International Journal of Research and Reviews in Computer Science (IJRRCS)
Vol. 3, No. 4, August 2012, ISSN: 2079-2557
© Science Academy Publisher, United Kingdom
www.sciacademypublisher.com/journals/index.php/IJRRCS

Multi-objective Genetic Programming Techniques for Binary Classification

Khaled M. Kridan, Alaa Rohim, and Khaled M. Badran

Computer Department, MTC, Cairo, Egypt

Email: kkridan@yahoo.com, alaa_rohim@yahoo.com, khaledbadran@yahoo.com

Abstract – In this paper, we examine the feature extraction and argue that effective feature extraction can significantly enhance the performance of pattern recognition systems with simple classifiers. Three methods employing multi-objective Genetic Programming (GP) are presented in binary classification problems. In the first place methods (rule based, decision trees and Discrimnant function) are generated. We have applied three GP methods (feature extraction) to real world five datasets from the UCI Machine Learning database to verify approaches. the results of three approaches are compared and conclude that the effective method is to evolve optimal feature extractors that transform input pattern space into a decision space in which maximal class separability is obtained and then its is followed by applying of simple classifiers.

Keywords – Genetic Programming, Feature Extraction, Binary Classification

1. Introduction

1.1. *Feature Extraction*

Many theoretical results have been obtained in the classification domain. Nonetheless, feature extraction retains a key position in the field since the performance of a pattern classifier is well-known to be enhanced by proper preprocessing of the raw measurement data – this topic is the main focus of this work. Fig 1 shows a prototypical pattern recognition system in which a vector of raw measurements is mapped into a decision space. In this paper we focus on comparing between using Multi-objective Genetic Program (MOGP) as a feature extractor and between employing the same technique of MOGP to directly generate classifiers.

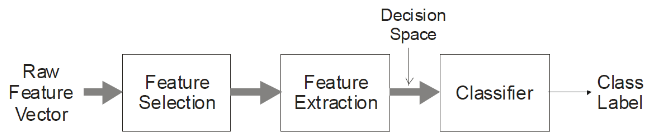

Figure 1. Prototypical pattern recognition system.

Designing feature extraction stage of a classifier usually requires deep domain-specific knowledge. Ideally, we would require some measure of class separability in the transformed decision space to be maximized. Most importantly, domain-independent feature extraction methodology exists to create or search for good feature extractors.

Often the focus here is on reducing the dimensionality of the problem by projecting the data down onto a sub-space which captures the greatest amount of variability. Even within the constraint of a fixed training set, optimality is hard to guarantee with such methods.

For feature selection, we are using wrappers methods. Wrapper methods, as illustrated in Figure 2, use a search strategy to iteratively navigate in feature subset space until a stopping condition is met, while evaluation of the subset selection uses the feedback obtained from the classifier used in order to optimise the performance of the classifier. Wrapper methods use cross-validation techniques or set performance bounds to validate the performance of the classifier. Wrapper methods can find the most useful features but are prone to over-fitting.

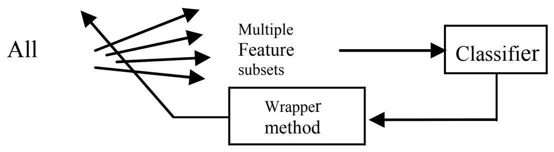

Figure 2. Prototypical Wrapper feature selection approach[14].

In the sense that the feature extraction preprocessing stage is a transformation or mapping from input space to decision space, for a given classification problem we seek the mapping which maximizes the separability of the classes in decision space. Thus the feature extraction can be regarded as finding an optimal sequence of operations subject to some criterion.

1.2. *Genetic programming (GP)*

It's an evolutionary learning technique that offers a great potential for classification. GP is a very flexible heuristic technique that allows us to use complex pattern representations such as trees. For example, any kind of operation or function can be used inside that representation and domain knowledge can be used in the learning process.

The application of GP to classification offers some interesting advantages, the main one being its flexibility, which allows the technique to be adapted to the needs of each particular problem. GP can be employed to construct classifiers using different kinds of representations (decision trees, classification rules and discriminant functions). GP can be useful not only for inducing classifiers, but also for other preprocessing and post processing tasks aimed at the enhancement of classifiers.

