Destructive Learning Analysis and Constructive Algorithm for Rule Extraction based on a Trained Neural Network using Gene Expression Programming

Marghny H. Mohamed  
Professor of Computer Science  
Faculty of Computer and Information  
Assiut University

Yasmeen T. Mahmoud  
Faculty of Science  
Department of Computer Science  
Assiut University

Saad Z. Rida  
Professor of Mathematics  
Faculty of Science  
South Valley University

ABSTRACT
The present paper introduces destructive neural network learning techniques and presents the analysis of the convergence rate of the error in a neural network with and without threshold. Also, a constructive algorithm for rule extraction based on a trained neural network using Gene Expression Programming (GEP) is proposed. The rules are not an easy task due to the large number of examples entered into the input layer. Thus, we can use GEP to encode the rules in the form of logic expression. Finally, the proposed model is evaluated on different public-domain datasets and compared with standard learning models from WEKA, and then the results accentuate that the set of rules extraction from the proposed method is more accurate and brief compared with those achieved by the other models.

General Terms
Neural network, Destructive learning, Rule Extraction, Gene expression programming.

Keywords
Neural Network, Destructive Learning, Constructive Learning, Pruning, Rule Extraction, Classification Rules, Gene Expression Programming.

1. INTRODUCTION
Large databases are regularly being collected in science, medicines and business. The goal of Knowledge-discovery in databases is to extract usable knowledge from a large amount of data. To produce models that provide vision into data, it draws upon methods in statistics, signal processing, pattern recognition, information theory, machine learning, and neural networks. Such models are expected to be accurate and comprehensible to experts in the field. Although artificial neural network (ANN) usually reaches high classification accuracy, the obtained results sometimes may be incomprehensible. This fact is causing a serious problem in data mining applications. Various methods have been improved to extract rules that are resulted from ANN and needed to be formed to solve this problem. There are several black-box rule extraction algorithms using evolutionary approaches. In essence, a population consists of candidate solutions and the fitness function reflects the chosen optimization criterion. Most often, the fitness is based on fidelity, but may also include terms clearly targeting improved comprehensibility or accuracy. Comprehensibility is generally controlled by using a length consequence, while accuracy can be targeted by including training instances not used when building the opaque model. Therefore, in this paper a study on rule extraction from trained ANNs for classification problems is carried out. The proposed approach makes use of particle geneexpression algorithm to transform the behaviors of trained ANNs into accurate and comprehensible classification rules.

This paper is organized as follows. An Overview of Rule Extraction is presented in section 2. Gene expression programming is introduced in section 3. The Architectures of the networks is shown in Section 4, the analysis of the convergence rate in a destructive neural network without and with thresholds in the output layer has been presented in Sections 5 and 6, respectively. In Section 7, we will discuss how to use GEP to extract rules from trained artificial neural network.

The experimental results and the dataset used are introduced in section 8. The result and discussion are reported in section 9. Finally, the conclusion is presented in section 10

2. OVERVIEW OF RULE EXTRACTION
In data mining research, rule extraction has become a progressively important topic, and ANNs is applied for machine learning in a variety of real world applications via an increasing number of researchers and practitioners [1, 2, 3, 4, 5]. An inherit imperfection of ANNs is that the learned knowledge is disguised in a large amount of connections, which leads to the poor explanation ability and poor limpidity of knowledge [6]. To compensate this imperfection, developing algorithms to extract emblematic rules from trained neural networks has been a significant topic in recent years.

