Destructive Algorithm for Rule Extraction based on a Trained Neural Network

M.E. ElAlami

Department of Computer Science
Mansoura University
Mansoura Egypt, 35516

ABSTRACT
The present paper introduces a new destructive algorithm for rule extraction based on a trained neural network. The degree of complexity of neural network increases exponentially as a factor of the numbers of input and hidden nodes. Therefore, the dimensionality of the trained neural network is reduced by using a proposed destructive algorithm to extract only the most effective values of the input attributes which have higher impact on the output result for each class. Thus, the searching efficiency is highly increased and the computation is dramatically reduced for extracting rules. The generated rules from the proposed model are fired through two levels for each class. As for the first level, it deals with each individual effective input value, and the second level is concerned with each possible conjunction of the effective input values. Moreover, the proposed model extracts the strongest rules which represent a large number of instances from the database by adjusting the similarity measure threshold value. Finally, the proposed model is evaluated on different public-domain datasets and compared with standard learning models from WEKA, then the results assert that the set of rules extraction from the proposed method is more accurate and concise compared with those obtained by the other models.

General Terms
Knowledge extraction, Neural network, Decompositional techniques.

Keywords
Rule extraction, Supervised learning, Neural network, Destructive technique, Performance measures.

1. INTRODUCTION
Knowledge Discovery (KD) is a rapidly expanding field in computer science. It has become very important because of an increased demand for methodologies and tools that can help the analysis and understanding of huge amounts of data generated on a daily basis by institutions. Knowledge discovery has been successfully used in various application areas: engineering [1], education [2], business and finance [3], insurance [4], telecommunication [5], chemistry [6], and medicine [7]. There are many techniques which are used for knowledge extraction from databases such as neural networks [8-9], genetic algorithms [10-11], decision tree [12-13], instance-based learning [14-15], rule induction [16-17], and support vector machine [18-19]. However, the ANN is still one of the most widely used techniques for knowledge extraction due to their advantages compared with the other techniques. It has advantages of nonlinear mapping, high tolerance to errors and robustness to noise. These properties are very promising in rule extraction and worthy of studying in depth. Therefore, research work in the area of extracting rules from trained neural networks has witnessed much activity recently. Rule extraction techniques from neural network are grouped into three approaches named as decompositional, pedagogical and eclectic. The decompositional algorithms analyze the hidden unit activations and connection weights for better understanding of network configurations. The pedagogical approach treats the ANN as a black-box and generates a knowledge representation that has the same (or similar) input-output mapping, disregarding the specific architecture of the network. Input-output pairs are generated using the trained network, and rules are extracted from this new database. Finally, eclectic approaches incorporate elements of both pedagogical and decompositional techniques [20]. The earliest decompositional rule extraction method is the KT algorithm developed by LiMin. Fu [21]. 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 [22]. 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. The REFANN algorithm extracts rules from trained ANN for non-linear function approximation or regression was developed by Setiono [23]. An optimization minimizes the search space by sorting the weights which have been proposed by Krishnan [24]. Towell et al. described a very interesting method nicknamed KBANN which is used to refine existing rules [25]. Its main idea is to encode the existing domain knowledge inside the network structure, then train such a specially initialized network, and finally extract new and better rules. Representatives of the pedagogical techniques category include Validity Interval Analysis (VIA) [26], TREPAN [27], Decision Tree Extractor (Dectext) [28], etc. VIA was designed as a general purpose rule extraction procedure, extracting symbolic knowledge from network. TREPAN, developed by Craven, it treats the network as an oracle used to statistically verify the correctness and significance of the generated rules. Dectext trained network and extract a classical decision tree from the network. Zhou et al. [29] developed an algorithm REFNE, by using an ensemble neural network to generate new data instances, and then extract symbolic rules from these instances. Garcez et al. [30] developed a method to extract rules from a neural net by first defining a partial ordering on the set of input vectors. Then, eclectic techniques combine the elements of the decompositional and the pedagogical approaches. They analyze an ANN at the individual unit level but also extract rules at the global level. One example of this approach is the DEDEC algorithm [31], which extracts if-then rules from MLP networks trained with the back-propagation algorithm. DEDEC extracts symbolic rules efficiently from a set of individual cases. It ranks the cases to be examined in order of importance. This is achieved by using the magnitude of the weight vectors in the trained ANN to rank the input units according to the relative share of their contribution to the

output units. The focus is on extracting rules from those cases that involve what are deemed to be the most important input units.

