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 to 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], etcare 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. REFNE 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

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

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

$$f(t) = \frac{1 - e^{-t}}{1 + e^{-t}}, \qquad (2)$$

$$g(t) = \frac{1}{1 + e^{-t}}, \qquad (3)$$

respectively. We write (1) as

$$y = g\left(Wf(Vx)\right), \qquad (4)$$

Where we set $x = (x_1, x_2, \dots, x_I, x_{I+1})^t$with $x_{I+1} = -1$, $y = (y_1, y_2, \dots, y_l)^t$,$V = (v_{jk})$and $W = (w_{ij})$.Moreover, $f(Vx)$means $\left(f(V_1.x), f(V_2.x), \dots, f(V_h.x)\right)^t$and $g\left(Wf(Vx)\right)$indicates $\left(g\left(W_1.f(Vx)\right), g\left(W_2.f(Vx)\right), \dots, g\left(W_l.f(Vx)\right)\right)^t$, where $V_j = \left(v_{j1}, v_{j2}, \dots, v_{j,n+1}\right)^t$and $W_i = (w_{i1}, w_{i2}, \dots, w_{ih})^t$. This network is shown in fig 1 Let$(x^v, y^v), v = 1,2, \dots, m$, 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

$$J(V, W) = \sum_{v=1}^{m} \|g^{-1}(y^v) - Wf(Vx^v)\|^2, \qquad (5)$$

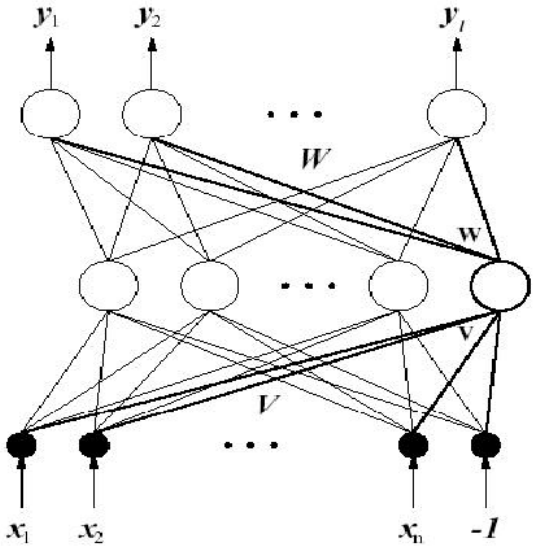

Fig 1: Neural network before removing one unit in the hidden layer.

where$g^{-1}(y^v) = \left(g^{-1}(y_1^v), g^{-1}(y_2^v), \dots, g^{-1}(y_l^v)\right)^t$is the inverse of the function, and $\|.\|$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 vector 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 $\tilde{V}$and $\tilde{W}$, respectively.

Then we can rewrite (5) as

$$J(V, W) = \sum_{v=1}^{m} \left\|g^{-1}(y^v) - \tilde{W} f(Vx^v) - wf(v.x^v)\right\|^2$$

Where
$Wf(Vx^v) = \tilde{W} f(Vx^v) + wf(v.x^v)andw$ is the removing weight vector.

The function can be written as follows:

$$J(V, W) = J\left(\tilde{V}, \tilde{W}\right) - 2\langle d, w\rangle + a\|w\|^2 (6)$$

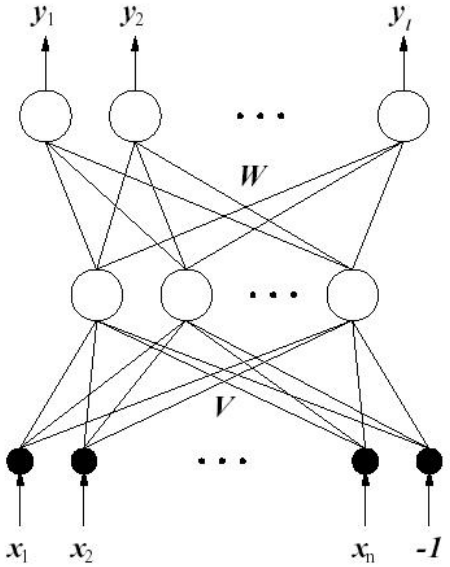

Fig 2: Neural network after removing one unit in the hidden layer.

