UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

COS710 Assignment 2

Genetic Programming and Data Classification

Student Number: u15024522

1 Introduction

This assignment involves employing genetic programming to produce a classifier for postoperative patient diagnosis.

More specifically, the toolset provided by ECJ [1] - a java-based evolutionary computation research system - was used to construct a genetic programming approach to evolve a population of decision trees for classifying postoperative patient diagnosis for the patients provided in the Post-Operative Patient Data Set, which can be obtained from the UCI Machine Learning repository [2].

The aforementioned data set consist of 90 records, each with 9 attributes which can be described as follows:

- L-CORE (patient's internal temperature in C): high (> 37), mid (≥ 36 and ≤ 37), low (< 36)
- L-SURF (patient's surface temperature in C): high (> 36.5), mid (≥ 36.5 and ≤ 35), low (< 35)
- L-O2 (oxygen saturation in excellent (≥ 98), good (≥ 90 and < 98), fair (≥ 80 and < 90), poor (< 80)
- L-BP (last measurement of blood pressure): high ($> 130/90$), mid ($\leq 130/90$ and $\geq 90/70$), low ($< 90/70$)
- SURF-STBL (stability of patient's surface temperature): stable, moderate, unstable
- CORE-STBL (stability of patient's core temperature) stable, moderate, unstable
- BP-STBL (stability of patient's blood pressure) stable, moderate, unstable
- COMFORT (patient's perceived comfort at discharge, measured as an integer between 0 and 20)
- ADM-DECS (discharge decision): I (patient sent to Intensive Care Unit), S (patient prepared to go home), A (patient sent to general hospital floor)

The classification task of this data set is to determine where patients in a post-operative recovery area should be sent to next (the discharge decision). For this assignment, the *COMFORT* attribute is not considered for the classification problem as it contains missing values. So, the classifier, in the form of a decision tree, which is produced will use 7 patient attributes to classify patient's into one of three discharge decisions.

All of the code that was used to employ the genetic algorithm can be found on github at `https://github.com/marcus-bornman/cos_710_assignment_2`. The repository also contains all information necessary to reproduce the results that are discussed within this assignment.

2 Representation

2.1 Nodes

The possible nodes for each individual, or decision tree, comprise decision nodes and end nodes. Decision nodes take a certain number of children and select one of the children based on the value of one of the patient's attributes. End nodes represent one of the three available discharge decisions.

There are 7 decision nodes - one for each available patient attribute. The number of children for each decision node is dependent on the number of possible values for the associated attribute. The Internal Temperature (IT) node, for instance, accepts 3 children - the first is selected when the patient's internal temperature is *low*, the second when it is *mid* and the third when it is *high*.

In addition, there are 3 end nodes - I, S and A. These end nodes, as well as the aforementioned decision nodes, are described in more detail in table 1.

Symbol	**Arity**	**Description**
IT	3	The internal temperature decision node - If the patient's internal temperature is low then select the first child - If the patient's internal temperature is mid then select the second child - If the patient's internal temperature is high then select the third child
ITS	3	The internal temperature stability decision node - If the patient's internal temperature is unstable then select the first child - If the patient's internal temperature is moderately stable then select the second child - If the patient's internal temperature is stable then select the third child
ST	3	The surface temperature decision node - If the patient's surface temperature is low then select the first child - If the patient's surface temperature is mid then select the second child - If the patient's surface temperature is high then select the third child
STS	3	The surface temperature stability decision node - If the patient's surface temperature is unstable then select the first child - If the patient's surface temperature is moderately stable then select the second child - If the patient's surface temperature is stable then select the third child
BP	3	The blood pressure decision node - If the patient's blood pressure is low then select the first child - If the patient's blood pressure is mid then select the second child - If the patient's blood pressure is high then select the third child
BPS	3	The blood pressure stability decision node - If the patient's blood pressure is unstable then select the first child - If the patient's blood pressure is moderately stable then select the second child - If the patient's blood pressure is stable then select the third child
OS	4	The oxygen saturation decision node - If the patient's oxygen saturation is poor then select the first child - If the patient's oxygen saturation is fair then select the second child - If the patient's oxygen saturation is good then select the third child - If the patient's oxygen saturation is excellent then select the fourth child
I	0	The end node for the patient to be sent to an intensive care unit
S	0	The end node for the patient to be sent home
A	0	The end node for the patient to be sent to the general hospital floor

