# Mining Classification Rules from Database by Using Artificial Neural Network

Mrs. S. V. Shinde, Dr. U.V. Kulkarni

*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 Network (ANN). Although ANN usually reaches high classification accuracy, the obtained results sometimes may be incomprehensible. Due to this fact various methods have been proposed to extract the rules from ANN which justifies the classification results given by ANN. The Proposed research work presents the overview of the various methods used to extract the rules from ANN and their comparisons.**

***Keywords:*** **ANN, Classification, Data Mining, Rule Extractio*n*.**

## I. INTRODUCTION

**D**ata mining (DM), also known as ‘‘knowledge discovery in databases” (KDD), is the process of discovering meaningful patterns in huge databases [11]. 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. It is interactive and iterative, involving the following steps [4]:

**Step 1.** Application domain identification: Investigate and understand the application domain and the relevant prior knowledge.
**Step 2.** Target dataset selection: Select a suitable dataset, or focus on a subset of variables or data samples where data relevant to the analysis task are retrieved from the database.
**Step 3.** Data preprocessing: the DM basic operations include ‘data cleaning’ and ‘data reduction’.
**Step 4**. Data mining: This is an essential process, where AI methods are applied in order to search for meaningful or desired patterns in a particular representational form, such as association rule mining, classification trees, and clustering techniques.
**Step 5.** Knowledge Extraction: Based on the above steps it is possible to visualize the extracted patterns or visualize the data depending on the extraction models.
**Step 6**. Knowledge Application: Here, we apply the found knowledge directly into the current application domain or in other fields for further action.
**Step 7**. Knowledge Evaluation: Here, we identify the most interesting patterns representing knowledge based data on some measure of interest.

The more common model functions in the current data mining process include the following [5].
**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 or Probability density models.
**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.

In a classification or prediction problem, neural network techniques are used as a tool to analyze datasets. A multilayer feed-forward network is an important class of neural networks [7].

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 provide analytical solution to the problem. The most popular forms of learning are [4]:
**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. 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 back propagation network (BPN) is a neural network that uses a supervised learning method and feed-forward architecture [4].

## II. LITERATURE SURVEY

Many rule extraction algorithms have been designed to generate classification rules from NNs that have been trained to distinguish data samples from different classes [1]. One of the first rule extraction techniques from neural networks was proposed by Gallant [2]. He was working on connectionist expert systems. In this work, each ANN node represents a conceptual entity. Towell and Shavlik showed how to use ANNs for rule refinement [3]. The algorithm was called SUBSET, which is based on the analysis of the weights that make a specific neuron active. Alexander and Mozer developed a rule extraction method, based on connection weights, that supposes activation functions showing approximately Boolean behavior.

Many algorithms assume that the input data attributes are discrete in order to make the rule extraction process more manageable. NeuroRule [7] 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 network's output in terms of the discretized hidden unit activations values and rules that explain that discretized activation values in terms o the discretized attributes of the input data.

Rule eXtraction (RX) [6] 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 created and trained with the discretized activation values as the target outputs.

Trepan, an algorithm that was developed by Craven and Shalvic [8] 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 spilt 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.

## A. Recursive Rule Extraction: The RE-RX Algorithm :

In the recursive algorithm for rule extraction (RE-RX) [1] from an ANN that has been trained for solving a classification problem having mixed discrete and continuous input data attributes. This algorithm shares some similarities with other existing rule extraction algorithms. It assumes the trained network has been pruned so that irrelevant and redundant network connections and units have been removed. This also makes the use of the decision tree method C4.5 to generate rules with only discrete attributes in their conditions. The novel feature of the recursive algorithm is in the rule set generated. The rules are hierarchical such that only those rules 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 . Such a rule conditions would greatly increase the comprehensibility of the rules and hence greatly pave the way to open the NN "black box" wider.

