Neurocomputing ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at SciVerse ScienceDirect

Neurocomputing

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

# Two-stage learning for multi-class classification using genetic programming

Hajira Jabeen [a],*, Abdul Rauf Baig [b]

[a] Abasyn University, Islamabad, Pakistan
[b] National University of Computer and Emerging Sciences, Islamabad, Pakistan

ARTICLE INFO

*Keywords:*
Classification
Genetic programming
Classifier
Expression
Rule
Algorithm

ABSTRACT

This paper introduces a two-stage strategy for multi-class classification problems. The proposed technique is an advancement of tradition binary decomposition method. In the first stage, the classifiers are trained for each class versus the remaining classes. A modified fitness value is used to select good discriminators for the imbalanced data. In the second stage, the classifiers are integrated and treated as a single chromosome that can classify any of the classes from the dataset. A population of such classifier-chromosomes is created from good classifiers (for individual classes) of the first phase. This population is evolved further, with a fitness that combines accuracy and conflicts. The proposed method encourages the classifier combination with good discrimination among all classes and less conflicts. The two-stage learning has been tested on several benchmark datasets and results are found encouraging.

## 1. Introduction

Data classification finds its application in many real world problems, like fraud detection, face recognition, speech recognition and knowledge extraction from databases. The field of data classification is receiving increased importance due to unpredictability and complexity of real-world data. Evolutionary algorithms have shown evident performance for classification tasks. Genetic Programming (GP) is one of the evolutionary algorithms introduced by Koza [1] for automatic evolution of computer programs (including classifiers). GP has been successfully used for evolution of classifier-programs like decision trees [2]. Other GP based classification approaches include evolution of neural networks [3–5], autonomous classification systems [6], rule induction algorithms [7], fuzzy rule based systems and fuzzy petri nets [5,8]. Most of these methods involve defining a grammar that is used to create and evolve classification algorithms using GP.

Various researchers [9–13] have used GP for evolution of classification rules. The rule based systems include, atomic representations proposed by Eggermont [14,15] and SQL based representations proposed by Freitas et al. [12]. Tunsel and Jamshidi [16], Berlanga et al. [17] and Mendes et al. [18] introduced evolution of fuzzy rules using GP. Chien et al. [19] used fuzzy discrimination function for classification. Falco et al. [20] discovered comprehensive classification rules that use continuous value attributes. Bozarczuk et al. [21,22] used different set of functions applicable to different type of attributes that represent rules as disjunctive normal form. This type of GP is also referred as constrained syntax GP. Tsakonas et al. [23] introduced two GP based systems for medical domain and achieved noticeable performance. Lin et al. [24] proposed a layered GP, where different layers correspond to different populations that perform feature extraction and classification. Another method is evolution of arithmetic expressions for classification. The arithmetic expressions can be used for numerical data and they output a real value. This real value is translated into the classification decision using different thresholds. This includes static thresholds [25,26], dynamic thresholds [26,27] and slotted thresholds [28].

Multi-class classification problems are common in the real world applications for the tasks like object recognition, character recognition, person recognition, disease diagnosis and several others. 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 Baye's classifiers, and support vector machines [29]. GP also needs to be extended for multiclass classification problems. Several methods have been presented to use GP for multi-class classification problems. Most noticeable among them is the one-versus-all method also known as binary decomposition method. This method has been used widely in GP based 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. This method has been explored by many researchers [30–34]. Another relatively different method proposed by Muni et al. [35], uses a multi-tree representation, where a single classifier is an integrated version of individual

* Corresponding author. Tel.: +92 321 5708908.
*E-mail addresses:* hajirajabeen@hotmail.com (H. Jabeen), rauf.baig@nu.edu.pk (A.R. Baig).

0925-2312/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2012.01.048

classifiers for all classes. This amalgamated classifier is evolved in search of best classifier that has the ability to classify any of the class in one evolution.

Several other methods like 'all versus all' [36], error correcting output codes [37], and generalized error correcting output codes [38] have also been used to tackle multi-class classification problems by binary classification algorithms. However, none of them has been used in GP due to the large number of computations.