Table 1: Decision Tree Nodes

2.2 Individuals

Each chromosome within a population consists of a single decision tree for determining the discharge decision for a patient. To clarify, internal nodes indicate the attributes of a patient while links represent each of the possible values for an attribute. Given the possible nodes identified in table 1, an example of one such decision tree is depicted in figure 1.

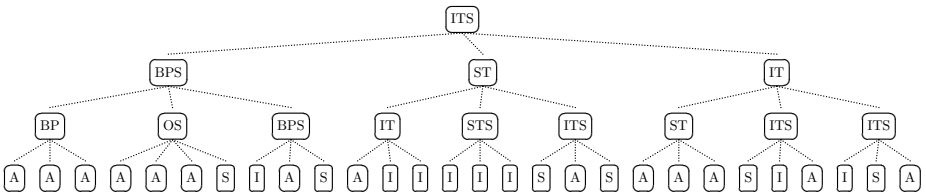

Figure 1: Decision Tree

2.3 Populations

Of course, a population consists of a collection of the individuals that have been described in section 2.2. Importantly, the genetic algorithm applied only made use of a single population consisting of 2048 individuals. This means that only a single population was evolved and cross-population breeding was not applied.

The initial population was generated using the ramped-half-and-half method proposed by Koza [3] to enhance population diversity of structure. The maximum tree depth allowed during the initial population generation was 4. So, in other words, the population is divided equally among individuals to be initialized with trees having depths 2, 3, and 4. For each depth group, half of the trees are initialized with the full technique and half with the grow technique.

In addition, to cover as much of the search space as possible an effort was made to reduce duplicates within the initial population. The strategy used to reduce the number of duplicates was as follows: if a duplicate individual is created as part of the initial population, it will be regenerated up to 100 times to try replace it with a new, original individual. After 100 attempts, if no unique individual was generated, the duplicate individual will be used.

3 Fitness Function

To determine the fitness of individuals, we use the number of patients for which the decision tree classified the discharge decision incorrectly. To be more specific, for a particular patient the predicted discharge decision is determined using the decision tree of the individual in question; then, if the predicted discharge decision is not the same as the recorded discharge decision the fitness is increased by 1. This process is repeated for all patients in the training set.

Of course, using this calculation for the fitness of an individual means an individual with a lower fitness value is deemed better than one with a higher fitness value. Algorithm 1 depicts the fitness function in its entirety.

Algorithm 1: Fitness Function

```
foreach patient in trainingData do
    recordedDecision = trainingData.dischargeDecision(patient);
    predictedDecision = individual.decisionTree.predict(patient);
    if predictedDecision != recordedDecision then
        numErrors++;
end
return numErrors;
```

4 Selection Method

To generate a new population, specific genetic operators are applied to members of an existing population with specific probabilities. These operators are explained in more detail in section 5. However, before these operators can be applied, the members of the existing population to which they need to be applied must be determined - hence, a selection method is necessary.

For this assignment, tournament selection [4] was used as the selection method for all genetic operators. This entails selecting T individuals entirely at random from the existing population. Then, from this set of individuals (ie. the *tournament*) the fittest individual is selected. After some initial test runs the size, T, of the tournament used was 48. On average, this tournament size produced the best balance between genetic diversity and convergence.

In the cases where genetic operators required more than one individual from the existing population, the tournament selection method was simply reapplied to select each individual.

5 Genetic Operators

As has been mentioned, genetic operators are applied to members of an existing population to generate a new population. For this assignment the operators used included reproduction, crossover and mutation. As detailed by Koza and Poli [5], these operators can be briefly summarized as follows:

- Reproduction: Copy the selected individual program to the new population as is.
- Crossover: Create offspring for the new population by combining randomly chosen parts from two selected programs.
- Mutation: Create one new offspring for the new population by randomly mutating a randomly chosen part of a selected individual.

