A HYBRID LEARNING ALGORITHM

Jihane BOULAHIA SMIRANI and Fériel MOURIA BEJI (Member IEEE)
Département d'informatique
Ecole Nationale des Sciences d'Informatique (ENSI)
Le complexe universitaire la Manouba
TUNISIA
Jihane.Smirani@fsb.rnu.tn
pdg@ati.tn

Abstract: The integration of symbolic prior knowledge and neural networks in so-called Knowledge and neural networks is becoming increasingly popular for solving difficult real-world problems[1]. Hybrid intelligent systems that combine and artificial neural network systems typically have four phases involving domain knowledge representation, mapping into connectionist network, network training, and rule extraction respectively. In order to obtain a concise set of symbolic rules, redundant and irrelevant units and connections of a trained neural network are usually removed by a network pruning algorithm before rule are extracted Typical pruning algorithms require retraining the network, which incurs additional cost. In this paper, we introduce a new rule extraction technique without network retraining. Our technique is a universal and comprehensive approach that extracts all embedded knowledge in a trained artificial neural network and represents it in a rule base format. Experimental results show that the size and the predictive accuracy of the rule generated are comparable to those extracted by another method, which prunes and retrains the network.

Key Words: neural network, hybrid neuro-symbolic system, rule extraction, pruning algorithm.

,

1 INTRODUCTION

The lack of validation tools is often one of the reasons for not using neural systems in practice. For instance, physicians cannot trust a diagnosis system without explanation of its responses. The difficulty of justification of neural network responses is due to its distributed internal representation. More particularly, the overall network decision mechanism is represented onto a space of connection weights and activation values which has an exponential size and so in practice it cannot be entirely explored.
Researchers in the field of symbolic rule extraction from neural networks have proposed several algorithms. They are concisely described by the taxonomy proposed by Andrews at al.[1]. Briefly, symbolic rule extraction methods belong to three categories: pedagogical, decompositional, and electric. In the pedagogical approach symbolic rules are generated by an inductive symbolic algorithm which globally analyses expressions related to the input and the output layer. For the decompositional approach, symbolic rules are determined by analysing the weights at the level of each hidden neuron and each output neuron. Finally, the electric approach is a combination of the pedagogical and decompositional strategies.

In order to obtain a concise set of symbolic rules, redundant and irrelevant units and connections of a trained neural network are usually removed by a network pruning algorithm before rules are extracted [1]. This process can be time-consuming as most algorithms for neural network such as Optimal Brain Surgeon[2], Hagiwara algorithm[3]. They retrain the network after removing some connections or units. The retrained network is then checked to see if any of its remaining units or connections meets the criteria for further removal. More often than not, the amount of computations incurred during retraining is much higher than that needed to train the original fully connected network. This paper proposes a new rule extraction technique from trained feedforward neural networks with a single hidden layer. The technique does not require network pruning and hence no network retraining is necessary. By eliminating the need to retrain the network, we can speed up the process of rule extraction considerably and thus make neural networks an attractive tool for generating symbolic classification rules.

2 THE ALGORITHM

The proposed technique consists of three main parts: the first part is a network training algorithm that minimizes a cross-entropy error function augmented by a penalty function. The minimization of the augmented error function ensures that connections from irrelevant inputs have very small weights. Such connections can be removed without affecting the network's classification accuracy. The second part is consisting in the distinction

between the relevant network inputs from the irrelevant ones. We have developed a simple criterion for removing the network connections from the input units to the hidden unit that does not affect the network's classification accuracy. A group of connections from the input units to a hidden unit can be removed at once if they satisfy this criterion. The third part is the extraction of the symbolic rules relating input with network' outputs.