Indeed, GP has been used before to optimize feature extraction and selection [2][3]. Sherrah *et al.* [5] proposed an Evolutionary Pre-Processor (EPrep) system which used GP to evolve a good feature mapping by minimizing misclassification error.

1.3. Bloat in Genetic Programming

The chromosomes in a genetic programming (GP) population will 'bloat' – that is, grow without limit without any accompanying improvement in fitness. Research on the causes of bloat has been recently summarised in [11]. Indeed, Langdon and Poli [6, 10] have shown that any variable-length representation suffers from bloating. They have summarised the three main approaches to control bloat in GP:

- Limiting the maximum permissible tree depth (or size) to a pre-defined value.
- Tailoring the genetic operators.
- Employing parsimony to exert selective pressure which favours smaller trees.

In this paper we focus on the parsimony pressure approach. It use a multi-objective method in which the (strictly) non-commensurable objectives of problem-specific error and tree complexity are handled in a Pareto optimisation framework [12]. The results from the Pareto framework is not a single unique solution but a set of equivalent solutions which lie on a Pareto front (or surface) in objective space and which delineate the fundamental trade-offs in the problem. No point on the Pareto front can be modified to improve one objective without simultaneously degrading another. Multi-objective GP (MOGP) has a number of advantages: As well as controlling bloat very effectively, it does not require a pre-determined depth-limit parameter and the tree depth is free to adjust to suit the problem at hand.

This paper is organized as follows: We present our generic framework to evolve optimal feature extractors with multiple objectives in Section 2. Section 3 and section 4 represent GP implementation and experimental results. Comparison with three Genetic programming classifiers is also introduced by other researchers. Conclusion is given on Section 6.

2. Methodology

2.1. Multiple Objectives

Within the multiobjective framework, we have used a two-dimensional fitness vector of objectives comprising: Tree complexity and misclassification error as follows:

2.1.1. Tree complexity measurement

As pointed-out above, there is a danger that trees evolved by GP will become very large due to tree bloat. The huge trees could produce an extremely small error over the training set but a very poor error estimated over an independent validation set.

Broadly, for a given training error, the simpler individual is preferred. Thus we have used node count in the tree as a straightforward measure of tree complexity to be one of our fitness vector elements driving the evolution. Thus we impose a selective pressure that favors small trees, all other things being equal.

2.1.2. Misclassification error

The second element we use in the fitness vector is the conventional one of the fraction of misclassified patterns counted over the training set.

Since we are projecting the input pattern to a one-dimensional decision space we use a simple threshold classifier. The threshold is adapted as part of the fitness value to give the minimum error. This means we are trying to evolve a feature extractor which maps the original pattern space into a new feature space where thresholding is able to yield the smallest possible misclassification.

2.2. Multiple Objectives Genetic Programming (MOGP)

The optimal feature extraction algorithms are designed in a pattern recognition system. Instead of designing a single uniform classifier, MOGP is proposed to automatically adapt the classification paradigm to individual problems. Kumar and Rockett [6] proposed the Pareto convergence genetic algorithm (PCGA) used an elitist, steady-state strategy. In each generation, only two individuals are generated and added to the population to replace the worst two individuals. The ranking mechanism of Fonseca and Fleming [7] was used. Roulette wheel section was employed to select two individuals for crossover. Mutation was always applied to the results of the crossover operation. Their method improved the sampling on the Pareto front to ensure convergence of the evolution without the need for sharing/niching techniques with significantly less computational effort.

In the proposed binary tournament selection, we randomly select two trees from the union of the population and the non-dominated set. If both trees have been drawn from the same set we compare the normalized fitnesses to determine a winner. If not, we use the raw fitnesses to decide which should be chosen.