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 $\langle .,. \rangle$indicates an inner product in R^l. When the vector v is fixed, the vector w which minimizes the error function by

$$w = \frac{d}{a}$$

So the error after removing a hidden unit can be expressed as

$$\begin{aligned}\tilde{c}^v &= g^{-1}(y^v) - \widetilde{W} f\left(\tilde{V} x^v\right) \\ &= g^{-1}(y^v) - W f(V x^v) + w f(v x^v) \\ &= c^v + \frac{d}{a} f(v.x^v).\end{aligned}$$

Furthermore, using the symbols $\tilde{C}_i^t = (c_i^1, c_i^2, \dots, c_i^m)$, $C_i^t = \left(c_i^1, c_i^2, \dots, c_i^m\right)$ and $S^t = (S_1, S_2, \dots, S_m)$ with $s_v = f(v.x^v)$, we have

$$\tilde{C}_i^t = C_i^t + \frac{1}{a} C_i^t S S^t . \qquad (7)$$

Moreover we can rewrite (7) as

$$\tilde{C}_i^t = C_i^t \left(I_m + \frac{1}{a} S S^t\right) = C_i^t \Gamma_1(v),$$

where I_m is the unit matrix and

$$\Gamma_1(v) = I_m + \frac{S S^t}{a}.$$

It is easy to show that the matrix $\Gamma_1(v)$ satisfies

$$1 \le \frac{\langle \Gamma_1(v) U, U\rangle}{\|U\|^2} \le 2$$

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 propriety of the matrix $\Gamma_1(v)$ is

$$\begin{aligned}\langle \Gamma_1(v) U, U\rangle &= \|U\|^2 + \frac{1}{a}\langle S, U\rangle^2 \\ &\le \|U\|^2 + \frac{1}{a}\|S\|^2\|U\|^2 = 2 \qquad (8)\end{aligned}$$

since $a = \|S\|^2$, and the other is

$$\langle \Gamma_1(v) U, U\rangle = \|U\|^2 + \frac{1}{a}\langle S, U\rangle^2 \ge \|U\|^2 \qquad (9)$$

From (8) and (9) we get

$$1 \le \frac{\langle \Gamma_1(v) U, U\rangle}{\|U\|^2} \le 2$$

which implies

$$1 \le max\ \lambda_v \le 2$$

Where $\lambda_v, v = 1, \dots, m$ are the eigenvalues of the matrix $\Gamma_1(v)$.

Next, we calculate the eigenvalues of the matrix $\Gamma_1(v)$ to examine the Convergence rate of the error in more detail. The matrix $\Gamma_1(v)$ can be written as

$$\Gamma_1(v) = \begin{pmatrix} z_{11} & -z_{12} & \dots & -z_{1m} \\ -z_{21} & z_{22} & \dots & -z_{2m} \\ \dots & \dots & \dots & \dots \\ -z_{m1} & -z_{m2} & \dots & z_{mm} \end{pmatrix},$$

Where $z_{ii} = 1 + s_i^2/a$, $z_{ij} = -s_i s_j/a$, and $z_{ji} = z_{ji}$. The characteristic equation of this matrix is

$$\gamma_1(\lambda) = \begin{vmatrix} \lambda - z_{11} & z_{12} & \dots & z_{1m} \\ z_{21} & \lambda - z_{22} & \dots & z_{2m} \\ \dots & \dots & \dots & \dots \\ z_{m1} & z_{m2} & \dots & \lambda - z_{mm} \end{vmatrix} = 0.$$

By putting $\eta_i = (1-\lambda)\, a/s_i^2$, we can reform $\gamma_1(\lambda)$ as follows:

$$\gamma_1(\lambda) = \frac{-\prod_{i=1}^m s_i^2}{a^m} L_1(\lambda) = 0,$$

where

$$L_1(\lambda) = \begin{vmatrix} \eta_1 + 1 & 1 & \dots & 1 \\ 1 & \eta_2 + 1 & \dots & 1 \\ \dots & \dots & \dots & \dots \\ 1 & 1 & \dots & \eta_m + 1 \end{vmatrix}.$$

