# Artificial Neural Network for Detection of Hepatitis using RBF NN

Dr.P.T.Karule[1]

[1]Professor in Department of Electronics Engineering, YCCE, Nagpur, ptkarule@gmail.com

**Abstract**—*Research into the problem of designing DSS (Decision Support System) for diagnosis of Hepatitis has been taken up as a challenging task for the neural networks. It appears from the literature review that for the MLP neural network trained with backpropagation, reported average classification accuracy was about 83.75% on the test instances. This paper investigates the design of an optimal classifier using a Radial Basis Function neural network. From the rigorous experimentation, it is shown that the proposed RBF NN works as an optimal DSS with an average classification accuracy of 93.77% and a specificity of 0.99325%. The area under the Receiver Operating Characteristics (ROC) Curve on test data set is found as 0.99494, which is very close to unity and this clearly confirms the excellent quality of the proposed classifier The results show that the proposed RBF neural network DSS clearly outperforms the MLP NN based one in various performance measures such as MSE, NMSE, correlation coefficients, area under the ROC curve and classification accuracy on different testing datasets even after attempting randomization of samples.*

**Keywords**— DSS, Multi-layer Perceptron Artificial Neural Network, back propagation algorithm, Radial Basis Function Neural Network, Receiver Operating Characteristics

## 1. Introduction

Diagnosis of the diseases is one of the most important problems in medicine. Medical diagnostics is quite difficult and visual task, which is mostly done by expert doctors. Two problems are the most common in the field of automatic diagnostic: the selection of necessary parameter set for right diagnostics and forming of steady and powerful algorithm which doesn't require long time to run [l].

Decision Support System (DSS) carries out pattern recognition, which is formally defined as the process whereby a received pattern/signal is assigned to one of a prescribed number of classes (categories). The goal of pattern-recognition is to build machines, called, classifiers, that will automatically assign measurements to classes. A natural way to make class assignment is to define the decision surface. The decision surface is not trivially determined for many real-world problems. The central problem in pattern-recognition is to define the shape and placement of the boundary so that the class-assignment errors are minimized. In classification problem, the task is to assign new inputs to one of a number of discrete classes or categories. Here, the functions that we seek to approximate are the probabilities of membership of the different classes expressed as functions of the input variables.

In recent times, neural networks have become a widely used method for designing DSS for disease-diagnosis. The paper is organized as follows. First the optimal MLP NN based DSS is designed to diagnose the given Hepatitis data base. Later, the RBF NN based DSS is developed for the binary classification task. Next, a comparison between these two classifiers is carried out on the basis of their validation performance with respect to the performance measures such as MSE, NMSE (normalized mean square error), *r* (correlation coefficients), percent classification accuracy and area under ROC curve on the testing instances. Finally, the conclusions are discussed with a recommendation to use the proposed RBF NN based classifier as a DSS.

## II. Hepatitis Database

Hepatitis disease is inflammation and damage to hepatocytes in the liver and can he caused by infections with viruses, bacteria, fungi, exposure to toxins such as alcohol and autoimmunity. Usually, the liver can handle significant amounts of damage, and the liver function is still effective. However, it will decline if the disease is not fully controlled at an early stage. Hepatitis may be acute with recovery within six months. However, it also may lead to death if significant decomposition occurs. In addition, there are five types of hepatitis: hepatitis A, hepatitis B, hepatitis C, hepatitis D and hepatitis G. [4]

Hepatitis database [3] contains 154 samples. All samples have 19 attributes that are shown as follows:

1- Age : ranges from 20 to 72 years old;
2- Sex : male or female, and is represented by 1 or 2 respectively;
3- Steroid: value 1 for no, value 2 for yes;
4- Antiviral: value 1 for no, value 2 for yes;
5- Fatigue: value 1 for no, value 2 for yes;
6- Malaise: value 1 for no, value 2 for yes;
7- Anorexia: value 1 for no, value 2 for yes;
8- Liver Big: value 1 for no, value 2 for yes;
9- Liver Firm: value 1 for no, value 2 for yes;

