Automatic Classification Using Neural Networks

Gamal A. M. Al-Shawadfi and Hindi A. Al-Hindi

College of Business and Economics,
King Saud University,
Al-Qasseem Branch, Al-Melaida
Saudia Arabia

ABSTRACT

This paper proposes an artificial neural network (ANN) to perform linear and nonlinear classification of objects into several classes. The theoretical and practical aspects of the proposed approach are introduced, and its validity was evaluated by the rate of correct classifications. A Matlab macro program was written for automatic classification using an artificial neural network for linear and nonlinear classification problems. The network is designed, trained and tested with different sample sizes. The results were compared to those obtained from Fisher discriminant function. In contrast with the classical classification procedures, ANNs do not require any pre assumptions about types of data, distributions or the variance covariance matrices. The numerical results illustrate the capabilities of ANNs in solving linear and nonlinear classification problems.

Keywords: *Automatic Classification, Artificial Neural Networks, Fisher Discriminant Function.*

1. INTRODUCTION

Classification is a multivariate technique concerned with allocating new objects (or observations) into previously defined groups (populations). A distinction should be made between 'classification' and ‘discriminant analysis’. Discriminant analysis is a multivariate technique concerned with separating distinct sets of objects and often employed on a one time basis in order to investigate observed differences when causal relationships are not well understood (Johnson and Wichern, 1992). On the other hand, classification is the problem in which an object is assigned to one of several classes and usually requires supervised learning methods in which objects are assigned to known groups (Fausett, 1994).

There are many studies related to classification and neural network fields. Wernecke et al. (1995) discussed the validation of Classification Trees. Arminger and Enache (1995) illustrated the relation between statistical models and artificial neural networks. More details on fundamentals of neural networks, architectures, algorithms,

applications and classification can be found in Fausett (1994). The classical classification approaches are found in some text books, see for example, Johnson and Wichern (1992) and Stevens (1992). Some neural network related concepts and applications are found, for example, in Elman (1993), Hertz et al. (1991), Nerrand et al. (1993) and Tsoi and Tan (1997).

In recent years, the use of artificial neural networks has increased for solving a wide range of problems including pattern recognition, classification and functional approximation. An artificial neural network consists of a number of simple processing units or neurons and connections. Each simple processing unit is associated with two major functions: a summation function and an activation function. The summation function sums the inputs coming to the neuron from other neurons or from the outside environment. The activation function determines the activation level of the neuron based on the inputs it receives. Connections represent the links between nodes and each connection is associated with a weight that reflects the strength of the relationship between the connected nodes. Training the network is the process of developing a function that maps input vectors to output vectors with minimum errors using a set of training examples that includes input and output data pairs.

For many practical cases, a feed forward neural network with three different layers is appropriate because it has been proven to be an efficient system for representing nonlinear relationships between a set of input and output vectors to a high degree of accuracy. The first layer is the input layer whose nodes receive the inputs from outside. The second layer is the hidden layer whose nodes receive inputs from the input neurons and propagates them to the output nodes. The third layer is the output layer whose nodes determine the outputs of the neural network.

In this paper, we propose a neural network approach to perform automatic classification and compare its performance with the classical classification method. With the classical classification method, all groups are assumed to have the same variance-covariance matrix and the distributions are normal. In the absence of these two assumptions, linear or quadratic classification rules are inadequate (Johnson and Wichern, 1992). The neural network classification technique does not impose such assumptions.

The performance of the two classification techniques is illustrated with three data sets that use two variables in classifying objects into three categories. The first data set describes the case of linear classification. The second and third describe the case of nonlinear classification with varying degrees of nonlinearity.

The rest of the paper is organized as follows. The second section presents the proposed automatic classification method. The third section presents the simulated classification problems to test the proposed approach. The fourth section summarizes this paper.

2. PROPOSED ANN AUTOMATIC CLASSIFICATION METHOD

Let us start with a description of the kind of data for which the proposed classification method is appropriate. The data consists of objects, and their corresponding descriptions. The objects may be quantitative, for example students cumulative rates, or qualitative, for example documents, keywords, hand written characters and species. Qualitative data may be replaced with quantitative data according to suitable coding scheme. Then the classification method is used to summarize and simplify the data.