The present paper introduces a new decompositional approach for rule extraction via trained neural network. A destructive technique is used to reduce the dimensionality of the neural network by eliminating the hidden nodes or weights between layers if they are no longer actively used. Therefore, the most effective input values are used to extract the strongest rules from a given database.

This paper is organized as follows. The knowledge representation is performed in section 2. The proposed algorithm is described in section 3. The performance evaluation measures are introduced in section 4. The application and results are reported in section 5. Finally, the conclusion is presented in section 6.

2. KNOWLEDGE REPRESENTATION

A neural network is a powerful data modeling tool that is able to capture and represent complex input/output relationships. The true power and advantage of neural networks lie in their ability to represent both linear and non-linear relationships and in their ability to learn these relationships directly from the data being modeled. The most common neural network model is the multilayer perceptron (MLP). This type of neural network is known as a supervised network because it requires a desired output in order to learn. The goal of this type of network is to create a model that correctly maps the input to the output using historical data so that the model can then be used to produce the output when the desired output is unknown. Generally, a model of neural network consisting of layers of highly interconnected processing units can be trained to perform classification tasks. Patterns of input and output are first presented to the model for training. The sub-symbolic knowledge of a trained model is implicitly stored in the weights of the connections. Various methods exist to train a neural network; the most frequently used is back-propagation with the generalized delta rule [32]. With back-propagation, the user defines the number of hidden layers and nodes in each layer. Then, the model generates a first output, based on random weights of the connections. This output is compared with the desired output, and the difference between model prediction and desired output is calculated. The total squared sum of the calculated differences is then returned into the model, and the weights of the connections are changed to minimize the error. This procedure is repeated many times for all combinations of input and output. The ultimate goal for the model is to find a single set of weights that satisfies all the pairs of input and output presented to it, which is generalized to classify new data correctly. There are many techniques that have been developed to extract a low-level internal representation of ANN and convert it into a higher-level representation of the knowledge that can be interpreted more easily by humans [33-34]. Such a representation should be reasonably understood by humans, and to be formally treated by expert systems or inference engines. One of the most frequently used forms of representing knowledge extracted from neural networks are if-then rules, which are also a common representation mechanism for expert systems. A rule generally represents knowledge in the form of IF-THEN rules as follows: IF ‘‘condition(s)” THEN ‘‘class”. The antecedent (conditions) part of the rule contains a logical combination of predictor attributes and the consequent (class) part of the rule contains the class predicted for cases whose predictor attributes satisfy the antecedent part of the rule. The advantages of rules are the natural interpretation by humans and modularity during construction. In addition, it is relevant that production rules are also a formal way of presenting knowledge and in this way a good starting point for practical realization of the decision support system. Thus, the main goal of rule extraction is to discover the hidden knowledge and explain it understandably, to extract previously unknown relations and to ensure reasoning and defining capability.

3. THE PROPOSED ALGORITHM

A classification of the rule extraction algorithms from neural network may characterize different methods using five dimensions [35]: (a) the ‘expressive power’ of the extracted rules (types of rules extracted); (b) the ‘quality’ of the extracted rules (accuracy, fidelity comparing to the underlying network, comprehensibility and consistency of the extracted rules); (c) the ‘translucency’ of the method, based on local-global use of the neural network (analysis of individual nodes versus analysis of the total network function); (d) the algorithmic complexity of the method; (e) specialized network training schemes. One should add one more dimension to this scheme, (f) the treatment of linguistic variables: some methods work only with binary variables, other with discretized inputs, and yet other with continuous variables that are converted to linguistic variables automatically. Most of the rule extraction algorithms from neural network have basically two motivations. On the one hand, some authors noticed the need for simplification of neural networks to facilitate the rule extraction process. On the other hand, some papers have proposed algorithms mainly intended to clarify the knowledge encoded in previously trained ANNs [36]. One of the most problematic issues that arises in rule extraction algorithms which are cast as a search problem, is that the size of the hypothesis space for searching rules can be very large, which generally results in computationally expensive methods. For a problem domain with "n" binary features (values in {positive, negative, absent}) there are 3n possible conjunctive rules that can represent the underlying problem. In other words, the search space grows exponentially with the number of input features and the values that they contain in addition to the number of hidden nodes.

