International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:4, 2008

An Efficient Technique for Extracting Fuzzy Rules from Neural Networks

Besa Muslimi, Miriam A. M. Capretz, and Jagath Samarabandu

***Abstract*—**Artificial neural networks (ANN) have the ability to model input-output relationships from processing raw data. This characteristic makes them invaluable in industry domains where such knowledge is scarce at best. In the recent decades, in order to overcome the black-box characteristic of ANNs, researchers have attempted to extract the knowledge embedded within ANNs in the form of rules that can be used in inference systems. This paper presents a new technique that is able to extract a small set of rules from a two-layer ANN. The extracted rules yield high classification accuracy when implemented within a fuzzy inference system. The technique targets industry domains that possess less complex problems for which no expert knowledge exists and for which a simpler solution is preferred to a complex one. The proposed technique is more efficient, simple, and applicable than most of the previously proposed techniques.

***Keywords*—**fuzzy rule extraction, fuzzy systems, knowledge acquisition, pattern recognition, artificial neural networks.

I. INTRODUCTION

ARTIFICIAL neural networks (ANN) are low-level, parallel-processing computational structures that have been proven to be universal approximators [6]; that is, they have the ability to learn to approximate any input-output relationship from simply processing raw data. In addition, neural networks can tolerate incomplete or noisy inputs. These characteristics make artificial neural networks useful in domains where the input-output relationship of a system is unknown or difficult to model with regression techniques. However, their applicability is often hindered by the "black box" nature of ANNs, as the reasoning the network uses to determine an output is impossible to trace.

In 1988, Gallant made one of the first attempts to make neural networks more comprehensible [4]. This was done by accompanying each output determined by the network with a rule that summarized the reasoning behind the output. Other researchers quickly followed, and many concentrated on extracting rules that summarize the knowledge embedded within the architecture and weights of a trained neural network. These rules could then be used to develop an expert inference system. In fact, some research suggests that the extracted rule set can sometimes outperform the generalization of the trained artificial neural network from which the rules were extracted [1]. During the course of the past couple of decades, research has spread in several directions: extracting rules by analyzing individual neurons, extracting rules by observing overall network behavior, and refining a rule base through neural networks. With the publication of Buckley et al.'s proof of equivalence between artificial neural networks and fuzzy inference systems [2], research also expanded to include fuzzy rule extraction from trained artificial neural networks, [5], [8], [12]. A fuzzy logic system is a mathematical inference model that allows for a problem to be described in high-level linguistic terms and deals well with uncertainty. However, fuzzy inference systems can only be built if the input-output relationship is known or can be acquired from domain experts. Extracting fuzzy rules from a trained neural network offers the advantage of being able to build a fuzzy system, which is transparent to the user, even when domain expert knowledge is unavailable.

To date, some of the most popular and prominent rule extraction techniques are Fu's KT algorithm [3] and Towell and Shavlik's M-of-N method [15]. Many other techniques have also been developed, however most of them have been developed to extract rules from a multilayered neural network used for complex problems. As a result, they tend to be very complex and inefficient, they often require special neural network architectures or apriori knowledge, and they often generate a very large number of not-so-comprehensible rules. Yet there are many industrial domains where the problems are somewhat simple and no apriori knowledge exists. Furthermore, there are industrial domains that prefer to use artificial intelligence systems as verification tools that aid domain experts in their decisions. For such domains, a set of concise and general rules that describe the input-output relationship and that are easily verifiable by a domain expert is sufficient. Also, in such domains, complexity and inefficiency are vastly unfavorable.

In this paper, a new method for extracting rules from a two-layer neural network is introduced, one which is far simpler and more efficient than the current methods. The technique can be implemented on the two-layer perceptron and it extracts a small set of general and concise rules that are able to accurately describe the input-output relationship. The rule extraction method proposed in this paper is based on the method described by Huang and Xing [7] and can extract rules far more accurately and robustly.

The paper is organized as follows: In section 2 of this paper, an explanation of the rule extraction problem is given. Section

This work was supported in part by Canada's National Science and Engineering Research Council Industrial Postgraduate Scholarship.

Besa Muslimi is a graduate student in the Department of Electrical and Computer Engineering at The University of Western Ontario, London, ON, Canada (bmuslimi@uwo.ca).