Classification rules are developed from training samples. Usually, the data are separated into two subsets, where the first, called training set, is used to fit a suitable discriminant function, and the second, called the test set, is used to check the validity of the discriminant function for classification. In the ANN model, the data need to be normalized usually into the range [0,1] before training. In this research, the following equation is used to normalize the data:

$$x_i = 0.8 \left(\frac{y_{max} - y_i}{y_{max} - y_{min}} \right) + 0.1 \qquad (1)$$

where y_{max} and y_{min} are the respective maximum and minimum values of all observations.

Training in neural networks is the process of adjusting the weights of the network connections in order to obtain the desired outputs (z_i) given a set of inputs ($x_{i1}, x_{i2}, \ldots, x_{im}$). The discriminant function is:

$$z_i = f_i(x_{i1}, x_{i2}, \ldots, x_{im}) + u_i \qquad (2)$$

where z_i represents the classes (outputs), i=1,2,…..,k and $f_i(x_{i1}, x_{i2}, \ldots, x_{im})$ is a linear or nonlinear function in the input variables $x_{i1}, x_{i2}, \ldots, x_{im}$ and u_i represents errors between the input and the output values.

Equation (2) represents a class of artificial neural networks which may be interpreted as a complex multivariate statistical model for the approximation of an unknown expectation function of a random variable z given an explanatory variables x_i's, $i=1,2,\ldots,m$ This class of models represents a discriminant function.

The training process includes two stages: forward and backward pass. In the forward pass, the input neurons receive input examples and pass them to hidden layer. Each neuron calculates its activation level using the summation function to sum the weighted inputs. This sum is then used by the activation function to determine the output of the neuron. Usually, the activation function is a linear combinations of parameters and inputs. In practice, many types of transfer functions may be used, which include: linear combination, normal distribution, logistic distribution, hyperbolic tangent, indicator function and threshold functions. The selected activation function here is a sigmoidal function. Sigmoid activation functions are easy to differentiate and make computation of the gradient easy. The sigmoid function has the form:

$$f_j(x_1, \ldots, x_n) = \frac{1}{1 + e^{\sum_{i=1}^{n} w_{ij} + b_j}} \qquad (3)$$

where $f_j(x_1, \ldots, x_n)$ is the output, x is the input observations, w called weights, and b is the bias term. The values of w and b should be initialized before the training phase. In Matlab package, the function INITFF is used to determine initial values for w and b.

The outputs of the hidden neurons are used as input to neurons in the output layer. Neurons in the output layer use the summation and activation functions to determine the outputs of the neural network. The generated outputs is then compared to the actual outputs and the resulting errors is propagated back through the network in the backward pass of the training algorithm.

The weights w_{jk} that connect neurons in the hidden layer with neurons in the output layer are updated first. Then, the weights w_{ij} that connect neurons in the input layer with neurons in the hidden layer are updated. There

are two strategies that can be used to update the connection weights. The first is to calculate the error for each training example and propagate this error back to update the weights. The second strategy is to calculate the errors for all training examples and use this sum to update the weights. The iterative process of the back propagation training algorithm continues until the error function reaches a predetermined level (say 0.001), or the number of iterations is satisfied. The error function is:

$$MSE = \frac{1}{n} \sum (c_j - c_j(NNC))^2 \qquad (4)$$

where MSE is the mean of sum of squares of the difference between the actual output (c_j) and NN estimated output $c_j(NNC)$.

Due to their capabilities, neural networks may be used to approximate continuous mapping functions. The following theorem of Kolmogorov shows the existence of the mapping neural network.

Theorem: Any continuous function $f(x_i), i = 1,2, \ldots n$ of several variables defined on I^n, where $n \geq 2$ and I =[0,1] , can be represented in the form :

$$f(x) = b_j \sum_{j=1}^{2n+1} w_{ij} x_i \qquad (5)$$

where b_j and w_{ij} are continuous functions of one variable and w_{ij} are monotonic functions that do not depend on *f*.

The theorem states that: "a feed forward neural network with three layers of neurons (input, hidden and output units) can represent any continuous function exactly", see Fausett (1994) pp.328-329.

A good classification procedure should result in few misclassifications. The validity of the proposed NN classification method may be checked using the apparent error rate (APER). The apparent error rate is the fraction of observations in the test set that are misclassified by the discriminant function. The apparent error rate is calculated according to the equation:

$$APER = \frac{\sum_{i=1}^{k} n_i}{\sum_{i=1}^{k} N_i} \qquad (6)$$

where N_i is the number of observations in group i, i = 1,2,….,k , and n_i is the number of items misclassified in another group other than i. However, we can measure the validity of the neural network classification procedure by calculating the rate of correct classification from the following equation:

$$CCR = 1 - \frac{\sum_{i=1}^{k} n_i}{\sum_{i=1}^{k} N_i} \qquad (7)$$

The correct classification rate (CCR) represents the item in the test set that are correctly classified. This measure does not depend on the form of the parent population, also it can be calculated for any classification procedure.

