THREE TECHNIQUES FOR EXTRACTING RULES FROM FEEDFORWARD NETWORKS

Ismail Taha and Joydeep Ghosh [1]
Department of Electrical and Computer Engineering,
University of Texas, Austin, TX 78712-1084
E-mail: {ismail,ghosh}@pine.ece.utexas.edu

ABSTRACT

Hybrid intelligent systems that combine knowledge based and artificial neural network systems typically have four phases involving domain knowledge representation, mapping into connectionist network, network training, and rule extraction respectively. The final phase is important because it can provide a trained connectionist architecture with explanation power and validate its output decisions. Moreover, it can be used to refine and maintain the initial knowledge acquired from domain experts. In this paper, we introduce three new rule extraction techniques. The first technique extracts a set of binary rules from any neural network regardless of its kind (MLP, RBF etc.,). The second technique extracts partial rules that represent the most important embedded knowledge in a trained MLP. The fidelity of the second technique is adjustable to the desired level of knowledge extraction. The third technique is a universal and comprehensive approach that extracts almost all embedded knowledge in a trained artificial neural network and represents it in a rule base format. Experimental results of implementing these three rule extraction techniques are presented.
Keyword: Knowledge Extraction. Also: Hybrid Artificial Neural Network Systems.

1 INTRODUCTION

Several researchers have investigated the design of hybrid systems that combine expert and connectionist subsystems. The main motivation of such hybrid systems is to incorporate the complementary features of knowledge based and neural network paradigms. HIA[Taha and Ghosh (1995)], KBANN[Towell and Shavlik (1994)], RAPTURE[Mahoney and Mooney (1993)] and KBCNN[Fu (1994)] are examples of these hybrid systems. The typical result is a Knowledge Based Neural Network (KBNN) system with four phases: (i) the representation phase, where initial domain knowledge is represented in a symbolic format (ii) the mapping phase, where this knowledge is mapped into an initial connectionist architecture (iii) the training phase (iv) the rule extraction phase, where the trained (and thus modified) connectionist architecture is mapped back into an updated rule-based system to provide explanation power.

Research work in the area of extracting symbolic knowledge from trained ANNs has witnessed much activity recently. KT[Fu (1994)], Subset, MofN[Towell and Shavlik (1993)], and NeuroRule[Setiono and Liu (1996)] are examples of rule extraction techniques from MLP networks. We categorize all these approaches as *Link Rule Extraction (LRE) techniques* because they all search for weighted links that cause a node (hidden or output) to be active. Then these combinations of weighted links are used to generate symbolic rules. Heuristic methods are commonly used in the *LRE* category to bound the search space for rules and to increase the comprehensibility of the extracted rules. BRAINNE, DEDEC, RuleNet and RULEX are other rule extraction techniques that extract rules from trained networks regardless the type or the structure of the neural network and hence named *Black-box Rule Extraction (BRE)* techniques. A very rich source of literature review of different rule extraction approaches can be found in[Andrews et al. (1995)].

This paper proposes three new rule extraction techniques for KBNN hybrid systems and presents their implementation results. The suitability of each technique depends on the network type, input nature, complexity, the application nature, and the required transparency level.

[1] This research was supported in part by ARO contract DAAH04-94-G-0417, DAAH04-95-10494, and ATP grant #442. Ismail Taha was also supported by the Egyptian Government Ph.D. Fellowship in Electrical and Computer Engineering.