The drawback of binary decomposition method is the conflicting situations, where more than one classifier outputs a positive signal or none of the classifier outputs a belong-to signal. This situation degrades the classification accuracy. Several conflict resolution methods have been devised for this problem but they require extra processing during training and classification step. Another problem is the presence of skewed data. The data appears unbalanced for classification of a single class versus remaining classes. This problem is solved by increasing the number of training instances to make them appear balanced for each class [30,36]. This is named 'interleaved data format' where the samples belonging to class under consideration are repeated and alternately placed between samples belonging to other classes. This strategy increases the training data as well as the training time.

The proposed staged approach overcomes these two problems. It evolves the classifiers in two different stages that perform discrimination and integration, and incorporates a discriminative fitness function which takes care of skewed data without increasing the computation. The integrated evolution eliminates the conflicting situations decreasing the evaluation time required for conflict resolution. The proposed algorithm is detailed in the next section.

## 2. Proposed methodology

Many attempts have been made to develop general approaches to multi-class classification. One of the well known methods, in machine learning community, is one vs. all method. It involves learning a discriminator for each pair of class labels. The proposed classification mechanism uses the same principle but divides the training process into two phases. The first stage resembles the traditional binary decomposition method. The output, given by this phase, is a set of classifier populations for each class in the data. The second phase uses this population to populate the individual chromosomes using some selection criteria. Once the chromosome population is created, it is evolved in search of better accuracy. The output of second phase is a single chromosome that can classify any class present in the data.

### 2.1. Stage 1

In this stage, populations of arithmetic classifiers are trained to discriminate between absence and presence of a particular class amongst many classes present in the training data. Therefore we will have as many populations, as there are classes in the training data.

#### 2.1.1. Classifier representation

The arithmetic classifier expression (ACE) is represented as a binary tree created using the function set={+,−,/,∗} and terminal set={real attributes of the data, ephemeral constant}. The classifier represents an arithmetic relationship between attributes of the data. For each instance, it outputs a real value. For binary classification, the positive real value indicates the presence of a class and a negative value represents the absence of that class.

For example, consider a dataset with four attributes [$A_1$,$A_2$,$A_3$,$A_4$] and two classes $C_1$ and $C_2$. two classes.

An arbitrary classifier for class $C_1$ can be

$$(\mathrm{A3} \times \mathrm{A4})/(\mathrm{A1}+\mathrm{A2}) \tag{2}$$

For an instance with values

$$[1,2,3,4] \text{ belonging to the class } C_1 \tag{3}$$

The classifier will output a real value '4'.

This indicates that the instance belongs to the class $C_1$.

#### 2.1.2. Initialization

Initialization plays an important role in success of any evolutionary algorithm. A diverse and efficient initialization technique can lead to effective search during the evolutionary process. We have used the well known ramped half and half method [1] for initialization of the population. The ramped half and half method utilizes advantages of both full and grow initialization schemes with equal probability for each depth level till the maximum allowed depth. This method has been widely used for initialization in classification problems [30,35].

#### 2.1.3. Fitness

For a particular class, the classifier should be trained to output a positive response (accept) for the class under consideration and negative response (reject) for the instances belonging to the other classes. This measure leads to the problem of skewed data i.e. the number of positive, and a negative instance is not same. The researchers suggest that, in case of skewed data high accuracy may not represent good discrimination ability of a classifier [44]. On the other hand, a classifier with high area under the convex hull AUC has better discrimination ability. Therefore, we have used the discriminative power of the classifier as its fitness function.

The AUC function is considered better than accuracy in case of skewed data, a classifier with 80% accuracy may represent a case where a classifier always output a negative value and 80% of the data does not belong to the class $C_i$. On the other hand, 80% of AUC means the classifier can successfully discriminate between 80% of the samples, as belonging or not belonging to the class $C_i$. Several classifiers like rotation forests also deal with the skewed data [42,43].

Let '$n$' be the number of classes present in the data. Let $C_i$ be the class under consideration, $P_i$ be the number of samples belonging to the class $C_i$ (positive samples) and $N_i$ be the number of samples belonging to other classes $C_i'$ (negative samples).

