Journal of Engineering Science and Technology
Vol. 3, No. 3 (2008) 243 - 255
© School of Engineering, Taylor's University College

INVESTIGATION OF DATA MINING USING PRUNED ARTIFICIAL NEURAL NETWORK TREE

S.M.A.KALAIARASI*, G.SAINARAYANAN, ALI CHEKIMA, JASON TEO

School of Engineering and Information Technology, University Malaysia Sabah,
Locked Bag No. 2073, 88999 Kota Kinabalu, Sabah, MALAYSIA.
* Corresponding Author: anbakala@ums.edu.my

Abstract

A major drawback associated with the use of artificial neural networks for data mining is their lack of explanation capability. While they can achieve a high predictive accuracy rate, the knowledge captured is not transparent and cannot be verified by domain experts. In this paper, Artificial Neural Network Tree (ANNT), i.e. ANN training preceded by Decision Tree rules extraction method is presented to overcome the comprehensibility problem of ANN. Two pruning techniques are used with the ANNT algorithm; one is to prune the neural network and another to prune the tree. Both of these pruning methods are evaluated to see the effect on ANNT in terms of accuracy, comprehensibility and fidelity.

Keywords: Data mining, Artificial Neural Network, Comprehensibility, Pruning.

1. Introduction

In the last decade, the capabilities for generating and collecting data have grown explosively. The amounts of data stored in computer systems are so huge. This explosive growth creates the necessity of automated knowledge discovery from data. Knowledge discovery or data mining, attempts to identify valid and useful patterns from huge volume of data [1]. The pattern (knowledge) extracted must be in an understandable form because the main purpose of data mining is to aid human in decision making. Thus, comprehensibility is an important factor in data mining that is data mining algorithm must be able to produce understandable patterns.

Artificial neural networks (ANN) have shown to be very powerful pattern recognition techniques for classification in a variety of domains. Their ability to

Nomenclatures

f	Activation function
In	Input of the network
T	Set of cases associated at the node (Decision Tree)
V_{ji}	Weight connecting the ith input unit to the jth hidden unit
v_{oj}	Bias of the jth hidden unit
w_j	Weights connecting the jth hidden unit to the output unit
w_o	Bias of the output unit
y_k	Actual output for kth output
z_j	Net output for hidden neuron
z_{inj}	Net input for hidden neuron

generalize and learn from data imitates the human's ability to learn from experience and often outperform the other models in terms of predictive accuracy. However, most of these applications primarily focus at developing networks with high predictive accuracy without paying any attention to explain how the classifications are being made or the knowledge that is learned. In ANN, the knowledge learned is represented at a subsymbolic level in terms of connections and weights. It is like a black box providing little insight into how decisions are made. They have no explicit, declarative knowledge structure that allows the representation and generation of explanation structures. Thus, knowledge captured by ANN is not transparent to users and cannot be verified by domain experts.

In data mining however, for ANN to gain wider degree of acceptance, it is highly desirable that an explanation capability becomes an integral of the functionality of a trained ANN [2]. By extracting rules, explanation capability is added to ANN, with this data miners will understand what the networks have learned and how the ANN classify or predict. Rule extraction originates from Gallant's work on connectionist expert system [3].

According to the taxonomy presented by Andrews et al. [2], rule extraction algorithms can be roughly categorized by

1. the schemes of extracting rules from ANN,
2. the portability of the rule extraction technique,
3. the expressive power of the rule extracted (type of rule extracted)

The schema of extracting rules can be categorized into decompositional or pedagogical algorithms. The decompositional algorithms extract rules by decomposing the ANN and extracting rules from each unit in ANN and aggregate them. The earliest decompositional rule extraction method is the KT algorithm developed by Fu [4]. KT is derived from the word "Knowledgetron", which was coined by the author to refer a neural network with knowledge. The KT algorithm generates rules for each concept corresponding to a hidden or output unit whose summed weights exceed the threshold of the unit. The same idea is incorporated in the Subset algorithm developed by Towell and Shavlik [5]. This algorithm searches explicitly on incoming weights that exceed the bias of the unit. If exceeded the weight, then these are rewritten as rules. NeuroRule by Setiono and Liu [6], constructed and pruned a multilayer feedforward neural network, built a decision tree from the hidden node activation values of the pruned network, then obtained an oblique decision tree by replacing each split at the nodes of the tree by a linear combination of original input attributes.

