ARTIFICIAL NEURAL NETWORK TREE APPROACH IN DATA MINING

Kalaiarasi Sonai Muthu Anbananthen[1], Gopala Sainarayanan[2], Ali Chekima[3], Jason Teo[4]
[1,3,4] School of Engineering and Information Technology,
Universiti Malaysia Sabah, 88999 Kota Kinabalu, Sabah, Malaysia. Email: anbakala@ums.edu.my
[2] Department of Electrical and Electronics Engineering
New Horizon College of Engineering, Bangalore, India. Email: jgksai@yahoo.com

ABSTRACT

Artificial neural networks (ANN) have demonstrated good predictive performance in a wide variety of real world problems. However, there are strong arguments as to why ANNs are insufficient for data mining. The arguments are the poor comprehensibility of the learned ANNs, which is the inability to represent the learned knowledge in an understandable way to the users. 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. Experimental results on three data sets show that the proposed algorithm generates rules that are better than C4.5. This paper provides an evaluation of the proposed method in terms of accuracy, comprehensibility and fidelity.

Keywords: ***Data mining, Comprehensibility, Artificial Neural Network, Decision Tree.***

1.0 INTRODUCTION

In the last decade, the capabilities for generating and collecting data have grown explosively. The amount of data stored in computer systems is 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 volumes 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. In other words, the data mining algorithm must be able to produce patterns understandable to common people.

ANN is very successful in identifying hidden pattern from data, which is ANN can recognize relationships and spot trends in huge amounts of data that is not apparent to humans [2]. Its ability to learn from data imitates the human's ability to learn from experience. Although ANN has been used in many applications with remarkable success and has been frequently found to outperform conventional approaches [3, 4], it has not been well exploited in data mining. In data mining, algorithms with good comprehensibility are much desired, but in ANN, this is the main limitation. It is regarded as a "black box", the knowledge learned by ANN is hard to be understood by users because it is concealed in a large amount of connections. Moreover, it is difficult to give an explicit explanation for the reasoning process of trained ANN, which not only weakens the trust of human beings to ANN but also makes it difficult to apply this technology to knowledge discovery or data mining.

2.0 RULE EXTRACTION

The limitation of ANN can be addressed by extracting symbolic rules from trained ANN. Using these extracted rules, ANN's users can understand what the networks have learned and how ANN classify or predict. In other words rules extracted from ANN can be used to explain the reasoning behind the output of ANN. The rules extracted could also interpret the relationship between input and output variables in data. By extracting rules, explanation capability is added to ANN and making it a complete data mining tool.

According to the taxonomy presented by Andrews et al. [5], rule extraction algorithms can be roughly categorized by

i. the schemes of extracting rules from ANN
ii. the portability of the rule extraction technique
iii. the expressive power of the rule extracted (types of rules 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 [6]. KT is derived from the word "Knowledgetron" which was coined by the author to refer to 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 [7]. This algorithm checks each subset and tries to find out if any of these links exceed the bias. If exceeded, then these weights are rewritten as rules. REFANN, a rule extraction was developed by Setiono [14]. This method extracts rules from trained ANN for non-linear function approximation or regression.

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) [8], TREPAN [9], Decision Tree Extractor (Dectext) [10], etc. VIA was designed as a general purpose rule extraction procedure, 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 trained network and extract a classical decision tree from the network. Zhou et al. [15] developed an algorithm REFNE, by using an ensemble neural network to generate new data instances, and then extract symbolic rules from the instances. Garcez et al. [16] developed a method to extract rules from a neural net by first defining a partial ordering on the set of input vectors.

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

The rule extracted from ANN can be expressed in production rule, 'n' of 'm' tree, etc. Dextect extract production rule, where as TREPAN extracts a 'n' of 'm' tree. Both of these algorithms use the decision tree approach to extract rules from trained ANN.

In this paper, an approach named Artificial Neural Network Tree (ANNT), i.e. ANN training preceded by Decision Tree 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 use DT to approximate the function of the trained ANN to improve the comprehensibility of ANN.