2.1 Training phase

It has been shown that the cross-entropy error function improves the convergence of network training over the standard least-squares error function [5,6,7], while the penalty function F(w,v) is added to encourage weight decay [8]. Each network connection that has nonzero weight incurs a cost. By minimizing the augmented error function we expect those connections that are not useful for classifying the patterns to have small weights. Given an input pattern p, p =1,2,..., P, the network's output unit value S_{ip} and hidden unit activation value H_{jp} are computed as follows:

$$S_{ip} = d\left(\sum_{j=1}^{J} v_{ij} H_{jp} \right)$$

$$H_{jp} = d\left(w_j x_p\right) = d\left(\sum_{k=1}^{K} w_{jk} x_{kp} \right)$$

Where $x_{kp} \in [0,1]$ is the value of input unit k given pattern p, w_{jk} is the weight of the connection from input unit k to hidden unit j, v_{ij} is the weight of the connection from hidden unit j to output unit i, and $d(e)$ is the sigmoid function $1/1+e^{-e}$. J and K are the number of hidden units and input units, respectively. Each pattern x_p belongs to one of the C possible classes $C_1, C_2, ..., C_c$. The target value for pattern p at output unit i is denoted by t_{ip}. For binary classification problem, one output unit with binary encoding is used. For classification problems with C>2 classes, C output units are used in the network. If pattern p belongs to class c, then $t_{cp} = 1$ and $t_{ip} = 0, \forall i \neq c$. The network is trained to minimize the augmented cross-entropy error function

$$q(w,v) = F(w,v) - \sum_{i=1}^{C} \sum_{p=1}^{P} t_{ip} \log S_{ip} + (1 - t_{ip}) \log(1 - S_{ip})$$

With this minimisation, we expect those connections that are not useful for classifying the patterns to have small weights. Compared to the standard backpropagation method, this method has been shown to converge much faster [9].

2.2 Identification of relevant units Phase

The information gain method is used to identify the relevant hidden unit. For this purpose, the C4.5 algorithm is employed.

Given a data set S, recursively a decision tree is generated:

1.If S contains no example, the most frequent class at the parent of this node is chosen as the class, stop.

2. If S contains one or more examples, all examples belonging to a single class C_C.

3 If S contains examples belonging to a mixture of classes, information gain is then used as a heuristic to split S into partitions based on the values of a single feature

The decision tree is built using the hidden unit activations of training patterns that have been correctly classified by the neural network along with the patterns' class labels.

Suppose that each pattern in the data set S belongs to one of the C classes, and n_C is the number of patterns in class C_C, the expected information for classification is

$$I(S) = -\sum_{c=1}^{C} \frac{n_C}{N} \log_2 \frac{n_C}{N}$$

where the number of patterns in the set S is

$$N = \sum_{c=1}^{C} n_c .$$

For hidden unit j, its activation values, its activation values H_{jp} in response to patterns p, p =1,2,…,n., the information gained by splitting the data set S into S1 and S2 is:

$$Gain(H_{jp}) = -\sum_{c=1}^{C} \frac{n_c}{N} \log_2 \frac{n_c}{N} - \left[-\sum_{c=1}^{C} \frac{n_{c1}}{N_1} \log_2 \frac{n_{c1}}{N_1} + \sum_{c=1}^{C} \frac{n_{c2}}{N_2} \log_2 \frac{n_{c2}}{N_2} \right]$$

The normalized gain is

$$NGain(H_{jp}) = Gain(H_{jp}) / \left[-\sum_{j=1}^{2} (N_j / N) \log_2 (N_j / N) \right]$$

The root node of the decision tree contains a test condition, which involves the hidden unit whose activation values give the highest normalized gain. The complete decision tree is generated by applying the same procedure to the subsets of the data at the tow branches of a decision node. To identify the relevant input connections, for each hidden unit j, one or more of its connection weights from the input units may be sufficiently small that they can be removed without affecting the overall classification accuracy. The criterion for removing these irrelevant connections is given below. Let the splitting condition for a node in the decision tree is $H_{jp} \leq H_{jt}$ for some t. Let S be the set of input units whose connections to hidden unit j satisfy the following condition:

$$\sum_{k \in S} |w_{jk}| < 2(H_{j,t+1} - H_{jt})$$

And S' be the complement of S. Then, by changing the splitting condition to

$$H_{jp} \le (H_{j,t+1} + H_{jt}) / 2$$

2.3. Rule extraction Phase

The data set used for extraction rules contain only relevant unit that we have determined in section 2. Each cluster in a hidden unit forms a class. That is, if there are x relevant hidden units left in the network, there will be x such data sets. The number of classes in each data set is solely determined by the number of clusters in the corresponding hidden unit. By computing the inverse of the sigmoid function for all node splitting conditions in a decision tree, we obtain conditions that are linear combinations of the input attributes of the data. After that we remove negative weights. This may be possible by replacing the corresponding inputs with their complement. For example, suppose the attribute X has 2 discrete values $\{x_1, x_2\}$ and 2 binary inputs (I_1, I_2) have been used to represent them: $X = x_1 \Leftrightarrow (I_1, I_2) = (1,0)$ and $X = x_2 \Leftrightarrow (I_1, I_2) = (0,1)$. If the weight w_1 is negative, then we can replace x_1 by its complement, which is x_2:

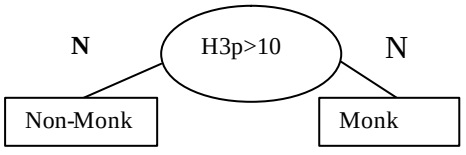
$$w_1 x_1 = w_1(1 - x_2) = -w_1 x_2 + w_1$$

Finally, we divide all the weights by the smallest w_i.

3. ILLUSTRATIVE EXAMPLE: Monk3 problem

A pattern is classified as monk in this problem if (jacket_color=green and holding=sword) or (jacket_color ? blue and body_shape ? octagon). Our algorithm generates tree with a total of 3 nodes, the root node and 2 child nodes. This indicates that the patterns in the data set of this problem are linearly separable, i.e., there exists a hyperplane such that all the monks are on one side of it and the no-monks on the other side. For problems with linearly separable patterns.

An example of a decision tree that is generated is as follows:



Our algorithm removes irrelevant connections to the hidden unit 3 of the network and obtains :

If (-5.2 x4 + 2.8 x11 –5.2 x12 + 2.6 x13) > 0.34 then class0 (not monk) else if (-5.2 x4 + 2.8 x11 –5.2 x12 + 2.6 x13) > 0.34 then class1 (monk)

Twelve different combinations of {x4,x11,x12,x13} are possible (two for x4, two for x11 and three for x12 and 5 for x13) and they are all represented in the training data set. The algorithm takes as input these 12 different combinations along with the corresponding class labels and outputs the following table where the symbol * is used to indicate "do not care" value.

Rule	x4	x11	x12	x13	Monk?
0	0	*	0	*	Yes
1	*	*	1	*	No
2	*	1	0	1	Yes
3	1	*	*	0	No
4	1	0	*	*	No

The generated rules are :

Rule 0 : If x4=x12=0, then monk,

Rule2: If x11=x13=1 and x12=0, then monk.

In terms of the original attributes, the equivalent rules are

Rule0: If (body_shape <> octagon) and (jacket_color <> blue) then monk.

Rule 2: If (holding = sword) and (jacket_color=green) then monk.

4.EXPERIMENTATION RESULTS

The effectiveness of ANPREX has been tested on 15 problems listed in figure 1 and figure 2. The data sets were obtained from UCI repository [10] or generated according to the function definitions given by vitalta, Blix and Rendell[11]. Each data set was randomly divided into three subsets: the training set (40%), the cross validation set (10%) and the test set(50%). For all experiments, the initial number of hidden units in the network was 10. Figure 1 and Figure 2 compares the tree size (the number of nodes) and the predictive accuracy of the decision trees generated by ANPREX with those generated by C4.5 and ANREX an Algorithm for Neural Rule EXtraction that we have implemented and tested [12], this algorithm extract rules from pruned neural network. The decision tree extracted by ANPREX is smaller than the tree generated by C4.5.

ANPREX and C4.5 do not require the cross validation set, hence 50% of the data was used for each training session. The comparison between ANREX and ANPREX **shows** that there is no significant difference in the predictive accuracy and the size of the decision trees generated by both methods. It is hard to make direct comparisons between our contribution: ANPREX and other methods as published work include results obtained from only a small number of data sets. Any good neural network rule extraction algorithm can be expected to extract rules with similar test accuracy as the networks. We also compare ANPREX performance with that of a method that extracts rules from pruned networks ANREX and find that there is no significant difference in the predictive accuracy and the size of the decision trees generated by both methods.