The fitness function for evolving classifiers for class $C_i$ is

$$F_i = 1/2[\,(\text{true}-\text{positives}/P_i)+(\text{true}-\text{negatives}/N_i)] \times 100 \tag{4}$$

A classifier is evolved to discriminate between one class and rest of the classes such that output O of classifier F for class $C_i$ is positive for instances belonging to class $C_i$ and negative for instances not belonging to class $C_i$.

$$O[F(C_i)] \in +\mathrm{Z}\ \forall \text{ instances of class } C_i \tag{5}$$

and

$$O[F(C_i)] \in -\mathrm{Z}\ \forall \text{ instances of class } C_i' \tag{6}$$

where

$$C_i' = [C_1, C_2 \ldots C_n] - C_i \tag{7}$$

These positive and negative real values are converted into boolean values by considering positive values as 1 and negative values as 0.

Let the number of samples be ‘$k$’

$$\text{true}-\text{positives}/\text{Pi} = \left[\sum\nolimits_{i=1}^{k} O[F(C_i)] = 1 \forall k \in C_i\right]/p_i, k \in C_i] \tag{8}$$

$$\text{true}-\text{negatives}/\text{Ni} = \left[\sum\nolimits_{i=1}^{k} O[F(C_i)] = 0 \forall k \notin C_i\right]/N_i, k \notin C_i] \tag{9}$$

We have used two layered fitness [39,40]. The classifier with better fitness is always preferred and if the fitness of two classifiers is equal, then, the one with less number of nodes is selected. The fitness algorithm for one class is explained in Algorithm 1.

**Algorithm 1.** Fitness for individual classifiers for Stage I.

```
Step 1. Begin
Step 2. int true_positive=0
Step 3. int true_negative=0
Step 4. For all instances in the training data N
          a. Evaluate the classifier expression using the
             attribute values from the given instance (Value)
          b. If Value>=0 and class = desired class
             i.  true_positive ++
          c. if Value<0 and class = not desired class
             i.  true_negative ++
          d. End if
Step 5. End for
Step 6. Fitness=1/2[(true_positive/number of instances
        belonging to the class i)+(true_negative/number of
        instances not belonging to the class)]*100
Step 7. End
```

### 2.1.4. *Operators*

The operators used in the proposed algorithm are depthlimited crossover [40], point mutation [35] and reproduction [35].

**Algorithm 2.** DepthLimited Crossover.

```
Step 1. Begin
Step 2. Select two parents $P_1$, $P_2$ through selection mechanism
Step 3. Calculate depths $D_1$, $D_2$ of both parents
Step 4. If ($D_1$>$D_2$)
            ▪ Choose a random subtree $S_2$ from $P_2$
            ▪ Calculate depth of subtree $DS_2$
            ▪ Choose a random subtree $S_1$ from $P_1$ such that $DS_1$ =$DS_2$
            ▪ Swap $S_1$ and $S_2$ to create two children $C_1$ and $C_2$
Step 5. Else
            ▪ Choose a random subtree $S_1$ from $P_1$
            ▪ Calculate depth of subtree $DS_1$
            ▪ Choose a random subtree $S_2$ from $P_2$ such that $DS_2$ =$DS_1$
            ▪ Swap $S_1$ and $S_2$ to create two children $C_1$ and $C_2$
Step 6. End if
Step 7. Return $C_1$ and $C_2$
Step 8. End
```

The depth limited crossover selects two parents using tournament selection and selects a random subtree from larger parent; the other subtree of same depth is selected from the second parent. Both subtrees are then swapped. The algorithm of depth-limited crossover is explained in Algorithm 2 and illustrated in Fig. 1.

We have used the point mutation operator which selects a random point from the tree and replace a function node with the function node and a terminal node a random terminal node until a probability is met. The reproduction operator selects a random classifier to be the part of the next generation.

The description of the classifier evolution algorithm is given in Algorithm 3. The output of this phase is a population of ACE that is trained to differentiate between two classes by its response. It would output zero or greater value for one class and negative value for the other classes.

The process is repeated for each class. At the end of first phase there will be ‘$n$’ populations corresponding to each class. The next phase takes an amalgamated view of the classifiers and evolves them in a single run.