Dectext and TREPAN are similar to our research work in extracting rules using the decision tree approach. The difference between ANNT with Dectext and TREPAN is the schemes of extracting rules from ANN. ANNT is a decompositional approach, where as Dectext and TREPAN are pedagogical techniques. More detailed information can be obtained by ANNT approach, compared to the Dectext and TREPAN approach. ANNT shows the role of each layer, especially the hidden layer, that is, the ANNT approach decomposes the network and extracts the rules from output layer to hidden layer then map to input layer, where as in Dectext and TREPAN, only the input and output data are used to extract the rules. Another difference between ANNT and Dectext is that ANNT works on discrete or continuous data whereas Dectext works only with discrete data type. The If-Then rules extracted from ANNT are easier to understand compared to the 'n' of 'm' tree extracted by TREPAN. The ANNT method will not be subjected to any special training or architecture.

3.0 ANNT APPROACH

ANNT represents the knowledge learnt 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 the form of rules using the tree approach. To reduce the number of rulers extracted, that is, to improve the comprehensibility of the rules, the DT is pruned. Reduced error pruning based on statistical confidence estimation is used to prune the DT. This pruning technique calculates the confidence interval

for the error [12]. In this research, rule comprehensibility is reported as the number of rules. The smaller the rule set, the more understandable it will be.

3.1 ANN Learning

The basic structure of ANN used in this research is a standard three layer network consisting 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. One of the most commonly used cost functions is the Sum Squared Errors (SSE). Before learning starts, weights are initialized to small random values. Training data are then presented to the network through the input layer. Nodes in the hidden layers process the inputs they get from the input layer and pass them to the output layer. An error value is calculated based on the differences between the network's output and expected outputs. 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, the net input (z_{in}) and output (z) of hidden neuron are computed as follows:

$$z_{in} = v_o + \sum^{N} xv_i \qquad (1)$$

$$z = f(z_{in}) \qquad (2)$$

where x denotes the input, v_i denotes the weight connecting the input unit to the hidden unit and v_o is the bias of the hidden unit. The activation function (*f*) is hyperbolic tangent:

$$f(z_{in}) = \frac{e^{z_{in}} - e^{-z_{in}}}{e^{z_{in}} + e^{-z_{in}}} \qquad (3)$$

The output y is

$$y = w_0 + \sum^{H} zw_j \qquad (4)$$

where w_j denotes the weights connecting the hidden unit to the output unit and w_o is the bias of the output unit.

Error value is calculated based on the differences between the network's output (y) and expected outputs. This error value is the basis for the backward pass, where the errors are passed back through the network by computing the contribution of each processing unit and deriving the responding adjustment needed to produce the correct output. Weights are then updated to make the actual output (y) move closer to the expected output.

3.2 Knowledge Extraction

The knowledge that has been learnt 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 a comprehensible form is to extract the knowledge in the form of rules. In other words, the 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 learnt by ANN is extracted in the form of rules using the tree approach. First, the ANN is decomposed into two parts; hidden to output layer and input to hidden layer.

3.2.1 Building a Output Tree

To build the tree, create a data set (T) with of the output from the hidden layer (z) from Equation 2 as an attribute and the actual output (y) from the network (Equation 4) as class. The values of the class are denoted with y_1, y_2….y_{NClass}. At the beginning, only the root is present. At each node the following algorithm (Fig. 1) is executed,

trying to exploit the locally best choice. Let 'T' be the set of cases associated at the node. The frequency (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 [13]. 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 $T_1, T_2, \ldots T_s$ 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, and *s* = *h* for discrete attributes with h possible values.

```
Form Tree
    Step 1:  ComputeFrequency(T);
    Srep 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 T1 in the splitting of T
    Step 7:      if T1 is Empty
                     Child of N is aleaf
             else
    Step 8:          Child of N = FormTree;
```

Fig. 1: Tree Construction Algorithm

If 'T_1' 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_1' (step 8).

3.2.2 Input Trees

In step 2, the input trees are built in accordance with the branch of the output tree in step 1. Each branch is reflecting the value of z (Equation 2). For every branch, an input tree will be build. The train data (p), which is used in training ANN is used as an 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 trees will result with a smaller number of rules and increase the comprehensibility of the rules.

3.2.3 Rules

In step 3, convert each input tree 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.

3.3 Knowledge extraction : An Illustrative Example

Thyroid data set [17] is used for ANNT illustration. 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) is used as the name of input instead of the actual attribute name. As shown in Fig. 2, a three layer network with 4 hidden units, is trained.

