A PROJECT REPORT

ON

"MINING CLASSIFICATION RULES FROM DATABASE USING ARTIFICIAL NEURAL NETWORK"

Under the guidance of

HOD Prof. Mrs. S. Shinde

Submitted by

B8338503	Prashant P. Avaghade
B8338513	Datta G. Diware
B8338528	Mahesh S. kshatriya
B8338559	Nikhilesh S. Shinde
B8338570	Sumit S.Wankhede

Towards the Partial fulfillment of Final year degree course in

B.E. Information Technology

Of

University of Pune In the academic year 2011-12

DEPARTMENT OF INFORMATION TECHNOLOGY,

PIMPRI CHINCHWAD COLLEGE OF ENGINEERING,

PUNE-46

(2011-2012)

PIMPRI CHINCHWAD EDUCATION TRUSTS PIMPRI CHINCHWADCOLLEGE OF ENGINEERING AKURDI, PUNE-411044

CERTIFICATE

This is to certify that

Project work

ON

"MINING CLASSIFICATION RULES FROM DATABASE USING ARTIFICIAL NEURAL NETWORK"

Submitted by

B8338503	Prashant P. Avaghade
B8338513	Datta G. Diware
B8338528	Mahesh S. kshatriya
B8338559	Nikhilesh S. Shinde
B8338570	Sumit S.Wankhede

Under the guidance of

HOD Prof. Mrs. S. Shinde

Towards the Partial fulfillment of Final year degree course in

B.E. Information Technology Of

University of Pune In the academic year 2011-12

Prof. S. V. Shinde	**Prof. R. G. Pise**	**Prof. S. V.Shinde**	**Prof.A.M.Fulambarkar**
Project Guide	**Project Coordinator**	**HoD**	**Principal**

ACKNOWLEDGEMENT

Any accomplishment requires converted efforts of many people and this project is no different. We are highly obliged and grateful to all those who have helped us in the development of this Project and provided all encouragement that has helped us immensely in our project work.

It is our pleasure to present the project entitled **"MINING CLASSIFICATION RULES FROM DATABASE USING ARTIFICIAL NEURAL NETWORK"** which has been a throughout joint effort. Since the time we have started working on this project, we have thoroughly been engrossed with the technological exploration of the field. We acquired knowledge by reading different books and searching on the Internet.

We take this opportunity to thank our respected **Project Guide Prof.Swati V. Shinde** without whose constant support, help and encouragement, this endeavor of ours wouldnt have materialized.

We thank our Principal **Dr.A.M.Fulambarkar** Project coordinator **Prof.R.G.Pise** and HOD **Prof.S.V.Shinde** for creating a wonderful learning environment. Finally, we thank all teaching and non-teaching staff who has associated directly/indirectly in the execution of this project work.

Last but not least, we are also grateful to our friends for their valuable suggestions.

B8338503 Prashant P. Avaghade

B8338513 Datta G. Diware

B8338528 Mahesh S. kshatriya

B8338559 Nikhilesh S. Shinde

B8338570 Sumit S.Wankhede

ABSTRACT

The important task in data mining is the classification of the data into the predefined groups or classes. One of the commonly used classifier technique is Artificial Neural Networks (ANN) [1]. Although ANN usually reaches high classification accuracy, the obtained results sometimes may be incomprehensible [2]. Due to this fact various methods have been proposed to extract the rules from ANN which justifies the classification results given by ANN. For this the trained neural network is pruned for removing the inputs that are not needed for solving the problem. The pruned network serves to filter noise that might be present in the data [2]. The proposed research work shall be based on extracting optimized the rules from ANN which justifies the classification results given by ANN.

Contents

List of Tables

List of Figures

1 INTRODUCTION

Neural Networks are successful in acquiring hidden knowledge in datasets. Their biggest weakness is that the knowledge they acquire is represented in a form not understandable to humans. Researchers tried to address this problem by extracting rules from trained Neural Networks. Most of the proposed rule extraction methods required specialized type of Neural Networks; some required binary inputs and some were computationally Lack of explanation capability is one of the most important reasons why Neural Networks do not get the necessary interest in some parts of the industry.

In most of the real world applications (especially in safety critical applications) users want to know the reasoning behind the conclusion of a learning system or an expert system. Extracting If-Then rules is usually accepted as the best way of extracting the knowledge represented in the Neural Network. Rules extracted from the trained net can be used for explaining the reasoning behind the output of the system. They can also be used in other systems, like expert systems or in systems for discovering previously unknown features in the data (data mining).

Generalization of the system can also be improved by having a better feature representation. Quality of the rules can be measured by Accuracy, Fidelity, and Comprehensibility. Fidelity is the fraction of instances on which Neural Network and the extracted rules give the same output. Comprehensibility is measured by the size of the rule set and by the number of antecedents in each rule.

1.1 Basic Concepts

1.1.1 Data Mining

Data mining (DM), also known as knowledge discovery in databases (KDD), is the process of discovering meaningful patterns in huge databases [3]. In addition, it is also an application that can provide significant competitive advantages for making the right decision. DM is an explorative and complicated process involving multiple iterative steps.

1.1.2 Data Mining Tasks

- Classification : Classifies a data item into one of several predefined categories.
- Regression : Maps a data item to a real-valued prediction variable.
- Clustering : Maps a data item into a cluster, where clusters are natural groupings of data items based on similarity metrics.
- Association rules : Describes association relationship among different attributes.
- Summarization : Provides a compact description for a subset of data.
- Dependency modeling : Describes significant dependencies among variables.
- Sequence analysis : Models sequential patterns, like time-series analysis. The goal is to model the state of the process generating the sequence or to extract and report deviations and trends over time.

1.1.3 Classification Techniques

- Decision Tree based Methods
- Rule-based Methods
- Memory based reasoning
- Genetic Algorithms
- Support Vector Machines
- Neural Networks

1.1.4 Classification

Learn a method for predicting the instance class from pre-labeled (classified) t Instances.

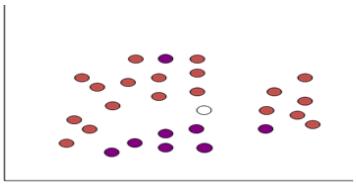

Figure 1: Classification Graph

1.1.5 Decision Trees

An internal node is a test on an attribute. A branch represents an outcome of the test, e.g. Color=red. A leaf node represents a class label or class label distribution. At each node, one attribute is chosen to split training examples into distinct classes as much as possible A new instance is classified by following a matching path to a leaf node.

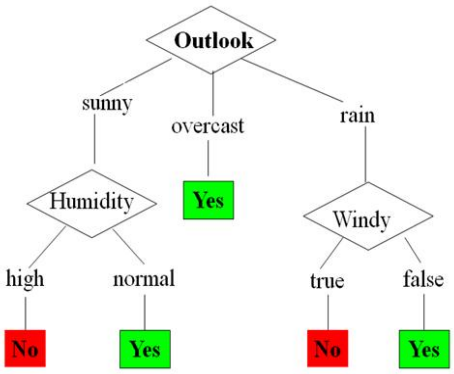

Figure 2: Example Tree for Play?

1.1.6 Neural Network

1. What is Neural Network?

   A neural network is a powerful data modeling tool that is able to capture and represent complex input/output relationships. It is a biologically motivated approach to machine learning.

   Neural networks resemble the human brain in the following two ways:

   - A neural network acquires knowledge through learning.
   - A neural network's knowledge is stored within inter-neuron connection strengths known as synaptic weights.

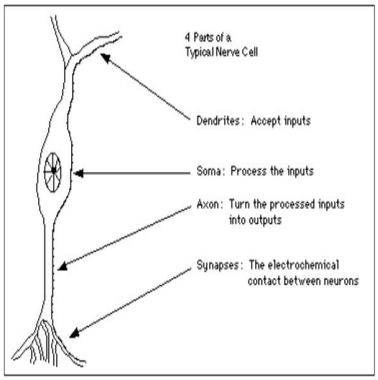

Figure 3: Biological Neuron

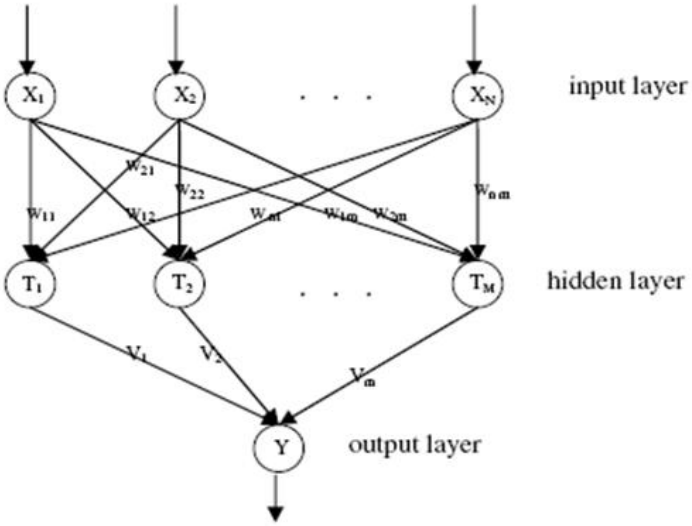

Figure 4: Basic Architecture of Neural Network

2. Back propagation and Feed-Forward NN

   It is a supervised learning method, and is a generalization of the delta rule. It requires a teacher that knows, or can calculate, the desired output for any input in the training set. It is most useful

for feed-forward networks (networks that have no feedback, or simply, that have no connections that loop). The term is an abbreviation for "backward propagation of errors". Back propagation requires that the activation function used by the artificial neurons (or "nodes") be differentiable

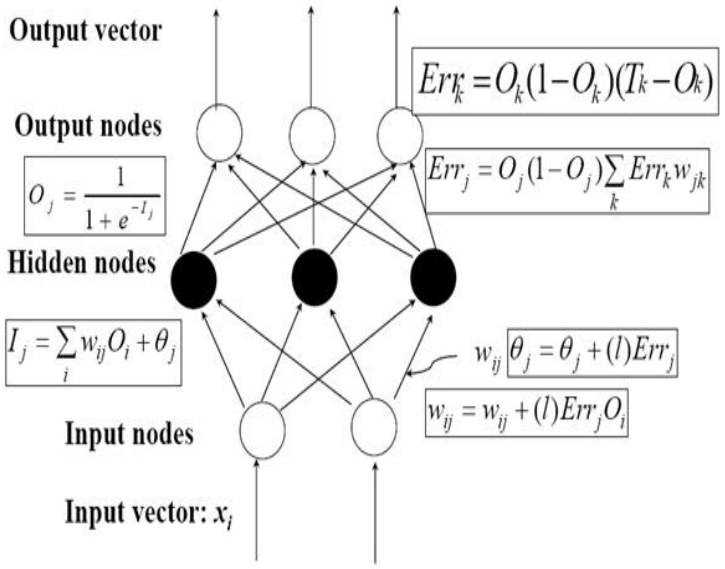

Figure 5: Backproagation Neural Network

The ANN is composed of richly interconnected non-linear nodes that do processing in parallel. The connection weights are modifiable, allowing ANN to learn directly from examples without requiring or providing an analytical solution to the problem. The most popular forms of learning are:

- Supervised learning: Patterns for which both their inputs and outputs are known are presented to the ANN. The task of the supervised learner is to predict the value of the function for any valid input object after having seen a number of training examples. ANN employing supervised learning has been widely utilized for the solution of function approximation and classification problems.
- Unsupervised learning: Patterns are presented to the ANN in the form of feature values. It is distinguished from supervised learning by the fact that there is no a priori output. ANN employing unsupervised learning has been successfully employed for data mining and classification tasks. The self-organizing map (SOM) and adaptive resonance theory (ART) constitutes the most popular examples of this class. A backpropagation network (BPN) is a neural network that uses a supervised learning method and feed-forward architecture. A BPN is one of the most frequently utilized neural network techniques for classification and prediction and is considered an advanced multiple regression analysis that can accommodate

complex and non-linear data relationships.

