Expert Systems With Applications 60 (2016) 211–221

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A genetically optimized neural network model for multi-class classification

Arpit Bhardwaj*, Aruna Tiwari, Harshit Bhardwaj, Aditi Bhardwaj

Computer Science and Engineering Department, BML Munjal University Gurgaon, India, Computer Science and Engineering Department, Indian Institute of Technology Indore, Indore, India

ARTICLE INFO

Article history:
Received 9 April 2015
Revised 7 April 2016
Accepted 28 April 2016
Available online 7 May 2016

Keywords:
Genetically optimized neural network
Multi-class classification
Multi-tree

ABSTRACT

Multi-class classification is one of the major challenges in real world application. Classification algorithms are generally binary in nature and must be extended for multi-class problems. Therefore, in this paper, we proposed an enhanced Genetically Optimized Neural Network (GONN) algorithm, for solving multi-class classification problems. We used a multi-tree GONN representation which integrates multiple GONN trees; each individual is a single GONN classifier. Thus enhanced classifier is an integrated version of individual GONN classifiers for all classes. The integrated version of classifiers is evolved genetically to optimize its architecture for multi-class classification. To demonstrate our results, we had taken seven datasets from UCI Machine Learning repository and compared the classification accuracy and training time of enhanced GONN with classical Koza's model and classical Back propagation model. Our algorithm gives better classification accuracy of almost 5% and 8% than Koza's model and Back propagation model respectively even for complex and real multi-class data in lesser amount of time. This enhanced GONN algorithm produces better results than popular classification algorithms like Genetic Algorithm, Support Vector Machine and Neural Network which makes it a good alternative to the well-known machine learning methods for solving multi-class classification problems. Even for datasets containing noise and complex features, the results produced by enhanced GONN is much better than other machine learning algorithms. The proposed enhanced GONN can be applied to expert and intelligent systems for effectively classifying large, complex and noisy real time multi-class data.

1. Introduction

Multi-class classification involves training of instances for different categories so as to further enable the identification of categories for various unknown instances. Data classification is one of the biggest problems due to the amount of growing data nowadays. It finds its application in many real world problems, like fraud detection, medical diagnosis, face recognition, speech recognition, vehicle detection in military warfare and knowledge extraction from databases (De Chazal, Dwyer, & Reilly, 2004; Ng & Gong, 2002; Zhang, Gao, & Lou, 2007; Iounousse, Er-Raki, El Motassadeq, & Chehouani, 2015; Bhardwaj & Tiwari, 2013).The field of data classification is receiving increased importance due to unpredictability and complexity of real world data. A classification algorithm require huge amount of accuracy and reliability which is very difficult for human programmers. Therefore, there is an immense need to develop an automated computer based classification systems (De Chazal, Dwyer, & Reilly, 2004) which can classify the required objects.

Artificial Neural Networks (ANN)(Hopfield, 1988), is a very popular machine learning technique which is biologically inspired by the anatomy of human brain. It offers a great potential for multi-class classification. Recently, there have been number of reports on applying ANN techniques to range of classification problems (Aprile, Castellano, & Eramo, 2014; Grbatinić, Marić, & Milošević, 2015; Iounousse, Er-Raki, El Motassadeq, & Chehouani, 2015; Lam et al., 2014; Mohammed, Badr, & Abdelhalim, 2015; Nie, Jin, Fei, & Ma, 2015). It consists of good adaptation/learning ability. Multilayer Perceptron (MLP) (Haykin, Haykin, Haykin, & Haykin, 2009) is one of the important artificial neural networks and back-propagation algorithm is one of the most widely used MLP training technique, which iteratively changes the weights between the neurons in a direction that minimizes the error. Thus, ANN can easily adapt to new trends in data but the problem with this is its slow rate of convergence and the risk of being trapped in a local optimum (Rumelhart, Hinton, & Williams, 1985). Another problem with BPA is in deciding its structure; the number of layers, number of

* Corresponding author.
E-mail addresses: arpit.bhardwaj@bml.edu.in, phd12110102@iiti.ac.in (A. Bhardwaj), artiwari@iiti.ac.in (A. Tiwari), harshitbhardwaj15@gmail.com (H. Bhardwaj), aditi.bhardwaj@bml.edu.in (A. Bhardwaj).

http://dx.doi.org/10.1016/j.eswa.2016.04.036
0957-4174/© 2016 Elsevier Ltd. All rights reserved.

hidden neurons in each layer and the connectivity between them. Koza (1992) brought a new dimension in this area, by using genetic evolution to optimize both, the weight and structure of a neural network. There have been many other works in this area(Khan, Ahmad, Khan, & Miller, 2013; Palmes, Hayasaka, & Usui, 2005; Rivero, Dorado, Rabuñal, & Pazos, 2010; Tsai & Lin, 2011; Turner, Dudek, & Ritchie, 2010). That motivates a new research dimension to model a hybrid solution i.e. evolution of artificial neural network using genetic programming. Recently Bhardwaj and Tiwari (2015) proposed a novel Genetically Optimized Neural Network (GONN) model which finds the optimal structure and weights of neural network architecture using Genetic Programming (GP) for WBCD database which is a two class problem. The results showed that GONN outperformed the other work from literature for WBCD database.

The aim of this paper is to develop an expert and intelligent system that can able to classify multi-class data. The reason for developing such system is that many algorithms work better for binary classification but they are not able to produce better results when they are applied on multi-class data. Therefore, in this paper, an enhanced Genetically Optimized Neural Network (GONN) model for multi-class classification is proposed which is an extension of Genetically Optimized Neural Network (GONN) model for two class problem (Bhardwaj & Tiwari, 2015). In this enhanced GONN model, we used a multi-tree representation, which integrates multiple GONN trees, each individual is a single GONN classifier and enhanced classifier is an integrated version of individual GONN classifiers for all classes. This amalgamated classifier is evolved in search of best classifier using modified crossover and mutation operators (Bhardwaj & Tiwari, 2015) that has the ability to classify any of the class in one single evolution. To measure the performance of the proposed algorithm we used seven datasets from the UCI Machine Learning Repository (Frank & Asuncion, 2010). It is observed that the proposed algorithm yielded a very high accuracy with lesser amount of time for all the multi-class datasets with different training-testing partitions. Mann-Whitney two tailed test are used to show the significance of the results produced by our methods in comparison to other methods. To show the dominance of our approach, we compared our method with a classical Koza and Rice (1991) model, classical Back propagation neural network (BPNN) (Hagan, Demuth, Beale, & De Jesús, 1996) and also with recently proposed algorithms applied on the multi-class data. The results show that our approach works well with the multi-class data and outperforms the other well-known machine learning methods like Neural Network (NN) (Örkcü & Bal, 2011), Genetic Algorithm (GA) (Örkcü & Bal, 2011), Hybrid Decision tree classifier (Farid, Zhang, Rahman, Hossain, & Strachan, 2014), Hybrid Naive Bayes classifier (Farid et al., 2014) and Support Vector machine (SVM) (Ramanan, Suppharangsan, & Niranjan, 2007).

