, $a_i < u_{ki}$ or $a_i > v_{ki}$

and $\forall$ i, $u_{Ji}^{(new)} \le a_i \le v_{Ji}^{(new)}$

[*Condition*5] $\exists$ q ( q $\neq$ p ), $v_{Jq}^{(new)} < u_{kq}$ or $u_{Jq}^{(new)} > v_{kq}$

therefore we get the rule

[*Rule*3] If either condition 3 or condition 4 is satisfied, nonoverlap if and only if condition 5 is true.