Rule extraction techniques can characterize extracted model in form of M of N rules, If-Then rules, decision table, decision tree, etc.Opaque models like ANN and SVM shows improved accuracy performance in comparison to white box or apparent model like decision tree [7, 8, 9]. Rule extraction is the task of making an opaque model apparent and expectantly comprehensible [10]. Comprehensibility is essential in many application areas and to make neural networks more comprehensible we need rule extraction from neural network. Rule extraction techniques from neural network are gathered into three approaches pedagogical, decompositional and eclectic. The pedagogical method deals with the ANN as a black-box and produces a knowledge representation that has the same input-output mapping, ignoring the specific architecture of the network. Input-output pairs are produced
using the trained network, and rules are extracted from this new database. The decompositional techniques study the hidden unit activations and connection weights for improved understanding of network configurations. Finally, eclectic approaches integrate components of both pedagogical and decompositional techniques [11]. The first decompositional rule extraction technique is the KT algorithm developed by LiMin. Fu [12]. The KT algorithm creates rules for each concept consistent to a hidden or output unit whose summed weights exceed the threshold of the unit. The same idea is involved in the Subset algorithm developed by Towell and Shavlik [13]. This algorithm checks each subset and tries to find out if any of these links exceed the bias. If exceeded, then these weights are rephrased as rules. The REFANN algorithm extracts rules from trained ANN for non-linear function approximation or regression was developed by Setiono [14]. Krishnan had projected an optimization minimizes the search space by sorting the weights [15]. Towell et al. defined a method named KBANN which is used to progress existing rules [16]. Its main idea is to encode the existing domain knowledge inside the network structure, then train such an initialized network, and finally extract new and improved rules. Validity Interval Analysis (VIA) [17], TREPAN [18], Decision Tree Extractor (Dectext) [19], etc were contained in representatives of the pedagogical techniques category. A general purpose rule extraction procedure that extracting characteristic knowledge from network designed in VIA algorithm. Craven developed the TREPAN procedure which treats the network as an oracle used to statistically verify the significance and correctness of the generated rules. Dectext was designed to trained network and extract a classical decision tree from the network. REFINN algorithm was developed by Zhou et al. [20], by using an assembly neural network to generate new data instances, and then extract emblematic rules from these instances. A method to extract rules from a neural network was developed by Garcez et al. [21], by first defining a partial ordering on the set of input vectors. Then, eclectic techniques merge the elements of the decompositional and the pedagogical approaches. They analyze an ANN at the individual unit level but also extract rules at the comprehensive level. The DEDEC algorithm is an example of this approach [22], which extracts if-then rules from MLP networks trained with the back-propagation algorithm. Characteristic rules are extracted by DEDEC efficiently from a set of individual cases. It arranges the cases to be examined in order of importance. Using the magnitude of the weight vectors in the trained ANN to order the input units according to the relative share of their donation to the output units achieves that target. The emphasis is on extracting rules from those cases that occupy what are considered to be the most important input units.

3. GENE EXPRESSION PROGRAMMING

Evolutionary approaches are using for several black-box rule extraction algorithms. The most recent technique of evolutionary algorithms is Gene expression programming [23].

Gene expression programming (GEP) is, like genetic programming (GP) and genetic algorithms (GAs), a genetic algorithm as it uses populations of individuals, selects them based on fitness, and introduces genetic disparity using one or more genetic operators [24]. The important difference between the three algorithms exists in the nature of the individuals, the individuals are nonlinear entities of different shapes and sizes in GP, the individuals are linear strings of fixed length in GAs, and the individuals are encrypted as linear strings of fixed length which are subsequently expressed as nonlinear entities of different shapes and sizes in GEP.

The interaction of expression trees and chromosomes in GEP denotes an evident translation system for translating the language of chromosomes into the language of expression trees (ETs). In this work, the structural organization of GEP chromosomes is presented; allows a truly functional genotype/phenotype relationship, as any modification made in the genome always results in syntactically correct ETs or programs. Certainly, the diverse set of genetic operators developed to introduce genetic diversity in GEP populations always produces valid ETs. So, GEP is an artificial life system, well recognized beyond the replicator threshold, able to evolution and adaptation.

The advantages of a system like GEP are clear from nature, but the most important should be emphasized. First, the chromosomes are simple entities: relatively small, linear, easy to manipulate genetically (replicate, mutate, recombine, transpose, etc.), compact. Second, the ETs are exclusively the expression of their individual chromosomes; they are selected to reproduce with modification according to fitness, and they are the entities upon which selection acts. The chromosomes of the individuals, not the ETs, are reproduced with modification and transmitted to the next generation during reproduction.

