Rough Set Theoretic OptimizedSingle Hidden Layer Artificial Neural Networks based Classification System

[1]S.Surekha
[1]Assistant Professor
[1] Department of Computer Science and Engineering,
[1] JNTUK-University College of Engineering Vizianagaram, Andhra Pradesh,India

Abstract: Artificial Neural Networks (ANNs) are the digitized model of the human brain and can be effectively used in the context where it is very difficult for humans to detect the underlying complex non-linear relationships in the data. In this paper, a feed forward single hidden layer neural network is trained using Back propagation algorithm for data classification. In real world, patterns are characterized by many features and not all features are relevant for a particular task and using all the features for training an ANN often reduces the learning rate. Hence, to increase the learning speed of the ANN, the most popular Rough Set Theory(RST) algorithms have been used to determine the most significant features and then training ANN only on the relevant features reduce the time complexity of the Neural Networks as well as helps in increasing the generalization ability. The trained ANN is 10-fold cross validated by conducting experiments on the datasets taken from UCI ML repository and the results revealed that the removal of superfluous features in the training data enhances the performance of ANN in increasing the learning speed and classification accuracy.

***IndexTerms* - Artificial Neural Network, Feed Forward NN, Multi-Layer Neural Network, Back Propagation Algorithm, Rough Set Theory;**

I. INTRODUCTION

In recent years, Artificial Neural Networks (ANNs)[1,2,3] are being widely used in many applications involving Classification, Recommendation and Prediction tasks. Artificial Neural Networks[4,5] are the mathematical algorithms generated by simulating the functioning of biological neurons in human brain. The basic computational unit of ANN is the artificial neuron and an ANN consists of a number of artificial neurons organized as structured layers, each node is composed of several artificial neurons [6]. Depending on the number of layers, ANNs are broadly categorized into Single and Multilayer neural networks. In Single-layer ANN, only Input layer and Output layers exist, and the NN is trained by giving the training data to the Input layer. Even though the single-layer ANN is simple, it fails to identify the non-linear relationships in the submitted training data and hence, Multi-layer ANN has been evolved. In multi-layer ANN[7], a hidden layer is introduced in between the Input and Output layers. For solving simple real world problems single hidden layer is sufficient, but to solve a complex problem multiple hidden layers are required. The number of nodes in the Input, hidden, and output layer are determined based on the training data. The NN is trained by submitting the values of the features given in the training data to the nodes in the input layer and after several computations, compare the obtained output at the output layer with the actual output. If the obtained result is not acceptable for a predefined threshold, then the neural network is iteratively trained by updating the weights of the connecting nodes using the Back Propagation weight updating algorithm. The performance of NN[1] depends on various parameters like Activation function, number of hidden nodes, weight updating algorithm, and finally on the number of training iterations also. In universe, patterns are represented by a number of features and sometimes all the features are not required for a particular classification and the existence of the irrelevant features simply increases the training time of the NN. Hence, there is a great need to identify the relevant features so as to decrease the training time of the NN[8,9]. So, in this paper, RST based reduct generation algorithm is used to eliminate irrelevant features in the training data.

As there is a great demand for predictive mining models in health care for disease diagnosis [10.11], the trained ANN is tested on the medical datasets taken from the UCI repository and a series of tests are conducted to measure the classification accuracy of the build model.

The rest of the paper is organized as follows, Section 2 explains the RST based Reduct Generation Algorithm, Section 3 illustrates the construction of Artificial Neural Network using Back propagation algorithm and Section 4 gives the experimental analysis Finally, Section 5 concludes the paper.

2. Rough Set Theory based Reduct Generation Algorithm

Rough Set Theory (RST)[12], introduced by Z.Pawlak is a mathematical approach to deal with vagueness and uncertainty in data The basic concepts of RST can be found in[13,14]. The RST based Improved Quick reduct algorithm[14] generates the reduct based on the degree of dependency. The degree of dependency of an attribute is defined as,

$$\gamma_C(D) = \frac{|POS\ C(D|)}{|U|} \qquad (1)$$

Where, $POS_C(D)$ is the positive region.

Initially, calculate the dependency degree of all conditional attributes and the attribute with highest dependency degree is chosen first and then subsequently select the attribute with next highest dependency degree from the remaining attributes and repeat the process of adding remaining attributes to the reduct set until the dependency degree of the reduct set is equal to the dependency degree of the full set of attributes.

2.1Example

To clearly illustrate the RST based reduct generation algorithm a sample dataset with four conditional attributes {A1,A2, A3, andA4} and a Class attribute {C} is considered and is given in Table 1.

