Chapter 5

Performance analysis of Neural Network Classifier for the Different Number of Hidden Units
CHAPTER 5
PERFORMANCE ANALYSIS OF NEURAL NETWORK CLASSIFIER FOR THE DIFFERENT NUMBER OF HIDDEN UNITS

5.1 Introduction

When data mining algorithms are talked about, people usually talk about the decision trees or neural networks. Of the two, neural networks have probably been of greater interest through the formative stages of data mining technology. Neural network simulations appear to be useful for many real time applications in recent times. Even though, this field was established before the advent of computers, and has survived at least one major setback through several eras. The field of neural networks was originally kindled by psychologists and neurobiologists McCulloch and Pitts who sought to develop and test computational analogues of neurons [123].

They are generally recognized as being the designers of the first neural network. They recognized that combining many simple processing units together could lead to an overall increase in computational power. Their premise was that if two neurons were active at the same time then the strength between them should be increased. The paper first outlined the idea that simple processing units (such as the individual neurons in the human brain) could be connected together in large networks to create a system that could solve difficult problems and display behavior that was much more complex than the simple pieces that made it up. Since then, much progress has been made in finding ways to apply artificial neural networks to real world prediction problems and in improving the performance of the algorithm in general. In recent years, the greatest breakthroughs in neural networks are in their application to more mundane real world problems [124, 125].

Perceptron was invented in 1958 by psychologist Frank Rosenblatt and it was intended to model how the human brain processed visual data and learned to recognize
objects [126]. Rosenblatt’s model consisted of three layers, (1) a “retina” that
distributed inputs to the second layer, (2) “association units” that combine the inputs
with weights and trigger a threshold step function which feeds to the output layer, (3)
the output layer which combines the values. In the fifties and throughout the sixties
many researchers worked on the perceptron [127-129]. The name perceptron is now
used as a synonym for single-layer, feed-forward networks. They were first studied in
the 1950’s and although other network architectures were known about the perceptron
was the only network that was known to be capable of learning and thus most of the
research at that time concentrated on perceptrons. The neural network model can be
proved to converge to the correct weights, if there are weights that will solve the
problem.

Unfortunately, the use of a step function in the neurons made the perceptions
difficult or impossible to train. A critical analysis of perceptrons published in 1969 by
Marvin Minsky and Seymore Papert pointed out a number of critical weaknesses of
perceptrons, and, for a period of time, interest in perceptrons waned [130]. Interest in
neural networks was revived in 1986 when David Rumelhart, Geoffrey Hinton and
Ronald Williams published “Learning Internal Representations by Error Propagation”
[131]. They proposed a multilayer neural network with nonlinear but differentiable
transfer functions that avoided the pitfalls of the original perceptron’s step functions.
They also provided a reasonably effective training algorithm for neural networks.

Broadly speaking, there are two methods for training the neural network,
depending on the problem it must solve. A self-organizing neural network often called
a Kohonen is exposed to large amounts of data and tends to discover patterns and
relationships in that data [132]. Researchers often use this type to analyze
experimental data. A back-propagation neural network, conversely, is trained by
humans to perform specific tasks.

Neural networks are used in a wide variety of applications. Artificial neural
networks have proved useful in a variety of real-world applications that deal with
complex, often incomplete data, visual pattern recognition and speech recognition, text-to-speech, handwriting analysis programs, control machinery, adjust temperature settings and diagnose malfunctions. Large financial institutions have used Artificial Neural Network to improve performance in such areas as bond rating, credit scoring, target marketing and evaluating loan applications, analyze credit card transactions to detect likely instances of fraud and other kinds of crime, too [133-140 ].

Neural network produces different accuracies for different number of hidden layers. Network design is a trial-and-error process and may affect the accuracy of the resulting training network. Hence the experiments have been carried out for the hidden units varying from 1 to 10 for both the databases and the best performance Neural network classifier has been identified. The classifier which has been identified as best is allowed to participate in the proposed DCMCS.

5.2 Neural networks versus conventional computers

Neural networks take a different approach to problem solving than that of conventional computers [141, 142]. Conventional computers use an algorithmic approach i.e. the computer follows a set of instructions in order to solve a problem. Unless the specific steps that the computer needs to follow are known the computer cannot solve the problem. That restricts the problem solving capability of conventional computers to problems that we already understand and know how to solve. But computers would be so much more useful if they could do things that we don't exactly know how to do.

