15th Triennial World Congress, Barcelona, Spain

# INCREASING CLASSIFICATION EFFIECIENCY USING HOMOGENEITY-BASED CLUSTER RE-ARRANGEMENT IN FAULT DIAGNOSIS

**László Kovács*, Gábor Z. Terstyánszky****

** Dept. of Information Technology, University of Miskolc, Hungary*
*kovacs@iit.uni-miskolc.hu*
*** Dept. of Software Engineering, University of Westminster, United Kingdom*
*terstyg@.wmin.ac.uk*

Abstract: The regions with uncertain decisions are typical in fault diagnosis. These regions appear as a result of model mismatches, noises and unknown inputs. To increase the efficiency of fault diagnosis it is required to improve classification accuracy in these regions. The homogeneity distribution of the codebook vectors is a key element in the accuracy of the classification process. The paper defines an appropriate homogeneity measure that is strongly correlated with the optimal misclassification error. The classification process of the Counter Propagation neural network (CPN) is investigated. Based on this homogeneity value, the paper proposes two modification algorithms for the original CPN classification algorithm to reduce the misclassification error in the regions of uncertain decisions. The accuracy of the proposed algorithm is tested with a case study. The first method alters the learning rate using the homogeneity value of the region. The second method generates an R-tree decomposition of the input space and performs redistribution of the codebook vectors. Both methods provide significant improvement in classification. *Copyright © 2002 IFAC*

Keyword: neural networks, learning algorithms, classification, fault diagnosis

## 1. INTRODUCTION

The neural networks (NN) have nowadays a widespread application in different areas. Among these areas is the fault diagnosis based on classification A lot of case studies have been published in recent years analysing the efficiency of the different neural network-based (NN) classification methods. A general conclusion of these studies, according to Holstrom et al. (1997), is that the good NN classifiers are based on the learning vector quantisation (LVQ) or on the radial basis function (RBF) method. The counter propagation network (CPN) was selected to perform the classification in fault diagnosis. Dalmi at el. (1998) and Leonard et al (1992) proved that the CPN trains faster than other neural networks and it defines better decision boundaries. The CPN is recommended to use for classification problems when training data are sparse and unrepresentative.

## 2. CLASSIFICATION USING THE COUNTER PROPAGATION NEURAL NETWORK

The CPN is a feedforward network. Its architecture is a combination of the self-organising map (SOM) of Kohonen and the outstar structure of Grossberg. Two types of layers are used. The hidden layer is a Kohonen layer with competitive nodes performing unsupervised learning. The output layer is a Grossberg layer, which is fully connected with the hidden layer and is not competitive. The teaching of the CPN is performed in two consequent phases. In the first phase the competitive layer learns how to give the best response to a particular input pattern vector. During the training the weights of the winning node is updated according to:

$$\mathbf{w}_{ij}(k+1) = \mathbf{w}_{ij}(k) + \alpha[\mathbf{x}_j(k)-\mathbf{w}_{ij}(k)] \qquad (1)$$

where $\mathbf{w}_{ij}$ - weight of edge from input $i$ to node $j$,
$\mathbf{x}_j$ - $j$-th component of the input vector,
$\alpha$ - learning rate,
$k$ - discrete time.

In the second phase, the weights of connections from the winning competitive cluster or codebook vector to the output nodes are modified:

$$\mathbf{w}_{ps}(k+1) = \mathbf{w}_{ps}(k) + \beta[\mathbf{y}_s(k)-\mathbf{w}_{ps}(k)] \qquad (2)$$

where $\mathbf{w}_{ps}$ - weight of the connection from the hidden node $p$ to output node $s$,
$\mathbf{y}_s$ - $s$-th component of the output vector,
$b$ - learning rate

The input vector $\mathbf{x}$ is an n-dimensional vector, i.e. $\mathbf{x} = (\mathbf{x}_1,\ldots,\mathbf{x}_n) \in \mathbf{R}^n$. Every input vector is associated with a class $c_j$, where the total number of classes is $m$. A classifier $g(\mathbf{x})$ can be given by a function

$$g(\mathbf{x}) : \mathbf{R}^n \rightarrow \{c_1,\ldots,c_m\} \quad (3)$$

The optimal classification function must minimise the misclassification risk. The misclassification risk $R$ can be measured by a cost value. The value $R$ depends on the probability of the different classes and on the misclassification cost of the classes.

$$R(g(\mathbf{x}) \mid \mathbf{x}) = \Sigma_{cj}\, b(g(\mathbf{x}) \rightarrow c_j) P(c_j \mid \mathbf{x}) \quad (4)$$

where

$P(c_j \mid \mathbf{x})$ - the conditional probability of class $c_j$ for the pattern vector **x** and

$b(c_i \text{®}\ c_j)$ - the cost value of deciding in favour of $c_i$ instead of the correct class $c_j$.

