A Framework for Optimization of Genetic Programming Evolved Classifier Expressions Using Particle Swarm Optimization

Hajira Jabeen and Abdul Rauf Baig

National University of Computer and Emerging Sciences,
H-11/4, Islamabad, Pakistan
{hajira.jabeen,rauf.baig}@nu.edu.pk

Abstract. Genetic Programming has emerged as an efficient algorithm for classification. It offers several prominent features like transparency, flexibility and efficient data modeling ability. However, GP requires long training times and suffers from increase in average population size during evolution. The aim of this paper is to introduce a framework to increase the accuracy of classifiers by performing a PSO based optimization approach. The proposed hybrid framework has been found efficient in increasing the accuracy of classifiers (expressed in the form of binary expression trees) in comparatively lesser number of function evaluations. The technique has been tested using five datasets from the UCI ML repository and found efficient.

Keywords: Classifier Optimization, Data Classification, Genetic Programming, Particle Swarm Optimization.

1 Introduction

The potential of GP to efficiently handle the task of data classification has been recognized since its inception [1]. GP has been used for classification in several different ways. It enjoys an outstanding position amongst other classifier evolution techniques like Genetic Algorithms [2]. The GP based classifier evolution has certain advantages over other techniques. The GP evolved classifiers are transparent and comprehensible. The size and constituent elements of a GP tree are not fixed, which offers a flexible search space to probe for the best classification rule. GP is readily applicable to the data in its original form and no transformation of data is required. GP can eliminate attributes unnecessary for classification task, discarding the need of any explicit feature extraction algorithm.

In addition to above mentioned benefits, GP has a well known drawback of increase in population complexity during evolution. In case of classifier evolution one is always interested in evolving simple and comprehensible classifiers because larger trees tend to over fit the training data. On the other hand complex population adds more computation time. These factors raise the need of reducing the computation yet maintain the benefits of flexible classification methodology.

M. Graña Romay et al. (Eds.): HAIS 2010, Part I, LNAI 6076, pp. 56–63, 2010.

In this paper we have proposed a method to optimize the classifier expressions using Particle Swarm Optimization. The classifier expression is an arithmetic expression trained to output a positive real value for one class and negative real value for the other, in binary classification problems. This method has been extensively investigated in the literature [3-5] and found efficient for classification tasks. Our framework proposes addition of weights associated with all the attributes and constants (all leaf nodes) present in a classifier and optimize the values of these weights to achieve better classification accuracy.

The proposed method is efficient in terms of classification accuracy and lesser number of function calls. The proposed framework:-

- Uses a new hybrid classification methodology
- Avoid bloat by limiting the generations for evolution
- Achieve compatible classification results

After a brief introduction in Section 1, Section 2 discusses work relevant to GP based classification and different optimization techniques in GP. Section 3 presents and explains the work done by authors. Obtained results are organized and discussed in Section 4. Section 5 concludes the findings and future work is discussed.

2 Literature Review

GP has been an area of interest for various researchers during the previous years. It had been applied to solve various problems, one of those being data classification. Several methods have been proposed to tackle data classification using GP. They can be broadly categorized into three different types. The first is evolution of classification algorithms using GP. This includes simple algorithms like decision trees which[6,7], or complex algorithms like neural networks [8-10], autonomous systems [11], rule induction algorithms [12], fuzzy rule based systems and fuzzy Petri nets [13], [10]. In the second method GP is used to evolve classification rules[14-18]. The rule based systems include, atomic representations proposed by Eggermont [19] and SQL based representations proposed by Freitas [20]. Tunsel [21], Berlanga [22] and Mendes [23] introduced evolution of fuzzy classification rules using GP. The third method is evolution of discriminating classification expressions. Arithmetic expressions use (real or integer value) attributes of data as variables in the expressions and output a real value that is mapped to class decision. For multiclass problems the real output of expressions are applied thresholds. The methods include static thresholds [24], [25], dynamic thresholds [26], [25] and slotted thresholds [3]. Another method for multiclass 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 all other classes. Binary decomposition methods have been explored in [4], [5].