Table 1 Sample dataset

U	A1	A2	A3	A4	Class
1	1	1	1	3	1
2	1	1	2	1	1
3	2	1	1	1	1
4	2	2	1	3	2
5	1	2	1	2	2
6	2	2	3	3	3

The equivalence classes for the data represented in Table 1 are obtained as,

U/A1={ {1,2,5}, {3,4,6}} , U/A2={ {1,2,3}, {4,5,6}}, U/A3={ {1,3,4,5}, {2},{6}}, U/A4={ {2,3}, {5},{1,4,6}}

U/{A1,A2,A3,A4} = { { 1},{2},{3},{4},{5},{6}} and U/C={ {1,2,3}, {4,5},{6}}

Calculating the dependency degrees of each attribute using Eq.(1) are obtained as,

$\gamma_{\{A1,A2,A3,A4\}}$ (C) = $|POS_{\{A1,A2,A3,A4\}}(C)| / |U|$ = = |{1,2,3,4,5,6}|/6 = 6/6=1

γ_{A1} (C) = $|POS_{A1}(C)| / |U|$ = 0/6 = 0

γ_{A2} (C) = $|POS_{A2}(C)| / |U|$ = | {1,2,3}|/6 = 3/6=0.5

γ_{A3} (C) = $|POS_{A3}(C)| / |U|$ = = | {2,6}|/6 = 2/6=0.33

γ_{A4} (C) = $|POS_{A4}(C)| / |U|$ = = | {2,3,5}|/6 = 3/6=0.5

The attributes A2 and A4 have the same dependency measure. So, randomly select any one of them as the initial Reduct set and then subsequently add the attributes with next highest dependency degree.

So, adding A4 to the existing Reduct set,

U/{A2,A4} = { {1}, {2,3},{4,6}, {5} } and $\gamma_{\{A2,A4\}}$ (C) = $|POS_{\{A2,A4\}}(C)| / |U|$ = | {1,2,3,5}|/6 = 4/6=0.66

U/{ A2,A3,A4} = { { 1},{2},{3},{4},{5},{6}} and $\gamma_{\{A2,A3,A4\}}$ (C) = $|POS_{\{A2,A3,A4\}}(C)| / |U|$ = |{1,2,3,4,5,6}|/6 = 6/6=1

Now, the combined dependency degree of A2,A3 and A4 is equal to the dependency degree for full attribute set. So, terminate the process and the reduct set is { A2,A3,A4}. For the sample dataset represented in Table1, the attributes A2, A3, and A4 are identified as significant attributes i.e., for a simple dataset with only 4 features and 6 records the RST based Improved Quick reduct algorithm generated only 3 attributes as the relevant attributes. The beauty of the RST based approach is that it identifies inconsistencies in the dataset and the Improved Quick Reduct algorithm could able to identify the most prominent features irrespective of the underlying machine learning algorithm. So, the results obtained by the RST based algorithm can be submitted for any machine learning algorithm.

3. Prepare Multi-Layer Artificial Neural Network

This section clearly illustrates the training of a Single hidden layer Feed forward Multi-Layer Artificial Neural Network using the most popular Back propagation algorithm. The Sigmoid and Sigmoid gradient functions are used as the activation functions. Training a Multi-Layer ANN is a two step process.

1. Construction of Feed forward ANN
2. Training using Back propagation algorithm

3.1 Construction of Feed Forward ANN

The number of nodes in the input layer is determined by the number of conditional attributes in the training data and an additional bias node. The number of nodes in the output layer is same as that of the number of target classes in the training data. The number of hidden nodes is computed by the Constructive algorithm [15].

The Sigmoid function[16] is defined as,

$$\sigma(y) = \frac{1}{1+e^{-y}} \qquad (2)$$

The Sigmoid Gradient function[16] is defined as,

$$\sigma'(y) = \sigma(y)\big(1-\sigma(y)\big) \qquad (3)$$

The regularized cost function[16] for training ANN is defined as,

$$Cost = \frac{1}{P}\sum_{i=1}^{P}\sum_{j=1}^{O}\left[-AO_{ji}\, log\left(OP_{ji}\right) + (1-AO_{ji})(1-log\left(1-OP_{ji}\right))\right] + \frac{\lambda}{2P}\left[\sum_{k=1}^{H}\sum_{l=1}^{P}((WH_{kl})^2) + \sum_{k=1}^{O}\sum_{l=1}^{H}((WH_{kl})^2)\right] \quad (4)$$

Where, P is the number of training patterns, O is the number of classes, H is the number of hidden nodes,
AO_{ji} is the Actual output of the ith input pattern at jth output node, and
OP_{ji} is the Obtained output of the ith pattern at jth output node and can be obtained as,

$$OP_{ji} = \sigma\left(\sum_{m=1}^{H}\left(\sigma((X_i)^T * WH_{im}) * WO_{jm}\right)\right) \qquad (5)$$

Where, X_i is the ith training pattern, WH_m is the weights connecting the ith input layer to the mthhidden layer,
WO_{jm} is the weights connecting the mth hidden layer to the jth output layer.

3.2 Training ANN using Back propagation algorithm

The most popular Back propagation algorithm[16] is used to train the ANN by updating weights as per the gradients calculated by Delta rule and is given below.