Of course, these operators will not be used uniformly to generate new populations. Rather, every operator is associated with a probability which determines the likelihood that it will be used to select the next individual for the new population. In an attempt to determine the most effective probabilities for the operators, 10 tests were run using each of three different combinations of prob-

abilities. The best fitness achieved by each combination is summarized in table 2.

Crossover Probability	Reproduction Probability	Mutation Probability	Best Fitness
0.8	0.1	0.1	9
0.9	0.05	0.05	10
0.95	0.025	0.025	10

Table 2: Genetic Operator Combinations

Following the results of the tests summarized in table 2, the probability for crossover was selected as 0.8, the probability for mutation was selected as 0.1 and the probability for mutation was also selected as 0.1.

In addition, when applying operators decisions must be made regarding whether to select internal or leaf nodes. The probability for selecting a leaf node was chosen as 0.1 whereas the probability for selecting an internal node was chosen as 0.9.

Finally, a maximum tree depth of 16 was applied as a constraint to the individuals of a new population. In the case of crossover and mutation, the operators would try to produce a valid individual up to 10 times before proceeding to the application of the next operator.

With the selection methods and genetic operators having been identified, figure 2 depicts the entire breeding pipeline for generating a new population from an existing population.

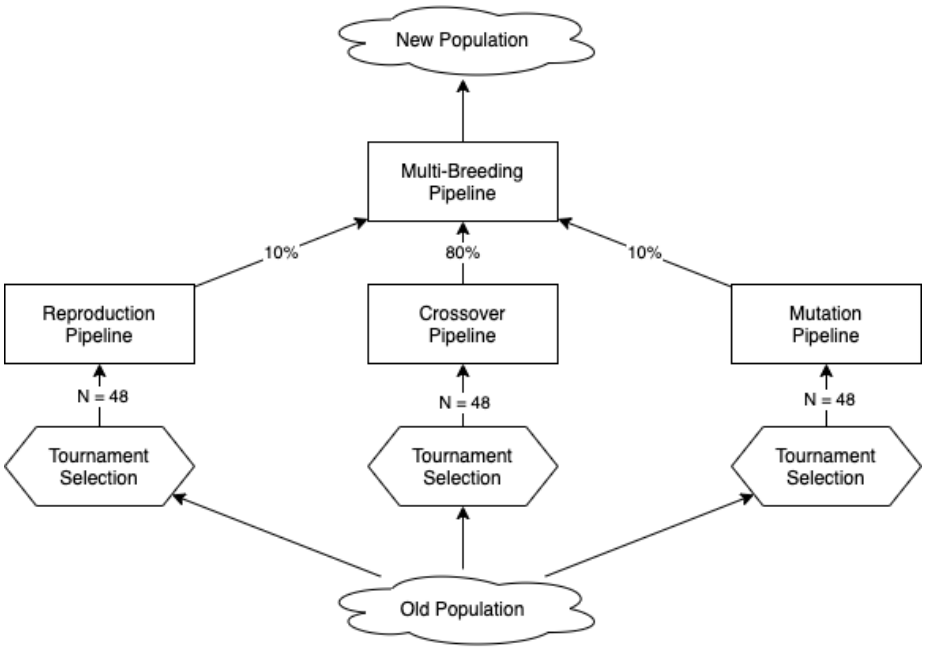

Figure 2: Breeding Pipeline

6 Termination Criteria

As is customary for most genetic programming approaches, certain termination criteria are necessary to ensure a finite amount of time before a solution is produced. For this assignment we include a maximum number of generations to be run as well as a problem-specific success predicate as the termination criteria.

The success predicate entails an individual predicting the discharge decisions for all patients in the training set correctly. The maximum number of generations allowed was chosen as 256. Notably, in the tests that were run the success predicate was never attained.

7 Results

Firstly, the machine used for development and testing purposes was a 2015 15-inch Macbook Pro with a 2.5 GHz Quad-Core Intel Core i7 processor, 16GB of memory and 500GB of storage. At the time of experimentation the machine had a battery cycle count of 684.