The present paper introduces a proposed algorithm based on a primary neural network for the simplification of neural networks to facilitate the rule extraction process. There is no need to enumerate the overall space of solutions for the method to extract rule, therefore the searching efficiency is highly increased and the computation is dramatically reduced. The proposed algorithm uses a three-layer artificial neural network namely; input layer, hidden layer and output layer. The architecture of the proposed neural network is set as follows; number of input nodes is equal to the number of input values of all attributes, number of hidden nodes is chosen randomly, number of output nodes is equal to number of output classes, the learning and momentum coefficients are determined randomly. Each node in a layer is connected to all nodes in the adjacent layer and each connection between nodes has a weight. In order to activate the neural network, a set of training inputs and corresponding outputs are required. If the nodes in the input layer are represented by X_1, X_2, X_3, ….. ,X_m, the nodes in the hidden layer are H_1, H_2, H_3, ... , H_n, and Wij is the weight on the connection between I_i and H_j, and the output value of j^{th} node in the hidden layer can be represented as follows:

$$H_j = TF\left(\sum_{i=1}^{m} W_{ij} . X_i\right) \qquad 1$$

The proposed algorithm uses a standard sigmoid function as a non-linear transfer function *TF(x)* and it can be represented as:

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

When calculating the value of an output node, the same transfer function is applied after summing up the results from the previous layer. Therefore, the final value of the k^{th} output node, O_K, is given by:

$$O_k = TF\left(\sum_{i=1}^{n} W_{jk} . H_j\right) \quad (3)$$

Where W_{jk} is the weight of the connection between H_j and O_K. The weights, W_{ij} and W_{jk}, are set randomly at the beginning of the training and are iteratively modified to obtain a structure of a network which minimizes the error between the neural network outputs and the desired outputs.

The proposed algorithm makes use of a destructive approach as pruning procedures to reach the optimal architecture of the trained neural network. This approach removes the hidden nodes or connections between layers which are no longer actively used. We assume that the most important connection has higher weights and its connected nodes have a higher impact on the output result and contain valuable information about the input data. Therefore, for each output node (class) the effective input values which have the more important connections (positives weights only) between the input–hidden layers and hidden-output layers can be extracted. Thus, the most effective values for each class are determined. So, the rule extraction belonging to a specific class can be generated through two levels, the first level checks each individual effective input value with the corresponding class, if any individual effective input value belongs completely to a specific class then generates a rule which contains the input value and belongs to this class. In the second level, the effective input values are combined alternatively in order to create different conjunctions. So, for each pattern in the database, we compute the similarity measure of each possible conjunction with each class. The similarity measure equals one if all values of conjunction are similar (exist) and equal zero otherwise. The rule is generated for a specific class if and only if the total similarity measure (TSM) of any conjunction belonging to this class is equal or greater than a threshold value, and the total similarity measure of the same conjunction with other classes must equal zero. The threshold value must be at least one, and it is used to control the number of extracted rules. Therefore, the higher threshold value leads to reduce the number of generated rules and extract the strongest rules which represent a large number of instances from the database.

4. PERFORMANCE EVALUATION

The commonly measures used to evaluate the performance of the learning algorithm are Accuracy, Precision, Sensitivity and Specificity [37]. The Accuracy is the number of correctly classified instances compared to the total number of instances presented to the system. It is defined as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

Precision is the percentage of true positives compared to the total number of instances classified as positive events, one can define the precision as:

$$Precision = \frac{TP}{TP + FP} \quad (5)$$

The sensitivity measure (also called recall rate) is the percentage of positive labeled instances that were predicted as positive. It is defined by:

$$Sensitivity = \frac{TP}{TP + FN} \quad (6)$$

The specificity is the percentage of negative labeled instances that were predicted as negative and it can be defined as:

$$Specificity = \frac{TN}{TN + FP} \quad (7)$$

Where;

TP (True Positives): is the number of instances covered by the rule which have the same class label as the rule.

FP (False Positives): is the number of instances covered by the rule which have a different class label from the rule.

FN (False Negatives): is the number of instances which are not covered by the rule but have the same class label as the rule.

TN (True Negatives): is the number of instances which are not covered by the rule and do not have the same class label as the rule.

5. APPLICATION AND RESULTS

The proposed model is tested on publicly available data sets in order to check its effectiveness. Three benchmark data sets are used namely; Play Tennis problem [38], Monk's problems [39], and Wisconsin breast cancer dataset [39]. The performance evaluation of the proposed model is compared with other learning models introduced by Waikato Environment for Knowledge Analysis (WEKA). WEKA [40] is an open source software which consists of a collection of machine learning algorithms for data mining tasks such as Decision Tree, Bayesian Networks, Radial Basis Function (RBF) Networks, and Single Conjunctive Rule Learner.