Pedagogical techniques treat the network as a 'black box' and make no attempt to disassemble its architecture to examine how it works. Instead this approach extracts rules by examining the relationship between the inputs and outputs. Representatives of this category include Validity Interval Analysis (VIA) [7], TREPAN [8], Decision Tree Extractor (Dectext) [9], Artificial Neural Network Tree (ANN-DT) [10], etc. VIA was designed as a general purpose rule extraction procedure of extracting symbolic knowledge from network. TREPAN, developed by Craven, regards the rule extraction process as an inductive learning problem and uses oracle queries to induce an M-of-N decision tree that approximates the concept represented by a given network. Dectext trains a network and extracts a classical decision tree from the network. Compared to pedagogical approach, the decompositional approach shows the role of each hidden unit and more detail information can be obtained. Schmitz et al. [10] proposed an algorithm named ANN-DT which used neural network to generate outputs for examples interpolated from the training data set and then extracted a univariate binary decision tree from the network.

Portability means the extent to which the underlying ANN incorporates specialized architecture, training regimes or data type that is needed. Most of rule extraction methods required specialized type of ANN architecture or special type of training. Examples of such techniques include BRAINNE [11] and KBANN [5]. Most of the rule extraction algorithms require discrete inputs [4-6, 9, 10] or continuous inputs [7]. Only few algorithms deal with both [8].

The rule extracted from ANN can be expressed in production rule, M-of-N tree, etc. Dextect and NeuoRule extract production rule, where as TREPAN extracts an M-of-N tree. These algorithms use decision tree approach to approximate the function of the trained that is to use DT approach to extract rules.

In this paper, an approach named Artificial Neural Network Tree (ANNT), i.e. ANN training preceded by Decision Tree (DT) rules extraction method is presented for solving the understandability of ANN. ANNT was motivated by the lack of an intelligent procedure for interpreting the knowledge learned by ANN and utilizing it as a data mining tool. ANNT translates the knowledge represented by ANN into DT. In other words, ANNT uses DT to approximate the function of the trained ANN to improve the comprehensibility of ANN. To further improve on ANNT approach, pruning is used in this work.

Dectext, NeuroRule TREPAN and ANN-DT, are similar to our research work in extracting rules using decision tree approach. The difference between ANNT with Dectext, TREPAN, and ANN-DT is the schemes of extracting rules from ANN. ANNT is decompositional approach, where as Dectext, TREPAN and ANN-DT are pedagogical techniques. ANNT can show the role of each hidden unit and more information can be obtained in the form of rules compared to Dectext and TREPAN. On the other hand, NeuroRule is decompositional approach but works only on discrete inputs. ANNT method doesn't require any special training or architecture and works on continuous or discrete data.

2. ANNT Approach

ANNT represents the knowledge learned by ANN in a more understandable form (If-Then rules). DT algorithms use only the data to create the decision tree. ANNT has both the data and the model of the data in the form of ANN. The ANNT approach starts with learning by ANN. Learning in ANN means the adjustment of weights by training the network with training data. By learning, an

ANN becomes knowledgeable. The next stage is to extract the implicit knowledge encoded in ANN in form of rules, using decision tree approach.

Two pruning techniques are used with the ANNT algorithm. One is used to prune the ANN and another to prune the tree. Optimal Brain Damage (OBD) [12] algorithm is used to prune the ANN. Pruning a network will reduce its complexity by removing the redundant connections. Reduced error pruning based on statistical confidence estimation is used to prune the DT. This pruning technique calculates the confidence interval for the error [13]. Both of these pruning methods are evaluated to see the effect on ANNT in terms of accuracy, comprehensibility and fidelity.