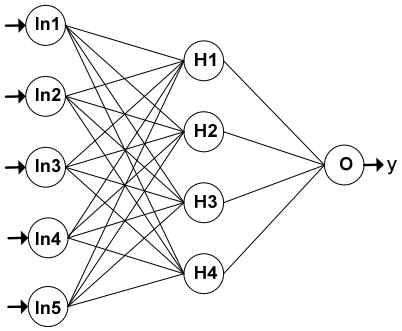

Fig. 2: Three Layer Neural Network

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

Using the output from hidden neurons as attribute and the actual output of the ANN as class (Fig. 3), an output tree is built. In other words, the outcome of each hidden neuron is used as input data and the actual output from the ANN is used as output data. It should be mentioned here that any 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 equals to -0.0515 is reflecting the value of z (Equation 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 branches are used to classify. Two input trees must be built based on H4 and H1.

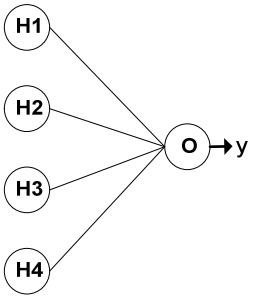

Fig. 3: Decomposition of the neural network from hidden layer to output

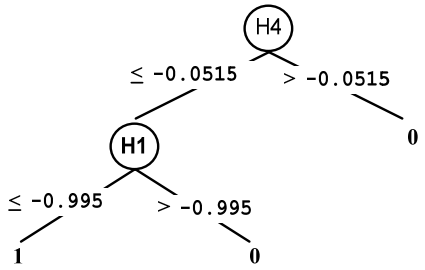

Fig. 4: The output from hidden neurons is used as attributes and the actual output from the network is used as class to build this output tree

Fig. 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 equals to -0.0515, the outcome is converted to 1, which means the patient is having thyroid. Using these data, an input tree is build (Fig. 6).

The 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 tree (unpruned rules) and ANNT using pruned tree (pruned rules) is 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:

$$\text{Accuracy} = \frac{\text{Total number of correctly classified data}}{\text{Total number of data}} \times 100 \qquad (5)$$

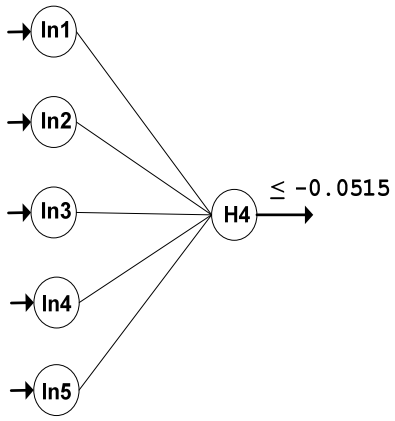

Fig. 5: Decomposition of the neural network from the input layer to hidden layer when hidden neuron (H4) is less than or equal to -0.0515.

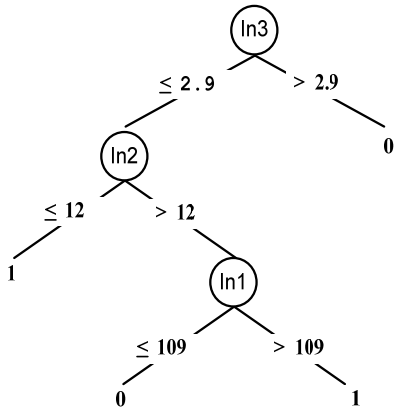

Fig. 6: Input tree for H4 ≤ -0.0515. The training data is used as attributes and the output from the H4 is used as class to build this input tree

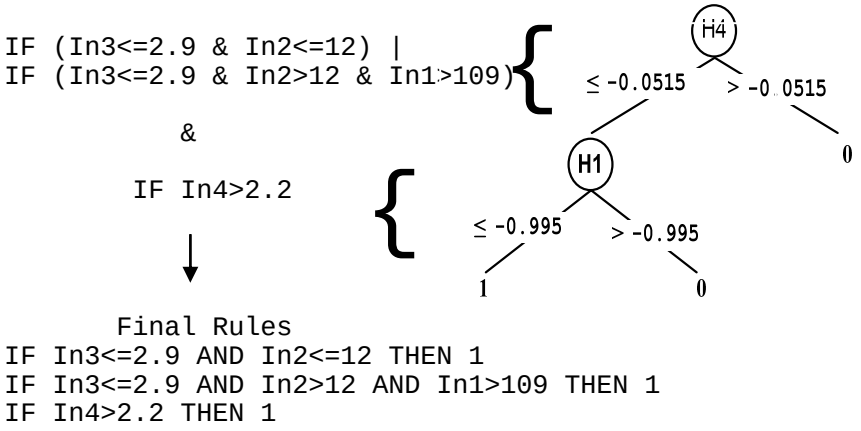

Fig. 7: Output tree to Final Rules

The fidelity of the rule extraction is a measure of the agreement between the ANN and the extracted rule (how well the extracted rule set mimics the behaviour of the network). Rule comprehensibility is reported as the number of rules.