To do automatic classification for data in several classes, we designed a Matlab macro program presented in the appendix. In the program we use essentially three functions from neural network toolbox. INITFF to initialize feed-forward network up to three layers. TRAINBPX to train a feed-forward network with fast back propagation and can be invoked with 1, 2, or 3 sets of weights. SIMUFF to simulate a feed-forward network with up to 3 layers.

In summary, the steps of the proposed NN classification approach are:

1. Designing the network: Use equation 1 to scale the data between 0, 1 interval, select a suitable discriminant function, determine the number of training epochs, and error level, and select initial values for the weights and biases.

2. Training the network: Spilt the data into two groups, and use the first group to train the network and estimate the parameters w_i, b_i of the discriminant function.

3. Checking the validity: Check the validity of the proposed discriminant function, using the second group of data with equation (7) which gives the rate of the correct classifications.

4. Classifying new observations: If step 3 is satisfied, use the proposed discriminant function in classifying new observations into suitable groups, otherwise repeat the training process with other function and weights to obtain a more efficient discriminant function.

The following sets of examples are used to compare the performance of neural networks with classical classification.

3. SOME EXAMPLES

Three different groups, with different properties regarding the relationships between variables, are used in the comparative procedure. The first group shows the classification in case of linear relationship between inputs and outputs. The second and the third groups show the classification in the case of nonlinear relationships between the inputs and the outputs. In all examples, both the proposed neural network and Fisher classification approaches are used in solving the problem of classifying objects into three categories according to some inputs. The inputs are two variables related linearly or nonlinearly to the output variable. For each group, different sample sizes are used to show the effect of increasing the sample size on classification accuracy. The selected samples sizes are 10, 30, 50, 70, 90, 110, 130 and 150. The validity of the classification results are tested by the rate of correct classifications (CCR).

The four steps of the proposed neural network classification mentioned in section (2) were applied here: designing the network, training the network, checking the validity, and classifying new observations.

3.1 The Case of Linear Classification

In this case, the output variable (or object) is classified into three categories according to two variables (inputs). The data samples are generated according to the following linear equation:

$$y_3 = 0.5\ y_1 + .5\ y_2 \qquad (8)$$

where y_1 and y_2 are input variables and y_3 is the output. Our aim is to use the proposed NN procedure to find a function to allocate an observation from y_3 into class 1, 2 or 3 according to the values of y_1 and y_2.

In Fisher classification procedure, sample discriminants are used to allocate x_i into population k if:

$$\sum_{j=1}^{r} (y_j - \bar{y}_{kj})^2 = \sum_{j=1}^{r} (\lambda_j(\hat{x} - \bar{x}_k))^2 \ ? \qquad (9)$$
$$\sum_{j=1}^{r} (\lambda_j(\hat{x} - \bar{x}_i))^2$$

For all i ? k, where λ_j is the eigenvectors of $w^{-1}B_0$ and r ? s.

Table 1:
The NN and classical classifications of objects into 3 classes
$y_3 = 0.5\ y_1 + 0.5\ y_2$

N	NN CCR	F. CCR
10	0.9000	0.8000
30	0.8333	0.6667
50	0.7400	0.7600
70	0.9000	0.7571
90	0.7778	0.9333
110	0.9727	1.0000
130	0.8385	0.9923
150	0.8600	0.9667
average	0.8528	0.8595

Table 1 shows the rate of correct classification (CCR) for each method. The first column represents the sample size. The second column shows the rate of correct classification (CCR) for the neural network approach. Column three shows the rate for the Fisher classification method. As it is shown in the table, the rates of correct classification for the two methods are very close in average, the average of correct classification for the neural network approach is 85.29% and 85.95% for the classical classification approach.