10- Spleen Palpable: value 1 for no, value 2 for yes;
11 - Spiders: red capillary tufts in the skin that blanches on pressure, and is represented By 1 or 2 respectively;
12- Ascites: accumulation of fluid in the abdominal cavity, and is represented by 1 or 2 respectively;
13- Varices : dilated veins, and is represented by 1 or 2 respectively;
14- Bilirubin: a bile pigment cleared from the blood by the liver;
15- Alkaline phosphates: protein found in bile duct cell membranes;
16- Aspartate transaminase (SCOT): enzymes that catalyze protein transformations within hepatocytes;
17- Albumin: a protein in the serum that transports substances such as drugs and Prevents leakage of fluid into the surrounding tissues;
18- Pro-time: the pro-thrombin time in serum;
19- Histology: value 1 for no, value 2 for yes;

There are two classes of the hepatitis disease: death and life. First class has 32 instances and second class has 122 instances. Hepatitis database constitutes 154 samples with 19 continuous-valued inputs and one output denoting the class of the instance. Two different data partitions are used with different randomization. In the first case, the first 110 samples are used for training; the next 44 samples for testing and DSS comparison purpose. In the second set after randomizing the 120 samples are used for training and the 34 samples for testing of DSS.

**III. Design of a MLP NN based DSS**

MLPs are feed-forward neural networks trained with the standard back-propagation algorithm. They are supervised networks, so they need a desired response to be trained. A distinct, though related issue is whether a MLP network can classify correctly a given set of data points which have been labeled as belonging to one of two classes (a dichotomy). It is shown that a network having a single layer of threshold units could classify a set of points perfectly if they were linearly separable [5]. This would always be the case if the number of data points was at most equal to $d+1$, where $d$ is the dimensionality of the input space. It is shown that for a set of *N* data points, a two-layer network of threshold units with $N-1$ units in the hidden layer could exactly separate an arbitrary dichotomy [6]. This result has been improved by showing that for *N* points in general position (i.e., excluding exact degeneracies) in *d*-dimensional space; a network with $[N/d]$ hidden units in a single hidden layer could separate them correctly into two classes [7]. Here $[N/d]$ denotes the smallest integer, which is greater than or equal to $[N/d]$.

When a NN has been trained, the next step is to evaluate it. This is done by a standard method in statistics called *independent validation*. This method divides the available data into a training set and a test set. The entire data set is usually randomized first. The training data are next split into two partitions; the first partition is used to update the weights in the network, and the second partition is used to assess (or cross-validate) the training performance. The test data are then used to assess how well the network has generalized. The learning and generalization ability of the estimated NN based classifier is assessed on the basis of certain performance measures such as MSE, NMSE, correlation coefficients, area under the ROC curve, and the rate of correct classification. Nevertheless, for classifiers, area under the ROC curve and the percentage of correct classification are the most crucial parameters.

Since it is very likely that one ends up in a "bad" local minimum, the network should be trained a couple of times (typically at least three times), starting from different initial weights. NeuroSolutions (version 5) and Neural Network Toolbox for MATLAB (version 7.0) are specifically used for obtaining results.

As has been pointed out in earlier discussion, a MLP NN is chosen as a classifier. In order to design a proper architecture of the MLP NN model; an experiment is conducted where the number of hidden units is varied gradually from 1 to 20. It is found that the performance of the selected model is optimal for 17 neurons in the hidden layer with regard to the MSE, NMSE, correlation coefficients, area under the ROC curve, and percent classification accuracy for the testing data set. When we attempted to increase the number above 17 in the hidden layer, the performance of the classifier was not seen to improve. Similarly, by increasing the number of hidden layers, the performance of the classifier is not seen to improve significantly. On the contrary, it takes too long for training because of higher complexity of the classifier. As there are 19 numeric inputs and one symbolic output (translated into two numeric-valued outputs, where "live" is 0 1 and "death" is 1 0) for the given system, the number of input and output processing elements is chosen as nineteen and two, respectively. The total number of free parameters (connection weights) for this model is 376. Now the NN model (19-17-2) is trained (three times with different weight initialization) with 1000 iterations of the static backpropagation algorithm with momentum term. For classification, the output processing element must be nonlinear. Here, as the outputs are 1 and 0, the operating point of the hidden processing elements is normally driven to saturation. However, for comparison, linear transfer function is also considered in the output layer along with the other nonlinear transfer functions such as tanh, lintanh, and softmax. The softmax may be used as the output of any MLP to allow interpretation of the output as a probability, as normally is the case in classification. Since the ultimate objective of a pattern classifier is to achieve an acceptable rate of correct classification, this criterion is used to judge when the variable parameters of the MLP (used as a pattern classifier) are optimal. Variation of average of minimum MSEs with respect to PEs in hidden layer is graphed in Fig. 1.

