UNSYS digital

International Journal of Robotics and Mechatronics

Classification Process Using Hybrid Model of Rough Neural Networks and Gene Expression Programming

Yasser Fouad

Alexandria University, Egypt

***Abstract*—**The paper will introduce a new model of rough neural networks based on learning using gene expression programming for classification support. The Objective of gene expression programming rough neural networks approach is to obtain new classified data with minimum error in training and testing process. Starting point of gene expression programming rough neural networks approach is an information system and the output from this approach is a structure of rough neural networks which is including the weights and thresholds.

***Keywords*—**Rough sets, gene expression programming, rough neural networks, classification.

I. Introduction

The term "classification" concerns any context in which some decision is taken or a forecast is made on the basis of currently available knowledge or information. The main basic function executed by the human brain is the classification. In the moments of life the human imposes classification among two or more objects. The human can analyze objects using some characteristics to perceive the differences and similarities. So, it is possible to classify animals as friendly or dangerous, healthy eaten plant or not, etc.

In this paper, we present an algorithm based on three methods. Firstly: Rough set theory (RST) that can be used for decision rule generation and data reduction from a set of observed data set [2][8][11]. Secondly: Gene expression programming (GEP) [3][4][6] which is a powerful evolutionary method derived from genetic programming (GP) [12] for knowledge discovery and learning model. Thirdly: Rough neural networks to maximize the classification accuracy percentage (minimize classification error) [1][5]. The proposed algorithm can be applied to many practical areas such as solving system regression problems and fitting economic prediction curves.

The main issue tackled in this paper is auto-adaptation occurred in this method that generates the behaviors from the study of local interactions between genes and the environment. Some key issues associated with the understanding and representations of such emerging behaviors in such system are introduced. A classification algorithm in this paper has been implemented and included in the Rosetta software [13] and GeneXproTools software [14]. Two real-world examples are presented in the experiment in this paper to solve classification problem.

The content of this paper is organized as follows. In Section II, a basic methods will be introduced, namely, rough sets theory (RST), gene expression programming (GEP) technique and rough neural networks. Detailed explanation of (GEP-RNN) approach for classification problem and some illustrative examples are presented in Section III. Experimental methodology and result obtained are providing in Section IV. Finally, conclusion is given in Section V.

II. Basic Methods

A. Rough Set Theory

Rough Set Theory (RST) was developed by Z. Pawlak and his coworkers in the early 1980s [9], it is a widely recognized data analysis method to deal with vagueness and uncertainty of data.

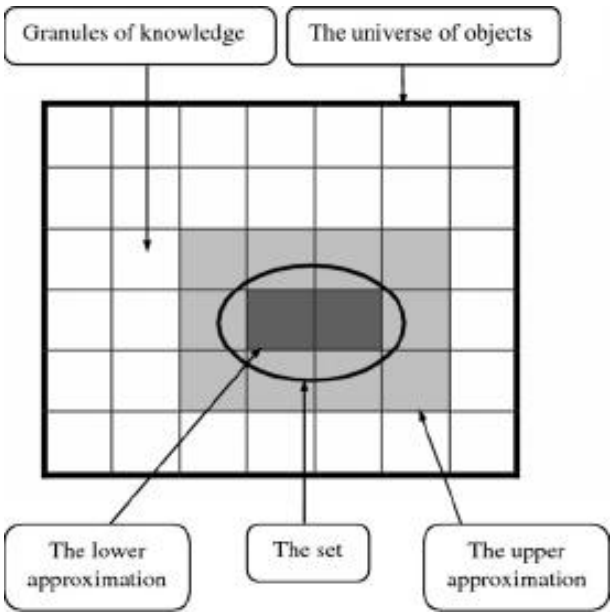

Figure 1 A rough sets

Corresponding author: Yasser Fouad (e-mail: y.fouad@sci.alexu.edu.eg).
This paper was submitted on December 4, 2015; revised on December 20, 2015; and accepted on December 27, 2015.

Rough set theory assumes that every objects of the universe have some information, objects characterized by same information are indiscernible (similar) in view of the available information about them. The indiscernibility relation generated in this way is the mathematical basis of the rough set theory. Any set of all indiscernible (similar) objects is called elementary set and form basic granule (atom) of knowledge about the universe. Any union of some elementary sets is referred to as crisp (precise) set, otherwise; a set is rough (imprecise, vague) (see **Figure 1**).

Suppose we are given a finite set of objects U called the universe, and a binary relation I over U called the indiscernibility relation. Any vague concept is characterized by pair of precise concepts called the lower and the upper approximation of the vague concept. The lower approximation consists of all objects which surely belong to the concept while the upper approximation contains all objects which possible belong to the concept. Obviously the difference between the upper and the lower approximation constitute the boundary region of the vague concept (see **Figure 1**).

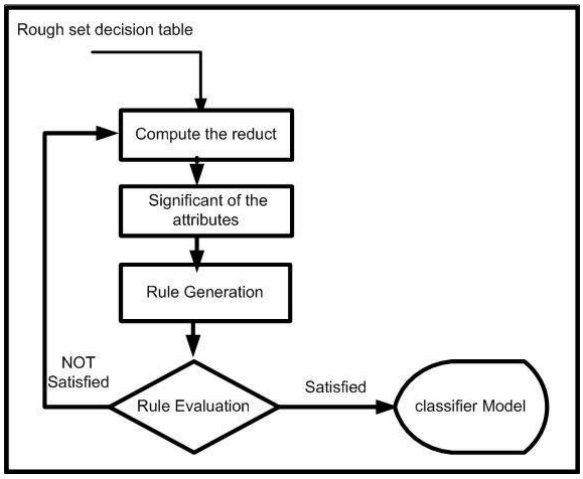

Figure 2 Rough sets algorithm for classification

Approximations are two basic operations in the rough set theory. Let $X \subseteq U$

$$I_*(X) = \{x \in U: I(x) \subseteq X\} \quad (1)$$
$$I^*(X) = \{x \in U: I(x) \cap X \neq \emptyset\} \quad (2)$$

where $I_*(X)$ and $I^*(X)$ called the I-lower and the I-upper approximation of X, respectively. $I(x)$ denote the set of all objects indiscernible with X. The boundary region of X is the set

$$BN_I(X) = I^*(X) - I_*(X) \quad (3)$$

Classification problem which can be solved using the rough set approach is obtained by the following Algorithm 1 (see **Figure 2**).