Miriam A. M. Capretz is an associate professor in the Department of Electrical and Computer Engineering at The University of Western Ontario, London, ON, Canada (mcapretz@eng.uwo.ca).

Jagath Samarabandu is an assistant professor in the Department of Electrical and Computer Engineering at The University of Western Ontario, London, ON, Canada (jagath@uwo.ca).

3 contains a summary of the existing rule extraction approaches. Section 4 introduces the new algorithm and the details of how it works. Finally, section 5 contains the experimental results of comparing existing methods with the new algorithm and an analysis of the results, followed by the conclusions in section 6.

II. THE RULE EXTRACTION PROBLEM

As stated before, research for extracting rules from a trained artificial neural network has spread in many directions. The main classification scheme for such algorithms is in the way they extract rules. Based on this criteria, an algorithm can be described as decompositional, pedagogical, or eclectic. In the decompositional approach, each hidden and output node is analyzed individually, and a rule is extracted from it [1]. In the pedagogical approach, the overall behavior of the trained ANN is observed in order to extract rules that describe the input-output function [1]. The eclectic approach is a combination of the first two approaches [1].
The proposed algorithm falls in the category of decompositional algorithms and in the next section we examine this class of algorithms as applied to feed-forward neural networks.

In a feed-forward neural network, the output of each neuron is calculated as:

$$A_j = Act\left(\left(\sum_i w_{ij} \times o_i\right) + \theta_j\right) \quad (1)$$

where

$$Act(x) = \frac{1}{1+e^{-\alpha x}}. \quad (2)$$

In (1), A_j is the activation of neuron j, w_{ij} is the weight on the link from neuron i to neuron j, o_i is the activation of neuron i, θ_j is the bias on the neuron j, and the activation function *Act()* is usually modeled as the sigmoidal function, as shown in (2), where α is a parameter controlling the steepness of the sigmoidal function, which approximates the step function.
The most important characteristic of the decompositional approach is that all neurons in the ANN have activations of approximately 0 or 1. This ensures that the links that are incoming to a neuron carry a signal that is equal to the size of the weight or zero. In the hidden layer neurons, binary inputs allow for this to happen. In the output layer neurons, this is made possible by increasing the α parameter of the activation functions (to approximately 10) of the hidden neurons, to ensure that the neurons approximate boolean behavior.
Decompositional approaches extract rules from each of the hidden and output layer neurons by finding the combination of the incoming weights whose sum exceeds the bias of the neuron. Rules extracted from the hidden layer neurons and the output layer neurons are then combined to create rules describing characteristics of the input-output relationship.

III. RELATED WORKS

In the past two decades, many different decompositional rule-extracting algorithms have been proposed. Fu proposed the KT algorithm, where, for each hidden and output neuron, it searches for a single link with a large enough weight to exceed the bias of the neuron [3]. If such a link is found, a rule is written. Next, the algorithm searches for subsets of two links that exceed the bias, followed by a subset of three, and so on. When all the neurons have been searched, rules extracted from the hidden layer neurons are combined with rules extracted from the output layer neurons to create input-output relationship rules and rules subsumed by the more general rules are eliminated.
The search space is reduced by restraining the activation of every node of the network to the interval [0,1], which allows for the assumption that negatively weighted links can only give rise to negated antecedents and positively weighted links can only give rise to non-negated antecedents [3]. Fu also constrains the number of antecedents in a rule and uses three heuristics to further reduce the search space and the number of extracted rules [3]. However, in spite of all this, the algorithm is still of exponential complexity [3]. In addition, imposing a maximum number of antecedents in a rule can significantly affect the quality of the rule set [1].
Towell and Shavlik present a similar algorithm [15] that is implemented on a special multilayer network developed by them [16] called the knowledge-based neural network (KBNN). The existing knowledge about the domain is first inserted into the architecture of the network and the network is trained with the backpropagation algorithm. Then links with similar weights are combined into clusters and the average of the cluster's weight is used as the weight of each link belonging to that cluster. Clusters with low link weights (relative to the rest of the clusters) and few members are then eliminated as they are assumed to have little influence on the outcome of the network. The weights of the links are then fixed and the network is retrained with the backpropagation algorithm to adapt the biases of the network. Finally, a rule is written for each hidden unit and output unit in the form of