A simple cost function $b$ may have the following form:

$$b(c_i \text{®}\ c_j) = 0, \text{ if } c_i = c_j \quad \text{and} \quad (5)$$

$$1, \text{ if } c_i \neq c_j.$$

Using the cost function $b$, the misclassification error value can be given by

$$R(g(\mathbf{x}) \mid \mathbf{x}) = \Sigma_{g(x) \neq cj} P(c_j \mid \mathbf{x}) \quad (6)$$

The optimal classification function minimises the $R(g(\mathbf{x}) \mid \mathbf{x})$ value.

$$S_{cj}\, P(c_j \mid \mathbf{x}) = 1 \quad (7)$$

Based on (7)

if $P(g(\mathbf{x}) \mid \mathbf{x}) \rightarrow max$

then $R(g(\mathbf{x}) \mid \mathbf{x}) \rightarrow min$

The decision rule, which minimises the average risk, is the Bayes rule. It assigns the **x** pattern vector to the class that has the greatest probability for **x**. The Bayes classifier that minimises the misclassification error is defined by

$$qB(\mathbf{x}) = \text{argmax}\ q_j(\mathbf{x}) \quad (8)$$

where

$q_j$ - a posteriori probability of class $c_j$ at pattern **x**:

$$q_j(\mathbf{x}) = P(c_j \mid \mathbf{x}) \quad (9)$$

The misclassification cost is equal to

$$R(g(\mathbf{x}) \mid \mathbf{x}) = 1 - qB(\mathbf{x}) \quad (10)$$

The lower is the $qB(\mathbf{x})$ value the greater is the misclassification cost. The greatest cost is yielded if every class has the same probability for the pattern vector **x**:

$$q_j(\mathbf{x}) = 1/m \quad (11)$$

$$R(g(\mathbf{x}) \mid \mathbf{x}) = 1 - 1/m \quad (12)$$

The lowest misclassification value is equal to zero. This occurs if only one class has a non-zero probability for the pattern vector, i.e. $qB(\mathbf{x}) = 1$.

The Bayesian classifiers calculate for every **x** feature vector the probabilities of different classes and determine the class with greatest probability. If the type of the probability function is known, the parameters of the class distribution are calculated using some type of mathematical optimisation methods. If the form of the probability density function is not known, a non-parametric estimation method can be used to approximate the distribution function. One of the most widely used method of this family is the k-nearest neighbour density estimation method. The classification is performed in the following basic steps:

- out of the $N$ training vectors, identify the $k$ nearest neighbours of input element **x** irrespective of class label
- out of these $k_i$ samples, identify the number of vectors that belong to class **i**
- assign **x** to the class with maximal $k_i$

If the input space is linearly separable, then a linear classifier method can be applied. One of the best known linear classifier method is the perceptron algorithm. The boundary hyperplane is given in a parametric form:

$$g(\mathbf{x}) = \mathbf{w}\mathbf{x} + \mathbf{w}_0 \quad (13)$$

The optimal parameter values of the hyperplane are calculated by using a gradient descent method based on the perceptron cost objective function.

For the cases when the input space is not linearly separable, the family of nonlinear classifiers provides a solution. Usually, the nonlienar classifiers map the input space into a linearly separable space by applying successive phases of elementary linear classifications. Using the generalized linear classifiers, the feature vector s of the input space are mapped immediately into a new vector space where the class distribution is linearly separable.

Holstrom et. al recommended as a general conclusion of studies on complex classification problems is that a good NN classifier is based on the LVQ or the RBF methods.