Algorithm 1:

Step 1. Description of objects in terms of attribute values

Step 2. Dependencies (full or partial) between attributes

Step 3. Reduction of attributes

Step 4. Decision rules generation □

B. Gene Expression Programming (GEP)

Gene expression programming (GEP) was invented by Ferreira 2001 [6]. It is a full-fledged genotype/phenotype system that evolves computer programs encoded in linear chromosomes of fixed length. The main players in GEP are only two: the chromosomes and the expression trees (ETs) being the latter the expression of the genetic information encoded in the former. As in nature, the process of information decoding is called translation and this translation implies obviously a kind of code and a set of rules. The genetic code is very simple one-to-one relationship between the symbols of the chromosome and the functions or terminals they represent. The rules are also very simple; they determine the spatial organization of the functions and terminals in the ETs. In GEP there are therefore two languages: the language of genes and the language of ETs. In GEP, we know the rules that determine the structure of ETs and their interactions; it is possible to infer immediately the phenotype given the sequence of a gene and vice versa. This system is called Karva language. The structural and functional organization of GEP chromosomes is based on how the chromosomes are translated into expression trees and how the chromosomes function as genotype and the expression trees as phenotype. In GEP, the genome or chromosome consists of a linear symbolic string of fixed length composed of one or more genes. The structural organization of GEP genes is better understood in terms of open reading frames (ORFs). In GEP the start site is always the first position of a gene and the termination point not always coincides with the last position of a gene. It is common for GEP genes to have non-coding regions downstream of the termination point. Consider the simple mathematical expression a*b+c. This can be encoded as an expression tree of the form in **Figure 3**.

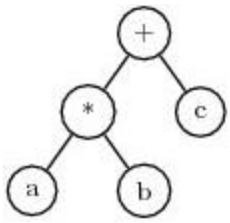

Figure 3 Expression tree of a mathematical expression (a*b + c)

To convert an expression tree to the Karva notation start at the left-most symbol in the top line of the tree and scan symbols left-to-right and top-to-bottom. Each time a symbol is encountered add it to the K-expression in left-to-right order. When there are no more symbols on a line advance to the left end of the following line. Using this method, the tree shown above is converted to the K-expression: "+∗cab".

Note that "+" is the first symbol found on the first line at the end of that line. Scanning begins on the second line and finds "∗" followed by "c". It then starts with the third line and finds "a" and "b".

A gene has two sections the head and the tail. The head is used to encode functions for the expression. The tail is a reservoir of extra terminal symbols that can be used if there are not enough terminals in the head to provide arguments for the functions. Thus the head can contain functions, variables and constants but the tail can contain only variables and constants (i.e. terminals). The number of symbols in the head of a gene is specified as a parameter for the analysis. The number of symbols in the tail is determined by the equation

$$t = h * (\text{MaxArg} - 1) + 1 \quad (4)$$

where t is the number of symbols in the tail, h is the number of symbols in the head and MaxArg is the maximum number of arguments required by any function that is allowed to be used in the expression. For example, if the head length is 6 and the allowable set of functions consists of binary operators (+, −, ∗, /) then the tail length is: $t = 6 * (2 - 1) + 1 = 7$. The purpose of the tail is to provide a reservoir of terminal symbols (variables and constants) that can be used as arguments for functions in the head if there are not enough terminals in the head.

The success of a problem greatly depends on the way the fitness function is designed. The goal must be clearly and correctly defined in order to make the system evolve in that direction. In order for a population to improve from generation to generation, innovations must occur that cause some individuals to have qualities never before seen. These innovations come about from

- Mutation – Simple mutation just replaces symbols in genes with replacement symbols. Symbols in the heads of genes can be replaced by functions or terminals (variables and constants). Symbols in the tail sections can be replaced only by terminals.
- Inversion – Inversion reverses the order of symbols in a section of a gene.
- Transposition – Transposition selects a group of symbols and moves the symbols to a different position within the same gene. Gene transposition moves entire genes around in the chromosome.
- Recombination – During recombination, two chromosomes are randomly selected, and genetic material is exchanged between them to produce two new chromosomes.

C. Rough Neural Networks

Rough neural networks (see **Figure 4**) consist of a combination of rough neurons and conventional neurons. Rough neurons use pairs of upper and lower bounds as values for input and output. The concept of upper and lower bound has been used in a variety of applications in artificial intelligence [1][5]. The errors in estimation from rough neural network models are significantly lower than the conventional neural network model [7][10]. Moreover, the addition of rough neurons in hidden layer seems to improve the prediction performance.

A rough neuron Rn is a pair of usual rough neurons Rn = (Un, Ln) where Un and Ln are the upper rough neuron and the lower rough neuron, respectively. It has three types of connections:

- Connection 1: Input-Output connection to Un
- Connection 2: Input-Output connections to Ln
- Connection 3: Connection between Un and Ln