Neural networks process information in a similar way the human brain does. The network is composed of a large number of highly interconnected processing elements (neurones) working in parallel to solve a specific problem. Neural networks learn by example. They cannot be programmed to perform a specific task. The examples must be selected carefully otherwise useful time is wasted or even worse the network might be functioning incorrectly. The disadvantage is that because the network finds out how to solve the problem by itself, its operation can be
unpredictable. On the other hand, conventional computers use a cognitive approach to problem solving; the way the problem is solved must be known and stated in small unambiguous instructions. These instructions are then converted to a high level language program and then into machine code that the computer can understand. These machines are totally predictable; if anything goes wrong is due to a software or hardware fault.

Neural networks and conventional algorithmic computers are not in competition but complement each other. There are tasks that are more suited to an algorithmic approach like arithmetic operations and tasks that are more suited to neural networks. Even more, a large number of tasks, require systems that use a combination of the two approaches (normally a conventional computer is used to supervise the neural network) in order to perform at maximum efficiency.

5.3 Normalization in Neural network

The Neural Network is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. The Neural network is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of neural network as well [143, 144].

There are also many important steps required for preprocessing the data that goes into a neural network [15]. Most often, there is a requirement to normalized numeric data between 0.0 and 1.0 and categorical predictors may need to be broke up into virtual predictors that are 0 or 1 for each value of the original categorical predictor. And, as is always, understanding what the data in the database means and having a clear definition of the business problem to be solved are essential for
ensuring eventual success. The bottom line is that normal network provide no shortcuts. The weights in the network are initialized to small random number (e.g., ranging from −1.0 to 1.0, or −0.5 to 0.5). Each unit has a bias associated with it. The biases are similarly initialized to small random numbers.

5.4 Multilayer Feed Forward Neural Network

Neural network is a set of connected input/output units where each connection has a weight associated with it. During the learning phase, the network learns by adjusting the weights so as to be able to predict the correct class label of the input samples. Neural network learning is also referred to as connections learning due to the connections between units.

A neural network is loosely based on the way some people believe that the human brain is organized and how it learns. There are two main structures of consequence in the neural network [145].

- The node which loosely corresponding to the neuron in the human brain
- The link, which loosely corresponding to the connections between neurons (Axons. Dendrites, and synapses) in the human brain.

The most common neural network model is the Multilayer Perceptron (MLP) [146]. This type of neural network is known as a supervised network because it requires a desired output in order to learn. The goal of this type of network is to create a model that correctly maps the input to the output using historical data so that the model can then be used to produce the output when the desired output is unknown. The multi layer neural network is shown in Figure 5.1 has three layers of output units.

A network containing two hidden layers is called a three-layer neural network. The units in the hidden layers and output layer are sometimes referred to as neurodes, due to their symbolic biological basis, or as output units. The units are arranged in layers, with the first layer taking in inputs and the last layer producing outputs. The middle layers have no connection with the external world, and hence are called hidden layers. Each unit in one layer is connected to every unit on the next layer. Hence
information is constantly "fed forward" from one layer to the next, and this explains why these networks are called feed-forward networks. There is no connection among units in the same layer.

Figure 5.1 Multilayer Feed forward Neural Network

The inputs are fed into the input layer and get multiplied by interconnection weights as they are passed from the input layer to the first hidden layer. Within the first hidden layer, they get summed then processed by a nonlinear function (usually the hyperbolic tangent). As the processed data leaves the first hidden layer, again it gets multiplied by interconnection weights, then summed and processed by the second hidden layer. Finally the data is multiplied by interconnection weights then processed one last time within the output layer to produce the neural network output.

In order to make a prediction, the neural network accepts the values for the predictors on what are called the input nodes. These become the values for those nodes. Those values are then multiplied by values that are stored in the links. These values are then added together at the node at the far right (the output node), a special thresholding function is applied, and the resulting number is the prediction.
5.5 Defining a network Topology

Figure 5.2 illustrates a Multilayer feed forward neural network (MLFN) topology. It is a fully connected, three layer, feed forward neural network. The "Fully connected" means that the output from each input and hidden neuron is distributed to all of the neurons in the following layer. "Feed forward" means that the values only move from input to hidden to output layers; no values are fed back to earlier layers (a Recurrent Network allows values to be fed backward) [147].