The determinant $L_1(\lambda)$ can be transformed by Laplace theorem as

$$\begin{aligned}L_1(\lambda) &= \prod_{v=1}^m \eta_v + \sum_{i=1}^m \prod_{v=1}^{i-1} \eta_v \prod_{v=i+1}^m \eta_v \\ &= \prod_{v=1}^m \eta_v + \sum_{i=1}^m \frac{\prod_{v=1}^m \eta_v}{\eta_i}.\end{aligned}$$

Thus, we have

$$\begin{aligned}\gamma_1(\lambda) &= \frac{-\prod_{i=1}^m s_i^2}{a^m}\left(\prod_{v=1}^m \eta_v + \sum_{i=1}^m \frac{\prod_{v=1}^m \eta_v}{\eta_i}\right) \\ &= (2-\lambda)(1-\lambda)^{m-1} = 0.\end{aligned}$$

Hence the eigenvalues of this matrix are given by

$$\lambda = 2, 1, \dots, 1, 1.$$

This shows that the error after determining the weight w between the hidden and output layers is not convergent.

6. CONVERGENCE RATE IN A DESTRUCTIVE NEURAL NETWORK WITH THRESHOLDS IN THE OUTPUT LAYER

We consider the network with thresholds $\theta_i, i = 1, 2, \dots, l$, in its output layer before removing one unit in the hidden layer. In this case, we can write

$$y_i = g\left(\sum_{j=1}^h w_{ij}\, f\left(\sum_{k=1}^{n+1} v_{jk}\, x_k\right) - \theta_i\right), i = 1, 2, \dots, l$$

whose simple form is

$y = g\,(Wf(Vx) - \theta)$, where $\theta = (\theta_1, \theta_2, \dots, \theta_l)$
and $g\,(Wf(Vx) - \theta) = (g\,(W_1.f(Vx) - \theta_1),$
$g\,(W_2.f(Vx) - \theta_2), \dots, g\,(W_l.f(Vx) - \theta_l))$.

The error function that related to the present network takes the following form

$$J(V, W, \theta) = \sum_{v=1}^m \|g^{-1}(y^v) - W f(V x^v) + \theta\|^2.$$

We remove one unit from the hidden layer and represent removed weight vectors again by v and w. By removing one hidden unit, we can write the threshold vector θ *as* $\theta = \tilde{\theta} + \Delta\theta$.

The error function related to this network can be expressed as

$$J\left(\tilde{V}, \widetilde{W}, \tilde{\theta}\right) = \sum_{v=1}^m \left\|g^{-1}(y^v) - \widetilde{W} f\left(\tilde{V} x^v\right) + \tilde{\theta}\right\|^2,$$

where we have used again the symbols $\tilde{V} = (V, v)$ and $\widetilde{W} = (W, w)$. The same procedure as in the previous section will be applied in order to determine w and $\Delta\theta$ so that $J\left(\tilde{V}, \widetilde{W}, \tilde{\theta}\right)$ is minimum. Since $Wf(Vx^v) = \widetilde{W} f(Vx^v) + wf(v.x^v)$ and $\theta = \tilde{\theta} + \Delta\theta$, we can decompose the error function $J\left(\tilde{V}, \widetilde{W}, \tilde{\theta}\right)$ as

$$\begin{aligned}J\left(\tilde{V}, \widetilde{W}, \tilde{\theta}\right) &= J(V, W, \theta) + 2\sum_{v=1}^m f(v.x^v)\langle q^v + \Delta\theta, w\rangle \\ &- \sum_{v=1}^m f^2(v.x^v)\|w\|^2 - 2\sum_{v=1}^m \langle \Delta\theta, q^v\rangle - \sum_{v=1}^m \|\Delta\theta\|^2,\end{aligned}$$

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

$$w = \frac{md_1 - a_2 d_2}{b}, \tag{10}$$

$$\Delta\theta = \frac{a_2 d_1 - a_1 d_2}{b}, \tag{11}$$

where

$$a_1 = \sum_{v=1}^{m} f^2(v.x^v),$$
$$a_2 = \sum_{v=1}^{m} f(v.x^v),$$
$$d_1 = \sum_{v=1}^{m} f(v.x^v)\, q^v,$$
$$d_2 = \sum_{v=1}^{m} q^v,$$
$$b = ma_1 - a_2^2.$$