Two group classification problems can be handled by using the algorithm as follows:

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

**Input:** A set of data samples $S$ having the 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 $D'$ and $C'$ be the sets of discrete and continuous attributes still present in the network, respectively. Let $S'$ be the set of data samples that are correctly classified by the pruned network.
3) If $D' = Ø$, then generates a hyperplane to split the samples in $S'$ according to the values of their continuous attributes $C'$ and stop.
Otherwise, using only the discrete attributes $D'$, generate the set of classification rules $R$ for the data set $S'$.
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 $D' = Ø$, 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 $(S_i, D_i, C_i)$.
```

## B. Rule Extraction From ANN Trained With Adaptive Activation Function:

Humar Kahramanli and Novruz Allahverdi have presented the technique of mining the classification rules for Liver Disorders using adaptive activation function [9]. 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:

$$Ø(x) = A1\, e^{-x2} + \frac{A2}{1+e^{-B*x}} \quad .$$

Where, $A1$, $A2$ and $B$ are real variables which will be adjusted during training.

The algorithm used for training the neural network is as follows:

Algorithm works as follows:

1. Apply the input vector,
   x = $(x_1, x_2, \ldots\ldots\ldots x_N)$ to the input units.
2. Calculate the sum of weighted input signals to the hidden layer.
3. Calculate the outputs from the hidden layer.
4. Calculate the sum of weighted input signals to the outputs layer.
5. Calculate the outputs.
6. Calculate the error terms for the output units.
7. Calculate the error terms for the hidden units.
8. Update weights on the output and hidden layer.
9. Update real variables on the output and hidden layer.

This algorithm is not far from the back propagation algorithm. The difference is being used an updated real variables in this algorithm. The training data are presented until the energy function is acceptably low and the network converges.

The performance metrics used in this algorithm are Accuracy, sensitivity and specificity. 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 :

$$Accuracy = \frac{\text{Total number of correctly diagnosed cases}}{\text{Total number of cases}}$$

$$Sensitivity = \frac{\text{Total number of positive cases correctly diagnosed}}{\text{Total number of positive cases}}$$

$$Specificity = \frac{\text{Total number of negative cases correctly diagnosed}}{\text{Total number of negative cases}}$$

In summary, for rule extraction the first NN 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 %96 and %93 correctly diagnosis for Class 0 and Class 1 respectively. In summary the system correctly diagnosed %94.8 of whole samples.

## C. Rule Extraction from ANN Using Opt-aiNet:

In paper [10] 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. It's been observed that this method increased classification accuracy despite decreasing number of rules. This method consists of three-stages:
1- Minig assosiation rules and eliminating;
2-Classification of data;
3- Rule extraction.
1. Mining association rules and eliminating

In the first stage, the association rules which were discovered for classes and data that provide these rules have been eliminated. This provides the training time to become a little shorter. The Apriori algorithm has been used for mining association rules. Data elimination, which provide association rules, has been inspired by the work of Karabatak and Ince [13]. The problem of mining association rules from database of transactions was introduced by Agrawal et al. [12]. Let $A$ be a set of items, $X \subset A$, $T$ is a database of transactions and $n$ is the number of transactions. The support of an itemset $X$, is defined as follows:

$$sup(X) = \frac{freq(x)}{n}$$

where $freq(X)$ is the number of transactions in which $X$ occurs as a subset. A rule is an implication of the form $X \rightarrow Y$, where $X, Y \subset A$ and $X \cap Y = \emptyset$ are called as antecedent and consequent of the rule respectively. The support of the rule is expressed as follows:

$$sup(X \rightarrow Y) = \frac{freq(X \cup Y)}{n}.$$

The confidence of the rule is defined as follows:

$$conf(X \rightarrow Y) = \frac{sup(X \rightarrow Y)}{\sup(x)}.$$

The Apriori algorithm has been used to generate association rules. The Apriori algorithm works iteratively. It first finds the set of large 1-item sets, and then set of 2-itemsets, and so on. The number of scan over the transaction database is as many as the length of the maximal item set [13]. The algorithm works is as follows:
The algorithm finds the frequent sets $L$ in database $D$.
• Find frequent set $L_{k-1}$.
• Join Step.
$C_k$ is generated by joining $L_{k-1}$ with itself
• Prune Step.
Any $(k-1)$ -itemset that is not frequent cannot be a subset of a frequent $k$ -itemset, hence should be removed.
where ($C_k$: Candidate itemset of size $k$) And the Apriori pseudocode used is given as:

```
Apriori(T,ϵ)
L_1←{large 1-itemsets that appear in ϵ transactions}
k←2
while L_{k-1}≠Ø
C_k ←Generate (L_{k-1})
For transactions t ϵ T
C_t←Subset(C_k,t)
For candidates c ϵ C_t
count[c]=count[c]+1
L_k←{c ϵC_kIcount[c]≥ ϵ}
k←k+1
return U_k L_k.
```

2. In the second stage, neural network has been trained by using the backpropagation algorithm.
In the third stage, Opt-aiNET has been executed for extraction rules from this ANN:
The opt-aiNet algorithm used for rule extraction is as follows:
1. Initialization: create an initial random population of network antibodies;
2. Local search: while stopping criterion is not met, do:
• Clonal expansion: for each network antibody, determine its fitness and normalize the vector of fitnesses. Generate a clone for each antibody, i.e., a set of antibodies which are the exact copies of their antibody;
•Affinity maturation: mutate each clone inversely proportionally to the fitness of its parent antibody that is kept unmutated. For each mutated clone, select the antibody

with highest fitness, and calculate the average fitness of the selected antibodies;

• Local convergence: if the average fitness of the population does not vary significantly from one iteration to the other, go to the next step; else, return to Step 2;

3. Network interactions: determine the affinity (similarity) between each pair of network antibodies;

4. Network suppression: eliminate all network antibodies whose affinity is less than a pre-specified threshold, and determine the number of remaining antibodies in the network; these are named memory antibodies;

5. Diversity: introduce a number of new randomly generated antibodies into the network and return to step 2.

D. Use of Genetic Algorithm in Rule Extraction from ANN [11]:

The starting point of any rule-extraction system is firstly to train the network on the data till a satisfactory error level is reached. For classification problems, each input unit typically corresponds to a single feature in the real world, and each output unit to a class value or class. The first objective of our approach is to encode the network in such a way that a genetic algorithm can be run over the top of it. This is achieved by creating an n-dimensional weight space where n is the number of layers of weights. For example, Figure 1 depicts a simple neural network with five input units, three hidden units, and one output unit, with each node enumerated in this case except the output. From this encoding, genes can be created which, in turn, are used to construct chromosomes where there is at least one gene representing a node at the input layer and at least one gene representing a node at the hidden layer. A typical chromosome for the network depicted in Figure 1 could look something like this (Figure 2):

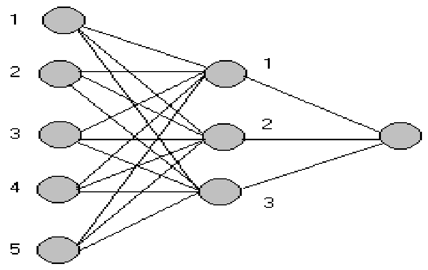

Fig. 1 - A typical encoding of a simple neural network with only one class value.

From this encoding, genes can be created which, in turn, are used to construct chromosomes where there is at least one gene representing a node at the input layer and at least one gene representing a node at the hidden layer. A typical chromosome for the network depicted in Figure 1 could look something like this (Figure 2):

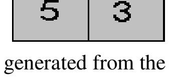

Fig. 2 - A typical chromosome generated from the encoded network for only one class value.

This chromosome corresponds to the fifth unit in the input layer and the third unit in the hidden layer. That is, the first gene contains the weight connecting input node 5 to hidden unit 3, and the second gene contains the weight connecting hidden unit 3 to the output class. Fitness is computed as a direct function of the weights which the chromosome represents. For chromosomes containing just two genes (one for the input unit, the other for the hidden unit), the fitness function is: **Fitness= Weight(Input→Hidden)*Weight(Hidden→Output)** where '→' signifies the weight between the two enumerated nodes.

