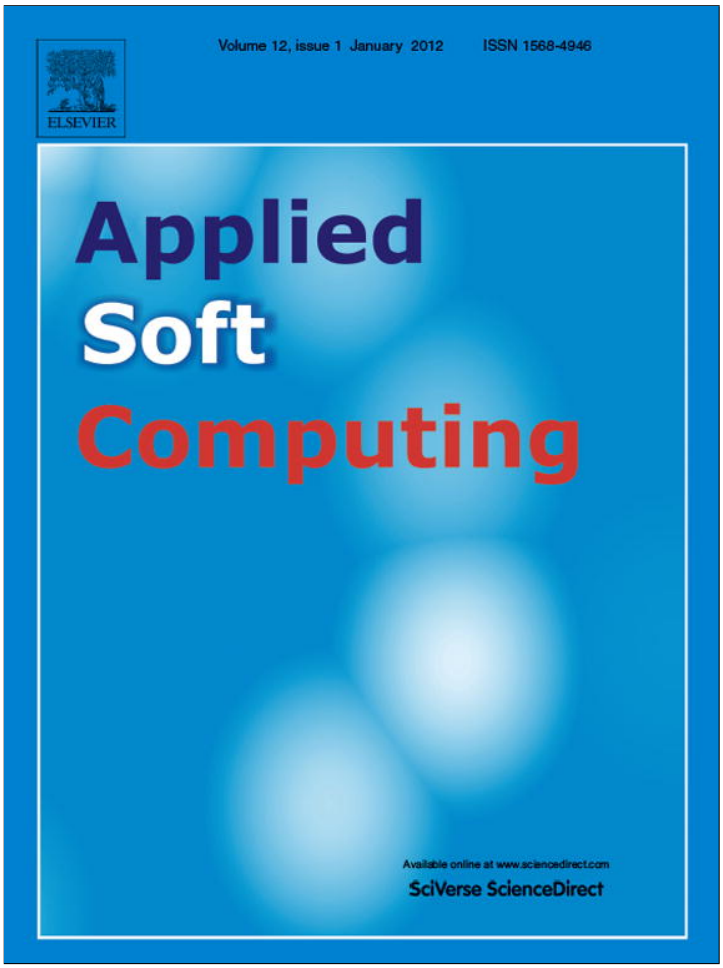

Applied Soft Computing 12 (2012) 416–422

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

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

Two layered Genetic Programming for mixed-attribute data classification

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

[a] Iqra University, 5 H-9\1, Islamabad, Pakistan
[b] National University of Computer and Emerging Sciences, Islamabad, Pakistan

ARTICLE INFO

Article history:
Received 27 November 2009
Received in revised form 26 February 2011
Accepted 14 August 2011
Available online 7 September 2011

Keywords:
Genetic Programming
Classification
Mixed attribute data
Mixed type data classification
Classifier

ABSTRACT

The important problem of data classification spans numerous real life applications. The classification problem has been tackled by using Genetic Programming in many successful ways. Most approaches focus on classification of only one type of data. However, most of the real-world data contain a mixture of categorical and continuous attributes. In this paper, we present an approach to classify mixed attribute data using Two Layered Genetic Programming (L2GP). The presented approach does not transform data into any other type and combines the properties of arithmetic expressions (using numerical data) and logical expressions (using categorical data). The outer layer contains logical functions and some nodes. These nodes contain the inner layer and are either logical or arithmetic expressions. Logical expressions give their Boolean output to the outer tree. The arithmetic expressions give a real value as their output. Positive real value is considered true and a negative value is considered false. These outputs of inner layers are used to evaluate the outer layer which determines the classification decision. The proposed classification technique has been applied on various heterogeneous data classification problems and found successful.

1. Introduction

Data classification is of high interest due to its applicability in several critical domains like disease diagnosis, feature recognition, fraud detection and decision making. The real-life-data is very unpredictable, which makes the classification a challenging task. This increases the need of automated classification systems with no or minimum human interference. Some properties of a good classification system are:

- Robustness: The classifier should be able to output good results over a variety of problems.
- Applicability: It should be readily applicable to data without any preprocessing.
- Accuracy: The resultant classifier should be reliable and exhibit good generalizing abilities.
- Efficient modeling: The structure should be flexible to adapt the data properties. It should be independent of data distribution.
- Comprehensibility: The classifier should be comprehensible to help in future decision making.
- Portability: The classifier should be portable to other tools for future use and efficacy.

Fortunately, one of the recent evolutionary algorithms, Genetic Programming (GP), possesses the above mentioned abilities. This important feature of GP has been recognized since its inception and GP has been widely used for the classification tasks. While being successful in various application domains [1–4], GP suffers from a few limitations like code bloat, long training time and lack of convergence, etc. Researchers have been trying to overcome these limitations to make the most out of this powerful classification tool.

Broadly, GP has been applied for classification in two different ways. One evolves classifiers as logical rules applicable to categorical data (continuous attributes need discretization). Another method is the evolution of arithmetic classifier expressions which is applicable to numerical attributes (categorical attributes are encoded into numeric values). The data transformations in either case can result in loss of information or biasness, in addition to added computational effort.