Based on these characteristics, GEP is tremendously flexible and significantly exceeds the existing evolutionary techniques. Actually, in the most complex problem, the evolution of cellular automata rules for the density-classification task, GEP exceeds GP by more than four orders of magnitude.

The gene expression algorithms can be apportioned into two categories: unsupervised and supervised.

The common task is predicting the state of gene expression samples for supervised methods, for example, cancer against normal. A lot of classification models are developed, for their huge application abilities. There are three typical main categorical methods for this task: statistical based methods, traditional machine learning methods, and association rule-based classifiers. The high graded genes and, the relation among genes is not fully explored are usually selected by The statistical-based methods [25,26,27,28]. Most of machine learning methods are black box and hard to construe, although they take the relation between the genes into consideration. For example, although SVM [29] achieves very high classification accuracy, it remains hard to construe the results.

In the unsupervised methods, association rules are used to find interesting gene expression patterns [28,31,32,33], clustering [30] is commonly used to find structured genes or similar samples, reconstruct gene regulatory network [34,35], and discover functional modules [36].
4. NEURAL NETWORK ARCHITECTURE

We consider an ANN that consists of an input layer with \( n+1 \) node, a hidden layer with \( h \) units, and an output layer with \( l \) units

\[
y_i = g \left( \sum_{j=1}^{h} w_{ij} f \left( \sum_{k=1}^{n+1} v_{jk} x_k \right) \right), \quad i = 1, 2, \ldots, l, \quad (1)
\]

where \( x_k \) indicates the \( k \)-th input value, \( y_i \) the \( i \)-th output value, \( v_{jk} \) a weight connecting the \( k \)-th input node with the \( j \)-th hidden unit, and \( w_{ij} \) a weight between the \( j \)-th hidden unit and the \( i \)-th output unit. The functions \( f(t) \) and \( g(t) \) are given by

\[
f(t) = \frac{1 - e^{-t}}{1 + e^{-t}},
\]

\[
g(t) = \frac{1}{1 + e^{-t}},
\]

respectively. We write (1) as

\[
y = g \left( Wf(Vx) \right), \quad (4)
\]

Where we set \( x = (x_1, x_2, \ldots, x_j, x_{j+1})^T \) with \( x_{j+1} = -1, y = (y_1, y_2, \ldots, y_i)^T, V = (v_{jk}) \) and \( W = (w_{ij}) \). Moreover, \( f(Vx) \) means \( (f(V_1, x), f(V_2, x), \ldots, f(V_h, x))^T \) and \( g(Wf(Vx)) \) indicates \( (g(W_1, f(Vx)), g(W_2, f(Vx)), \ldots, g(W_l, f(Vx)))^T \), where

\[
V_i = (v_{i1}, v_{i2}, \ldots, v_{ih+1})^T \quad \text{and} \quad W_i = (w_{i1}, w_{i2}, \ldots, w_{ih})^T.
\]

This network is shown in Fig 1. Let \((x^v, y^v), v = 1, 2, \ldots, m, \) be the training data for the network. We define an output error between the outputs of the network for the inputs \( x^v \) and the relevant outputs \( y^v \) by

\[
f(V, W) = \sum_{v=1}^{m} \|g^{-1}(y^v) - Wf(Vx^v)\|^2, \quad (5)
\]

where \( g^{-1}(y^v) = \left(g^{-1}(y_1^v), g^{-1}(y_2^v), \ldots, g^{-1}(y_l^v)\right)^T \) is the inverse of the function, and \( \| \cdot \| \) stands for the Euclidean norm. To determine \( V \) and \( W \), we need to minimize the error function (5).

5. CONVERGENCE RATE IN A DESTRUCTIVE NEURAL NETWORK WITHOUT THRESHOLDS IN THE OUTPUT LAYER

In this section we will present the analysis of the convergence of the error in destructive neural network.