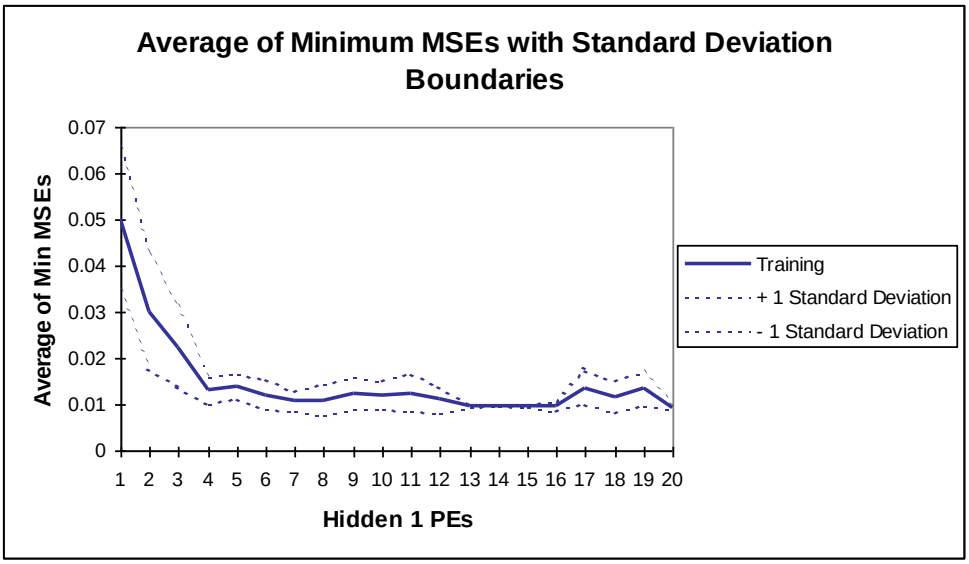


Fig. 1 Average of Minimum MSEs versus Number of PEs in the hidden

It is thus demonstrated that the best network should have *17* neurons in the hidden layer. In addition, the transfer function of neurons in hidden layer should be hyperbolic-tangent (tanh) and that for the output layer should be soft-max and the network should be trained using back-propagation algorithm for the best performance. The optimal parameter settings for MLP NN based classifier are displayed in Table 1.

Table 1: Optimal Parameters of MLP NN Classifier

Max. epochs = 1000, supervised learning, weight update in batch mode, No. of Input PEs=19

| **S. N.** | **Parameter** | **Hidden Layer#1** | **Output Layer** |
|---|---|---|---|
| 1 | Processing Elements | 17 | 2 |
| 2 | Transfer Function | tanh | Soft-max |
| 3 | Learning rule | Momentum | Delta Bar Delta |

## IV. Performance measures

### *MSE* (Mean Square Error):

The formula for the mean squared error is:

$$MSE = \frac{\sum_{j=0}^{P}\sum_{i=0}^{N}\left(d_{ij} - y_{ij}\right)^2}{NP} \quad (1)$$

Where $P$ = number of output processing elements, $N$ = number of exemplars in the data set, $y_{ij}$ = network output for exemplar $i$ at processing element $j$, $d_{ij}$ = desired output for exemplar $i$ at processing element $j$.

### *NMSE* (Normalized Mean Square Error):

The normalized mean squared error is defined by the following formula:

$$NMSE = \frac{P\,N\,MSE}{\sum_{j=0}^{P}\frac{N\sum_{i=0}^{N} d_{ij}^{\,2} - \left(\sum_{i=0}^{N} d_{ij}\right)^2}{N}} \quad (2)$$

Where $P$ = number of output processing elements, $N$ = number of exemplars in the data set, $MSE$ = mean square error, $d_{ij}$ = desired output for exemplar $i$ at processing element $j$.