2 RULE EXTRACTION: GAINS AND ISSUES

Extracting symbolic rules from trained ANNs is an important feature of powerful hybrid systems as it helps in: (i) Alleviating the knowledge acquisition problem and refining initial domain knowledge. (ii) Providing reasoning and explanation capabilities. (iii) Supporting cross-referencing and verification capabilities. (iv) Alleviating the catastrophic interference problem of ANNs since a different set of rules can be extracted after a network is retrained using the new environment examples. One can then examine the resulting sets of rules to find out which situation each set is suitable for. Due to these capabilities, extracting rules from trained ANNs may be essential for developing more powerful, robust, self-explanatory and self-maintained hybrid systems. Designing an efficient rule extraction module is in fact a difficult task. Based on our experience and some other reported research, we identified several factors that should be carefully considered while designing a rule extraction technique: (i) transparency of the extracted rules; (ii) granularity of the explanation feature; (iii) representation format; (iv) comprehensibility which is indicated by the number of rules and/or number of premises in each rule extracted from a trained network; (v) fidelity which is measured by the capability of the extracted rules to mimic the embedded knowledge in a trained network; (vi) Accuracy; and (vii) portability. The quality of extracted rules is improved by increasing the comprehensibility, the fidelity, and the accuracy of the rule extraction algorithm. However, to extract a comprehensive rule base from a trained ANN most, if not all, embedded knowledge in this ANN should be extracted. In this case, the comprehensibility of extracted rules will be degraded because the resulting rule base may have too many rules, some of which with many premises.

3 PROPOSED RULE EXTRACTION APPROACHES

The first proposed approach (BIO-RE) is categorized as a BRE technique, while the second and third techniques belong to the LRE category.

3.1 First Approach (BIO-RE)

It is named **Binarized Input-Output Rule Extraction (BIO-RE)** because it extracts binary rules from any neural network trained with "binary" inputs. If original inputs are not binary, they have to be binarized using Equation 1.

$$y_i = \begin{cases} 1 \text{ if } x_i \geq \mu_i \\ 0 \text{ otherwise} \end{cases} \tag{1}$$

where: x_i is the value of original input X_i, μ_i is the mean value of X_i, and y_i is the corresponding binarized input value of x_i. BIO-RE extracts rules based on the trained network input-output mapping. Some of the unique feature of BIO-RE are: (i) It does not require any information about the internal structure of the network. (ii) It can be used to extract rules from any kind of neural network (e.g. RBFs, MLPs, or Recurrent Networks). (iii) It does not require any specific training regime (supervised or unsupervised). Rules extracted by BIO-RE are represented in this format: If $[Not]$ *Input-Variable* $[And$ $[Not]$ *Input-Variable*$]^* \longrightarrow Consequent_j$, where: $[\cdot]$ is an optional term and the $[\cdot]^*$ means that the term $[\cdot]$ can be repeated 0 or n times. In terms of final rules, an extracted rule *"If* X_1 *And Not* X_2 *Then* O_1*"* is rewritten as *"If* $X_1 \geq \mu_1$ *And* $X_2 \leq \mu_2$ *Then* O_1*"*. The outline of the BIO-RE algorithm is as follows: *For a well trained neural network do:*

1. *Obtain the network output* $O(X_i) = \{o_j(x_i) \mid o_j \in \{0,1\}\}$ *corresponding to each binary input pattern* X_i*. If the number of input nodes is* **n** *then this conceptually requires* 2^n *input patterns. However, the problem specification may remove some combinations that will not occur.*
2. *Generate a truth table by concatenating each input* X_i *of step 1 and its corresponding output decision* $O(X_i)$ *from the trained network. An output* $o_j(x_i)$ *is set to 1 if its corresponding output node is active (above a threshold), otherwise it is 0.*
3. *Generate the corresponding boolean function (represented in the previously described binary rule format) from the truth table of step 2.*

Any available boolean simplification method can be used to perform step 3 of the BIO-RE algorithm. We used Espresso[2] to generate the extracted rules. The BIO-RE approach is suitable when the input/output variables are naturally binary or when binarization does not significantly degrade the performance. Also the input size (n) should be small. Given that the above conditions are satisfied, BIO-RE has some advantages: (i) It allows the use of available logic minimization tools. (ii) It does not require any rewriting procedure since extracted rules are optimal and cannot be simplified any further. (iii) Its extracted rules do not depend on the number of layers of the trained network.