As has been mentioned, the data set used for training and testing can be found at `https://archive.ics.uci.edu/ml/datasets/Post-Operative+Patient`. Importantly, the data set was split into a training set of 70 patients, used for the execution of the genetic algorithm, and an evaluation set of 20 patients, kept for testing generalization.

To run tests a script was written which would run the algorithm on the training set with 10 different sets of seeds whilst also testing the best individuals against the evaluation set. The script is available in the github repository for this project (available at `https://github.com/marcus-bornman/cos_710_assignment_2`). The results for the 10 test runs can be seen in table 3.

Run	**Correct Training Classifications**	**Correct Evaluation Classifications**	**Correct Overall Classifications**
1	59	15	74
2	55	14	69
3	55	14	69
4	56	13	69
5	60	13	73
6	61	13	74
7	60	14	74
8	60	10	70
9	60	14	74
10	60	12	72

Table 3: Test Results

As deduced from the results in table 3, the best, average and standard deviation of the accuracy for training and accuracy for testing are depicted in table 4.

	Average Accuracy	Best Accuracy	Standard Deviation
Training Set	83.71%	87.14%	3.14%
Evaluation Set	66.00%	75.00%	6.63%
Overall	79.77%	82.22%	2.43%

Table 4: Classification Accuracy

8 Discussion

Whereas little other research has been devoted to applying genetic programming in the context of the post-operative patient data set, there have been other classifiers used to attempt to solve the classification problem.

Firstly, Owen [6] applied multiple nearest-neighbor approaches to the problem, with the best performant approach achieving an accuracy of 75.8% using 88 cases (2 cases were not used due to the presence of null values).

Secondly, Luukka [7] explored the use of an expert system to determine - based on hypothermia condition - whether patients in a post-operative recovery area should be sent to Intensive Care Unit, general hospital floor or go home. The reported results indicate that the system provides a mean classification accuracy of 62.7%.

Finally, a comparative study of different classification techniques for the post operative patient data set [8] explored various classification problems to the data set. The best of these approaches, referred to in the study as *FuzzyRoughNN*, achieved a classification accuracy of 88.88%.

When compared to the results discussed above, the classifiers produced by the genetic programming approach applied for this assignment perform relatively well. For instance, the fittest individual achieved a classification accuracy of 87.14% on the training data and an accuracy of 82.22% on the entire data set. Whilst results may seem less optimistic when considering that the best accuracy achieved on unseen data was 75.00%, indicators are still positive that - with additional parameter tuning - better results can be achieved.

While this initial experimentation has proved promising, additional research is still necessary to determine whether a genetic programming approach can produce a classifier which achieves better accuracy than the *FuzzyRoughNN* [8] algorithm. Such research may explore the application of a GP approach using a larger data set which may lend itself better to the evolutionary approach.

References

[1] S. Luke, L. Panait, G. Balan, S. Paus, Z. Skolicki, J. Bassett, R. Hubley, and A. Chircop, "Ecj: A java-based evolutionary computation research system," *Downloadable versions and documentation can be found at the following url: http://cs. gmu. edu/eclab/projects/ecj*, vol. 880, 2006.

[2] D. Dua and C. Graff, "UCI machine learning repository," 2017.

[3] J. R. Koza and J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*, vol. 1. MIT press, 1992.

[4] B. L. Miller, D. E. Goldberg, *et al.*, "Genetic algorithms, tournament selection, and the effects of noise," *Complex systems*, vol. 9, no. 3, pp. 193–212, 1995.

[5] J. R. Koza and R. Poli, "Genetic programming," in *Search methodologies*, pp. 127–164, Springer, 2005.

[6] A. B. Owen, "Tubular neighbors for regression and classification," 1999.

[7] P. Luukka, "Pca for fuzzy data and similarity classifier in building recognition system for post-operative patient data," *Expert systems with applications*, vol. 36, no. 2, pp. 1222–1228, 2009.

[8] S. R. Dash and S. Dehuri, "Comparative study of different classification techniques for post operative patient dataset," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 1, no. 5, pp. 1101–1108, 2013.