\[
X = \begin{bmatrix}
    x_1 \\
    x_2 \\
    \vdots \\
    x_n
\end{bmatrix}
\]

\[
Y = \begin{bmatrix}
    y_1 \\
    y_2 \\
    \vdots \\
    y_m
\end{bmatrix}
\]

Figure 5.2 Network Topology of three layer Neural Network

All neural networks have an input layer and an output layer, but the number of hidden layers may vary. Figure 5.2 shows the Network Topology of three layers Neural Network. This network has an input layer (on the left) with three neurons, one hidden layer (in the middle) with three neurons and an output layer (on the right) with three neurons. There is one neuron in the input layer for each predictor variable. In the case of categorical variables, \(N-1\) neurons are used to represent the \(N\) categories of the variable.

5.5.1 Input Layer

A vector of predictor variable values \((x_1, \ldots, x_p)\) is presented to the input layer. The input layer (or processing before the input layer) standardizes these values by subtracting the median and dividing by the inter quartile range and distributes the values to each of the neurons in the hidden layer. In addition to the predictor variables,
there is a constant input of 1.0, called the bias that is fed to each of the hidden layers; the bias is multiplied by a weight and added to the sum going into the neuron.

5.5.2 Hidden Layer

Arriving at a neuron in the hidden layer, the value from each input neuron is multiplied by a weight \( w_{ji} \), and the resulting weighted values are added together producing a combined value \( u_j \). The weighted sum \( (u_j) \) is fed into a transfer function, \( s \), which outputs a value \( h_j \). The outputs from the hidden layer are distributed to the output layer.

5.5.3 Output Layer

Arriving at a neuron in the output layer, the value from each hidden layer neuron is multiplied by a weight \( w_{kj} \), and the resulting weighted values are added together producing a combined value \( v_j \). The weighted sum \( (v_j) \) is fed into a transfer function, \( s \), which outputs a value \( y_k \). The \( y \) values are the outputs of the network.

If a regression analysis is being performed with a continuous target variable, then there is a single neuron in the output layer, and it generates a single \( y \) value. For classification problems with categorical target variables, there are \( N \) neurons in the output layer producing \( N \) values, one for each of the \( N \) categories of the target variable.

Before training begins, the user must decide on the network topology by specifying the number of units in the input layer, the number of hidden layers, the number of units in each hidden layer, and the number of units in the output layer.

Normalizing the input value for each attribute measured in the training samples will help speed up the learning phase. Typically, input values are normalized so as to fall between 0.0 and 1.0. discrete-valued attributes may be encoded such that there is one input unit per domain value. For example, if the domain of an attribute \( A \) is \( \{a_0, a_1, a_2\} \), then we may assign three input units to represent \( A \), that is, we may have, say,
I₀, I₁, I₂ as input units. Each unit is initialized to 0 if A=a₀, then I₀ is set to 1. If A=a₁, I₁ is set to 1, and so on. One output unit may be used to represent two classes (where the value 1 represents one class, and the value 0 represents the other). If there are more than two classes, then one output unit per class is used.

There are no clear rules as to the "best" number of hidden layer units. Network design is a trial-and-error process and may affect the accuracy of the resulting training network. The initial values of the weights may also affect the resulting accuracy. Once a network has been trained and its accuracy is not considered acceptable, it is common to repeat the training process with a different network topology or a different set of initial weight [148].

One of the most important characteristics of a MLFN network is the number of neurons in the hidden layer(s). If an inadequate number of neurons are used, the network will be unable to model complex data, and the resulting fit will be poor. If too many neurons are used, the training time may become excessively long, and, worse, the network may overfit the data. When overfitting occurs, the network will begin to model random noise in the data. The result is that the model fits the training data extremely well, but it generalizes poorly to new, unseen data. Validation must be used to test for this.

5.6 Back Propagation

Backpropagation is a supervised learning technique used for training artificial neural networks. It was first described by Paul Werbos in 1974, and further developed by David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams in 1986 [131].