Back Propagation Algorithm

Input: **TD –** Training Dataset
Output: **WH-** Weights of Hidden Layer and **WO-** Weights of OutputLayer

Step 1: For each input pattern in TD,
Step 2: Compute the Activation functions at Hidden layer (OH) and Output layer (OP)
Step 3: Add a ***'+1'*** bias term to the obtained vectors
Step 4: Back Propagation of Error from Output Layer to Hidden Layer

For each Output unit in Output Layer,
Compute the Error, i.e., the difference between the Actual Output and Obtained Output as Delta3,

$$Delta_3 = AO - OP$$

Step 5: Back Propagation of Error from Hidden Layer to Input Layer

Back Propagate the Error to Input Layer by computing Delta2 matrix as,

$$Delta_2 = ((WO)^T * Delta_3) * \sigma'(IO)$$

Step 6: Accumulate the Theta Gradients to propagate the Error back to Input Layer as,

$$WH^G = WH^G + Delta_2 * (IP)^T$$
$$WO^G = WO^G + Delta_3 * (OH)^T$$

End for

Step 7: Obtain the Theta Gradients by dividing the accumulated Theta Gradients by the number of patterns trained

$$WH^G = \frac{1}{m} * (WH^G)$$
$$WO^G = \frac{1}{m} * (WO^G)$$

Step 8: Update the Weights of the connecting nodes as,

$$WH = WH + WH^G$$
$$WO = WO + WO^G$$

Step 9: return WH and WO

3.3Example

Let us consider the same dataset represented by Table 1. The number of nodes in the Input, Hidden and Output layer can be determined as follows,

No. of nodes in Input Layer = number of Conditional Attributes +1(bias node) =5

No. of nodes in Output Layer = number of Class Labels = 3

No. of Hidden nodes are obtained as 5

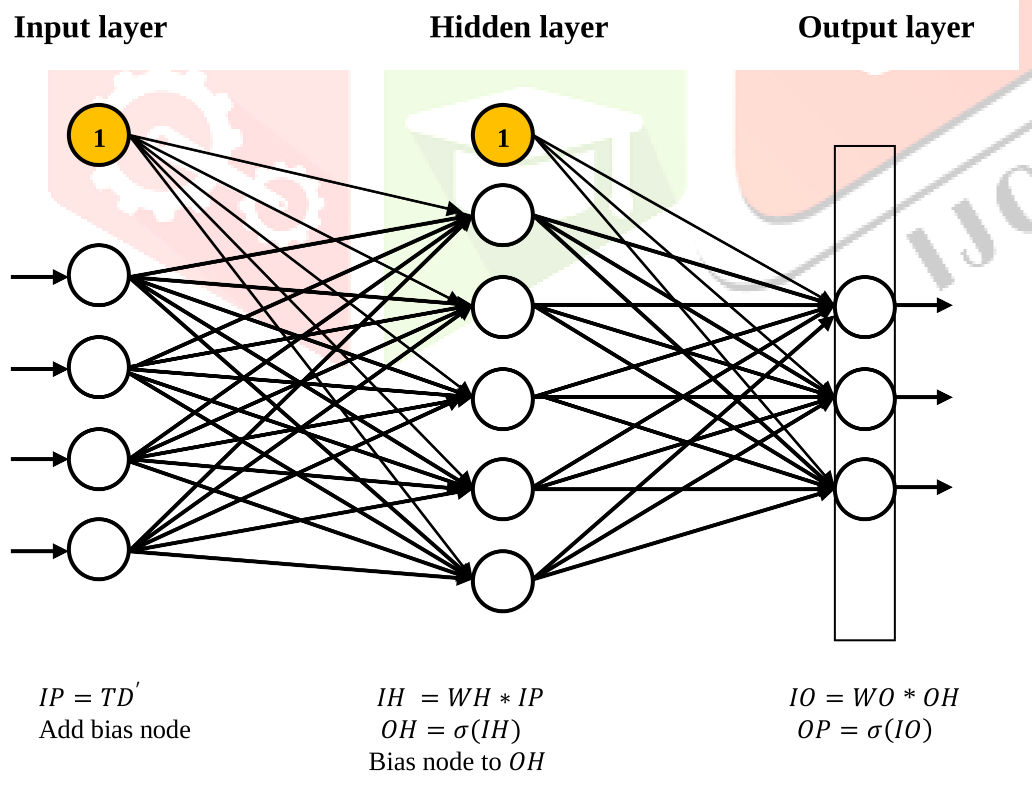


Fig. 1. Architecture of Fully connected Multi-Layer Artificial Neural Network for the sample dataset given in Table 1.

WH: Weight matrix representing the weights connecting Input nodes to Hidden nodes
WO: Weight matrix representing the weights connecting Hidden nodes to Output nodes
Number of Hidden nodes are obtained as 5 and hence the sizes of WH and WO are as follows,

$$Size\ of\ WH = no.of\ nodes\ in\ Input\ Layer + 1\ \text{X}\ no.of\ nodes\ in\ Hidden\ Layer = 5\text{ X }5$$

$$Size\ of\ WO = no.of\ nodes\ in\ Output\ Layer\ \text{X}\ no.of\ nodes\ in\ Hidden\ Layer + 1 = 3\text{ X }6$$

Initialize weights WH and WO with the random values in the range of 0 and 1,

$$WH = \begin{bmatrix} 0.8212 & 0.7317 & 0.7447 & 0.6256 & 0.4868 \\ 0.0154 & 0.6477 & 0.1890 & 0.7802 & 0.4359 \\ 0.0430 & 0.4509 & 0.6868 & 0.0811 & 0.4468 \\ 0.1690 & 0.5470 & 0.1835 & 0.9294 & 0.3063 \\ 0.6491 & 0.2963 & 0.3685 & 0.7757 & 0.5085 \end{bmatrix}$$

$$WO = \begin{bmatrix} 0.5108 & 0.6443 & 0.5328 & 0.8759 & 0.5870 & 0.4709 \\ 0.8176 & 0.3786 & 0.3507 & 0.5502 & 0.2077 & 0.2305 \\ 0.7948 & 0.8116 & 0.9390 & 0.6225 & 0.3012 & 0.8443 \end{bmatrix}$$