3. Neural Network and data mining

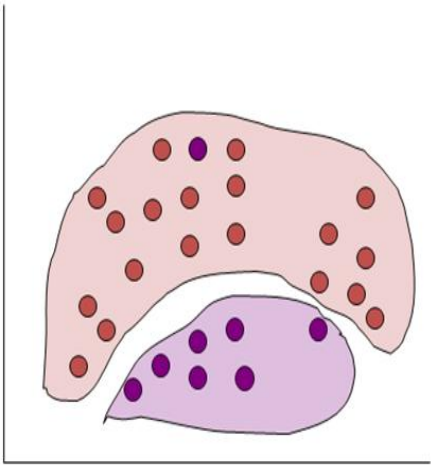

Figure 6: Neural Network in Data Mining

- Can select more complex regions.
- Can be more accurate.
- Also can overfit the data find patterns in random noise.

1.1.7 Application

- Key finance decision making problem e.g. credit card approval.
- Medical Application e.g. cancer patient classification depending upon symptoms attributes.
- Pattern Recognition.
- Image Processing.

2 Literature Survey

To reveal the information concealed in an ANN, researchers have proposed a number of rule extraction techniques. One of the first rule extraction techniques from neural network was proposed by Galant . He was working on connectionist expert systems. In this work, each ANN node is represented as a conceptual entity.

Yueh-Min Huanga and Chun-Min Hunga [8] have addressed the problem of imbalanced class distributions. This paper analyzes different classification algorithms that were employed to predict the creditworthiness of a banks customers based on checking account information. A series of experiments were conducted to test the different techniques. The objective is to determine a range of credit scores that could be implemented by a manager for risk management. Also a strategy of data cleaning for handling such a real case with imbalanced distribution data is presented by them.

Wei-Sen Chen and Yin-Kuan Du have used the artificial neural network (ANN) and data mining (DM) techniques to construct the financial distress prediction model . In this paper traditional Data Mining-Clustering approach is compared with ANN approach. After this experimentation authors have stated that the ANN approach obtains better prediction accuracy than the DM clustering approach. Therefore, the authors have concluded that the artificial intelligent (AI) approach could be a more suitable methodology than traditional statistics for predicting the potential financial distress of a company.

Humar Kahramanli and Novruz Allahverdi [7] have presented the technique of mining the classification rules for Liver Disorders using adaptive activation function. In this study the authors have first trained the neural network with adaptive activation function. Then the rules are extracted from this trained neural network by using the OptaiNET that is an Artificial Immune Algorithm (AIS).

The used Neuro-Adaptive function is as follows:

$$(x) = \frac{A1e^{-x2} + A2}{(1+e^{-B*x})}$$

Where, A1, A2 and B are real variables which will be adjusted during training. In this paper, for rule extraction the first ANN which classifies the dataset was designed. Then Opt-aiNET algorithm was executed for extraction of rules from this ANN. Finally, the extracted rules were decoded. Produced rules diagnosed correctly 192 samples from 200 belong to Class 0 and 135 samples from 145 belongs to Class1. It means system achieve 96In paper [9] association rules have been composed using Apriori algorithm and transactions, which provide these rules, were eliminated. This provides shrinking database. Then ANN has been trained and used Opt-aiNET for composing rule set. Its been observed that this method increased classification accuracy despite decreasing number of rules.

Humar Kahramanli and Novruz Allahverdi [1] have presented the method that uses Artificial Immune Systems (AIS) algorithm to extract rules from trained hybrid neural network [6]. The datasets used are Cleveland heart disease and Hepatitis data taken from UCI machine learning repository.

The performance metrics used in this algorithm are accuracy, sensitivity and specificity which are the common performance metrics used in medical diagnosis tasks. The measure of the ability of the classifier to produce accurate diagnosis is determined by accuracy. The measure of the ability of the model to

identify the occurrence of a target class accurately is determined by sensitivity. The measure of the ability of the model to separate the target class is determined by specificity. So that accuracy, sensitivity and specificity are calculated as follows:

$$Accuracy = \frac{Totalnumberofcorrectlydiagnosedcases}{Totalnumberofcases}$$

$$Sensitivity = \frac{Totalnumberofpositivecasescorrectlydiagnosed}{Totalnumberofpositivecases}$$

$$Specificity = \frac{Totalnumberofnegativecasescorrectlydiagnosed T}{Totalnumberofnegativecases}$$

This method achieves accuracy values of 96.4Miguel Rocha and Paulo Cortez [10] have presented the use of Evolutionary Computation (EC) as a promising alternative for ANN optimization. In this paper two hybrid EC/ANN algorithms are presented: the first evolves neural topologies while the latter performs simultaneous optimization of architectures and weights. Sixteen real-world tasks were used to test these strategies.

Richi Nayak [11] has given a novel methodology Gyan that represents the knowledge of a trained network in the form of restricted first-order predicate rules.

In paper [5] the mathematical properties of several error functions with a focus on MLP data classification are analyzed. This analysis led us to propose two parameterized error functions for MLP training. The first one, ESMF, is a monotonic error function applicable only to two-class problems and was shown to perform equally well as other functions. However, it should be extended to the general multi-class problem for a better evaluation of its capability. The second one, EExp, an exponential-type error function, this function is able to emulate the behaviors of classic error functions and, as a matter of fact, by single parameter adjustment it is able to implement an infinite family of error functions, with different behavior of the error gradient weighting.

By Ajit Narayanan, Edward Keedwell and Dragan Savic [12] have presented a novel approach using genetic algorithms to search for symbolic rules in a trained neural network. Many rule extraction algorithms have been designed to generate classification rules from NNs that have been trained to distinguish data samples from different classes. These algorithms frequently assume that the input data attributes are discrete in order to make the rule extraction process more manageable. NeuroRule [14] is one such algorithm. A component of NeuroRule is an automatic rule generation method called rule generation (RG). Each rule is generated by RG such that it covers as many samples from the same class as possible with the minimum number of attributes in the rule condition. RG is applied to generate rules that explain the networks output in terms of the discretized hidden unit activation values and rules that explain the discretized activation values in terms of the discretized attributes of the input data.

Rule eXtraction (RX) [15] is another NN rule extraction algorithm that works on discrete data. RX recursively generates rules by analyzing the discretized hidden unit activations of a pruned network with one hidden layer. When the number of input connections to a hidden unit is larger than a certain threshold, a new NN is created and trained with the discretized activation values as the target output.

The generalized analytic rule extraction (GLARE) algorithm [16] does not require network pruning. It

does, however, require that the continuous attributes be first converted to nominal, and then to binary attributes before the algorithm can be applied. While the algorithm is shown to work well on discrete data sets, it does not perform as well as decision tree methods on data sets with continuous attributes.
The orthogonal search-based rule extraction (OSRE) method [17] assumes that the data has been 1-from-N encoded; hence its application is also limited to data with only nominal or ordinal attributes.
Trepan, an algorithm that was developed by Craven and Shavlik [18] also extracts M-of-N rules from an NN. It treats the NN as an oracle to generate additional data samples. This is an important step in trepan as it grows a decision tree by recursive partitioning. As the tree grows, fewer and fewer training samples are available for deciding if a node should be split further. Additional samples are generated by taking into account the distribution of the existing data samples and their labels are determined by the NN oracle. A node in the tree becomes a leaf node if it has sufficiently a high proportion of samples that belong to one class or if the number of internal nodes in the tree has reached the maximum.
There exist algorithms that generate fuzzy rules from NNs. Nefclass [19] is one such algorithm. It works as a three-layer fuzzy NN. The difference between this type of networks and the standard backpropagation NNs lies in the connection weights which represent fuzzy sets and in the activation functions which act as fuzzy set operators. Nefclass employs a fuzzy variant of the backpropagation algorithm to find the characteristic parameters of the membership functions.
Benitez [20] proposed a method for translating the knowledge embedded in an NN into a fuzzy-rule-based system.
Tsukimoto [21] developed an NN rule extraction algorithm that could be applied to continuous data directly. Instead of linear combinations of the inputs as the rule conditions, the rules are represented as continuous Boolean function of the attributes. The algorithm has been tested only on the Iris data set and the accuracy obtained was not as good as the accuracy of the decision tree method C4.5.
The algorithm full rule extraction (full-RE) [22] is able to extract accurate rules from the Iris data set without the need to binarize or normalize the continuous attributes of the data prior to network training. For each hidden node in the network, full-RE generates an intermediate rule that predicts its output according to the linear combinations of the input attributes. In order to obtain the final set of classification rules that do not involve network weights in their rule conditions, full-RE discretizes the input attributes and then solves a linear programming problem to select the relevant discretization boundary.
A recent rule extraction algorithm that works on discrete and continuous data was proposed by Rabual [23]. There is, however, no provision for generating rules with separate rule conditions involving discrete and continuous attributes.
Rudy Setiono and Bart Baesens [2] have presented a recursive algorithm for extracting classification rules (RE-RX) from feedforward neural networks that have been trained on data sets having both discrete and continuous attributes. This algorithm shares some similarities with other existing rule extraction algorithms. It assumes that the trained network has been pruned so that irrelevant and redundant network connections and units have been removed. It also makes use of the decision tree method C4.5 to generate rules with only discrete attributes in their conditions. The novel feature of the proposed recursive algorithm lies in the rule set generated. The rules are hierarchical such that only those rules

at the deepest level have rule conditions that involve linear combinations of the continuous attributes, while the conditions of all the other rules involve only the discrete attributes. We believe that such rule conditions would greatly increase the comprehensibility of the rules, and hence greatly pave the way to open the NN black box wider.

3 System requirement and specification

3.1 Project Statement

Mining classification rules from database using ANN and optimizing the knowledge generated in terms of classification accuracy.

3.2 Proposing theme of the project

Recursive Neural Network Rule Extraction for Data with Mixed attributes By Rudy Setiono,Senior Member, IEEE, Bart Baesens, and Christophe Mues [IEEE Transaction on Neural Networks, Vol. 19 No. 2, February 2008 pp. 299-307] In this paper, a recursive algorithm for extracting classification rules from feedforward neural networks that have been trained on data sets having both discrete and continuous attributes is presented Real-world classification problems usually involve both discrete and continuous input attributes. For such problems, all the continuous attributes must be discretized if algorithms such as NeuroRule, GLARE, and OSRE are to be used for rule extraction.The drawback of discretizing the continuous attributes is that the accuracy of the networks, and hence the accuracy of the rules extracted from the networks, may decrease. This is because discretization leads to a division of the input space into hyperrectangular regions. Each condition of the extracted rulescorresponds to one of these hyperrectangular regions where all data samples are predicted to belong to one class. Clearly, a data preprocessing step that divides the input space into rectangular subregions may impose unnecessary restrictions on the NNs as classifiers. It is highly likely that the boundaries of the regions that contain data samples from the same class are nonrectangular given that some of the data attributes are continuous.

3.3 Overall Description

The proposed system presents a justified neural network using rule extraction for classifying pima Indian diabetes patiens.System provides recursively extracted hierarchical rules for describing hidden knowledge of the trained neural network. A recursive rule extraction algorithm proposed by Rudy Setiono is used for implementing this system

3.3.1 Overall Steps

1. Step 1: (Data collection) The data to be used for classification is collected.
2. Step 2: (Training and testing data separation) The available data are divided into training and testing data sets of size 80% and 20% respectively.
3. Step 3: (Network architecture) network architecture and a learning method are selected. Important considerations are the exact number of perceptrons and the number of layers.