We use non-destructive, depth-dependent crossover [4] in order to avoid the breaking of building blocks. The relative contributions from mutation and crossover to effective search have been discussed at length elsewhere. We have used a depth-dependent crossover and mutation operators in the current implementation. We choose a sub-tree based on its complexity (*i.e.* the number of nodes) using the *depth-fair* operator [4]. Thus one of the sub-trees at the chosen depth will be picked by roulette wheel selection, biased in its complexity. Having chosen depth, *d* in a tree, there are *N* sub-trees, each comprising *M1, M2,…, MN* nodes, respectively. The probability of selecting the *i*-th sub-tree is given

$$\Pr[i] = \sum_{j=1}^{i} M_j / M \qquad i \in [0 \ldots N] \tag{1}$$

The crossover operation is self-explanatory; in mutation, the selected sub-tree is then replaced by a new, randomly created sub-tree attached at the original mutation point. We retain only those offspring which dominate either of their parents. In this way, we were able to maintain diversity in the population and avoid being trapped in local minima in the early stages.

3. Genetic Program Implementation

As a basis for comparison with MOGP, we have used three model extractions [15] (Decision trees, Rule base and Discrimint Functions).

3.1. Decision tree

A decision tree contains zero or more internal nodes and one or more leaf nodes. All internal nodes have two or more child nodes. All internal nodes contain splits, which test the value of an expression of the attributes(X) as show in figure.4. Arcs from an internal node *t* to its children are labeled with distinct outcomes of the test at *t*. Each leaf node has a class label associated with it.

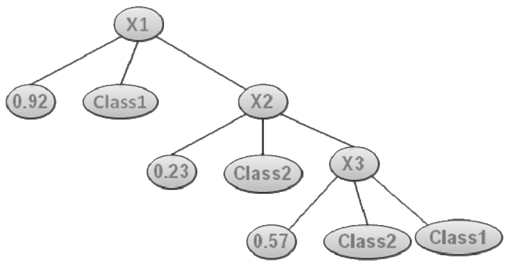

Figure 3. Decision Tree representation.

3.2. Rule-Based

Rules are a simple and easily interpretable way to representation. The rule has two parts, the antecedent and the consequent. The rule antecedent contains a combination of conditions for the predicting attributes(X). Typically, conditions form a conjunction by means of the AND logical operators, but in general any logical operator can be used to connect elemental conditions. The rule consequent contains the value predicted for the class. This way, a rule assigns a data instance to the class pointed out by the consequent if the values of the predicting attributes satisfy the conditions expressed in the antecedent; hence a classifier is represented as a rule set.

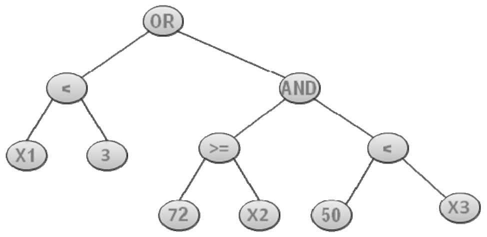

Figure 4. Rule Base representation.

3.3. Discriminant functions

Discriminant functions are mathematical expressions in which different kinds of operators are applied to the attributes(X) of a data instance that must be classified. A single output value is computed from the operations performed on the values of the attributes. The value computed by the function indicates the class predicted. Usually, this is accomplished by means of a threshold or set of thresholds. For binary classification problems, a single function is enough; if the output value is greater than a given threshold, the example is assigned to a certain class, otherwise it is assigned to the other one.

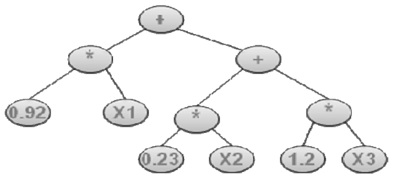

Figure 5. Discriminant Functions representation.

