COSC 4P82 Genetic Programming: Assignment 1

Instructor: B. Ross
Due date: 9am Friday, February 16, 2024. **No lates accepted.**
Goal: Introduction to genetic programming.

Hand in: (see sections for exact requirements)

- Part A: A short 2-3 page summary report.
- Part B: A full report about the experiments.
- You should work in pairs on this assignment. One submission per group.
- Submit electronic copies in a ZIP file on Brightspace which includes data, GP source files that you edited, GP parameter files, and report.

System: You are free to use any GP system you want. ECJ and DEAP have been the most popular choices.

A. Symbolic regression

This question will introduce you to genetic programming. Many GP systems like ECJ and DEAP include symbolic regression as a tutorial example. You are to take an existing symbolic regression example (or create one if not available for your system), prepare it for the platform you are using, and execute it to get some results. Once you get it running, modify the training initialization setup (if necessary), by reading in the training points from a text file. This will let you run your system on new data sets, without having to recompile the GP system. (Note that this modification will be useful for part B).

Run the system 10 times (runs) per experiment (each run has a new random seed), on new data you have put into a new training file. You will investigate 4 parameter variations:

1. Crossover/Mutation tests: (i) 90% crossover, 10% mutation; (ii) 100% crossover; (iii) 100% mutation. Set Elitism to 2 for all of these.
2. Elitism test: Take the runs from (i) above, and compare it to a variant that uses no elitism. So this adds a new set of 10 runs that do not use elitism.

Make a 2-3 page summary report of your experiments. It should have the following format:

- Short description of problem explored (symbolic regression).
- Table of GP parameters, including user and defaults. Indicate the parameter variants.
- Short description of fitness evaluation: how data processed, fitness formula.
- Short description of the independent variable changed. That is the parameter you are experimenting with (probability crossover/mutation, elitism). The dependent variable is what you are measuring when you change the independent variable (e.g. fitness quality of solutions). Also indicate which experiment gave the best fitness performance.
- Two fitness plots averaged over the 10 runs of your experiment variants. One plot shows the crossover/mutation parameter runs in (1) above, and the other plot shows the elitism runs (2). The horizontal axis is "# Generations", while the vertical axis is "Fitness". You should plot the average population fitness per generation, and best fitness per generation, for each of the 2 variants. Use different colours for each plot line (and include a legend). The values plotted should be averaged over the 10 runs of each variant.
- A short conclusion that discusses the overall results of the comparisons. What is the most successful choice of parameters?

Note: Please do not show each run's performance plotted individually! Always compute the average of 10 runs, and plot that. Excel is your friend.

B. Rice Classification

The Rice (Cammeo and Osmancik) classification problem is from the UCI Machine Learning Repository:

https://archive.ics.uci.edu/dataset/545/rice+cammeo+and+osmancik

You are to use GP to evolve a rule that predicts which one of 2 classes of rice is being analyzed. The image data has been pre-processed, and the values in the data set are summary measurements of these images, using various image processing algorithms. The data set has 3810 records (rice images). Each record has the following numeric attributes (see above URL for descriptions):

1. Area
2. Perimeter
3. Major_Axis_Length
4. Minor_Axis_Length
5. Eccentricity
6. Convex_Area
7. Extent
8. *Class: (binary) This is the class label for the record.

Here are 2 examples from the database:

15231,525.5789794921875,229.7498779296875,85.09378814697266,0.9288820028305054,15617,0.5728955268859863,Cammeo

12318,442.5870056152344,189.56443786621094,83.20392608642578,0.8985253572463989,12495,0.6517460346221924,Osmancik

The first 7 attributes will be used in the GP tree to predict attribute 8. Note that there is a mix of integer and floating point values in the data. You can assume all will be floating point in the GP tree (make sure that integers are properly converted to floats when read).

Attribute 8 is the classification of the record. You want GP to predict this value. It will be used for fitness evaluation, and performance evaluation. Of course, **do not give it to the GP tree** -- it will make its job too easy! In other words, it should not be supplied as a GP language terminal.

Here is a recommended approach to follow.

1. Read the database into a large table (2D array), one row per example. You should randomly shuffle the rows. Then split the table into 2 independent sets of examples – a training set, and a testing set. The exact size of these sets is up to you, and may have to be experimentally determined. Be aware that machine learning is usually affected by the size of training sets. For example, sometimes smaller training sets are more effective, so long as they are representative of the entire data set.

2. Define a set of language primitives to be used by GP. These primitives should work sensibly on the input data. A possible set of primitives to consider are:
    a. functions: arithmetic operators (see part A), min and max, if-then-else,...
    b. terminals: Data attribute variables (7), and ephemeral random constants.