3.2 Second Approach (Partial-RE)

The idea underlying Partial-RE algorithm is that it first sorts both positive and negative incoming links to all hidden and output nodes in descending order into two different sets based on their weight values. Starting from the highest positive weight (say i), it searches for individual incoming links that can cause a node j (hidden/output) to be active regardless of other input links to this node. If such a link exists, it generates a rule: $If\ Node_i \xrightarrow{cf} Node_j$, where cf represents the measure of belief in the extracted rule and is equal to the activation value of $node_j$ (see Equation 3). If a node i was found strong enough to activate a node j, then this node is marked and cannot be used in any further combinations when checking the same node j. Partial-RE continues checking subsequent weights in the positive set until it finds one that cannot activate the current node j by itself. It is important to mention that Partial-RE assumes that all inputs are equally weighted and hence have the same effect on the next layer (i.e., have the same range of values or fall between say (0,1)). Therefore, if original inputs are not naturally normal then they need to be normalized before the training phase using Equation 2.

$$z_i = \frac{1.0}{1.0 + e^{-(\frac{x_i - \mu_i}{2.0\sigma_i})}} \tag{2}$$

where, z_i ($\in (0,1)$) is the corresponding normalized input value of the original input value x_i and σ_i is the standard deviation of input feature X_i.

If more detailed rules are required (i.e. the comprehensibility measure $p > 1$), then it starts looking for combinations of two unmarked links starting from the first (maximum) element of the positive set. This process continues until Partial-RE reaches its terminating criteria (maximum number of premises in rule $= p$). Moreover, it looks for small combinations of positive and negative links that can activate a hidden/output node. In this case extracted rules are represented as: "$If\ Node_i\ And\ Not\ Node_g \xrightarrow{cf} Node_j$", where the link between node i and j is positive and between g and j is negative. After extracting all rules, a rewriting procedure takes place. Within this rewriting procedure any premise that represents an intermediate concept (i.e a hidden unit) is replaced by the corresponding set of conjuncted input features that causes it to be active. Final rules are written in this format: "$If\ X_i \geq \mu_i\ And\ X_g \leq \mu_g \xrightarrow{cf} Consequent_j$" (see Table 2 for examples). Experimental results show that Partial-RE algorithm is suitable for large size problems, since extracting all possible rules is NP-hard and extracting only the most effective rules is a practical alternative.

3.3 Third Approach (Full-RE)

Like the Partial-RE approach, Full-RE falls in the LRE category. It is notable because: (i) It extracts rules with certainty factors from trained feedforward ANNs. (ii) It extracts all possible rules that represent the semantic interpretation of the internal structure of the trained neural network that they were extracted from. (iii) It is universal since there is no restriction on the values that any input feature can take. (iv) It is applicable to any neural network node (unit) with a monotonically increasing activation function.

After examining different possible combinations of incoming links and feed-forwarding their effect on output nodes, Full-RE generates an intermediate rule in this format:

[2]Espresso is a software package for logic design[Ruddel and Sangiovanni-Vincentelli (1985)].

$If\ [\ (c_1 \cdot X_1 + c_2 \cdot X_2 + \cdots + c_n \cdot X_n) >= (\theta_j + \Delta)] \xrightarrow{cf} Consequent_j$
where: c_i is a constant represents the effect (or the importance) of the i^{th} input (X_i) on $Consequent_j$. If node j is in the layer above node i then c_i represents the weight value w_{ji} of the link between these two nodes. Since Full-RE is capable of extracting rules from networks trained with continuous inputs, then different values of X_is that satisfy the intermediate rule, have to be computed then all possible rules can be generated. Full-RE performs this step by discretizing each input feature $X_i \in (a_i, b_i)$ into k intervals: ($X_i = \{a_i, d_{i,1}, \cdots, d_{i,k-1}, b_i\}$, where: $d_{i,l-1}$ and $d_{i,l}$ are the lower and upper boundary values of interval l of input X_i respectively. After extracting its intermediate rules, Full-RE uses The Chi2[Liu and Setiono (1995)] algorithm[3], a powerful discretization tool, to compute discretization boundaries of input features. If Full-RE finds more than one discretization value of an input X_i that can satisfy the intermediate equation (i.e., the equation has more one feasible solution) then it chooses the minimum of these values if the corresponding effect parameter c_i is positive, or it chooses the maximum of them if X_i has a negative effect parameter c_i^-. The decision of selecting the minimum or the maximum discretization value of an input X_i is taken based on the sign of its corresponding effect parameters c_i of the intermediate rule. However, all selected discretization values should satisfy the intermediate equation and the boundary constraints of all input features. The Full-RE method can be summarized in the following steps: *For each hidden node j in a well trained MLP do:*