3.2 The Case of Nonlinear Classification

For this case, data samples were generated to show the capabilities of the proposed neural network approach in solving nonlinear classification problems. Each new observation is classified into one of three categories according to two input variables. The data were generated according to the equation:

$$y_3 = 0.5\ y_1 + 0.5\ y_2^2 \qquad (10)$$

Table 2 shows the NN and Fisher classifications of objects into 3 classes for the nonlinear case. We find that the overall performance of the neural network nonlinear classification approach is fairly very good, in the sense that the average rate of correct classifications is 84.74%. With respect to the sample sizes, these rates fluctuate between 63.33% and 100%. On the other hand, the rates of Fisher classification approach fluctuate between 60% and 84.29%.

Table 2:
The NN and classical classification of objects into 3 classes
$y_3 = .5y_1 + .5y_2^2$

N	NN CCR	F. CCR
10	1.0000	0.7000
30	0.6333	0.6000
50	0.8400	0.8200
70	0.7000	0.8429
90	0.9556	0.7556
110	0.9200	0.7200
130	0.8700	0.8300
150	0.8600	0.8100
average	0.8474	0.7598

The other set of data samples for the nonlinear classification was generated according to the following equation:

$$y_3 = 0.5\ y_1 + 0.5\ y_2^3 \qquad (11)$$

From table 3, we find that the performance of the NN nonlinear classification approach is extremely high where the average of correct classification rate is 97.5%. The rates fluctuate between 80% and 100%. On the other hand, the rates of the classical classification approach fluctuate between 72.86% and 96.67%. Also, the average rate is only 81.4%.

Table 3:
The NN and classical classifications of objects into 3 classes
$y_3 = .5y_1 + .5y_2^3$

N	NN CCR	F. CCR
10	0.8000	0.8000
30	1.0000	0.9667
50	1.0000	0.8200
70	1.0000	0.7286
90	1.0000	0.7667
110	1.0000	0.7600
130	1.0000	0.8200
150	1.0000	0.8500
average	0.9750	0.8140

4 CONCLUSION

This paper proposes a neural network method to do nonparametric automatic classification. The proposed method may be used in classifying a sample of n objects into k classes according to linear or nonlinear discriminant function. The discriminant function is obtained from a training process of the neural network. The selected discriminant function is the one which minimizes the difference between the actual and the expected numbers of classes. The validity of the proposed discriminant function is measured by the rate of correct classifications (CCR), where the discriminant function is acceptable if CCR is more than some ratio (say 50%).

Classification using NN has several advantages compared to the classical methods. One advantage is the possibility of constructing nonlinear discriminant function for solving sophisticated classification problems. Using the proposed approach, a very complex nonlinear approximation functions can be built from simple components. The complexity of the discriminant function depends on the number of layers as well as the number of units in the hidden layer (hidden units). Another advantage is that the method does not put any constraints on population distribution or require the equivalence of the population variance-covariance matrices. The numerical examples support the suitability of the proposed approach in data classification.

The findings of the study draw several distinctions between neural network and classical classification: i) for linear classification, the performance of the two methods is equivalent, ii) for nonlinear classification, neural networks outperform the classical classification method and iii) the relative performance of the neural network increases as the level of nonlinearity increases.

REFERENCES

Arminger, G. and D. Enache (1995), "Statistical Models and Artificial Neural Networks", *In Proceedings of the 19th annual conference of the Gesellschaft fur classification e. V.*, University of Basel, H.-H. Bock and W. Polasek Editors , Springers , Germany.

Cox, Hinkly (1979), *Theoretical Statistics*, Chapman and Hall: London, U.K.

David, K. Hildebrand and Lyman oat (1991) *Statistical Thinking for Managers*, Fourth Edition. Wadsworth Publishing Company: London, England.

Devillers , J. and W. Karcher (1991), *Applied Multivariate Analysis in SAR and Environmental Studies*, Kluwer academic Publishers: London , U.K.

Elman, J. L. (1993), "Learning and Development in Neural Networks: the

Importance of Starting Small", *Cognition*, 48, pp.71-99.

Fausett, Laurene (1994), *Fundamentals of neural networks, Architectures, algorithms And Applications*, Prentice Hall Inc.: New York.

Hertz J. and et al. (1991), *Introduction to the theory of neural computation*, Lecture notes of the Santa-Fe Institute, vol. 1, Reading MA: Addison-Wesley.

Iman, Ronald L. and W. J. Conover (1983), *A Modern Approach to Statistics*, John Wiley & sons: New York, U.S.A.

Johnson, Richard A. and Dean, W. Wichern (1992), *Applied Multivariate Statistical Analysis*, Third Edition, Prentice Hall, Englewood Cliffs: New Jersey, U.S.A.