The MLP and many other neural networks learn using the backpropagation algorithm. With backpropagation, the input data is repeatedly presented to the neural network. The inputs correspond to the attributes measured for each training sample. The inputs are fed simultaneously into a layer of units making up the input layer. The weighted outputs of these units are, in turn, fed simultaneously to a second
layer of "neuronlike" units, known as a hidden layer. The hidden layer's weighted
outputs can be input to another hidden layer, and so on. The number of hidden layer is
arbitrary. With each presentation the output of the neural network is compared to the
desired output and an error is computed. This error is then fed back (backpropagated)
to the neural network and used to adjust the weights such that the error decreases with
each iteration and the neural model gets closer and closer to producing the desired
output. This process is known as "training". The weight adjustment are made in the
"backwards" direction, that is, from the output layer, through each hidden layer down
to the first hidden layer and hence it is called backpropagation. The weighted output
of the last hidden layer are input to units making up the output layer, which emits the
network's prediction for given samples. [149-151]

5.6.1 Algorithm
The Algorithm has been implemented using EDM Data mining tool.

Input
The training samples, samples; the learning rate, \( \eta \); a multiplayer feed-forward
network, network.

Output
A neural network trained to classify the samples.

Method
1. Initialize all weights and biases in network;
2. While terminating condition is not satisfied
3. {
4. for each training sample \( X \) in samples
5. {
6. for each hidden or output layer unit \( j \)
7. {
8. \[ I_j = \sum_i w_{ij} O_i + \theta_j; \]
9. \[ O_j = \frac{1}{1+e^{-I_j}} \]
10. }
11. for each unit \( j \) in the output layer

90
The net input and output of each unit in the hidden and output layers are computed. First, the training sample is fed to the input layer of the network. Note that for unit $j$ in the input layer, its output is equal to its input, that is, $O_j = I_j$ for input unit $j$. The net input to each unit in the hidden and output layers is computed as a linear combination of its inputs. To help illustrate this, a hidden layer or output layer unit is shown in figure 5.3.

The inputs to the unit are, in fact, the outputs of the units connected to it in the previous layer. To compute the net input to the units, each input connected to the unit is multiplied by its corresponding weight, and this is summed. Given a unit $j$ in a hidden or output layer, the net input, $I_j$, to unit $j$ is

$$ I_j = \sum w_{ij} O_i + \theta_j $$  \hspace{1cm} (5.1)
Where $w_{ij}$ is the weight of the connection from unit $i$ in the previous layer to unit $j$; $O_i$ is the output of unit $I$ from the previous layer; and $\theta_j$ is the bias of the unit. The bias acts as a threshold in that it serves to vary the activity of the unit.

Each unit in the hidden and output layers takes its net input and then applies an activation function to it, as illustrated in Figure 5.2. The function symbolizes the activation of the neuron represented by the unit. The logistic, or sigmoid, function is used. Given the net input $I_j$ to unit $j$, then $O_j$, the output of unit $j$, is computed as

$$O_j = \frac{1}{1 + e^{-\theta_j}}$$

(5.2)

This function is also referred to as a squashing function, since it maps a large input domain onto the smaller range of 0 to 1. The logistic function is nonlinear and differentiable, allowing the backpropagation algorithm to model classification problems that are linearly inseparable.
The error is propagated backwards by updating the weight and biases to reflect the error of the network’s prediction. For a unit \( j \) in the output layer the error \( E_{rj} \) is computed by

\[
E_{rj} = O_j (1-O_j) (T_j-O_j)
\]  \hspace{1cm} (5.3)

Where \( O_j \) is the actual output of unit \( j \), and \( T_j \) is the true output, based on the known class label of the given training sample note that \( O_j (1-O_j) \) is the derivative of the logistic function. To compute the error of a hidden layer unit \( j \), the weighted sum of the errors of the units connected to unit \( j \) in the next layer are considered. The error of a hidden layer unit is

\[
E_{rj} = O_j (1-O_j) \sum_k Err_k w_{jk}
\]  \hspace{1cm} (5.4)

Where \( w_{jk} \) is the weight of the connection from unit \( j \) to a unit \( k \) in the next higher layer, and \( Err_k \) is the error of unit \( k \).