***r* (correlation coefficient):**

The size of the mean square error (MSE) can be used to determine how well the network output fits the desired output, but it doesn't necessarily reflect whether the two sets of data move in the same direction. For instance, by simply scaling the network output, we can change the MSE without changing the directionality of the data. The correlation coefficient (*r*) solves this problem. By definition, the correlation coefficient between a network output *x* and a desired output *d* is:

$$r = \frac{\frac{\sum_i \left(x_i - \bar{x}\right)\left(d_i - \bar{d}\right)}{N}}{\sqrt{\frac{\sum_i \left(d_i - \bar{d}\right)^2}{N}}\sqrt{\frac{\sum_i \left(x_i - \bar{x}\right)^2}{N}}}$$

$$\text{where} \quad \bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \quad \text{and} \quad \bar{d} = \frac{1}{N}\sum_{i=1}^{N} d_i \qquad (3)$$

The correlation coefficient is confined to the range [-1, 1]. When $r = 1$ there is a perfect positive linear correlation between *x* and *d*, that is, they covary, which means that they vary by the same amount. When $r = -1$, there is a perfectly linear negative correlation between *x* and *d*, that is, they vary in opposite ways (when *x* increases, *d* decreases by the same amount). When $r = 0$ there is no correlation between *x* and *d*, i.e. the variables are called uncorrelated. Intermediate values describe partial correlations. For example, a correlation coefficient of 0.88 means that the fit of the model to the data is reasonably good.

**Receiver Operating Characteristics (ROC):**

Receiver Operating Characteristic (ROC) matrices are used to show how changing the detection threshold affects detections versus false alarms. If the threshold is set too high then the system will miss too many detection. Conversely, if the threshold is set too low then there will be too many false alarms. The percentage of detections classified correctly (Sensitivity or true positive rate) is plotted against the percentage of non-detections incorrectly classified as detections (i.e. false alarms or false positive rate) as a function of the detection threshold. ROC enables the user to evaluate a model in terms of the trade-offs between sensitivity and specificity. It is the best way to evaluate a detector. The performance of classification for test data set is assessed by calculating the area under the ROC curve ($A_Z$). It is noticed that the values for $A_z$ range from 0.5 for chance to 1.0 for a perfect classifier.

**Confusion Matrices:**

A confusion matrix is a simple methodology for displaying the classification results of a network. The confusion matrix is defined by labeling the desired classification on the rows and the predicted classifications on the columns. For each exemplar, a 1 is added to the cell entry defined by (desired classification, predicted classification). Since we want the predicted classification to be the same as the desired classification, the ideal situation is to have all the exemplars end up on the diagonal cells of the matrix (the diagonal that connects the upper-left corner to the lower right).

**V. DSS using RBF NN**

A design of neural network can be viewed as a curve-fitting (approximation) problem in a high-dimensional space. According to this viewpoint, learning is equivalent to finding a surface in a multidimensional space that provides a best fit to the training data, with the criterion for "best fit" being measured in some statistical sense. Correspondingly, generalization is equivalent to the use of this multidimensional surface to interpolate the test data. Such a viewpoint is the motivation behind the method of radial-basis functions in the sense that it draws upon research work on traditional strict interpolation in a multidimensional space. In the context of a neural network, the hidden units provide a set of "functions" that constitute an arbitrary "basis" for the input patterns (vectors) when they are expanded into the hidden space; these functions are called radial-basis functions. Radial-basis functions were first introduced in the solution of the real multivariate interpolation problem. [11,12].