## 3. UNCERTAIN DECISION AREAS

To optimise the classification function an input pattern distribution should be created with

$$\Sigma_x\, qB(\mathbf{x}) \rightarrow max. \quad (14)$$

It means that a vector distribution must to have a sharp decision class for every input pattern. To achieve this goal, the regions with uncertain decision should be excluded. A typical area with uncertain decisions is the boundary region between two classes. According to Bottou and Vapnik (1992) these regions are created by:

- low sampling regions, i.e. regions where the pattern probability density is low and which is hardly represented in the training set,
- regions where the probabilities of the different classes are very similar

The goal is to minimize the decision or classification error in the boundary region

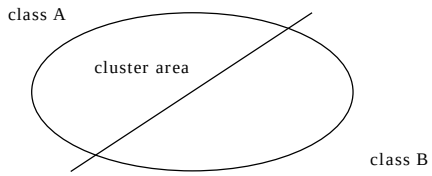


Fig. 1. Area with uncertain decisions

There are several proposals in the literature to cope with this type of uncertainty. The most known method is the active sampling method elaborated by Leisch et al. It tries to determine the distribution of training data and samples with a distribution different from the natural distribution of the data in the learning environment. The goal of active sampling is to sample more data near the class boundaries and sample with less density inside the classes.

The basic assumption of the active sampling methods is that the sampling data set can be influenced, the learning data set can be modified. In most cases, this assumption is valid, but there are situations where the training data set can not be modified, for example it would be to risky to modify the running system or it has no possibility to perform modifications. In these cases, the active sampling method can not be applied to improve the classification efficiency. In the next section, a modified CPN classification algorithm introduced to deal with these cases.

To improve the efficiency of the classification in the uncertain decision areas, we have developed and tested some new algorithms by modifying the existing basic algorithms. The aim of the investigation was to improve the classification accuracy at the border zone between two class areas. The quality of the classification process depends on several factors of the classification process. Among these factors we can stress the following ones:

- classification algorithm
- parameter values of the algorithm
- content and sequence order of the training set.

The kernel part of every algorithm is a training cycle, where the training set is presented to the network. As every phase of the algorithm affects the accuracy, it is reasonable to investigate the potential modifications in every phase. So we distinguish

- pre training cycle,
- in training cycle and
- post training cycle

modification methods. In this paper, we will show two new investigated methods. The first is the homogeneity based learning rate method which belongs to the family of the in training methods. The second is an after training cycle method. Both of the methods can increase the classification accuracy by a significant rate.

## 4. HOMOGENEITY MEASURE

There are several proposals for the homogeneity measure. The following measure was selected:

$$H = \Sigma_j \Sigma_s cnt_{js} \qquad (15)$$

where

$j$ - the cluster index,
$s$ - the class index, and
$cnt_{js}$ - the number of training vectors of the cluster $j$ that belong to the class $s$.

Training vectors are summarised in all classes but the winner class, i.e.

$$s \neq g(\mathbf{y}_s). \qquad (16)$$

The homogeneity measure $H$ is strongly related to the optimal theoretical misclassification error if the classification process meets the following two conditions:

- the classification process is based on the Bayesian decision rule,
- the classification process is based on the k-nearest neighbour estimation rule.

The k-nearest neighbour estimation rule identifies the $k$ nearest neighbours using $n$ training vectors not taking into account their class assignments. Having the nearest neighbours, it defines the number of vectors that belongs to a class $s$ from these $k$ vectors. Finally, it assigns the input vector to the class $s$ with the maximum value $k$.

In the CPN network every $\mathbf{x}$ input vector is assigned to $g(\mathbf{y}_i)$, where $i$ is the index of the winner cluster. Using the simplified cost value given by (12), the cost of classification for the input vector is

0 - if $g(\mathbf{y}_i)$ is the real class of the input vector, and
1 - otherwise.

Testing all training vectors, the total error value is equal to the $H$ homogeneity value given by (14). Thus, the $H$ value can be used to evaluate the efficiency of classification algorithms. The $H$ value is global value for the whole input space if the summing is performed on all clusters. If the summing is completed only on subspaces of the input space, a local homogeneity value is obtained. The local homogeneity value for cluster $j$ is denoted by $H_j$.

## 5. SUPERVISED CODEBOOK VECTOR RE-ARRANGEMENT

The regions with uncertain decisions are typical in fault diagnosis. These regions appear as a result of model mismatches, noises and unknown inputs. To increase the efficiency of fault diagnosis it is required to improve classification accuracy in these regions.