Now we consider the convergence rate of the error as in the previous section.
Since

$$\begin{aligned}\tilde{q}^v &= g^{-1}(y^v) - \tilde{W}f(\tilde{V}x^v) + \tilde{\theta} \\ &= g^{-1}(y^v) - Wf(Vx^v) + wf(v.x^v) + \theta - \Delta\theta \\ &= q^v + wf(v.x^v) - \Delta\theta,\end{aligned}$$

we can write the error at *i*-th component in the output layer as

$$\tilde{q}_i^v = q_i^v + w_i f(v.x^v) - \Delta\theta_i. \tag{12}$$

From (10), the *i*-th component of *w* can be written as

$$\begin{aligned}w_i &= \frac{m}{b}d_{1i} - \frac{a_2}{b}d_{2i} \\ &= \frac{m}{b}\sum_{v=1}^{m} s_v q_i^v - \frac{a_2}{b}\sum_{v=1}^{m} q_i^v \\ &= \frac{m}{b}\,{}^tQ_i S - \frac{a_2}{b}\,{}^tQ_i 1 \\ &= {}^tQ_i\left[\frac{m}{b}S - \frac{a_2}{b}1\right].\end{aligned}$$

Similarly from (11), we can also write $\Delta\theta_i$ as

$$\Delta\theta_i = {}^tQ_i\left[\frac{a_2}{b}S - \frac{a_2}{b}1\right],$$

Where ${}^t\tilde{Q}_i = (\tilde{q}_i^1, \tilde{q}_i^2, \dots, \tilde{q}_i^m)$, ${}^tQ_i = (q_i^1, q_i^2, \dots, q_i^m)$, ${}^tS = (s_1, s_2, \dots, s_m)$, and ${}^t1 = (1, 1, \dots, 1)$ with $s_v = f(v.x^v)$.Hence, (12) can be rewritten as

$$\tilde{q}_i^v = q_i^v + \frac{m}{b}\,{}^tQ_i S s_v - \frac{a_2}{b}\,{}^tQ_i 1 s_v - \frac{a_2}{b}\,{}^tQ_i S + \frac{a_1}{b}\,{}^tQ_i 1$$

or,

$${}^t\tilde{Q}_i = {}^tQ_i\left[I_m + \frac{m}{b}S\,{}^tS - \frac{a_2}{b}S1\,{}^tS - \frac{a_2}{b}\,{}^t1 + \frac{a_1}{b}1\,{}^t1\right] = {}^tQ_i\Gamma_2(v),$$

where

$$\Gamma_2(v) = I_m + \frac{m}{b}S\,{}^tS - \frac{a_2}{b}S1\,{}^tS - \frac{a_2}{b}\,{}^t1 + \frac{a_1}{b}1\,{}^t1.$$

We will show that the matrix $\Gamma_2(v)$satisfies the stability condition

$$1 \le \frac{\langle\Gamma_2(v)U,U\rangle}{\|U\|^2} \le 2$$

for arbitrary vector U.

One propriety of the matrix $\Gamma_2(v)$ is

$$\begin{aligned}\langle\Gamma_2(v)U,U\rangle &= \|U\|^2 + \frac{m}{b}\langle S,U\rangle^2 - \frac{2a_2}{b}\langle S,U\rangle\langle 1,U\rangle \\ &\quad + \frac{a_1}{b}\langle 1,U\rangle^2 \\ &\le \|U\|^2 + \frac{m}{b}\|S\|^2\|U\|^2 \\ &\le 2\|U\|^2.\end{aligned} \tag{13}$$

and the other is

$$\begin{aligned}\langle\Gamma_2(v)U,U\rangle &\ge \|U\|^2 + \frac{m}{b}\left(\langle S,U\rangle - \frac{a_2}{m}\langle 1,U\rangle\right)^2 \\ &\quad + \frac{1}{m}\|1\|^2\|U\|^2 \\ &= \frac{m}{b}\left(\langle S,U\rangle - \frac{a_2}{m}\langle 1,U\rangle\right)^2 + \frac{m}{b}\left(\langle S - \frac{a_2}{m}1, U\rangle\right)^2 \\ &\ge \frac{m}{b}\left\|S - \frac{a_2}{m}1\right\|^2\|U\|^2 \\ &= \|U\|^2.\end{aligned} \tag{14}$$

From (13) and (14) we get

$$1 \le max\ \lambda_v \le 2,$$

Where $\lambda_v, v = 1, \dots, m$ are the eigenvalues of the matrix $\Gamma_2(v)$.