$$\text{Fidelity} = \frac{\text{Total number of data} - (N_{ANN} - N_{Rules})}{\text{Total number of data}} \times 100 \qquad (6)$$

where N_{ANN} is total number correctly classified data in ANN and N_{Rules} is total number correctly classified data in rule set

4.0 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 [17] while rainfall data were collected from the 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 (150 healthy and 120 sick) instances with 5 continuous value attributes. The thyroid data set, which consists of 5 attributes and 2 classes (patient having thyroid or not), has 215 (150 normal and 65 having thyroid) instances 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 (standardized for all the data sets 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.

Tables 1 to 3 show the train and test set classification (accuracy) results of ANNs on heart, thyroid and rainfall data sets. Each row in the table represents the minimum, maximum, average (in percentage) and standard division of each group for training and testing data sets. The average classification accuracy of the trained ANNs is quite high (above 89.9%) for heart and thyroid data, where as the classification accuracy of trained ANNs is around 83% for rainfall data set. The analysis of the results for rainfall data shows that most of the misclassified patterns belong to the *it will rain* class. Prolonged training (increase iteration) does not help to increase the number of correctly classified patterns.

Table 1: Performance of ANN for Heart Data Set

	Training Set				**Test Set**			
	Minimum	Maximum	Mean	Std Dev	Minimum	Maximum	Mean	Std Dev
Group A	87.04	93.21	90.12	2.21	79.63	85.19	82.56	2.30
Group B	85.80	90.74	88.40	1.82	81.48	85.19	83.15	1.52
Group C	88.89	93.83	90.74	1.66	81.48	93.83	85.24	4.46
Group D	87.65	90.12	89.30	1.08	81.48	85.19	83.80	1.52
Group E	90.12	91.36	90.74	0.39	81.48	84.26	82.10	1.12

Table 2: Performance of ANN for Thyroid Data Set

	Training Set				**Test Set**			
	Minimum	Maximum	Mean	Std Dev	Minimum	Maximum	Mean	Std Dev
Group A	88.37	96.12	92.38	3.15	88.37	96.51	92.25	3.09
Group B	89.92	93.80	92.51	2.00	89.54	91.86	90.50	0.88
Group C	93.02	97.67	95.61	1.81	86.05	97.67	91.67	4.61
Group D	89.15	95.35	93.15	2.27	84.88	93.02	88.76	3.50
Group E	89.15	93.02	91.60	1.50	87.21	91.86	90.51	1.86

Table 3: Performance for ANN on Rain Data Set

	Training Set				**Test Set**			
	Minimum	Maximum	Mean	Std Dev	Minimum	Maximum	Mean	Std Dev
Group A	80.64	83.6	82.61	1.13	78.16	80.89	79.98	0.98
Group B	80.18	84.06	81.74	1.33	76.11	80.55	78.55	1.64
Group C	81.78	85.65	83.56	1.41	75.09	79.86	77.87	1.99
Group D	82.46	85.42	83.90	1.18	74.74	79.86	77.30	1.74
Group E	82.69	84.97	83.83	0.96	73.72	78.84	76.96	2.03

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. The thyroid data set generalized better than the heart and rain fall data sets.

Tables 4 to 6 report 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 the rule extracted from the data sets is based on the actual output from the ANN. To check the accuracy of the rules, the original data set is used.