Matlab Reference Manual (1997), Version 5.3, Mathworks, U.S.A.

Naftaly, U.; Intrator, N. and Horn, D., (1997), "Optimal Ensemble Averaging of Neural Networks", *Network: Computation in Neural Systems*, 8, pp. 283-296.

Nerrand, O.; Roussel-Ragot, P.; Personnaz, L.; Dreyfus, G. and Marcos, S. (1993), "Neural Networks and Nonlinear Adaptive Filtering: Unifying Concepts and New Algorithms", *Neural Computation*, 5, pp.165-199.

Sahay, Surottam N. and et al. (1996), "Software Review", *The Journal of the Royal Economic Society*, Vol. 106, Isis. 439, pp 1820-1829.

Stevens, James (1992), *Applied Multivariate Statistics for the Social, Sciences*, Second Edition, Lawrence, Erlbaum Associates Publishers: Hillsdale, New Jersey, U.S.A.

Tsoi A.C. and Tan S. (1997), "Recurrent neural networks: A constructive algorithm and its properties", *Neurocomputing*, 15, pp.309-326.

Wernecke, K. D.; Possinger, K. and Kalb, G. (1995), "On the Validation of Classification Trees", *Proceedings of the 19th Annual Conference of the Gesellschaft fur classification e.V.*, University of Basel, March 8-10 , 1995 , Bock and Polasek Editors, Springer: U.S.A.

Zar, Jerrold H. (1984), *Bio-statistical Analysis*, Second Edition, Prentice – Hall International, Inc.: London, U.K.