Next, we calculate the eigenvalues of the matrix $\Gamma_2(v)$to examine the Convergence rate of the error in more detail. The matrix $\Gamma_2(v)$can be written as

$$\Gamma_2(v) = \begin{pmatrix} z_{11} & z_{12} & \cdots & z_{1m} \\ z_{21} & z_{22} & \cdots & z_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ z_{m1} & z_{m2} & \cdots & z_{mm} \end{pmatrix},$$

Where$z_{ii} = 1 + \frac{ms_i^2}{b} - \frac{2a_2 s_i}{b} + \frac{a_1}{b}$,$z_{ij} = \frac{ms_i s_j}{b} - \frac{a_2(s_i + s_j)}{b} + \frac{a_1}{b}$, and $z_{ij} = z_{ji}$.The characteristic equation of this matrix is

$$\gamma_2(\lambda) = \begin{vmatrix} \lambda - z_{11} & -z_{12} & \cdots & -z_{12} \\ -z_{21} & \lambda - z_{11} & \cdots & -z_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ -z_{m1} & -z_{m2} & \cdots & \lambda - z_{mm} \end{vmatrix}$$

$$= \begin{vmatrix} \lambda - 1 + p_{11} & p_{12} & \cdots & p_{1m} \\ p_{21} & \lambda - 1 + p_{22} & \cdots & p_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ p_{m1} & p_{m2} & \cdots & \lambda - 1 + p_{mm} \end{vmatrix}$$

$$= \begin{vmatrix} \lambda - 1 + p_{11} & p_{12} & \cdots & p_{1m} & 1 \\ p_{21} & \lambda - 1 + p_{22} & \cdots & p_{2m} & 1 \\ \cdots & \cdots & \cdots & \cdots & \\ p_{m1} & p_{m2} & \cdots & \lambda - 1 + p_{mm} & 1 \\ 0 & 0 & \cdots & 0 & 1 \end{vmatrix} = 0,$$

Where
$p_{ii} = -mu_i^2/b - 1/m$, and$p_{ij} = -mu_i u_j/b - 1/m$, with $u_i = s_i - a_2/m$.By putting $t_v = b(1-\lambda)/(mu_v^2) - 1$, we can reform $\gamma_2(\lambda)$ as follows:

$$\gamma_2(\lambda) = \frac{\prod_{i=1}^{m} u_i^2}{m}\left(\frac{m}{b}\right)^{m-1} L_2(\lambda)$$

with

$$L_2(\lambda)=\begin{vmatrix} t_1 & 1 & 1 & \dots & 1 & 1 & \frac{1}{u_1} \\ 1 & t_2 & 1 & \dots & 1 & 1 & \frac{1}{u_2} \\ \dots & & & & & & \\ 1 & 1 & 1 & \dots & t_{m-1} & 1 & \frac{1}{u_{m-1}} \\ 1 & 1 & 1 & \dots & 1 & t_m & \frac{1}{u_m} \\ \frac{1}{u_1} & \frac{1}{u_2} & \frac{1}{u_3} & \dots & \frac{1}{u_{m-1}} & \frac{1}{u_m} & \frac{-m^2}{b} \end{vmatrix}$$

Using the Laplace theorem, we get

$$\gamma_2(\lambda)=\frac{\prod_{i=1}^{m}u_i^2}{m}\left(\frac{m}{b}\right)^{m-1}\left[-\frac{m^2}{b}A+B\right], \qquad (15)$$

where

$$A=\begin{vmatrix} t_1 & 1 & \dots & 1 \\ 1 & t_2 & \dots & 1 \\ \dots & & & \\ 1 & 1 & \dots & t_m \end{vmatrix}$$

and

$$B=-\sum_{i=1}^{m}\frac{1}{u_i^2}T_i+\sum_{i=1}^{m}\sum_{\substack{j=1\\ j\neq i}}^{m}\frac{1}{u_iu_j}T_{ij}$$