1. *Consider the equation $\sum_{i=1}^{n} w_{ji} \cdot x_i \geq \lambda_j$, where λ_j is high enough to make node j active. If activation function of node j is sigmoid then $\lambda_j = \theta_j - \ln(\frac{1}{\Delta} - 1)$.*
2. *Given the discretization boundaries of each input feature, consider the Linear Programming problem of* Minimizing $w_{j1}X_1 + w_{j2}X_2 + \cdots + w_{jn}X_n$ *such that:* $w_{j1}X_1 + w_{j2}X_2 + \cdots + w_{jn}X_n \geq \lambda_j$, *and* $X_i = \{a_i, d_{i,1}, \cdots, d_{i,k-1}, b_i\}\ \forall_i, i = 1 \cdots n$

Full-RE solves this LP problem by examining the discretization boundaries of X_is that determine the feasible solution(s) and satisfy all given constraints. Note that: any linear programming tools can also be used to solve this standard LP problem[4]. Full-RE computes certainty factors of extracted rules based on[5] Equation 3.

$$cf = \begin{cases} \frac{1}{1+\exp^{(\sum_{i=1}^{n} w_{ji} \cdot x_i - \theta_j - \Delta)}} & \text{if act(j) is a sigmoid} \\ \sum_{i=1}^{n} w_{ji} \cdot x_i - \theta_j - \Delta & \text{if act(j) is a linear threshold} \\ 1 & \text{if act(j) is a hard limiting and } \sum_{i=1}^{n} w_{ji} \cdot x_i \geq 0.5 + \Delta \\ 0 & \text{if act(j) is a hard limiting and } \sum_{i=1}^{n} w_{ji} \cdot x_i \leq 0.5 + \Delta \end{cases} \tag{3}$$

Since, activation values of hidden nodes are bounded between (0,1), Full-RE uses a simplified version of the above procedure to extract rules between hidden and output nodes where discretizing the outputs from hidden nodes is no longer required. In each rule extracted between a hidden and output node, Full-RE replaces each hidden node (h_j) by the left hand side of the rule(s) whose right hand side is h_j. Final rules extracted by Full-RE are represented in the same format of Partial-RE expect that each μ_i is replaced by one of the discretization boundaries (say $d_{i,l}$) selected by Full-RE (see Table 3). Note that there is no restriction on the number of premises in the final rules extracted by the Full-RE.

4 IMPLEMENTATION

We applied all three rule extraction techniques to two problems: (i) an artificial rule-based system which has six rules relating four binary inputs and four binary outputs. (ii) Iris database, a simple classification problem which contains 50 examples each of three classes. These 150 instances were divides into two subsets, the first subset, used for training, is of size

[3]We are thankful to Liu and Setiono for making their Chi2 source code available to us.
[4]We also used Mathematica to find out feasible solutions.
[5]Full-RE only generates a rule if its cf computed by Eqn. 3 is ≥ 0.5.

Table 1. Rules extracted by BIO-RE technique.