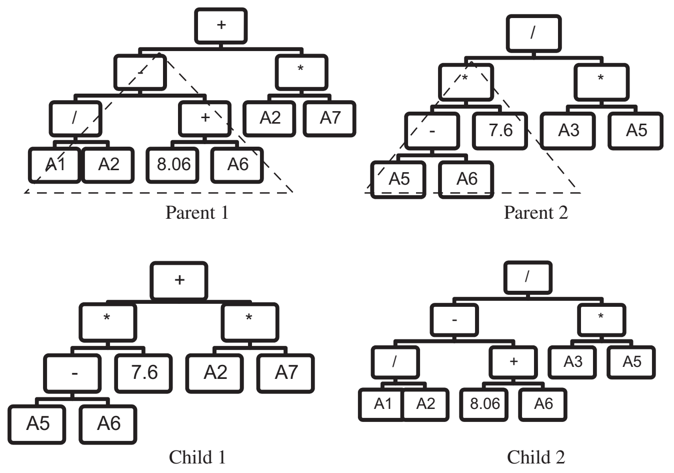

**Fig. 1.** DepthLimited crossover operator.

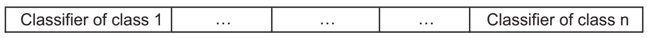

**Fig. 2.** A classifier chromosome for ‘n’ classes.

**Algorithm 3.** Stage I.

```
Step 1.  Begin
Step 2.  For each class $C_i$ in the data set
Step 3.  gen = 0, number of generations elapsed
Step 4.  $fit_g$ = 0, best fitness found till the generation gen
Step 5.  Initialize generations to user defined value
Step 6.  Select $C_i$ as ‘desired’ and others ‘not desired’ class
Step 7.  Initialize GP-ACE population using ramped half and
         half method
Step 8.  While (gen <= generations or $Fit_g$= 100 )
            a. Evaluate fitness of each member in population
               (Algorithm Fitness)
            b. Find best in population and update $Ace_{Best}$
            c. $Fit_g$=fitness($Ace_{Best}$)
            d. Perform evolutionary operators (Algorithm GP
               Evolution )
            e. gen = gen + 1;
Step 9.  End while
Step 10. Save the population of last generation
Step 11. End For Classes
Step 12. End
```

## 2.2. *Stage II*

This stage takes the individual discriminators from the previous phase and combines them in a single chromosome. These chromosome classifiers are evolved for a few more generations, in search of a good integrated classifier. This stage eliminates the need of any conflict resolution mechanism that requires extra computations.

### 2.2.1. *Classifier representation*

In this stage, a chromosome is an integrated version of individual classifiers for each class. An example classifier is shown in Fig. 2

The amalgamated version of classifier will output a vector of real values corresponding to each classifier/class. Only one value must be positive in ideal case. This means that the classifier for exactly one class generates a ‘belong to’ signal and other classifiers do not recognize that sample. As mentioned in the previous phase, a positive value represents ‘belong to’ signal, whereas the negative output represents absence of the particular class.

Let the vector be V and size of vector will be ‘n’ (number of classes/classifiers), each ‘i’ corresponds to one class present in the data. This vector can be converted into binary vector as follows:

$$\forall i \in n\{\text{if } [V_i] >= 0, \quad V_i = 1 \text{ else } V_i = 0\} \tag{10}$$

In ideal case, this vector must have exactly one non-zero output. If more than one value is positive, this means a conflict has occurred, which will reduce the classification accuracy of the amalgamated-chromosome classifier.

### 2.2.2. Initialization

A population of amalgamated classifiers is created using populations from previous phase. A tournament of given size is performed on each population and one member is selected, similarly one member is selected from each population and these 'n'; members are integrated into one chromosome. This process is repeated for each chromosome in the new population.

### 2.2.3. Fitness

The fitness function for each classifier is the number correctly classified samples associated with the conflict measure. The accuracy is number of samples classified correctly divided by the number of conflicts incurred. This decreases the accuracy of classifier which suffers from more conflicts. On the other hand, the classifiers with better accuracy and less conflict will be selected more often during the evolution process. The algorithm classification is explained in Algorithm 4.

**Algorithm 4.** Fitness of chromosome classifiers for stage II.

```
Step 1. Begin
Step 2. For each sample 'n' in the training data
          a. correct=0
          b. conflict=0
          c. For each class 'i' in the classifier
                i. Calculate output 'o[i]' for sample 'n'
          d. End class
                i. If n ∈ class i and o[i]==1 and ∀ o[j]!=1,
                   where, j ≠ i
                      1. correct ++
               ii. else
                      1. conflict ++
              iii. End if
          e. End samples
Step 3. Fitness=correct/conflict
```

### 2.2.4. Operators

The evolution operators are crossover and point mutation, where crossover selects a random tree in the chromosome and swaps two sub-trees from that tree whilst other trees are swapped as whole. This is represented in Fig. 3, where crossover between classifier chromosomes for three class problem is shown.