The weights and biases are updated to reflect the propagated errors. Weight are updated by the following equations, where \( \Delta w_{ij} \) is the change in weight \( w_{ij} \):

\[
\Delta w_{ij} = (l) Err_j O_i
\]  \hspace{1cm} (5.5)

\[
w_{ji} = w_{ij} + \Delta w_{ij}
\]  \hspace{1cm} (5.6)

The variable \( l \) is the learning rate, a constant typically having a value between 0.0 and 1.0. Backpropagation learns using a method of gradient descent to search for a set of weights that can model the given classification problem so as to minimize the mean squared distance between the network’s class prediction and the actual class label of the samples. The learning rate helps to avoid getting stuck at a local minimum in decision space and encourages finding the global minimum. If the learning rate is too small, then learning will occur at a very slow pace. If the learning rate is thumb is to set the learning rate to \( 1/t \), where \( t \) is the number of iterations through the training set so far. Biases are updated by the following equation below, where \( \Delta \theta_j \) is the change in bias \( \theta_j \):

93
\[ \Delta \theta_j = (1) \text{Err}_j \]  
\[ \theta_j = \theta_j + \Delta \theta_j \]  

Updating the weights and biases after the presentation of each sample is referred to as case updating. Alternatively, the weight and bias increments could be accumulated in variables, so that the weights and biases are updated after all of the samples in the training set have been presented. This latter strategy is called epoch updating, where one iterations through the training set is an epoch. In theory, the mathematical derivation of back propagation employs epoch updating, yet in practice, case updating is more common since it tends to yield more accurate results.

5.6.2 Terminating condition

Training stops when

- All \( \Delta w_{ij} \) in the previous epoch were so small as to be below some specified threshold, or
- The percentage of samples misclassified in the previous epoch is below some threshold, or
- A prespecified number of epochs has expired.

In practice, several hundreds of thousands of epochs may be required before the weights will converge.

5.7 Experimental Results

5.7.1 Adult database

The adult database consists of two class label such as class 1 and class 2. Class 1 represents those cases whose annual income is greater than 50K. Class 2 represents those cases whose annual income less than or equal to 50K. The Prediction task of adult database is to determine whether a person makes over 50K a year. The threshold have been computed and fixed for the training data set as shown in chapter 2. In every iteration, a 25% of random samples have been selected from the training database and
the statistical information about the random samples has been also reported in chapter 2. Table 5.1 shows the Accuracy of Neural Network classifier for the individual classes and also the overall accuracy. The experiments have been carried out for the 10 different random samples in the adult database and the accuracies have been estimated.

Figure 5.4 compares the accuracies of different classes in different iterations using the Adult database. It also shows that the accuracy of class 2 always outperforms the accuracy of class 1. The accuracy of class 1 is in the range 66% to 71% and the accuracy of class 2 is in the range 81% to 86%. The accuracy of class 1 is less than class 2 in all 10 iterations. The reason is that the number of samples in class 1 is less by 2000 to 5000 samples than in class 2. So the percentage of accuracy is also less for class 1 compared to class 2 and it is consistent with other classifiers discussed in chapter 3 and 4.

Table 5.1 Individual Class performance of Neural Network Classifier for the Adult database

<table>
<thead>
<tr>
<th>Iterations</th>
<th>Probability</th>
<th>Unsatisfied Class</th>
<th>Accuracy</th>
<th>Overall Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Class 1</td>
<td>Class 2</td>
<td>Class 1</td>
<td>Class 2</td>
</tr>
<tr>
<td>1</td>
<td>0.213</td>
<td>0.787</td>
<td>1</td>
<td>67.42</td>
</tr>
<tr>
<td>2</td>
<td>0.219</td>
<td>0.781</td>
<td>1</td>
<td>67.53</td>
</tr>
<tr>
<td>3</td>
<td>0.331</td>
<td>0.669</td>
<td>2</td>
<td>70.93</td>
</tr>
<tr>
<td>4</td>
<td>0.217</td>
<td>0.783</td>
<td>1</td>
<td>67.62</td>
</tr>
<tr>
<td>5</td>
<td>0.206</td>
<td>0.794</td>
<td>1</td>
<td>67.04</td>
</tr>
<tr>
<td>6</td>
<td>0.211</td>
<td>0.789</td>
<td>1</td>
<td>67.83</td>
</tr>
<tr>
<td>7</td>
<td>0.207</td>
<td>0.793</td>
<td>1</td>
<td>67.57</td>
</tr>
<tr>
<td>8</td>
<td>0.214</td>
<td>0.786</td>
<td>1</td>
<td>66.97</td>
</tr>
<tr>
<td>9</td>
<td>0.215</td>
<td>0.785</td>
<td>1</td>
<td>67.03</td>
</tr>
<tr>
<td>10</td>
<td>0.231</td>
<td>0.769</td>
<td>Nil</td>
<td>68.98</td>
</tr>
</tbody>
</table>

95
Figure 5.4 also shows that a class which doesn’t satisfy the threshold in an iteration have produced less accuracy than the same class which satisfy the threshold in another iteration. From the 10 iterations, the 10\textsuperscript{th} iteration only satisfies the threshold for all the classes. The individual class accuracies of all the classes have been improved in the 10\textsuperscript{th} iteration only.