5.1 Play Tennis Problem

The Play Tennis problem has four attributes and one target class as shown in table 1. The four attributes of a given database are {Outlook, Temperature, Humidity, Wind}. The attribute Outlook has three possible values {Sunny, Overcast, Rain}, while the attribute Temperature has three possible values {Hot, Mild, Cool}, and the attribute Humidity has two possible values {High, Normal}, finally the attribute Wind has two possible values {Weak, Strong}. Indeed, the target class has two different values (Don't Play and Play).

Table 1. Example of play tennis [38]

Outlook	Temperature	Humidity	Wind	Target
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

The linguistic values of the given database are encoded as a binary form. The method of data representation is that if the position of input node matches with linguistic term, it will be represented by one otherwise it will be represented by zero. For example, if the input attribute Outlook has three linguistic values which are Sunny, Overcast, and Rain, we will represent the input attribute by 3 nodes which are [X_1, X_2, X_3] and represent the positions by [Sunny, Overcast, Rain]. Thus, the representation of Sunny value will be represented by [1, 0, 0]. The representation of Overcast value will be represented by [0, 1, 0], and the representation of Rain value will be represented by [0, 0, 1]. In representing the value of the target attribute, we use the same method which means that if we have two different classes of the target attribute, Don't Play and Play, then we will represent the target attribute by 2 nodes which are [O_1, O_2]. Then, the representation of Don't Play value will be represented by [1, 0] and the representation of Play value will be represented by [0, 1].

The multiplayer neural network using the back-propagation algorithm is trained on the encoded database. The final parameters of ANN are described as follows; number of input nodes are 10, number of hidden nodes are set to 4, number of output nodes are 2, the learning rate is set to 0.24, the momentum is set to 0.62, the allowable error is set to 0.000001 and the number of iteration is set to 30000. After the training is stopped, the weighs between input and hidden layer, and the weighs between hidden and output layer are extracted. Now, we are pruning the ANN by removing the hidden nodes or connections between layers which are no longer actively used. We assume that a more important connection has positive weights and its connected nodes have a higher impact on the output result. Therefore, for each output node (class) the effective input values which have the more important connections between the input–hidden layers and hidden-output layers can be extracted. Figure 1 shows the most effective input values which have the highest impact on the output class Don't Play (O_1).

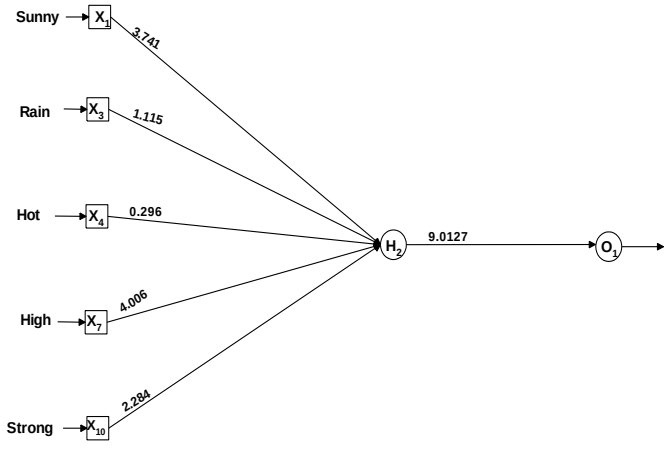

Fig 1: The most effective values for Don't Play class (O_1)

Therefore, the most effective input values for Don't Play class (O_1) are (Outlook $\in$ {Sunny, Rain}, while Temperature $\in$ {Hot}, and Humidity $\in$ {High}, finally Wind $\in$ {Strong}). The rule extraction can be generated in two levels, the first level for individual effective input value and the second level for each possible effective input values conjunction as shown on table 3. The rules extracted at the total similarity measure threshold $\geq$ 1.

Table 3. The extracted rules for Don't Play class (O_1)

Conjunction	$(TSM)_{Class_1}$	$(TSM)_{Class_2}$	Rules
There is no individual input value belongs completely to this class			----------
Outlook $\in$ {Sunny, Rain}	0	0	----------
Outlook $\in$ {Sunny} & Temperature $\in$ {Hot}	2	0	If Outlook is Sunny and Temperature is Hot Then Class is not play
Outlook $\in$ {Sunny} & Humidity $\in$ {High}	3	0	If Outlook is Sunny and Humidity is High Then Class is not play
Outlook $\in$ {Sunny} & Wind $\in$ {Strong}	1	1	----------
Outlook $\in$ { Rain} & Temperature $\in$ {Hot}	0	0	----------
Outlook $\in$ {Rain} & Humidity $\in$ {High}	1	1	----------
Outlook $\in$ { Rain} & Wind $\in$ {Strong}	2	0	If Outlook is Rain and Wind is Strong Then Class is not play
Temperature $\in$ {Hot} & Humidity $\in$ {High}	2	2	----------
Temperature $\in$ {Hot} & Wind $\in$ {Strong}	1	0	If Temperature is Hot and Wind is Strong Then Class is not play
Humidity $\in$ {High} & Wind $\in$ {Strong}	2	1	----------

For extracting rules that belong to the output class Play, (O_2), the previous technique is applied and the most effective input values which have the highest impact on it are shown in figure 2.