In mutation operator, a chromosome is selected randomly. A tree is selected from this chromosome, inversely proportional to its fitness, to undergo the mutation operation. The mutation is point operation where a function node is replaced by a randomly selected function node and a terminal node is replaced by a randomly selected terminal node.

The reproduction operation simply selects a random chromosome and copies it into the new population.

The new population of chromosome classifiers is evolved for certain number of generations and the best classifier is returned at the end of the evolutionary process.

## 3. Results

Five benchmark multi-class classification problems have been selected from UCI ML repository [41], for performance evaluation of this work. We have selected the datasets based on following properties:

(1) Dataset should be real or numerical valued.
(2) Problem should be multi-class classification.
(3) There should be no missing values.

The datasets have been chosen from various dimensions of life to show the applicability of GP classification as well as generalization of our proposed optimization technique.

### 3.1. Experimental settings

The properties of datasets used for experimentation is summarized in Table 1 given below.

For each data set, a ten-fold partitioning is created and GP is applied ten times for each fold. This process of creation of folds and GP is repeated ten times. This makes a total hundred GP runs and ten times ten-fold-cross-validation process. The parameters used for GP evolution are mentioned in Table 2.

### 3.2. Accuracy

The result of classification accuracy of the proposed algorithm is presented in Table 3. The BDGP column represent the average results obtained by using the traditional binary decomposition

**Table 1**
Datasets used for experimentation.

| Datasets | Referred as | Classes | Attributes | Type | Instances |
|---|---|---|---|---|---|
| Iris | IRIS | 3 | 4 | Real | 150 |
| Wine | WINE | 3 | 13 | Integer, Real | 178 |
| Statlog (Vehicle Silhouettes) | VEHICLE | 4 | 18 | Integer | 946 |
| Glass identification | GLASS | 6 | 10 | Real | 214 |
| Yeast | YEAST | 10 | 8 | Real | 1484 |

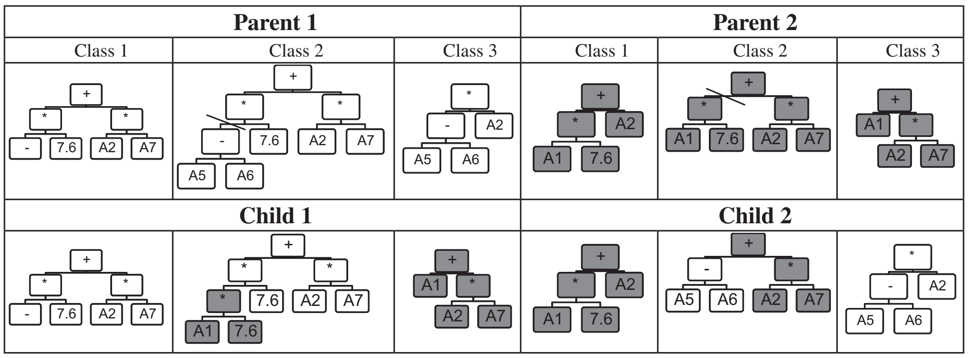

**Fig. 3.** Crossover operator.

**Table 2**
GP parameters.

| Parameters | |
|---|---|
| Population size | 600 |
| Crossover rate | 0.50 |
| Mutation rate | 0.25 |
| Reproduction rate | 0.25 |
| Selection for DepthLimited cross over | Tournament selection with size 7 |
| Selection for mutation | Random |
| Selection for reproduction | Fitness proportionate selection |
| Mutation type | Point mutation |
| Initialization method | Ramped half and half method with initial depth 6 |
| Function set | +, −,*,/ (protected division,division by zero is zero) |
| Terminals | Data attributes A1,A2…An, Ephemeral constant [0,10] |
| Termination criteria | User specified generations(120) or 100% training accuracy of classifier |
| PhaseI generations | Variable |
| PhaseII generations | 50 |