Particle Swarm Optimization algorithm has been originally formulated by Kennedy and Eberhart in 1995 [27]. It is efficient at solving optimization problems by modeling the sociological principle of animal groupings during their movements. The algorithm usually operates upon set of real multidimensional points scattered in the search

space. These points move with certain velocity in the search space, mimicking bird's flight in search of optimal solution. The velocity of a particle in a given iteration is a function of the velocity of the previous step, its previous best position and the global best position. The algorithm has been compared with various evolutionary algorithms and found efficient in terms of faster convergence. Following are the update equations for particles in standard PSO.

$$V_{i+1} = \omega V_i + C_0 rand(0,1)(X_{lbest} - X_i) + C_1 rand(0,1)(X_{gbest} - X_i) \quad (1)$$

$$X_{i+1} = X_i + V_i \quad (2)$$

Where X_{gbest} is the global best or local best particle and X_{pbest} is the personal best of each particle. The values C_0 and C_1 are problem specific constants.

The Equation (1) is used to update velocity of a particle and Equation (2) is used to update the position of a particle during the PSO evolution process.

GP is a very efficient innovative technique to handle to problem of data classification but it suffers from inefficient code growth (bloat) during evolution. This increases the program complexity during the evolution process without effective increase in fitness. Another issue with GP based classification is long training time, which increases many folds with the increase in average tree size during evolution. In this paper we have proposed a method that eliminates the need of evolving GP for longer number of generations and optimizes the GP evolved intelligent structures using PSO.

3 Proposed Framework

Figure 1 shows the overview of proposed hybrid framework for classification. The first step of proposed technique is to evolve classifier expressions using GP. A classifier is an arithmetic expression trained to output a positive real value for one class and

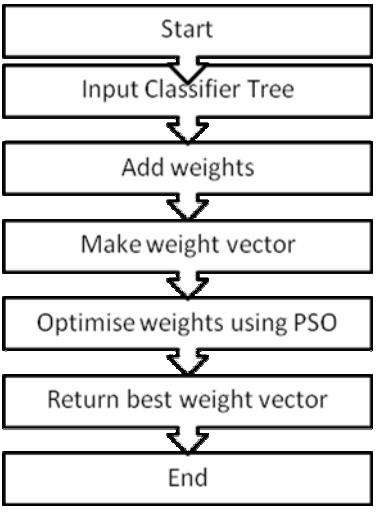

Fig. 1. Expression optimization algorithm

negative real value for the other. In order to clearly portray the efficiency of our proposed hybrid classification technique we have limited our investigation to binary classification problems in this paper.

The classification algorithm is explained as follows:-

```
Step 1. Begin
Step 2. Initialize generations to user defined value
Step 3. Select one class as 'desired' and other 'not
        desired' class
Step 4. Initialize GP-ACE population using ramped half and
        half method
Step 5. While (gen <= generations or Fit_g= 100 )
        a.   Evaluate fitness(accuracy) of each member in
             population
        b.   Find best in population and update P_Best
        c.   Fit_g=fitness(P_Best)
        d.   Perform evolutionary operators
        e.   gen = gen + 1;
Step 6. End while
Step 7. Output P_Best as classifier
Step 8. End
```

The output of above mentioned classification algorithm is a best arithmetic expression where the *fitness* of the expression is its classification accuracy.

The next step is to add weights along all the terminals present in the expression. For example consider an expression $(A_1+A_2)/A_3$ where A_1, A_2, and A_3 are attribute 1, 2 and 3 respectively. This tree will become $[(A_1*W_1) + (A_2*W_2)] / (A_3*W_3)$, after weight addition, where W_1, W_2 and W_3 are weights associated to each terminal. The weight chromosome for this tree will be $[W_1, W_2, W_3]$. Let t be the number of terminals in the classifier expression then the weight vector for it will be :-

$$[W_j] \text{ where } j=1:t \tag{3}$$

This process increases the complexity of the ACE by increasing its depth by '1'. If the number of terminals present in the tree is equal to 't' then the increase in number of nodes in tuned tree is *2't'* where t nodes are function nodes having value '*' and 't' nodes are terminal nodes having weights as their values. This method scales the input of each terminal according to its weight. Let old terminal be T_o and new terminal be T_n, then the value of new terminal would be interpreted as

$$T_{nj}=W_j*T_{oj} \text{ where } j=1:t \tag{4}$$

For the sake of optimization, a population of random weight particles is initialized. These weights are assigned random values between -1 and 1. This creates a multidimensional point in hyper space that has as many dimensions as there are weights in a GP chromosome corresponding to each terminal. PSO is used to evolve these weights for optimal values. The *fitness* of each particle is calculated by putting the values of weights in their corresponding positions and evaluating the accuracy of classifier for training data. We have used cognitive-social model that keeps track of previous best as well as the global best particle. These weight particles are evolved for optimal value for a few generations until termination criteria is fulfilled.