We have proposed a novel GP based classification system applicable to mixed attribute data without preprocessing of data. This is a two layered approach where the outer layer is a logical expression tree with some leaf nodes. These leaf nodes form the inner layer trees. The inner layer expressions can be of two types, logical expressions for categorical attributes and arithmetic expressions for continuous attributes of the data. The logical expressions give a Boolean value as output, which can be used by the outer layer, readily. On the other hand, the real output by arithmetic expressions is considered true for positive values and false for negative values. These outputs of inner layers are used to evaluate the outer layer

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

1568-4946/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2011.08.029

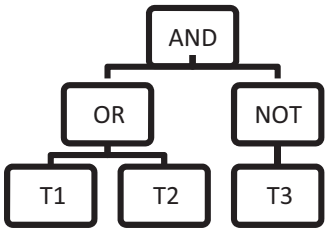

Fig. 1. Outer logical tree.

which determines the classification decision. This novel GP based representation has been tested on various binary datasets from UCI repository [5]. The method has been found compatible with various other GP based classification algorithms.

2. Related work

GP was introduced by Koza [6] in 1992 for automatic evolution of computer programs. Its ability to evolve classifiers has been realized since its beginning. Decision trees are one of the simpler classifiers and GP has been successfully used for decision tree evolution since 1991 [7]. Several advancements are being made [8] to date. Other classifier evolution approaches include evolution of neural networks [9–11], autonomous systems [12], rule induction algorithms [13], fuzzy rule based systems and fuzzy petri nets [11,14]. Most of these methods involve defining a grammar that is used to create and evolve classification algorithms using GP.

GP has been successfully used to evolve classification rules by various researchers [15–19]. The rule based systems include, atomic representations proposed by Eggermont [20] and SQL based representations proposed by Freitas [21]. The evolution of fuzzy rules, using GP, was introduced by Tunstel [22], Berlanga [23] and Mendes [24]. Chien [25] used fuzzy discrimination function for classification. Falco [26] discovered comprehensible classification rules that use continuous value attributes. Bojarczuk [27,28] used constrained syntax GP to represent classification rules in disjunctive normal form. Tsakonas [29] introduced two GP based classification systems in the medical domain and achieved noticeable performance. Lin [30] proposed a layered GP where different layers correspond to different populations performing the task of feature extraction and classification.

A relatively new and GP specific approach to classification is evolution of arithmetic expressions for classification. Arithmetic expressions use (real or integer value) attributes of data as variables in the expressions. The arithmetic expressions give a real value as output which determines classification decision. For binary classification problems, the boundary of positive and negative numbers is usually used. For multi-class classification problems, static threshold [31,32], dynamic threshold [32,33] and slotted threshold [34] application methods have been proposed. Another method for multi-class classification is binary decomposition or one versus all method. In this method, N classifiers are evolved for N class classification problem. Where, each classifier is trained to recognize samples belonging to one class and reject samples belonging to other classes. Binary decomposition method has been explored by various researchers [35–39]. Lichodzijewski [40] proposed a cooperative bid based mechanism for multiclass classification.

Very few researchers have focused on classifying multiple types of data. Eggermont [21] presented comprehensible atomic representations for mixed data classification. Loveard [41] proposed different strategies to use nominal attributes for classification; his proposed methods include execution branching and numeric conversions. Ong [42] used the discretization of continuous attributes present in data for credit approval. Most of the above mentioned methods involve data conversions requiring a preprocessing step. This step increases the complexity and can be a possible source of biasness and loss of information.

The logical rule based classifiers, and arithmetic expression based classifiers, have shown promising results. However, both methods are applicable to homogeneous data, the expression-based method is applicable to numerical data, and logical representation is suitable for categorical data. Mixed data must be preprocessed to be used by any method. Next section presents the proposed technique that uses the combination of logical and arithmetic expressions for mixed-type data classification.

3. Methodology

The first step in any GP system is defining the solution representation, by selecting a function and a terminal set. The proposed solution (classifier) is a two-layered tree. The outer layer is a logical tree with function and terminal nodes (antecedents). The function set contains '*and*', '*or*' and '*not*' operators. The terminal nodes of the outer layer tree are the inner layer trees. One instance of the outer layer logical tree (Fig. 1) is:

$$\text{Outer Logical Tree} = [(\text{Inner tree1}) \ \text{OR} \ (\text{Inner tree2})] \ \text{AND NOT} \ (\text{Inner tree3}) \quad (1)$$

There are two types of inner trees in our approach:

(1) Logical inner tree
(2) Arithmetic inner tree

A logical inner tree uses logical functions like '*and*', '*or*' and '*not*' and categorical attributes of data. The node in this tree is 'categorical attribute = *any value*' or 'categorical attribute $\neq$ *any value*'. The output of a logical tree is a Boolean value for each data instance. An example of a logical inner tree (Fig. 2b) is:

$$\text{Logical Inner Tree} = [(C_1 = \text{'a'}) \ \text{AND} \ (C_2 = \text{'b'})] \ \text{OR} \ [(C_4 \neq \text{'c'}) \ \text{OR} \ (C_1 = \text{'d'})] \quad (2)$$