It is assumed that there is a given training data set to teach the CPN network that performs the classification. The codebook vectors can be managed

as the centres of the clusters in the hidden layer. Every codebook vector is assigned to a set of classes during the training process. In the test phase the input vector **x** is assigned to a winning cluster centre, i.e. to a codebook vector $\mathbf{y}_x$. Next, the winner classes belonging to the codebook vector are assigned to the input vector **x**, i.e.

$$qB(\mathbf{x}) = qB(\mathbf{y}_x) \qquad (17)$$

According to previous considerations, the $qB(\mathbf{y})$ values should be maximised in order to minimise the misclassification cost. As there are no extra training patterns in uncertain regions, this approach is based on re-arrangement of the existing codebook vectors in the training phase. Inside a class region every codebook vector has a homogenous class distribution, i.e. every training vector belonging to a particular codebook vector is assigned to the same class. On other hand, the training vectors, which are near to each other and laying on the class boundaries may be assigned to different classes. Thus, if they belong to the same codebook vector, the corresponding $qB(\mathbf{y})$ value is getting relative low. This value can be improved if only the training vectors of the same class are assigned to the same codebook vector. The codebook vector distribution density should be increased in the class boundary regions because every training vector is assigned to the nearest codebook vector. The codebook vector density should be decreased in other regions because the number of codebook vectors is constant. The density reduction can be done in the homogenous regions.

The codebook vector re-arrangement is performed during the training phase. The re-arrangement tries to increase the codebook vector density in the inhomogeneous regions and decrease it in the homogenous regions. According to the normal CPN method, the density of the codebook vectors is independent from the class homogeneity as it is based on the Kohonen SOM unsupervised learning method. Two codebook vector re-arrangement methods are presented for the CPN network in the paper. The first is a fast non-iterative algorithm; the second is a slow iterative method. A key element of both re-arrangement algorithms is the homogeneity value of the current codebook vector distribution.

## 6. NON-ITERATIVE RE-ARRANGEMENT ALGORITHM

In the modified CPN algorithm, the SOM method is changed to add the class homogeneity. First, the codebook vector and the output layer adjustment are performed together. To provide the required codebook vector density, such $\alpha$ learning rate is selected that depends not only on time but also on the $H_j$ homogeneity value, where $j$ denotes the index of the winner cluster. Thus, the modified weight update algorithm is:

$$\mathbf{w}_{ij}(k+1) = \mathbf{w}_{ij}(k) + \alpha(k, H_j)[\mathbf{x}_j(k)-\mathbf{w}_{ij}(k)]. \qquad (18)$$

The codebook vector density should be low in the homogenous regions and should be high in the non-homogenous regions. Thus, the $\alpha(k, H_j)$ function should be inversely proportional with the $H_j$ value, i.e. if $H_j$ is high, $\alpha(k, H_j)$ should be low and if $H_j$ is low, $\alpha(k, H_j)$ should be high. In regions where the class distribution is non-homogeneous like the class boundaries, the input pattern vectors have a larger attraction power than in the homogenous regions. As an effect, the density of the codebook vectors in inhomogeneous regions using this algorithm is getting larger than using the normal CPN algorithm.

To compare the efficiency of the normal CPN and the modified HOM method, tests were performed using the well-known "Circle in Square" case study. The test checks whether points of the square are included in the circle. The neural network has the following parameters:

| | |
|---|---|
| number of input nodes : | 2 |
| number of nodes in the hidden layer : | 50 |
| number of output nodes : | 2 |

The accuracy of the classification was measured by the correlation value between the calculated and actual value. The modified algorithm gave an accuracy improvement of about 5% as it is given in Figure 2. According to the test results, the improvement in classification is reduced by the fact that a codebook vector replaced by the algorithm forgets the information gained in the previous training phases and it cannot restart the training process. On the other hand, it became clear that it is not always worth to move the cluster centres close each other in the boundary zone.