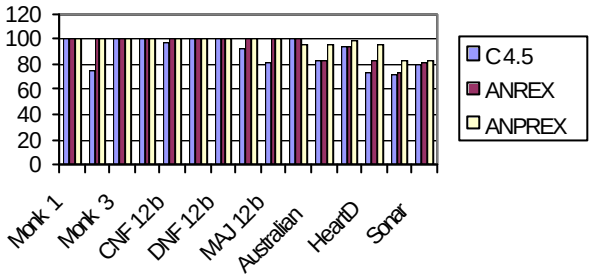


Figure1. Predictive accuracies of C4.5, ANREX and ANPREX

5. CONCLUSION

To conclude, we have presented in this paper ANPREX, A Non Pruning Algorithm for neural rule EXtraction. Our experimental results show that even though the algorithm does not perform network retraining after identifying the relevant hidden units and connections, the decision trees that **are** generates are comparable in terms of predictive accuracy and tree size to those generated by another method which requires network pruning and retraining. The algorithm employs C4.5 to generate a decision tree using the hidden unit activations as inputs. For a data set with discrete attributes, the node splitting conditions in the tree can be replaced by their equivalent symbolic rules after irrelevant input connections are removed. Simple criteria for identifying such connections are given. The criteria ensure that the removal of these connections will not affect the classification accuracy of the tree.

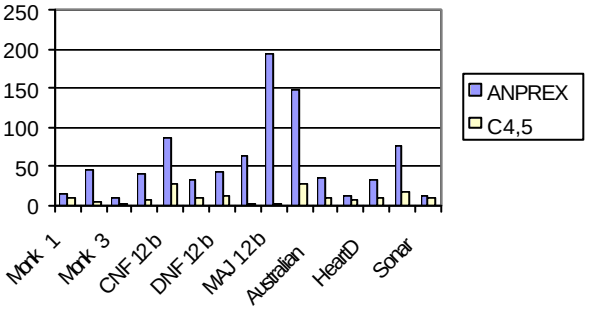


Figure 2. Tree size of ANPREX and C4.5

6 REFERENCES

[1] Andrews R., Diederich J., Tickle A.B. : Survey and critique of techniques for Extracting Rules from Trained Artificial Neural Networks. Knowledge-Based Systems, vol8, no.6, 373-389 (1995).

[2] B. hassibi and D.G. Stork, "Second order derivative for network pruning: Optimal Brain Surgeon", in Advances in Neural Information Processing Systems 5, 1993,pp. 164-171, San Mateo, CA: Morgan Kaufmann.

[3] M. Hagiwara, "A simple and effective method for re moval of hidden units and weights", eurocomputing, vol. 6, 1994, pp. 207-218.[4] S.B. Thurn, et al. The Monk's problems- a performance comparison of different learning algorithm. Preprint CMU-CS-91-97, Carnegie Mellon University, Pittsburgh.1991.

[5] A. van Ooyen, A. and B. Nienhuis, B. "Improving the convergence of the backpropagation algorithm", Machine learning, vol. 13, no1, 71-101, 1993.

[6] J. Boulahia Smirani and F. Beji . "Extraction and insertion rules during the training process of neural network". ACIDCA'2000. International Conference on Artificial and Computational Intelligence for Decision, Control and Automation in Engineering and Industrial Applications.55-60.2000.

[7] J. Boulahia Smirani. "HLS: Hybrid learning systems". Tunisian-German Conference.2000.

[8] J. hertz, A. Krogh, and R.G. palmer, Introduction to the theory of neural computation. Redwood City, CA : Addition Wesley, 1991.

[9] R.L. Watrous, "Learning algorithms for connectionist networks : Applied gradient methods for nonlinear optimization", in Proc. IEEE 1st Int. Conf. Neural Networks, San Diego, CA, 1987, pp. 619-627.

[10]C. Metz, and P. Murphy, UCI repository of machine learning databases http://www.ics.uci.edu/~mlearn/Mlrepository.html, Irvine, CA: University of California, Dept of Info and Comp. Sci.1996.

[11] R.L. Watrous, "Learning algorithms for connectionist networks: Applied gradient methods for non linear optimization," in Proc. IEEE 1st Int. Conf. Neural network, San Diego,CA,1987,pp. 619-627.

[12] J. Boulahia Smirani and F. Beji . "ANREX : An algorithm for Neural Rule EXtraction". IEEE-SMC.CESA'98 Computational Engineering in Systems Applications Tunisia.