![Figure 5.4 Comparing the Class performance of Neural Network Classifier for the Adult database](image)

There are no clear rules as to the "best" number of hidden layers. Network design is a trial-and-error process and may affect the accuracy of the resulting training network. So, the network has been trained for different number of hidden units varying from 1 to 10 and performance has been analyzed. Different accuracies have been produced by the Neural Network using different number of hidden units for the same set of data samples. For each data set, the results relating to the number of hidden units which produced the highest accuracy is shown in Figure 5.5.

Figure 5.6 shows the ROC Curve for the Neural Network classifier. It shows the relationship between false positives and true positives. The curve was originally used to examine the percentage of retrieved that are not relevant versus the percentage of retrieved that are relevant. The ROC graph is a plot with the false positive rate on the X axis and the true positive rate on the Y axis.
The Choice
Hidden Units
Accuracy (%)
79 80 81 82 83 84 85
1 2 3 4 5 6 7 8 9 10
Iteration Number

Figure 5.5 Highest accuracy of Neural Network classifier for the different number of Hidden units in different iterations for the Adult database

Figure 5.6 ROC Curve of Neural Network Classifier for the Adult database

The point (0,1) is the perfect classifier: it classifies all positive cases and negative cases correctly. The point (1,1) corresponds to a classifier that predicts every case to be positive. The point (0,0) represents a classifier that predicts all cases to be negative, while Point (1,0) is the classifier that is incorrect for all classifications. The False positive range obtained for the Neural Network classifier for the adult database is 0.167 has been varying from 0.1 to 0.3 in different iterations. The true positive range has been obtained in the range from 0.7 to 0.9 for different random samples. The purple curve is obtained by the classifier which is constructed from the dataset those
satisfies all threshold. The performance of this classifier outperforms other classifiers obtained in different iterations those does not satisfy the threshold.

5.7.2 Earthquake Database

The earthquake database consists of 2,51,843 number of cases and 11 features. It consists of three class label attributes class 1, class 2 and class 3. The magnitude from 4.0 to less than 5.5 is in class 1, the magnitude from 5.5 to less than 7 is in class 2 and the magnitude from 7 and above is in class 3. The statistical details of Earthquake database is discussed in chapter 2. Threshold have been computed and fixed for the individual classes as shown in chapter 2. The random sampling has been performed on the earthquake database until the random satisfies the threshold. Table 5.2 shows the Accuracy of Neural Network classifier for the individual classes.

Table 5.2 Individual Class performance of Neural Network Classifier for the Earth Quake Database

<table>
<thead>
<tr>
<th>Iterations</th>
<th>Probability Unsatisfied Class</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Class 1</td>
<td>Class 2</td>
</tr>
<tr>
<td>1</td>
<td>0.942</td>
<td>0.0562</td>
</tr>
<tr>
<td>2</td>
<td>0.921</td>
<td>0.0776</td>
</tr>
<tr>
<td>3</td>
<td>0.901</td>
<td>0.0977</td>
</tr>
<tr>
<td>4</td>
<td>0.923</td>
<td>0.049</td>
</tr>
<tr>
<td>5</td>
<td>0.946</td>
<td>0.0529</td>
</tr>
<tr>
<td>6</td>
<td>0.945</td>
<td>0.0540</td>
</tr>
<tr>
<td>7</td>
<td>0.951</td>
<td>0.0473</td>
</tr>
<tr>
<td>8</td>
<td>0.933</td>
<td>0.0654</td>
</tr>
<tr>
<td>9</td>
<td>0.937</td>
<td>0.0619</td>
</tr>
<tr>
<td>10</td>
<td>0.947</td>
<td>0.0513</td>
</tr>
<tr>
<td>11</td>
<td>0.93</td>
<td>0.0681</td>
</tr>
<tr>
<td>12</td>
<td>0.912</td>
<td>0.0792</td>
</tr>
</tbody>
</table>
The experiments have been carried out for the 12 different random samples using the adult database. Figure 5.7 compares the accuracies of different classes in different iterations using the Earthquake database. It also shows that the accuracy of class 1 is always outperforms the accuracy of class 2 & 3 and the accuracy of class 2 outperforms class 3 in most of the iterations. The accuracy of class 1 is in the range 94% to 96%, the accuracy of class 2 is in the range 50% to 58% and the accuracy of class 3 is in the range 39% to 53%. The reason is that the number of samples in class 1 is more than class 2 & 3 and also the number of samples in class 2 is more than class 3. So the percentage of accuracy is less for class 3 and more for class 1.