If M of N antecedents are TRUE then C

where each antecedent is associated with a weight and the rule is associated with a threshold, given by the bias. The Final step of the algorithm involves simplifying the rules by removing the weights of the antecedents and the thresholds of the rules where possible. There are several shortcomings to the M-of-N algorithm: first of all, it is implemented on a KBNN as opposed to a standard multilayer perceptron, and it requires a clustering algorithm. These two requirements do not allow for the algorithm to be portable across different ANN architectures. Second, the primary goal of the M-of-N algorithm is to refine rules contained in the initial rule base. This limits its use to domains where the input-output relationship knowledge exists. Also, it does not allow for new and unexpected knowledge to be discovered. Finally, the

complexity of the M-of-N algorithm is exponential, although the authors argue that it is approximately cubic [15].

Krishnan et al. provide another technique for extracting rules from feed-forward neural network [10]. The main idea is to sort the incoming weights of a neuron in descending order, then create combinations of weights of all possible sizes and order the combinations in descending order based on the sum of the weights in the combination. Their main goal is to reduce the search space by checking first if the combinations that yield the highest sum of weights exceed the bias. If these combinations fail, the subsumed combinations can be pruned without being checked. Despite the effort however, the algorithm is still of exponential complexity [10].

While the above algorithms extract crisp rules and only deal with discrete inputs, other researchers have proposed algorithms for extracting fuzzy rules or systems that deal with fuzzy inputs. Hayashi and Imura proposed a fuzzy neural expert system with automated extraction of fuzzy rules that can handle fuzzy and crisp inputs [5]. In addition, each extracted rule is associated with a fuzzy truth value such as *Very True* or *Possibly True*, and each antecedent in a rule is associated with a fuzzy importance value such as *Very Important* or *Moderately Important*. However the accuracy of the system is only 75.5% [5].

Kasabov's REFuNN algorithm [8],[9], applied to the specially constructed FuNN fuzzy neural network also has the ability to extract fuzzy weighted rules as well as simple fuzzy rules; however, the number of rules extracted for a fairly simple problem such as the Iris classification data [14] is very large.

NEFCLASS (Neuro Fuzzy CLASSification) is a neuro-fuzzy system for the classification of data that is presented by Nauck and Kruse in [12], [13]. The goal of the system is to learn fuzzy rules from the training data patterns as it classifies each pattern into crisp classes. Although the system performs well (96.67% accuracy), the rules are never tested on a fuzzy inference system and the system does not perform any better than the standard three layer perceptron. In addition, a ceiling is placed on the maximum number of rules extractable, a constraint that could seriously hinder the quality of the rule set.

As mentioned in the introduction, above techniques attempt to extract rules for complex problems and therefore are complex, inefficient, and not general enough to be applied across different domains. Huang and Xing presented a new technique for extracting rules that can be used for domains with simpler problems [7]. Their technique consists of several steps. In the first step, given n-continuous-valued input parameters I_i, i=1,2,…,n, each input parameter is classified into two or more equally populated sets. Then each set is represented with a binary scheme. For example, if each input parameter is divided into two sets, small and large, the set {I_i is small} is represented as [1 0], and the set {I_i is large} is represented as [0 1]. As a result, a problem of n continuous-valued input parameters is transformed into a problem of $2n$ input parameters where each input is binary.

Next, a two layer feed-forward back-propagation neural network is constructed, with $2n$ inputs and as many output nodes as the number of classes in the data. Once the network is trained, the most dominant rule from each output neuron is extracted. This is done by determining, for each input I_i, its binary input with the highest weight and assuming that input to be 1. Therefore, the antecedents of the extracted rule include all input parameters, some of which can then be pruned. The pruning process allows for the rule to be more general and therefore yield more accurate results. The pruning algorithm first sorts the input parameters in ascending order of their maximum weights. Then, the algorithm prunes the parameters one at a time, starting with the input parameter with the smallest maximum weight, so long as the neuron remains activated even when the maximum-weight binary input of the input parameter is off and the minimum-weight binary input is on.

The main problem with this rule-extraction technique lies in the pruning process:

It assumes that how an input parameter affects the activation of an output neuron depends only on the maximum weight of the parameter, and not on the minimum weight. This incorrect assumption causes antecedents that can be pruned to sometimes escape pruning, making the rules less general, and consequently diminishing the accuracy of the fuzzy inference system implementing the extracted rules.