4. Step 4: (Parameter tuning and weight initialization) There are parameters for tuning the network to the desired learning performance level. Part of this step is initialization of the network weights and parameters, followed by modification of the parameters as training performance feedback is received.

   Initialize weight and biases to the random numbers distributed over a small range of values: $[\frac{-\alpha}{\sqrt{Ni}}, \frac{-\alpha}{\sqrt{Ni}}]$

   $Where\, N_i - No.\, of\, units\, to\, i^{th}\, unit,$

   α - integer between 1 to 3.

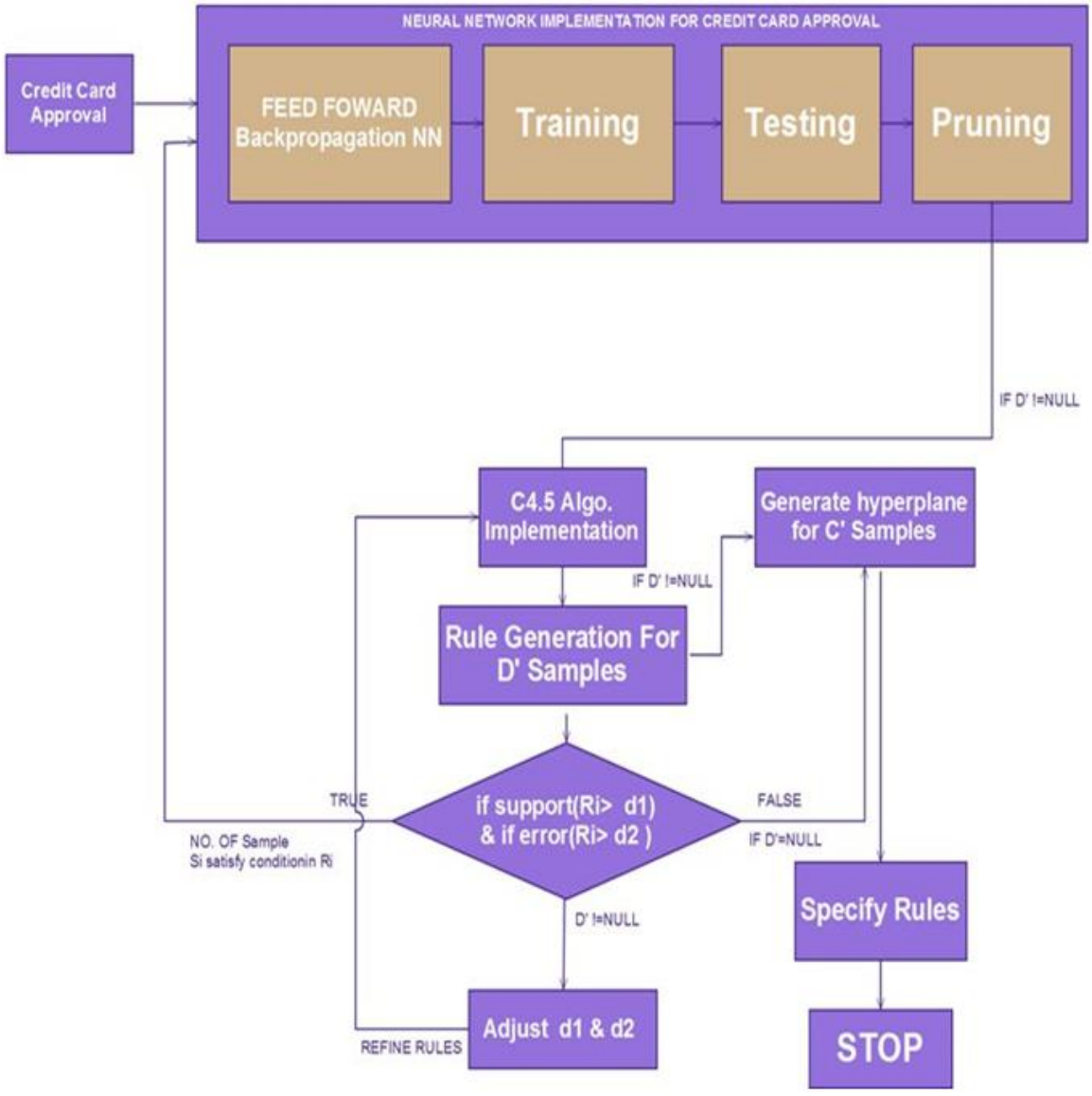

Figure 7: Neural Network Implementation

5. Step 5: (Data Normalization) transforms the application data into the type and format required by the ANN. All data must be normalized i.e. all values of the attributes in the database are changed to contain in the interval [0,1] or [-1,1]. Two normalization techniques are used:

(a) Max-Min Normalization.

(b) Decimal scaling Normalization.

6. Step 6: (Training) Training is conducted iteratively by presenting input and desired or output data to the ANN. The ANN computes the outputs and adjusts the weights until the computed outputs are within an acceptable tolerance of the known outputs for the input cases.

7. Step 7: (Testing) The testing examines the performance of the network using the derived weights by measuring the ability of the network to classify the testing data correctly.

8. Step 8: (Implementation) Now a stable set of weights are obtained. Now the network can reproduce the desired output for the given inputs like those in the training set. The network is ready to use as a stand-alone system or as part of another software system where new input data will be presented to it and it output will be a recommended decision.

3.3.2 Pima Indian Diabetes Dataset

1. Title: Pima Indians Diabetes Database
2. Sources: Original owners: National Institute of Diabetes and Digestive and Kidney Diseases
3. Past Usage: Smith, J. W., Everhart, J. E., Dickson, W. C., Knowler, W. C., & Johannes, R. S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Symposium on Computer Applications and Medical Care* (pp. 261–265). IEEE Computer Society Press.
4. Number of Instances: 768
5. Number of Attributes: 8 plus class
6. For Each Attribute: (all numeric-valued)
   (a) Number of times pregnant.
   (b) Plasma glucose concentration a 2 hours in an oral glucose tolerance test
   (c) Diastolic blood pressure (mm Hg)
   (d) Triceps skin fold thickness (mm)
   (e) 2-Hour serum insulin (mu U/ml)
   (f) Body mass index (weight in kg/ $(height\,in\,m)^2$)
   (g) Diabetes pedigree function
   (h) Age (years)
   (i) Class variable (0 or 1)
7. Missing Attribute Values: Yes
8. Class Distribution: (class value 1 is interpreted as "Tested positive for diabetes")

Class Value	For Number of instances
0	500
1	268

Table 1: Class Distribution

9. Brief statistical analysis:

Attribute number	Mean	Deviation
1	3.8	34
1	3.8	3-4
2	120.9	32.0
3	69.1	19.4
4	20.5	16.0
5	79.8	115.2
6	32.0	7.9
7	0.5	0.3
8	33.2	11.8

Table 2: Statistical analysis of the dataset

3.3.3 Discrete and continuous attributes

- **Discrete Attributes:** Any data measurements that are not quantified on and infinitely divisible numeric scale is called discrete attributes.
- **Continuous Attributes:** Any data measurements that are quantified on and infinitely divisible numeric scale is called continues attributes.

3.3.4 Feed forward Neural Network

Neural network training:

Find an optimal weight (W,V).Minimize a function that measures how well the network predicts the desired outputs (class label).

- Error in prediction for $i^{th} sample : e^i = (desiredoutput)^i - (predictedoutput)^i$
- Sum of squared error function: $E(W,V) = \sum e^i$
- Crossentropy error function:
  $E(W,V) = \sum d_i log p_i + (1 - di) log(1 - p_i)$
  d is the desired output either 0 or 1 d_i output, 1 Many optimization methods.
- Gradient descent/error back propagation
- Conjugate gradient
- Quasi Newton method
- Genetic algorithm
- Network is considered well trained if it can predict training data and cross validation data with acceptable accuracy.

3.3.5 Pruning

Artificial neural networks are considered as efficient computing models and as the universal approximates .The predictive accuracy of neural network is higher than that of other methods or human experts, it is generally difficult to understand how the network arrives at a particular decision due to the complexity of a particular architecture. One of the major criticism is their being black boxes, since no satisfactory explanation of their behavior has been offered. This is because of the complexity of the interconnections between layers and the network size [18]. As such, an optimal network size with minimal number of interconnection will give insight into how neural network performs. Another motivation for network simplification and pruning is related to time complexity of learning time.

1. **Pruning Algorithm**

   Network pruning offers another approach for dynamically determining an appropriate network topology. Pruning techniques [11] begin by training a larger than necessary network and then eliminate weights and neurons that are deemed redundant. Typically, methods for removing weights involve adding a penalty term to the error function [5]. It is hoped that adding a penalty term to the error function, unnecessary connection will have smaller weights and therefore complexity of the network can be significantly reduced. This paper aims at pruning the network size both in number of neurons and number of interconnections between the neurons. The pruning strategies along with the penalty function are described in the subsequent sections

2. **Redundant weight Pruning**

   Penalty function is used for weight decay. As such we can eliminate redundant weights with the following Weight Elimination Algorithm as suggested in different literature.

   (a) Weight Elimination Algorithm:

      i. Let $\eta_1 \, and \, \eta_2 \, be \, positive \, scalars \, such \, that \, \eta_1 + \eta_2 < 0.5$.

      ii. Pick a fully connected network and train this network such that error condition is satisfied by all input patterns. Let (w, v) be the weights of this network.

      iii. For each $mmlv \, , if \, mmlvw \, \leq \, 4 \, \eta_2 \, (\sigma), then \, remove \, mmlv$ from the network

      iv. For each vm , If vm $\leq 4 \, \eta_2(7)$,then remove vm from the Network.

      v. If no weight satisfies condition (6) or condition (7) then remove m wl with the smallest product m m l v w .

      vi. Retrain the network. If classification rate of the network falls below an acceptable level then stop. Otherwise go to Step 3.

   (b) Input and Hidden Node Pruning :

   A node-pruning algorithm is presented below to remove redundant nodes in the input and hidden layer.

   step 1 : Create an initial network with as many input neurons as required by the specific problem description and with one hidden unit. Randomly initialize the connection weights of the network within a certain range.

step 2 : Partially train the network on the training set for a certain number of training epochs using a training algorithm. The number of training epochs, , is specified by the user.

step 3 : Eliminate the redundant weights by using weight elimination algorithm.

step 4 : Test this network. If the accuracy of this network falls below an acceptable range then add one more hidden unit and go to step 2.

step 5 : If there is any input node xl with mlw = 0, for m = 1,2,....,h. then remove this node.

step 6 : Test the generalization ability of the network with test set. If the network successfully converges then terminate, otherwise, go to step 1.

3.3.6 Decision Tree Algorithm

Decision tree learning, used in statistics, data mining and machine learning, uses a decision tree as a predictive model which maps observations about an item to conclusions about the item's target value. More descriptive names for such tree models are classification trees or regression trees. In these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels.

In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data but not decisions; rather the resulting classification tree can be an input for decision making. This page deals with decision trees in data mining.

3.3.7 C 4.5 Algorithms

C4.5 is an algorithm used to generate a decision tree developed by Ross Quinlan. C4.5 is an extension of Quinlan's earlier ID3 algorithm. The decision trees generated by C4.5 can be used for classification, and for this reason, C4.5 is often referred to as a statistical classifier.

C4.5 pseudo code:

1. Check for base cases.
2. For each attribute "a" find the normalized information gain from splitting on "a".
3. Let, a_best be the attribute with the highest normalized information gain.
4. Create a decision node that splits on a_best.
5. Recurse on the sub list obtained by splitting on a_best a_best and add those nodes as children of node.