The construction of a RBF network, in its most basic form, involves three layers with entirely different roles. The input layer is made up of source nodes (sensory units) that connect the network to its environment. The second layer,

the only hidden layer in the network, applies a nonlinear transformation from the input space to the hidden space; in most applications the hidden space is of high dimensionality. The output layer is linear, supplying the response of the network to the activation pattern (signal) applied to the input layer. A mathematical justification for the rationale of a nonlinear transformation followed by a linear transformation may be traced back to an early paper by Cover [8]. According to this paper, a pattern-classification problem cast in a high-dimensional space is more likely to be linearly separable than in a low-dimensional space—hence the reason for frequently making the dimension of the hidden space in an RBF network high. Another important point is the fact that the dimension of the hidden space is directly related to the capacity of the network to approximate a smooth input-output mapping [9,10]; the higher the dimension of the hidden space, the more accurate the approximation will be.

Different initial conditions are tried to make sure that we are really converging to the absolute minimum. Therefore, the network is run at least three times to gauge performance. The training of RBF constitutes 100 epochs in unsupervised learning mode for setting the centers and width of the Gaussians and at least 1000 epochs in the supervised learning mode to compute the connection weights in the output layer. The supervised learning may terminate earlier if the minimum specified error threshold of 0.01 is reached earlier. An exhaustive experimental study has been carried out to design the optimal parameters of RBF NN based classifier. The following table indicates the average values of each parameter computed over "live" and "death" instances. Each parameter is varied with the setting of other parameters at their nominal default values. For each case, the RBF NN is run three times with different weight-initializations with the specified training epochs as shown in Table.

| **S. N.** | **Parameter of RBF NN** | |
|---|---|---|
| 1 | No. of Clusters | 120 |
| 2 | Transfer Function | Softmax |
| 3 | Supervised learning rule | DeltaBarDelta |
| 4 | Competitive learning metric | Euclidean |
| 5 | Unsupervised learning rule | Conscience-full |

**Table2. Results of testing on different NN (Performance Matrix)**

| NN model | Performance index | Test on data set1 (%) | | Test on data set2 (%) | |
|---|---|---|---|---|---|
| ↓ | ↓ | Live | Death | Live | Death |
| MLP | Classification accuracy | 95 | 83.33 | 83.87 | 100 |
| | MSE | 0.1089 | 0.1066 | 0.1108 | 0.1078 |
| | NMSE | 0.759 | 0.73 | 0.9269 | 0.9019 |
| | r | 0.628 | 0.628 | 0.6616 | 0.6616 |
| | Area under ROC | 0.990476 | | 1.0 | |
| | % ERROR | 9.12 | | 8.89 | |
| RBF | Classification accuracy | 95 | 83.33 | 96.77 | 100 |
| | MSE | 0.0654 | 0.06409 | 0.04395 | 0.04277 |
| | NMSE | 0.5773 | 0.565 | 0.3675 | 0.3576 |
| | r | 0.6787 | 0.6787 | 0.827 | 0.827 |
| | Area under ROC | 0.989899 | | 0.9722 | |
| | % ERROR | 7.53 | | 5.113 | |

**ROC of MLP DSS on Test dataset 1 , 2**

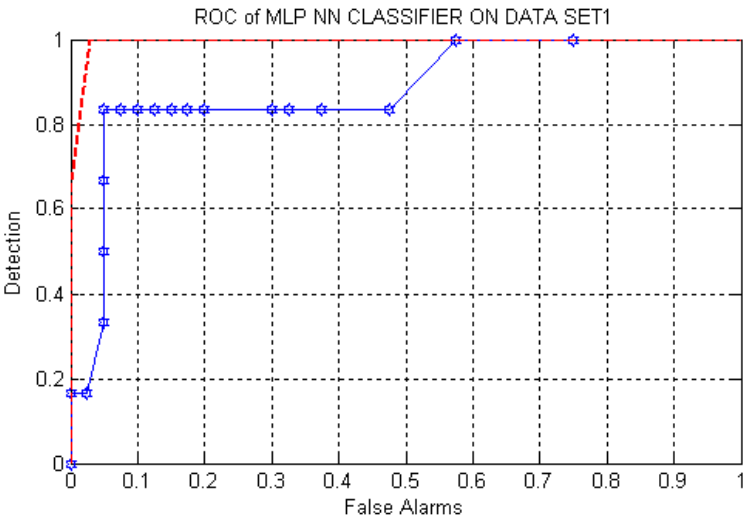


Fig. 1 ROC curve for MLP NN based classifier on test data set1 Area under ROC = 0.990476 and Area under convex hull of an ROC curve = 0.995238