with

$$T_i=\begin{vmatrix} t_1 & 1 & \dots & 0 & \dots & 1 \\ 1 & t_2 & \dots & 0 & \dots & 1 \\ \dots & & & & & \\ 0 & 0 & \dots & 1 & \dots & 0 \\ \dots & & & & & \\ 1 & 1 & \dots & 0 & \dots & t_m \end{vmatrix},$$

$$T_{ij}=\begin{vmatrix} t_1 & 1 & \dots & 0 & \dots & 1 & \dots & 1 \\ 1 & t_2 & \dots & 0 & \dots & 1 & \dots & 1 \\ \dots & & & & & & & \\ 0 & 0 & \dots & 1 & \dots & 0 & \dots & 0 \\ \dots & & & & & & & \\ 1 & 1 & \dots & 0 & \dots & 1 & \dots & 1 \\ \dots & & & & & & & \\ 1 & 1 & \dots & 0 & \dots & 1 & \dots & t_m \end{vmatrix}$$

Generally, we have

$$\begin{vmatrix} t_1 & a & \dots & a & a \\ a & t_2 & \dots & a & a \\ \dots & & & & \\ a & a & \dots & t_{m-1} & a \\ a & a & & a & t_m \end{vmatrix}=\begin{vmatrix} t_{1-a} & 0 & \dots & 0 & a \\ a & t_2 & \dots & 0 & a \\ -t_2 & -a & & & \\ \dots & & & & \\ 0 & 0 & \dots & t_{m-1}-a & a \\ 0 & 0 & \dots & a-t_m & t_m \end{vmatrix}$$

$$=\prod_{v=1}^{m}(t_v-a)+a\sum_{i=1}^{m}\prod_{v=1}^{i-1}(t_v-a)\prod_{v=i+1}^{m}(t_v-a).$$

According to the above result we can reform A,T_i,and$T_i$as follows:

$$A=\prod_{v=1}^{m}(t_v-1)+\sum_{i=1}^{m}\prod_{v=1}^{i-1}(t_v-1)\prod_{v=i+1}^{m}(t_v-1),$$

$$T_i=\frac{\prod_{v=1}^{m}(t_v-1)}{(t_i-1)}+\sum_{\substack{j=1\\ j\neq i}}^{m}\frac{\prod_{v=1}^{m}(t_v-1)}{(t_j-1)(t_i-1)},$$

and

$$T_{ij}=\frac{\prod_{v=1}^{m}(t_v-1)}{(t_i-1)(t_j-1)}.$$

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

$$A=(2-\lambda)(1-\lambda)^{m-1}\left(\frac{b}{m}\right)^{m}\prod_{v-1}^{m}\frac{1}{u_v^2} \qquad (16)$$

and

$$B=-\sum_{i=1}^{m}\frac{1}{u_i^2}\frac{\prod_{v=1}^{m}(t_v-1)}{(t_i-1)}-\sum_{i=1}^{m}\frac{1}{u_i^2}\sum_{\substack{j=1\\ j\neq i}}^{m}\frac{\prod_{v=1}^{m}(t_v-1)}{(t_j-1)(t_i-1)}+\sum_{i=1}^{m}\sum_{\substack{j=1\\ j\neq 1}}^{m}\frac{1}{u_iu_j}\frac{\prod_{v=1}^{m}(t_v-1)}{(t_j-1)(t_i-1)}$$

$$=\prod_{v=1}^{m}\frac{1}{u_v^2}\left(\frac{b}{m}(1-\lambda)\right)^{m-2}\left[-m(1-\lambda)\frac{b}{m}-\sum_{i=1}^{m}\sum_{\substack{j=1\\ j\neq i}}^{m}u_j^2+\sum_{i=1}^{m}\sum_{\substack{j=1\\ j\neq 1}}^{m}u_iu_j\right]$$

$$=K\left[-(1-\lambda)b-b+\frac{b}{m}-\frac{b}{m}\right]=-bK(2-\lambda), \qquad (17)$$