So the fitness of the chromosome in Figure 2 is: ***Fitness = Weight(5→3)*Weight(3→Output)***

This fitness is computed for an initial set of random chromosomes, and the population is sorted according to fitness. An elitist strategy is then used whereby a subset of the top chromosomes is selected for inclusion in the next generation. Crossover and mutation are then performed on these chromosomes to create the rest of the next population. The chromosome is then easily converted into IF…THEN rules with an attached weighting. This is achieved by using the template: 'IF <gene1> THEN output is <class> (weighting)', with the weighting being the fitness of the gene and the class signifies which output unit is being switched on. The weighting is a major part of the rule generation procedure because the value of this is a direct measure of how the network interprets the data. Since 'Gene 1' above corresponds to the weight between an input unit and a hidden unit, the template is essentially stating that the consequent of the rule is caused by the activation on that particular input node and its connection to a hidden unit (not specified explicitly in the rule). The rule template above therefore allows the extraction of single-condition rules.

**Experimentation:**

This experiment uses a sunburn dataset(Winston, 1992) to show that our approach can find optimized rule set comparable to those found with purely symbolic methods of data-mining.

Table 1 - The Sunburn Dataset

| Name | Hair | Height | Weight | Lotion | Result |
|---|---|---|---|---|---|
| Sarah | Blonde | Average | Light | No | Sunburned |
| Dana | Blonde | Tall | Average | Yes | Not sunburned |
| Alex | Brown | Short | Average | Yes | Not sunburned |
| Annie | Blonde | Short | Average | No | Sunburned |
| Emily | Red | Average | Heavy | No | Sunburned |
| Pete | Brown | Tall | Heavy | No | Not sunburned |
| John | Brown | Average | Average | No | Not sunburned |
| Katie | Blonde | Short | Light | Yes | Not sunburned |

This dataset is converted as follows into a form suitable for input to the ANN:

Table 2 - Neural Network Conversion of Data in Table 1

| | | |
|---|---|---|
| **Hair** | Blonde | 100 |
| | Brown | 010 |
| | Red | 001 |
| **Height** | Short | 100 |
| | Average | 010 |
| | Tall | 001 |
| **Weight** | Light | 100 |
| | Average | 010 |
| | Heavy | 001 |
| **Lotion** | No | 10 |
| | Yes | 01 |
| **Class** | Sunburned | 10 |
| | Not Sunburned | 01 |

One example of input is therefore**: 10001010010**, which represents a blonde haired (100), average height (010), light (100), no-lotion used (10) individual (i.e. Sarah). Note that we are dealing with a supervised learning network, where the class in which the sample falls is explicitly represented for training purposes. So, in the case of Sarah, the output 10 (sunburned) is used for supervised training. '10' here signifies that the first output node is switched on and the second is not. A neural network with 11 input, 5 hidden and 2 output units was created. The input to the network was a string of 0's and 1's which corresponded to the records in the data set above. The network was then trained (using back-propagation) until a mean square error of 0.001 was achieved. The network weights were then recorded and the genetic algorithm process started. The weights between the 11 input and 5 hidden units are as follows:

**Hidden Unit 1** (all eleven input units):
-2.029721 1.632389 -1.702274 -1.369853 0.133539 0.296253 -0.465295 0.680639 -0.610233 -1.432447 -1.462687

**Hidden Unit 2:**
0.960469 1.304169 -0.558034 -0.870080 0.394558 0.537783 0.047991 0.575487 -1.571345 0.476647 -0.0034666

**Hidden Unit 3:**
0.952550 -2.791922 1.133562 0.518217 1.647397 -1.801673 -1.518900 -0.245973 0.450328 -0.169588 -1.979129

**Hidden Unit 4:**
-1.720175 1.247111 1.095436 0.365523 0.350067 0.584151 0.773993 1.216627 -1.174810 -1.624518 2.342727