2.1. ANN learning

The basic structure of ANN used in this research is a standard three layer network which consists of one layer of input units, a layer of *H* nonlinear hidden units and a layer of one output unit. The available data are randomly divided into 2 subsets: the training and test sets. ANN learns by changing the weights between the nodes in order to optimize a cost function. The training data are presented to the network through the input layer. They are passed to the nodes in the hidden layer after they are multiplied with weights on the connections. Nodes in the hidden layers process the inputs they get and pass to output layer. The product of the output from hidden layer and the weights on the connection between the hidden layer and output layer are processed to find the network outputs. In other words, ANN learns by changing the weights between the nodes in order to optimize a cost function. One of the most commonly used cost functions is the Sum of the Squared Errors (SSE). Before the learning starts, weights are initialized to small random values. Then, each training instance is presented to the network one at a time. An error value is calculated based on the differences between the network output and expected output. Weights are then updated in order to reduce the error value. This is repeated until the error drops to an acceptable level or certain criteria are met.

Given an N dimensional training data p, p=1, 2, 3 …., P, for the j^{th} hidden neuron, the net input z_{inj} and the output activation z_j are computed as follows

$$z_{inj} = v_{oj} + \sum_{j=1}^{N} x_i v_{ij} \tag{1}$$

$$z_j = f(z_{inj}) \tag{2}$$

where x_i denote the i^{th} element of input x, v_{ij} denotes the weight connecting the i^{th} input unit to the j^{th} hidden unit, and v_{oj} is the bias of the j^{th} hidden unit. The activation function f is the hyperbolic tangent expressed by

$$f(z_{inj}) = \frac{e^{z_{inj}} - e^{-z_{inj}}}{e^{z_{inj}} + e^{-z_{inj}}} \tag{3}$$

The output y is expressed in the following equation

$$y = w_o + \sum^{H} z w_j \tag{4}$$

where w_j denotes the weights connecting the j^{th} hidden unit to the output unit and w_{o} is the bias of the output unit.

2.2. Knowledge extraction

The knowledge that has been learned by ANN is very difficult to interpret because it is distributed into the weight set and is represented in analog form. One method of converting the knowledge into comprehensible form is to extract the knowledge in form of rules. In other words, rule extraction is the process of interpreting the knowledge of trained ANN into comprehensible form to the user and at the same time it is useful for gaining insights (relations and pattern) into the training data.

In this stage, the knowledge that is learned by the ANN is extracted in the form of rules using decision tree approach. First, the ANN is decomposed into two parts; hidden to output and input to hidden layer.

2.2.1. Building an output decision tree

Using the output from hidden layer, Eq. (2), as attribute and the actual output from the network, Eq. (4), as class, the output DT is built. The values of the class are denoted with Y_1, Y_2,….Y_{NClass}. At the beginning, only the root is present. Let T be the set of cases associated at the node. The frequency freq(Y_i,T) is computed (Fig. 1: step 1) of cases in T whose class is Y_i for $i \in [1, Nclass]$. If all cases (step 2) in T belong to a same class Y_j then the node is a leaf, with associated class Y_j.

If T contains cases belonging to two or more classes (step 3), then the information gain of each attribute is calculated, see section 2.2.1.1 and [13] for more details. The attribute with the highest information gain (step 4) is selected for test at the node. Tests of discrete attributes consider partitions of T into n subsets, one for each possible value of the attributes. Tests of continuous attributes partition T into two subsets, and certain threshold values have to be determined (step 5). A decision node has s children if T_1, T_2,….T_s are the sets of the splitting produced by the test on the selected attribute (step 6). Obviously s=2 when the selected attribute is continuous, Eq. (6), and s=h for discrete attributes with h possible values, Eq. (5).

```
Form Tree (T)
    Step 1:  ComputeFrequency(T);
    Step 2:  if OneClass or FewCases
                 return a leaf;
         create a decision node N;
    Step 3:  ForEach Attribute A
               ComputeGain(A);
    Step 4:  N.test = AttributeWithBestGAin;
    Step 5:  if N.test is continuous
               find Threshold;
    Step 6:  ForEach T' in the splitting of T
    Step 7:  if T' is Empty
               Child of N is a leaf
         else
    Step 8:  Child of N = FormTree(T');
```

Fig. 1. Tree Construction Algorithm.

If T_i is empty, (step 7) the child node is directly set to be a leaf, with the associated class being the most frequent class at the node. The divide and conquer approach consists of recursively applying the same operations on each non-empty T_i (step 8).