## 7. ITERATIVE, R-TREE BASED RE-ARRANGEMENT ALGORITHM

An iterative R-tree based re-arrangement algorithm was also tested. In this method the codebook vectors are re-arranged after every iteration phase to move them into the inhomogeneous area. The homogeneity is measured in the whole input space using an R-tree construction. The R-tree provides a hierarchical decomposition of the input space using rectangles. The splitting process stops when number of elements in the current rectangle is below a $S$ threshold or the homogeneity is above a given $T$ threshold. Using these conditions, the leaves of the R-tree may be at different depth levels. In a homogeneous zone the leaf rectangles are large near to the root. In the inhomogeneous areas smaller rectangles are obtained at the leaf levels. The algorithm consists of the following steps:

step No.1 initial normal training phase,
step No.2 R-tree generation,
step No.3 redistribution of the codebook vectors based on the R-tree,

step No.4 re-training using the new codebook vector set,

step No.5 depending on the homogeneity value the algorithm either exits or returns to step No.2.

In this algorithm, the R-tree was used to estimate the homogeneity distribution. The R-tree provides a hierarchical decomposition of the input space using rectangles. The termination condition for the rectangle splitting is:

- number of codebook vectors in the rectangle is below a threshold value
- the homogeneity of the rectangle is better than a threshold value.

Using these termination conditions, the leaves of the R-tree may be at different depth levels. In a homogeneous zone the leaf rectangles are large near to the root. In the inhomogeneous areas smaller rectangles are obtained at the leaf levels.

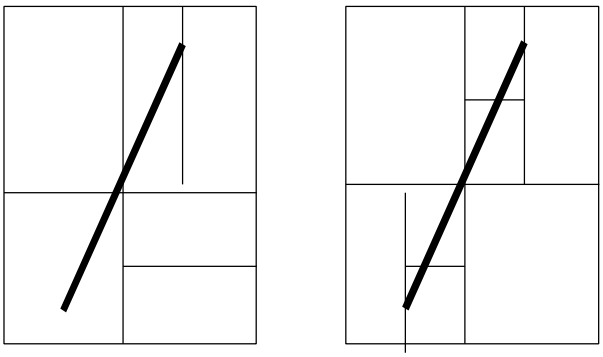

Fig. 3. R-tree before and after iteration

The redistribution of the codebook vectors also involves some random elements:

- The first *j* codebook vectors, which are selected, based on the local homogeneity value.
- These codebook vectors first, removed from the host rectangle next, moved to new positions.
- The number of new codebook vectors, to be inserted into a rectangle, depends on the homogeneity value of the rectangle. The greater is the non-homogeneity more new codebook vectors are inserted.
- The position for a new codebook vector within a rectangle is generated randomly.

The new codebook vector distribution after the first iteration cycle is evaluated and re-arranged in the next iteration cycle.

A test program was developed to demonstrate the efficiency of the modified classification method. Some test results are given in Table 1.

where

new1 - the modified SOM learning method,

new2 - the R-tree based learning method,

Rr - the redistribution rate.

| H_old | H_new1 | H_new2 | Rr |
|---|---|---|---|
| 21 | 19 | 21 | 0.35 |
| 31 | 21 | 24 | |
| 30 | 32 | 22 | |
| 44 | 37 | 30 | |
| 45 | 31 | 29 | |
| 21 | 19 | 20 | 0.15 |
| 31 | 21 | 31 | |
| 30 | 32 | 26 | |
| 44 | 37 | 34 | |
| 45 | 37 | 37 | |
| 21 | 19 | 35 | 0.55 |
| 31 | 21 | 20 | |
| 30 | 32 | 18 | |
| 44 | 37 | 28 | |
| 45 | 31 | 29 | |

Table 1.

The R-tree decomposition with codebook vector homogeneity distribution in two dimensions is shown in Figure 4 and Figure 5. The improvement of homogeneity and classification is usually about 25% but in some cases more than 50% improvement could be achieved.

## 8. POST-TRAINING RE-ARRANGEMENT ALGORITHM

The codebook vectors are moving from an initial position to a final position during the training. The current codebook vector assignment also depends on previous positions because the class assignment is the results of the whole training process. The class assignment of the codebook vectors after training may be different from the actual class distribution of the end position because class distribution of initial and final positions may differ significantly.

The modified re-arrangement algorithm reduces the impact of codebook vectors re-assignment during the training phase. The largest improvement in accuracy can be achieved if the training cycle is performed again with the original training set but it is a very time consuming process.