2. Related work

Many classification algorithms are binary in nature and must be extended for multi-class classification. These include neural networks, decision trees, k-nearest neighbor, naive Bayes classifiers, and support vector machines (Aly, 2005). Therefore, GONN can also be extended for Multi-class classification. There are many methods available in literature to utilize GP for multi-class classification problems. Binary Decomposition method also known as one-versus-all is the most widely used method in GP for multi-class classification. In this method, one classifier is evolved for each class, discriminating a particular class from other classes present in the data. The final decision is made by presenting the input vector to classifiers of all classes. The classifier with positive or highest output is declared the winner. Many researchers have explored this method for multi-class classification (Bojarczuk, Lopes, & Freitas, 2000; Kishore, Patnaik, Mani, & Agrawal, 2000; Loveard & Ciesielski, 2001; Smart & Zhang, 2005; Teredesai & Govindaraju, 2004). There are several other methods like all versus all (Hastie & Tibshirani, 1998), generalized error correcting output codes (Allwein, Schapire, & Singer, 2001) and error correcting output codes (Dietterich & Bakiri, 1995) have also been used to tackle multi-class classification problems by binary classification algorithms. Muni, Pal, and Das (2004) used a multi-tree representation for multi-class classification, where a single classifier is an integrated version of individual classifiers for all classes.

Various work has been proposed in the literature for solving multi-class classification (Fu & Lee, 2012; Lorena, De Carvalho, & Gama, 2008; Sánchez-Morillo, López-Gordo, & León, 2014) using different machine learning algorithms. Bhardwaj and Tiwari (2013) proposed a constructive crossover operator using Best First Search (BFS) technique. In this, they generated all the possible combination off-spring from the parent during crossover and find the 2 off-spring having the highest fitness among them and transfer them to the next generation. The problem with their approach is that it is very time consuming. If the size of the parent is large than many possible combination of off-spring can be generated. So, there is no point in adding BFS in crossover operation, because it increases the time required to reach the solution drastically. Jabeen and Baig (2013) proposed two stage learning for multi-class classification problems. In the first stage, the classifiers are trained for each class versus the remaining classes. In the second stage, the classifiers are integrated and treated as a single chromosome that can classify any of the classes from the dataset. They compared their method with binary decomposition method and other machine learning algorithms. MontañéS, Barranquero, DíEz, and Del Coz (2013) used Directed Binary Trees for multi-class classification and improved the classification performance and evaluation time of previous decomposition approaches based on trees. They showed that there results are very competitive with respect to the one-vs.-one approach. Kim and Choi (2015) proposed a hierarchical multi-class classification method using logical analysis of data (LAD) based on a one versus all (OvA)-binary tree, called hierarchical multi-class LAD (HMC-LAD). They construct an OvA-binary tree by partitioning a node with K($\geq$ 2) classes into two subnodes by identifying one distinct class from the remaining (K-1) classes repeatedly. They showed that the classification performance of HMC-LAD is superior to existing multi-class LAD algorithms and other supervised learning approaches. Ramanan et al. (2007) proposed SVM-based hierarchical multi-class classifiers such as Directed Acyclic Graph (DAG)-SVM and Unbalanced Decision Tree (UDT)-based SVM. They used SVM for binary tree partition at each node and UDT-SVM uses OvA-type binary tree

It is also been observed in past that hybrid algorithms performed better for classification with their counter parts (Aci, İnan, & Avci, 2010; Bhardwaj & Tiwari, 2015). In the last decade, lots of work has been done for solving multi-class classification problem using hybrid algorithms (Chou, Cheng, & Wu, 2013; Lee & Lee, 2015; Polat & Güneş, 2009). Farid et al. (2014) proposed two hybrid algorithms respectively for a DT classifier and a NB classifier for multi-class classification tasks. The first proposed hybrid DT algorithm finds the troublesome instances in the training data using a NB classifier and removes these instances from the training set before constructing the learning tree for decision making. Their second proposed hybrid NB algorithm finds the most crucial subset of attributes using a DT induction. Khashei, Hamadani, and Bijari (2012) proposed a new hybrid model of artificial neural networks for classification problems using the multiple linear regression models. The main aim of the proposed model is to use the unique advantages of the multiple linear regression models in linear modeling in order to overcome the linear modeling deficiency of the traditional artificial neural networks. Melin, Amezcua, Valdez, and Castillo (2014) describes the application of competitive neural networks with the LVQ algorithm for classification of electrocardiogram signals (ECG). Seera and Lim (2014) proposed a hybrid intelligent system that consists of the Fuzzy MinMax neural network, the Classification and Regression Tree, and the Random Forest model, and presented its efficacy as a decision support tool for medical data classification. Therefore, in this paper we propose a hybrid algorithm an Enhanced Genetically Optimized Neural Network model that optimizes the weight and structure of neural network architecture for multi-class data. The detailed description of our proposed model is presented next.

3. Enhanced genetically optimized neural network

In this paper, we enhanced the Genetically Optimized Neural Network (GONN) model for multi-class classification. To design the enhanced GONN architecture, we used the concept given by Muni et al. (2004), in which each class is represented by a GP tree. So rather than a single GP tree we represent each class with a single GONN tree and then the complete classifier is represented as GONN classifier. A typical implementation for developing a Multi-class GONN Classifier involves the following steps:

3.1. GONN multi-tree classifier

A classifier D is a mapping, D $:R^p \longrightarrow N_{hc}$, where R^p is the p-dimensional real space and N_{hc} is the set of label vectors for a c-class problem and is defined as

$$N_{hc} = y \in R^c : y_i \in \{0,1\} \forall_i, \sum_{i=1}^{c} y_i = 1. \tag{1}$$

For any vector, $x \in R^p$ is a vector in c-dimension with only one component as 1 and all others as 0. Our objective is to design classifier D using GP. Given a set of training data $X = \{x_1 , x_2 , ...x_N \}$ and its associated set of label vectors $Y = \{y_1 , y_2 , ...y_N \}$. For a two class problem, a possible GONN classifier or a GONN individual is generally represented by a single GONN tree ($GONN_T(x)$). For a pattern x, the single GONN tree is constructed for two classes as follows:

$$\begin{aligned} &if\ GONN_T(x) \geq 0, x \in class1 \\ &else, x \in class2 \end{aligned} \tag{2}$$

The single tree representation of the GONN classifier is sufficient for a two-class problem. This scheme can be extended to a multi-class classification problem. In our design, every GONN individual will have a tree for every class.

So, a possible solution or an individual for the GONN is represented by c trees denoted by $(GONN_{T_1} , GONN_{T_2} , ...GONN_{T_c})$. For a pattern x, the condition of belongingness corresponding to a class is given as follows:

$$\begin{aligned} &if\ GONN_{T_i}(x) \geq 0\ and\ GONN_{T_j}(x) < 0\ \forall j \neq i, j \in \{1,2,...c\} \\ &then\ x \in class\ i \end{aligned} \tag{3}$$

It means that if the output of $GONN_{T_1}$ is greater than 0 the data belongs to class 1 otherwise it belongs to some other class. Similarly we check for all the GONN trees and finally we get that in which class the data actually belongs. For one sample only one tree will output one others will show output zero.

3.2. Initialization

GP evolves computer functions, traditionally represented in memory as tree structures. Every internal tree node has an operator function and every terminal node has an operand, making mathematical expressions easy to evolve and evaluate. We restrict the tree generation in GP in such a way that it preserves neural network architecture, so that we can convert the GP based classifier to an ANN. For achieving this, let a possible function set F and Terminal set T for a GP tree are,

$$F = \{P, W, +, -, *, \%\} \tag{4}$$

$$T = \{feature\ variables\ of\ dataset, R\} \tag{5}$$

where, P is an activation function which is standard sigmoid function in our case whose range is [0,1].

W is the weighting function for a signal going into P.

$+, -, *$ and % are normal arithmetic functions

D_0, D_1, .. D_n are the feature variables (input data signal) of dataset.

R contains randomly generated floating point constants between 0.0 and 10.0, because the input variables in our dataset have values in the range of 1 to 10.