**Hidden Unit 5:**
-1.217552 2.288170 -1.088214 -0.389681 -0.919714 1.168223 0.579115 1.039906 1.499586 -2.902985 2.754642

The weights between the five hidden units and the two output units are as follows:

**Output Unit 1** (all 5 hidden units):
-2.299536 -0.933331 2.137592 -2.556154 -4.569341

**Output Unit 2:**
2.235369 -0.597022 -3.967368 1.887921 3.682286

A random number generator was used to create the initial population of five chromosomes for the detection of rules, where an extra gene is added to the end of the chromosome to represent one of the two output class values. The alleles for this gene are either 1 or 2 (to represent the output node values of 10 (sunburned) and 01 (not sunburned).

The following decisions were taken:

1. The fittest chromosome of each generation goes through to the next generation.
2. The next chromosome is chosen at random, but a greater fitness gives a greater chance of being chosen. Negative fitnesses were not included. (A 'roulette wheel' selection.).
3. The remaining four chromosomes are created as a mutation of the two chosen above and crossover on these same two. Duplicate chromosomes are removed.
4. Fitness was computed simply as *Weight(input_to_hidden)*Weight(hidden_to_output).*

The more positive the number, the greater the fitness.

An example run (first three generations only) for extracting rules dealing with the first output node only (i.e. for sunburn cases only) is given in Figure 5.

**Results:**

A traditional symbolic learning algorithm running on this dataset will find the following four rules:

(a) If person has red hair then person is sunburned;
(b) If person is brown haired then person is not sunburned;
(c) If person has blonde hair and no lotion used then person is sunburned; and
(d) If person has blonde hair and lotion used then person is not sunburned.

Our approach identified the following five single condition rules in ten generations, with a maximum population of 6 in each generation:

(i) 'IF unit1 is 1 THEN output is 1 (fitness 4.667)', which corresponds to: 'IF hair
colour=blonde THEN result is sunburned'. The fitness here is calculated as follows:
input unit 1 to hidden unit 1 weight of -2.029721* hidden unit 1 to output unit 1 weight of -2.299536.

(ii) 'IF unit 3 is 1 THEN output is 1 (fitness 3.908)', which corresponds to `IF hair colour=red THEN result is sunburned' (input unit 3 to hidden unit 1 weight of -1.702274 * hidden unit 1 to output unit 1 weight of -2.299536).

(iii) 'IF unit 10 is 1 then output is 1 (fitness 4.154), which corresponds to 'IF no lotion used THEN result is sunburned' (input unit 10 to hidden unit 4 weight of -1.624518 *hidden unit 4 to output weight of -2.556154).

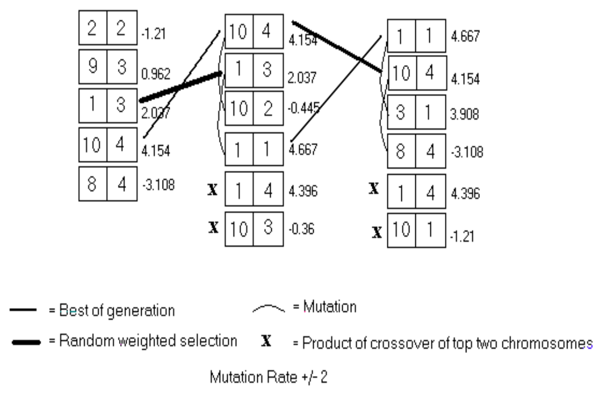
= Best of generation
= Mutation
= Random weighted selection
X = Product of crossover of top two chromosomes
Mutation Rate +/- 2

Fig.3. First three generations of chromosome evolution in the extraction of rules.

(iv) 'IF unit 2 is 1 THEN output is 2 (fitness 8.43)', which corresponds to: 'IF hair colour=brown THEN result is not sunburned' (input unit 2 to hidden unit 5 weighting of 2.288170 * hidden unit 5 to output unit 2 weighting of 3.682286, with rounding)

(v) 'IF unit 11 is 1 THEN output is 2 (fitness 10.12)', which corresponds to 'IF lotion used THEN result is not sunburned' (input unit 11 to hidden unit 5 weighting of 2.754642 * hidden unit 5 to output unit 2 weighting of 3.682286, with rounding).

Figure 5 shows that, for the sunburnt cases (rules (i) – (iii) above), there is early convergence (within three generations) to these rules. The fitness values cited in the rule set above may not be the maximum attainable but are nevertheless significantly above 0.