Information gain

The information gain of an attribute a for a set of cases T is calculated as follows. If a is discrete and T_1, T_2,....T_s are the subsets of T consisting of cases with distinct value for attribute a, then:

$$\text{gain} = \text{info}(T) - \sum_{j=1}^{S} \frac{|T_i|}{|T|} \times \text{info}(T_i) \tag{5}$$

If a is a continuous attribute, cases in T are ordered with respect to the value of a. Assume that the ordered values are v_1, v_2, ...,v_m. Consider for $i \in [1,m\text{-}1]$ the value $v = (v_i + v_{i+1})/2$ and the splitting:

$$T_1^v = \{v_j | v_j \leq v\}, T_2^v = \{v_j | v_j > v\} \tag{6}$$

2.2.2. Input decision tree

In step 2, the input trees are built in accordance with branch of the output tree in step 1. Each branch is reflecting the value of z in Eq. (2). For every branch, an input tree will be build. The train data (p), which is used in training ANN is used as attribute and the value of output from hidden neuron, z, is used as classes to build these input trees. To produce smaller input trees, pruning is used. Pruning will simplify the tree by pruning away various subtrees and replacing them with leaves. Smaller tree will result with a smaller number of rules and increases the comprehensibility of the rules.

2.2.3. Rules

In step 3, each input tree is converted into rules. The number of leaves in the tree corresponds to the number of rules. The path from the root to each leaf specifies the conditions for each rule. Rules that are generated from each of the input tree will comprehensibly describe the function of hidden neuron. Combine all the rules extracted to get the final rules, which describe the relationship between the input and the output of the ANN.

2.3. Knowledge extraction: An illustrative example

For ANNT illustration, a network with 4 hidden units as shown in Fig. 2, is trained to classify whether a patient is having thyroid or not. The number of input units corresponds to the number of attributes in the data set. This data set is described by five attributes (resin uptake test, thyroxin, serum triiodothyronine, basal thyroid-stimulating hormone and thyrotropin) with two classes (patient having thyroid as 1 and patient not having thyroid as 0). For simplicity, In1 to In5 (Fig. 2) are used as the name of input instead of the actual attribute name.

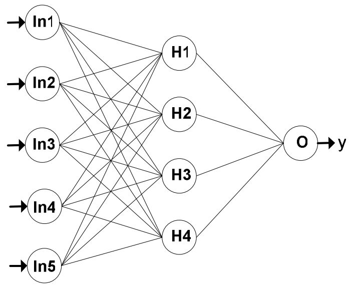

Fig. 2. Two-Layer Neural Network.

The knowledge that is learned in training is stored in the connection links as weights. To extract this knowledge in understandable form, the ANN is decomposed using tree approach. The ANN will be decomposed into two parts; from output layer to hidden layer (Fig. 3), and then from hidden layer to input layer (Fig. 5).

Using the output from hidden neurons as attribute and the actual output of the ANN as class, an output tree is built (Fig. 3). In other words, the outcome of each hidden neurons is used as input data and the actual output from the ANN is used as output data. It should be mentioned that any decision tree method can be utilized to build the tree. The algorithm for building this tree is given in Fig. 1. In Fig. 4, branch H4 is less or equal to -0.0515 is reflecting the value of *z*, Eq. (2), for hidden neuron 4 (H4). From Fig. 4, when H4 is less than or equal to -0.0515 and H1 is less or equal to -0.995, then the patient is having thyroid. It is noted here that two branched are used to classify. Two input trees must be built based on H4 and H1.

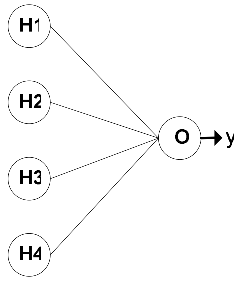

Fig. 3. Decompose of the Neural Network from Hidden Layer to Output.

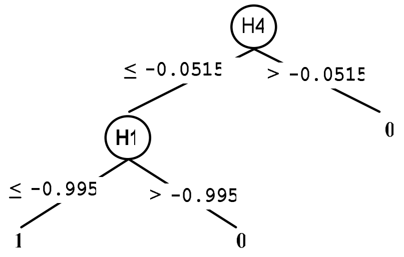

Fig. 4. The Output from Hidden Neurons is Used as Attributes and the Actual Output from the Network is Used as Class to Build this Output Tree.

