Lazy Learning for Multi-class Classification Using Genetic Programming

Hajira Jabeen[1] and Abdul Rauf Baig[2]

[1] Iqra University
[2] National University of Computer and Emerging Sciences
Islamabad, Pakistan
hajira@iqraisb.edu.pk, rauf.baig@nu.edu.pk

Abstract. In this paper we have proposed a lazy learning mechanism for multiclass classification using genetic programming. This method is an improvement of traditional binary decomposition method for multiclass classification. We train classifiers for individual classes for a certain number of generations. Individual trained classifiers for each class are combined in a single chromosome. A population of such chromosomes is created and evolved further. This method suppresses the conflicting situations common in binary decomposition method. The proposed lazy learning method has performed better than traditional binary decomposition method over five benchmark datasets taken from UCI ML repository.

Keywords: Classification, Genetic Programming, Classifier, Expression, Rule, Algorithm.

1 Introduction

Data Classification has received considerable interest in the recent years, due to its applicability in many real world applications like fraud detection, face recognition, speech recognition and knowledge extraction from databases. Data classification is a two-step process. In the first step, data are analyzed to develop a relationship (classifier) among variables to predict class labels. This classifier is tested on unseen data in the second step. The task of data classification is challenging due to unpredictability and varying properties of data.

GP was introduced Koza by [1] in 1992 for automatic evolution of computer programs. Its ability to evolve classifiers has been realized since its inception. Decision trees are one of the simpler classifiers and GP has been successfully used for decision tree evolution [2]. Other classifier evolution approaches include evolution of neural networks [3], autonomous systems [4], rule induction algorithms [5], fuzzy rule based systems and fuzzy Petri nets [6]. 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 [7, 8]. The rule based systems include, atomic representations proposed by Eggermont [9] and SQL based representations proposed by Freitas [10]. Tunsel [11] introduced evolution of fuzzy rules using GP. Chien [12] used fuzzy discrimination function for classification. Falco [13] discovered comprehensible classification rules that use continuous value attributes. Tsakonas [14] introduced two GP based systems for medical domains and achieved noticeable performance. 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 real value as output, that is mapped to class decision. The positive and negative output is used as the threshold for binary classification problems. For multiclass problems, a threshold is applied on the real output of expressions. The methods include static thresholds [15, 16], dynamic thresholds [16, 17] and slotted thresholds [18]. Another method for multiclass classification is binary decomposition or one versus all method. In this method, one classifier for each class is evolved; and best classifiers for each class are used to retrieve final decision. The classifier with positive output or maximum output is declared the winner. Binary decomposition methods have been explored in [19, 20]. On the other hand, relatively different, GA inspired, method for multiclass classification has been proposed by Durga [21], an amalgamated chromosome (vector) of classifiers for all classes is evolved in single GP run.

The drawback of binary decomposition method is conflicting situations. We need some intelligent envelop to cover up these conflicting situations where more than one classifier output a 'belong to' signal or none of the classifier output a 'belong to signal'. Various conflict resolution mechanisms are present in the literature that try to maximize the accuracy based upon weights assigned to classifier, heuristic rules, error correcting output codes etc.

In this paper, we have focused on binary decomposition method and the problem of conflicting situations associated with this method. Next section discusses the proposed methodology.

2 Proposed Methodology

We have divided the algorithm into two phases; the first part is similar to traditional binary decomposition method. The difference in our approach is that we retain whole population for further use rather than saving only best classifier. The output of this phase is a number of classifier populations for each class in the data. Second phase uses this population to populate its individual chromosomes to create final classifier chromosomes using some selection criteria. Once the chromosome population is created, it is evolved in search of better fitness. The output of this phase is a single chromosome classifier having best fitness.

Each classifier is represented in the form of arithmetic relationship between the attributes of the data. For example, consider a dataset with four attributes and two classes. An instance of such dataset can be $[A_1,A_2,A_3,A_4] \in C_1$. A possible classifier for such data would be $(A_3*A_4) / (A_1+A_2)$. Consider the input [1,2,3,4], the classifier will output a real value '4'. The positive output indicates presence of class C_1. This threshold of positive and negative numbers has been used to estimate classification accuracy of a classifier. In case of more than two classes, we accentuate one class (by positive output) and other classes are treated as a single, not desired class (by negative output).