Figure 5.7 also shows that classes that didn’t satisfy the threshold in an iteration have produced less accuracy than the one which satisfies the threshold in another iteration. From the 12 iterations, the 12th iteration only satisfies the threshold for all the classes. The individual class accuracies of all the classes have been improved in the 12th iteration only.

Figure 5.8 shows the ROC Curve for the Neural Network classifier using the Earthquake database. It shows the relationship between false positives and true positives. The False positive range obtained for the Neural Network classifier for the earthquake database has been varying from 0.01 to 0.2 in different iterations. The true positive range has been obtained in the range from 0.815 to 0.95 for the different random samples. The purple curve is obtained by the classifier which is constructed from the dataset those satisfies all threshold. The performance of this classifier outperforms other classifiers obtained in different iterations those does not satisfy the threshold. The performance the classifier obtained in the last iteration is near the perfect classifier in the ROC graph.

The network has been trained for different number of hidden units varying from 1 to 10 and performance has been analyzed. Different accuracies have been produced by the different number of hidden units for the same set of data samples. Figure 5.9 shows the accuracy of BPN for different number of hidden units for the
same set of samples. For each data set, the results related to the number of hidden units, which produced the highest accuracy is identified.

Figure 5.7 Individual Class performance of Neural Network Classifier for the Earth Quake database

Figure 5.8 ROC Curve of Neural Network Classifier for the Earth Quake database
5.8 Conclusion

The computing world has a lot to gain from neural networks. Their ability to learn by example, makes them very flexible and powerful. Furthermore there is no need to devise an algorithm in order to perform a specific task; i.e. there is no need to understand the internal mechanisms of that task. They are also very well suited for real time systems because of their fast response and computational times which are due to their parallel architecture.

Figure 5.10 compares the performance of Adult database and the earthquake database for the Back propagation neural network. The figure shows that the Accuracy obtained by Earthquake database is higher than the accuracy of adult database in different iterations. In the adult database, a 33% of the attributes are continuous. But in the earthquake database, 88% of the attributes are continuous. Hence the Neural Network classifier have also produced more accuracy to the database containing numerical attributes than the database containing categorical attributes and it is consistent with KNN classifier which is discussed in chapter 4.
Neural networks have high tolerance to noisy data as well as their ability to classify patterns on which they have not been trained. Neural networks could outperform other techniques because they “learn” and improve over time whereas the other techniques are static. These are the highly accurate predictive models that can be applied to a variety of different types of problems. These factors contribute towards the usefulness of neural networks for classification in data mining. It can handle large number of features.

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyse. This expert can then be used to provide projections given new situations of interest and answer "what if" questions. It has the ability to learn how to do tasks based on the data given for training or initial experience.
Neural network can learn more complicated tree boundaries. But its training time is very large. A major disadvantage of neural networks lies in their knowledge representation. Neural network are hard to interpret and implement. It has been criticized for their poor interpretability, since it is difficult for humans to interpret the symbolic meaning behind the learned weights. Acquired knowledge in the form of a network of units connected by weighted links is difficult to interpret. Because of the complexity of these techniques, much effort has been expensed in trying to increase the clarity with which the model can be understood by the end user. There is a trial and error strategy for choosing the number of nodes.

Based on the experimental results, the best performance classifier is identified for both the databases. In the Adult database, the classifier obtained in the 10\textsuperscript{th} iteration produces highest accuracy for the number of hidden layers 7. Likewise in the earthquake database, the classifier obtained in the 12\textsuperscript{th} iteration produces highest accuracy for the number hidden layers 5. The selected classifiers are allowed to participate in the proposed DCMCS.

The experiments have been carried out in chapter 3, 4 and 5 using various random samples obtained in chapter 2 for both the databases. In each chapter, the best performance classifier was identified. In the next chapter, the newly proposed DCMCS is analysed using the best performance classifiers identified in chapter 3, 4 & 5.