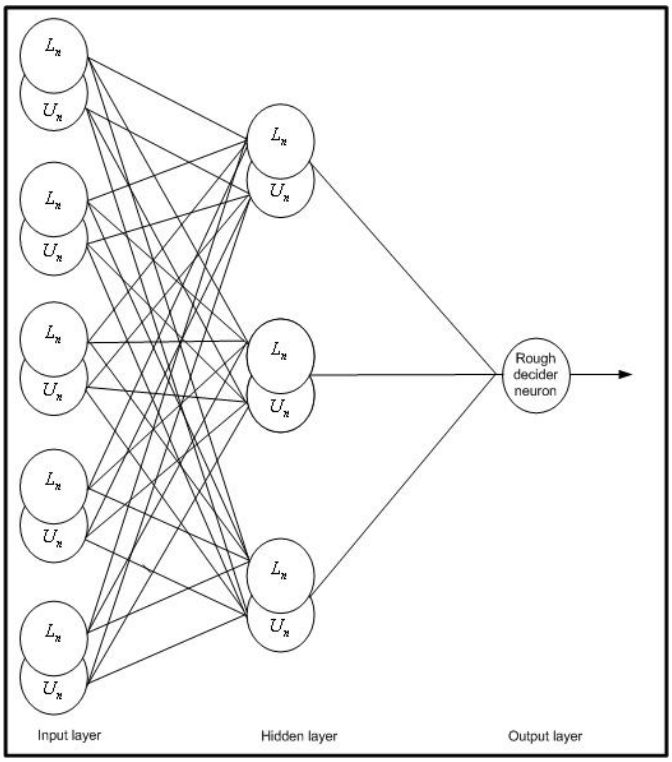

Figure 4 Rough neural networks structure

The rough neural network classification algorithm is described as in Algorithm 2.

Algorithm 2:

Input: Set of neurons inputs and the set of rules.

Processing:

Step 1. Compute the input upper/ lower rough neuron by the equation.

$$Ir_{Ln} = \sum_{j=1}^{n} w_{Lnj} On_j \quad (5)$$

$$Ir_{Un} = \sum_{j=1}^{n} w_{Unj} On_j \quad (6)$$

where (Ir_{Ln},Ir_{Un}) are the input lower/upper rough neuron r.

Step 2. Build rough neural networks structure.

Step 3. Compute the relative error.

Step 4. Training rough neural networks.

Step 5. Repeat Step 4 until the error becomes minimum.

Step 6. Compute the output upper/ lower rough neuron by the equations.

$$Or_{Ln} = \min\left(f\left(Ir_{Ln}\right), f\left(Ir_{Un}\right)\right) \quad (7)$$

$$Or_{Un} = \max\left(f\left(Ir_{Ln}\right), f\left(Ir_{Un}\right)\right) \quad (8)$$

where (Or_{Ln},Or_{Un}) are the output lower/upper rough neuron r.

Step 7. Compute the output of the rough neuron (O_{rn}) will be computed using the following equation:

$$O_{rn} = \frac{Or_{Un} - Or_{Ln}}{\text{average}\left(Or_{Un}, Or_{Ln}\right)} \quad (9)$$

Step 8. Return the structure with minimum error.

Output: The final network classification structure. □

III. Combination of Gene Expression Programming and Rough Neural Networks (GEP-RNN) Approach

Most classification algorithms seek models that attain the highest accuracy or equivalently the lowest error rate (classification error) when applied to the test set. Gene expression programming rough neural network algorithm for classification is presented in Algorithm 3. The algorithm covers the fundamental steps in the creation of classification models as shown.

Algorithm 3:

Gene expression programming rough neural networks (GEP-RNN) algorithm for classification

Step 1. Start Step.

- Import data set.

Step 2. Preprocessing Step.

- Create an information system.
- Solve incomplete problems (elimination of redundant attributes using genetic algorithm RSES).
- Construct reduct (using genetic algorithm).
- Decision rules generation.

Step 3. Processing Step

3.1. The evolution process of gene expression programming (GEP).

- Determine training data set.
- Determine test data set.
- Express chromosomes (chromosome length, number of genes on each chromosome, head size, tail size and the linking function).
- Start with the random generation of chromosomes of the initial population.
- Compute the fitness value for each individual by comparing the predict target value with the actual target value for all training cases.
- Stop the evolution, if the fitness value is sufficiently good, if the maximum number of generations has been evolved, or if the maximum execution time has been reached, else go to Step 3.1.
- Transfer the best (most fit) individual to the next generation.
- Select the individuals according to the fitness value to reproduce with modification, leaving progeny with new traits (use roulette-wheel sampling to select individual to the next generation).
- Perform genetic operators (i-e Mutation, Inversion, Transposition and Recombination)
- Repeat the process for the next evolution cycle.
- Keep the best gene as nonlinear entities of different sizes and shapes (expression trees (ET)).

3.2. Step of converting expression tree (ET) into rough neural networks (RNN) structure (Done by algorithm 4). □

Algorithm 4:

Step 1. Scan ET left-to-right and from top-to- bottom.

Step 2. Replace the first element of ET (root node of ET) with another element node (which represents the output of RNN).

Step 3. Go to the next line of the ET after passing the root, check if the next node represents an input or represents a function.

Step 4. If the node is an input, replace each input node on the ET with two nodes, one for the upper and the other for lower (upper/lower neuron).

Step 5. Replace each function node on the ET with hidden node and check its arguments.

Step 6. Repeat the steps (1- 5) until a line is formed (reach to the last element of ET).

Step 7. Construct connections between inputs and output (lower/upper neurons).

Step 8. Initialization weights and thresholds.

Step 9. Construct network learning.

Step 10. Return new networks with minimum error value if there is no significant change in the results for these networks. □

To illustrate how the algorithm works, consider the approach begins in **Figure 5**. The data sets are called an information system whose columns are labeled by attributes, rows, which are labeled by objects of interest, while the entries of the table are attribute values.

Preprocessing and analyzing data with rough sets require three parts: The formation of decision table, the discretization of serially attribute and attribute reduction [16]. There are several reasons for reducing the data set before presenting them to a gene expression programming phase. The most obvious reason is to reduce the training data, and the other is to reduce the learning time when training rough neural networks.

Dealing with finding minimal reducts of decision table based on the rough set theory, is a goal to develop an algorithm

capable of finding such reducts. Genetic algorithm was employed to obtain the minimal reduction of decision tables under existing conditions by combining its outstanding ability for overall searching with rough set theory [17]. In this paper, we use Rosetta software that apply rough sets algorithm which contains a reducer algorithm called (genetic algorithm RSES). It searches for reduct(s), either until the search space is exhausted or until a given maximum number of reduct(s) have been found.