The structure consists of an activation function (the P function) that process weighted inputs to produce a discrete signal (i.e. 0 or 1) as its output. Here, the standard sigmoid function is used as an activation function whose range is [0,1] and is written as follows:

$$y = \frac{1}{1 + e^{-x}} \tag{6}$$

where $x \in$ [-1,1]. P adds up its two weighted input lines and emits a 1 if the sum exceeds the value of 0.5 and emits a 0 otherwise. The weighting function W is, in fact, merely multiplication, given a special name to distinguish it from the ordinary arithmetic operation of multiplication. Both P and W can have varying number of arguments. The four arithmetic operations are used to create and modify the numerical constants (weights) of the neural network.

3.3. Fitness function

The most challenging and yet an essential concept of GP is the fitness function. The fitness function determines the ability of a program to solve the problem. Fundamentally, GP is guided by a fitness function, which explores for an efficient program to solve a given problem. It helps in determining the possible solutions for evolving into next generation of solutions.

The fitness evaluation is done by simply obtaining the mean-squared error of the GONN output.

$$Fitness_{GONN} = \frac{1}{1 + \frac{1}{X}\sum_{Y}^{X}(DES_Y - ACT_Y)^2} \tag{7}$$

here X is the size of the entire training dataset, DES_Y is the desired output from using the training dataset number Y, and ACT_Y is the GONN output when using the training dataset number Y as the input.

In GONN multi tree classifier we evaluate the fitness of all the GONN trees and then we do the average of all the GONN trees fitness present in that classifier which eventually becomes the fitness of enhanced GONN classifier.

3.4. GONN operators

After applying the reproduction operator, in which we select the top P_r% of top fitness individuals and transfer them to next generation, we apply a modified crossover and mutation operators Bhardwaj and Tiwari (2015). During modified crossover, 90% chance to functional nodes and 10% to terminal nodes are given for swapping. The newly generated offspring are then evaluated for their fitness. An offspring is allowed to enter the next generation only if it is better than its parent, in terms of its fitness value.

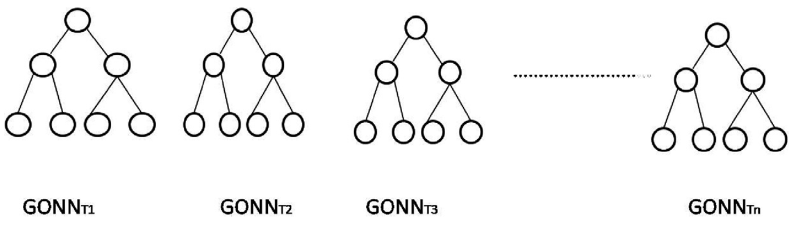


Fig. 1. Enhanced GONN multi-tree classifier for class n.

In other cases, the crossover operation is applied again until the performance of offspring improves. All the generated offspring are kept in a sorted order in a table. A certain top percentage (P_c%) of the new individuals are then selected to be carried forward to the next generation. The parent trees (P_m%) which did not generate good fitness offspring are selected for mutation operation.

The individuals which are not good for producing better offspring than themselves are changed in our mutation and we bring the diversity in them very early. However, the bias in crossover brings in a disadvantage that in later GP generations, it becomes more difficult to fine tune candidate solutions by changing a leaf of the tree. To overcome this problem, during mutation operation, 90% bias to terminal nodes and 10% to functional nodes are given for swapping. If the fitness value of child is less than the parent, the solution is discarded and mutation function is applied again on the same parent, until it generates better child than himself. Thus, making the crossover and mutation operators more constructive.

Algorithm 1 Algorithm for enhanced genetically optimized neural network (GONN) architecture for multi-class classification.

```
1: Input Multi-class training data and GP parameters seeTable 2.
2: Output enhanced Genetically Optimized Neural Network (GONN) classifier for multi-class.
3: Begin
4: Initial Population Generate initial classifier population of size k with input data.
5: while No. of fitness evaluations < Maximum number of fitness evaluations. do
6:     Fitness Evaluation Calculate the fitness value of all GONN individuals.
7:     Average fitness of all GONN individuals will become the fitness of GONN classifier.
8:     Reproduction Select top $P_r$% of individuals to be carried forward to the next generation.
9:     Crossover Apply modified crossover operation on all the remaining classifiers.
10:    for all crossover pairs do
11:        Repeat till we get offspring better than parents Take the parent pair and generate two offspring from them by giving more chances to function nodes (90%) to swap.
12:        Sort and store Place the top offspring ofthat pair into a table sorted according to fitness values of classifiers.
13:    end for
14:    Selection Select the top $P_c$% offspring from the sorted table and transfer them to the next generation.
15:    Mutation Take the parents of remaining $P_m$% individuals in the table and apply modified mutation.
16:    for all mutation parents do
17:        Repeat till we get offspring better than parents Take the parent and generate child from them by giving more chances to terminal nodes (90%) to mutate.
18:    end for
19:    Selection Transfer these $P_m$% offspring to the next generation.
20: end while
21: return Best classifier in terms of fitness value, which is considered as Genetically Optimized Neural Network (GONN) multi-class classifier.
22: End
```

3.4.1. Termination criteria

In this enhanced GONN approach, the learning/evolutionary process is terminated, if it meets with any of the below mentioned conditions.

1. If the number of fitness evaluation reaches its maximum count.
2. If the mean square error reaches 0.01.

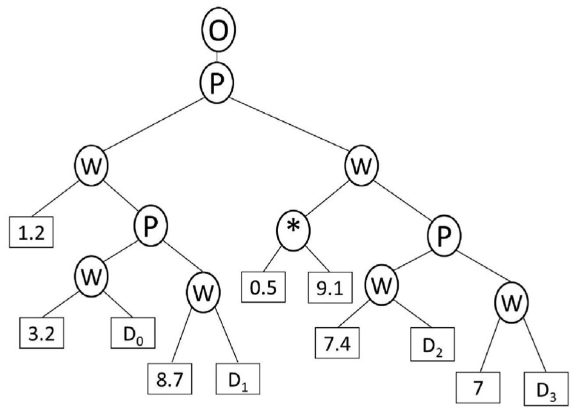
Through evolution, the newly generated classifiers are better than their previous population. The best GONN classifier in terms of fitness value is considered as the optimal GONN classifier. The enhanced GONN classifier is an integrated version of individual classifiers. An example classifier is shown in Fig. 1.

The step-wise procedure for evolving GONN multi-class classifier is described in the Algorithm 1. After that, mapping of each GONN tree is done to its equivalent Feed Forward neural network and then the complete GONN classifier for multi-class is used for testing data. Fig. 2a shows an example of a tree representation of a NN generated by GONN. Fig. 2b shows the same NN that has been reduced from the GONN tree. It contains one input layer with four inputs D_0, D_1, D_2, D_3, one hidden layer with two neurons and one output layer. In the same way, all the GONN trees of a GONN classifier are converted into its equivalent Neural Networks and then all the Neural Networks of an optimal GONN classifier is used for testing multi-class data.

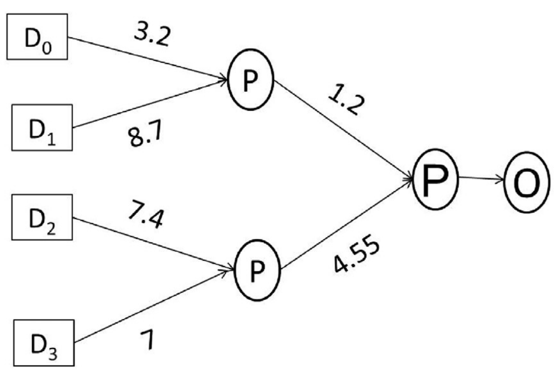
3.5. Mapping of a GONN classifier to its equivalent feed forward neural network classifier

During training phase, we evolve the Feed Forward Neural Network classifier using GONN architecture. After getting the optimal GONN classifier, we convert it into an equivalent Feed Forward Neural Network classifier. For converting the optimal GONN classifier to equivalent Feed Forward Neural Network classifier, we followed the following rules:

1. Left child of Weight function (W) will become the weight value for the activation function (from which the weight function is connected in GONN tree).
2. If the left child of W is arithmetic function than output of that arithmetic function will become the weight value of the activation function (from which the weight function is connected in GONN tree).
3. Right child of Weight function (W) will become the input for the activation function (from which the weight function is connected in GONN tree).
4. If the right child of W is an activation function P, then P becomes the neuron of a hidden layer of neural network.