The arithmetic inner tree combines the numerical attributes of the data by arithmetic functions. The function set used in this representation is:

$$\text{Function set} = \{+, \ -, \ *, \ /\}$$

The terminals in this representation are the numerical attribute names which are replaced by values of an instance for evaluation. The output of an arithmetic tree is a real value. One of the possible representations of an arithmetic inner tree (Fig 2a) is:

$$\text{Arithmetic Inner Tree} = (A_1/A_2) + (A_2 * A_3) \quad (3)$$

Several arithmetic and logical trees provide their output as input (nodes) to outer logical tree. The output of a logical inner tree can be directly used by the outer logical tree. On the other hand, the real output of arithmetic inner tree is considered true for positive values and false for negative values.

The number of arithmetic and logical inner trees in an outer tree is determined by arithmetic and logical probability which is the ratio of arithmetic attributes in the data or consequently logical attributes of data.

$$\text{Arithmetic Inner Tree probability} = \frac{\text{number of numerical attributes}}{\text{total number of attributes}} \quad (4)$$

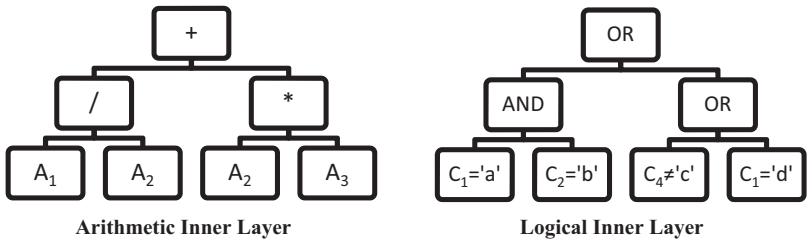

Fig. 2. Inner layers.

```
Data:[ A1 A2 A3 A4 C1 C2 C3 C4 ],A1..A4=numerical attributes,C1..C4=categorical Attributes
Instance: [ 1 2 3 -2 a b c d ] Є class 1
Arithmetic Inner Tree Probability : 4/8=0.5
Logical Inner Tree Probability : 4/8=0.5
Inner Tree 1= logical Inner Tree=[(C1='a')AND(C2='b')]OR[(C4≠'c')OR(C1='d')] => True
Inner Tree 2= Arithmetic Inner Tree = (A1/A2) + (A2*A3)=> ½+2*3 => 6.5 => True
Inner Tree 3= Arithmetic Inner Tree = A1+A4 => 1-2 => -1 => False
Outer Logical Tree = [(Inner tree1) OR (Inner tree2)] AND NOT (Inner tree3) => True
```

Fig. 3. An example of layered classification.

$$\text{Logical Inner Tree probability} = \frac{\text{number of categorical attributes}}{\text{total number of attributes}} \quad (5)$$

The total number of attributes is the sum of categorical and numerical attributes:

$$\text{Logical Inner Tree probability} = 1 - \text{Arithmetic Inner Tree probability} \quad (6)$$

The next step of proposed algorithm is initialization of two layered GP classifier trees or expressions. We have used ramped half and half initialization method [6] which allows diverse and flexible population initialization and has been found successful for a variety of classification problems. The maximum depth limit for the outer layer has been set between 2 and 4. The inner layers trees are also created using ramped half and half method with depth limits 2–4.

This implies that the maximum depth of any tree is limited to eight. The algorithm for creation of layered trees for classification is explained in Algorithm 1.

Algorithm 1. Layered Tree Initialization

Step 1. Begin
Step 2. For population size
  a. Initialize outer layer tree using ramped half and half method.
  b. For each node in outer tree
    i. If (arithmetic probability is met)
      1. Create arithmetic inner tree using ramped half and half method.
    ii. Else
      1. Create logical inner tree using ramped half and half method.
  c. End for
Step 3. End for
Step 4. End

Fig. 3 elaborates an example of the proposed layered classification decision for one data instance. Such evaluations are carried out for each instance in the training data to estimate the fitness of a classifier.