Table 4: Rule Extraction Performance for Heart Data Set

	Training Set				**Test Set**			
	Minimum	Maximum	Mean	Std Dev	Minimum	Maximum	Mean	Std Dev
Group A	72.14	86.42	80.75	5.66	73.15	83.33	79.48	3.58
Group B	75.31	81.48	79.63	2.54	71.56	83.33	79.13	4.64
Group C	76.54	90.12	82.10	5.38	74.07	87.04	79.63	5.59
Group D	74.07	83.33	79.11	4.18	74.07	85.19	80.71	5.09
Group E	76.54	87.65	83.74	3.84	74.07	75.93	74.85	0.70

Table 5: Rule Extraction Performance for Thyroid Data Set

	Training Set				**Test Set**			
	Minimum	Maximum	Mean	Std Dev	Minimum	Maximum	Mean	Std Dev
Group A	89.15	94.57	91.99	2.23	87.21	94.19	91.28	2.73
Group B	87.60	94.57	91.47	2.64	87.21	90.70	89.15	1.20
Group C	91.47	96.9	94.57	1.96	86.05	94.19	90.12	3.01
Group D	89.92	94.57	92.89	1.86	83.72	93.02	89.73	3.63
Group E	89.15	94.57	91.86	2.06	84.88	93.02	90.31	2.91

Table 6: Rule Extraction Performance for Rain Data Set

	Training Set				**Test Set**			
	Minimum	Maximum	Mean	Std Dev	Minimum	Maximum	Mean	Std Dev
Group A	78.36	81.09	79.57	1.08	73.72	79.86	77.02	2.10
Group B	77.22	79.95	79.35	1.06	73.72	80.20	77.25	2.39
Group C	79.73	82.46	81.51	0.97	70.31	78.16	75.32	2.62
Group D	79.73	83.37	81.59	1.53	73.04	78.84	76.39	2.04
Group E	78.82	82.46	81.13	1.32	72.35	78.16	75.54	2.32

Tables 4 to 6 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.

Tables 7 to 9 show the performance of the extracted rule from pruned tree. For the heart data set, extracted rule from pruned tree increases the accuracy of the rules (0.87% for training rules and 0.4% for testing rules). For the thyroid data set, the extracted rules accuracy increases by about 0.17 % for the training set and decrease 0.2% for testing set. Accuracy of the rules decreases (0.2% of the train set and 0.1% of the test set) for the rainfall data set.

Table 7: Rule Pruning Performance on Heart Data Set

	Training Set				**Test Set**			
	Minimum	Maximum	Mean	Std Dev	Minimum	Maximum	Mean	Std Dev
Group A	77.16	87.04	83.23	3.54	74.07	85.19	80.40	3.63
Group B	70.03	83.33	80.18	5.72	71.30	83.33	77.29	5.49
Group C	76.54	85.80	81.89	4.31	72.22	87.04	78.86	6.53
Group D	75.31	83.33	79.63	3.70	75.00	85.19	82.72	4.05
Group E	79.01	87.04	83.95	3.03	73.15	84.26	76.08	4.20

Table 8: Rule Pruning Performance for Thyroid Data Set

	Training Set				**Test Set**			
	Minimum	Maximum	Mean	Std Dev	Minimum	Maximum	Mean	Std Dev
Group A	89.15	95.35	92.12	2.42	87.21	94.19	91.28	2.73
Group B	88.37	94.57	91.21	2.62	88.37	91.86	89.53	1.27
Group C	91.47	97.67	94.70	2.22	86.05	94.19	89.92	2.91
Group D	91.47	94.57	93.28	1.17	82.56	93.02	89.54	4.03
Group E	90.7	94.57	92.25	1.62	84.88	91.86	89.34	2.49

Table 9: Rule Pruning Performance for Rain Data Set

	Training Set				**Test Set**			
	Minimum	Maximum	Mean	Std Dev	Minimum	Maximum	Mean	Std Dev
Group A	77.90	81.55	79.61	1.41	74.74	79.18	76.96	1.54
Group B	77.90	80.64	79.69	0.96	73.72	80.55	77.53	2.35
Group C	79.50	82.69	81.40	1.06	70.31	78.50	75.26	2.76
Group D	79.04	83.60	81.25	1.72	72.35	78.84	75.88	2.46
Group E	78.36	82.23	80.30	1.27	73.04	78.16	75.31	2.20

Tables 10 to 12 show the rule extraction performance in term of fidelity and comprehensibility. The thyroid and rainfall data sets show high fidelity (around 99.4% and 97.5%) relatively, compared to the heart data set (91.2%). The comprehensibility of the extracted rules after pruning increases 38.7% for the heart data set, 20% for thyroid dthe ata set and 21.8% for the rain data set. Tree pruning increases the comprehensibility and decreases the accuracy of the rule extracted.