According to the modified algorithm the cost of the second training phase is reduced through decreasing the number of training vectors. It reduces costs because the training cost is proportional to the number of the training vectors. Some of the training vectors are selected and stored for a later use during the training process. The probability of selection is proportional to the importance of the input vector. The importance is measured by the position of the input vector. The vector closer to the class border has greater importance than vectors in the homogenous areas. As a result, a set of so-called representative points is generated by the end of the “normal” training phase. The set includes input vectors located in the regions of uncertain decisions. In the repeated training phase only the input vectors of this set are used to train the

network. In this phase the LVQ-based learning algorithm is used to adjust the positions of the codebook vectors:

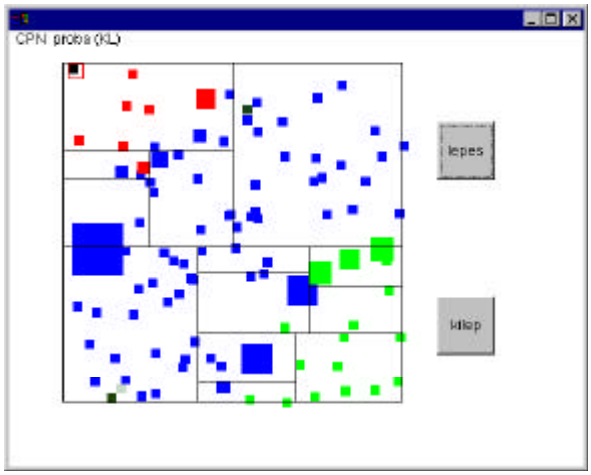


Fig. 4. Initial R-tree decomposition

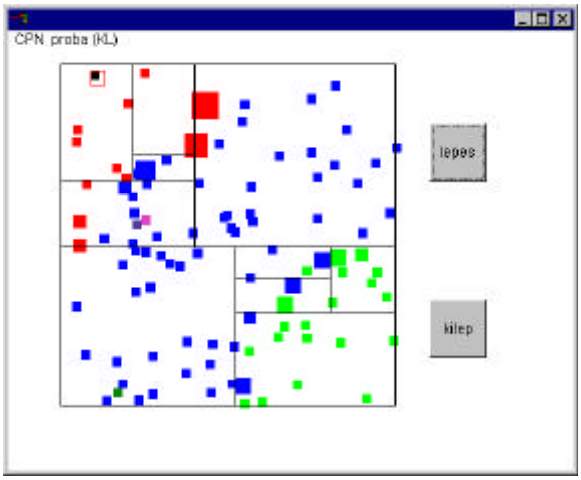


Fig. 5. R-tree decomposition after re-arrangement

The modified algorithm includes the following steps:

Step No.1 - Create a set of so called input vector representative points (SRP).

Step No.2 - SRP is changed dynamically where the SPR contains only elements at the border.

Step No. 3 - At the end of training phase, the SPR elements of strange class are used to tune the position of the codebook vectors.

According to test results the modified algorithm gives 5% improvement in accuracy.

## 9. CONCLUSION

The regions with uncertain decisions are typical in fault diagnosis. These regions appear as a result of model mismatches, noises and unknown inputs. To increase the efficiency of fault diagnosis it is required to improve classification accuracy in these regions.

To improve the classification efficiency, the classification process should also manage the region of uncertain decisions. The paper presented two modified algorithms to reduce the misclassification error. The first algorithm changes the learning rate based on the homogeneity value of the region. The second algorithm generates a R-tree decomposition of the input space and performs redistribution of the codeboook vectors. It trains the network in an iterative way. Both methods yielded in significant classification improvement.

## REFERENCES

Holstrom L, Koistien P, Laaksonen J., Oja E: Neural and Statistical Classifiers - Taxonomy and Two Case Studies, IEEE Trans. On Neural Networks, **Vol 8**, No 1, 1997.

Dalmi I, Kovacs L, Lorant I, Terstyanszky G: Adaptive Learning and Neural Networks in Fault Diagnosis, CONTROL'98, 01-04 Sep. 1998, Swansea, United Kingdom, Proceedings Vol. 1. pp. 284-289

Leonard, Kramer, Ungar: Using radial Basis Functions to Approximate a Function and its Error Bounds IEEE Transaction on Neural Networks, 1992, pp 624-627

Bottou L, Vapnik V: Local Learning Algorithms, Neural Comput, 1992 **Vol. 4**, pp.888-900

Leisch F, Jain L, Hornik K: Cross-Validation with Active Pattern Selection for Neural Network Classifiers, IEEE Trans. On Neural Networks, 1997, **Vol 9**, No 1, 19