We have used three operators for the evolutionary process, which are, reproduction, mutation and crossover. The reproduction operator simply transfers an individual to the next generation. The individual member is chosen to use fitness proportionate

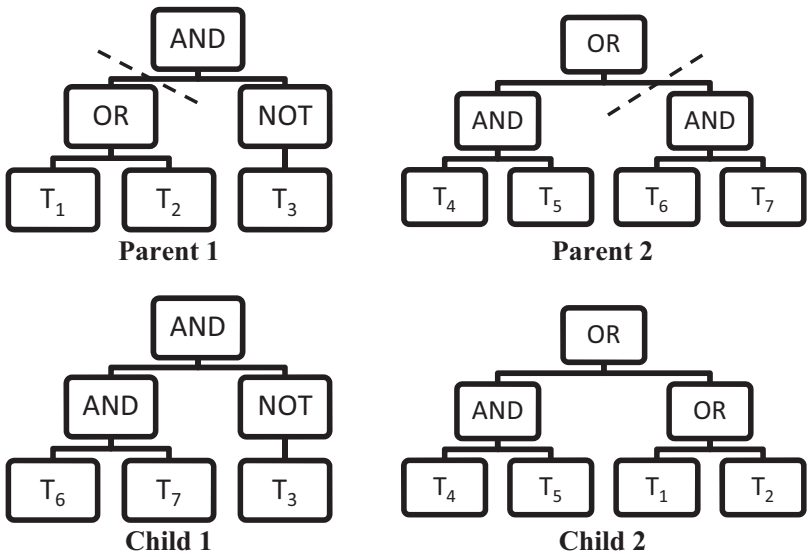

Fig. 4. Crossover operator.

selection. The individual for mutation is chosen randomly. The mutation operator randomly selects an inner subtree with 50% probability. If the root of subtree is a logical node, the subtree is replaced with a logical expression. On the other hand, if the root of the subtree is an arithmetic node, the subtree is replaced with a randomly generated arithmetic subtree. If the selected node belongs to the outer tree, a random outer layered subtree is generated which replaces the selected subtree.

For crossover, a random node is selected from the outer tree of both parents and swapped. The only restriction placed on both 'outer layer' subtrees is that they should have the same depth. This restriction has been adopted from the author's previous work named DepthLimited Crossover [43] where subtrees of the same depth are swapped to prevent the increase in classifier size during evolution. The process is illustrated in Fig. 4 where T_1–T_7 are the inner trees which may have varying depths. The depth of outer subtrees must be same for the crossover.

We have investigated two different fitness functions in our approach. One is the classification accuracy for the classifier and, the other is the area under the convex hull. The researchers suggest that, in case of skewed data (the number of instances belonging to one class are larger in number as compared to other one), high accuracy may not represent good discrimination ability of a classifier [44]. On the other hand, a classifier with higher area under the convex hull AUC has better discrimination ability. Both fitness functions are described in Algorithms 2 and 3.

Algorithm 2. Fitness ACC

Step 1: Begin
Step 2: For each data instance
  a. Evaluate logical antecedents using attributes values of given Instance (Inner Layer)
  b. Evaluate arithmetic antecedents using attributes values of given Instance (Inner Layer)
  c. Evaluate the outer expression by substituting antecedent values
  d. If expression == true and data instance ∈ class 1
  e. Increment CorrectCount
  f. If expression = false and data instance ∉ class 1
  g. Increment CorrectCount
Step 3: End for
Step 4: Fitness = CorrectCount/number of data instances
Step 5: End

In an evolutionary algorithm, the fitness of every population member is evaluated in search of a better solution. In our case, every member is a classifier. All the training instances are used to calculate fitness (ACC or AUC) of each classifier.

Algorithm 3. Fitness AUC

Step 1: Begin
Step 2: For each data instance
  a. Evaluate logical antecedents using attributes values of given Instance (Inner Layer)
  b. Evaluate arithmetic antecedents using attributes values of given Instance (Inner Layer)
  c. Evaluate the outer expression by substituting antecedent values
  d. Count TruePositive, TrueNegative, FalsePositive and FalseNegative
Step 3: End for
Step 4:
Fitness = 0.5 * [(TruePositive/(TruePositive + FalseNegative)) + (TrueNegative/(TrueNegative + FalsePositive))]
Step 5: End

The algorithm is continued for a certain number of generations in search of better classifiers and terminated if a better solution has been found or given number of generations have elapsed. The overall classification algorithm is illustrated in Fig. 5.

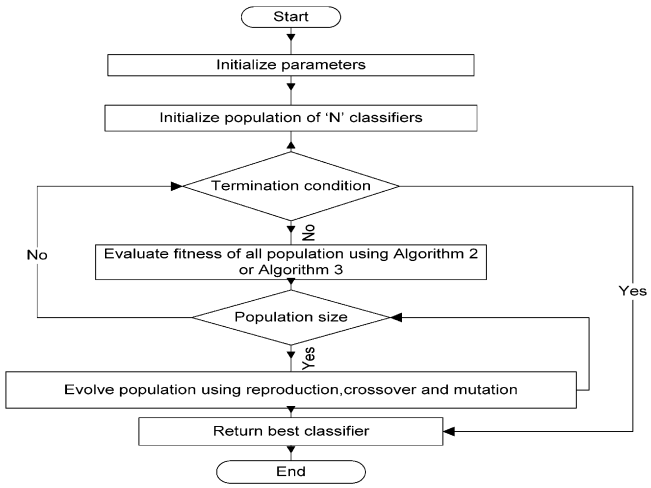

Fig. 5. Algorithm L2GP.

Table 1
GP parameters.