3.3.8 Support Vector machine

A support vector machine constructs a hyper plane or set of hyper planes in a high or infinite dimensional space, which can be used for classification, regression, all other tasks. Intuitively a good separation is achieved by hyper plane that has a largest distance to any nearer training data point of any class (so called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

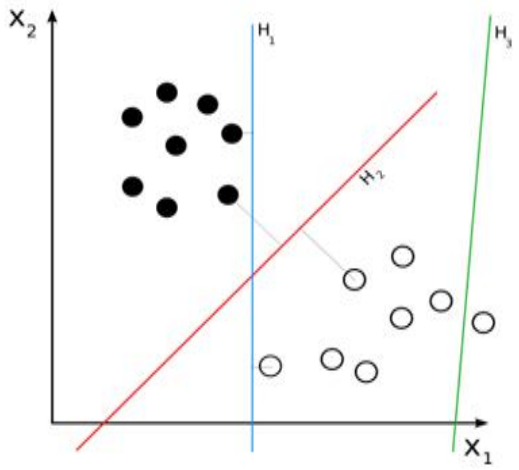

Figure 8: Support Vector Machine

3.3.9 Support and Error

- **Support:** Support refers to the number of correctly classified samples by the rule, divided by the number of correctly classified samples by the NN.
- **Error:** Error refers to number of wrong classified samples by the rule, divided by number of samples in the rule.

3.4 Software Requirements

3.4.1 Functional Requirement

Creating a User Interface for demonstrating diabetes patients classification in positive or negative test with justification.

- The process involves in step one of module one is creating feed-forward neural network with input, hidden and output layer and training of it using appropriate training algorithm. Defining goal and hidden layers neurons for achieving good accuracy.
- In the second step pruning process applied on trained network to prune the unwanted and zeroed valued network connections with attributed values.
- Second module involves use of decision tree algorithm for discrete rule generation form trained network .Support of each rule is validated using separate C module for rule refinement.
- Third module involves use of support vector machine for hyper plane generation for contentious attributes.

3.4.2 Non-Functional Requirement

The project requires standard dataset of the Pima Indian diabetes patients.

- **Maintainability:** All modules interfaces must be clearly define to change for the future technologies. Through thoughtful and effective software engineering, all steps of the software development process will be well documented to ensure maintainability of the product throughout its life time. All development will be provided with good documentation.
- **Performance:** The response time, utilization and throughput behavior of the system. Care is taken so as to ensure a system with comparatively high performance.
- **Usability:** The ease of use and training the end users of the system is usability. System should have qualities like- learning ability, efficiency, affect, control. The main aim of the project is to reduce server failure and gets high performance of server and reduces the rework of the programmer.
- **Modifiability:** The ease with which a software system can accommodate changes to its software is modifiability. Our project is easily adaptable for changes that is useful for the application to withstand the needs of the users.
- **Portability:** The ability of the system to run under different computing environments. The environment types can be either hardware or software, but is usually a combination of two.
- **Reusability:** The extent to which existing application can be reused in new application. Our application can be reused a number of times without any technical difficulties.
- **Security:** The factors that protect the software from accidental or malicious access, use, modification, destruction, or disclosure. Security can be ensured as the project involves authenticating the users.

3.5 Methodologies/Techniques to be used for knowledge extraction

The Multilayer Perceptrons are used in this study make use of biases, sigmoid activation functions and one hidden layer with a variable number of nodes.

- Back propagation algorithm for training ANN (Gradient steepest descent algorithm)
- C 4.5 algorithm
- MATLAB (MATrix LABoratory) Neural Network Toolbox 4.0 from Math Works Corp. is applied to perform the training exercises.
- Neural network packages: Neurodimensions Neurosolutions v3.0 and Thinkspro v1.05 by Logical Designs Consulting.
- Support Vector Machine algorithm
- Credit Approval dataset from UCI machine repository.

3.6 Features

- Easy user interface
- Online database creation for new and unknown data samples.
- Clear justification of the test report using rule extracted from the network..
- High accuracy on standard dataset.
- Can handle unknown and null values samples.
- Performance is good as advances training algorithm is used.

4 PROJECT PLANNING AND MANAGEMENT

4.1 Project Process Management

The V-model represents a software development process (also applicable to hardware development) which may be considered an extension of the waterfall model. Instead of moving down in a linear way, the process steps are bent upwards after the coding phase, to form the typical V shape. The V-Model demonstrates the relationships between each phase of the development life cycle and its associated phase of testing. The horizontal and vertical axes represents time or project completeness (left-to-right) and level of abstraction (coarsest-grain abstraction uppermost), respectively.

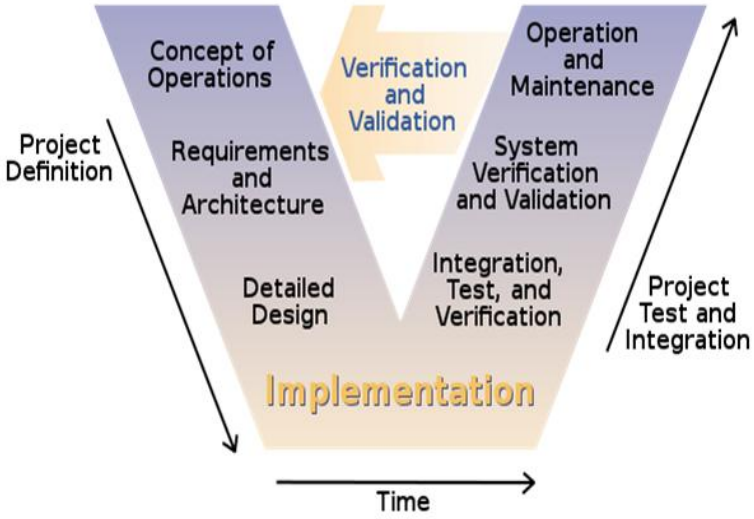

Figure 9: V Life Cycle Model

4.2 Verification Phases

4.2.1 Requirements analysis

In the Requirements analysis phase, the first step in the verification process, the requirements of the proposed system are collected by analyzing the needs of the user(s). This phase is concerned with establishing what the ideal system has to perform. However it does not determine how the software will be designed or built. Usually, the users are interviewed and a document called the user requirements document is generated.

The user requirements document will typically describe the systems functional, interface, performance, data, security, etc requirements as expected by the user. It is used by business analysts to communicate their understanding of the system to the users. The users carefully review this document as this document would serve as the guideline for the system designers in the system design phase. The user acceptance tests are designed in this phase. See also Functional requirements. this is parallel processing.

There are different methods for gathering requirements of both soft and hard methodologies including; interviews, questionnaires, document analysis, observation, throw-away prototypes, use cases and status and dynamic views with users.

4.2.2 System Design

Systems design is the phase where system engineers analyze and understand the business of the proposed system by studying the user requirements document. They figure out possibilities and techniques by which the user requirements can be implemented. If any of the requirements are not feasible, the user is informed of the issue. A resolution is found and the user requirement document is edited accordingly.

The software specification document which serves as a blueprint for the development phase is generated. This document contains the general system organization, menu structures, data structures etc. It may also hold example business scenarios, sample windows, reports for the better understanding. Other technical documentation like entity diagrams, data dictionary will also be produced in this phase. The documents for system testing are prepared in this phase.

4.2.3 Architecture Design

The phase of the design of computer architecture and software architecture can also be referred to as high-level design. The baseline in selecting the architecture is that it should realize all which typically consists of the list of modules, brief functionality of each module, their interface relationships, dependencies, database tables, architecture diagrams, technology details etc. The integration testing design is carried out in the particular phase.

4.2.4 Module Design

The module design phase can also be referred to as low-level design. The designed system is broken up into smaller units or modules and each of them is explained so that the programmer can start coding directly. The low level design document or program specifications will contain a detailed functional logic of the module, in pseudocode:

- Database tables, with all elements, including their type and size.
- All interface details with complete API references.
- All dependency issues.
- Error message listings.
- Complete input and outputs for a module.
- The unit test design is developed in this stage.

4.2.5 Module Design

- Unit Testing
- Integration Testing
- System Testing
- User Acceptance Testing
- Acceptance testing helps

4.2.6 Feasibility Study

Literature survey basically focuses on answer to, how well our project does satisfy the four dimensions:

1. Technology Feasibility
2. Economic Feasibility
3. Time Feasibility
4. Resources Feasibility

Answer to these questions could be very easy for projects in established area where we get lots of time and resources to use and as the risks increases the feasibility study would result Negative. As far our current project is concerned we have carried out the initial risk architecture and design so, that the above questions would be accurately answered.

1. **Technology Feasibility:**
   The project needs knowledge of MATLAB,Data Mining,Neural Network,Decision Tree Algorithm,Weka tool. As we have enough resources of both the technologies, we are capable to satisfy the technological needs of the project completely as well we do not need additional software or hardware to develop application.
2. **Financial Feasibility:**
   As this project does not need any special hardware or licensed software and as the developer are going through their training, financial feasibility is not the problem for the organization. Moreover the organization already has enough infrastructures to support their application well so there is no need for any finance.
3. **Time Feasibility:**
   The project has to be completed by 10 months approx., as we have gathered requirements and analyzed them properly; it seems that project will be completed in time span provided. Therefore the project is feasible in time.
4. **Resource Feasibility:**
   The hardware and software components and human resources availability is sufficient enough to engineer the project. Hence the software is feasible considering resources.

4.3 Cost and Effort Estimation

- **Total Resources Estimation :**

  We know that the cost of developing software, up until the point it is accepted, is only a fraction of the total cost of the system over the typical life cycle of the product. However for the purpose of this study we will exclude to maintenance costs, and will speak only of the development costs up until acceptance.

- **Functions:**

  Cost estimation based on expected functionality of the system was first purposed by Albrecht in 1979, and has since been researched by several people. This Cost estimation realizes on function points and requires the identification of all occurrences of five unique functions types :External Input ,External Output ,Logical internal files ,External Interfaces and queries .The sum of all occurrences is called RAW-FUNCTION-COUNTS (FC).This value must be modifies by the weighted rating of complexity factors ,giving a Technical Complexity Factor (TCF).The function points are equivalent to FC*TCF for any given project.

  - Estimation of Hardware
    - Minimum Requirement for Development
    - Pentium 4 Processor
    - 1 GB Of RAM
    - 80 GB Hard Disk Space
  - Minimum Requirement for Execution
    - Pentium 4 Processor
    - 1 GB Of RAM
    - 80 GB Hard Disk Space on server
  - Estimation Of Software
    - Software Resources
    - MATLAB 7 or Higher version
    - WEKA

4.4 Risk analysis and Management

Risk management is the identification, assessment, and prioritization of risks (defined in ISO 31000 as the effect of uncertainty on objectives, whether positive or negative) followed by coordinated and economical application of resources to minimize, monitor, and control the probability and/or impact of unfortunate events or to maximize the realization of opportunities.

Risks can come from uncertainty in financial markets, project failures (at any phase in design, development, production, or sustainment life-cycles), legal liabilities, credit risk, accidents, natural causes

and disasters as well as deliberate attack from an adversary, or events of uncertain or unpredictable root-cause.

The strategies to manage risk typically include transferring the risk to another party, avoiding the risk, reducing the negative effect or probability of the risk, or even accepting some or all of the potential or actual consequences of a particular risk. Risk management also faces difficulties in allocating resources. This is the idea of opportunity cost. Resources spent on risk management could have been spent on more profitable activities. Again, ideal risk management minimizes spending (or manpower or other resources) and also minimizes the negative effects of risks.

4.4.1 Method

For the most part, these methods consist of the following elements, performed, more or less, in the following order.