Let ‘n’ be the number of classes present in the data. A classifier is evolved to discriminate between one class and rest classes such that output O of classifier C for class C_i is positive for instances belonging to class C_i and negative for instances not belonging to class C_i, *where i=1…n.*

For each instance, the estimated and actual output is compared, and if both are same, the result is declared accurate. This helps in calculating classification accuracy of each classifier present in the population.

A random population of classifiers is generated using ramped half and half method. The function set for the population is arithmetic operators and terminal set contains attributes of the data and ephemeral constant. We have used three evolutionary operators: crossover, mutation and reproduction. The evolutionary process is repeated for each class. At the end of the first phase, we will have ‘n’ populations representing ‘n’ classes of the data.

After that, we create a new population where a chromosome contains ‘n’ classifiers, one for each class. Each member for this chromosome is selected from one of the ‘n’ populations using tournament selection. This newly created population of chromosomes is evolved for a certain number of generations.

The accuracy of an amalgamated classifier is used as the fitness function for these classifier chromosomes. 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 can be seen in Figure 2.

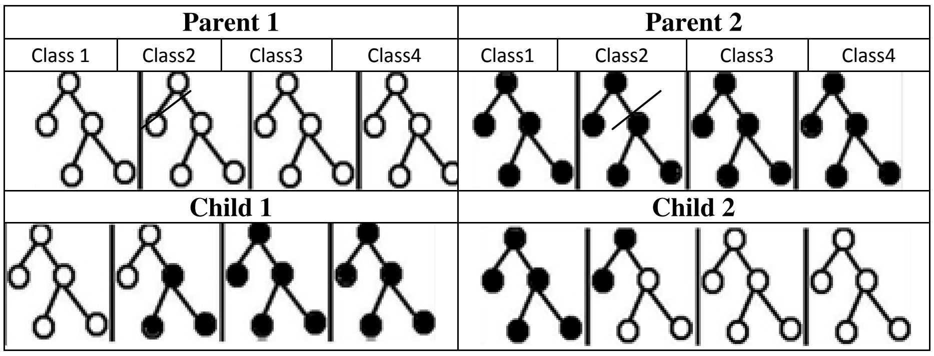

Fig. 1. Crossover between two chromosome classifiers

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

3 Results

We have used five datasets from the UCI ML repository. The properties of datasets used for experimentation is summarized in Table 1 given below.

Table 1. Datasets used for experimentation

Datasets	Classes	Attributes	Type	Instances
IRIS	3	4	Real	150
WINE	3	13	Integer, Real	178
VEHICLE	5	18	Integer	946
GLASS	6	10	Real	214
YEAST	10	8	Real	1484

The parameters used for GP evolution are mentioned in Table 2.

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 A_1,A_2…A_n, Ephemeral Constant [0,10]
Termination Criteria	120 generations or 100% training accuracy of classifier
PhaseI generations	90
PhaseII generations	30

The results are shown in Table 3. BDGP column represent the results obtained by using traditional binary decomposition method [19] and the MTGP presents the Multi-Tree based classification [21]}. The presented values are the classification accuracies in percentage, averaged for 30 GP runs. We have preformed ten-fold cross validation on three different random partitions of the data.

BDGP selects best classifiers from each population and combines them for final classifier. LGP represents the proposed lazy learning mechanism. We can see that the proposed lazy learning mechanism has performed better as compared to traditional binary decomposition method. The major advantage in proposed approach is fewer conflicts which deteriorate the performance of traditional binary decomposition method.

Table 3. Classification Results in Percentage for Different Datasets

Datasets	**Training**			**Testing**		
	BDGP %	**LGP %**	**MTGP %**	**BDGP %**	**LGP %**	**MTGP %**
IRIS	94.4	94.6	93.0	93.2	95.4	92.5
WINE	91.0	91.2	73.0	78.5	83.0	74.5
GLASS	61.2	62.0	49.0	54.7	62.0	52.0
YEAST	37.4	44.3	39.8	34.3	43.3	36.7
VEHICLE	38.8	54.6	40.5	39.2	53.4	42.3

4 Conclusion

In this paper we have proposed a new lazy learning mechanism for multiclass classification for GP. This method reduces the conflicts between individual classifiers for each class increasing the classification accuracy as well as reliability of classifiers.

A drawback of this method is long training times and more evolutions we evolve as many times as classes and once more for combined chromosomes.