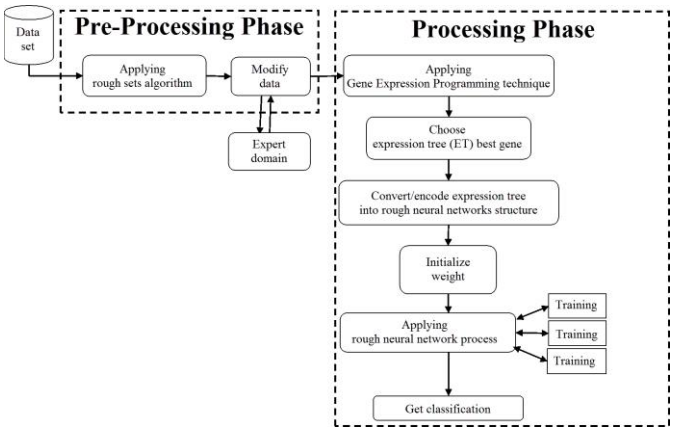

Figure 5 Gene expression programming - rough neural network (GEP-RNN) approach

A. Finding reduct(s) using genetic algorithm (GA)

The genetic algorithm (GA), invented by Holland in 1975, was inspired by organism evolution. It is an adaptive heuristic search algorithm and is used to obtain a solution for the rough-set decision problem. It includes three basic processes shown in **Figure 6**.

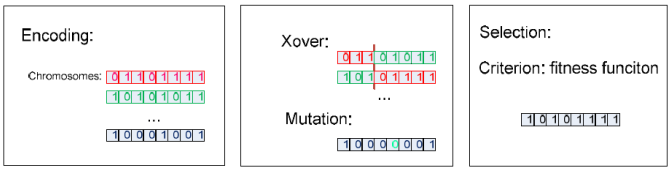

Figure 6 Key Process in GA

The idea of genetic algorithms is based on the Darwinian principle of "natural selection". In a case of classical genetic algorithms (see [18]) given a state space S (finite but large) and a function $f: S \to R_+$. Our goal is to find $x_o : f(x_o) = \max\{f(x): x \in S\}$. Elements of the sets are "individuals". Treating a value of the function (*f*) as an ability to survive in the environment ("fitness") while processing of the evolution is as follow:

1. Choose the representation scheme: a mapping from a space of "individuals" into "chromosomes"- usually bit string.
2. Randomly choose the set of chromosomes as an initial population.
3. Calculate "fitness F(c) of each chromosome c as a value of f(s(c)), where s(c) is the individual encoded by c. Then create a new population, replacing the chromosome with low fitness by those with higher fitness.
4. Randomly affect the new population by genetic operator, *e.g.*, mutation (small random modifications of chromosomes) and crossing-over (exchange of "genetic material" between some pairs of chromosomes).
5. Repeat 3-4 with the new population until a stopping criterion is satisfied.

For obtaining a solution for the rough-set decision problem by using genetic algorithm, we must find an encoding of potential solutions to the problem. This encoding must be meaningful to the problem and the corresponding solution.

Our problem is to find the minimum reduct of a decision table. The basic way to describe the reduct is a bitmap, which is shown in **Figure 1**. According to the reduct definition, the reduct representation is only a subset of attributes. Therefore, we create a binary string, the length of which is the same as the number of attributes in the decision table. Each bit of the string corresponds to one of the attribute of the decision table. If a certain bit is set to one, it means that the corresponding attribute belongs to the reduct. It is possible that when the genetic algorithm ends, we may have multiple solutions because these strings have the same fitness score [19].

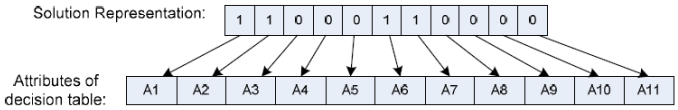

Figure 7 Solution representation of a reduct in GA with {A1, A2, A6, A7} is a reduct.

In this phase, we take into account some of the considerations as follow. Firstly, applying gene expression programming technique on "modified data" which is obtained from the previous phase. Secondly, converting an expression tree into rough neural networks. Thirdly, training/testing the structure of rough neural networks (RNN) to get the minimum classification error.

In gene expression programming (GEP), there are two different learning algorithms. The first is the basic gene expression algorithm (GEP) which does not support the direct manipulation of random numerical constants, whereas the second is the GEP with random numerical constants (GEP-RNC for short) [4, 6]. These two algorithms search the solution landscape differently. GEP with random numerical constants models (GEP-RNC) are usually more compact than models generated without random numerical constant which is used in this paper. Finally, keep the best gene which is in the form of expression trees (ET).

B. Illustrative Examples of converting expression tree (ET) into rough neural networks (RNN) structure

This section based on converting the linear chromosomes of fixed length (genotype) which represented also in the form of expression trees (ET) (the phenotype) into rough neural networks of different sizes and shapes (RNN) (see Algorithm 4). Each expression tree (ET) is translated (decoded) into rough neural networks (RNN) as follow:

Example (1)

We could linearize the gene tree in the form of open reading frames (ORFs) obtained from gene expression programming with sequences upstream from the start node and sequences downstream from the stop node in expression tree which is the straightforward reading of the ET from left to right and from top to bottom. Suppose we have a gene in open reading frames (ORFs) notation represents as in **TABLE I**TABLE .

TABLE I EXPRESSION OF GENE IN ORF

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
–	Avg2	Avg2	Not	–	Not	+	C_2	d2	d1	C_0	C_3	d1	C_2	C_7	d1	d1

Thus the function sets F = {−, + ,Avg2,Not} where "−,+" represents a subtraction and an addition of two arguments. The average"Avg2" represents an average of two inputs. The "Not" represents complement function that takes one argument, *i.e.*, $(1 - X)$. The terminal set in gene1 is T = {C_0,C_2,C_3,C_7,d1,d2}. Expression tree ET1 of gene1 is represented in Figure 8. Note that ET1 ends at position 13 in open reading frames (ORFs) notation.

It can also represent as in **Figure 9**, where "D" represents a function of two arguments and "T" represents a function of one argument converted into rough neural networks structure as shown in **Figure 10**.