Figure 5 shows the decomposition of neural network for H4. Here, the train data (*p*) is used as attribute and the output from the hidden neuron (*z*) is used as class. If the outcome of H4 is less or equal to -0.0515, the outcome is converted to 1, which means patient is having thyroid. Using these data, an input decision tree is built (Fig. 6).

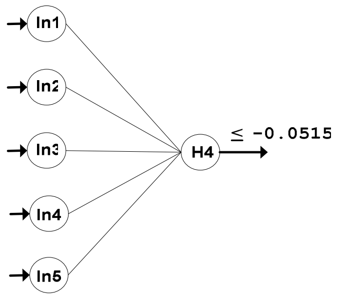

Fig. 5. Decompose of the Neural Network from the Input Layer to Hidden Layer when Hidden Neuron (H4) is less than or equal to -0.0515.

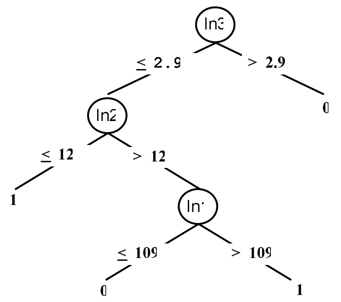

Fig. 6. Input Tree for H4 ≤ -0.0515 The Training Data are used as Attributes and the Output from the H4 is used as Class to Build this Input Tree.

The decision tree in Fig. 6 can be easily converted into *IF – THEN* rule which is shown in Fig. 7. The same procedure is done to decompose H1. The final rule is obtained by combining the rules of H4 and H1 for this example (Fig. 7).

The performance of ANNT approach using unpruned DT (unpruned rules) and ANNT using pruned DT (pruned rules) are evaluated in terms of accuracy, comprehensibility and fidelity. The original data is used to estimate the accuracy of the extracted rules by measuring how well the rules extracted represent the original data. It should be noted here that the rule is extracted based on the actual output from ANN, but the accuracy is evaluated based on the original data set. The accuracy of the rule extracted can be expressed as:

$$\text{Accuracy} = \frac{\text{number of correctly classified data}}{\text{total number of data}} \times 100 \tag{7}$$

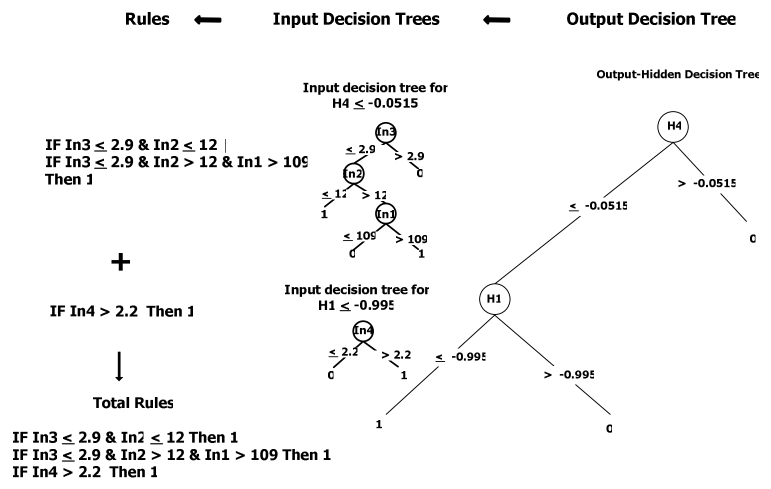
Rules
Input Decision Trees
Output Decision Tree
Input decision tree for H4 ≤ -0.0515
Output-Hidden Decision Tree
IF In3 ≤ 2.9 & In2 ≤ 12 |
IF In3 ≤ 2.9 & In2 > 12 & In1 > 109
Then 1
+
IF In4 > 2.2 Then 1
Input decision tree for H1 ≤ -0.995
Total Rules
IF In3 ≤ 2.9 & In2 ≤ 12 Then 1
IF In3 ≤ 2.9 & In2 > 12 & In1 > 109 Then 1
IF In4 > 2.2 Then 1
In3
≤ 2.9
> 2.9
0
In2
≤ 12
> 12
1
In1
≤ 109
> 109
0
1
In4
≤ 2.2
> 2.2
0
1
H4
≤ -0.0515
> -0.0515
0
H1
≤ -0.995
> -0.995
1
0

Fig. 7. Output Tree to Final Rules.