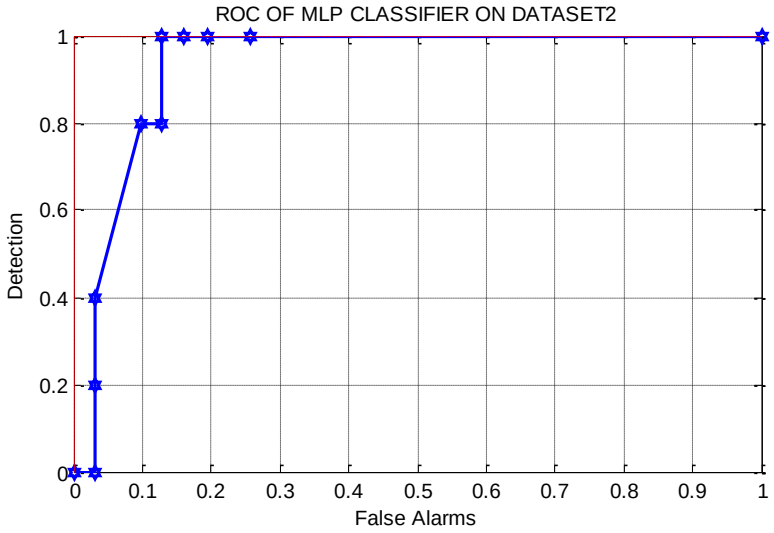


Fig. 2 ROC curve for MLP NN based classifier on test data set2 Area under ROC = 1.0 and Area under convex hull of an ROC curve = 1.0

**ROC of RBF DSS on Test dataset 1 , 2**

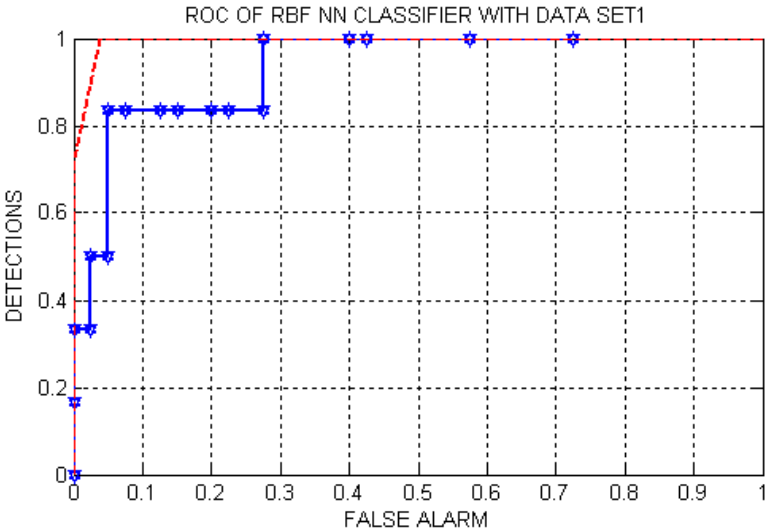


Fig. 3 ROC curve for RBF NN based classifier on test data set1 Area under ROC = 0.989899 and Area under convex hull of an ROC curve = 0.994949