IV. EXPERIMENT

We report results of experiments on two datasets: Iris Plant dataset and Breast Cancer dataset. The datasets obtained from the UCI machine learning repository center of machine learning and intelligent system (more details about the datasets are given in [20], and [21].

A. Iris Plant dataset

This data is used to classify the type of iris plant. The dataset consists of 100 samples divided into two species. Four features are measured for each sample: the length and the width of the sepals and petals in centimeters (see **TABLE II**), no missing attribute values in this dataset.

TABLE II ATTRIBUTES DESCRIPTION OF IRIS PLANT DATASET

Attribute	Attributes Types
1. sepal length in cm	Real
2. sepal width in cm	Real
3. petal length in cm	Real
4. petal width in cm	Real

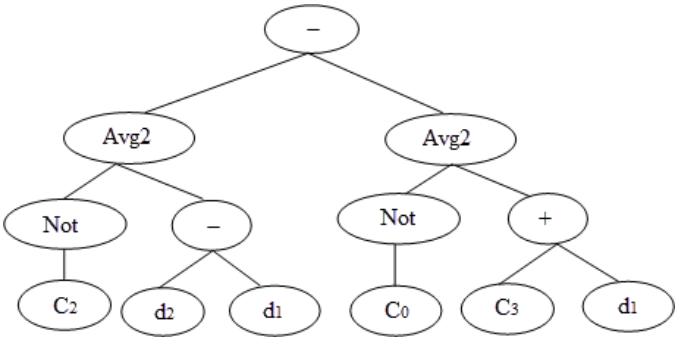

Figure 8 Expression Tree (ET1) of gene1

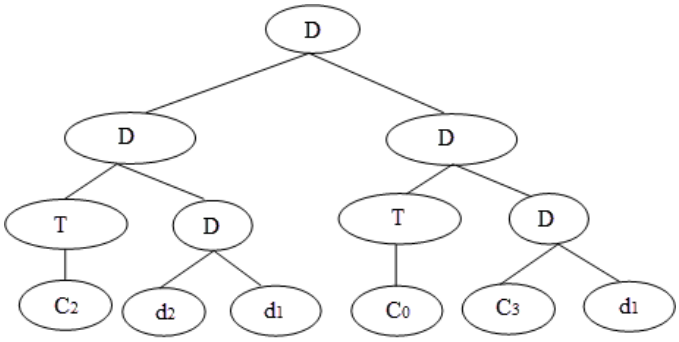

Figure 9 Representation of expression tree (ET1)

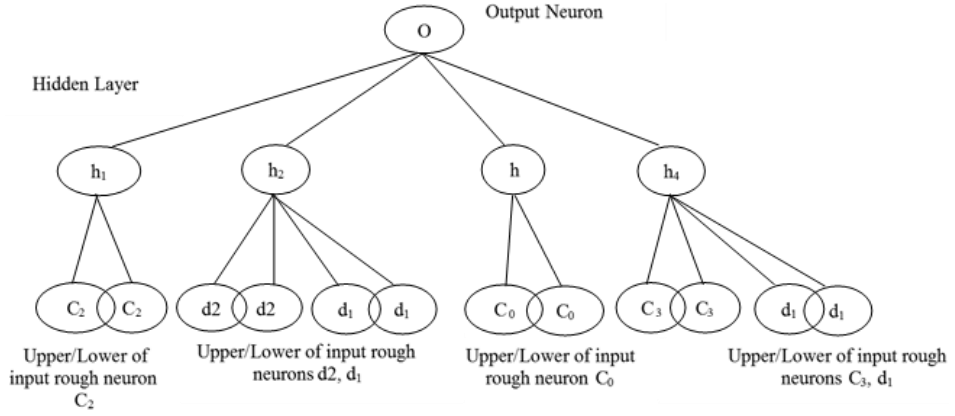

Figure 10 Rough Neural Networks structure of gene1

A rough set algorithm is applied on a dataset using Rosetta software [13]. Rosetta is a comprehensive software system for conducting data analyses within the framework of a rough set theory. The reduct {sepal width in cm, petal length in cm, petal width in cm} from the reduct set obtained is selected randomly, and the gene expression programming technique is applied on this reduct by using GeneXproTools software [14]. The learning algorithms used is the GEP with random numerical constants (GEP-RNC) [4][6]. All statistics results are obtained in **TABLE III**.

TABLE III TRAINING PARAMETERS OF GEP BASED MODELS

Data	
Independent Variables	3
Training Records	100 instances
Validation Records	100 instances
Program Structure	
Program Size	19
Used Variables (reduct choice)	d0 (2) represent 'sepal width in cm' is used (2) d1 (1) 'petal length in cm' (1) d2 (4) 'petal width in cm' (4)
Function Set	
Function Symbol	Arity
Addition + Subtraction – Multiplication * Division /	2 2 2 2
General Setting	
Chromosomes	30
Genes	3
Head Size	8
Tail Size	9
Linking Function	Addition
Run Information	
Genetic Operators Strategy: Optimal Evolution Mutation:0.00206 Function Insertion:0.00206 Leaf Mutation:0.00546 Inversion:0.00546 Tail Mutation:0.00546 Tail Inversion:0.00546 IS Transposition:0.00546 RIS Transposition:0.00546 One-Point Recombination:0.00277 Two-Point Recombination:0.00277 Gene Recombination:0.00277 Gene Transposition:0.00277 Random Chromosomes:0.0026	

Gene1: d2.–.+.+.*.*.d2.+.d0.d2.d0.d2.d2.d2.d2.d0.d1
+
Gene2: +.–.d2.+.–.*.–.+.d1.c5.c4.c1.d0.d0.d2.c7.d2
+
Gene 3: +.d2.c7.+.+.d2.–.+.d1.d2.d0.c8.c1.d2.d1.d2.c9

Figure 11 Genes of "Iris Plant" dataset in ORFs format

where (+) between each gene are a linking function and the numerical constants used in each gene is in **Figure 12**.