4 Results

The data sets used to test the proposed classification framework are taken from UCI repository. These data sets are Bupa liver disorder, Haberman's survival, Parkinson disease, Pima indians diabetes and Wisconsin breast cancer. All these datasets are real valued data sets with varying number of classes and attributes. This is to prove the effectiveness of the proposed algorithm.

Each tree in the GP evolved classifier chromosome is appended by weights at its terminals, and the weights are evolved using PSO. The results reported in this section have been averaged after tenfold cross validation. PSO has been applied ten times on each single classifier. So the results reported with PSO are averaged for 100 executions on ten different classifier expressions.

Table 1. GP Parameters

S.No	Name	Value
1	Population size	600
2	Generations	120
3	Maximum Depth	5
4	Function set	+ , - , * , / (protected division)
5	Terminal set	Attributes of data, Ephemeral constants

Table 1 lists the GP parameters used for the experimentation. Table 2 lists the parameters used for PSO. The results reported after tuning are averaged for 10 executions of PSO.

Table 2. PSO parameters

S.No	Name	Value
1	No of particles	20
2	Initial value range	[+1 , -1]
3	Number of iterations	30
4	C_1, C_2	1.49
5	W	0.7

4.1 Fitness versus Average Population Size

This is evident from various literature instances that the average population size increases during the GP evolution. In this section we have made a comparison of increase in average number of nodes in population versus average fitness of the population. This comparison has been made to show the effectiveness of proposed methodology that stops the evolutionary process earlier and increases its performance in lesser number of function evaluations. This "less" function evaluations also corresponds to the "simpler" function evaluations when keeping in mind, the graph shown below.

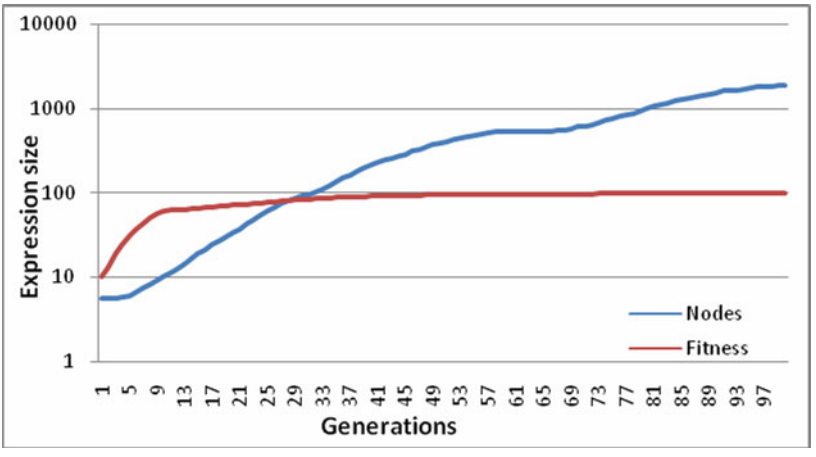

Fig. 2. Increase in expression size during evolution

4.2 Optimization Result

Table 3 presents elementary testing accuracy, after GP evolution, and accuracy achieved after PSO based optimization. We can see that PSO based optimization process has efficiently increased the accuracy of classifier expressions in considerably lesser number of function evaluations. For example in case of Bupa data the accuracy achieved by PSO is 72.38% by optimizing a classifier having 69.25% accuracy. While noting this less number of function evaluations this should also be kept in mind that more number of function evaluations in GP also corresponds to *'more complex"* evaluations due to bloat. Similar trend can be observed in other data sets as well.

Table 3. Increase in accuracy after optimization

Gen#	BUPA		HABER		PARKINSON		PIMA		WBC	
	GP	PSO	GP	PSO	GP	PSO	GP	PSO	GP	PSO
50	68.94%	**72.06%**	77.63%	**82.12%**	79.24%	**83.16%**	66.22%	**68.49%**	94.81%	**96.84%**
100	69.25%	**72.38%**	79.36%	**82.44%**	84.34%	**87.71%**	66.89%	**70.5%**	95.52%	**97.37%**

5 Conclusions

In this paper we have proposed a framework for the optimization of classifier expressions evolved by GP, it has been shown that this method tends to increase the training as well as testing accuracy of the classifiers. This method can eliminate the need for evolving GP classifiers for longer number of generations in search of better accuracy. It also helps in reducing the number of function evaluations desired for GP evolution. The more number of generations in GP also means increase in GP tree sizes over generation, making the task more complex. On the other hand, in case of PSO based optimization we can get better results in much lesser number of function evaluations

at the expense of increase in depth of trees by only one level. This increase in tree complexity gives an attractive reward of increase in corresponding accuracy. Future work includes determination of optimal parameters for PSO for tuning and use of different variants of PSO for tuning.