IV. A NEW RULE-EXTRACTING ALGORITHM

In this section we describe the proposed rule extraction algorithm that can overcome the above disadvantages; We also show that this algorithm can extract rules that yield much higher accuracy and robustness.

Fig. 1 shows a simplified neural network with one output node and six binary inputs, representing the three continuous-valued input parameters, I_1, I_2, and I_3.

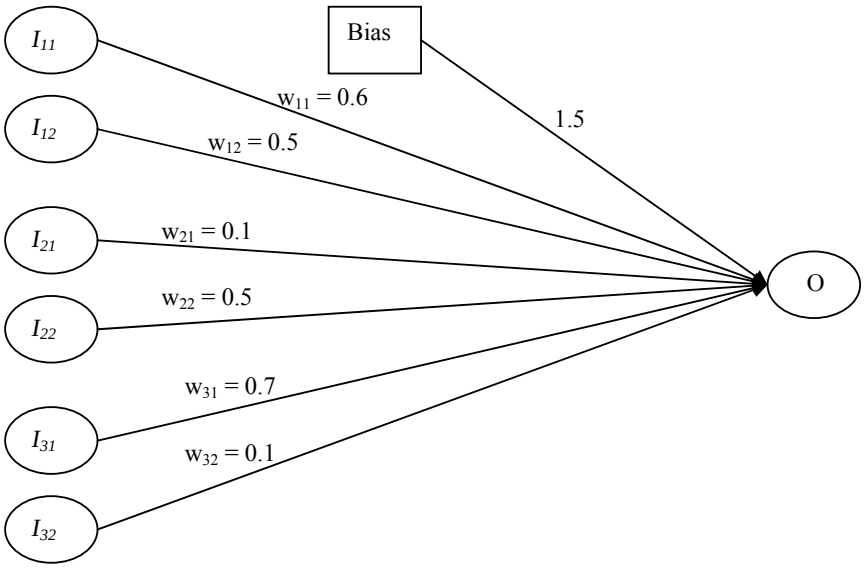

Fig. 1 A neural network with 3 input parameters, two Boolean inputs for each input parameter, and one output.

It should be noted that the binary inputs within each input parameter group are dependent. For example, in Fig. 1, if the value of parameter I_1 is "small" then I_{11} will be 1 and I_{12} must be 0.

To extract the most dominant rule for the output node, the maximum weight w_{im_i} of each input parameter I_i is determined:

$w_{1m_1} = 0.6$, $w_{2m_2} = 0.5$, and $w_{3m_3} = 0.7$

Therefore, the extracted rule is:
If I_1 is I_{12} AND I_2 is I_{21} AND I_3 is I_{32} then *Output* is *O*.

In order to prune all the antecedents of a rule that do not affect the activation of the output neuron, for each input parameter I_i, the absolute difference between the maximum weight w_{im_i} and the minimum weight w_{il_i} of its binary inputs is calculated. The input parameters are then sorted in ascending order of this absolute difference. Finally, the algorithm prunes each input parameter, starting with the one with the smallest absolute difference, so long as the neuron remains activated if its maximum-weight binary input is off and the minimum-weight binary input is on. Pseudo code for the pruning algorithm is as follows:

For $i = 1, 2, \ldots, n$

Find l_i such that $w_{il_i} = \min[w_{ij}]$, $j = 1,2,3$.

Find $d_i = w_{im_i} - w_{il_i}$

Sort I_i in ascending order of $\boldsymbol{d_i}$

Let $S = \sum_{i=1}^{n} w_{im_i} - B$

For $i = 1, 2, \ldots, n$ (note that now $\boldsymbol{d_i}$ are sorted)

Let $S = S - w_{im_i} + w_{il_i}$

If S<0

Exit

Else

Remove antecedent that involves parameter I_i

For the network shown in Fig. 1, the rule antecedents would be pruned as follows according to the new algorithm:

Find w_{im_i}, w_{il_i} and $\boldsymbol{d_i}$ for each input parameter:

$w_{1m_1} = 0.6$, $w_{2m_2} = 0.5$, and $w_{3m_3} = 0.7$

$w_{1l_1} = 0.5$, $w_{2l_2} = 0.1$, $w_{3l_3} = 0.1$

$\boldsymbol{d_1} = 0.6 - 0.5 = 0.1$, $\boldsymbol{d_2} = 0.5 - 0.1 = 0.4$, $\boldsymbol{d_3} = 0.7 - 0.1 = 0.6$