Iteration 1:

TD = 6 instances of the Training dataset and C = class labels of training dataset

$$TD = \begin{bmatrix} 1 & 1 & 1 & 3 \\ 1 & 1 & 2 & 1 \\ 2 & 1 & 1 & 1 \\ 2 & 2 & 1 & 3 \\ 1 & 2 & 1 & 2 \\ 2 & 2 & 3 & 3 \end{bmatrix} \text{ And } \quad C = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 2 \\ 2 \\ 3 \end{bmatrix}$$

After adding bias node to the Input Layer, the values of IP are obtained as follows,

$$IP = TD' = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ 1 & 1 & 2 & 2 & 1 & 2 \\ 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 2 & 1 & 1 & 1 & 3 \\ 3 & 1 & 1 & 3 & 2 & 3 \end{bmatrix}$$

Input to the Hidden Layer is denoted as WH and is calculated as given below,

$$IH = WH * IP$$

$$IH = \begin{bmatrix} 4.3836 & 4.0356 & 4.1417 & 5.8600 & 4.6415 & 7.1112 \\ 2.9400 & 2.8484 & 2.7159 & 3.7767 & 2.6931 & 5.3371 \\ 2.6022 & 1.7897 & 2.1595 & 3.7399 & 2.8422 & 3.9021 \\ 2.7478 & 3.0646 & 2.6822 & 3.4783 & 2.6250 & 5.3371 \\ 3.6151 & 3.3738 & 2.8944 & 4.2799 & 3.4751 & 5.8313 \end{bmatrix}$$

Output of the Hidden Layer is denoted as IP and is obtained by applying Sigmoid activation function i.e., using Eq.(2) on the Input IH and adding the bias node weight '+1' as shown below,

$$IP = \sigma(IH)$$

$$OH = \begin{bmatrix} 1.0000 & 1.0000 & 1.0000 & 1.0000 & 1.0000 & 1.0000 \\ 0.9876 & 0.9826 & 0.9843 & 0.9971 & 0.9904 & 0.9991 \\ 0.9497 & 0.9452 & 0.9379 & 0.9776 & 0.9366 & 0.9952 \\ 0.9310 & 0.8568 & 0.8965 & 0.9767 & 0.9449 & 0.9802 \\ 0.9397 & 0.9554 & 0.9359 & 0.9700 & 0.9324 & 0.9952 \\ 0.9737 & 0.9668 & 0.9475 & 0.9863 & 0.9699 & 0.9970 \end{bmatrix}$$

Then, Input to the Output Layer is IO and is calculated as shown below,

$IO = WO * OH$

$$IO = \begin{bmatrix} 3.4788 & 3.4142 & 3.4256 & 3.5636 & 3.4797 & 3.5970 \\ 2.4565 & 2.4138 & 2.4253 & 2.5042 & 2.4581 & 2.5207 \\ 4.1730 & 4.1173 & 4.1144 & 4.2550 & 4.1661 & 4.2920 \end{bmatrix}$$

Output of the Output Layer is denoted as OP and is obtained by applying Sigmoid activation function on the Input IO

$$OP = \begin{bmatrix} 0.9700 & 0.9681 & 0.9684 & 0.9724 & 0.9701 & 0.9733 \\ 0.9210 & 0.9178 & 0.9187 & 0.9244 & 0.9211 & 0.9255 \\ 0.9848 & 0.9839 & 0.9839 & 0.9860 & 0.9847 & 0.9865 \end{bmatrix}$$

The corresponding actual Output vector is initialised with zeros except the pattern class values column as '1' as given below,

$$AO = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

By substituting the above values in the Regularized cost function given in Eq. (4), the cost is obtained as -1.956291.

Now, to minimize the cost function, compute the error terms by comparing the actual and obtained outputs. Then apply the Back Propagation algorithm to update the weights connecting the Input to Hidden layer and Hidden to Output layers.

The following steps illustrate the calculation of the gradient values to update the respective weights.Initialize the weight gradients namely WH^G, WO^G to 0 and then calculate WH^G, WO^G using delta rule as shown below.