**Table 3**
Accuracy achieved using binary decomposition (BDGP) and the two stage learning (S2GP).

| Gen BDGP | IRIS | | WINE | | VEHICLE | | GLASS | | YEAST | |
|---|---|---|---|---|---|---|---|---|---|---|
| | BDGP (%) | S2GP (%) | BDGP (%) | S2GP (%) | BDGP (%) | S2GP (%) | BDGP (%) | S2GP (%) | BDGP (%) | S2GP (%) |
| 10 | 92.5 | 94.2 | 77.9 | 81.4 | 42.3 | 47.9 | 44.7 | 53.7 | 38.7 | 50.9 |
| 20 | 93.1 | 94.3 | 78.5 | 81.5 | 43.3 | 50.2 | 48.1 | 54.9 | 40.9 | 51.0 |
| 30 | 93.2 | 94.5 | 78.6 | 81.7 | 44.9 | 50.3 | 52.2 | 55.1 | 45.9 | 51.4 |
| 40 | 93.7 | 94.6 | 79.3 | 82.0 | 46.6 | 51.6 | 52.3 | 55.2 | 49.1 | 52.6 |
| 50 | 93.7 | 94.6 | 80.3 | 82.7 | 47.1 | 52.9 | 52.4 | 55.3 | 50.0 | 53.3 |
| 60 | 94.1 | 95.0 | 81.4 | 83.6 | 47.2 | 52.9 | 52.9 | 55.4 | 50.8 | 54.4 |
| 70 | 94.6 | 95.2 | 81.4 | 84.4 | 47.6 | 53.3 | 53.1 | 56.9 | 52.3 | 56.1 |
| 80 | 95.0 | 95.3 | 81.5 | 84.8 | 47.9 | 53.7 | 54.0 | 58.9 | 54.3 | 56.9 |
| 90 | 95.0 | 95.5 | 81.6 | 85.0 | 48.7 | 54.1 | 54.3 | 60.2 | 55.8 | 57.4 |
| 100 | 95.0 | 95.8 | 81.7 | 85.2 | 48.8 | 54.9 | 54.4 | 61.1 | 56.4 | 58.8 |
| 110 | 95.0 | 96.0 | 82.0 | 85.4 | 48.8 | 55.0 | 54.5 | 62.5 | 56.6 | 60.4 |
| 120 | 95.0 | 96.0 | 82.7 | 85.5 | 49.0 | 55.5 | 54.7 | 63.7 | 56.8 | 61.1 |

**Table 4**
Comparison with other algorithms.

| Dataset | One nearest neighbor (%) | Decision tree (%) | Support vector machine (%) | Two stage genetic programming (%) |
|---|---|---|---|---|
| IRIS | 95 | 91 | 94 | 96 |
| WINE | 84 | 84 | 83 | 85 |
| VEHICLE | 54 | 51 | 51 | 56 |
| GLASS | 60 | 62 | 63 | 64 |
| YEAST | 50 | 55 | 58 | 61 |

method [30]. The results presented in S2GP are average results obtained for the proposed staged-learning algorithm with variable first stage generations and 50 generations of second stage.

The results for all the datasets show considerable increase in accuracy using an amalgamated training approach. These results are better for all the data sets. The second phase adds 50 more generations of training to the first phase, but we can see that additional 50 generations of first phase alone (presented in the row 6), cannot achieve the accuracy obtained by the first phase. The proposed two-stage method has outperformed the traditional binary decomposition method for all the five datasets presented in this work.

We have also compared the proposed classification algorithm with state of the art classification algorithms in Table 4. This includes Support Vector Machines [45], Decision Trees [46] and [47] k-Nearest Neighbors with parameters from Galar et al. [48].

The results show that the performance of the proposed scheme is compatible with other state of the art classification algorithms.

## 4. Conclusions

The proposed two stage learning mechanism for multi-class classification using Genetic Programming has yielded better results when compared to one-versus-all or binary decomposition method. This is due to the fact that binary decomposition method suffers from conflicting situations. On the other hand, we have used a fitness measure that favors accurate classifiers and less conflicting outputs. The proposed method reduces the computation required to perform the conflict resolution during the classification process. This method has shown promising results on the benchmark problems from the UCI ML repository. The future work includes exploring this method for more complex problems and reducing the time complexity associated with more number of GP evolutions.