Each gene codes for a sub-ET. The sub-ETs interact with one another forming a more complex multi-subunit ET. The corresponding sub-ETs of the classified model are shown in **Figure 13**.

Gene2:
c5 = 6.46900845362712
c4 = -6.97378460036012
c1 = 7.40592669454024
c7 = 8.22931608020264
Gene 3:
c7 = 5.04989776299326

Figure 12 Numerical constants used in the model of "Iris Plant" dataset

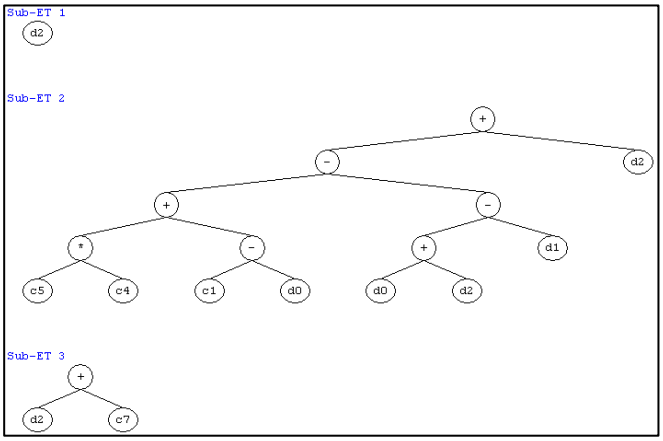

Figure 13 Sub-ETs of Iris Plant dataset

Encoded and converted sub-ETs are obtained into a rough neural network as in **Figure 14**.

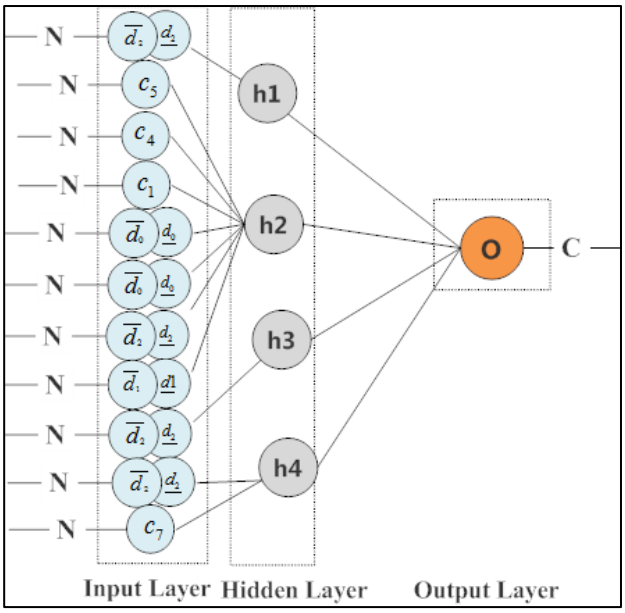

Figure 14 Rough neural network structure of Iris Plant dataset

The structure above is trained/tested by using rough neural network algorithm (see Algorithm 2) until the error becomes minimum and returns the final classification with minimum error. Training/testing the structure of rough neural networks is done by using the backpropagation algorithm [22], where the program is coded by Java, compiled, and run by version "j2sdk1. 4.1_02".

The result in **TABLE IV** is obtained when using cross validation method (60 records for training and 40 for test), the number of cross-validation folds is 3.

The network architecture is the number of neurons in each network layer including input and output layers. The network architecture is RNN: 11-4-1; that is, 3 layers with 11 input rough neurons in the input layer, 1 hidden layer with 4 neurons, and 1 output layer with 1 neuron.

TABLE IV THE RESULT OF ROUGH NEURAL NETWORKS

Run information	
Type of analysis:	Classification
Number of data rows:	100
Validation method: Number of cross-validation folds:	Cross validation 3
Using Back-Propagation Learning algorithm, variable number of hidden layers (1-10).	
Learning Rate:	0.1
Transfer Function	Sigmoid function
Momentum Rate:	0.9
Num Inputs:	11
Output Layer:	1
Hidden Layer:	1
Process with Training Data Average error:	 0.0233
Process with Testing Data Average error:	 0.0666

B. Wisconsin Breast Cancer dataset

This breast cancer databases was obtained from the University of Wisconsin Hospitals [21]. The dataset contains 699 instances and the number of attributes is 9 plus the class attribute (class attribute has been moved to last column). **TABLE V** shows the attribute information about the dataset.

TABLE V THE ATTRIBUTE DESCRIPTION OF BREAST CANCER DATASET

Attribute	Domain
0. Clump Thickness	1 - 10
1. Uniformity of Cell Size	1 - 10
2. Uniformity of Cell Shape	1 - 10
3. Marginal Adhesion	1 - 10
4. Single Epithelial Cell Size	1 - 10
5. Bare Nuclei	1 - 10
6. Bland Chromatin	1 - 10
7. Normal Nucleoli	1 - 10
8. Mitoses	1 - 10
9. Class: (0 for benign, 1 for malignant)	

A rough set algorithm is applied on the dataset using Rosetta software [13]. Consequently, getting new modified data (reduct sets) as input to the next phase. Reduct {1, 2, 3, 5} is selected from reduct sets which represented as {Uniformity of Cell Size, Uniformity of Cell Shape, Marginal Adhesion, Bare Nuclei} respectively.

Gene expression programming GEP technique is applied on this reduct by using GeneXproTools software [14]. The learning algorithms used is also the GEP with random numerical constants. When the classification model is being created by the learning algorithm, it would lead to getting the statistical results shown in **TABLE VI**.