The fidelity of the rule extraction is a measure of the agreement between the ANN and the extracted rule, in other words, how well the extracted rule set mimics the behaviour of the network. Rule comprehensibility is reported as the number of rules. The smaller the rule set, the more understandable it will be.

3. Pruning

In the present work, pruning algorithms have been proposed to extract the rules from the contributing attributes and also indirectly reducing the redundant rules. Pruning will eliminate the unnecessary connection (weights or neurons) without degrading the network's performance. The small number of connections may be easier to interpret the logic behind their decisions as the network has less opportunity to spread functions over many nodes [14]. A side benefit of pruning is that the small resulting networks have advantages such as economy and speed of operation.

Optimal Brain Damage (OBD) pruning [12] was used to prune the network. OBD is a method to determine the effect of each weight on the cost. The change in the cost function as result of removing a weight is called the saliency. A low saliency is assumed to imply that the weight has a negative influence on the generalization skill. The network optimization is done by ranking the weights according to saliency and then removing the least significant ones. Then an estimate of the generalization error is computed and compared to the best generalization error seen so far. If an improvement is found the network structure is saved. OBD method is based on the second derivatives (Hessian matrix) of the error function.

4. Experimental Results

Three case studies, heart problem, thyroid, and rainfall in Kota Kinabalu have been used to analyze the performance of ANNT. The first two data sets can be obtained from UCI repository [15], while rainfall data are collected from Metrological Department in Kota Kinabalu, Sabah, Malaysia. The heart disease data set is used to find out whether a person has heart disease. This data set, consists of 13 attributes and 2 classes (patient healthy or sick), has 270 instances (150 healthy and 120 sick) with 5 continuous value attributes. The thyroid data set, which consists of 5 attributes and 2 classes (patient having thyroid or not), has 215 instances (150 normal and 65 having thyroid) with all 5 attributes being continuous value. The rainfall data set is used to predict the rainfall pattern in Kota Kinabalu. It contains 732 instances with 10 attributes and two classes (will rain or not). All the attributes are continuous value.

These data sets were randomly divided into two subsets: the training set (60%), and test set (40%). Five groups each with six networks were trained. Each group used the same training data set with randomized initialized weights. In each group, the average results of the six networks are regarded as the result of the group. It should be mentioned that the architecture of the networks has not been finely tuned (have been standardized for all the data set for comparison). The hidden neurons are activated using hyperbolic sigmoid activation function with linear output units. The learning rate is fixed at 0.001 with maximum epoch set at 2000 with 9 hidden neurons chosen.

Table 1 shows the performance of ANNs on heart disease, thyroid and rainfall data sets. The classification accuracy of the trained ANNs is quite high (above 89.9%) for heart disease and thyroid data, where as the classification accuracy of trained ANNs is around 83% for rainfall data set. The analysis of the result shows that most of the misclassified patterns belong to *it will rain* class. Prolonged training (increase the number of iteration) does not help to increase the number of correctly classified patterns.

Table 1. Performance of ANN.

	Training Set				**Test Set**			
	Min	**Max**	**Mean**	**Std Dev**	**Min**	**Max**	**Mean**	**Std Dev**
Heart	87.90	91.85	89.86	1.43	81.11	86.73	83.37	2.18
Thyroid	89.92	95.19	93.05	2.15	87.21	94.18	90.74	2.79
Rain	81.55	84.74	83.13	1.20	75.56	80.00	78.13	1.68

The overall performance of ANNs is determined by measuring the classification accuracy achieved by the ANN on an unseen pattern (generalization) for each data set. Thyroid data set generalized better than heart disease and rain fall data sets.

Table 2 shows the performance of ANNs with OBD pruning on heart disease, thyroid and rainfall data sets. The generalisation of ANN pruned with OBD method increases by 1.41% for thyroid data set, 0.16% for rain and for heart data set, it decreases by 0.47%.

Table 2. Performance of ANN with OBD Pruning.