Sort I_i in ascending order of $\boldsymbol{d_i}$: I_1 I_2 I_3
S = 0.6 + 0.5 + 0.7 – 1.5 = 0.3
For $i = 1$
S = 0.3 – 0.6 + 0.5 = 0.2
0.2>0
Remove antecedent that involves parameter I_1
For $i = 2$
S = 0.2 – 0.5 + 0.1 = -0.2
-0.2<0, Exit

So the rule from the improved pruning algorithm is:
If I_2 is I_{21} AND I_3 is I_{32} then *Output* is *O*.

V. EXPERIMENTAL RESULTS & ANALYSIS

5.1 Experimental Results

The new algorithm proposed in this paper targets the problem of efficiently extracting a small set of general rules that yield

high accuracy. In order to demonstrate the accuracy of the new algorithm, we compare it against existing algorithms using the Iris classification data set [14]. The data set contains 3 classes (Iris-Virginica, Iris-Versicolor, or Iris-Setosa) where each class refers to a type of iris plant. There are 150 data records, 50 for each type of class. The Iris-Setosa class is linearly separable from the Iris-Versicolor and Iris-Virginica classes but the latter two are not linearly separable from each other. The classification problem consists of finding which class an Iris plant belongs to based on the four continuous-valued input parameters: sepal length, sepal width, petal length, and petal width.

For the experiment, each input parameter of the Iris data set was separated into 3 sets: small, medium, and large. The data was then converted to binary data.. In order to better measure the accuracy of each algorithm, testing was automated using a program developed for this purpose.

For the experiments, two parameters were varied: data used to train the networks and the shape of the membership functions of the fuzzy inference systems in which the extracted rules were implemented. The network training data varied between all the data points in the Iris data set and only the data points in the data set that did not introduce a nonlinear separation between the Iris-Versicolor and the Iris-Virginica classes (from hereon referred to as the filtered Iris data set). The shape of the membership functions used in the fuzzy systems varied between triangular and generalized bell. The variations of these two parameters created four test cases. In the first test case (All_Bell) the neural network was trained with all the data in the Iris data set and the extracted rules were implemented in a fuzzy inference system with bell membership functions. In the second test case (All_Triangular), the network was also trained with all the data, but the extracted rules were implemented in a fuzzy system with triangular membership functions. Likewise, for the third and fourth test cases, the networks were trained with the filtered data set, and in the third test case (Filtered_Bell), the extracted rules were implemented on a fuzzy system with bell membership functions while in the fourth test case (Filtered_Triangular) they were implemented in a fuzzy system with triangular membership functions.

One thousand tests were conducted for each test case, totaling to four thousand tests, for which four thousand two-layer feedforward networks were trained using the back-propagation algorithm. Of these, the networks that performed poorly (achieved a mean square error measurement of greater than 0.05) were removed. This is because it is obvious that extracting rules from networks with poor performance will yield poor rules and in practice such networks would not be used to extract rules. Once such networks were removed from the experimental data, each test case comprised of anywhere between 775 tests and 835 tests. The average accuracy results of each test case are shown in Table I, together with the average variance. The average classification accuracy and variance of the network is also shown in Table I.

TABLE I
EXPERIMENTAL RESULTS FOR THE IRIS DATA SET

Test Case	Average Network Accuracy	Average Network Variance	Average Accuracy of the Original Algorithm's Rules	Average Variance of the Original Algorithm's Rules	Average Accuracy of the New Algorithm's Rules	Average Variance of the New Algorithm's Rules
All_Bell	0.97	0.000049	0.76	0.014	0.84	0.0038
All_Triangular	0.97	0.000049	0.77	0.015	0.84	0.0039
Filtered_Bell	0.99	0.00012	0.80	0.0091	0.85	0.0014
Filtered_Triangular	0.99	0.00012	0.80	0.011	0.86	0.0010
AVERAGE	0.018	0.000083	0.78	0.12	0.85	0.0025

As it can be seen from Table I, the new algorithm performs much better than the original algorithm. The average accuracy of the new algorithm is 85%, as shown in the last row of Table I. The average accuracy of the original algorithm is 78%. In addition, the variance in accuracy is lower for the new algorithm by a factor of 5: The average variance in accuracy of the rules extracted by the original algorithm [7] is 0.012, while the average variance accuracy of the rules extracted by the new algorithm is 0.0025. The lower variance of the new algorithm illustrate that the new algorithm is not only more accurate, but also more reliable in its accuracy.