Let \( v \) denote a connection weight vector between the \( (h + 1) \)-th hidden unit and the input layer, and let \( w \) be a weight connecting the \( (h + 1) \)-th hidden unit with the output layer. The ANN after removing the \( (h + 1) \)-th hidden unit is shown in Fig 2. We denote the weight matrices \((V, v) \) and \((W, w) \) by \( \bar{V} \) and \( \bar{W} \), respectively.

Then we can rewrite (5) as

\[
f(V, W) = \sum_{v=1}^{m} \|g^{-1}(y^v) - \bar{W}f(Vx^v) - wf(v, x^v)\|^2
\]

Where \( Wf(Vx^v) = \bar{W}f(Vx^v) + wf(v, x^v) \) and \( w \) is the removing weight vector.

The function can be written as follows:

\[
f(V, W) = f(\bar{V}, \bar{W}) - 2(d, w) + a\|w\|^2
\]

Fig 2: Neural network after removing one unit in the hidden layer.

in which \( d \) and \( a \) denote

\[
d = \sum_{v=1}^{m} f(v, x^v)c^v,
\]

and

\[
a = \sum_{v=1}^{m} f^2(v, x^v),
\]

Where \( c^v = g^{-1}(y^v) - Wf(Vx^v) \) and the symbol \( \langle \ldots \rangle \) indicates an inner product in \( R^l \). When the vector \( v \) is fixed, the vector \( w \) which minimizes the error function by

\[
w = \frac{d}{a}
\]
where \( \theta \) is the unit matrix and 
\[ \sum_{i=1}^{m} \theta_{ij} = 1 \]
for arbitrary vector \( U \).

This shows that the error after determining the weight \( w \) between the hidden and output layers is not convergent. In addition, the eigenvalues of this matrix can be obtained as follows:

One property of the matrix \( I_i(v) \) is

\[
|I_i(v) U, U \rangle = \| U \|^2 + \frac{1}{a} \| S \|^2 \| U \|^2 = 2
\]

and the other is

\[
|I_i(v) U, U \rangle = \| U \|^2 + \frac{1}{a} \| S \|^2 \| U \|^2 \geq \| U \|^2.
\]

From (8) and (9) we get

\[
1 \leq \frac{|I_i(v) U, U \rangle}{\| U \|^2} \leq 2
\]

which implies

\[
1 \leq \max_{\lambda} \lambda_i \leq 2
\]

Where \( \lambda_i, i = 1, \ldots, m \) are the eigenvalues of the matrix \( I_i(v) \).

Next, we calculate the eigenvalues of the matrix \( I_i(v) \) to examine the Convergence rate of the error in more detail. The matrix \( I_i(v) \) can be written as

\[
I_i(v) = \begin{pmatrix}
    z_{i1} & -z_{i2} & \cdots & -z_{im} \\
    -z_{i2} & z_{i1} & \cdots & -z_{im} \\
    \vdots & \vdots & \ddots & \vdots \\
    -z_{im} & -z_{i1} & \cdots & z_{i2} \\
\end{pmatrix}
\]

Where \( z_{ij} = 1 + s_i^2 / a, z_{ij} = -s_i g_i / a, \) and \( z_{ij} = z_{ij} \). The characteristic equation of this matrix is

\[
\gamma_i(\lambda) = \lambda - z_{i1} z_{i2} \cdots z_{im} = 0.
\]