	Training Set				Test Set			
	Min	**Max**	**Mean**	**Std Dev**	**Min**	**Max**	**Mean**	**Std Dev**
Heart	80.99	92.34	88.01	4.27	80.37	86.73	82.98	2.47
Thyroid	91.63	96.12	94.03	1.81	89.30	94.88	92.02	2.29
Rain	80.37	84.78	82.92	1.78	75.97	79.86	78.26	1.52

Table 3 reports the performance of the extracted rules from the three data sets. The rule accuracy (classification) is measured (in percentage) as the ratio of correctly classified patterns to the total number of patterns (training or testing). It should be mentioned here that rule extracted from the data set is based on the actual output from the ANN. To check the accuracy of the rules, original data set is used.

Table 3 shows that the accuracy of the extracted rule from the training set as well as the generalization of the extracted rules are slightly lower than the performance of ANN for all the data sets.

Table 4 shows that the accuracy of the extracted rule from the training set as well as the generalization of the extracted rules for ANNT with OBD pruning. The performances of ANNT with OBD pruning are lower than ANN with OBD pruning for all the data sets. ANNT with pruning shows 1% improvement on thyroid data set compared to ANNT without pruning, where as on rain data set it decreases by 3%. For heart disease data set, there are no changes.

Table 3. Performance of ANNT.

	Training Set				Test Set			
	Min	**Max**	**Mean**	**Std Dev**	**Min**	**Max**	**Mean**	**Std Dev**
Heart	74.92	85.80	81.07	4.32	72.18	82.96	78.76	4.43
Thyroid	89.46	95.04	92.56	2.15	85.81	93.02	90.12	2.70
Rain	78.77	81.87	80.63	1.19	72.63	79.04	76.30	2.29

Table 4. Performance of ANNT with OBD Pruning.

	Training Set				Test Set			
	Min	**Max**	**Mean**	**Std Dev**	**Min**	**Max**	**Mean**	**Std Dev**
Heart	72.45	84.69	78.88	4.65	74.48	82.96	78.75	3.48
Thyroid	90.70	96.12	93.39	2.15	88.14	94.19	91.05	2.41
Rain	73.43	77.17	76.01	1.72	70.66	76.18	73.83	2.10

Table 5 shows the performance of the ANNT with pruned tree. For the heart disease data set, accuracy of ANNT with pruned tree increase 0.9% for training rules and 0.4 % testing rules. For the thyroid data set, the rule accuracy increases by about 0.2% for the training and decrease 0.2% for testing set, where as ANNT with tree pruning decrease 0.2% of the training set and 0.1% of the test set for rain data.

Table 6 shows the performance of the extracted rule from pruned network and pruned tree. For heart disease and rain data set, the accuracy of the rules extracted increases, by 1.4% and 0.08% respectively, where as the rules accuracy decrease 0.075% for rain set.

The generalization of the rules with network and tree pruning increase 1.04% and 1.2% for heart disease and thyroid respectively and 3% dropped in rain set compared to tree pruning only.

Tables 7 and 8 show the rule extraction performance in term of fidelity and comprehensibility. ANNT shows better performance in term of fidelity compared to ANNT with OBD network pruning for all the three data set. The accuracy in term of fidelity decreases, 0.6% and 0.2%, for heart and thyroid data set respectively, where as rainfall data set decrease 5.5%. The comprehensibility of the extracted rules after tree pruning increases for all the three data set. 38.7% for heart data set, 20% for thyroid data set respectively and 21.8% for rain data set. Comprehensibility of ANNT with tree and network pruning increases, by 15.2% for rain data set and 3.51% for heart data set, where as the comprehensibility dropped 24% for thyroid data set.

Test set accuracy results for ANNT and C4.5 are listed in Table 9. The results are the average of five groups for all the three data sets. The accuracy of ANNT rules is better than C4.5 rules for these data sets.

Table 5. Performance of ANNT with Tree Pruning.

	Training Set				**Test Set**			
	Min	**Max**	**Mean**	**Std Dev**	**Min**	**Max**	**Mean**	**Std Dev**
Heart	75.61	85.31	81.78	4.06	73.15	85.00	79.07	4.78
Thyroid	90.23	95.35	92.71	2.01	85.81	93.02	89.92	2.69
Rain	78.54	82.14	80.45	1.28	72.83	79.05	76.19	2.26

Table 6. Performance of ANNT with OBD and Tree Pruning.