Parameter	Value
Initialization method	Ramped half and half(depth 2–)
Outer logical layer function set	AND, OR, NOT
Arithmetic inner layer function set	+, *, −, / (protected division)
Inner logical layer function set	AND, OR, NOT
Inner tree maximum depth	4
Outer tree maximum depth	4
Population size 'N'	600
Termination criteria	User defined generations(250) or 100% training accuracy
Arithmetic tree terminals	Numerical attributes, Constants[0,10]
Logical tree terminals	Categorical attribute values
Mutation probability	0.25
Reproduction probability	0.25
Crossover probability	0.5
Mutation criteria	Random
Crossover selection	Tournament Selection with size 7
Reproduction criteria	Fitness proportionate selection

4. Results

We have used ten-fold-cross-validation method to obtain the classification results. The data is divided into ten equal parts, nine parts are used for training and one part is used for testing phase, this process is repeated to keep each of the ten parts as testing data once. The ten-fold-cross-validation process is repeated 5 times. For each new fold, we have performed two independent runs with different random seeds. This means that, for each dataset, there are total 100 runs. All the reported results have been averaged for testing data for these 100 runs [35,45]. Table 1 lists the parameters used in the layered GP system.

The datasets used in the empirical analysis have been taken from the UCI ML repository [5]. The properties of data sets are mentioned in Table 2. In this paper, our investigation is restricted to binary classification problems only.

Different performance measures have been used, to analyze the proposed approach. These measures are [44]:

4.1. Accuracy

Accuracy is the prediction rate of a classifier and shows the percentage of the number of instances correctly classified.

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn} \tag{7}$$

where tp = true positives, tn = true negatives, fp = false positives and fn = false negatives.

Table 2
Datasets properties.

Dataset	Instances	Attributes	Distribution
Japanese credit (CRD)	690	15, 6(continuous), 9(categorical)	44.5, 55.5
Australian credit (AUS)	690	14, 6(numerical), 8 (categorical)	44.5, 55.5
German credit (GER)	1000	20, 7(numerical), 13 categorical	30, 70
Heart disease (HRT)	270	13, 6(real), 7(binary, nominal, ordered)	44.4, 55.6
Hepititus (HEP)	155	20, 5(numerical), 15(categorical)	20.6, 79.4

Table 3
Average results (test) achieved using accuracy as fitness function.

Data sets	tp	tn	fp	fn	Accuracy	Sensitivity	Specificity	Auc	Auc(acc)
AUS	30.7	30.9	3.0	3.2	90.79	0.90	0.91	0.90	1.00
GER	07.0	72.0	0.0	21	79.00	0.25	1.00	0.62	0.79
HRT	10.5	11.0	2.0	2.9	82.67	0.76	0.84	0.80	0.99
CRD	28.2	28.6	4.3	4.7	86.26	0.85	0.86	0.86	1.00
HEP	06.7	05.4	2.2	0.7	81.00	0.91	0.71	0.80	0.99

Table 4
Average results (test) achieved using AUC as fitness function.

Data sets	tp	tn	fp	fn	Accuracy	Sensitivity	Specificity	Auc	Auc(acc)
AUS	32.0	28.0	2.0	5.0	90.0	0.86	0.93	0.90	1.00
GER	15.0	75.0	4.0	6.0	89.8	0.71	0.94	0.83	0.92
HRT	10.5	11.0	2.0	2.0	81.2	0.80	0.80	0.80	0.99
CRD	28.3	29.2	3.7	4.6	87.3	0.86	0.89	0.87	1.00
HEP	10.0	03.0	0.8	2.0	82.2	0.83	0.78	0.81	0.98

4.2. Sensitivity

The sensitivity is the probability that a positive test, results in positive. High sensitivity means a large number of positive results are correctly detected. It is also known as TPR.

$$\text{Sensitivity} = \frac{tp}{tp + fn} \tag{8}$$

4.3. Specificity

It measures the proportion of negatives identified as negatives correctly.

$$\text{Specificity} = \frac{tn}{tn + fp} \tag{9}$$

4.4. Area under the convex hull

Area under the convex hull (AUC) has been defined as a measure that correctly evaluates the discriminating power of a classifier and acts as a better measure for classifier efficiency than the traditional classification accuracy, especially for imbalanced data sets.

$$\text{Area under the convex hull (AUC)} = \left(\frac{1}{2}\frac{tp}{tp + fn} + \frac{tn}{tn + fp}\right) \tag{10}$$

4.5. Area under the convex hull combined with accuracy

This is a new, finer measure that makes use of some other methods in evaluating performance. It is correlated with the root mean square error.

$$\text{AUC (ACC)} = \frac{\text{sensitivity+specificity}}{2 * \text{Accuracy}} \tag{11}$$

Tables 3 and 4 present the classification results for the five data sets using two fitness measures ACC and AUC. The accuracy in both cases is comparable except for GER and HEP data. The reason is imbalanced class distribution in these datasets. Here, accuracy does not represent better discrimination power of a classifier therefore, it results in less value of AUC. In Table 4, we can see that better value of AUC has also resulted in better accuracy, but vice versa is not true for imbalanced datasets. Hence, the cases where the data is nearly balanced, there is no-noticeable difference in ACC and AUC of classifiers. As mentioned earlier, the reported results are average of best individuals obtained in 100 executions for each dataset after 250 generations.