    A recommended GP language is to treat the GP expression as a mathematical expression that processes numeric values. It processes one record from the training data, in order to give a binary classification for the image represented by that record. The root of the tree will produce a float. If the float value is greater than or equal to zero, then the GP answer is interpreted to be **Cammeo**. Otherwise, when negative, it is **Osmancik**

3. The top-level wrapper that executes a GP classifier tree as follows. The GP tree processes one record at a time. The 7 attribute values, from a row in the table, are copied into a set of 7 variables accessible by the GP expression, and which will take the form of leaf terminals within the tree. The program is executed by the GP system interpreter using these values. The resulting answer is recorded. The fitness function will then compare this answer to the known solution label from the example table, and the fitness score will be appropriately updated. The following pseudo-code shows the general execution strategy:

```
float area, perimeter, ..., extent;
float GP_ans;

loop for i=0 to N-1 :  // N training examples from table
     area = training[i][1];
     perimeter = training[i][2];
     (... ending at training[i][7])
     GP_ans = execute_tree(t); // t is the current GP tree
// now the predicted quality found in ans can be compared to
// real recorded quality of that example,
// perhaps stored in integer array ans[i] (i.e. training[i][0]).
```

4. TRAINING: Fitness is based upon how well the GP tree classifications match the “correct” classifications in the training data. When they agree, we call it a "hit". The higher the hits, the better. The simplest fitness formula is:

```
If (GP_ans >= 0.0  and ans[i] = "Cammeo")
   then hits = hits + 1;
Else if (GP_ans  < 0.0 and ans[i] = "Osmancik")
   then hits = hits + 1;
```

Thus, if the number of computed hits matches the number training examples, then the GP has perfectly predicted all the training cases.

5. TESTING: At the end of the run, you must test the **best** (most fit) GP solution by running it on the testing set (all the examples not used for training). This gives an idea of the generality of your evolved solution when given new data it has never seen. Record the test performance of each run. This information will be included in the report.

Do 10 runs, each with different random number seeds and shuffled data sets. Collate the performance of your runs together (Excel is your friend). Your report should have a table summarizing all the runs, for both the training scores and testing scores. You should report the average best solution scores, as well as the scores for the overall best solution from all runs. Include a confusion matrix with each experiment. Also, as in part A, include performance graphs showing the population fitness and best fitness plotted over generations (averaged over for all the runs).

You should do at least **one comparative experiment**. You are free to choose any parameter (or other) variation that you wish to compare. There should therefore be 2 sets of runs (1 per comparison set), with each set having 10 runs with unique random seeds.

Your report should describe all relevant aspects of your experiment. The goal of the report is to give enough information so that someone else could reproduce your experiments. Describe your problem set (and its source), and describe what your GP program is attempting to do. Be sure to describe your GP language, all GP parameters, and your fitness evaluation methodology. Include the tree code of evolved GP programs for your 2 best rules obtained. If they are large, they can be put into an appendix of your report. Include a discussion on your results. Tables and graphs do not speak for

themselves, and so you must describe and discuss the important results that they are showing. Also include a conclusion section that summarizes the strengths or weaknesses of your experiment, and what could be done in the future to improve the results (aka "future work").

The report format should be:

- Introduction: problem description
- Experiment details:
  - parameters for GP
  - fitness function
  - GP language
  - format of data
  - variations of experiments
- Results
  - tables of results (best, avg, etc.), confusion matrices
  - performance plots
  - A few examples of the best GP expressions evolved (these may need to be in an appendix!)
  - Analytical discussion of results
- Conclusions
- Bibliography (data set URL, references,...)

Comments:

- Please see the ECJ FAQ file on the 4P82 home page. It has very useful tips on how to use ECJ. Also see the ECJ Owners Manual. (Link below).
- Run your experiments multiple times with different random number seeds. However, please do not include the actual seed values within your report, nor the scores obtained in each separate run. This is considered TMI ("too much information"). You should summarize the 10 runs using useful statistics, such as average and (maybe) standard deviation (can be useful for statistical significance such as T-test, but not required in this assignment).
- Feel free to experiment with different GP parameter settings for the experiments. But keep the parameters consistent throughout all runs, except the independent variables being tested.

Useful web sites:

- Rice home page at ICU ML database: https://archive.ics.uci.edu/dataset/545/rice+cammeo+and+osmancik
- ECJ: https://cs.gmu.edu/~eclab/projects/ecj/
- ECJ Owners Manual: https://cs.gmu.edu/~eclab/projects/ecj/manual.pdf
- DEAP: https://github.com/DEAP/deap
- Confusion matrix: https://en.wikipedia.org/wiki/Confusion_matrix
- Also see the course web site for a few examples of student assignment reports.