where

$$K=\left(\frac{b}{m}(1-\lambda)\right)^{m-2}\prod_{v=1}^{m}\frac{1}{u_v^2}.$$

By substituting (16) and (17) into (15), we have

$$\gamma_2(\lambda)=-K(2-\lambda)^2\left(\frac{m}{b}\right)^{m-2}\frac{1}{m}\prod_{i=1}^{m}u_i^2=0$$

which implies

$$(2-\lambda)^2(1-\lambda)^{m-2}=0,$$

and finally we obtain

$$\lambda=2,2,1,1\dots,1.$$

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 $a_1, \ldots., a_6$ which have two, three, or four discrete possible values. Itmeans, for our learning algorithms we encoded the problems into a17 dimensional Boolean vector.
Monk #1: An example is mapped to a class "1" if and only if $(a_1 = a_2)$or $(a_5 = 1)$. 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 (=, #), and attribute values of the data set being mined.

The obtained rules from running GEP are:

Rule1: if $(\mathrm{A1} = 1 \wedge \mathrm{A2} = 1)$ then Class1

Rule2: if A5 = 1 then Class1

Rule3: if $(\mathrm{A1} = 3 \wedge \mathrm{A2} = 3)$then Class1

Rule4: if $(\mathrm{A2} = 2 \wedge \mathrm{A1} = 2)$then Class 1

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:

$$Accuraccy = \frac{TP + TN}{TP + TN + FP + FN} \qquad (18)$$

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

$$Precision = \frac{TP}{TP + FP} \qquad (19)$$

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

$$sensitivity = \frac{TP}{TP + FN} \qquad (20)$$

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

$$\text{specificity} = \frac{TN}{TN + FP} \qquad (21)$$

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 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 1. The performance measures of various models for Monk 1.

Models	Accuracy (%)	Precision (%)	Sensitivity (%)	Specificity (%)
Nave Bayesian	79.83	69.35	87.75	74.66
RBF Network	83.06	77.41	87.27	79.71
REP Tree	95.96	95.16	96.72	95.23
Bagging	100	100	100	100
Our Model	100	100	100	100

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

[1] Baesens, B., Setiono, R.; Mues, C., Vanthienen, J. 2003Using neural network rule extraction and decision tables for credit-risk evaluation. *Manage. Sci., 49,* 312-329.

[2] Jacobsson, H. 2005Rule extraction from recurrent neural networks: A taxonomy and review. *Neural Comput.,17,* 1223-1263.

[3] Kahramanli, H.; Allahverdi, N. 2009Rule extraction from trained adaptive neural networks using artificial immune systems. *Expert Syst. Appl., 36,* 1513-1522.

[4] Setiono, R.; Baesens, B.; Mues, C. 2009A note on knowledge discovery using neural networks and its application to credit screening, *Eur. J. Operation. Res., 192,* 326-332.

[5] Tickle, A.B., Andrews, R.; Golea, M.; Diederich, J. 1998The truth will come to light: Directions and challenges in extracting the knowledge embedded within trained artificial neural networks. *IEEE Trans. Neural Netw.,9,* 1057-1067.

[6] Setiono, R.; Leow, W.K. 2000FERNN: An algorithm for fast extraction of rules from neural networks.*Appl. Intell.,12,* 15-25.

[7] Baesens, B. T.,Gestel,V. S.,Viaene, M. Stepanova, J. Suykens and Vanthienen, J. 2003Benchmarking state of the art classification algorithms for credit scoring, vol. 56, no. 6, 627-635.

[8] Johansson, U., Konig, R. and Niklasson, L. 2005Automatically balancing accuracy and comprehensibility in predictive modeling.

[9] Thrun,S. e. a.,1991 The MONK's problems: A performance comparison of different learning algorithms, *Pittsburgh*, 91-197.

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

[11] HumarK.,NovruzA., 2009"Rule extraction from trained adaptive neural networks using artificial immune systems", Expert Systems with Applications 36, 1513–1522.

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

[13] Towell, G. and Shavlik,J. 1993TheExtraction of Refined Rules From Knowledge Based Neural Networks, Machine Learning, Vol. 131, 71-101.

[14] Setiono, R.,Wee,K. H. and Zurada,M. J. 2002 Extraction of Rules from artificial neural network for nonlinear regression, IEEE Transaction Neural Networks, Vol. 23 No. 23, 564-577.