By putting \( \eta_i = (1 - \lambda) a / s_i^2 \), we can reform \( \gamma_i(\lambda) \) as follows:

\[
\gamma_i(\lambda) = -\frac{1}{a m} \sum_{i=1}^{m} \lambda_i \gamma_i(\lambda) = 0.
\]
where 
\[ q^v = g^{-1}(y^v) - Wf(Vx^v) + \theta. \]
By minimizing this error function, we can easily determine \( w \) and \( \Delta \theta \) as
\[
\begin{align*}
    w &= \frac{md_1 - a_2d_2}{b}, \\
    \Delta \theta &= \frac{a_2d_1 - a_2d_2}{b},
\end{align*}
\] (10) (11)
where
\[
\begin{align*}
    a_1 &= \sum_{i=1}^{m} f^2(v_i x^v), \\
    a_2 &= \sum_{i=1}^{m} f(v_i x^v), \\
    d_1 &= \sum_{i=1}^{m} f(v_i x^v) q^v, \\
    d_2 &= \sum_{i=1}^{m} q^v, \\
    b &= m a_1 - a_2^2.
\end{align*}
\]
Now we consider the convergence rate of the error as in the previous section. Since
\[
\begin{align*}
\tilde{q}^v &= g^{-1}(y^v) - \bar{W}f(\bar{V}x^v) + \bar{\theta} \\
    &= g^{-1}(y^v) - Wf(Vx^v) + w f(v_i x^v) + \theta - \Delta \theta, \\
\end{align*}
\]
we can write the error at \( i \)-th component in the output layer as
\[
\tilde{q}^v_i = q^v_i + w f(v_i x^v) - \Delta \theta_i. 
\] (12)
From (10), the \( i \)-th component of \( w \) can be written as
\[
w_i = \frac{m}{b} \frac{d_1}{d_2} - \frac{a_2}{b} d_2 = \frac{m}{b} \sum_{i=1}^{m} s_i q^v - a_2 \frac{m}{b} \sum_{i=1}^{m} q^v = \frac{m}{b} \left[ Qs - a_2 \frac{m}{b} Q, 1 \right] = \left[ Q, \frac{m}{b} S - a_2 \frac{m}{b} 1 \right].
\]
Similarly from (11), we can also write \( \Delta \theta_i \) as
\[
\Delta \theta_i = \left[ Q, \frac{a_2}{b} S - \frac{a_2}{b} 1 \right].
\]
Where \( \tilde{q} = (\tilde{q}^v_1, \tilde{q}^v_2, \ldots, \tilde{q}^v_m) \), \( Q = (q^v_1, q^v_2, \ldots, q^v_m) \), \( \tilde{s} = (s_1, s_2, \ldots, s_m) \), and \( \tilde{1} = (1, 1, \ldots, 1) \) with \( s_i = f(v_i x^v) \). Hence, (12) can be rewritten as
\[
\tilde{q}^v_i = q^v_i + \frac{m}{b} \tilde{Q}_i s_i - a_2 \frac{m}{b} \tilde{Q}_i \tilde{v}_i - a_2 \frac{m}{b} \tilde{Q}_i S + \frac{a_2}{b} \tilde{Q}_i 1
\] or,
\[
\tilde{q}_i = \left[ Q, \frac{m}{b} S - a_2 \frac{m}{b} 1 \right] \left[ \tilde{Q}_i \tilde{f}_i^v, \tilde{v}_i \right],
\]
where
\[
\tilde{f}_i^v = \frac{m}{b} \tilde{Q}_i S - a_2 \frac{m}{b} \tilde{Q}_i 1 - a_2 \frac{m}{b} \tilde{Q}_i \tilde{v}_i + \frac{a_2}{b} \tilde{Q}_i 1.
\]
We will show that the matrix \( f^v_i \) satisfies the stability condition
\[
1 \leq \frac{\langle f^v_i(U,U) \rangle}{\| U \|^2} \leq 2
\]
for arbitrary vector \( U \).
One property of the matrix \( f^v_i \) is
\[
\begin{align*}
\langle f^v_i(U,U) \rangle &= \| U \|^2 + \frac{m}{b} (S, U) - \frac{a_2}{m} (1, U) \| U \|^2 \leq \| U \|^2 + \frac{m}{b} \| S \|^2 \| U \|^2 \leq 2 \| U \|^2.
\end{align*}
\] (13)
and the other is
\[
\begin{align*}
\langle f^v_i(U,U) \rangle &\geq \| U \|^2 + \frac{m}{b} (S, U) - \frac{a_2}{m} (1, U) \| U \|^2 + \frac{1}{m} \| U \|^2 \| U \|^2 \\
&= \frac{m}{b} (S, U) - \frac{a_2}{m} (1, U) \| U \|^2 + \frac{m}{b} (S, U) - \frac{a_2}{m} (1, U) \| U \|^2 \\
&\geq \frac{m}{b} \| S - \frac{a_2}{m} 1 \|^2 \| U \|^2 = \| U \|^2. 
\end{align*}
\] (14)
From (13) and (14) we get
\[
1 \leq \max \lambda \leq 2.
\]
Where \( \lambda, v = 1, \ldots, m \) are the eigenvalues of the matrix \( f^v_i \).
Next, we calculate the eigenvalues of the matrix \( f^v_i \) to examine the Convergence Rate of the error in more detail. The matrix \( f^v_i \) can be written as
\[
\begin{pmatrix}
    z_{11} & z_{12} & \ldots & z_{1m} \\
    z_{21} & z_{22} & \ldots & z_{2m} \\
    \vdots & \vdots & \ddots & \vdots \\
    z_{m1} & z_{m2} & \ldots & z_{mm}
\end{pmatrix}
\]
Where \( m \) is the number of neurons in the output layer. The characteristic equation of this matrix is
\[
\gamma_2(\lambda) = \begin{vmatrix}
    \lambda - x_1 & -z_{12} & \ldots & -z_{1m} \\
    -z_{21} & \lambda - x_2 & \ldots & -z_{2m} \\
    \vdots & \vdots & \ddots & \vdots \\
    -z_{m1} & -z_{m2} & \ldots & \lambda - x_m \\
\end{vmatrix} = 0.
\]
Where
\[
p_{m1} = mu_i^2/b - 1/m, \text{and } p_{m1} = -mu_i^2/b - 1/m, \text{with } u_i = s_i - a_2/m.\]
By putting \( t = b(1 - \lambda)/(mu_i^2) - 1 \), we can reform \( \gamma_2(\lambda) \) as follows:
\[
\gamma_2(\lambda) = \frac{\prod_{i=1}^{m} (t_i - 1/m)^{-1-m} \deltaz}{m} L_z(\lambda)
\]
Generally, we have