(a) GONN representation of a NN. This figure is an example of one NN optimized by GONN. The O is the output node, P indicates the activation function, W indicates a weight, and $D_0 - D_n$ are the NN inputs



(b) Feed-forward NN representation of the GONN in Fig. 2a. To generate this NN, each weight in Fig 2a was computed to produce a single value

Fig. 2. Conversion of GONN architecture to its equivalent feed forward neural network architecture.

5. Now Repeat Step 1 4 for all the GONN individuals present in the optimal GONN classifier.

Now this Feed Forward Neural Network classifier of optimal GONN classifier is used for testing multi-class dataset. The Algorithm 2 represents the Feed Forward neural network classifier generated by GONN classifier applied on testing multi-class dataset. In this, we evaluate the GONN individuals present in the optimal GONN classifier. In this our first individual represent class 1, second individual represent class 2 and so on. We calculate the output value of each neuron (net) by summing the product of inputs and weights layer by layer and applying the sigmoid activation function. If the output of net is greater than 0.5 the value of net becomes 1 otherwise 0. The output of first layer becomes the input of second layer and in this way the final output of the architecture (OutputGONN) is evaluated. If the final output of our first GONN individual in GONN classifier is 1 then we say that sample belong to class 1. If the final output of our second GONN individual in GONN classifier is 1 then we say that sample belong to class 2 and so on. In this way, we predict the class of multi-class data by Genetically Optimized Neural Network.

Algorithm 2 Algorithm for testing feed forward neural network classifier generated by GONN classifier.

```
1: Input Multi-class testing data and optimal Feed Forward Neural
   Network classifier generated by GONN classifier.
2: Output Classification of testing samples.
3: Begin
4: for all k=1 to c Feed Forward Neural Network in an optimal
   classifier do
5:   for all test samples do
6:     for all  i = 1 to m layers do
7:       for all j = 1 to n neurons in each layer do
8:         $net_j^i = \sum W_j^{i-1} * D_j^{i-1}$
9:         if  $f(net_j^i) \geq 0.5$ then
10:          set $net_j^i = 1$
11:        else
12:          set $net_j^i = 0$
13:        end if
14:        $D_j^i = net_j^i$
15:      end for
16:    end for
17:    if $Output_{GONN} = 1$  then
18:      test sample $\in$ class k
19:    else
20:      test sample $\notin$ class k
21:    end if
22:  end for
23: end for
24: End
```

Table 1
Datasets used for experimentation.

Name of data set	No. of classes	No. of features	Number of samples
IRIS	3	4	150 (50+50+50)
WINE	3	13	178(59+71+48)
VEHICLE	4	18	846(212+217+218+199)
GLASS	6	10	214(70+76+17+13+38)
DERMATOLOGY	6	34	358(112+61+72+49+52+20)
E. COLI	8	8	336(143+77+52+35+20+5+2+2)
YEAST	10	8	1484(463+429+244+163+51+44 +37+30+20+5)

4. Results and discussion

The proposed GONN as a classifier for Multi-class problem was implemented in Java (Java SE 6 Update 45) and on a Pentium IV computer of 3.4 GHz with 2 GB of RAM. This algorithm was applied to the multi-class data taken from UCI repository. The number of classes, number of features and number of samples present in these datasets are described in Table 1.

In machine learning field, it is common to partition the dataset into two separate sets: a training set and a testing set. To evaluate the generalizability of our approach and to compare our work with existing work in literature, we divided the training and testing data into four different partitions. A standard 50-50 methodology was used, where half of the samples are used for training the classifier, and the rest to testing. We also divided the dataset into 60-40 and 70-30 training-testing ratios to show the importance of training data for our proposed approach. Here, a 10-fold cross validation technique (Hastie, Tibshirani, & Friedman, 2009) was also used to calculate classification accuracy of our model. In this method, entire dataset is divided into ten blocks of approximately equal size. While implementing our algorithm, we use 90% of data to train our model, and the rest 10% for testing. This

Table 2
Parameters Value for GONN and Koza and Rice (1991) models.

Parameters	Value
Probability of crossover operation (P_c)	60%
Probability of reproduction operation (P_r)	20%
Probability of mutation operation (P_m)	20%
Population size (k)	100
Initialization method	Ramped half and half
Initial maximum depth of tree	6
Initial minimum Depth of tree	3
Function set	+, -, ∗, / (protected division, division by zero is zero)
Terminals	Feature variables from datasets, floating point constants [0.0,10.0]
Termination criteria	40,000 Fitness Evaluation or MSE =0.01

Table 3
Parameters for classical back propagation algorithm.

Parameters	Structures	
	1-1-1	1-2-1
Number of hidden layers	1	2
Number of output neurons	1	1
Learning rate	0.1	0.1
Momentum constant	0.9	0.9
Termination criteria	MSE=0.01 or Total numberof epochs = 2000	MSE=0.01 or Total number of epochs = 2000

process is repeated 10 times, with a different data block left out for testing every time. So, total of 100 GP runs are evaluated. For 50-50, 60-40, 70-30 methodology also, 100 GP runs are evaluated. We considered two structures in this paper, in order to present the fair comparison between all the methods for all the datasets, first structure contains one input layer, one hidden layer and one output layer (1-1-1) and second structure contains one input layer, two hidden layer and one output layer (1-2-1). The parameters value for GONN and Koza's models are presented in Table 2 and the parameters value for classical back propagation model is presented in Table 3. The value of GONN parameters is primarily chosen based on the heuristic guidelines on the choice of parameters

Table 4
Comparison of classification accuracies of BPNN, Koza and Rice (1991) and GONN classifiers with different validation techniques for 1-1-1 and 1-2-1 network architecture 1-1-1 network architecture.