Future work includes analysis of conflict resolution methods and some optimization method that could minimize the number of evolutions required for multiclass classification problems.

References

1. Koza, J.R.: Genetic Programming: On the Programming of by Means of Natural Selection, MA, Cambridge (1992)
2. Koza, J.R.: Concept Formation and Decision Tree Induction Using the Genetic Programming Paradigm. In: Schwefel, H.-P., Männer, R. (eds.) PPSN 1990. LNCS, vol. 496, pp. 124–128. Springer, Heidelberg (1991)
3. Rivero, D., Rabunal, J.R., Pazos, A.: Modifying Genetic Programming for Artificial Neural Network Development for Data Mining. Soft Computing 13, 291–305 (2008)
4. Oltean, M., Diosan, L.: An Autonomous GP-based System for Regression and Classification Problems. Applied Soft Computing 9, 49–60 (2009)
5. Pappa, G.A., Freitas, A.A.: Evolving Rule Induction Algorithms with Multiobjective Grammer based Genetic Programming. Knowledge and Information Systems (2008)
6. Eggermont, J.: Evolving Fuzzy Decision Trees for Data Classification. In: Proceedings of the 14th Belgium Netherlands Artificial Intelligence Conference (2002)
7. Konig, R., Johansson, U., Niklasson, L.: Genetic Programming - A Tool for Flexible Rule Extraction. In: IEEE Congress on Evolutionary Computation (2007)
8. Engelbrecht, A.P., Schoeman, L., Rouwhorst, S.: A Building Block Approach to Genetic Programming for Rule Discovery. In: Data Mining: A Heuristic Approach, pp. 175–189. Idea Group Publishing, USA (2001)
9. Eggermont, J., Eiben, A.E., Hemert, J.I.: A Comparison of Genetic Programming Variants for Data Classification. In: Proceedings of the Eleventh Belgium Netherlands Conference on Artificial Intelligence, pp. 253–254 (1999)
10. Eggermont, J., Kok, J.N., Kosters, W.A.: GP For Data Classification, Partitioning The Search Space. In: Proceedings of the 2004 Symposium on Applied Computing, pp. 1001–1005 (2004)

11. Tunstel, E., Jamshidi, M.: On Genetic Programming of Fuzzy Rule-Based Systems for Intelligent Control. International Journal of Intelligent Automation and Soft Computing, 273–284 (1996)
12. Chien, B.C., Lin, J.Y., Hong, T.P.: Learning Discriminant Functions with Fuzzy Attributes for Classification Using Genetic Programming. Expert Systems with Applications 23(1), 31–37 (2002)
13. Falco, I.D., Cioppa, A.D., Tarantino, E.: Discovering Interesting Classification Rules With GP. Applied Soft Computing, 257–269 (2002)
14. Tsakonas, A., Dounias, G., Jantzen, J., Axer, H., Bjerregaard, B.: Evolving Rule-based Systems in Two Medical Domains Using Genetic Programming. Artificial Intelligence in Medicine, 195–216 (2004)
15. Zhang, M., Ciesielski, V.: Genetic Programming For Multiple Class Object Detection. In: Proceedings of the 12th Australian Joint Conference on Artificial Intelligence, Australia, pp. 180–192 (1999)
16. Parrott, D., Li, X., Ciesielski, V.: Multi-objective Techniques in Genetic Programming for Evolving Classifiers. In: IEEE Congress on Evolutionary Computation, pp. 183–190 (2005)
17. Smart, W.R., Zhang, M.: Classification Strategies for Image Classification in Genetic Programming. In: Proceeding of Image and Vision Computing NZ International Conference, pp. 402–407 (2003)
18. Zhang, M., Smart, W.: Multiclass Object Classification Using Genetic Programming. LNCS, pp. 367–376. Springer, Heidelberg (2004)
19. Kishore, J.K., Patnaik, L.M., Mani, A., Agrawal, V.K.: Application of Genetic Programming for Multicategory Pattern Classification. IEEE Transactions on Eolutionary Computation (2000)
20. Loveard, T., Ciesielski, V.: Representing Classification Problems in Genetic Programming. In: IEEE Congress on Evolutionary Computation, pp. 1070–1077 (2001)
21. Muni, D.P., Pal, N.R., Das, J.: A Novel Approach To Design Classifiers Using GP. IEEE Transactions of Evolutionary Computation (2004)