TABLE VI STATISTICAL TRAINING RESULT OBTAINED USING GEP TECHNIQUE

Independent Variables	4
Training Records	350
Validation Records	349
Program Structure	
Program Size	33
Used Variables (reduct choice)	do(3) i-e attribute uniformity_cell_size is used (3), d1(3) attribute uniformity_cell_shape is used (3), d2(4) attribute marginal_adhesion is used (4), d3(3) attribute bare_nuclei is used (3)
Chromosomes	30
Genes	3
Head Size	8
Tail Size	9
Constants per Gene	10
Linking Function	Addition
Genetic Operators Strategy: Optimal Evolution Mutation:0.00138 Function Insertion:0.00206 Leaf Mutation:0.00546 Inversion:0.00546 IS Transposition:0.00546 RIS Transposition:0.00546 Uniform Recombination:0.00755 Uniform Gene Recombination:0.00755 One-Point Recombination:0.00277 Two-Point Recombination:0.00277 Gene Recombination:0.00277 Gene Transposition:0.00277	

Genes obtained in the form of (ORFs) are in **Figure 15**.

```
  Gene1: /.c5.+.+.c8.d2.d3.c4.d2.c3.d0.d3.c6.d2.d3.c9.d0
+
  Gene2: /.-.+./.d2.-.*.d1.d2.c0.d0.d3.d0.d1.c6.c6.d0
+
 Gene 3: +.d1.-.+.+.d3.+./.c9.d2.d1.c0.d0.d2.c4.d2.c6
```

Figure 15 Genes of "Breast Cancer" dataset in ORFs format

The numerical constants used in each gene are in Figure 16.

```
Gene 1:
        C5 = -2.19031342509232
        C8 = -8.62971892452773
Gene 2:
        C0 = 6.17603076265755
Gene 3:
        C0 = 1.73863948484756
        C9 = -3.67229224524674
```

Figure 16 Numerical constants used in the model of "Breast Cancer" dataset

The corresponding sub-ETs of classified" Breast Cancer" dataset model is shown in **Figure 17** below.

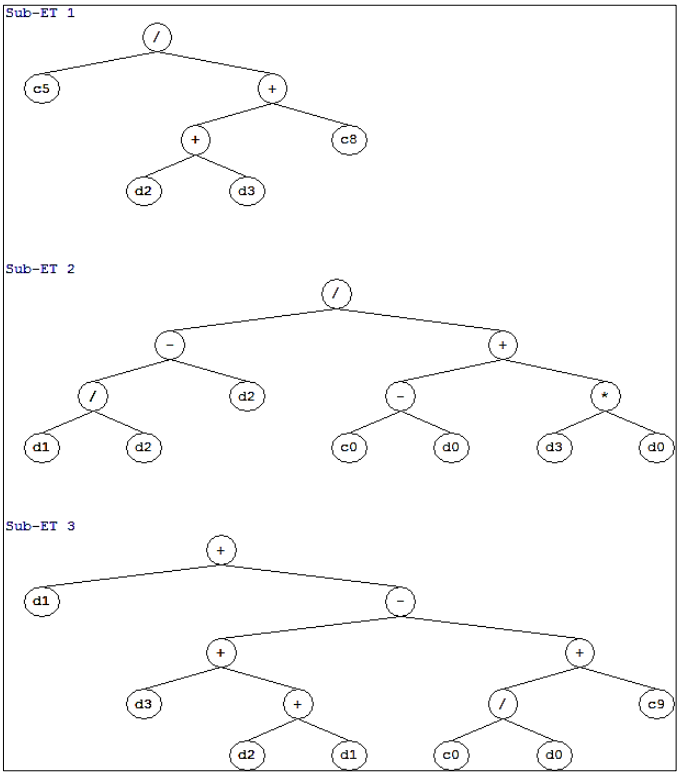

Figure 17 Sub-ETs of Breast Cancer dataset

The corresponding rough neural networks are shown in **Figure 18**. The network architecture is RNN: 18-5-1; that is, 3 layers with 18 input rough neurons in the input layer, 1 hidden layer with 5 neurons and 1 output layer with 1 neuron.

The result in table 6 is obtained when using cross validation method (350 records for training and the rest for test), the number of cross-validation folds is 3.

TABLE VII THE RESULT OF RNNS OF BREAST CANCER DATASET

Run information	
Type of analysis:	Classification
Validation method: Number of cross-validation folds:	Cross validation 3
Number of data rows:	699
Using Back-Propagation Learning algorithm, variable number of hidden layers (1-10).	
Learning Rate:	0.1
Transfer Function	Sigmoid function
Momentum Rate:	0.9
Num Inputs:	18
Output Layer:	1
Hidden Layer:	1
Process with Training Data Average error rate :	0.0266
Process with Testing Data Average error rate :	0.0410

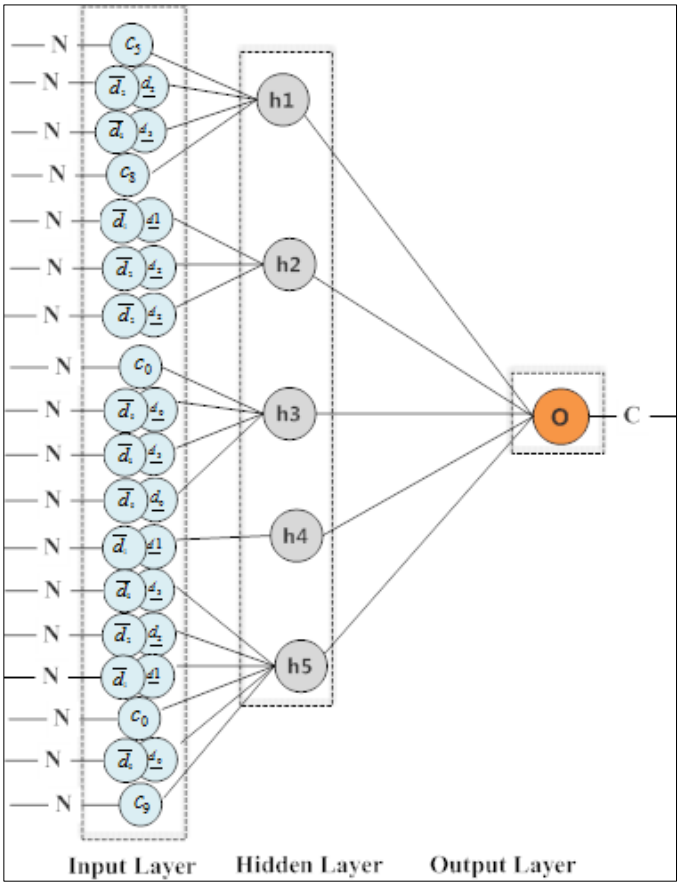

Figure 18 Rough neural network structure of Breast Cancer dataset

For more accurate results of the model gene expression programming rough neural networks GEP-RNN compared with the traditional neural network NNs and decision tree DT. The comparison result is displayed in **TABLE VIII**.