\begin{align*}
\tau_1 & \quad a \quad \ldots \quad a \quad \ldots \quad a \\
\tau_2 & \quad a \quad \ldots \quad a \\
\vdots & \quad \vdots \\
a & \quad a \quad \ldots \quad t_{m-1} \quad a \\
a & \quad a \quad \ldots \quad t_m
\end{align*}

\begin{align*}
\Delta \theta & = \left( \begin{array}{cccccc}
t_1 & 1 & \ldots & 1 & 1 & 1 \\
t_2 & 1 & \ldots & 1 & 1 & 1 \\
& \vdots & \ddots & \vdots & \vdots & \vdots \\
t_m & 1 & \ldots & 1 & 1 & 1 \\
\end{array} \right) - \left( \begin{array}{cccccc}
t_1 & 1 & \ldots & 1 & 1 & 1 \\
t_2 & 1 & \ldots & 1 & 1 & 1 \\
& \vdots & \ddots & \vdots & \vdots & \vdots \\
t_m & 1 & \ldots & 1 & 1 & 1 \\
\end{array} \right) - \frac{1}{u_1} \left( \begin{array}{cccccc}
t_1 & 1 & \ldots & 1 & 1 & 1 \\
t_2 & 1 & \ldots & 1 & 1 & 1 \\
& \vdots & \ddots & \vdots & \vdots & \vdots \\
t_m & 1 & \ldots & 1 & 1 & 1 \\
\end{array} \right) \\
& = \sum_{v=1}^{m} (t_v - a) \sum_{i=1}^{m} \prod_{j=i+1}^{m} (t_j - a).
\end{align*}