Table 5 presents an example of two layered rule for German credit data set, which contains 7 numerical and 13 categorical attributes. In this rule, there are two leaf nodes in the outer layer, one is an arithmetic expression and the other is a logical expression.

Table 6 shows the average accuracy and respective standard deviations during training and testing phase for all datasets. These results are reported after every 20 generations, shown in the first row of the table. The result for testing is generated by extracting the best classifier after given generations and testing its performance on test data.

Fig. 6 presents the behavior of average best population members during the evolution. We can note increase in average accuracy of trees for all datasets during evolution. This increase reaches a stable state around 200 iterations.

We have also compared the average results of best evolved classifiers with other GP based classification approaches. It can be observed that the presented approach yields compatible results with respect to accuracy. In Table 7, some techniques [41,46] involve preprocessing while others [26,47] use constrained syntax GP, to incorporate mixed-type data.

This can be noted that the L2GP approach has produced compatible results when compared to other GP based techniques while

Table 5
Two layered rule for German credit data.

Outer Layer	(NOT (T1))OR (T2) (outer layer)
T1	A15 = "A151" Logical Inner Tree
T2	(((A18/A8)/A2) + (A5 * (A18 − A16)) * A13) Arithmetic Inner Tree

Table 6
Average accuracy of population for training and testing data.

Generations	20.00	40.00	60.00	80.00	100.00	120.00	140.00	160.00	180.00	200.00	220.00	240.00	
AUS	Train%	80.97	82.35	83.73	83.92	84.83	85.39	86.39	87.67	89.65	91.91	92.37	93.84
	Stdev	00.16	00.27	00.33	00.42	00.54	00.59	00.67	00.67	00.74	00.76	00.74	00.71
	Test%	81.43	82.70	83.08	83.40	84.03	84.41	85.32	86.17	88.20	88.65	89.62	90.00
	Stdev	00.21	00.21	00.10	00.42	00.31	00.10	00.10	00.00	00.31	00.31	00.21	00.10
GER	Train%	56.31	59.60	61.28	63.37	64.79	67.00	71.00	81.00	85.00	86.00	89.00	90.00
	Stdev	00.01	00.28	00.21	00.17	00.26	00.31	00.31	00.31	00.40	00.38	00.47	00.42
	Test%	58.02	60.98	63.02	65.29	66.45	67.32	69.34	74.59	83.89	88.25	88.81	89.89
	Stdev	00.00	00.43	01.71	01.93	01.21	01.00	01.50	01.86	01.86	01.64	01.28	01.21
HRT	Train%	68.10	71.47	73.25	75.72	76.92	78.13	80.53	81.97	83.89	84.62	86.06	89.42
	Stdev	00.05	00.28	00.55	00.54	00.38	00.34	00.41	00.54	00.62	00.61	00.58	00.58
	Test%	69.84	71.35	72.29	73.62	73.89	75.22	76.02	77.21	77.88	78.28	79.60	81.20
	Stdev	00.41	00.41	00.60	00.73	00.81	00.53	00.20	00.09	00.10	00.37	00.98	01.39
CRD	Train%	80.87	83.62	84.75	85.32	86.36	87.41	87.88	88.35	89.83	91.64	91.92	91.92
	Stdev	00.02	00.12	00.07	00.09	00.05	00.02	00.02	00.02	00.09	00.07	00.12	00.09
	Test%	82.39	83.40	83.70	84.18	84.67	85.15	85.25	85.78	86.71	87.27	87.27	87.28
	Stdev	00.21	00.43	02.79	03.22	01.50	00.86	00.64	00.85	00.43	00.22	00.22	00.86
HEP	Train%	55.36	60.42	69.35	72.32	81.25	85.71	86.61	95.54	98.00	98.30	98.30	99.40
	Stdev	00.15	00.22	00.25	00.25	00.2	00.12	00.07	03.05	02.00	01.70	01.70	00.60
	Test%	57.29	61.89	65.36	68.14	73.35	73.87	76.47	81.68	82.20	82.20	82.20	82.20
	Stdev	01.01	01.01	01.01	00.76	00.51	00.76	01.01	01.52	02.53	03.54	04.55	05.05

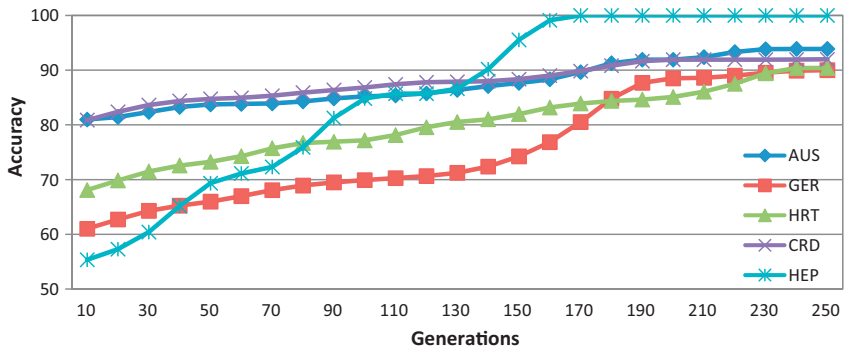

Fig. 6. Evolution of best trees during training.