- Identify, characterize, and assess threats.
- Assess the vulnerability of critical assets to specific threats.
- Determine the risk (i.e. the expected likelihood and consequences of specific types of attacks on specific assets).
- Identify ways to reduce those risks.
- Prioritize risk reduction measures based on a strategy.

4.4.2 Risk management activities as applied to project management

In project management, risk management includes the following activities:

- Planning how risk will be managed in the particular project. Plans should include risk management tasks, responsibilities, activities and budget.
- Assigning a risk officer - a team member other than a project manager who is responsible for foreseeing potential project problems. Typical characteristic of risk officer is a healthy skepticism.
- Maintaining live project risk database. Each risk should have the following attributes: opening date, title, short description, probability and importance. Optionally a risk may have an assigned person responsible for its resolution and a date by which the risk must be resolved.
- Creating anonymous risk reporting channel. Each team member should have the possibility to report risks that he/she foresees in the project.
- Preparing mitigation plans for risks that are chosen to be mitigated. The purpose of the mitigation plan is to describe how this particular risk will be handled what, when, by who and how will it be done to avoid it or minimize consequences if it becomes a liability.
- Summarizing planned and faced risks, effectiveness of mitigation activities, and effort spent for the risk management.

4.4.3 Project Scheduling (timeline chart)

Index	State	Tasks	Time
1	Requirement Gathering	Requirements understanding, Use case design, functional requirements	7 weeks
2	Analysis	Design or select Architecture, framework Development, use case realization,Data Set preprocessing	7 weeks
3	Design	Develop Design model, MATLAB Network model designing,System Desingning	11 weeks
4	Implementation	Environmen setup, production rollout	5 weeks
5	Testing	Goal matching,Fact table analysis,Accuracy mapping,Rule refinement	4 weeks
6	Deployment	Integration of modules and deployment of application	2 Week

Table 3: Plan of Execution

Gantt chart:

Figure 10: Gantt chart

Timeline chart:

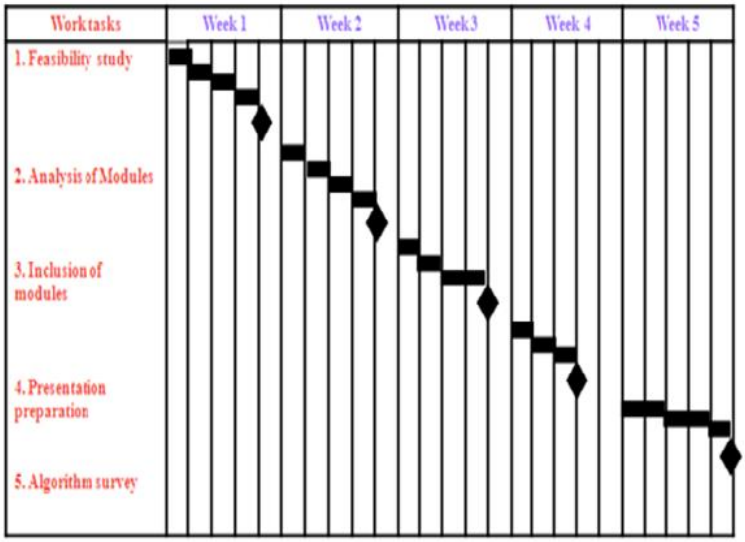

Figure 11: Timeline chart

5 ANALYSIS AND DESIGN

Design modeling uses a combination of text and diagrammatic forms to depict the requirements for data, function and behavior in a way that s relatively easy to understand and more importantly straightforward to review for correctness, completeness and consistency.

The unified modeling language (uml) is a graphical language for visualization, specifying, constructing and documenting the artifacts of a software intensive system. The uml gives a standard way to write systems blue prints, covering conceptual things, such as business processes and system functions, as well as concrete things, such as classes written in a specific programming language, database schemas and reusable software components.

Here we have included the following uml diagrams:

5.1 Use Case Diagram

A use case diagram is a type of behavioral diagram defined by the Unified Modeling Language (UML) and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actors. Roles of the actors in the system can be depicted.

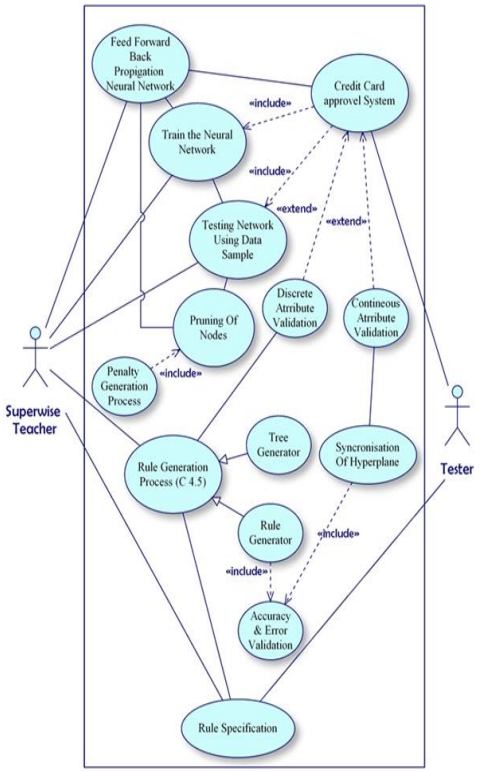

Figure 12: Use cases

5.2 Activity diagram

An activity diagram is a diagram that shows activities and actions to describe workflows. In the Unified Modeling Language an activity diagram represents the business and operational step-by-step workflows of components in a system.

An activity diagram shows the overall flow of control.Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

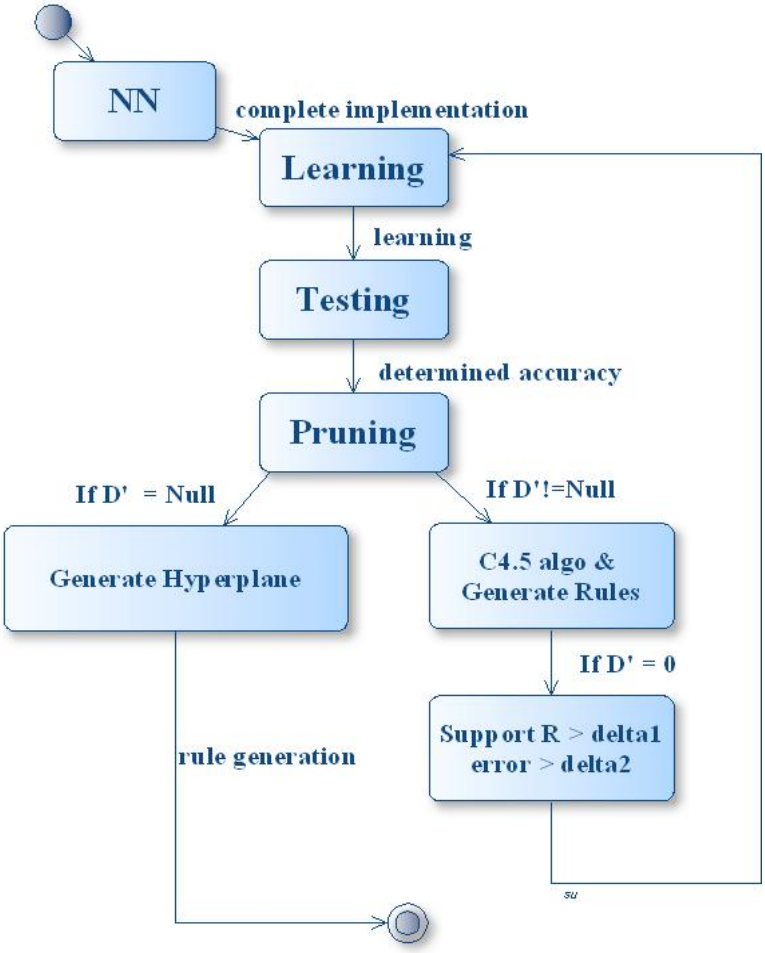

Figure 13: Activity diagram

6 IMPLEMENTATION AND USER INTERFACES

6.1 Network Architecture

The first step in training a feedforward network is to create the network object. The function newff creates a feedforward network. It requires four inputs and returns the network object.

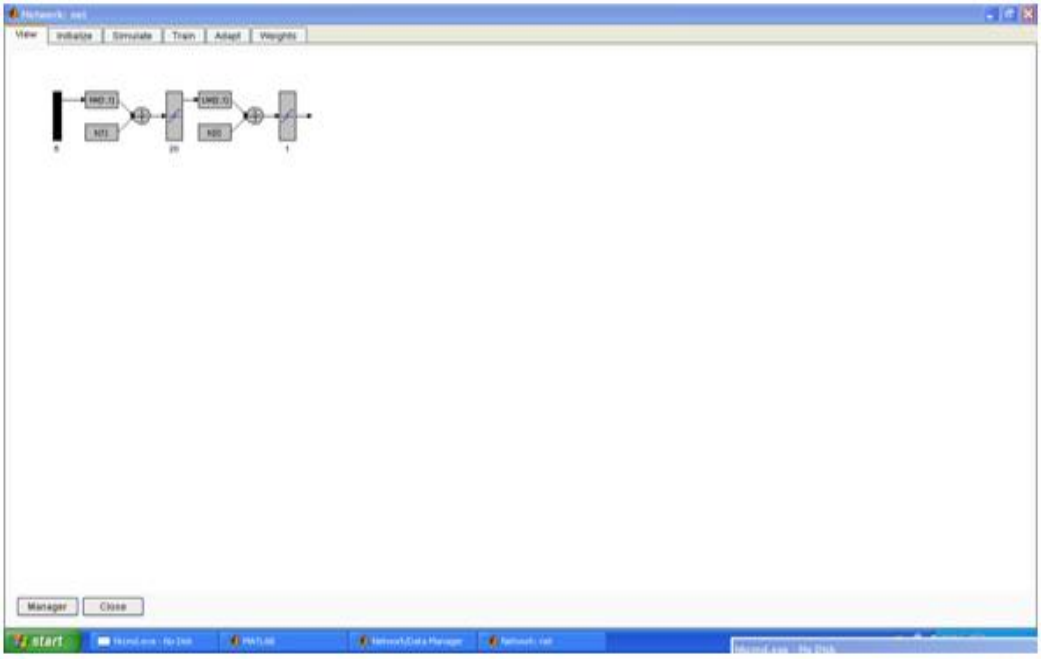

Figure 14: ANN Architecture in MATLAB

6.2 Network setup

Network training involves numbers of attributes those are essential to achieve higher accuracy and optimal network.

- **Network learning rate:** 0.1
- **Numbers of epoch:** 1, 83525
- **Network mean square error:** 0.1
- **Goal achieved:** 0.099999

6.3 Training of Neural network

The training process requires a set of examples of proper network behavior - network inputs p and target outputs t. During training the weights and biases of the network are iteratively adjusted to minimize the network performance function net.performFcn.