## References

[1] J.R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, 1992, MA.
[2] J.R. Koza, Concept formation and decision tree induction using the genetic programming paradigm, Lect. Notes Comput. Sci. 496 (1991) 124–128.
[3] D. Rivero, J.R. Rabunal, A. Pazos, Modifying genetic programming for artificial neural network development for data mining, Soft Comput. 13 (2008) 291–305.
[4] M.D. Ritchie, et al., Genetic programming neural networks: a powerful bioinformatics tool for human genetics, Appl. Soft Comput. (2007) 471–479.
[5] A. Tsakonas, A comparison of classification accuracy of four genetic programming-evolved intelligent structures, Inf. Sci. (2006) 691–724.
[6] M. Oltean, L. Diosan, An autonomous GP-based system for regression and classification problems, Appl. Soft Comput. 9 (2009) 49–60.
[7] G.A. Pappa, A.A. Freitas, Evolving rule induction algorithms with multi-objective grammer based genetic programming, Knowl. Inf. Syst. (2008).
[8] J., Eggermont, Evolving Fuzzy Decision Trees for Data Classification, Proceedings of the 14th Belgium Netherlands Artificial Intelligence Conference, 2002.
[9] R. Konig, U. Johansson, L. Niklasson, Genetic programming—a tool for flexible rule extraction, IEEE Cong. Evol. Comput. (2007).
[10] A.P. Engelbrecht, L. Schoeman, S. Rouwhorst, A building block approach to genetic programming for rule discovery, in data mining: a heuristic approach, in: H.A. Abbass, R. Sarkar, C. Newton (Eds.), Data Mining, Idea Group Publishing, 2001, pp. 175–189.
[11] E. Carreno, G. Leguizamon, N. Wagner, Evolution of classification rules for comprehensible knowledge discovery, IEEE Cong. Evol. Comput. (2007) 1261–1268.
[12] A.A. Freitas, A Genetic Programming Framework for Two Data Mining Tasks : Classification and Generalized Rule Induction, Morgan Kaufmann, CA, USA, 1997, Genetic Programming, pp. 96–101.
[13] C.S. Kuo, T.P. Hong, C.L. Chen, Applying genetic programming technique in classification trees, Soft Computing 11 (2007) 1165–1172.
[14] J., Eggermont, A.E., Eiben, J.I., Hemert, A comparison of genetic programming variants for data classification. Proceedings of the Eleventh Belgium Netherlands Conference on Artificial Intelligence, 1999 pp. 253–254.
[15] J. Eggermont, J.N. Kok, W.A. Kosters, GP For Data Classification, Partitioning The Search Space, Proceedings of the 2004 Symposium on Applied Computing, 2004, pp. 1001–1005.
[16] E. Tunstel, M. Jamshidi, On genetic programming of fuzzy rule-based systems for intelligent control, Int. J. Intelligent Autom. Soft Comput. (1996) 273–284.
[17] F.J. Berlanga, et al., A genetic-programming-based approach for the learning of compact fuzzy rule-based classification systems. Artificial Intelligence and Soft Computing–ICAISC 2006 (2006) 182–191.
[18] R.R.F. Mendes, et al., Discovering fuzzy classification rules with genetic programming and co-evolution, Principles of Data Mining and Knowledge Discovery (2001) 314–325.
[19] B.C. Chien, J.Y. Lin, T.P. Hong, Learning discriminant functions with fuzzy attributes for classification using genetic programming, Expert Syst. Appl. 23 (1) (2002) 31–37.
[20] I.D. Falco, A.D. Cioppa, E. Tarantino, Discovering interesting classificationrules with GP, Appl. Soft Comput. 4 (2002) 257–269.