The first pattern of the training data TD i.e., [1 1 1 3] add bias node weight '+1' to get the complete features as,

$$IP = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 3 \end{bmatrix}$$

Now, calculating the IH and OH are obtained as,

$$IH = \begin{bmatrix} 4.3836 \\ 2.9400 \\ 2.6022 \\ 2.7478 \\ 3.6151 \end{bmatrix} \quad \text{And} \quad OH = \begin{bmatrix} 1.0000 \\ 0.9877 \\ 0.9498 \\ 0.9310 \\ 0.9398 \\ 0.9738 \end{bmatrix}$$

Then, computing IO and OP,

$$IO = \begin{bmatrix} 3.4789 \\ 2.4565 \\ 4.1730 \end{bmatrix} \quad \text{And } OP = \begin{bmatrix} 0.9701 \\ 0.9210 \\ 0.9848 \end{bmatrix}$$

The first patterns belongs to class '1' and hence the AO for the first pattern is,

$$AO = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Calculate the difference between the Actual output and the Obtained output is denoted as $Delta_3$

$$Delta_3 = \begin{bmatrix} -0.0299 \\ +0.9210 \\ +0.9848 \end{bmatrix}$$

Using $Delta_3$, calculate the difference at Hidden layer denoted as $Delta_2$,

$$Delta_2 = ((WO)^T * Delta_3) * \sigma^{'}(OH)$$

$$Delta_2 = \begin{bmatrix} 0.0137 \\ 0.0587 \\ 0.0702 \\ 0.0266 \\ 0.0263 \end{bmatrix}$$

After obtaining $Delta_2$ and $Delta_3$ the gradients can be calculated as follows,

$$WH^G = WH^G + Delta_2 * IP^T$$
$$WO^G = WO^G + Delta_3 * OH^T$$

$$WH^G = \begin{bmatrix} 0.0137 & 0.0137 & 0.0137 & 0.0137 & 0.0412 \\ 0.0587 & 0.0587 & 0.0587 & 0.0587 & 0.1762 \\ 0.0702 & 0.0702 & 0.0702 & 0.0702 & 0.2107 \\ 0.0266 & 0.0266 & 0.0266 & 0.0266 & 0.0798 \\ 0.0263 & 0.0263 & 0.0263 & 0.0263 & 0.0788 \end{bmatrix}$$

$$WO^G = \begin{bmatrix} -0.0299 & -0.0296 & -0.0284 & -0.0279 & -0.0281 & -0.0291 \\ 0.9210 & 0.9097 & 0.8748 & 0.8575 & 0.8656 & 0.8969 \\ 0.9848 & 0.9727 & 0.9354 & 0.9169 & 0.9255 & 0.9590 \end{bmatrix}$$

Similarly, calculate for every pattern and follow the same process until all the instances in training data are trained.

The gradient for the neural network cost function is obtained by dividing the accumulated gradients by the number of patterns trained.The final gradients are obtained as,

$$WH^G = \begin{bmatrix} 0.0114 & 0.0151 & 0.0144 & 0.0148 & 0.0197 \\ 0.0521 & 0.0699 & 0.0720 & 0.0641 & 0.0974 \\ 0.0729 & 0.0995 & 0.0950 & 0.1039 & 0.1281 \\ 0.0261 & 0.0355 & 0.0397 & 0.0306 & 0.0533 \\ 0.0277 & 0.0394 & 0.0371 & 0.0338 & 0.0490 \end{bmatrix}$$

$$WO^G = \begin{bmatrix} 0.4704 & 0.4685 & 0.4566 & 0.4562 & 0.4548 & 0.4635 \\ 0.5881 & 0.5812 & 0.5629 & 0.5378 & 0.5628 & 0.5711 \\ 0.8183 & 0.8089 & 0.7769 & 0.7538 & 0.7746 & 0.7928 \end{bmatrix}$$

Now these are the updated weights and training of ANN is to be done for more number of iterations so that the weights are to be stabilized. The modified WH and WO values are

$$WH = \begin{bmatrix} 0.0114 & 0.1370 & 0.1385 & 0.1191 & 0.1009 \\ 0.0521 & 0.1778 & 0.1035 & 0.1941 & 0.1700 \\ 0.0729 & 0.1747 & 0.2094 & 0.1174 & 0.2026 \\ 0.0261 & 0.1267 & 0.0703 & 0.1855 & 0.1044 \\ 0.0277 & 0.0888 & 0.0985 & 0.1631 & 0.1338 \end{bmatrix}$$

$$WO = \begin{bmatrix} 0.4704 & 0.5759 & 0.5454 & 0.6022 & 0.5526 & 0.5420 \\ 0.5881 & 0.6443 & 0.6214 & 0.6295 & 0.5974 & 0.6096 \\ 0.8183 & 0.9441 & 0.9334 & 0.8575 & 0.8248 & 0.9335 \end{bmatrix}$$

Error for 10 training iterations is shown in Figure 2.

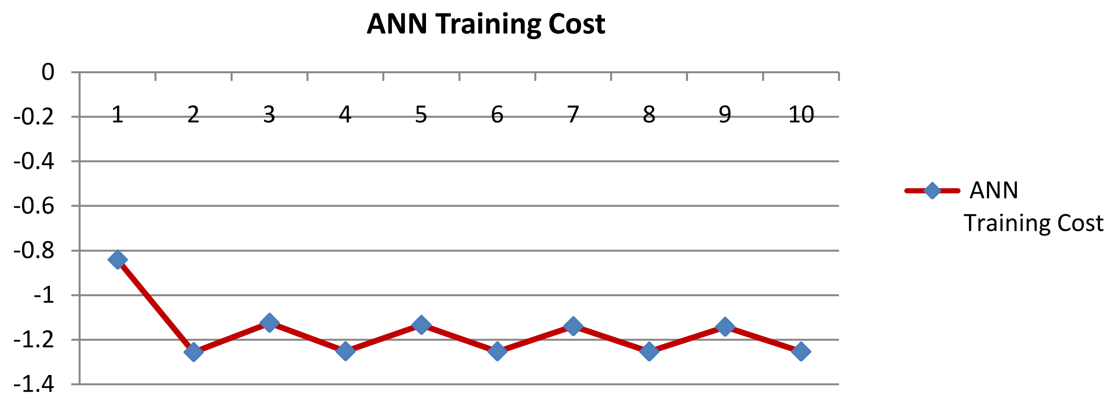


Fig. 2.ANN ClassificationError for 10 training iterations

4. Experimental Analysis

Experimental analysis has been carried out on the bench mark data sets taken from UCI repository [17] and the details are listed below in the Table 2.