The fact that the new algorithm performs with almost the same accuracy whether the data is filtered or not, shows its robustness: the algorithm is capable of learning the most general and accurate rules even when bad data points are present in the training set. In practice, this has a great impact on the applicability of the algorithm: Most industry domains are not aware which data points introduce the nonlinearity in the data.

The robustness of the proposed algorithm is also shown by the fact that the accuracy of the extracted rules accuracy does not vary with the shape of membership functions used. In fact, no matter which data set the networks are trained with, the average accuracy difference between fuzzy systems with triangular membership functions and those with bell membership functions is less than 0.4%.

From Table I, we can also compare how each algorithm performs in comparison to the trained neural networks. This is an important comparison because if only a small accuracy drop exists between the performance of the network and the performance of extracted rules, then this small drop is often a small price to pay given the traceability and transparency gained by using a fuzzy inference system instead of a neural network. In our experiments, we found that the networks trained with all the data achieve 97% accuracy. In comparison, when the rules extracted from it using the new algorithm, are implemented in a fuzzy inference system, they yield an accuracy of 84%, regardless of the type of membership functions used. Therefore, a 13% drop in

accuracy occurs. On the other hand, when the rules extracted by the existing algorithm [7] are implemented in a fuzzy inference system, they achieve an accuracy of 76% and 77% (depending on the type of membership function used). Thus, an accuracy drop of 21% and 20% occurs. This difference illustrates the superiority of the new algorithm.

Similar results are obtained from comparing the accuracy of the networks trained with the filtered Iris data set and the rules extracted from it by each of the two algorithms. The average accuracy that the networks achieve is 99%, whereas the rules extracted from them with the new algorithm achieve an accuracy of 85% and 86% (depending on the type of membership function used) resulting in an average accuracy drop of 13.5%. In contrast, the rules extracted from the networks using the original algorithm [7] yield an accuracy of 80%. Once again, the accuracy drop of 19% versus the 13.5% demonstrates the pre-eminence of the new algorithm.

5.2 Analysis of Experimental Results

The main goal of a rule-extraction algorithm is to extract as few, maximally general rules as possible that provide high classification accuracy. Our algorithm possesses this ability because its pruning algorithm ensures that only the most critical antecedents are kept in the rule and the rest are pruned. This characteristic guarantees that each rule is in its most general form and therefore it can be applied to most data points. Thus, the proposed algorithm finds for each class, the characteristics that most often distinguish that class from the other classes and translates it into a rule. It is also this ability that instigates the robustness of the rules extracted by the new algorithm. Because the algorithm is able to extract only the most general rules, a few bad data points that introduce nonlinearity among the output classes do not drastically affect the performance of the algorithm. In addition, because the algorithm is able to extract only the most dominant rules that describe the input-output relationships, the accuracy is not affected by the type of the membership function shape used in the fuzzy inference system within which the extracted rules are implemented.

Finally, it can be seen that the computational complexity of the proposed algorithm is O(n), where n is the number of input parameters. This is very low compared to the exponential computational complexity that most rule extracting algorithms possess, making the new algorithm much more efficient.

VI. CONCLUSION

While artificial neural networks possess the invaluable ability to learn input-output relationships from simply processing data, they also exhibit black-box behavior, a characteristic which often hinders their applicability.

As early as 1988 [4], researchers have tried to translate the knowledge embedded within an artificial neural network into comprehensible rules that could be used in an expert or fuzzy inference system. Much of this research has concentrated on extracting knowledge from multilayered neural networks that can solve complex problems. However, the solutions yielded from such research are often computationally inefficient, logically complex, and often impose specific restrictions such as required *a-priori* knowledge. Such characteristics hinder their applicability in industry domains where the problems are less complicated and therefore solution simplicity, efficiency, and generality take precedence.