R#	Rule Body	Class	Times Correct	Times Misfired
1	If $I_2 \leq 3.0$ $I_3 \geq 3.7$ and $I_4 \leq 1.2$	Versicolor	10/50	0/150
2	If $I_1 \leq 5.8$ and $I_3 \leq 3.7$ and $I_4 \leq 1.2$	Setosa	50/50	5/150
3	If $I_1 \geq 5.8$ and $I_3 \geq 3.7$ and $I_4 \geq 1.2$	Virginica	47/50	24/150
4	If $I_1 \leq 5.8$ and $I_2 \leq 3.0$ and $I_4 \geq 1.2$	Versicolor	11/50	3/150

Table 2. Rules extracted by Partial-RE technique.

R#	Rule Body	Class	CF	Correct	Misfired
1	If $I_2 \geq 3.0$ and $I_3 \leq 3.7$	Setosa	0.98	48/50	0/150
2	If $I_1 \geq 5.8$ and $I_3 \geq 3.7$ and $I_4 \geq 1.2$	Virginica	0.99	47/50	27/150
3	If $I_1 \leq 5.8$ and $I_2 \leq 3.0$ and $I_3 \geq 3.7$	Versicolor	0.74	16/50	3/150
4	If $I_1 \leq 5.8$ and $I_2 \leq 3.0$ and $I_4 \geq 1.2$	Versicolor	0.72	1/50	0/150
6	Default Class	Versicolor	1.0	6/50	2/150

Table 3. Rules extracted by Full-RE technique.

R#	Rule Body	Class	CF	Times Correct	Times Misfired
1	If $I_3 \leq 2.1$	Setosa	0.99	50/50	0/150
2	If $I_3 \leq 5.1$ and $I_4 \leq 1.7$	Versicolor	0.97	49/50	3/150
3	If $I_3 \geq 4.8$	Virginica	0.98	47/50	1/150

89 and the second is of size 61 and used for testing. Each input pattern has four continuous input features: I_1 = Sepal-length , I_2 = Sepal-width, I_3 = Petal-length, and I_4 =Petal-width. In both experiments, an MLP network was trained using the backpropagation algorithm with momentum term and regularization term P which contributes to the weight update term in the backpropagation equation by $\frac{-2*\lambda*w_{jk}*{w_0}^2}{{w_0}^2+w_{jk}^2}$ [Ghosh and Tumer (1994)].

4.1 An Artificial Binary Problem

The original rules were as follows:

Rule #1: $If\ A\ And\ B \xrightarrow{0.8} O_1$ *Rule #2:* $If\ B\ And\ C\ And\ D \xrightarrow{0.7} O_2$
Rule #3: $If\ Not\ C \xrightarrow{0.6} O_3$ *Rule #4:* $If\ Not\ A\ And\ D \xrightarrow{0.7} O_4$
Rule #5: $If\ B\ And\ D \xrightarrow{0.7} O_1$ *Rule #6:* $If\ D \xrightarrow{0.8} O_1$

These six rules were mapped by the Node Links Algorithm [Taha and Ghosh (1995)], into a network with 4 input nodes, 6 hidden nodes, and 4 output nodes. **The first experiment:** is designed to check the capability of the three techniques of extracting the original rules from the mapped network. Therefore, the network was not trained before extraction procedures were applied. (a) BIO-RE extracts the same set of binary rules but without certainty factors. (b) Partial-RE with $p = 2$ (i.e maximum 2 conditions per rule) extracts all 5 original rules with two conditions or less. On increasing p to 3, rule#2 was also extracted. The certainty factors attached to each output decision were approximately the same as the original rules. (c) Full-RE extracts the same six original rules from the same untrained network. **The second Experiment:** based on the original rules, 2^4 binary patterns were generated. After training the previous network, we applied the three approaches to the final network architecture (i.e., the adapted one): (a) Both BIO-RE and Full-RE extract the same six original rules. (b) Partial-RE extracts all the six rules plus an extra one: Rule #7: $If\ B$ And $D \xrightarrow{0.74} O_2$. This rule was extracted when $p = 3$.