Table 2. Data set description

Data Set	Instances	Total features	Features	Continuous Features	No. of Classes
Liver Disorder	337	7	Mean corpuscular volume, Alkaline phosphotase,Alamine aminotransferase, Aspartate aminotransferase, gamma - glutamyl transpetidase, drinks	1	Liver Diseas Not a liver disease
Pima Indian Diabetes	768	9	Pregnant,Plasma,Pressure,Skin, Insulin,Mass, Pedi, Age	2	Tested Positive Tested Negative
Heart Stat log	270	13	Age, sex, pain type, blood pressure, serum cholestrol, resting blood pressure, serum cholestrol, fasting blood pressure, resting electrocardiographic results, maximum heart rate, exercise induced angina, old peak, slope of the peak exercise, no. of major vessels, thal	1	Tested positive Tested negative

All the continuous valued attributes are discretized using Minimum Description Length Principle (MDLP)supervised discretization technique[18,19]. Table 3 gives the analysis of the MDLP discretization techniques on the above mentioned datasets.

Table 3. Number of possible values of Continuous valued attributes after applying MDLP

Data Set	Features	Before discretization	After discretization
Liver disorder	Drinks	15	5
PimaIndian Diabetes	Mass	250	2
	Pedi	517	2
Heart Statlog	Old peak	41	2

The classification accuracy of the trained neural network on continuous values and discrete values is given in Table 4.

Table 4. ANN Classification accuracy(%) on Undiscretized and Discretized data

Data Sets	Undiscretized Data	Discretized Data
Liver Disorder	81.91	82.13
Pima Indian Diabetes	88.66	93.5
Heart Stat Log	83.11	82.3

By submitting the datasets to Improved Quick Reduct generation algorithm, the reduced feature subset is given in Table 5.

Table 5. Feature subset generated by Improved QRG algorithm

Data Set	No.of Features	No.of Reduced Features	Reduced Features
Liver Disorder	6	3	gamma - glutamyl transpetidase, Alamine aminotransferase,Aspartate aminotransferase
PimaIndian Diabetes	8	3	Insulin, Plasma, Age
Heart Statlog	13	3	Serum cholesterol,Age, resting electrocardiographic results

When considering only relevant attributes for training the neural network, the training time is reduced since the network is a bit more simplified. The training time (in Seconds) of the ANN on full features and relevant features is shown in figure 3.

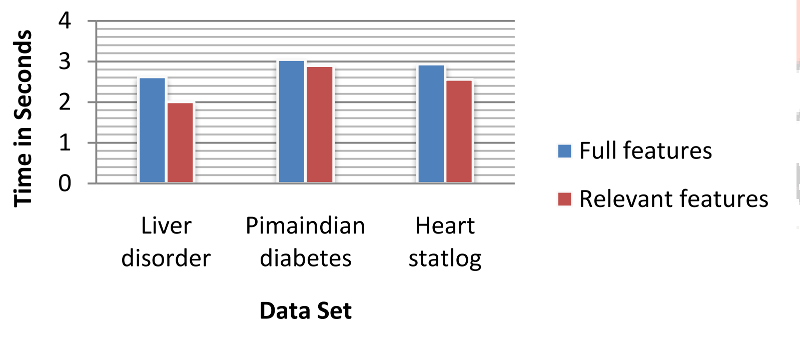


Fig. 3. Training time (in seconds) of ANN on Relevant and Full Features

The average 10-fold classification accuracy[20] of training ANN for 500 iterations each, on the above mentioned three datasets is shown in figure 4. It is observed that the accuracy of ANN has been increased when the attribute values are discretized and also the accuracy for relevant features is more when compared to full set of features.

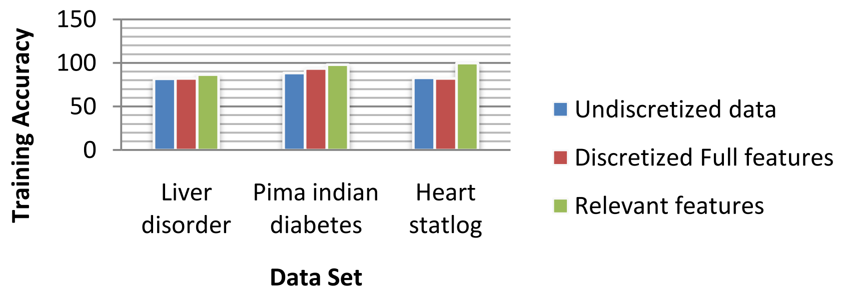


Fig. 4. Complete Analysis of ANN Classification Accuracy.