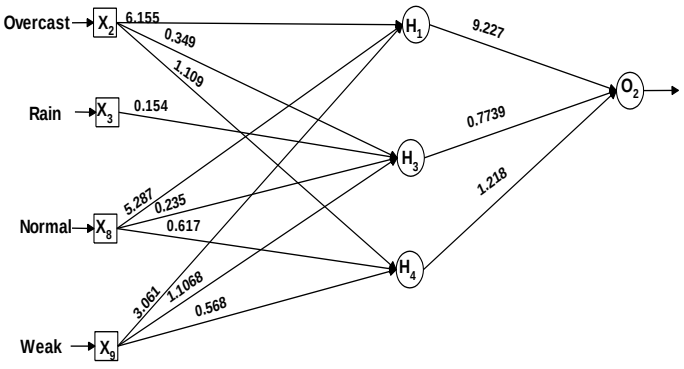

Fig 2: The most effective values for Play class (O_2)

Therefore, the most effective input values for the Play class (O_2) are (Outlook $\in$ {Overcast, Rain}, Humidity $\in$ {Normal}, and Wind $\in$ {Weak}). Table 4 shows the extracted rules that belong to the Play class (O_2) according to the two levels at the total similarity measure threshold $\geq$ 2.

Table 4. The extracted rules for Play class (O_2)

Conjunction	$(TSM)_{Class_2}$	$(TSM)_{Class_1}$	Rules
The input value "Outlook ∈ {Overcast}" belongs completely to this class			If Outlook is Overcast Then Class is play
Outlook ∈ {Rain} & Humidity ∈ {Normal}	2	1	----------
Outlook ∈ {Rain} & Wind ∈ {Weak}	3	0	If Outlook is Rain and Wind is Weak Then Class is play
Humidity ∈ {Normal} & Wind ∈ {Weak}	4	0	If Humidity is Normal and Wind is Weak Then Class is play

The comparison of the performance measures for the proposed model and other learning models from WEKA is shown on table 5. The results show that the proposed model achieved the highest performance compared with the other models.

Table 5. The performance measures of various models for Play Tennis Problem

Models	Acc (%)	Pre (%)	Sen (%)	Spec (%)
Decision Tree	65.26	100	65.25	100
Bayesian Networks	92.68	100	90.18	100
RBF Networks	100	100	100	100
Single Conjunctive Rule Learner	96.5	100	98.47	100
The proposed model	100	100	100	100

5.2 Monk's Problems

Monk's problems are a classical set of benchmarks which are widely used in classification task. The Monk's problems are a collection of three binary classification problems Monk-1, Monk-2 and Monk-3 which are described by the following six attributes: head-shape ∈{round, square, octagon}, body-shape ∈{round, square, octagon}, is-smiling ∈{yes, no}, holding ∈{sword, baloon, flag}, jacket-color ∈{red, yellow, green, blue}, and has-tie ∈{yes, no}. The proposed model is performed on Monk-1 problem which has 124 instances and the linguistic values of the given database are encoded as a binary form. The ANN is trained on the encoded database and the final parameters of ANN are adjusted as; number of input nodes are 17, number of hidden nodes are set to 5, number of output nodes are 2, the learning rate is set to 0.37, the momentum is set to 0.81, the allowable error is set to 0.0000001 and the number of iteration is set to 32870. The ANN performs pruning by removing the hidden nodes or connections between layers which are no longer actively used. Therefore, the most effective input values which have the highest impact on each output class are extracted. Consequently, the most effective input values for target class (O_1) are (head-shape ∈{round, square, octagon} & body-shape ∈{round, square, octagon} & jacket-color ∈{red}& holding ∈{sword} & has-tie ∈ {yes}) and the most effective input values for don't target class (O_2) are (head-shape ∈{round, square, octagon} & body-shape ∈{round, square, octagon} & jacket-color ∈{ yellow, green, blue}& is-smiling ∈{no}). Finally, the proposed model generates rules for each class in two levels, the first level for individual effective input value and the second level for each possible effective input values conjunction as shown on table 6 and table 7. The total similarity measure threshold adjusted at $\geq$ 3 for the first class (O_1) and $\geq$ 7 for the second class (O_2).