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

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

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

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

[19] OlcayB., 2002Extracting decision tree from trained neural networks, ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 456-461.

[20] Zhou, Z. H., Jiang, Y., Yang, Y. B. and Chen, S.F. 2003Extracting neural networks from trained neural network Ensembles, AI Communications, Vol. 16 No.1, 3-15.

[21] Garcez, A., d'Avila,S., Broda, K., Gabbay, D.M. 2001Symbolic knowledge extraction from trained neural networks: A sound approach, Artificial Intelligence, Vol. 125, 155-207.

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

[23] Bojarczuk,C. C., Lopes, H. S., Freitas, A.A. and Michalk., 2004 A constrained-syntax genetic programming system for discovering classification rules: application to medial database. Artificial Intelligence in Medicine, Volume 30, Issue 1, 27-48.

[24] Mitchell,M. MIT Press, 1996an Introduction to Genetic Algorithms.

[25] Dudoit,S., Yang, Y. H., Callow, M. J. and Speed,T. P. 2002Statistical Methods for Identifying Differentially Expressed Genes in Replicated cDNA Microarray Experiments, StatisticaSinica, vol. 12, pages 111-139.

[26] Reiner,A., Yekutieli, D. and Benjamini, Y. 2003Identifying ifferentially Expressed Genes Using False Discovery Rate Controlling Procedures, Bioinformatics, vol. 19, no. 3 , pages 368-375.

[27] Efron,A.,Tibshirani,R.,Storey,J. D. and Tusher,V. 2001Empirical Bayes Analysis of a Microarray Experiment, J. Am. Statistical Assoc., vol. 96, pages 1151-1160.

[28] Creighton,C. and Hanash,S. 2003Mining Gene Expression Databases for Association rules, Bioinformatics, vol. 19, no. 1, 79-86.

[29] Brown, M. P. S. et al., 2000Knowledge-Based Analysis of Microarray Gene Expression Data by Using Support Vector Machines,Proc. Nat'l Academy of Sciences USA, vol. 97, no. 1, pages 262-267.

[30] Jiang, D., Tang,C. and Zhang, A. 2004Cluster Analysis for Gene Expression Data: A Survey, IEEE Trans. Knowledge and Data Eng., vol. 16, no. 11, (Nov.2004) pages 1370-1386.

[31] Pan,F. et al.,2003 "Carpenter: Finding Closed Patterns in Long Biological Datasets," Proc. Ninth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '03).

[32] Cong ,G. et al., 2004Farmer: Finding Interesting Rule Groups in Microarray Datasets, Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '04).

[33] Cong , G. et al., 2005Mining Top-k Covering Rule Groups for Gene Expression Data, Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '05).

[34] Wang,H. C. and Lee,Y.S. 2005 Gene Network Prediction from Microarray Data by Association Rule and Dynamic Bayesian Network, Proc. Int'l Conf. Computational Science and Its Applications (ICCSA), pages 309-317.

[35] Shang,X. Q., Zhao, Q. and Li,Z. H. 2009Mining High-Correlation Association Rules for Inferring Gene Regulation Networks, Proc. 11th Int'l Conf. Data Warehousing and Knowledge Discovery (DaWaK '09),pages 244-255.

[36] Pandey,G., Atluri, G.,Steinbach,M. and Kumar, V. 2008Association Analysis Techniques for Discovering Functional Modules from Microarray Data, Nature Precedings.

[37] Marghny. H. M, Minamoto,T. and Niijima,K. 1990"Rules extraction by constructive learning of neural networks and hidden unit clustering", Lecture Notes in Artificial Intelligence, 1721, Springer, Proc. of the Second International Conference on Discovery Science, pages 343-344.

[38] Marghny. H. M, and Niijima,K. 2000"Extracting rules from neural network by removing unnecessary connections", Proc. of the Second ICSC Symposium on Neural Computation, pages 322-328.

[39] Marghny. H. M, 2011 Rules extraction from constructively trained neural networks based on genetic algorithms

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

[41] Marghny. H. M, and Niijima,K., 2000 " Redundant connections effect on the error rate for the neural networks", Proc. of the Seventh International Conference on Neural Computation Processing, pages 981-985.