The complete analysis of the training time(in seconds) of the neural network on undiscretized, discretized features and relevant features is shown in figure 5.

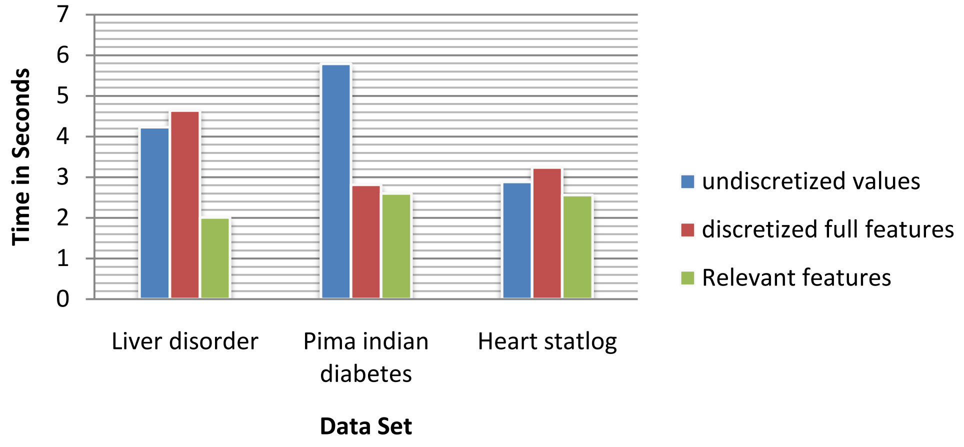
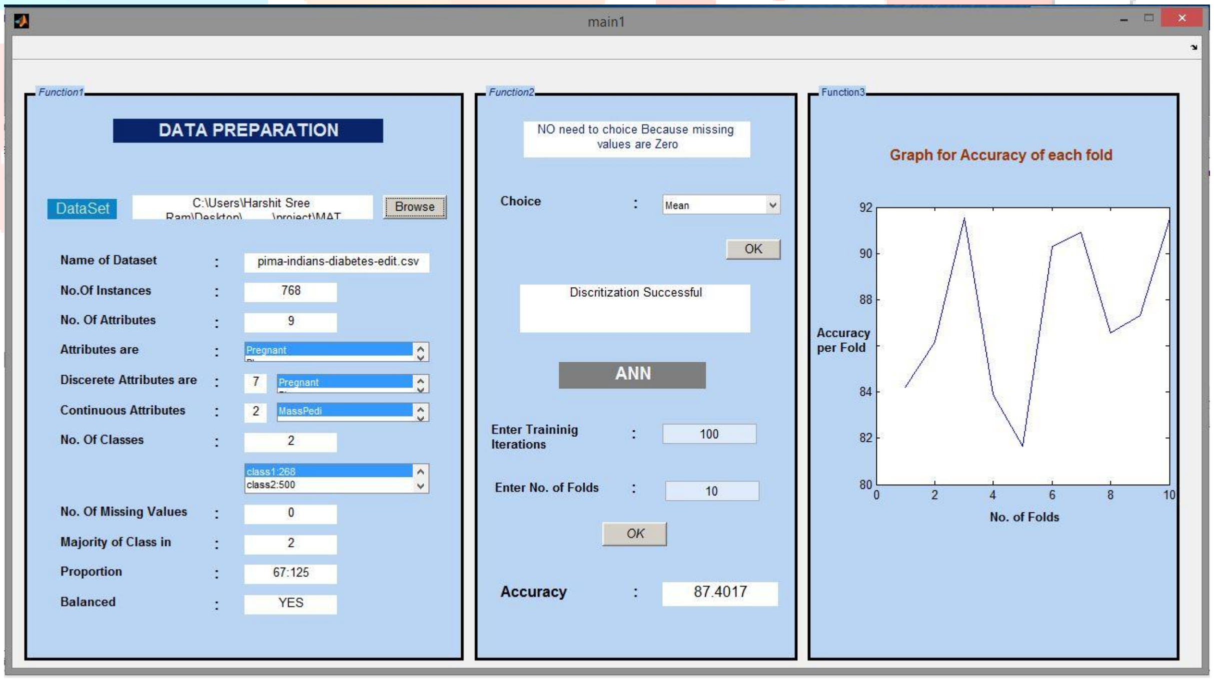


Fig. 5. Complete Analysis of ANN Training Time(in Seconds)

From the figure 5, it is clearly observed that the training time does not vary constantly for undiscretized and discretized data. But, observed a difference when training neural network on relevant and irrelevant features. The time taken to train a neural network on relevant features is less when compared to train neural network on full features.