		1-1-1 network architecture						1-2-1 network architecture					
		BPNN		(Koza & Rice, 1991)		GONN		BPNN		(Koza & Rice, 1991)		GONN	
Datasets	Training-Testing partition	Mean	Sdv	Mean	Sdv	Mean	Sdv	Mean	Sdv	Mean	Sdv	Mean	Sdv
IRIS	50-50	89.88	1.05	91.88	1.05	97.44	0.52	90.48	1.08	92.65	1.56	98.25	0.89
	60-40	90.11	1.25	93.25	1.25	98.25	0.48	91.25	0.56	93.25	1.21	99.05	0.21
	70-30	90.89	0.89	93.58	1.89	99.02	0.56	91.59	1.26	94.12	0.89	99.15	0.56
	10-fold cross validation	89.48	1.25	92.26	1.05	99.04	0.52	90.99	1.28	93.85	1.26	99.20	0.49
WINE	50-50	82.77	2.63	85.66	1.58	91.66	0.5	83.59	2.14	88.58	1.48	92.65	0.85
	60-40	83.65	2.10	86.54	1.26	93.56	0.54	84.25	1.89	89.25	1.89	93.89	1.25
	70-30	84.11	1.02	87.21	0.78	94.12	1.25	84.99	1.02	90.02	1.04	94.99	1.25
	10-fold cross validation	82.47	2.13	85.96	1.78	93.66	0.59	82.59	1.15	87.58	1.88	94.95	0.85
VEHICLE	50-50	57.66	2.0	61.11	1.05	79.88	1.05	59.65	1.25	64.21	1.24	81.25	1.65
	60-40	58.79	1.25	62.35	2.54	81.25	1.25	61.25	0.25	65.24	1.09	83.26	1.90
	70-30	59.12	1.56	63.54	1.59	83.65	1.69	63.21	1.25	66.29	1.58	85.14	1.11
	10-fold cross validation	56.86	2.14	60.11	1.75	83.88	1.25	58.65	1.55	63.21	1.54	85.25	1.75
GLASS	50-50	60.33	1.00	64.11	1.05	79.77	2.63	62.21	1.02	66.25	1.58	81.25	0.98
	60-40	61.21	.078	65.25	1.25	81.25	1.25	63.26	1.25	68.28	1.29	84.29	1.26
	70-30	62.14	1.25	66.35	1.35	82.98	1.47	64.58	1.69	69.87	1.45	86.36	1.58
	10-fold cross validation	60.59	1.20	65.11	1.45	82.77	2.13	61.91	1.12	66.45	1.48	86.25	1.98
DERMAT-OLOGY	50-50	86.11	1.05	88.11	1.05	94.77	2.01	88.25	1.25	91.25	1.58	96.25	1.59
	60-40	87.12	1.89	89.65	1.48	95.21	2.36	89.65	1.24	91.41	1.02	96.89	1.48
	70-30	88.15	1.45	90.24	1.24	96.21	1.84	90.25	1.11	92.25	1.58	97.58	1.65
	10-fold cross validation	86.81	1.55	88.98	1.45	96.77	2.21	88.89	1.05	91.95	1.88	97.98	1.89
E.COLI	50-50	70	1.5	71.77	2.10	83.22	2.10	72.25	1.48	73.26	1.98	85.65	1.48
	60-40	70.56	1.05	72.59	2.35	84.69	1.65	74.25	1.25	75.65	1.63	87.54	1.65
	70-30	71.24	1.25	73.65	1.65	85.96	1.54	75.65	1.24	76.98	1.65	88.65	1.65
	10-fold cross validation	70.25	1.58	71.97	2.59	85.65	2.12	71.89	1.78	74.16	0.98	88.95	1.48
YEAST	50-50	60.11	1.05	61.33	1.58	75.88	1.05	62.58	1.89	63.25	1.78	79.58	1.25
	60-40	60.25	1.89	62.58	1.45	76.98	2.35	63.59	1.25	65.98	1.87	81.25	2.25
	70-30	61.27	1.87	64.89	1.54	78.58	1.54	65.55	1.55	67.45	1.21	82.98	1.58
	10-fold cross validation	60.98	1.25	63.25	1.56	78.98	1.74	63.58	1.58	65.58	1.25	84.52	1.56

Table 5
Comparison of training of BPNN, Koza and Rice (1991) and GONN classifiers with different validation techniques for 1-1-1 and 1-2-1 network architecture 1-1-1 network architecture.

		1-1-1 network architecture						1-2-1 network architecture					
		BPNN		(Koza & Rice, 1991)		GONN		BPNN		(Koza & Rice, 1991)		GONN	
Datasets	Training-Testing partition	Mean	Sdv	Mean	Sdv	Mean	Sdv	Mean	Sdv	Mean	Sdv	Mean	Sdv
IRIS	50-50	2:10	0:58	2:02	1:05	1:55	0.58	2:58	0:48	2:48	1:15	2:05	0:58
	60-40	2:58	1:42	2:45	1:16	2:15	0:56	3:25	1:28	3:15	0:58	2:28	0:48
	70-30	3:57	1:54	3:48	1:25	2:50	0:58	4:11	1:05	4:05	0:48	2:44	0:54
	10-fold cross validation	4:12	1:48	4:00	1:11	2:55	0:45	4:47	1:54	4:14	1:15	2:59	0:56
WINE	50-50	2:30	0:48	2:22	1:15	2:05	0.58	3:18	0:48	3:02	1:35	2:25	0:55
	60-40	3:38	1:22	3:15	1:16	2:35	0:46	3:55	1:18	3:45	0:28	2:55	0:28
	70-30	4:27	1:51	4:18	1:45	3:10	0:44	4:51	1:45	4:35	0:58	3:24	0:34
	10-fold cross validation	4:42	1:58	4:20	1:41	3:05	0:55	5:17	1:24	4:46	1:25	3:25	0:46
VEHICLE	50-50	8:30	2:48	7:22	3:15	4:05	1.58	9:18	3:48	8:02	2:35	4:25	1:55
	60-40	9:18	2:22	8:35	2:16	4:35	1:46	9:55	1:27	9:25	3:28	4:55	1:28
	70-30	9:27	3:51	8:51	2:45	4:50	1:44	10:11	3:45	9:45	2:58	5:24	1:34
	10-fold cross validation	10:12	2:58	9:20	2:41	4:05	1:55	10:47	2:24	9:46	2:25	4:35	0:46
GLASS	50-50	3:30	1:48	3:12	1:25	2:45	1.18	4:18	1:48	4:02	1:55	3:15	0:45
	60-40	3:58	1:42	3:35	1:36	2:59	1:06	4:35	1:28	3:59	1:18	3:15	0:48
	70-30	4:37	1:21	4:28	1:55	3:20	0:54	4:59	1:25	4:45	0:59	3:34	0:44
	10-fold cross validation	4:59	1:48	4:40	1:51	3:35	0:25	5:27	1:54	4:59	1:35	3:55	0:56
DERMATOLOGY	50-50	7:10	2:18	6:42	2:15	3:05	1.18	7:58	2:48	7:42	2:15	3:45	1:35
	60-40	8:18	2:29	7:35	2:36	3:35	1:56	8:55	1:47	8:25	3:48	3:55	1:28
	70-30	8:47	3:55	8:25	2:15	3:55	1:14	9:11	2:15	8:45	2:51	4:24	1:24
	10-fold cross validation	9:12	3:08	8:30	2:49	3:45	1:55	9:47	2:28	8:46	2:55	3:59	1:06
E.COLI	50-50	4:35	1:58	4:12	1:55	3:25	1.28	5:18	1:58	5:02	1:25	3:55	0:35
	60-40	4:58	1:52	4:35	2:36	3:45	1:26	5:35	1:29	4:59	2:18	3:59	0:38
	70-30	5:37	1:25	5:28	1:25	4:02	0:34	5:59	1:15	5:45	1:29	34:34	1:24
	10-fold cross validation	5:59	2:48	5:40	2:51	4:25	0:55	6:27	2:54	5:59	2:35	4:55	0:56
YEAST	50-50	10:30	3:12	9:25	3:11	6:15	2.58	11:18	4:48	10:22	3:25	6:45	2:55
	60-40	11:28	3:22	10:25	4:16	6:35	2:46	11:45	2:27	11:25	2:28	6:59	1:28
	70-30	11:27	4:51	10:51	3:45	6:50	2:44	12:21	4:55	11:45	2:28	7:04	2:30
	10-fold cross validation	12:22	2:28	11:22	3:21	6:55	2:55	12:47	3:24	11:26	4:25	7:05	1:46

Koza (1992) and an empirical search through initial experiments on GONN with the Standard crossover operator. The learning rate and momentum constant are chosen as 0.1 and 0.9, respectively for BPNN architecture. The activation function for all the algorithms is the standard sigmoid and it is same for all the neurons. Two different criterion were applied to stop the training: in one case it was stopped when the mean square error reached 0.01, and in the other the training was stopped when total number of fitness evaluation reached 40,000 for GONN and Koza's, and total number of epochs reached 2000, in case of BPNN model. The number of epochs, is the number of times the training samples were presented to the network. The mean and standard deviation (sdv) of the results are presented for 50 repetitions of all the algorithms for all the datasets for 50-50, 60-40, 70-30 training-testing partitions and for 10-fold cross validation scheme.