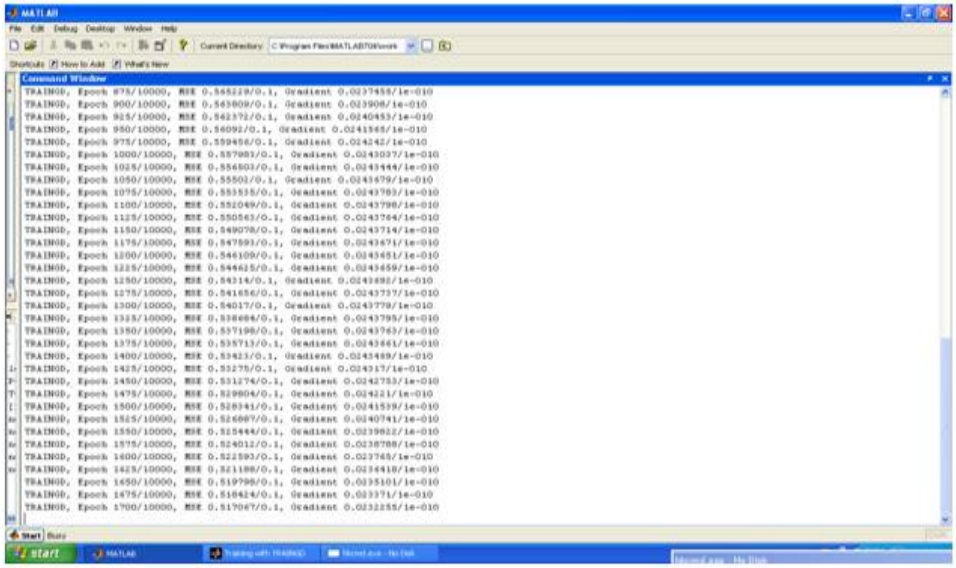

Figure 15: ANN Training Process in MATLAB

6.4 Training graph

The result of training is shown in the following figure.

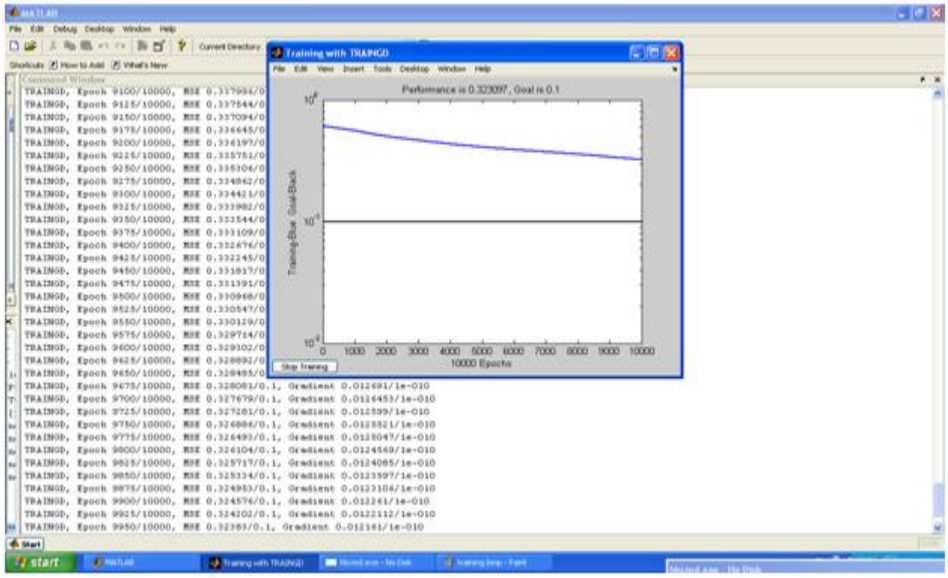

Figure 16: ANN Training Graph in MATLAB

6.5 Input & Target Co-relation Graph

Graph showing deviation between target and actual output.

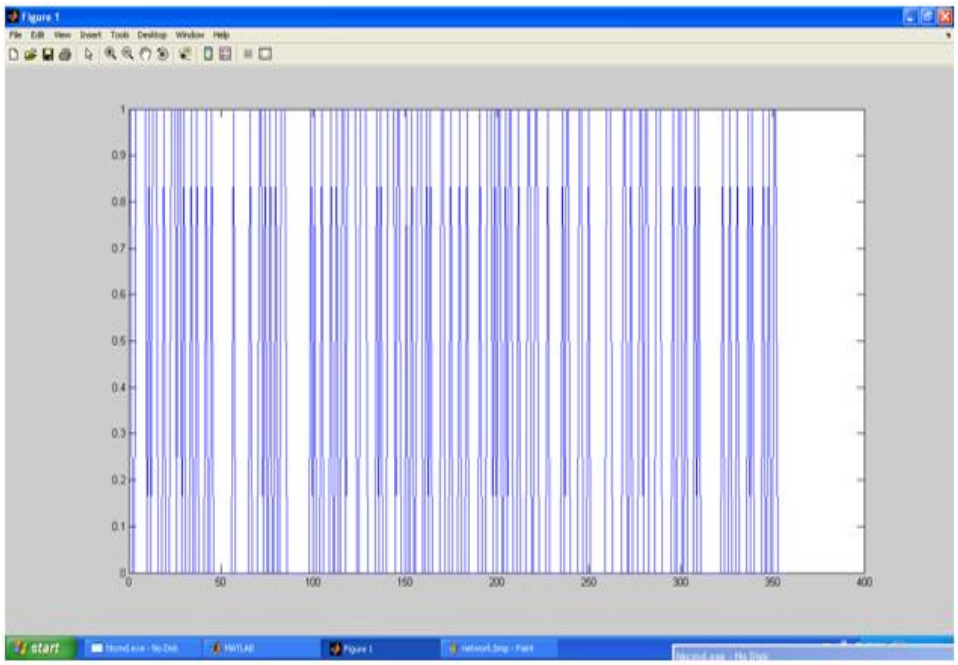

Figure 17: Deviation Graph in MATLAB

6.6 Weka Decision tree Statistics

Statesticis information generated by Weka on dataset using C4.5 algorithm.

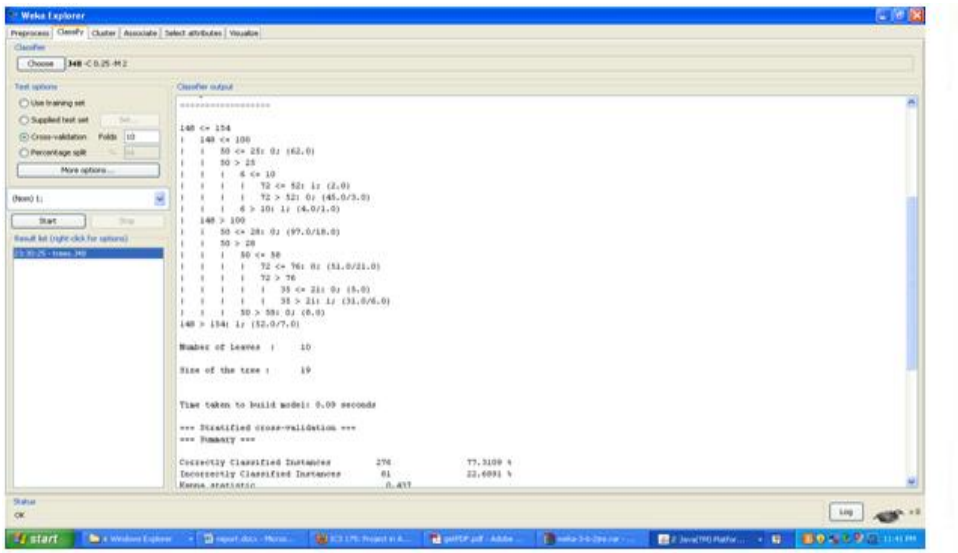

Figure 18: Weka Decision tree Statistics

6.7 Decision tree generated by Weka tool

Figure 19: Decision tree generated by Weka tool

6.8 GUI of the System

GUI tool for classifying user in positive or negative test with justification of NN output.

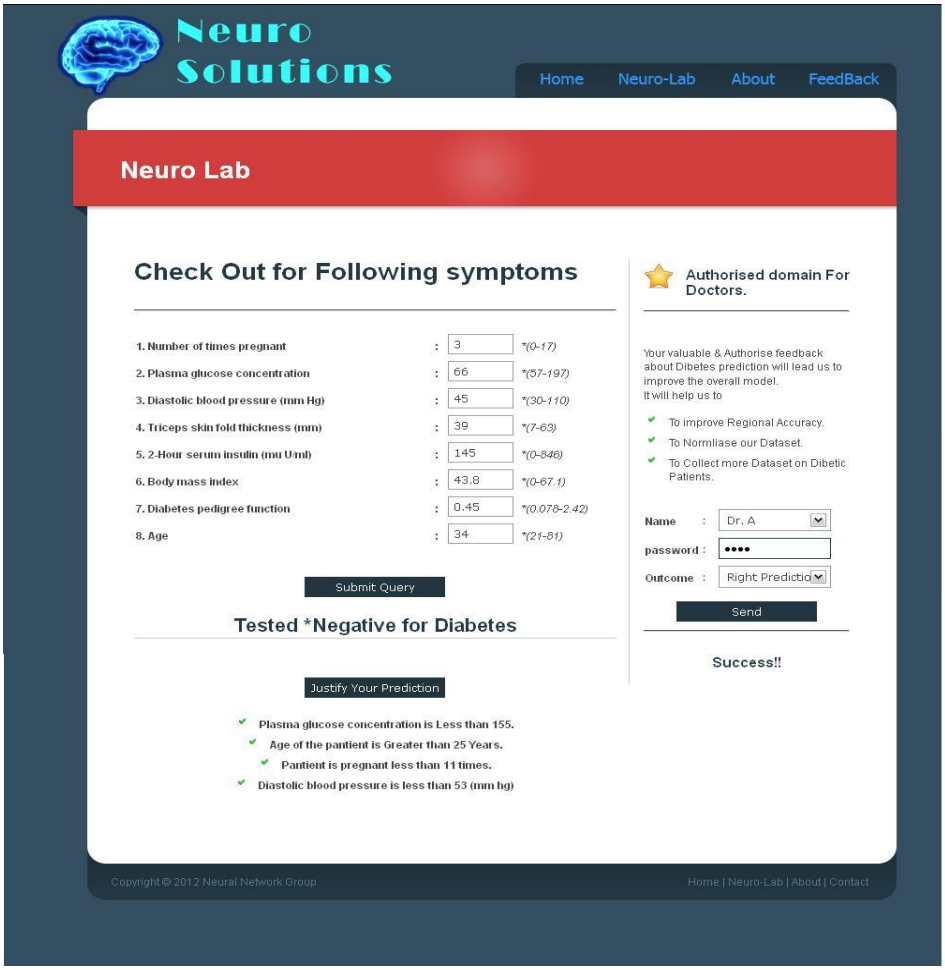

Figure 20: Graphical User Interface

6.9 Code

6.9.1 Main Classification module with GUI

- **index.jsp**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Page - Diabetis Prediction Model</title>
<link rel="stylesheet" href="styles/style.css" type="text/css" media="screen" />
</head>
<body>

<div id="wrapper">
<div id="header">
<h1>Diabetis Prediction Model</h1>

<div id="nav">
<ul>
<li><a href="index.jsp" id="nav1">Home</a></li>
<li><a href="page.jsp" id="nav3">Neuro-Lab</a></li>
<li><a href="about.jsp" id="nav4">About</a></li>
<li><a href="contact.jsp" id="nav5">Contact</a></li>
</ul>

</div>
<div class="clear"></div>
</div>

<div id="middlebar-small">
<h2>Neuro Lab</h2>
</div>

<div id="body">
<div id="body-inner">
<div id="sidebar">

    <h3 class="icon-feature">Authorised domain For Doctors.</h3>
<hr></br></br>
```

```
<p>Your valuable & Authorise feedback about Dibetes prediction will lead us to
                    improve the overall model.</br>
   It will help us to</br>
                </p>
            <ul>
<li>To improve Regional Accuracy.</li>

<li>To Normliase our Dataset.</a></li>
<li>To Collect more Dataset on Dibetic Patients.</li>
</ul>
                </br>
                <form id="form2" name="form2" method="post" action="SQLops_main.jsp">
<ul class="Dform">
<li><label for="Select Name">Name</label><font>:</font> 
  <select name="Uname">