Table 7
Comparison of classification results with other techniques.

Data set	L2GP (average accuracy of best classifiers %)	Others (accuracy %)
AUS	90	86.3 [21], 85.1 [26], 86 [41], 88.1 [46], 88.3 [42], 90 [47], 88 [48], 83.4 [49]
GER	89.8	72.9 [21], 77.4 [42], 78 [47], 76 [48]
HRT	81.2	77.9 [13], 78.7 [21], 82.2 [24],
CRD	87.3	84.8 [13], 84.7 [24]

enjoying the benefit of discarding the need of any manipulation on data.

5. Conclusions and future work

In this paper, we have proposed a novel GP based classification technique that operates upon data with mixed type of attributes. The technique does not require any transformation or preprocessing of the data. We have tested the system on several benchmark datasets and compared the performance with various GP based classification methods. The results have revealed that the presented technique offers compatible performance owing to its flexible two layered representation. The future works include application of this technique for multi-class classification problems.

Acknowledgement

The authors would like to thank Higher Education Commission, Pakistan for the financial support and providing the opportunity to perform this research.

References

[1] H.S. Lopes, Genetic programming for epileptic pattern recognition in electroencephalographic signals, Applied Soft Computing (2007) 343–352.

[2] A. Song, M.I. Heywood, Training genetic programming on half a million patterns: an example from anomaly detection, IEEE Transactions on Evolutionary Computation 9 (3) (2005) 225–239.
[3] G. Potgieter, A.P. Engelbrecht, Evolving model trees for mining data sets with continuous-valued classes, Expert Systems with Applications 35 (4) (2008) 1513–1532.
[4] T.E. McKee, T. Lensberg, Genetic programming and rough sets: a hybrid approach to bankruptcy classification, European Journal of Operational Research 138 (2) (2002) 436–451.
[5] A. Asuncion, D.J. Newman, Machine Learning Repository (2007), http://archive.ics.uci.edu/ml/.
[6] J.R. Koza, Genetic Programming: On the Programming of computers by Means of Natural Selection, MIT Press, MA, Cambridge, 1992.
[7] J.R. Koza, Concept Formation and Decision Tree Induction Using the Genetic Programming Paradigm. Lecture Notes in Computer Science, vol. 496, Springer-Verlag, 1991, pp. 124–128.
[8] Q. Li, et al., Dynamic split-point selection method for decision tree evolved by gene expression programming, in: IEEE Congress on Evolutionary Computation, IEEE Press, 2009.
[9] D. Rivero, J.R. Rabunal, A. Pazos, Modifying Genetic Programming for Artificial Neural Network Development for Data Mining. Soft Computing, vol. 13, Springer-Verlag, 2008, pp. 291–305.
[10] M.D. Ritchie, et al., Genetic programming neural networks: a powerful bioinformatics tool for human genetics, Applied Soft Computing (2007) 471–479.
[11] A. Tsakonas, A comparison of classification accuracy of four genetic programming-evolved intelligent structures, Information Sciences (2006) 691–724.
[12] M. Oltean, L. Diosan, An Autonomous GP-Based System for Regression and Classification Problems. Applied Soft Computing, vol. 9, Elsevier, 2009, pp. 49–60.
[13] G.A. Pappa, A.A. Freitas, Evolving rule induction algorithms with multiobjective grammer based genetic programming, Knowledge and Information Systems (2008).
[14] J. Eggermont, Evolving fuzzy decision trees for data classification, Proceedings of the 14th Belgium Netherlands Artificial Intelligence Conference (2002).
[15] R. Konig, U. Johansson, L. Niklasson, Genetic programming – a tool for flexible rule extraction, IEEE Congress on Evolutionary Computation (2007).
[16] 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.
[17] E. Carreno, G. Leguizamon, N. Wagner, Evolution of classification rules for comprehensible knowledge discovery, IEEE Congress on Evolutionary Computation (2007) 1261–1268.
[18] A.A. Freitas, A genetic programming framework for two data mining tasks: classification and generalized rule induction, in: Genetic Programming, Morgan Kaufmann, CA, USA, 1997, pp. 96–101.
[19] C.S. Kuo, T.P. Hong, C.L. Chen, Applying Genetic Programming Technique in Classification Trees. Soft Computing, vol. 11, Springer-Verlag, 2007, pp. 1165–1172.
[20] 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) 253–254.
[21] 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) 1001–1005.
[22] E. Tunstel, M. Jamshidi, On genetic programming of fuzzy rule-based systems for intelligent control, International Journal of Intelligent Automation and Soft Computing (1996) 273–284.
[23] F.J. Berlanga, et al., A genetic-programming-based approach for the learning of compact fuzzy rule-based classification systems, Lecture Notes on Artificial Intelligence (LNAI) (2006) 182–191.