- Generation Decision Tree representation: has done with some rule where the tree divided to three leaf
  Left leaf : should be constant node
  Middle leaf : Ternary Node or Class Node
  Right leaf: Ternary Node(feature) or Class Node
- Generation Rule Based representation: where tree divided in to binary subtree (leafs) where each leaf has logical function ,for last sub tree has comparative function with binary terminal node which represented as feature or constant node.
- Generation Discriminant Functions representation: where tree divided in to binary subtree (leafs) where each leaf has mathematical function ,for last sub tree has terminal node which represented as feature or constant node.
- Crossover Decision Tree representation: depends on selected leaf if
  Left leaf : the crossover is performed only within corresponding constant node
  Middle leaf : the crossover is performed only within TernaryNode or ClassNode
  Right leaf: the crossover is performed within TernaryNode or ClassNode
- Crossover Rule Based representation:The crossover that performed within constant node should be to corresponding constant node. The crossover that performed within feature node should be to corresponding feature node author wise performed crossover sub tree to sub tree(comparative or logical function node).
- Crossover Discriminant Functions representation:The crossover that performed within constant node or feature node should be to corresponding constant or feature node. The crossover performed sub tree should be to sub tree(mathematical function node).
- Mutation Decision Tree representation: depends on selected leaf if :
  Left leaf : the mutation is performed only within corresponding constant node
  Middle leaf : the mutation is performed only within Ternary Node or Class Node
  Right leaf: the mutation is performed within Ternary Node or Class Node
- Mutation Rule Based representation: depends on selected leaf if :
  Right leaf :if constant node the mutation is performed only within constant node.
  Left leaf: if feature node the mutation is performed within feature Node
  For any leaf :if comparative or logical operation node the mutation is performed only within comparative or logical node.
- Mutation Discriminant Functions representation: The

mutation that performed within constant node or feature node should be to constant or feature node. The mutation performed sub tree to sub tree(mathematical function node).

4. Experimental Results

In this section we examine the three techniques performance across a representative range of two-class classification problems from the UCI Machine Learning database [1].

We consider an extensive set of comparisons across five datasets with two-class learning problems. For each dataset we make a statistical comparison of the classification performance between the three MOGP tree representations and a range of established classifiers.

4.1. The UCI Datasets

The five datasets used in the current work are from the UCI Machine Learning database [1]:

Table 1. UCI Dataset

Name	Features	Size and Distributions
Glass	9	153 = 87 (float) + 76 (non-float)
BUPA	6	345 = 200 (Benign) + 145 (Malignant)
AUS	30	569 = 357 (Benign) + 212 (Malignant)
PID	7	532 = 355 + 177 (Diabetic)
WBC	10	699 = 458 (Benign) + 241 (Malignant)

4.2. Ten-Fold Cross Validation

This estimation will depend on the dataset partitioning method applied. TEN-fold cross-validation is commonly used to obtain a more accurate estimate of the empirical risk. In Ten-fold cross-validation the dataset is divided into ten disjoint subsets. One of the subsets is used as the test set and the training dataset is formed using the other N-1 subsets. The cross-validation process is then repeated ten times and the average of the errors estimated over these ten folds is computed. This method reduces the sensitivity of the estimation to how the data gets divided as eventually every example in the dataset is used both for training and testing, but this increase the computational effort needed to make an evaluation. Another variation of this method can be done by splitting the dataset into two folds and repeating this for n different splitting. For each splitting, one of the datasets is used as a training set and the other as the test data. Then the experiment is repeated interchanging the roles of the datasets.

4.3. Statistic

The mean of the test errors for three classification algorithms across the five datasets are summarized in Table 2. Here we have counted the number of the node in the trees and calculated the average of the errors over each test fold of the dataset for every classifier. Reassuringly, the MOGP algorithms return the lowest mean error for each dataset with number of nodes. Overall results show that the Discriminant Functions algorithm are better than of the other algorithms. Although this result needs to be treated -with some caution to be sure the statistical significance of these differences is clear.

Table 2. mean error comparisons of classifiers on each dataset

Dataset	Classifiers					
	MOGP RB	No. node	MOGP DF	No. node	MOGP DT	No. node
Glass	0.385953	7	0.134734	55	0.172779	34
BUPA	0.276301	59	0.213873	65	0.234104	40
PID	0.265365	55	0.190104	65	0.211719	43
WBC	0.0867	13	0.015227	15	0.019032	25
AUS	0.129565	33	0.132464	22	0.115087	25

In order to quantitatively compare statistical significance of the cross-validation experiments we have computed the Alpaydin F-statistic for the three MOGP algorithms compared with each other. The comparisons are summarized in Table3 where MOGP_DF represents superiority over the other algorithms.

Table 3. F-statistic comparisons of classifiers on each dataset

Dataset	Classifiers		
	RB_DF	RB_DT	DF_DT
Glass	87.4	216.1	1.7
BUPA	32.66	9.73	2.57
PID	30.4	62.71	4.361
WBC	597.4	135.93	2.539935
WDBC(AUS)	16.9	1.557	3.16

5. Comparisons with Interpretation

The mean error rates from these earlier studies are summarized, in Table 4. Typically, error rates were estimated using ten-fold cross-validation. The results of Bot & Langdon [8] are the mean validation error of the best individual of 30 runs. Table shows that MOGP has best result over all datasets.