APPENDIX

```
% ... Matlab Macro Program for Automatic Classification of Data ...
%... ...file name : train1394 ... output file oo1394.mat... ...
disp '... ...Examples of NN and Fisher Classification'
disp '... ...for Data Generated From Binomial Distribution '
diary ('oo1394')
clc;clear all;an=320;mm=an/40;aa(mm,4)=0;nn=15;
r1=binornd(nn,.5,an,1);r2=binornd(nn,.3,an,1);r3=.5*r1+.5*(r2);
for j =1:mm
   j
   n = 20+40*(j-1);
   n1=0;n2=0;n3=0;nn1=0;nn2=0;nn3=0;
   c1 = round(n/2);
   if n< 100
      c2=n-c1 ;
   else c2=50;
      end
   m = 5;k =3 ;d(k)=0;
   ss(k,1)= 0 ; ss0(k,1) = 0 ; ss1(k,1) = 0 ;st0(4)=0 ;st(k)=0;st1(4)=0;
   x1(k)=0;x2(k)=0;y1(c1)=0;y2(c1)=0;z3(c1) = 0 ;
   r03=r3(1:c1) ; z1=r1(1:c1) ; z2=r2(1:c1) ;
   %1... ... ... ... INITIALIZATAION OF DATA ... ... ... ...
    r  =(max(r03) - min(r03))/3;
    r0 = min(r03)+r;
    for i = 1 : c1
      if r03(i) < r0
         z3(i)  = 1 ;
         n1 = n1+1 ;
         x1(1)= x1(1)+z1(i);x2(1)=x2(1)+z2(i);
       elseif r3(i) > r0+r;
         z3(i)=3 ;
         n3=n3+1;
         x1(3)=x1(3)+z1(i);x2(3)=x2(3)+z2(i);
     else
         z3(i)= 2;
         n2=n2+1;
         x1(2)=x1(2)+z1(i);x2(2)=x2(2)+z2(i);
       end
   end
         % ... Fisher method for classification ... ... ...
   x01= ones(1,c1)*z1/c1;x02=ones(1,c1)*z2/c1;
   x11= x1(1)/n1 ; x21=x1(2)/n2 ; x31=x1(3)/n3;
   x12= x2(1)/n1 ; x22=x2(2)/n2 ; x32=x2(3)/n3;
   xxx= [x11 x12;x21 x22; x31 x32];
   xx1= [x11-x01;x21-x01;x31-x01];
   xx2= [x12-x02;x22-x02;x32-x02];
   for i = 1 : c1
      if r03(i) < r0
         y1(i)= z1(i)-x11;
         y2(i)= z2(i)-x12;
      elseif r3(i) > r0+r;
         y1(i)= z1(i)-x31;
         y2(i)= z2(i)-x32;
 else
         y1(i)= z1(i)-x21;
         y2(i)= z2(i)-x22;
      end
   end
   x =[ xx1   xx2] ;
```

```
  B0= transpose(x)*x;
  y=[y1 ; y2] ;
  yy= y*transpose(y);
  ww=inv(yy);
  ww1=ww*B0;
  [cv a] = eig(ww1);
  v01=cv(:,1);v02=cv(:,2);
  cc1=transpose(v01)*yy*v01;
  cc2=transpose(v02)*yy*v02;
  c01=sqrt((n-3)/cc1);
  c02=sqrt((n-3)/cc2);
  cv1=v01*c01;cv2=v02*c02;
  v=[cv1 cv2];
 % ... Proposed NN method for classification ... ... ...
  z3=transpose(z3);
  z=[z1 z2  z3];
  z0  = (0.8*(z-ones(c1,1)*min(z))./(ones(c1,1)*(max(z)-min(z))))+0.1;
  z01 = transpose(z0(:,1:2)); z02 = transpose(z0(:,3));
  k1='purelin' ; k2='logsig'  ; k3='tansig';
  [w1,b1,w2,b2]=initff(z01,m,k2,z02,k2) ;
  [w1 , b1 ,w2, b2,TE,TR]=trainbpx(w1,b1,k2,w2,b2,k2,z01,z02,[50,5000
  ,.001 ,.001]);
  %2... ... ... ...comparison and testing phase ... ... ... ...
  zz1=r1((c1+1):c1+c2,:); zz2=r2((c1+1):c1+c2,:); rr03=r3((c1+1):c1+c2,:);
  zzz=[zz1 zz2];
  v1 = zzz*v;
  v2 = xxx*v;
%1... ... ... ... INITIALIZATAION OF DATA ... ... ... ...
    rr =(max(rr03) - min(rr03))/3;
    rr0=min(rr03)+rr;
   for i = 1:c2
     if rr03(i)< rr0
        zz3(i) =1;nn1=nn1+1;
     elseif rr03(i) > rr0+rr
        zz3(i)=3;nn3=nn3+1;
     else
        zz3(i)= 2; nn2=nn2+1;
     end
  end
  zz3=transpose(zz3);
  zz=[zz1 zz2  zz3];
  zz0  = (0.8*(zz-ones(c2,1)*min(zz))./(ones(c2,1)*(max(zz)-
  min(zz))))+0.1;
  zz01 = transpose(zz0(:,1:2));
  zz02 = transpose(zz0(:,3));
  for i=1:c2
     d(1)= (v1(i,:)-v2(1,:))*transpose((v1(i,:)-v2(1,:)));
     d(2)= (v1(i,:)-v2(2,:))*transpose((v1(i,:)-v2(2,:)));
     d(3)= (v1(i,:)-v2(3,:))*transpose((v1(i,:)-v2(3,:)));
     [d1,dd1] = min(d);
     yf(i)=dd1;
     if     dd1  <  1.5          & zz3(i)==1  ;st1(1)=st1(1)+1;
     elseif dd1>= 1.5&dd1 < 2.5  & zz3(i)==2  ;st1(2)=st1(2)+1;
     elseif dd1>=2.5              & zz3(i)==3  ;st1(3)=st1(3)+1;
     else                                      ;st1(4)=st1(4)+1;
     end
     s   = simuff(zz01(:,i),w1,b1,k2,w2,b2,k2);
     s1= (((s- 0.1)*(max(zz3)-min(zz3)))/0.8 )+ min(zz3);
     y0(i)=s1;
     if       s1  <  1.5            & zz3(i)==1  ;st0(1)=st0(1)+1;
     elseif   s1 >= 1.5&s1 < 2.5    & zz3(i)==2 ;st0(2)=st0(2)+1;
     elseif   s1 >=2.5            & zz3(i)==3  ;st0(3)=st0(3)+1;
```

```
      else                                              ;st0(4)=st0(4)+1;
      end
   end
   ss1 = (st0(1)+st0(2)+st0(3))/c2;
   ss2 = (st1(1)+st1(2)+st1(3))/c2;
 %3...Results ... ...
 aa(j,:)=[j  c1  ss1  ss2];
% [zz3 transpose(yf) transpose(y0)]
 clear b* ; clear B* ; clear c* ;clear d* ; clear n* ;
 clear s* ;clear v* ; clear w* ;clear x* ; clear y* ;clear z*;
end
disp ' HIT RATE RESULTS Fisher class. &  Neural class.'
aa a1= mean(aa)
save o1394;
diary off
```