Test set accuracy results for ANNT and C4.5 are listed in Table 13. 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 10: Rule Extraction Performance on Fidelity and Comprehensibility Before and After Tree Pruning for Heart Data Set

	Fidelity	**Before Pruning Num of Rule**	**After Pruning Num of Rule**
Group A	90.62	56.33	32.33
Group B	91.24	89.20	53.20
Group C	91.36	63.67	31.50
Group D	89.81	58.50	42.67
Group E	93.00	42.67	28.67

Table 11: Rule Extraction Performance on Fidelity and Comprehensibility Before and After Tree Pruning for Thyroid Data Set

	Fidelity	**Before Pruning Num of Rule**	**After Pruning Num of Rule**
Group A	99.61	6.33	5.33
Group B	98.97	9.67	7.00
Group C	98.97	7.50	5.17
Group D	99.74	6.67	5.50
Group E	99.74	4.33	4.00

Table 12: Rule Extraction Performance on Fidelity and Comprehensibility Before and After Tree Pruning for Rain Data Set

	Fidelity	**Before Pruning Num of Rule**	**After Pruning Num of Rule**
Group A	96.96	405.33	319.33
Group B	97.61	384.50	280.50
Group C	97.95	379.33	308.00
Group D	97.68	478.50	373.83
Group E	97.30	505.83	404.67

Table 13: Test Set Accuracy Results for ANNT and C4.5

Method	**Heart**	**Thyroid**	**Rain fall**
ANNT	78.76	90.12	76.30
C4.5	71.79	89.75	74.91

5.0 DISCUSSION

It is hard to make a direct comparison between ANNT and the other methods as published works include results obtained from only a small number of data sets. Most of the earlier works are only trying to interpret ANN but in this research, attempts are also made to extract the knowledge (relationships) that exists from the data set. With this, ANNT is compared following the guidelines given in the work of Andrews et al. [5] following 4 dimensions:

i. The Expressive Power of the Extracted Rules

The ANNT method expresses the extracted knowledge as 'IF..THEN' statements (the nodes of the tree) combined with logical 'AND / OR' (the links). The knowledge produced in the form of rules are easy to understand by users. Examples of the extracted rules are shown in Fig. 7. From these extracted rules, the user can understand how ANN classifies or predicts.

ii. Quality of the rules extracted

The quality of the rules can be measured by accuracy, comprehensibility and fidelity. The ANNT's average accuracy for the heart data set 78.8%, thyroid 90% and rain fall 76.3%. These results indicate that there is a slight drop in ANNT accuracy by 5% for the heart data set, 0.7% for thyroid data set and 2.3% for rainfall data set compared to ANN accuracy. Comprehensibility of ANNT is compared before and after tree pruning in this research. For all three data sets, comprehensibility of the rule increase, 38.7% for the heart data set, 20% for the thyroid data set and 21.7% for the rainfall data sets after tree pruning. The trade off is between accuracy and comprehensibility. In term of fidelity, ANNT of thyroid and rainfall data sets shows high fidelity compared to heart data set. In this research, the concentration is more on the accuracy and comprehensibility than fidelity of the extracted rules.

iii. Portability

Portability means the extent to which the underlying ANN incorporates specialized training regimes. ANNT does not need any specialized training or architecture and can be used on binary, discrete or continuous data.

iv. Network Structure and Generalization

ANNT is flexible and can be used on any type of feed forward networks or groups of neural networks as a pre-process. Since the pre processor of ANNT is ANN, ANNT will have high robustness to noisy data. In this research it is also observed that the extracted rule set has shown better generalization performance than the C4.5 for all the three data sets.

6.0 CONCLUSION

In this paper, a method is presented to overcome the comprehensibility problem of ANN by extracting symbolic knowledge from a trained ANN using the tree approach. This enables the user to understand how 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 the ANNT approach does not depend on the structure or the training methods. It can also work on continuous and 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. Pruning of Decision Tree helps to improve the comprehensibility of the rules extracted.

REFERENCES

[1] Z. H. Zhou, "Comprehensibility of data mining algorithms", *Journal of Computer Science and Technology*, Vol. 19 No.2, 2004, pp. 249-253.

[2] A. F. Murray, *Applications of Neural Networks*, Boston: Kluwer Academic, 1995.