4.1. Performance evaluation methods

In order to compare and evaluate the performance of our proposed GONN architecture, we define and compute the classification accuracy and training time required to find the optimal structures. We also performed a statistical validation of the results with the Mann–Whitney (Corder & Foreman, 2009) two tailed test, to show the statistical difference in results. The formulations are as follows:

4.1.1. Accuracy

The measure of the ability of the classifier to produce accurate diagnosis is determined by accuracy.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100 \quad (8)$$

4.1.2. Training time

It is the training time required to reach the optimal classifier.

4.1.3. Mann–Whitney Test

Mann–Whitney (Corder & Foreman, 2009) two tailed test is used to show the statistical significance of the results. In this test, the p values are obtained by comparing the classification accuracy of methods. The difference between the two samples is highly significant if the value of $p < 0.001$ and for the value of $p > 0.05$ the results are not significantly different.

Table 6
p-value comparison using Mann–Whitney test.

			1-1-1 network architecture				1-2-1 network architecture			
			BPNN		(Koza & Rice, 1991)		BPNN		(Koza & Rice, 1991)	
	Datasets	Training- testing partition	*p*-value	Sig	*p*-value	Sig	*p*-value	Sig	*p*-value	Sig
GONN	IRIS	50-50	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS
		60-40	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS
		70-30	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS
		10-fold cross validation	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS
	WINE	50-50	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS
		60-40	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS
		70-30	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS
		10-fold cross validation	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS
	VEHICLE	50-50	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS
		60-40	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS
		70-30	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS
		10-fold cross validation	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS
	GLASS	50-50	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS
		60-40	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS
		70-30	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS
		10-fold cross validation	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS
	DERMATOLOGY	50-50	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS
		60-40	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS
		70-30	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS
		10-fold cross validation	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS
	E.COLI	50-50	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS
		60-40	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS
		70-30	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS
		10-fold cross validation	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS
	YEAST	50-50	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS
		60-40	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS
		70-30	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS
		10-fold cross validation	4.113×10^{-11}	HS	2.871×10^{-11}	HS	4.113×10^{-11}	HS	2.871×10^{-11}	HS

4.2. Experimental results

In this section, we present the experimental results to test the behavior of the enhanced GONN method for Multiclass classification, as well as to compare it with classical BPNN and Koza's models. The results of classification accuracy of all models with 1-1-1 and 1-2-1 network architectures are presented in Table 4 with different training and testing data. It is clear from the results that GONN model outperforms the classical BPNN and Koza's models in terms of classification accuracy for all training-testing data. Classification accuracy achieved by GONN model for 50-50 training-testing samples is better than all other methods, it shows that even for smaller training samples, GONN model is having good adaptation and generation capability even for multi-class problems and is able to provide better solutions. The classification accuracy of GONN model for 60-40 and 70-30 are also remarkable for every datasets which shows that if proper training data is provided to our model it is able to classify complex and real datasets. BPNN and Koza's models also shows improvement in classification accuracy when the training data increases but it is not equivalent to GONN. The classification accuracy of GONN for 10 fold cross validation method is much superior in comparison to BPNN and Koza's methods for all the datasets. It is confirm from the results of 10 fold cross validation that our method does not stuck in the problem of local convergence and guarantee us the global optimum solutions which is still a problem with BPNN and Koza's model.

The training time required by all models with 1-1-1 and 1-2-1 network architectures are presented in Table 5 with different training and testing data. It is clear from the results that enhanced GONN model requires very less time to reach the desired accuracy as compared to BPNN and Koza's models. The training time required by enhanced GONN even for complex data having large number of features is very less which shows the superiority of enhanced GONN model over BPNN and Koza's model. It is confirm from our results that enhanced GONN model is able to classify real world complex multi-class data with high accuracy and reduced training time.

The statistical difference in result is also presented using the Mann Whitney two tailed test in Table 6. It is clear from the results that the solution produce by our enhanced GONN model are statistically different from the BPNN and Koza's models for every training-testing partition for 1-1-1 and 1-2-1 architectures.

Table 7
Performance comparison with other work from literature.

Author year	Method	Datasets	Classification accuracy (%)	
Ramanan et al. (2007)	DAG-SVM	IRIS	96.67	
		VEHICLE	85.44	
		GLASS	65.54	
		DERMATOLOGY	93.04	
		E.COLI	81.99	
Örkcü and Bal (2011)	Backpropagation	IRIS	95.3	
		VEHICLE	74	
		GLASS	61	
		DERMATOLOGY	89.2	
		E.COLI	70.8	
		YEAST	65.7	
Örkcü and Bal (2011)	Binary coded Genetic Algorithm	IRIS	96	
		VEHICLE	74.7	
		GLASS	61.8	
		DERMATOLOGY	90	
		E.COLI	74	
		YEAST	69	
Örkcü and Bal (2011)	Real coded Genetic Algorithm	IRIS	97	
		VEHICLE	75.1	
		GLASS	64.5	
		DERMATOLOGY	92.5	
		E.COLI	74	
		YEAST	69	
Jabeen and Baig (2013)	Two-stage learning using Genetic programming	IRIS	96	
		VEHICLE	56	
		GLASS	64	
		WINE	85	
		YEAST	64	
Farid et al. (2014)	Hybrid Decision tree classifier	IRIS	98.66	
		VEHICLE	78.41	
		GLASS	76.27	
		WINE	90.17	
		YEAST	69.33	
Farid et al. (2014)	Hybrid Naive Bayes classifier	IRIS	98	
		VEHICLE	74.27	
		GLASS	52.33	
		WINE	86.41	
		YEAST	71.59	
Lee and Lee (2015)	SVM	IRIS	97.27	
		VEHICLE	85.44	
		GLASS	72.24	
		DERMATOLOGY	96.04	
		E.COLI	84.52	
		YEAST	59.82	
Kim and Choi (2015)	HMC-LAD	IRIS	96.00	
		VEHICLE	85.44	
		DERMATOLOGY	95.53	
		E.COLI	84.52	
This Study	*GONN*[1, 2]	IRIS	$97.44 \pm 0.52^{1a} - 98.25 \pm 0.89^{1b}$	$98.25 \pm 0.48^{2a} - 99.05 \pm 0.21^{2b}$
		WINE	$91.66 \pm 0.5^{1a} - 92.65 \pm 0.895^{1b}$	$93.56 \pm 0.54^{2a} - 93.89 \pm 1.25^{2b}$
		VEHICLE	$79.88 \pm 1.05^{1a} - 81.25 \pm 1.65^{1b}$	$81.25 \pm 1.25^{2a} - 83.26 \pm 1.90^{2b}$
		DERMATOLOGY	$94.77 \pm 2.01^{1a} - 96.25 \pm 1.59^{1b}$	$95.21 \pm 2.36^{2a} - 96.89 \pm 1.48^{2b}$
		GLASS	$79.77 \pm 2.63^{1a} - 81.25 \pm 1.65^{1b}$	$81.25 \pm 1.25^{2a} - 84.29 \pm 1.26^{2b}$
		E.COLI	$83.22 \pm 2.10^{1a} - 85.65 \pm 1.48^{1b}$	$84.69 \pm 1.65^{2a} - 87.54 \pm 1.65^{2b}$
		YEAST	$75.88 \pm 1.05^{1a} - 79.58 \pm 1.25^{1b}$	$76.98 \pm 2.35^{2a} - 81.25 \pm 2.25^{2b}$
This Study	*GONN*[3, 4]	IRIS	$99.02 \pm 0.56^{3a} - 99.15 \pm 0.56^{3b}$	$99.04 \pm 0.52^{4a} - 99.20 \pm 0.49^{4b}$
		WINE	$94.12 \pm 1.25^{3a} - 94.99 \pm 1.25^{3b}$	$93.66 \pm 0.59^{4a} - 94.95 \pm 0.85^{4b}$
		VEHICLE	$83.65 \pm 1.69^{3a} - 85.14 \pm 1.11^{3b}$	$83.88 \pm 1.25^{4a} - 85.25 \pm 1.75^{4b}$
		DERMATOLOGY	$96.21 \pm 1.84^{3a} - 97.58 \pm 1.65^{3b}$	$96.77 \pm 2.21^{4a} - 97.98 \pm 1.89^{4b}$
		GLASS	$82.98 \pm 1.47^{3a} - 86.36 \pm 1.58^{3b}$	$82.77 \pm 2.13^{4a} - 86.25 \pm 1.98^{4b}$
		E.COLI	$85.96 \pm 1.54^{3a} - 88.65 \pm 1.65^{3b}$	$85.65 \pm 2.12^{4a} - 88.95 \pm 1.48^{4b}$
		YEAST	$78.58 \pm 1.54^{3a} - 82.98 \pm 1.58^{3b}$	$78.98 \pm 1.74^{4a} - 84.52 \pm 1.56^{4b}$