Acknowledgements

The author Hajira Jabeen would like to acknowledge Higher Education Commission, Pakistan for providing the funding and resources for this work.

References

[1] Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
[2] Flockhart, I.W., Radcliffe, N.J.: GA-MINER: Parallel Data Mining with Hierarchical Genetic Algorithms. University of Edinburgh, Edinburgh (1995)
[3] Smart, W., Zhang, M.: Multiclass Object Classification using Genetic Programming. LNCS, pp. 367–376. Springer, Heidelberg (2004)
[4] Kishore, J.K., et al.: Application of Genetic Programming for Multicategory Pattern Classification. IEEE Transactions on Eolutionary Computation (2000)
[5] Bojarczuk, C.C., Lopes, H.S., Freitas, A.A.: Genetic Programming for Knowledge Discovery in Chest-Pain Giagnosis. IEEE Engineering in Medicine and Biology Magazine, 38–44 (2000)
[6] 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)
[7] Li, Q., et al.: Dynamic Split-Point Selection Method for Decision Tree Evolved by Gene Expression Programming. In: IEEE Congress on Evolutionary Computation. IEEE Press, Los Alamitos (2009)
[8] Rivero, D., Rabunal, J.R., Pazos, A.: Modifying Genetic Programming for Artificial Neural Network Development for Data Mining. Soft Computing 13, 291–305 (2008)
[9] Ritchie, M.D., et al.: Genetic programming Neural Networks: A powerful bioinformatics tool for human genetics. Applied Soft Computing, 471–479 (2007)
[10] Tsakonas, A.: A comparison of classification accuracy of four genetic programming-evolved intelligent structures. Information Sciences, 691–724 (2006)
[11] Oltean, M., Diosan, L.: An Autonomous GP-based System for Regression and Classification Problems. Applied Soft Computing 9, 49–60 (2009)
[12] Pappa, G.A., Freitas, A.A.: Evolving Rule Induction Algorithms with Multiobjective Grammer based Genetic Programming. Knowledge and Information Systems (2008)
[13] Eggermont, J.: Evolving Fuzzy Decision Trees for Data Classification. In: Proceedings of the 14th Belgium Netherlands Artificial Intelligence Conference (2002)
[14] Konig, R., Johansson, U., Niklasson, L.: Genetic Programming - A Tool for Flexible Rule Extraction. In: IEEE Congress on Evolutionary Computation (2007)
[15] Engelbrecht, A.P., Schoeman, L., Rouwhorst, S.: A Building Block Approach to Genetic Programming for Rule Discovery. In: Abbass, H.A., Sarkar, R., Newton, C. (eds.) Data Mining, pp. 175–189. Idea Group Publishing (2001)

[16] Carreno, E., Leguizamon, G., Wagner, N.: Evolution of Classification Rules for Comprehensible Knowledge Discovery. In: IEEE Congress on Evolutionary Computation, pp. 1261–1268 (2007)
[17] Freitas, A.A.: A Genetic Programming Framework For Two Data Mining Tasks: Classification And Generalized Rule Induction. In: Genetic Programming, pp. 96–101. Morgan Kaufmann, CA (1997)
[18] Kuo, C.S., Hong, T.P., Chen, C.L.: Applying genetic programming technique in classification trees. Soft Computing 11, 1165–1172 (2007)
[19] 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)
[20] 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)
[21] 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)
[22] Berlanga, F.J., et al.: A Genetic-Programming-Based Approach for the Learning of Compact Fuzzy Rule-Based Classification Systems. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Żurada, J.M., et al. (eds.) ICAISC 2006. LNCS (LNAI), vol. 4029, pp. 182–191. Springer, Heidelberg (2006)
[23] Mendes, R.R.F., et al.: Discovering Fuzzy Classification Rules with Genetic Programming and Co-Evolution. In: Siebes, A., De Raedt, L., et al. (eds.) PKDD 2001. LNCS (LNAI), vol. 2168, pp. 314–325. Springer, Heidelberg (2001)
[24] 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)
[25] 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)
[26] 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)
[27] Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)