4.2 Iris Classification

For the Iris classification problem, we used an MLP with 4 input, 6 hidden, and 3 output nodes. Since input features of iris are continuous and both BIO-RE and Partial-RE extracts rules from networks with binary/binarized and normalized inputs respectively, we used a binarized and a normalized version of the iris data for training and testing two network architectures both with the same number of nodes as stated above. Full-RE extracts rules from the same network structure trained with the original iris data set. Note that there is no prior knowledge for the Iris data set, so the networks were initialized randomly and then trained on samples from the data set. Table 1, 2, and 3 present the rules extracted by BIO-RE, Partial-RE, and Full-RE techniques respectively from their corresponding networks trained on the Iris data set. The performance of these corresponding trained networks are 72.67%, 93.33%, and 97.33% respectively. The tables also present the correct classification and the misclassification for each extracted rule. Note that Binarizing and normalizing continuous inputs may degrade the accuracy of the extracted rules (see Table 1 and 2) as well as the generalization of the corresponding trained networks.

5 CONCLUSION

In this paper we introduced three new rule extraction approaches. The suitability of each approach depends on the network type, inputs, complexity, the application nature, and the required transparency level. Experimental result shows that rules extracted by the three approaches are sound. All three methods are able to extract meaningful rules for the well known Iris database, where no pre-existing rules are available. Also, it shows that BIO-RE is suitable for small networks with binary inputs, both BIO-RE and Partial-RE are able to extract "certain" rules, and Full-RE is a universal rule extraction technique that can extract rules from MLPs with any kind of input (continuous, normal, binary). The rules extracted by Full-RE are more comprehensive than those extracted by the other two techniques. It is important to mention that extracting all possible rules is NP-hard and a feasible alternative is often to extract the key rules that cover most of the key concepts of the application domain. More progress is needed in determining when an adequate set of rules has been extracted. Another important issue that needs to be investigated is how the outputs of both the rule extraction and the trained ANN modules can be integrated to provide more robust decisions.

REFERENCES

Andrews, R., Diederich, J., Tickle, A., 1995, "A Survey And Critique Of Techniques For Extracting Rules From Trained Artificial Neural Networks," Neurocomputing Research Center, Queensland University of Technology, Queensland, Australia.

Fu, L., 1994, *Neural Networks in Computer Intelligence* McGraw-Hill, Inc.

Ghosh, J., Tumer, K., 1994, "Structural Adaptation and Generalization in Supervised Feed-Forward Networks," *Journal of Artificial Neural Networks*, Vol. 1, pp. 431–458.

Liu, H., Setiono, R., 1995, "Chi2: Feature Selection and Discretization of Numeric attributes," Proc. Proceedings of the Seventh International Conference on Tools with Artificial Intelligence, pp. 388–391.

Mahoney, J., Mooney, R., 1993, "Combining Connectionist and Symbolic Learning to Refine Certainty Factor Rule Bases," *Connection Science*, Vol. 5, pp. 339–364.

Ruddel, R., Sangiovanni-Vincentelli, A., 1985, "Espresso-MV: Algorithms for Multiple-Valued Logic Minimization," Proc. Proceedings of Cust. Int. Circ. Conf. Portland.

Setiono, R., Liu, H., 1996, "Symbolic Representation of Neural Networks," *IEEE Computer*, Vol., pp. 71–77.

Taha, I., Ghosh, J., 1995, "Controlling Water Reservoirs using A Hybrid Intelligent Architecture," Proc. Intelligent Engineering Systems Through Artificial Neural Networks, Vol. 5, pp. 63–68. ASME Press.

Towell, G., Shavlik, J., 1993, "The Extraction of Refined Rules from Knowledge-Based Neural Networks," *Machine Learning*, Vol. 13, pp. 71–101.

Towell, G., Shavlik, J., 1994, "Knowledge-Based Artificial Neural Networks," *Artificial Intelligence*, Vol. 70, pp. 119–165.