## 3. Conclusion

In this paper we reviewed various techniques for rule extraction from ANN. The rule extraction plays a very important role in the applications like Medical science where justification of the obtained results is important. The first technique described i.e. Re-RX algorithm is capable of extracting classification rules from NN trained using both discrete and continuous attributes.In the second techniques first the neural network is trained 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).In Opt-aiNET algorithm for rule extraction it's been observed that this method has increased classification accuracy despite decreasing number of rules. At the end of the paper the method of rule extraction from neural network by using genetic algorithm is experimented on the Sunburn dataset as it generates optimized ruleset.

## REFERENCES

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

[2] S.I. Gallant: Connectionist expert systems, Communications of the ACM, Vo1.31, No.2, pp. 152-169, (1988).

[3] G.G. Towell and J. Shavlik:Extracting refined rules from knowledge based neural networks, Machine learning, 13, pp.71-101, (1993).

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

[5] Humar Kahramanli , Novruz Allahverdi:Extracting rules for classification problems: AIS based approach, In Science direct : Expert Systems with Applications (2009).

[6] R.Setiono:Extracting rules from neural networks by pruning and hidden-nit splitting, Neural Comput., vol9, no. 1, pp. 205-225 (1997).

[7] R. Setiono:Symbolic representation of neural networks, IEEE Computer, vol. 29, no. 3, pp. 71-77, (1996).

[8] M. Craven and J. Shalvic:Extracting tree-structured representations of trained networks, in Advances in Neural Information Processing Systems (1996).

[9] Humar Kahramanli, Novruz Allahverdi:Mining Classification rules for liver disorders, International Journal Of Mathematics and Computers in Simulation, Issue 1, Volume3(2009).

[10] Humar Kahramanli, Novruz Allahverdi,: A new method for composing classification rules: AR+OPTBP ,5th International Advanced Technologies Symposium (IATS'09), Karabuk, (2009).

[11] Ajit Narayanan, Edward Keedwell and Dragan Savic:Data mining neural networks with genetic algorithms, School of Engineering and Computer Science [citeseer: paper 0.1.1.54.994.pdf]

[12] Agrawal, R., Imielinski, T., & Swami: Mining association rules between sets of items in large databases. In Proceedings of the ACM SIGMOD International Conference on Management of Data, 207–216, Washington D.C(2003).

[13] Karabatak, M., & Ince, M.C: An expert system for detection of breast cancer based on association rules and neural network, Expert Systems with Applications, doi:10.1016/j.eswa.2008.02.064 (2008).

[14] Miguel Rocha, Paulo Cortez, Jose Neves: Evolution of neural networks for classification and regression, In Science direct : Neurocomputing 70 2809–2816 (2008).

[15] Richi Nayak:Generating rules with predicates, terms and variables from the pruned neural networks, in Science direct : Neural Networks (2009).

[16] Luís M. Silva a, J. Marques de Sá a,b, Luís A. Alexandre:Data classification with Multilayer perceptrons using a generalized error function, In Science direct: Neural Networks 21 1302_1310 (2008).

[17] John Atkinson-Abutridy, Chris Mellish, and Stuart Aitken:Combining Information Extraction with genetic algorithm for text mining, University of Edinburgh, In IEEE INTELLIGENT SYSTEMS: Published by the IEEE Computer Society pp 22-30 (2004).

[18] Xianjun Ni:Research of Data Mining Based on Neural Networks,In proceeding of world academy of science, engineering and technology Vol 29 ISSN 1307-6884 (2008).

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

[20]

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

**Mrs. S. V. Shinde** has received her B.E. (CSE) degree from SRTM University, Nanded and M.E.(Computer) degree from Bharti Vidyapeeth Pune. She is working as a Assistant Professor in Pimpri Chichwad College of Engineering, Pune.
swaatii.shinde@gmail.com Mobile No: 9822517341