1a and 1b are the Result of GONN for 50-50 training-testing data for 1-1-1 and 1-2-1 architecture respectively 2a and 2b are the Result of GONN for 60-40 training-testing data for 1-1-1 and 1-2-1 architecture respectively 3a and 3b are the Result of GONN for 70-30 training-testing data for 1-1-1 and 1-2-1 architecture respectively 4a and 4b are the Result of GONN for 10 fold cross validation scheme for 1-1-1 and 1-2-1 architecture respectively

4.3. Comparison with other methods

Proposed method is compared with other work present in literature applied on multi-class datasets. It is clear from Table 7 that our GONN approach, is significantly better in terms of classification accuracy than Neural Network (Örkcü & Bal, 2011), Genetic Algorithm (Örkcü & Bal, 2011), Hybrid Decision tree classifier (Farid et al., 2014), Hybrid Naive Bayes classifier (Farid et al., 2014) and Support Vector machine algorithms (Lee & Lee, 2015; Ramanan et al., 2007). It is important to highlight that, the authors of these work do not specify whether the results provided by them are on which training-testing partition. Therefore, in Table 7, we had presented the mean ± standard deviation of our result on 50-50, 60-40, 70-30 training-testing partition and also on 10 fold cross validation with 1-1-1 and 1-2-1 network architecture.

5. Conclusion

In this work, a Genetically Optimized Neural Network is enhanced for classifying the Multi-class data. We used a multi-tree GONN representation, which integrates multiple GONN trees, each individual is a single GONN classifier. Thus enhanced classifier is an integrated version of individual GONN classifiers for all classes. The integrated version of classifiers is evolved genetically to optimize its architecture for multi-class classification. This work is an extension of our previous work (Bhardwaj & Tiwari, 2015) in which we optimized the neural network architecture for classification of breast cancer data (two class problem). The proposed method is experimented and compared with the classical BPNN and Koza's models applied to the multi-class data taken from UCI repository. An enhanced GONN achieves better classification accuracy than BPNN and Koza's models for all the multi-class data sets for different training-testing partitions. Also, the results produced by enhanced GONN model for 10 fold cross validation confirms that GONN model does not stuck in local optimum problem and produces generalized results. The time taken by enhanced GONN model to reach the result is very less as compared to classical BPNN and Koza's models. The main reason for such improved accuracy and time reduction is our modified crossover and mutation operators. The results shows that not only for binary classification our enhanced GONN model is very excellent tool for classifying the multi-class real and complex data.

We also compare our multi-class GONN model with other machine learning algorithms like Neural Network (Örkcü & Bal, 2011), Genetic Algorithm (GA) (Örkcü & Bal, 2011), Hybrid Decision tree classifier (Farid et al., 2014), Hybrid Naive Bayes classifier (Farid et al., 2014) and Support Vector machine (SVM) algorithms (Ramanan et al., 2007) for multi-class classification. Our enhanced GONN model shows the competent or better results when compared with this state-of the art methods applied to the multi-class database. The classification accuracy of enhanced GONN model for E.Coli dataset having significant noise is better than all the compared work that shows the dominance of enhanced GONN model over other models like GA, NN and SVM. Even for complex datasets like Vehicle and Yeast, the classification accuracy of enhanced GONN is much superior than these models which confirm that enhanced GONN model can be applied to real world complex problems.

In the aspects concerning the limitation of this research, it is important to add feature extraction, feature selection into enhanced GONN model which help to extract the features even from images and various signals used during medical diagnosis and reduce the irrelevant and redundant features to make enhanced GONN more efficient for data classification. Other point that can be cited here is that the GONN model is applied on dataset available on UCI repository. It is interesting to see its behavior on real time complex data. There are few aspects of this research that could be improved further or extended in nearest future. The proposed model is applied on numeric data only; it would be interesting to see its behavior when it is applied on different types of data available in medical field such as images, signals. The research can be carried out for screening of features that can reduce the computational burden of the algorithm with further improvement in the classification accuracy.

References

Aci, M., İnan, C., & Avci, M. (2010). A hybrid classification method of k nearest neighbor, bayesian methods and genetic algorithm. *Expert Systems with Applications, 37*(7), 5061–5067.

Allwein, E. L., Schapire, R. E., & Singer, Y. (2001). Reducing multiclass to binary: A unifying approach for margin classifiers. *The Journal of Machine Learning Research, 1*, 113–141.

Aly, M. (2005). Survey on multiclass classification methods. *Neural Network, vol. 1*, 1–9.

Aprile, A., Castellano, G., & Eramo, G. (2014). Combining image analysis and modular neural networks for classification of mineral inclusions and pores in archaeological potsherds. *Journal of Archaeological Science, 50*, 262–272.

Bhardwaj, A., & Tiwari, A. (2013). A novel genetic programming based classifier design using a new constructive crossover operator with a local search technique. In *Intelligent computing theories* (pp. 86–95). Springer.

Bhardwaj, A., & Tiwari, A. (2015). Breast cancer diagnosis using genetically optimized neural network model. *Expert Systems with Applications, 42*(10), 4611–4620.

Bojarczuk, C. C., Lopes, H. S., & Freitas, A. A. (2000). Genetic programming for knowledge discovery in chest-pain diagnosis. *Engineering in Medicine and Biology Magazine, IEEE, 19*(4), 38–44.

Chou, J.-S., Cheng, M.-Y., & Wu, Y.-W. (2013). Improving classification accuracy of project dispute resolution using hybrid artificial intelligence and support vector machine models. *Expert Systems with Applications, 40*(6), 2263–2274.

Corder, G. W., & Foreman, D. I. (2009). Comparing variables of ordinal or dichotomous scales: Spearman rank-order, point-biserial, and biserial correlations. In *Nonparametric statistics for non-statisticians: A step-by-step approach* (pp. 122–154).

De Chazal, P., Dwyer, M. O., & Reilly, R. B. (2004). Automatic classification of heartbeats using ecg morphology and heartbeat interval features. *Biomedical Engineering, IEEE Transactions on, 51*(7), 1196–1206.