[3] Z. H. Zhou, Yuan Jiang, S. F. Chen, "A General Neural Framework for Classification Rule Mining", *IJCSS*, Vol. 1 No.2, 2000, pp. 154-168.

[4] J. W. Shavlik, R.J. Mooney, and G.G. Towell, "Symbolic and neural learning algorithms: An experimental comparison", *Machine Learning*, Vol. 6, 1991, pp. 111-143.

[5] R. Andrews, J. Diederich, and A. B. Tickle, "Survey and critique of techniques for extracting rules from trained artificial neural networks", *Knowledge-Based Systems*, Vol. 8 No. 6, 1995, pp. 373-389.

[6] LiMin. Fu, "Rule generation from neural networks", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 24 No.8, 1994, pp.1114-1124.

[7] G. Towell and J. Shavlik, "The extraction of refined rules from knowledge based neural networks", *Machine Learning*, Vol. 131, 1993, pp. 71-101.

[8] S. B. Thrun, "Extracting provably correct rules from neural networks", in *Technical Report IAI-TR-93-5*, Institut fur Informatik III Universitat Bonn, 1994.

[9] M. W. Craven, "Extracting comprehensible models from trained neural networks", *Ph.D. Thesis*, University of Wisconsin, Madison, 1996.

[10] Olcay Boz, "Extracting decision tree from trained neural networks", *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002, pp. 456-461.

[11] S.Sestito and T.Dillon, *Automated Knowledge Acquisition*, Australia: Prentice, 1994.

[12] J. Quilan, "Simplifying Decision Tree", *International Journal of Man-Machine Studies*, Vol. 27, 1987, pp. 221-234.

[13] J. Quilan, *C4.5:Programs for Machine Learning*, San Mateo, CA, Morgan Kaufmann, 1993.

[14] R. Setiono, K. H. Wee, and M. J. Zurada, "Extraction of Rules from artificial neural network for nonlinear regression", *IEEE Transaction Neural Networks*, Vol. 23 No. 23, 2002, pp. 564-577.

[15] Z. H. Zhou, Y. Jiang, Y. B. Yang, and S.F. Chen, "Extracting neural networks from trained neural network Ensembles", *AI Communications*, Vol. 16 No.1, pp. 3-15, 2003.

[16] A. Garcez, S d'Avila, K. Broda, D.M. Gabbay, "Symbolic knowledge extraction from trained neural networks: A sound approach", *Artificial Intelligence*, Vol. 125, 2001, pp. 155-207.

[17] http://www.liacc.up.pt/ML/statlog/datasets/

BIOGRAPHY

Kalaiarasi Sonai Muthu Anbananthen is a lecturer in computer science at the School of Engineering and Information Technology. Her interests are in the areas of data mining and artificial intelligence. She has published a number of papers related to these areas.

Gopala Sainarayanan is currently a Lecturer in electrical and electronics engineering at the New Horizon College of Engineering, Bangalore, India. He received his B.E., M.E., and Ph.D. degrees, respectively, from Annamali University, India, Bharathiar University, India, and Universiti Malaysia Sabah, Malaysia, in 1998, 2000, and 2002. His research interests are in the areas of vision rehabilitation, medical imaging, intelligent control and artificial neural network.

Ali Chekima received his BEngg in Electronics from Ecole Nationale Polytechnique of Algiers in 1976 and his MSc and PhD both in Electrical Engineering from Rensselear Polytechnic Institute Troy, New York, in 1979 and 1984 respectively. He joined the Electronics Department at the Ecole Nationale Polytechnique in 1984, where he was chairman of the Scientific Committee of the Department as well as in-charge of the postgraduate program while teaching at both graduate and undergraduate levels. He was a member of several scientific committees at the national level. He has been working as an Associate Professor at the School of Engineering and Information Technology at Universiti Malaysia Sabah since 1996. His research interests include signal processing, pattern recognition, medical imaging, artificial intelligence and data mining.

Jason Teo is a senior lecturer in computer science at the School of Engineering and Information Technology and the deputy director of the Centre for Artificial Intelligence, Universiti Malaysia Sabah. He received his doctorate in information technology from the University of New South Wales, Australia, researching Pareto artificial evolution of virtual legged organisms. He has over 60 publications in the areas of artificial life, evolutionary robotics, evolutionary computing and swarm intelligence. His current research interests focus on the theory and applications of evolutionary multi-objective optimization algorithms in co-evolution, robotics and metaheuristics.