According to the above result we can reform \( A, T_i, \) and \( T_{ij} \) as follows:

\begin{align*}
A & = \prod_{i=1}^{m} (t_v - 1) + \prod_{i=1}^{m} (t_{v-1} - 1) \\
T_i & = \prod_{i=1}^{m} (t_v - 1) + \sum_{j \neq i}^{m} \prod_{i=1}^{m} (t_v - 1) \\
T_{ij} & = \prod_{i=1}^{m} (t_v - 1) + \prod_{j \neq i}^{m} (t_v - 1)
\end{align*}

and

\begin{align*}
T_i & = \prod_{i=1}^{m} (t_v - 1) + \sum_{j \neq i}^{m} \prod_{i=1}^{m} (t_v - 1) \\
T_{ij} & = \prod_{i=1}^{m} (t_v - 1) + \prod_{j \neq i}^{m} (t_v - 1)
\end{align*}

But we have \( t_v = b(1 - \lambda)/(mu_v^2) - 1 \) and so it follows that

\begin{align*}
A & = (2 - \lambda)(1 - \lambda)^{m-2} \left( \frac{b}{m} \right) m \prod_{i=1}^{m} \frac{1}{u_v^2} \\
B & = -m(1 - \lambda) \frac{b}{m} m \prod_{i=1}^{m} \frac{1}{u_v^2} + m \prod_{i=1}^{m} \frac{1}{u_v^2} \prod_{j \neq i}^{m} (t_v - 1) - \prod_{i=1}^{m} \frac{1}{u_v^2} \prod_{j \neq i}^{m} (t_v - 1)
\end{align*}

which implies

\begin{align*}
(2 - \lambda)^2(1 - \lambda)^{m-2} & = 0,
\end{align*}

and finally we obtain

\begin{align*}
\lambda & = 2, 2, 1, 1, \ldots, 1.
\end{align*}

This theorem shows that the error after determining the weight \( w \) and the correction \( \Delta \theta \) between the hidden and output layers is not convergent.

Thus, we can conclude from the previous two sections that it is difficult to remove unit from the hidden layer after learning. On the other hand, we can learn the neural networks by constructive learning, the description of this algorithm will be in the following section.
7. RULE EXTRACTION FROM TRAINED NEURAL NETWORK USING GENE EXPRESSION PROGRAMMING

A method to extract comprehensible rules from trained artificial neural network using gene expression algorithm will be described in this section.

A constructive learning algorithm of three layered feedforward neural network [37,38,39] is described to train the network by supervised learning of the output layer and consequence conditions of the connections between the layers. Then GEP is carrying to obtain the rules which are bimodal with consistency. These consistencies can be described theoretically or observably. Consequently, the rule is the knowledge frame in system consistencies. Hence, the bimodal relation between rule and consistency is reversed in the form of relation between meaning and function. The technique described here uses GEP algorithms to generate rule patterns that progress into a set of suitable rules which clarify the reasons behind the classifications which neural networks make given different inputs. A variety number of experiments by using different datasets will execute, and the obtained rules compared with those consequent from an illustrative package of data mining on the same datasets.

This technique is proposed to improve the unambiguousness of trained neural network that achieve classification tasks. It employ the trained neural networks to generate a set of instances their class label, and then extracts symbolic rules from those instances by GEP. Experiments with variety configurations show that, it can extract rules with high reliability that will describe the main function of trained neural network, or rules with powerful generalization ability that are even better than that are extracted from the trained neural network in prediction.

8. EXPERIMENTAL RESULT

To evaluate our rule extraction algorithm, we used data sets such as:

8.1 Monk’s Task

In Monk’s Task, an instance is characterized by six attributes $a_1, ..., a_6$ which have two, three, or four discrete possible values. In our learning algorithms we encoded the problems into a 17 dimensional Boolean vector.

Monk #1: An example is mapped to a class ‘1’ if and only if $(a_1 = a_2) \lor (a_5 = 1)$. In this problem, the training set contains 124 patterns for training and all 432 possible patterns are used to calculate the classification accuracy of the network.