Dietterich, T. G., & Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research, vol. 2*, 263–286.

Farid, D. M., Zhang, L., Rahman, C. M., Hossain, M. A., & Strachan, R. (2014). Hybrid decision tree and naïve bayes classifiers for multi-class classification tasks. *Expert Systems with Applications, 41*(4), 1937–1946.

Frank, A., & Asuncion, A. (2010). *Uci machine learning repository*: vol. 213. Irvine, CA: University of california, School of Information and Computer Science.[http://archive.ics.uci.edu/ml]

Fu, J., & Lee, S. (2012). A multi-class svm classification system based on learning methods from indistinguishable chinese official documents. *Expert Systems with Applications, 39*(3), 3127–3134.

Grbatinić, I., Marić, D. L., & Milošević, N. T. (2015). Neurons from the adult human dentate nucleus: Neural networks in the neuron classification. *Journal of theoretical biology, 370*, 11–20.

Hagan, M. T., Demuth, H. B., Beale, M. H., & De Jesús, O. (1996). *Neural network design*: vol. 20. Boston: PWS Publishing Company.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *Unsupervised learning*. New York: Springer.

Hastie, T., & Tibshirani, R. (1998). Classification by pairwise coupling. *The Annals of Statistics, 26*(2), 451–471.

Haykin, S. S., Haykin, S. S., Haykin, S. S., & Haykin, S. S. (2009). *Neural networks and learning machines*: (vol. 3. Upper Saddle River: Pearson Education.

Hopfield, J. J. (1988). Artificial neural networks. *Circuits and Devices Magazine, IEEE, 4*(5), 3–10.

Iounousse, J., Er-Raki, S., El Motassadeq, A., & Chehouani, H. (2015). Using an unsupervised approach of probabilistic neural network (PNN) for land use classification from multitemporal satellite images. *Applied Soft Computing, 30*, 1–13.

Jabeen, H., & Baig, A. R. (2013). Two-stage learning for multi-class classification using genetic programming. *Neurocomputing, 116*, 311–316.

Khan, M. M., Ahmad, A. M., Khan, G. M., & Miller, J. F. (2013). Fast learning neural networks using cartesian genetic programming. *Neurocomputing, 121*, 274–289.

Khashei, M., Hamadani, A. Z., & Bijari, M. (2012). A novel hybrid classification model of artificial neural networks and multiple linear regression models. *Expert Systems with Applications, 39*(3), 2606–2620.

Kim, H. H., & Choi, J. Y. (2015). Hierarchical multi-class lad based on ova-binary tree using genetic algorithm. *Expert Systems with Applications, 42*(21), 8134–8145.

Kishore, J. K., Patnaik, L. M., Mani, V., & Agrawal, V. (2000). Application of genetic programming for multicategory pattern classification. *Evolutionary Computation, IEEE Transactions on, 4*(3), 242–258.

Koza, J. R. (1992). *Genetic programming: on the programming of computers by means of natural selection*: vol. 1. Cambridge: MIT Press.

Koza, J. R., & Rice, J. P. (1991). Genetic generation of both the weights and architecture for a neural network. In *Neural networks, 1991., IJCNN-91-seattle international joint conference on: vol. 2* (pp. 397–404). IEEE.

Lam, H.-K., Ekong, U., Liu, H., Xiao, B., Araujo, H., Ling, S. H., & Chan, K. Y. (2014). A study of neural-network-based classifiers for material classification. *Neurocomputing, 144*, 367–377.

Lee, Y., & Lee, J. (2015). Binary tree optimization using genetic algorithm for multiclass support vector machine. *Expert Systems with Applications, 42*(8), 3843–3851.

Lorena, A. C., De Carvalho, A. C., & Gama, J. M. (2008). A review on the combination of binary classifiers in multiclass problems. *Artificial Intelligence Review, 30*(1-4), 19–37.

Loveard, T., & Ciesielski, V. (2001). Representing classification problems in genetic programming. In *Evolutionary computation, 2001. proceedings of the 2001 congress on: vol. 2* (pp. 1070–1077). IEEE.

Melin, P., Amezcua, J., Valdez, F., & Castillo, O. (2014). A new neural network model based on the lvq algorithm for multi-class classification of arrhythmias. *Information Sciences, 279*, 483–497.

Mohammed, M. M., Badr, A., & Abdelhalim, M. (2015). Image classification and retrieval using optimized pulse-coupled neural network. *Expert Systems with Applications, 42*(11), 4927–4936.

MontañéS, E., Barranquero, J., DíEz, J., & Del Coz, J. J. (2013). Enhancing directed binary trees for multi-class classification. *Information Sciences, 223*, 42–55.

Muni, D. P., Pal, N. R., & Das, J. (2004). A novel approach to design classifiers using genetic programming. *Evolutionary Computation, IEEE Transactions on, 8*(2), 183–196.

Ng, J., & Gong, S. (2002). Composite support vector machines for detection of faces across views and pose estimation. *Image and Vision Computing, 20*(5), 359–368.

Nie, Q., Jin, L., Fei, S., & Ma, J. (2015). Neural network for multi-class classification by boosting composite stumps. *Neurocomputing, 149*, 949–956.

Örkcü, H. H., & Bal, H. (2011). Comparing performances of backpropagation and genetic algorithms in the data classification. *Expert Systems with Applications, 38*(4), 3703–3709.

Palmes, P. P., Hayasaka, T., & Usui, S. (2005). Mutation-based genetic neural network. *Neural Networks, IEEE Transactions on, 16*(3), 587–600.

Polat, K., & Güneş, S. (2009). A novel hybrid intelligent method based on c4. 5 decision tree classifier and one-against-all approach for multi-class classification problems. *Expert Systems with Applications, 36*(2), 1587–1592.

Ramanan, A., Suppharangsan, S., & Niranjan, M. (2007). Unbalanced decision trees for multi-class classification. In *Industrial and information systems, 2007. ICIIS 2007. international conference on* (pp. 291–294). IEEE.

Rivero, D., Dorado, J., Rabuñal, J., & Pazos, A. (2010). Generation and simplification of artificial neural networks by means of genetic programming. *Neurocomputing, 73*(16), 3200–3223.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). Learning internal representations by error propagation. *Technical Report*. DTIC Document.

Sánchez-Morillo, D., López-Gordo, M., & León, A. (2014). Novel multiclass classification for home-based diagnosis of sleep apnea hypopnea syndrome. *Expert Systems with Applications, 41*(4), 1654–1662.

Seera, M., & Lim, C. P. (2014). A hybrid intelligent system for medical data classification. *Expert Systems with Applications, 41*(5), 2239–2249.

Smart, W., & Zhang, M. (2005). Using genetic programming for multiclass classification by simultaneously solving component binary classification problems. In *Genetic programming* (pp. 227–239). Berlin, Heidelberg: Springer.

Teredesai, A. M., & Govindaraju, V. (2004). Issues in evolving GP based classifiers for a pattern recognition task. In *Evolutionary computation, 2004. CEC2004. congress on: vol. 1* (pp. 509–515). IEEE.

Tsai, H.-C., & Lin, Y.-H. (2011). Modular neural network programming with genetic optimization. *Expert Systems with Applications, 38*(9), 11032–11039.

Turner, S. D., Dudek, S. M., & Ritchie, M. D. (2010). Grammatical evolution of neural networks for discovering epistasis among quantitative trait loci. In *Evolutionary computation, machine learning and data mining in bioinformatics* (pp. 86–97). Berlin, Heidelberg: Springer.

Zhang, M., Gao, X., & Lou, W. (2007). A new crossover operator in genetic programming for object classification. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 37*(5), 1332–1343.