[21] C.C. Bojarczuk, et al., A constrained-syntax genetic programming system for discovering classification rules: application to medical data sets, Artif. Intelligence Med. 30 (1) (2004) 27–48.
[22] C.C. Bojarczuk, H.S. Lopes, A.A. Freitas, An innovative application of a constrained-syntax genetic programming system to the problem of predicting survival of patients, Lect. Notes Comput. Sci., 2610, 2003.
[23] A. Tsakonas, et al., Evolving rule-based systems in two medical domains using genetic programming, Artif. Intelligence Med. (2004) 11–59.
[24] J.Y. Lin, et al., Classifier design with feature selection and feature extraction using layered genetic programming, Expert Syst. Appl. 34 (2007) 1384–1393.
[25] M. Zhang, V. Ciesielski, Genetic programming for multiple class object detection, Australia. Proceedings of the 12th Australian Joint Conference on Artificial Intelligence, 1999, pp. 180–192.
[26] D. Parrott, X. Li, V. Ciesielski, Multi-objective techniques in genetic programming for evolving classifiers, IEEE Cong. Evol. Comput. (2005) 183–190.
[27] W.R. Smart, M. Zhang, Classification Strategies for Image Classification in Genetic Programming, Proceeding of Image and Vision Computing NZ International Conference, 2003, pp. 402–407.
[28] M. Zhang, W. Smart, Multiclass object classification using genetic pro-gramming, Lect. Notes Comput. Sci. (2004) 367–376.
[29] M. Aly, Survey on Multiclass Classification Methods, 2005, 3005.
[30] J.K. Kishore, et al., Application of genetic programming for multicategory pattern classification, IEEE Trans. Evol. Comput. 4 (3) (2000) 242–258.
[31] T. Loveard, V. Ciesielski, Representing classification problems in genetic programming, IEEE Cong. Evol. Comput. 2 (2001) 1070–1077.
[32] W. Smart, M. Zhang, Using genetic programming for multiclass classification by simultaneously solving component Binary classification problems, Lect. Notes Comput. Sci. 3447 (2005).
[33] A. Teredesai,V. Govindaraju, Issues in evolving GP based classifiers for a pattern recognition task, In: Evolutionary Computation, 2004. CEC2004. Congress on, vol. 1, pp. 509–515. IEEE, 2004.
[34] C.C. Bojarczuk, H.S. Lopes, A.A. Freitas,Genetic programming for knowledge discovery in chest-pain diagnosis, Engineering in Medicine and Biology Magazine, IEEE 19, no. 4 (2000): 38–44.
[35] D.P. Muni, N.R. Pal, J. Das, A novel approach to design classifiers using GP, IEEE Trans. Evol. Comput. (2004): 183–196.
[36] T. Hastie, R. Tibshirani, Classification by pairwise coupling, Advances in Neural InformationProcessing Systems, Vol.10, The annals of statistics 26, no. 2 (1998): 451–471.
[37] T.G. Dietterich, G. Bakiri, Solving multiclass learning problems via error correcting output codes, J. Artif. Intelligence Res. 2 (1995) 1–38.
[38] E. Allwein, R. Shapire, Y. Singer, Reducing multiclass to binary: a unifying approach for margin classifiers, J. Mach. Learn. Res. (2000) 113–141.
[39] J. Eggermont, Data Mining using Genetic Programming: Classification and Symbolic Regression. Leiden University, Ph.D. Thesis, 2005.
[40] H. Jabeen, A.R. Baig, DepthLimited Crossover in genetic programming for classifier evolution, Ulsan, South Korea, International Conference on Intelligent Computing, 2009.
[41] A. Asuncion, D.J. Newman, Machine Learning Repository [Online] 2007. 〈http://archive.ics.uci.edu/ml/〉.
[42] J. Feng, J.H. Kyungsook, D.S. Huang, Sequence-based prediction of protein-protein interactions by means of rotation forest and autocorrelation descriptor, Protein Peptide Lett. 17 (2010) 137–145.
[43] K.H. Liu, D.S. Huang, Cancer classification using Rotation Forest, Comput. Biol. Med. 38 (2008) 601–610.
[44] M. Sokolova, G. Lapalme, Performance measures in classification of human communications, Lect. Notes Artif. Intelligence 4509 (2007) 150–170.
[45] V. Vapnik, Statistical Learning Theory, Wiley, New York, 1998.
[46] J.R. Quinlan, C4.5: Programs for Machine Learning, 1st ed., Morgan Kaufmann Publishers, San Mateo-California, 1993.
[47] G.J. McLachlan, Discriminant Analysis and Statistical Pattern Recognition, Wiley-Interscience, 544, 2004.
[48] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, F. Herrera, An overview of ensemble methods for binary classifiers in multi-class problems: experimental study on one-vs-one and one-vs-all schemes, Pattern Recognition 44 (8) (2011) 1761–1776.

**Hajira Jabeen** is working as an assistant professor at Iqra University, Islamabad, Pakistan since 2009. Her field of expertise include evolutionary Computation, swarm intelligence and data classification.

**Abdul Rauf Baig** has been assosiated with National University of Computing and Emerging Technologies, NU-FAST, Islamabad, Pakistan, since 2004. His field of expertise include Aritifical Intelligence, Data Mining and swarm intelligence.