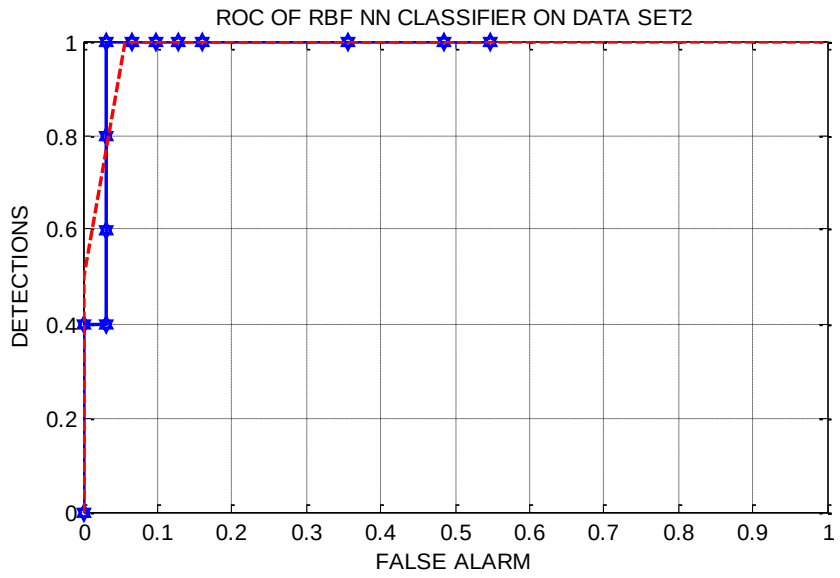


Fig. 1 ROC curve for RBF NN based classifier on test data set2 Area under ROC = 0.9722 and Area under convex hull of an ROC curve = 0.9861

Careful inspection of all tables as well as figures show that the RBF NN based classifier satisfies almost all the essential qualities and tests of a near-perfect (near-optimal) classifier up to the end-user's expectations. More importantly, its performance is seen to be consistently good. For the classification of radar returns from the ionosphere, the decision boundaries formed by the RBF NN classifier are seen to be more accurate than those formed by MLP NN classifier.

## VI. CONCLUSION

Radial-basis function NN based DSS operates as an excellent classifier for the given task under study. When the trained RBF NN based DSS is examined on the testing instances, it produces on an average 99.596775 % correct classifications and the area under the ROC curve is computed as 0.999403, which is indeed very close to unity. This performance is very close to that of a perfect classifier (average classification accuracy = 100 % and the area under the ROC curve = 1.0). Even after randomizing the data, the RBF NN based classifier is consistently seen to maintain its reasonable performance. It is worthwhile to notice that the RBF NN has consistently performed elegantly as a near-optimal classifier even after repeating the simulation experiments a number of times on different data partitions. From the results of the ROC analyses, we observe a reasonable trade-off between specificity and sensitivity (the percentage of detections classified correctly) with different data partitions used for training and testing.

Both MLP and RBF classifier are seen to deliver the best performance on soft-max activation function used for the PEs of the output layer. This is obvious because for classification, the output processing elements must be nonlinear for generation of arbitrary complex decision regions. All results suggest that as a DSS, RBF NN outperforms the MLP NN and its diagnostic ability is remarkable.

## REFERENCES


[1] Jajoo, R.; Mital, D., Prediction of hepatitis C using artificial neural network. *In Proc. Control, Automation, Robotics and Vision, 2002. ICARCV 2002. 7th International Conference on Volume 3, 2-5 Dec. 2002 Page(s):1545 - 1550 vol.3*

[2] Ozyilmaz, L.; Yildirim, T., Artificial neural networks for diagnosis of hepatitis disease. *Neural Networks, 2003. Proceedings of the International Joint Conference on Volume 1, 20-24 July 2003 Page(s):586 - 589 vol.1*

[3] http:/www.ics.uci.edu/pub/m1-repos/machine-learning-databases/2003

[4] http://www.hepatitis-central.com/hcv/whatis/whatishcv.html

[5] Duda, R. O., and Hart, P. E., (1973). "Pattern Classification and Scene Analysis," New York: John Wiley Nilsson, N. J., (1965).

[6] *Learning Machines: Foundations of Trainable Pattern-Classifying Systems*, New York: McGraw-Hill.

[7] Baum, E. B. (1988). "On the Capabilities of multilayer perceptrons," Journal of Complexity, vol. 4, pp. 193-215.

[8] Powell M. J. D., (1985). "Radial basis functions for multivariable interpolation: A review," *IMA Conference on Algorithms for the Approximation of Functions and Data*, pp. 143-167, RMCS, Shrivenham, England.

[9] Light W., (1992). "Ridge functions, sigmoidal functions and neural networks," in E. W. Cheney, C. K. Chui, and L. L. Schumaker, eds, Approximation Theory VII, pp. 163-206, Boston: Academic Press.

[10] Cover T. M., (1965). "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," IEEE Transactions on Electronic Computers, vol. EC-14, pp. 326-334.