COS710: Assignment 4

Myron Guanhao Ouyang
Department of Computer Science
University of Pretoria
South Africa
u16008368@tuks.co.za

***Abstract*—This paper analyzes the performance of solving a simple classification problem through a Genetic Programming approach. The results are compared with that of research done previously.**

I. Introduction

A genetic program (GP) and a genetic algorithm is relatively similar in the sense that both of these methods produce a result through evolving a population of individuals.

The objective of this paper is to employ a genetic program to perform multiclass classification on a data set containing different categories of Iris flowers.

The remainder of the paper follows the following format; Section II will be a high level discussion on the implementation of the program. Section III will be a more technical discussion, detailing the specifics of the implementation of the program, Section IV will show the results of the implementation and finally Section V will conclude the results of this paper.

II. Background

The background section will give a brief overview of the general set up of the genetic program.

A. Genetic Programming

A genetic program follows the sames steps as that of a genetic algorithm; an initial population is generated and it is slowly improved through a process of selection, cross-over and mutation of the individuals of the population. These steps are repeated on the population for a certain number of generations and after each generation the overall fitness of the population improves; as only the individuals with the 'better' fitness gets to move onto the next generation.

The GP is normally considered a subset of Genetic Algorithms (GA) with the difference being that individuals of a GP make use of more complex representations; such as trees containing function nodes and terminal nodes [1]. GP are also less likely to result in invalid states and the tend to have variable length representations as opposed to that of GA

B. Classification

To classify a data set means to be able to take in inputs with certain attributes and properly categorize the collection of attributes into a set of classifications.

A classifier is a model created to solve classification problems, it should be able to interpret the values of the input's attributes and correctly assign a classification for the input [1].

C. Classification using GP

A classification tree is built through GP. An initial population of classification trees are first generated, their fitness is based on the the number of correct classification they make on a set of data. Genetic operators are applied to the population to obtain children for the next generation. The majority of the next generation will be composed of individuals with a better fitness than those of the previous generation, eventually the population will comprise of classification trees that have a high percentage of accurately classifying the data.

III. Implementation

This section describes, in detail, the steps and thought processes followed to implement a GP to generate a classification tree.

A. Data

The data used in this report was obtained from https://archive.ics.uci.edu/ml/datasets/Iris. The data set contains 150 records of 3 types of iris plant, distributed evenly (50 records per plant). Each plant record consists of 4 attributes; petal length, petal width, sepal length and sepal width, all measured in centimeters. The objective of this report is to properly categorize each plant into their respective class based on their 4 attributes.

B. GP algorithm

The general pseudo code for evolutionary algorithms is as follows:
Genetic Programming:

1) Generate an initial population
2) Apply genetic operators on population, to produce potential candidates for the next generation
   - Selection
   - Cross-over
   - Mutation
3) Process each candidate through a fitness function
4) Take results from previous step and create a new population for the next generation with a bias towards individuals with a better fitness
5) Repeat Step 2-4 until population has converged or an upper limit has been reached

In this implementation, an individual is represented as a binary classification tree. This tree is made up of 2 types of

nodes; functional and terminal. The functional nodes represent the attributes of the plant and the terminal nodes represent the possible classes of the plant. Terminal nodes are always leaf nodes. Each functional node is also assigned a 'Compare Value' variable, this is a randomly generated value between the range of the min and max values for each of the respective attributes.

A record is classified by the tree as follows:
Traverse Classification Tree:

1) Start at root node
2) Compare appropriate plant attribute value to the 'Compare Value' variable of the current node
3) Visit left child of node if attribute is less than or equal to 'Compare Value'
4) Visit right child of node if attribute is greater than 'Compare Value'
5) Repeat Step 2-4 until terminal node is reached

The fitness for each tree is determined by the total number of correctly classified plants, therefor a higher fitness value is better.

A Full Method was applied when generating the initial population, of 100 individuals, each tree was instantiated with an initial height of 4 and it was ensured that all trees contained at least one of each functional and terminal nodes.

The selection method employed was a combination of Elitism and Tournament selection. The top 10% of individuals where directly copied over to the next generation with no modification to the individual. Those individuals are then removed from the current population. Children of specifically selected parents in this generation fill the remainder of the next generations population. The parents are selected through Tournament selection with a K-value of 30, this was done with an emphasis on elitism. With hopes that the program will ignore individuals with low immediate potential.

The cross-over method implemented involved randomly selecting a node in the current tree and randomly selecting another node in its partner tree. These two nodes are then swapped with one another. If the swapped node is the root node of a subtree, the entire subtree is swapped as well. It was noted that this created a greater variety of offspring and produced better overall results. This method was implemented as a hill climber, to ensure that the resulting offspring's fitness is always equal to or greater than one of its parents.

The mutation method implemented was relatively simple, it would traverse the entire tree and have a 20% chance to mutate the current node its on. The mutation would randomly re-assign the node value and the associating 'Compare Value' of the node (depending on whether it is a terminal node or a functional node). This method was also implemented as a hill climber.

This implementation was trained for 100 generations and the best individual was used to test. This entire process (training and testing) was done for 20 epochs. With each epoch consisting of a new starting population.

IV. Research Results

This section describes performance of the training and test results obtained from implementation of the GP.

Two different result sets were obtained. Table 1 shows the results for a ratio of 75% training data 25% testing data.

	Best	**Avg**	**Std**
Train	98.26%	96.96%	0.52780
Test	100.0%	98.99%	1.677551

Table I
Results - 1

Table 2 shows the results for a ratio of 50% training data 50% testing data.

	Best	**Avg**	**Std**
Train	100.0%	98.24%	0.762523
Test	97.26%	95.61%	1.303361

Table II
Results - 2

It can be noted that with a greater set of training data compared to testing data, the result is slightly better than that of equal amounts of training and testing data. This is probably due to the training sets being exposed to more possible edge cases.

It is evident from the standard deviation of both sets of data, that there is higher inconsistencies in the tested data when compared to trained data, but only slightly.

The performance of the evolved classifiers is significantly worse when compared to results of research done previously on the same data set [2] [3]. A likely reason to this larger standard deviation is probably due to the enforced hill climber implementation on the genetic operators, leading to results finding local maxima instead of a global maxima.

V. Conclusion

This paper implemented a Genetic Programming approach to a simple classification problem. Based on the results, the classification tree generated produces results with a 95% accuracy, however the standard deviation of results is comparatively high when compared to results of previous research.

References

[1] Sebastian Ventura Pedro G. Espejo and Francisco Herrera. A survey on the application of genetic programming to classification. 40, 3 2010.

[2] Ayhan Demiriz, Kristin Bennett, and M Embrechts. A genetic algorithm approach for semi-supervised clustering. *International Journal of Smart Engineering System Design*, 4, 02 2002.

[3] Jeroen Eggermont, Joost N. Kok, and Walter Kosters. Genetic programming for data classification: Partitioning the search space. volume 2, pages 1001–1005, 01 2004.

Appendix