Table 6. The extracted rules for Target class (O_1)

Conjunction	$(TSM)_{Class_1}$	$(TSM)_{Class_2}$	Rules
The input value "jacket-color ∈ { red}" belongs completely to Class =Yes			If jacket-color is red then Class = Yes
head-hape ∈ {octagon} & body-hape ∈ {octagon}	17	0	If head-shape is octagon and body-shape is octagon Then Class = Yes
head-shape ∈ {square} & body-shape ∈ {square}	15	0	If head-shape is square and body-shape is square Then Class = Yes
head-shape ∈ {round} & body-shape ∈ {round}	9	0	If head-shape is round and body-shape is round Then Class = Yes
head-hape ∈ {octagon} & holding ∈ {sword} & has-tie ∈ { yes}	8	0	If head-shape is octagon and holding is sword and has-tie is yes Then Class = Yes
head-hape ∈ {octagon} & body-shape ∈ {round} & has-tie ∈ { yes}	3	0	If head-shape is octagon and body-shape is round and has-tie is yes Then Class = Yes

The comparison between the proposed model and the other models of WEKA for Accuracy, Precision, Sensitivity and Specificity measures is shown on table 8. The results illustrated that the proposed model is more superior to the compared models.

Table 7. The extracted rules for Don't Play class (O_2)

Conjunction	$(TSM)_{Class_2}$	$(TSM)_{Class_1}$	Rules
There is no individual input value belongs completely to this class			----------
head-shape ∈ {round}& body-shape ∈ {square or octagon} & jacket-color ∈ {yellow or green or blue}	38	0	If head-shape is round and body-shape is square or octagon and jacket-color is yellow or green or blue Then Class = No
head-shape ∈ {round or square}& body-shape ∈ {octagon} & jacket-color ∈ {yellow or green or blue}	22	0	If head-shape is round or square and body-shape is octagon and jacket-color is yellow or green or blue Then Class = No
head-shape ∈ {square}& body-shape ∈ { round or octagon} & jacket-color ∈ {yellow or green or blue}	20	0	If head-shape is square and body-shape is round or octagon and jacket-color is yellow or green or blue Then Class = No
head-shape ∈ {round }& body-shape ∈ {square} & is-smiling ∈ {no}	7	0	If head-shape = round and body-shape = square and is-smiling = no Then Class = No

Table 8: The performance measures of various models for Monk-1 problem

Models	Acc (%)	Pre (%)	Sen (%)	Spec (%)
Decision Tree	95.68	94.79	96.13	95.16
Bayesian Networks	78.85	68.94	88.64	75.64
RBF Networks	84.21	77.34	87.49	79.83
Single Conjunctive Rule Learner	98.16	97.16	98.89	96.85
The proposed model	100	100	100	100

5.3 Wisconsin Breast Cancer

The Wisconsin breast cancer dataset is one of the favorite benchmark datasets. It contains 699 cases, with 458 benign (65.5%) and 241 (34.5%) malignant cases of cancer and it is described by the following nine attributes: Clump Thickness ∈{1-10 integer values}, Uniformity of Cell Size ∈{1-10 integer values}, Uniformity of Cell Shape ∈{1-10 integer values}, Marginal Adhesion ∈{1-10 integer values}, Single Epithelial Cell Size ∈{1-10 integer values}, Bare Nuclei ∈{1-10 integer values}, Bland Chromatin ∈{1-10 integer values}, Normal Nucleoli ∈{1-10 integer values}, and Mitoses ∈{1-10 integer values}. The ANN is trained on encoded database and the training parameters of ANN are set as; number of input nodes are 90, number of hidden nodes are set to 8, number of output nodes are 2, the learning rate is set to 0.28, the momentum is set to 0.62, the allowable error is set to 0.0000001 and the number of iteration is set to 42358. The ANN is pruning by removing the hidden nodes or connections between layers which are no longer actively used. Consequently, the most effective input values for benign class (O_1) are (Uniformity of Cell Size ∈{1-2}) and the most effective input values for malignant class (O_2) are (Clump Thickness ∈{1-5}& Uniformity of Cell Shape ∈{1-3}& Marginal Adhesion ∈{1-3}& Bare Nuclei ∈{1} & Bland Chromatin ∈{1-4}). The performance measures of the proposed model are compared with the other learning models from WEKA as shown on table 9. The results show that the proposed model achieved the best performance compared with the other models.