	Training Set				**Test Set**			
	Min	**Max**	**Mean**	**Std Dev**	**Min**	**Max**	**Mean**	**Std Dev**
Heart	74.81	84.81	80.11	3.87	74.63	84.26	79.89	3.78
Thyroid	90.39	95.81	93.33	2.03	88.22	93.95	90.98	2.41
Rain	73.34	77.81	75.70	1.73	70.66	75.97	73.89	1.97

Table 7. Fidelity Performance of Rule Extracted.

	ANNT	**ANNT with Network Prune**
Heart	90.21	89.65
Thyroid	99.48	99.32
Rain	96.99	91.67

Table 8. Comprehensibility of the Rule Extracted.

	Number of Rules			
	ANNT	**ANNT with Tree Pruning**	**ANNT with Network Prune**	**ANNT with Tree and Network Prune**
Heart	62.07	37.67	67.82	41.52
Thyroid	6.90	5.40	9.63	6.90
Rain	430.70	337.27	157.20	129.97

Table 9. Test Set Accuracy Results for ANNT and C4.5.

Method	Heart	Thyroid	Rain fall
ANNT	78.76	90.12	76.30
C4.5	75.00	77.14	76.10

5. Conclusions

In this paper, a method is presented to overcome the comprehensibility problem of ANN by extracting symbolic knowledge from a trained ANN using Tree approach. This enables the user to understand how the ANN is performing its task. This will lead to more confidence in accepting ANN as a data mining tool and could benefit data mining application because it has strong generalization ability. Rule extraction from ANNT approach does not depend on the structure or the training methods. It can also work on continuous as well as discrete data. The knowledge gained can enhance the decision making process and will be a valuable tool in data mining. Experimental results on three data sets show that the ANNT algorithm generates rules that are better than C4.5. Two pruning techniques are used with the ANNT algorithm; one is to prune the neural network and another to prune the tree. The pruned network shows better generalization on the network and tree pruning improves the comprehensibility of the rules.

References

1. Zhou, Z.H. (2004). Comprehensibility of data mining algorithms. *Journal of Computer Science and Technology*, Vol. 19 (2), 249-253.
2. Andrews, R., Diederich, J., and Tickle, A.B. (1995). Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, Vol. 8, No. 6, 373-389.
3. Gallant, S.I. (1983). Connectionist expert systems. *Communications of the ACM*, 31(2). 152-169.
4. LiMin. Fu. (1994). Rule generation from neural networks. *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 24(8), 1114-1124.

5. G. Towell, G., and Shavlik, J. (1993). The extraction of refined rules from knowledge based neural networks. *Machine Learning*, Vol. 13(1), 71-101.
6. Setiono, R., and Liu, H. (1996). Symbolic representation of neural networks. *IEEE Transactions on Computer*, Vol. 29(3), 71-77.
7. Thrun, S.B. (1994). Extracting provably correct rules from neural networks. in *Technical Report IAI-TR-93-5*, Institut fur Informatik III Universitat Bonn.
8. Craven, M.W. (1996). Extracting comprehensible models from trained neural networks. *Ph.D. Thesis*, University of Wisconsin, Madison.
9. Boz, Olcay. (2002). Extracting decision tree from trained neural networks. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 456-461.
10. Schmitz, G.P.J., Aldrich, C., and Gouws, F.S. (1999). ANN-DT: An algorithm for extraction of decision trees from artificial neural networks. *IEEE Transactions on Neural Networks*, Vol. 10(6): 1392-1401.
11. Sestito, S., and Dillon, T. (1994). *Automated knowledge acquisition*. Australia: Prentice-Hall.
12. Le Cun, Y., Denker, J.S., Solla, S.A. (1990). Optimal Brain Damage. *Advances in Neural Information Processing Systems.* 2, 396-404.
13. Quilan, J. (1993). C4.5: *Programs for machine learning*. Morgan Kaufmann, San Mateo, CA.
14. Towell, G., and Shavlik, J.W. (1992). Interpretation of artificial neural networks: Mapping knowledge-based neural networks into rules. In Moody, J.E., Hanson, S. J., and Lippmann, R. P., editors, *Advances in Neural Information Processing Systems* (4), 977—984. Morgan Kaufmann, San Mateo.
15. UCI repository of machine learning: http://www.ics.uci.edu/mlearn/ MLRepository.html.