Table 4. Reported error rates for other evolutionary feature detection / classification algorithm [21]

Trainig algorithm	Dataset			
	Glass	BUPA	PID	WBC
Muni	N/K	0.3007	N/K	0.0281
Bot	0.48	0.416	0.305	N/K
Bot & Langdon	0.368	N/K	0.25	N/K
Krawiec	0.3361	N/K	0.2359	N/K
Loveard	N/K	0.308	0.242	0.032
Lim MIN	N/K	0.29	0.22	0.03
MOGP	0.2271	0.2644	0.2057	0.02634

Although one of our multiple objectives has been tree size, used to suppress tree bloat, a number of the trees are not of the absolute minimum size and contain a few redundant sub-trees. In Figs. 6 – 10 show the trees which have been generated by the evolutionary algorithms.

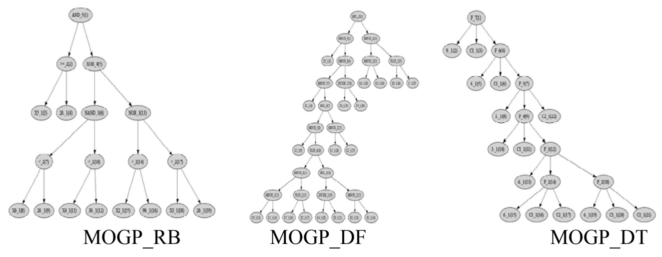

Figure 6. MOGP transformation evolved for the PID dataset.

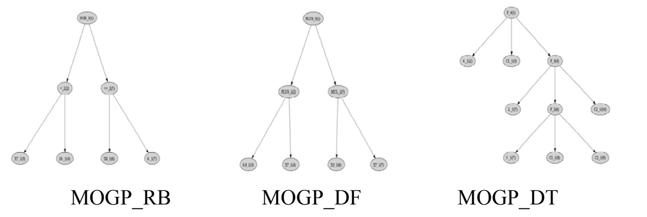

Figure 7. MOGP transformation evolved for the Glass dataset.

It is clear, however, that these are not completely optimal in that the identical classification performance could be obtained in some cases with slightly smaller trees.

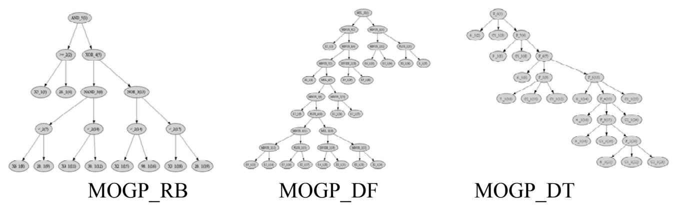

Figure 8. MOGP transformation evolved for the PUPA dataset.

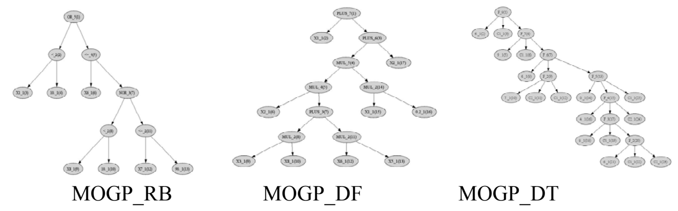

Figure 9. MOGP transformation evolved for the WBC dataset.

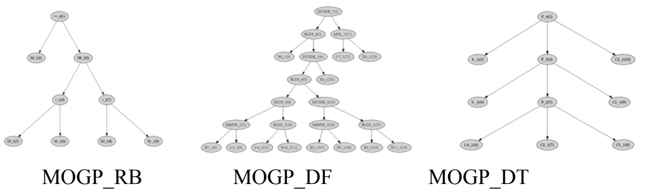

Figure 10. MOGP transformation evolved for the AUS dataset.