[24] R.R.F. Mendes, et al., Discovering Fuzzy Classification Rules with Genetic Programming and Co-Evolution. Lecture Notes in Artificial Intelligence, Springer, 2001, pp. 314–325.
[25] B.C. Chien, J.Y. Lin, T.P. Hong, Learning Discriminant Functions with Fuzzy Attributes for Classification Using Genetic Programming. Expert Systems with Applications, vol. 231, Elsevier, 2002, pp. 31–37.
[26] I.D. Falco, A.D. Cioppa, E. Tarantino, Discovering interesting classification rules with GP, Applied Soft Computing (2002) 257–269.
[27] C.C. Bojarczuk, et al., A constrained-syntax genetic programming system for discovering classification rules: application to medical data sets, Artificial Intelligence in Medicine (2004) 27–48.
[28] 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. Lecture Notes in Computer Science, vol. 2610, Springer-Verlag, 2003.
[29] A. Tsakonas, et al., Evolving rule-based systems in two medical domains using genetic programming, Artificial Intelligence in Medicine (2004) 195–216.
[30] J.Y. Lin, et al., Classifier design with feature selection and feature extraction using layered genetic programming, Expert Systems with Applications 34 (2007) 1384–1393.
[31] M. Zhang, V. Ciesielski, Genetic programming for multiple class object detection Australia, in: Proceedings of the 12th Australian Joint Conference on Artificial Intelligence, 1999, pp. 180–192.
[32] D. Parrott, X. Li, V. Ciesielski, Multi-objective techniques in genetic programming for evolving classifiers, IEEE Congress on Evolutionary Computation (2005) 183–190.
[33] W.R. Smart, M. Zhang, Classification strategies for image classification in genetic programming, Proceeding of Image and Vision Computing NZ International Conference (2003) 402–407.
[34] M. Zhang, W. Smart, Multiclass object classification using genetic programming, Lecture Notes in Computer Science (2004) 367–376.
[35] J.K. Kishore, et al., Application of genetic programming for multicategory pattern classification, IEEE Transactions on Evolutionary Computation (2000).
[36] T. Loveard, V. Ciesielski, Representing classification problems in genetic programming, IEEE Congress on Evolutionary Computation (2001) 1070–1077.
[37] W. Smart, M. Zhang, Using genetic programming for multiclass classification by simultaneously solving component binary classification problems, Lecture Notes in Computer Science (2005).
[38] A. Teredesai, V. Govindaraju, in evolving GP based classifiers for a pattern recognition task, IEEE Congress on Evolutionary Computation (2004) 509–515.
[39] C.C. Bojarczuk, H.S. Lopes, A.A. Freitas, Genetic programming for knowledge discovery in chest-pain diagnosis, IEEE Engineering in Medicine and Biology Magazine (2000) 38–44.
[40] P. Lichodzijewski, M.I. Heywood, Pareto-co evolutionary genetic programming for problem decomposition in multi-class classification, in: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, London, 2007.
[41] T. Loveard, V. Ciesielski, Employing nominal attributes in classification using genetic programming, in: 4th Asia Pacific Conference on Simulated Evolution and Learning, Singapore, 2002, pp. 487–491.
[42] C.S. Ong, J.J. Huang, G.H. Tzeng, Building credit scoring models using genetic programming, Expert Systems with Applications 29 (1) (2005) 41–47.
[43] H. Jabeen, A.R. Baig, Depth Limited Crossover in Genetic Programming for Classifier Evolution, Computers in Human Behaviour 27 (5) (2011) 1475–1481.
[44] M. Sokolova, G. Lapalme, Performance Measures in Classification of Human Communications. Lecture Notes in Artificial Intelligence, vol. 4509, Springer-Verlag, 2007, pp. 150–170.
[45] D.P. Muni, N.R. Pal, J. Das, A novel approach to design classifiers using GP, IEEE Transactions of Evolutionary Computation (2004).
[46] K. Badran, P. Rockett, Integrating categorical variables with multiobjective genetic programming for classifier construction, in: European Conference on Genetic Programming, LNCS, Springer-Verlag, 2008.
[47] J.J. Huang, C.S. Ong, G.H. Tzeng, Two-stage Genetic Programming (2SGP) for the credit scoring model Applied Mathematics and Computation, vol. 174(2), Elsevier, 2006, pp. 1039–1053.
[48] Y. Zhang, P. Rockett, A Comparison of Three Evolutionary Strategies for Multiobjective Genetic Programming. Artificial Intelligence Review, vol. 27, Springer-Verlag, 2007, pp. 149–163.
[49] S. Sakprasat, M.C. Sinclair, Classification rule mining for automatic credit approval using genetic programming, in: IEEE Congress on Evolutionary Computation, IEEE Press, Singapore, 2007, pp. 548–555.