Table 9. The performance measures of various models for Breast Cancer dataset

Models	Acc (%)	Pre (%)	Sen (%)	Spec (%)
Decision Tree	76.45	96.12	78.65	78.27
Bayesian Networks	75.49	84.12	83.36	59.33
RBF Networks	80.04	91.25	80.98	71.24
Single Conjunctive Rule Learner	83.58	95.47	86.76	84.59
The proposed model	97.85	96.23	100	91.89

6. CONCLUSION

Research work in the area of extracting rules from trained neural networks has witnessed much activity recently. However, the degree of complexity of ANN increases exponentially as a factor of the numbers of input and hidden nodes. The complexity problem can be alleviated by adopting heuristics to constrain the search space. The present paper introduces a new methodology for the simplification of the ANN by pruning the weights among neurons to obtain simple but substantial expressions of ANN and facilitate the rule extraction process. Therefore, the most effective values of inputs attributes are only extracted to generate the rules for each class. Consequently, there is no need to enumerate the overall space of solutions and generate a small number of linguistic rules that is easy for users to understand. The true positive rules of a specific class are extracted when the total similarity measure of the tested conjunction of each rule has a value greater than the threshold value and equal zero with other classes. The higher threshold value leads to reduce the number of generated rules and extract the strongest rules which represent a large number of instances from the database. Extensive experiments have been carried out in this study to evaluate how well the proposed model performed on three benchmark classification problems in comparison with the other models. Finally, the results indicate that the proposed model is the superior compared with other model.

REFERENCES

[1] Oscar Marbán, Javier Segovia, Ernestina Menasalvas, Covadonga Fernández-Baizán, "Toward data mining engineering: A software engineering approach", Information Systems, Volume 34, Issue 1, March 2009, Pages 87-107.

[2] Ali Buldu, Kerem Üçgün, "Data mining application on students' data", Procedia- Social and Behavioral Sciences, Volume 2, Issue 2, 2010, Pages 5251-5259.

[3] Flora S. Tsai, Agus T. Kwee, "Database optimization for novelty mining of business blogs", Expert Systems with Applications, Volume 38, Issue 9, September 2011, Pages 11040-11047.

[4] Chien-Hsing Wu, Shu-Chen Kao, Yann-Yean Su, Chuan-Chun Wu, "Targeting customers via discovery knowledge for the insurance industry", Expert Systems with Applications, Volume 29, Issue 2, August 2005, Pages 291-299.

[5] Tong-Yan Li, Xing-Ming Li, "Preprocessing expert system for mining association rules in telecommunication networks ", Expert Systems with Applications, Volume 38, Issue 3, March 2011, Pages 1709-1715

[6] Uko Maran, Sulev Sild, Iiris Kahn, Kalev Takkis, "Mining of the chemical information in GRID environment", Future Generation Computer Systems, Volume 23, Issue 1, 1 January 2007, Pages 76-83.

[7] Xuezhong Zhou, Shibo Chen, et el "Development of traditional Chinese medicine clinical data warehouse for medical knowledge discovery and decision support ", Artificial Intelligence in Medicine, Volume 48, Issues 2-3, February-March 2010, Pages 139-152.

[8] Eyal Kolman, Michael Margaliot, "Extracting symbolic knowledge from recurrent neural networks-A fuzzy logic approach", Fuzzy Sets and Systems, Volume 160, Issue 2, 16 January 2009, Pages 145-161.

[9] Jun Wang, Yunpeng Wu, Xuening Liu, Xiaoying Gao, "Knowledge acquisition method from domain text based on theme logic model and artificial neural network", Expert Systems with Applications, Volume 37, Issue 1, January 2010, Pages 267-275.

[10] C.K. Kwong, K.Y. Chan, Y.C. Tsim, "A genetic algorithm based knowledge discovery system for the

design of fluid dispensing processes for electronic packaging", Expert Systems with Applications, Volume 36, Issue 2, Part 2, March 2009, Pages 3829-3838.

[11] Muzaffer Kapanoglu, Mete Alikalfa, "Learning IF–THEN priority rules for dynamic job shops using genetic algorithms", Robotics and Computer-Integrated Manufacturing, Volume 27, Issue 1, February 2011, Pages 47-55.

[12] Leyli Mohammad Khanli, Farnaz Mahan, Ayaz Isazadeh, "Active rule learning using decision tree for resource management in Grid computing", Future Generation Computer Systems, Volume 27, Issue 6, June 2011, Pages 703-710.