TABLE VIII COMPARISON RESULTS OF GENE EXPRESSION PROGRAMMING – ROUGH NEURAL NETWORKS GEP-RNN MODEL WITH NEURAL NETWORKS NNS AND DECISION TREE DT

Comparison	Accuracy of GEP-RNN Approach in percentage form %		Accuracy of Neural Networks (NNs) in percentage form %		Accuracy of Decision Tree (DT) in percentage form %	
	Training (T)	Validation (V)	Training (T)	Validation (V)	Training (T)	Validation (V)
Component	Upper/Lower Neuron		Neurons			
Structure	Network Component		Network Component			
Classification of Iris Plant dataset	97.67 %	93.34%	95.71 %	93.33%	95. 00%	93.00 %
Classification of Breast Cancer dataset	97.34%	95.9%	96.23	94.28	95.57%	94.56%

Notice that the classification is more effective for two dataset presented. Comparing all results obtained from an experiment with neural networks and decision tree would lead to GEP-RNN approach giving better results. In addition, it is able to reduce the training data and therefore reduce the learning time by preprocessing the data.

V. Conclusion

From the above theoretical and experimental, the gene expression programming rough neural networks GEP-RNN approach for classification can give better solutions and offers new possibilities for solving more complex technological and scientific problems. For gene expression programming rough neural networks (GEP-RNN) approach, there is a one-to-one correspondence between the expression tree (ET) and the rough neural network structure. Constructing new (ETs) would result to new rough neural network (RNN) structure.

The proposed approach shows also some modification on how to encode expression tree ET into rough neural networks structure which is trained and tested using backpropagation algorithm. The approach in this paper can be used beneficially for symbolic regression problems as well as circumventing the open problem of the original GEP in defining adequate lengths for the head of a gene. Additionally, this approach has presented a way of improving the accuracy in the classification problems.

References

[1] Aboul Ella Hassanien (2006) Rough Neural Intelligent Approach for Image Classification: A Case of Patients with Suspected Breast Cancer. International Journal of Hybrid Intelligent System, IOS press, 2006 (to appear).

[2] Coaquira F., and Acuna, E. (2007). Applications of Rough sets theory to data preprocessing in Knowledge Discovery. To appear in Proceedings of the conference of Machine learning and data analysis to be held October 2007 at UC Berkeley, California.

[3] Ferreira, C. (2002) Genetic representation and genetic neutrality in gene expression programming. Advances in Complex Systems, 5 (4): 389-408. CrossRef

[4] Ferreira, C. (2003) Function finding and the creation of numerical constants in gene expression programming. In J. M. Benitez,O. Cordon, F. Hoffmann, and R. Roy, eds., Advances in Soft Computing: Engineering Design and Manufacturing, pp.257-266, Springer-Verlag. CrossRef

[5] Hassan Y-F. (2014) New model of rough neural networks based on rough dependency to adapt cellular automata model, Kokull Journal, Vol. 64, No. 5.

[6] Ferreira, C. (2001) Gene expression programming: Anew adaptive algorithm for solving problems. Complex Systems, 13(2): 87-129.

[7] Lingras P. J. (1996) Rough neural networks. In: Proc. Of the 6th Int. Conf. on In-formation Processing and Management of Uncertainty in Knowledge-based Systems (IPMU96), Granada, Spain, pp.1445-1450.

[8] Nordin, M.A.R., Yazid, M.M.S., Aziz, A.,Osman, A.M.T.: DNA Sequence Database Classification and Reduction: Rough Sets Theory Approach. Proc. of 2nd International Conference on Informatics, pp. 41--47 (2007).

[9] Pal S.K. Polkowski S.K. and Skowron A. (Eds.) (2002) Rough-Neuro Computing:Techniques for Computing with Words. Berlin: Springer-Verlag.

[10] Peters J.F. Liting H. and Ramanna S. (2001) Rough Neural Computing in Signal Analysis. Computational Intelligence vol. 17, no.3: pp. 493-513. CrossRef

[11] Shyng, J-Y., Wang, F-K., Tzeng, G-H., Wu, K-S.(2007) Rough Set Theory in Analyzing the Attributes of Combination Values for the Insurance Market. Journal of Expert Systems with Application. vol. 32, pp. 56—64. CrossRef

[12] J. R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, (MIT Press, Cambridge, MA, 1992)

[13] Rosetta, A Rough Set Toolkit for Analysis of Data, web address http://www.lcb.uu.se/tools/rosetta/index.php. May 12, 2009

[14] GeneXproTools application, [Online], Released February 19, 2014. VIEW

[15] NeuroIntelligence 2.2 application, [Online], 2005. VIEW

[16] Jan G. Bazan, Hung Son Nguyen, Sinh Hoa Nguyen, Rough Set Algorithms in Classification Problem (2000)

[17] Kursat. Z, Novruz. A, Nihat. A.: Genetic algorithm and rough sets based hybrid approach for reduction of input attributes in medical systems “. International journal of innovative computing, information and control. Vol.9, number 7.ICIC© 203 ISSN1349-4198 (July 2013)

[18] J. H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, second edition (MIT Press, 1992)

[19] S. Wang: Algorithms for solving the reducts problem in rough sets. Master's Projects. Paper 285. [Online] (2012). VIEW

[20] UCI Machine Learning Repository, Center for Machine Learning and Intelligent Systems, [Online], (1988). VIEW

[21] UCI Machine Learning Repository, Center for Machine Learning and Intelligent Systems, [Online], (1992). VIEW

[22] Goldberg D. E., Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley, 1989.