                      <option value="Dr.A">Dr. A</option>
                      <option value="Dr.B">Dr. B</option>
                      <option value="Dr.C">Dr. C</option>
                      <option value="Dr.D">Dr. D</option>
                      </select>
</li>
<li><label for="password">password</label><font>:</font> 

<input name="password" type="password" /></li>
<li><label for="Select Name">Outcome</label><font>:</font> 
  <select name="selectResult">
                      <option value="1">Right Prediction</option>
                      <option value="0">Wrong Prediction</option>
                      </select>
</li>
</br>
<li><center><input type="submit" value="Send" class="button" /></center>

</ul>
    <hr>
</form>
</br><center><h3>Success!!</h3></center>
</div>
```

```
<div id="main-content">
<form id="form1" name="form1" method="post" action="rules.jsp" >
<h1>Check Out for Following symptoms</h1>
</br><hr></br>

<ul class="form">
<li><label>1. Number of times pregnant</label><B>:</B>  <input name="ntp" type="text" value=
<li><label>2. Plasma glucose concentration</label><B>:</B>  <input name="pgc" type="text" va
<li><label>3. Diastolic blood pressure (mm Hg)</label><B>:</B>  <input name="dbp" type="text

<li><label>4. Triceps skin fold thickness (mm)</label><B>:</B>  <input name="tsft" type="te
<li><label>5. 2-Hour serum insulin (mu U/ml)</label><B>:</B>  <input name="si" type="text"
<li><label>6. Body mass index</label><B>:</B>  <input name="bmi" type="text" value="43.8" />

<li><label>7. Diabetes pedigree function</label><B>:</B>  <input name="dpf" type="text" valu
<li><label>8. Age</label><B>:</B>  <input name="age" type="text" value="34" />  <font>*
</br></br>
</ul>

       <center><input type="submit" value="Submit Query" class="button" /></center>
</form>
</br>
<center><h2>Tested *Negative for Diabetes</h2></center>
</br></br>
<form id="form1" name="form2" method="post" action="justfy.jsp" >
<center><input type="submit" value="Justify Your Prediction" class="button" /><center>
                </br>
<ul>

<li><b>Plasma glucose concentration is Less than 155.</b></li><li><b>Age of the pantient is Grea
</ul>
</form>
</div>
</div>
</div>

<div id="footer">
<div id="footer-right">
<a href="index.jsp">Home</a> |
```

```
<a href="page.jsp">Neuro-Lab</a> |
<a href="about.jsp">About</a> |
<a href="contact.jsp">Contact</a>
</div>

Copyright &copy; 2012 Neural Network Group
</div>
</div>

</body>
</html>
```

6.9.2 Module for feed-forward NN

```
function net = newff(pr,s,tf,btf,blf,pf)
if nargin < 2
net = newnet('newff');
return
end

% Defaults
if nargin < 4, btf = 'trainlm'; end
if nargin < 5, blf = 'learngdm'; end
if nargin < 6, pf = 'mse'; end

% Error checking
if (~isa(pr,'double')) | ~isreal(pr) | (size(pr,2) ~= 2)
error('Input ranges is not a two column matrix.')
end
if any(pr(:,1) > pr(:,2))
error('Input ranges has values in the second column larger in the values in the same row of the firs
end
if isa(s,'cell')
if (size(s,1) ~= 1)
error('Layer sizes is not a row vector of positive integers.')
end
for i=1:length(s)
si = s{i};
if ~isa(si,'double') | ~isreal(si) | any(size(si) ~= 1) | any(si<1) | any(round(si) ~= si)
error('Layer sizes is not a row vector of positive integers.')
```

```
end
end
s = cell2mat(s);
end
if (~isa(s,'double')) | ~isreal(s) | (size(s,1) ~= 1) | any(s<1) | any(round(s) ~= s)
error('Layer sizes is not a row vector of positive integers.')
end

% More defaults
Nl = length(s);
if nargin < 3, tf = {'tansig'}; tf = [tf(ones(1,Nl))]; end

% Architecture
net = network(1,Nl);
net.biasConnect = ones(Nl,1);
net.inputConnect(1,1) = 1;
[j,i] = meshgrid(1:Nl,1:Nl);
net.layerConnect = (j == (i-1));
net.outputConnect(Nl) = 1;
net.targetConnect(Nl) = 1;

% Simulation
net.inputs{1}.range = pr;
for i=1:Nl
net.layers{i}.size = s(i);
net.layers{i}.transferFcn = tf{i};
end

% Performance
net.performFcn = pf;

% Adaption
net.adaptfcn = 'trains';
net.inputWeights{1,1}.learnFcn = blf;
for i=1:Nl
net.biases{i}.learnFcn = blf;
net.layerWeights{i,:}.learnFcn = blf;
end
```

```
% Training
net.trainfcn = btf;

% Initialization
net.initFcn = 'initlay';
for i=1:Nl
net.layers{i}.initFcn = 'initnw';
end
net = init(net);
```

6.9.3 NN training module

```
function [net,tr,Y,E,Pf,Af]=train(net,P,T,Pi,Ai,VV,TV)
if nargin < 2
error('Not enough input arguments.');
end
if ~isa(net,'network')
error('First argument is not a network.');
end
if net.hint.zeroDelay
error('Network contains a zero-delay loop.')
end
switch nargin
case 2
[err,P,T,Pi,Ai,Q,TS,matrixForm] = trainargs(net,P);
VV = [];
TV = [];
case 3
[err,P,T,Pi,Ai,Q,TS,matrixForm] = trainargs(net,P,T);
VV = [];
TV = [];
case 4
[err,P,T,Pi,Ai,Q,TS,matrixForm] = trainargs(net,P,T,Pi);
VV = [];
TV = [];
case 5
[err,P,T,Pi,Ai,Q,TS,matrixForm] = trainargs(net,P,T,Pi,Ai);
VV = [];
TV = [];
```

```
case 6
[err,P,T,Pi,Ai,Q,TS,matrixForm] = trainargs(net,P,T,Pi,Ai);
if isempty(VV)
VV = [];
else
if ~hasfield(VV,'P'), error('VV.P must be defined or VV must be [].'), end
if ~hasfield(VV,'T'), VV.T = []; end
if ~hasfield(VV,'Pi'), VV.Pi = []; end
if ~hasfield(VV,'Ai'), VV.Ai = []; end
[err,VV.P,VV.T,VV.Pi,VV.Ai,VV.Q,VV.TS,matrixForm] = ...
trainargs(net,VV.P,VV.T,VV.Pi,VV.Ai);
if length(err)
disp('??? Error with validation vectors using ==> train')
error(err)
end
end
TV = [];
case 7
[err,P,T,Pi,Ai,Q,TS,matrixForm] = trainargs(net,P,T,Pi,Ai);
if isempty(VV)
VV = [];
else
if ~hasfield(VV,'P'), error('VV.P must be defined or VV must be [].'), end
if ~hasfield(VV,'T'), VV.T = []; end
if ~hasfield(VV,'Pi'), VV.Pi = []; end
if ~hasfield(VV,'Ai'), VV.Ai = []; end
[err,VV.P,VV.T,VV.Pi,VV.Ai,VV.Q,VV.TS,matrixForm] = ...
trainargs(net,VV.P,VV.T,VV.Pi,VV.Ai);
if length(err)
disp('??? Error with validation vectors using ==> train')
error(err)
end
end
if isempty(TV)
TV = [];
else
if ~hasfield(TV,'P'), error('TV.P must be defined or TV must be [].'), end
if ~hasfield(TV,'T'), TV.T = []; end
if ~hasfield(TV,'Pi'), TV.Pi = []; end
```

```
if ~hasfield(TV,'Ai'), TV.Ai = []; end
[err,TV.P,TV.T,TV.Pi,TV.Ai,TV.Q,TV.TS,matrixForm] = ...
trainargs(net,TV.P,TV.T,TV.Pi,TV.Ai);
if length(err)
disp('??? Error with test vectors using ==> train')
error(err)
end
end
end
if length(err), error(err), end

% TRAIN NETWORK
% -------------

% Training function
trainFcn = net.trainFcn;
if ~length(trainFcn)
error('Network "trainFcn" is undefined.')
end

% Delayed inputs, layer targets
Pc = [Pi P];
Pd = calcpd(net,TS,Q,Pc);
Tl = expandrows(T,net.hint.targetInd,net.numLayers);

% Validation and Test vectors
doValidation = ~isempty(VV);
doTest = ~isempty(TV);
if (doValidation)
VV.Pd = calcpd(net,VV.TS,VV.Q,[VV.Pi VV.P]);
VV.Tl = expandrows(VV.T,net.hint.targetInd,net.numLayers);
end
if (doTest)
TV.Pd = calcpd(net,TV.TS,TV.Q,[TV.Pi TV.P]);
TV.Tl = expandrows(TV.T,net.hint.targetInd,net.numLayers);
end

% Train network
net = struct(net);
```

```
flatten_time = (net.numLayerDelays == 0) & (TS > 1) & (~strcmp(trainFcn,'trains'));
if flatten_time
Pd = seq2con(Pd);
Tl = seq2con(Tl);
if (doValidation)
VV.Pd = seq2con(VV.Pd);
VV.Tl = seq2con(VV.Tl);
VV.Q = VV.Q*VV.TS;
VV.TS = 1;
end
if (doTest)
TV.Pd = seq2con(TV.Pd);
TV.Tl = seq2con(TV.Tl);
TV.Q = TV.Q*TV.TS;
TV.TS = 1;
end
[net,tr,Ac,El] = feval(trainFcn,net,Pd,Tl,Ai,Q*TS,1,VV,TV);
Ac = con2seq(Ac,TS);
El = con2seq(El,TS);
else
[net,tr,Ac,El] = feval(trainFcn,net,Pd,Tl,Ai,Q,TS,VV,TV);
end
net = class(net,'network');

% Network outputs, errors, final inputs
Y = Ac(net.hint.outputInd,net.numLayerDelays+[1:TS]);
E = El(net.hint.targetInd,:);
Pf = Pc(:,TS+[1:net.numInputDelays]);
Af = Ac(:,TS+[1:net.numLayerDelays]);

% FORMAT OUTPUT ARGUMENTS
% ----------------------

if (matrixForm)
Y = cell2mat(Y);
E = cell2mat(E);
Pf = cell2mat(Pf);
Af = cell2mat(Af);
end
```

```
% CLEAN UP TEMPORARY FILES
% ------------------------
nntempdir=fullfile(tempdir,'matlab_nnet');
if exist(nntempdir)
rmpath(nntempdir);
tf=dir(nntempdir);
for i=1:length(tf)
if (~tf(i).isdir)
delete(fullfile(nntempdir,tf(i).name));
end
end
rmdir(nntempdir)
end

% ============================================================
function [s2] = expandrows(s,ind,rows)

s2 = cell(rows,size(s,2));
s2(ind,:) = s;

% ============================================================
function [err,P,T,Pi,Ai,Q,TS,matrixForm] = trainargs(net,P,T,Pi,Ai);

% Check signals: all matrices or all cell arrays
% Change empty matrices/arrays to proper form
switch class(P)
case 'cell', matrixForm = 0; name = 'cell array'; default = {};
case {'double','logical'}, matrixForm = 1; name = 'matrix'; default = [];
otherwise, err = 'P must be a matrix or cell array.'; return
end
if (nargin < 3)
T = default;
elseif ((isa(T,'double')|isa(T,'logical')) ~= matrixForm)
if isempty(T)
T = default;
else
err = ['P is a ' name ', so T must be a ' name ' too.'];
return
```