The number of inputs of the network for Monk’s problems is 17 binary inputs. Since the hidden units have a threshold, so we add another input represented by 1. Therefore the total inputs to the network under training are 18. We start the training of the network with one unit in the hidden layer. While the optimal weights of the network are obtained after adding another two hidden units. Hence the total number of the hidden units after training is 3.

The classification accuracy of the network is still the same as the original one.

The rules can be acquired by applying GEP to instances and the corresponding classes as classification problem. Before running GEP system, some parameters must be considered, the function set and terminal set for GEP are determined according to users knowledge for each problem. In this experiment, the function set is the logical operators And, Or, Not, The terminal set consists of the attribute names, relational operators ($=$, $\neq$), and attribute values of the data set being mined.

The obtained rules from running GEP are:

Rule1: if ($A1 = 1 \land A2 = 1$) then Class1

Rule2: if $A5 = 1$ then Class1

Rule3: if ($A1 = 3 \land A2 = 3$) then Class1

Rule4: if ($A2 = 2 \land A1 = 2$) then Class1 default class #2

9. RESULTS & DISCUSSION

To evaluate the performance of the learning algorithm we will use the commonly measures 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:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (18)$$

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

$$\text{Precision} = \frac{TP}{TP + FN} \quad (19)$$

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

$$\text{sensitivity} = \frac{TP}{TP + FP} \quad (20)$$

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

$$\text{specificity} = \frac{TN}{TN + FN} \quad (21)$$

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.

The performance evaluation of our model is compared with other learning models presented by Waikato Environment for Knowledge Analysis (WEKA). WEKA [40] is an open source software tool for machine learning.
software which consists of a collection of machine learning algorithms for data mining tasks such as REP Tree, Bayesian Networks, Radial Basis Function (RBF) Networks, and Single Conjunctive Rule Learner.

<table>
<thead>
<tr>
<th>Models</th>
<th>Accuracy (%)</th>
<th>Precision (%)</th>
<th>Sensitivity (%)</th>
<th>Specificity (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nave Bayesian</td>
<td>79.83</td>
<td>69.35</td>
<td>87.75</td>
<td>74.66</td>
</tr>
<tr>
<td>RBF Network</td>
<td>83.06</td>
<td>77.41</td>
<td>87.27</td>
<td>79.71</td>
</tr>
<tr>
<td>REP Tree</td>
<td>95.96</td>
<td>95.16</td>
<td>96.72</td>
<td>95.23</td>
</tr>
<tr>
<td>Bagging</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Our Model</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

The accuracy rate of the discovered rules was 100% for the problem, larger than the accuracy rate of the neural network and C4.5. In addition, neural networks via genetic algorithms have been used to extract rules [41] and the accuracy rate was 99.77% for monk1 problem. In the context of data mining this minor reduction in accuracy rate is a small price to pay for the large gain in the comprehensibility of the discovered knowledge.

10. CONCLUSION

Research work in the area of extracting rules from trained neural networks has witnessed much activity recently. However, the knowledge obtained by ANNs is generally incomprehensible for humans.

In this paper we have introduced a method to extract accurate and comprehensible rules from a neural network and gene expression programming (GEP).

In the first part, we use ANNs that achieve high classification accuracy which was trained by constructive learning algorithm.

In the second part, an approach has been used for a gene expression algorithm to extract comprehensible rules from trained neural network for classification problems. From the features of instances and the labels of their classes we can use GEP to encode the rules in the form of logic expression. The system has been evaluated on some public domain data sets. The computational results have shown that the system extracted a very compact, comprehensible rule set without overly reducing the accuracy rate, in comparison with the accuracy rate of the rule set discovered by other methods.

Widespread 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.

11. REFERENCES


[10] Löfström, T. and Odqvist, P. 2004 RULE EXTRACTION IN DATA MINING - FROM A META LEARNING PERSPECTIVE.