In this paper we have presented an algorithm for extracting fuzzy rules from a two-layer feed-forward neural network. This algorithm is able to extract general rules that describe the input-output relationships with high accuracy. The rules extracted are also robust, as the accuracy of the fuzzy inference system in which the rules are implemented does not drastically vary with the type of membership function used. In addition, the algorithm exhibits the time complexity of O(n), where n is the number of input parameters. This is much lower than the exponential complexity that most rule-extracting algorithms exhibit. Finally, the rule-extracting technique presented in this paper imposes no preconditions for applicability, making the technique appropriate for most industry domains seeking a solution to a problem that is not complex but much about it is unknown.

Future work in this direction includes modifying the algorithm so that it is applicable to multi-layer neural networks. This can be achieved by increasing the slope of the sigmoid function of the hidden layer neurons so that their outputs approximate the binary activation function. Also, instead of dividing the data into equally-populated sets, a clustering method could be applied to more accurately divide the data into fuzzy sets. Finally, a neuro-fuzzy system could be used to implement the rules extracted, so that the membership functions used can be fine-tuned, leading to higher overall accuracy.

REFERENCES

[1] Andrews, R. Diederich, J., Tickle, A. Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks. *Knowledge-Based Systems*, Volume 8, Number 6, pp. 373-389. December 1995.

[2] Buckley, J.J., Hayashi, Y., Czogala, E. On the Equivalence of Neural Nets and Fuzzy Expert Systems. *Fuzzy Sets Systems*, vol. 53, no. 2, pp. 129-134, 1993.

[3] Fu, L. Rule Generation from Neural Networks. *IEEE Transactions on Systems, Man, and Cybernetic*s, Volume 24, Number 8, pp. 1114-1124. August 1994.

[4] Gallant, S. Connectionist Expert Systems. *Communication of ACM*, Volume 31, pp. 152-169, 1988.

[5] Hayashi, Y. and A. Imura. Fuzzy Neural Expert System with Automated Extraction of Fuzzy If-Then Rules from a Trained Neural Network. *Proceedings of First International Symposium on Uncertainty Modeling and Analysis*, pp. 489-494, 1990.

[6] Homik, K., Stinchcombe, M., White, H. Multilayer feedforward networks are universal approximators. *Neural Networks Archive*, Volume 2, Number 5, pp.359-366, 1989.

[7] Huang, S. and H. Xing. Extracting intelligible and concise fuzzy rules from neural networks. *Fuzzy Sets and Systems*, Volume 132, pp. 233-243. 2001.

[8] Kasabov, N. Learning Fuzzy Rules and Approximate Reasoning in Fuzzy Neural Networks and Hybrid Systems. *Fuzzy Sets and Systems*, Volume 82, pp. 135-149. 1996.

[9] Kasabov, N. Learning Fuzzy Rules through Neural Networks. *Proceedings of the 1st New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems*, 1993.

[10] Krishnan, R., Sivakumar, G., Bhattacharya, P. A Search Technique for Rule Extraction from Trained Neural Networks. Patterns Recognition Letters, Volume 20, pp. 273-280. 1999.

[11] Mitra, S. and Y. Hayashi. Neuro-Fuzzy Rule Generation: Survey in Soft Computing Framework. IEEE Transactions on Neural Networks, Volume 11, Number 3, pp.748-768. May 2000.

[12] Nauck, D. and R. Kruse. NEFCLASS – A Neuro-Fuzzy Approach for the Classification of Data. *Proceedings of the 1995 ACM Symposium on Applied Computing*, pp. 461-465, 1995.
[13] Nauck, D., Nauck, U., Kruse, R. Generating Classification Rules with the Neuro-Fuzzy System NEFCLASS. *Proceedings of the 1996 North American Fuzzy Information Processing Society Conference*, pp. 466-470, 1996.
[14] Newman, D.J. & Hettich, S. & Blake, C.L. & Merz, C.J. (1998). *UCI Repository of Machine Learning Databases.* Irvine, CA: University of California, Department of Information and Computer Science. http://www.ics.uci.edu/~mlearn/MLRepository.html
[15] Towell, G. and J. Shavlik. Extracting Refined Rules from Knowledge-Based Neural Networks. *Machine Learning*, Volume 13, pp. 71-101, 1993.
[16] Towell, G., Shavlik, J., Noordewier, M.O. Refinement of Approximately Correct Domain Theories by Knowledge-Based Neural Networks. *Proceedings of the Eigth National Conference on Artificial Intelligence*, pp.861-866, 1990.