Nonetheless, the work presented here produces *near*-optimal trees which are reasonable for a stochastic search method such as genetic programming and an advance on previous work on feature extraction. In practice, any redundant sub-trees could be easily removed by hand from the final solution.

6. Conclusion and future work

In this paper we have demonstrated the use of multi-objective genetic programming (MOGP) to evolve an "optimal" feature extractor which transforms input patterns into a decision space such that pattern separability is maximized.

In comparison with a number of other representative classifier paradigms, the performance of our MOGP method turns out to be better. The use of multiple objectives, particularly classification error objective has been shown to be effective in guiding and speeding the optimization.

The node number objective employed penalizes an individual according to its complexity. This appears to be essential in order to prevent tree bloat as well suppressing over-fitting of the training set leading to poor generalization.

We have applied our three MOGP algorithms to machine learning tasks from the UCI database. The MOGP_DT has less error classification than MOGP_RB and it seems MOGP_DF has better result than others in multi objectives technique.

Finally, developmental GP has the potential to contribute to the discovery and exploitation of knowledge in databases in significant ways[9].although we have treated only binary classification problems in this paper, extension to multiple classes is a logical development and is currently underway.

References

[1] C.L.Blake and C.J.Merz, UCI Repository of machine learning database [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science (1998).

[2] Z.J.Huang,M.Pei, E.Goodman, Y.Huang, and L.Gaoping. "Genetic algorithm optimized feature transformation– A comparison with different classifiers," In *GECCO 2003*, LNCS 2724, June. 2003, pp.2121–2133.

[3] M. Kotani, M. Nakai, and K. Akazawa, "Feature extraction using evolutionary computation," In Proceedings of the Congress of Evolutionary Computation, IEEE Press, July 1999, pp. 1230-1236.

[4] T. Ito, I. Iba, and S. Sato, "Non-destructive depth-dependent crossover for genetic programming," In Proceedings of the First European Workshop on Genetic Programming, LNCS, Paris, April 1998 , pp.14-15.

[5] J.R. Sherrah, R.E. Bogner, and A. Bouzerdoum, "The evolutionary pre-processor: Automatic feature extraction for supervised classification using genetic programming," *Genetic Programming 1997 :* Proceedings of the Second Annual Conference. Stanford University, CA, USA. pp. 304-312, 1997.

[6] M.C.J. Bot and W.B. Langdon, "Application of genetic programming to induction of linear classification trees," in *Proceedings of the Eleventh Belgium/Netherlands Conference on Artificial Intelligence (BNAIC99)*, 1999, pp.107-114.

[7] R. Kumar and P. I. Rockett, "Improved Sampling of the Pareto-Front in Multiobjective Genetic Optimizations by Steady-State Evolution: A Pareto Converging Genetic Algorithm," *Evolutionary Computation*, vol. 10, no. 3, pp. 283-314, 2002.

[8] C. M. Fonseca and P. J. Fleming, "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization," 5th International Conference of Genetic Algorithms, San Mateo, CA, pp. 416-423, 1993.

[9] T. Helmuth, And L. Spector , "Genetic programming Theory and Practice ", New York: Springer ,2012.

[10] W. B. Langdon and R. Poli, "Fitness Causes Bloat: Mutation " 1st European Workshop on Genetic Programming, Paris, pp. 37-48, 1998B.

[11] W. B. Langdon, "The Evolution of Size in Variable Length Representations " IEEE International Conference on Evolutionary Computation pp. 633-638, 1998.

[12] R. Poli, W. B. Langdon, and N. F. McPhee, A Field Guide to Genetic Programming: http://www.gp-field-guide.org.uk, 2008.

[13] Y. Zhang and P. I. Rockett, "Edge Detector Evolution Using Multidimensional Multiobjective Genetic Programming," Department of Electronic and Electrical Engineering, University of Sheffield, Sheffield, UK Technical Report No. VIE 2006/003,2006.

[14] Khaled Badran, "Multi-Objective Programming with an Application to Intrusion Detection in Computer," PHD thesis , University of Sheffield, June 2009.

[15] P. Espejo, S. Ventura and F. Herrera, "Survey On The Application Of Genetic Programming To Classification," IEEE Transaction on System, MAN, and Cybernetics-Part C: Applications and Reviews, VOL. 40, No.2, March 2010.