[13] Mouloud Boumahdi, Jean-Paul Dron, Saïd Rechak, Olivier Cousinard, "On the extraction of rules in the identification of bearing defects in rotating machinery using decision tree", Expert Systems with Applications, Volume 37, Issue 8, August 2010, Pages 5887-5894.

[14] Francesco Gagliardi, "Instance-based classifiers applied to medical databases: Diagnosis and knowledge extraction", Artificial Intelligence in Medicine, Volume 52, Issue 3, July 2011, Pages 123-139.

[15] Amelia Zafra, Cristóbal Romero, Sebastián Ventura, "Multiple instance learning for classifying students in learning management systems", Expert Systems with Applications, Volume 38, Issue 12, November-December 2011, Pages 15020-15031.

[16] Wouter Verbeke, David Martens, Christophe Mues, Bart Baesens, "Building comprehensible customer churn prediction models with advanced rule induction techniques", Expert Systems with Applications, Volume 38, Issue 3, March 2011, Pages 2354-2364.

[17] Jerzy Błaszczyński, Roman Słowiński, Marcin Szeląg, "Sequential covering rule induction algorithm for variable consistency rough set approaches", Information Sciences, Volume 181, Issue 5, 1 March 2011, Pages 987-1002.

[18] Nahla Barakat, Andrew P. Bradley, "Rule extraction from support vector machines: A review", Neurocomputing, Volume 74, Issues 1-3, December 2010, Pages 178-190.

[19] M.A.H. Farquad, V. Ravi, S. Bapi Raju, "Support vector regression based hybrid rule extraction methods for forecasting", Expert Systems with Applications, Volume 37, Issue 8, August 2010, Pages 5577-5589.

[20] Humar Kahramanli, Novruz Allahverdi, "Rule extraction from trained adaptive neural networks using artificial immune systems", Expert Systems with Applications 36 (2009) 1513–1522.

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

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

[23] 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.

[24] R. Krishnan, G. Sivakumar, and P. Bhattacharya, "A search technique for rule extraction from trained neural networks," Pattern Recognit. Lett., vol. 20, no. 3, pp. 273–280, Mar. 1999.

[25] G. G. Towell, J. W. Shavlik, and M. O. Noordewier, "Refinement of approximate domain theories by knowledge-based neural networks," in Proc. 8th Nat. Conf. Artif. Intell., Boston, MA, 1990, pp. 861–866.

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

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

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

[29] 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.

[30] 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.

[31] Tickle, A.B., Orlowski, M., and Diederich, J., "DEDEC: A Methodology for Extracting Rules from Trained Artificial Neural Networks", Proceedings of The Rule Extraction From Trained Artificial Neural Networks Workshop, 1996.

[32] Rumelhart, D. E.. G. E. Hinton. and R. J. Williams 1986. "Learning internal representations by error propagation", Page 318 in Parallel Distributed Processing: Explorations in the Micro-Structure of Cognition. Vol. 1. D. E. Rumelhart and J. L. McClelland, ed.MIT Press, Cambridge, MA.

[33] C. McMillan, M.C. Mozer, and P. Smolensky, "The connectionist science game: Rule extraction and refinement in a neural network", in Proceedings of the 13th Annual Conference of the Cognitive Science Society, 1991.

[34] R. Setiono and H. Liu, "Understanding neural networks via rule extraction", edited by Chris S. Mellish, in Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, San Mateo, August 20–25, 1995, Morgan Kaufmann, pp. 480–487.

[35] A. B. Tickle, R. Andrews, M. Golea, and J. Diederich, "The Truth Will Come to Light: Directions and Challenges in Extracting the Knowledge Embedded Within Trained Artificial Neural Networks", IEEE Trans. Neural Networks, vol 9, pp. 1057–1068, 1998.

[36] Humar Kahramanli, Novruz Allahverdi, "Rule extraction from trained adaptive neural networks using artificial immune systems", Expert Systems with Applications, Volume 36, Issue 2, Part 1, March 2009, Pages 1513-1522.

[37] K. C. TAN, Q. YU and J. H. ANG, "A co-evolutionary algorithm for rules discovery in data mining", International Journal of Systems Science Vol. 37, No. 12, 10 October 2006, 835–864.

[38] Tom M. Mitchell, "Machine Learning", McGraw-Hill Book Co, Copyright 1997.

[39] S. B. Thrun, "The MONK's problem: A performance comparison of different learning algorithms", Carnegie-Mellon University, Technical Report, 1991.

[40] WEKA at http://www.cs.waikato.ac.nz/~ml/wek