The Rough Set Theoretic Optimized Single Hidden Layer Artificial Neural Networks based Classification system is implemented in MATLAB with 8GB RAM is shown in figure 6.



main1

Function1

DATA PREPARATION

DataSet C:\Users\Harshit Sree Ram\Desktop\ \project\MAT Browse

Name of Dataset	:	pima-indians-diabetes-edit.csv
No.Of Instances	:	768
No. Of Attributes	:	9
Attributes are	:	Pregnant
Discerete Attributes are	:	7 Pregnant
Continuous Attributes	:	2 MassPedi
No. Of Classes	:	2
		class1:268 class2:500
No. Of Missing Values	:	0
Majority of Class in	:	2
Proportion	:	67:125
Balanced	:	YES

Function2

NO need to choice Because missing values are Zero

Choice : Mean

OK

Discritization Successful

ANN

Enter Traininig Iterations : 100

Enter No. of Folds : 10

OK

Accuracy : 87.4017

Function3

Graph for Accuracy of each fold

Fig. 6. Rough Set Theoretic Optimized Single Hidden Layer Artificial Neural Networks based Classification system (for Pima Indians Diabetes dataset)

5. Conclusion

This paper presents a single hidden layer feed forward neural network, trained on optimal features generated by the RST based Improved Quick Reduct generation algorithm. The performance of the ANN based classifier not only depends on training algorithm but also on data provided for training. So, any kind of incompleteness in the training data is handled by filling the missing values and also discretized all continuous valued attributes using MDLP supervised discretization technique. Applying Improved Quick Reduct generation algorithm on the discretized data generated optimal features for all three datasets used in experimentation. Training Neural Network only on relevant features resulted in reduced training time and also increases the prediction ability, when compared against the full set of features. The experimental results show that, applying two machine learning techniques for data analysis improves the generalization ability of the Artificial Neural Networks and is very effective for classification of abundant data.

REFERENCES

[1]Bose, N. K.; & Liang P. (1996). Neural network fundamentals with graphs, algorithms, and applications. McGraw-Hill.

[2]Karayiannis, N. B., and Venetsanopoulos, A. N. 1993. Artificial neural networks: Learning algorithms, performance evaluation, and applications. Boston: Kluwer Academic.

[3] Maruthaveni, R. and Renuka Devi, S.V. 2015. Efficient Data Mining for Mining Classification Using Neural Network, International Journal of Data Mining & Knowledge Management Process, 3(2): 3852-3863.

[4]Kamruzzaman, S.M. and Jehad Sarkar, A.M. 2011. A New Data Mining Scheme Using Artificial Neural Networks. Sensors, 11(5):4622-4647.

[5] Awodele,O. and Jegede, O. 2009. Neural Networks and Its Application in Engineering. Proceedings of Informing Science & IT Education Conference (InSITE) 2009, 83-95.

[6]Stergiou, C.and Siganos, D. (1996). Neural networks. Retrieved from [http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html].

[7]Md.Monirul Islam and K.Murase, 2001. A new algorithm to design compact two-hidden-layer artificial neural networks.*Neural Networks*, 14(9): 1265-1278.

[8]Durairaj, M. and Nandakumar, R. 2014. An Integrated Methodology of Artificial Neural Network and Rough Set Theory for Analyzing IVF Data. 2014 IEEE International Conference on Intelligent Computing Applications (ICICA), 2014 Mar 6 :126-129.

[9] Durairaj, M. and Meena, K. 2011. A Hybrid Prediction System Using Rough Setsand Artificial Neural Networks. International Journal Of Innovative Technology & Creative Engineering, 1(7):16-23.

[10] Amato, F. , Lopez., A., Pena-Mendez, E. M., Vanhara, P., Hampl, A., and Havel, J., 2013. Artificial neural networks in medical diagnosis. Journal of Applied Biomedicine, 11(2): 47-58.

[11] Bakpo, F. S. and Kabari, L. G. 2011. Diagnosing Skin Diseases Using an Artificial Neural Network, Artificial Neural Networks - Methodological Advances and Biomedical Applications, Prof. Kenji Suzuki (Ed.), InTech, DOI: 10.5772/16232.

[12] Pawlak, Z. 1982. Rough Sets, *International Journal of Computer and Information Science*, 11(5):341- 356.

[13]Pawlak, Z, 1997. Rough set approach to knowledge-based decision support, *European Journal of Operational Research*, 99(1):48-57.

[14]Surekha, S. 2017. A Comparative Study of Various Rough Set Theory Algorithms for Feature Selection, International Journal of Engineering Sciences & Research Technology, 6(4):21-30.

[15]Kwok, T.Y, and Yeung,D.Y. 1994. A theoretically sound learning algorithm for constructive neural networks.In Speech, Image Processing and Neural Networks, 1994. Proceedings, ISSIPNN'94., 1994: 389-392.

[16]Machine Learning Tutorial, Stanford University Retrieved from [https://github.com/rieder91/MachineLearning/blob/master/Exercise%204/ex4.pdf]

[17]Irvine UCI Machine Learning Repository. [http://archive.ics.uci.edu/ml/] A University of California, Center for Machine Learning and Intelligent Systems.

[18]Liu, H., Hussain, F., Tan, C.L., and Dash, M. 2002. Discretization: An Enabling Technique, In Data Mining and Knowledge Discovery, 6(4):393-423.

[19]Usama, M.F., and Irani, K.B. 1993. Multi-Interval Discretization Of Continuous-Valued Attributes For Classification Learning, Machne Learning, 1022-1027.

[20] Bengio, Y., and Grandvalet, Y. 2004. No Unbiased Estimator of the Variance of K-Fold Cross-Validation, Journal of Machine Learning Research, 5:1089–1105 .