```
end
end
if (nargin < 4)
Pi = default;
elseif ((isa(Pi,'double')|isa(Pi,'logical')) ~= matrixForm)
if isempty(Pi)
Pi = default;
else
err = ['P is a ' name ', so Pi must be a ' name ' too.'];
return
end
end
if (nargin < 5)
Ai = default;
elseif ((isa(Ai,'double')|isa(Ai,'logical')) ~= matrixForm)
if isempty(Ai)
Ai = default;
else
err = ['P is a ' name ', so Ai must be a ' name ' too.'];
return
end
end

% Check Matrices, Matrices -> Cell Arrays
if (matrixForm)
[R,Q] = size(P);
TS = 1;
[err,P] = formatp(net,P,Q); if length(err), return, end
[err,T] = formatt(net,T,Q,TS); if length(err), return, end
[err,Pi] = formatpi(net,Pi,Q); if length(err), return, end
[err,Ai] = formatai(net,Ai,Q); if length(err), return, end

% Check Cell Arrays
else
d  [R,TS] = size(P);
  [R1,Q] = size(P{1,1});
  [err] = checkp(net,P,Q,TS); if length(err), return, end
  [err,T] = checkt(net,T,Q,TS); if length(err), return, end
  [err,Pi] = checkpi(net,Pi,Q); if length(err), return, end
```

```
  [err,Ai] = checkai(net,Ai,Q); if length(err), return, end
end
```

7 TESTING

7.1 NN Result with 12 hidden nodes

Training+validation	Correctly Classified	Wrong Classified	%Accuracy
316	283	33	89.55%
Testing Samples	Correctly Classified	Wrong Classified	%Accuracy
216	162	54	75%

Table 4: Fact table with 12 hidden nodes

7.2 NN Result with 20 hidden nodes

Training+validation	Correctly Classified	Wrong Classified	%Accuracy
366	357	9	97.54%
Testing Samples	Correctly Classified	Wrong Classified	%Accuracy
166	121	45	72.89%

Table 5: Fact table with 20 hidden nodes

7.3 Rules Table

Rule No.	No. of samples	Correctly classified	Wrong Classified	%Support	%Error
R1	62	62	0	17.32	0
R2	2	2	0	0.56	0
R3	45	42	3	12.57	6.67
R4	4	3	1	1.12	25
R5	97	79	18	27.09	18.56
R6	52	30	22	14.52	42.31
R7	5	5	0	1.4	0
R8	31	25	6	8.66	19.35
R9	8	8	0	2.23	0
R10	52	45	7	14.52	13.45
	358	301	57		

Table 6: Rules Table

7.4 Rules generated using Discrete Attributes

```
Rule R1: If D2<=154 and D2<=100 and D5<=25, then Predict Class 0.
Rule R2: If D2<=154 and D2<=100 and D5>25 and D1<=10 and D3<=52, then Predict  Class1.
Rule R3: If D2<=154 and D2<=100 and D5>25 and D1<=10 and D3>52, then Predict Class0.
Rule R4: If D2<=154 and D2<=100 and D5>25 and D1>10, then Predict Class 1.
Rule R5: If D2<=154 and D2>100 and D5<=28, then Predict Class0.
Rule R6: If D2<=154 and D2>100 and D5>28 and D5<=58 and D3<=76, then Predict Class0.
Rule R7: If D2<=154 and D2>100 and D5>28 and D5<=58 and D3>76 and D4<=21, then Predict Class0.
Rule R8: If D2<=154 and D2>100 and D5>28 and D5<=58 and D3>76 and D4>21, then Predict Class1.
Rule R9: If D2<=154 and D2>100  and D5>58, then Predict class0
Rule 10: If D2>154 then, Predict Class 1.
```

8 APPLICATION

8.1 Business Applications

The 1988 DARPA Neural Network Study [DARP88] lists various neural network applications, beginning in about 1984 with the adaptive channel equalizer. This device, which is an outstanding commercial success, is a single- neuron network used in long-distance telephone systems to stabilize voice signals. The DARPA report goes on to list other commercial applications, including a small word recognizer, a process monitor, a sonar classifier, and a risk analysis system. Neural networks have been applied in many other fields since the DARPA report was written. A list of some applications mentioned in the literature follows:

8.2 Credit Card Activity Checking

Neural networks are used to spot unusual credit card activity that might possibly be associated with loss of a credit card.

8.3 Financial

Real estate appraisal, loan advisor, mortgage screening, corporate bond rating, credit-line use analysis, portfolio trading program, corporate financial analysis, currency price prediction.

8.4 Medical and Banking

Breast cancer cell analysis, EEG and ECG analysis, prosthesis design, optimization of transplant times, hospital expense reduction, hospital quality improvement, emergency-room test advisement How can I tell which transactions are likely to be fraudulent.?

8.5 FUTURE ENHANCEMENTS

- Rule optimization using Genetic Algorithm.
- Extending RE-RX to multiple group classification problem.
- Base can be changed from C4.5 to C5.0.
- Replacing hyper plane by the technique to handle inseparability.

9 CONCLUSTION

We have developed a new NN rule extraction algorithm Re-RX that is capable of extracting classification rules from NNs trained using both discrete and continuous attributes.The novel characteristic of the algorithm lies in its hierarchical nature of considering discrete variables before continuous variables,in a recursive way. We have illustrated the working of the algorithm using three real-world data sets. More specifically,we have considered application scoring where classification models are built to distinguish good payers from defaulters based on the characteristics provided at the time of application.Two key requirements of application scoring models are performance and interpretability. Especially, the latter is becoming more and more important since financial regulators and law enforcement bodies require all risk management models of a financial institution to be validated. Using real-life application scoring data sets, we have demonstrated that Re-RX is able to extract accurate and interpretable classification rules.There is an obvious limitation to the applicability of the proposed algorithm, however. When the class labels of the data are best described as a nonlinear function of the continuous inputs,the extracted rules will not be able to preserve the accuracy of the NN. This is a consequence of the property of Re-RX which generates only linear classifiers involving the continuous attributes in order to improve the comprehensibility of the overall rule set. It is possible to preserve the accuracy, for example, by having an NN or other nonlinear classifiers, in place of the linear hyperplane. However, such rules would not be as comprehensible as Re-RX generated rules. The issue of accuracy versus complexity and comprehensibility of the rules generated from trained NNs certainly deserves further research.

A Recursive Neural Network Rule Extraction for Data With Mixed Attributes [Re-Rx] Algorithm

---

Algorithm *Re-Rx(S,D,C)*

Input : A set of data samples S having discrete attributes D and continuous attributes C
Output: A set of classification rules.

1. Train and prune an NN using the data set S and all its attributes D and C.
2. Let $\acute{D}$ and $\acute{C}$ be the sets of discrete and continuous attributes still present in the network, respectively. Let $\acute{S}$ be the set of data samples that are correctly classified by the pruned network.
3. If $\acute{D} = \emptyset$, then generate a hyperplane to split the samples in $\acute{S}$ according to the values of their continuous attributes $\acute{C}$ and stop.
   Otherwise, using only the discrete attributes $\acute{D}$, generate the set of classification rules R for the data set $\acute{S}$.
4. For each rule R_i generated: If support $R_i > \delta_1$ and error $R_i > \delta_2$, then
   - Let S_i be the set of data samples that satisfy the condition of rule R_i and D_i be the set of discrete attributes that do not appear in rule condition of R_i .
   - If $\acute{D} = \emptyset$, then generate a hyperplane to split the samples in S_i according to the values of their continuous attributes C_i and stop.
   - Otherwise, call *Re-Rx($S_i D_i$,C_i)*

References

[1] Extracting rules for classification problems: AIS based approach By Humar Kahramanli , Novruz Allahverdi [Science direct : Expert Systems with Applications xxx (2009) article in press]

[2] Recursive Neural Network Rule Extraction for Data with Mixed attributes By Rudy Setiono,Senior Member, IEEE, Bart Baesens, and Christophe Mues [IEEE Transaction on Neural Networks, Vol. 19 No. 2, February 2008 pp. 299-307]

[3] Data Mining:Concepts and techniques By Han J. , Kamber, M. Y. and Lee S. C.(2001).. San Francisco, CA, USA: Morgan Kaufmann.

[4] Using neural networks and data mining techniques for the financial distress prediction model. By Wei-Sen Chen , Yin-Kuan Du [Science direct :Expert Systems with Applications 36 (2009) 40754086]

[5] Data classification with Multilayer perceptrons using a generalized error function By Lus M. Silva a, J. Marques de S a,b, Lus A. Alexandre [Science direct : Neural Networks 21 (2008) 1302-1310]

[6] Design of a hybrid system for the diabetes and heart diseases By Humar Kahramanli , Novruz Allahverdi [Science direct : Expert Systems with Applications 35 (2008) 82-89.

[7] Mining Classification rules for Liver Disorders By Humar Kahramanli , Novruz Allahverdi International Journal of Mathematics and Computers in Simulatin Issue1, Volume 3, 2009.

[8] Evaluation of neural networks and data mining methods on a credit assessment task for class imbalance problem. By Yueh-Min Huanga, Chun-Min Hunga, Hewijin Christine Jiaub [Science direct: Nonlinear Analysis: RealWorld Applications 7 (2006) 720 747]

[9] Humar Kahramanli , Novruz Allahverdi ,A new method for composing classification rules: AR+OPTBP, 5th International Advanced Technologies Symposium (IATS09), May 13-15, 2009, Karabuk, Turkey.

[10] Evolution of neural networks for classification and regression By Miguel Rocha, Paulo Cortez, Jose Neves [Science direct : Neurocomputing 70 (2007) 28092816]

[11] Generating rules with predicates, terms and variables from the pruned neural networks By Richi Nayak [Science direct : Neural Networks-Accepted in 6 Feb, 2009 Article in press]

[12] Gallant, S. I. (1988). Connection expert systems. Communications of the ACM, 31(2), 152169.

[13] R. Setiono and H. Liu, Symbolic representation of neural networks, IEEE Computer, vol. 29, no. 3, pp. 7177, Mar. 1996.

[14] R. Setiono, Extracting rules from neural networks by pruning and hidden-unit splitting, Neural Comput., vol. 9, no. 1, pp. 205225, 1997.

[15] A. Gupta, S. Park, and S. M. Lam, Generalized analytic rule extraction for feedforward neural networks, IEEE Trans. Knowl. Data Eng., vol. 11, no. 6, pp. 985991, Nov./Dec. 1999.

[16] T. A. Etchells and J. P. G. Lisboa, Orthogonal search-based rule extraction (OSRE) for trained neural-networks: A practical and efficient approach, IEEE Trans. Neural Netw., vol. 17, no. 2, pp. 374384, Mar. 2006.

[17] M. Craven and J. Shavlik, Extracting tree-structured representations of trained networks, in Advances in Neural Information Processing Systems (NIPS). Cambridge, MA:MIT Press, 1996, vol. 8, pp. 2430.

[18] D. Nauck and R. Kruse, Neuro-fuzzy methods in fuzzy rule generation, in Fuzzy Sets in Approximate Reasoning and Information Systems. Norwell, MA: Kluwer , 1999, pp. 305334.

[19] J. M. Benitez, J. L. Castro, and I. Requena, Are neural network black boxes?, IEEE Trans. Neural Netw., vol. 8, no. 5, pp. 11561164, Sep. 1997.

[20] H. Tsukimoto, Extracting rules from trained neural networks, IEEE Trans. Neural Netw., vol. 11, no. 2, pp. 377389, Mar. 2000.

[21] I. A. Taha and J. Ghosh, Symbolic interpretation of artificial neural networks, IEEE Trans. Knowl. Data Eng., vol. 11, no. 3, pp. 448463, May/Jun. 1999.

[22] J. R. Rabunal, J. Dorado, A. Pazos, J. Periera, and D. Rivero, A new approach to the extraction of ANN rules and to their generalization capacity through GP, Neural Comput., vol. 16, no. 7, pp. 14831523, 2004.