COS 710 Assignment 1 Report

Anonymous

March 21, 2025

1 How to Run GP Algorithm Source Code

1. Import the .ipynb file onto a Jupyter environment i.e. Google Colab.
2. Run all the cells.
    - For Google Colab click "Runtime" > "Run all"
    - For Jupyter Lab click "Kernel" > "Restart Kernel and Run All Cells"

2 Dataset and Data Pre-processing

The **192_vineyard** dataset was used. This dataset is a time series of 2 independent variables and 1 dependent variable. The first independent variable represents the quantity of grapes harvested in lugs for the year 1989 and the second variable represents the quantity for the year 1990. More information about this dataset can be found here. The dependent variable, which is the target for the regression task is the number of lugs produced in the year 1991.

The dataset underwent minimal pre-processing. The column headings were renamed: $lugs\_1989$ to x_1, $lugs\_1990$ to x_2 and $target$ to y. Before training, the dataset was shuffled to improve generalization and reduce bias. The dataset was split into train and test sets. The train set was used to evolve the regressors and the test set was used to evaluate the best regressors for each run to assess performance and ensure that the regressors did not overfit the data.

3 Description of the GP algorithm

The GP algorithm used is described in the following subsections.

3.1 Regressor Representation

The regressors were represented as hierarchical parse trees. Terminals were represented as leaf nodes and functions as internal nodes.

3.2 Primitives

Primitives are the subcomponents that make up the programs to be evolved by a GP algorithm. Primitives are divided into two sets: the terminal set and the function set. The terminal set comprises variables which are the inputs

to the problem, constants and operators that do not take any arguments. The terminal set has 3 elements:

1. x_1: representing the number of lugs for the year 1989.
2. x_2: representing the number of lugs for the year 1990.
3. Random ephemeral constant $\Re$ drawn randomly from a continuous uniform distribution of values in the range [1, 6).

The function set comprises primitives that operate on other primitives. Members of this set are functions which take arguments. The function set for the GP algorithm contains 4 elements each of arity (number of arguments) 2:

1. $+$: addition operator
2. $-$: subtraction operator
3. $*$: multiplication operator
4. Protected division operator, which returns 1 as the result if the divisor is a zero.

3.3 Fitness Function

The fitness function used was **Mean Absolute Error** calculated by the equation:

$$MeanAbsoluteError = \frac{\sum_{j=1}^{N} |s(i,j) - C(j)|}{N} \tag{1}$$

where:

- N is the number of fitness cases
- $s(i,j)$ is the result return by program i for the fitness case j
- $C(j)$ is the correct value for fitness case j

3.4 Selection Method

The selection method used was tournament selection. In tournament selection, given tournament size m, m individuals are randomly drawn from the population with replacement. The fittest individual in the drawn tournament sample is the one selected. A tournament size of 2 was used in this work.

3.5 Genetic Operators

Genetic operators refer to mechanisms that direct the search for the solution in the search space. They direct the search by creating the population for the next generation. Genetic operators used in this work were:

- Crossover: This operator is analogous to genetic exchange between chromosomes. In GP, this operator swaps parts (subtree, node or leaf) of 2 parents (2 selected individuals).
- Mutation: in mutation, a part (subtree, node or leaf) of the program(individual) is replaced by a randomly generated part (subtree, node or leaf).

3.6 Termination Criteria

Termination criteria refer to the conditions set to ensure that a GP algorithm terminates. Termination criteria used were:

1. Fitness threshold: a minimum Mean Absolute Error threshold was set at 0.01 such that the GP algorithm would terminate once an individual's Mean Absolute Error (fitness) is equal to or below 0.01
2. Number of generations: the maximum number of generations per run was set at 50. The GP algorithm would terminate once 50 generations are concluded if no individual was found that was meeting the fitness threshold.

4 Parameters Tuning for GP Algorithm

The parameters for the GP algorithm are outlined in Table 1. All the parameters were hand-tuned, that is, by observing the behaviour and performance of the GP algorithm and its outputs (evolved programs), the values used were adjusted by trial and error.

Parameter	Value	Intuition
Population Size	100	Generate a sufficiently diverse initial population. The grow method was used. No duplicates were allowed.
Max Depth	7	Ensure the trees are sufficiently complex, not more not less.
Tournament Size	2	A low tournament size to reduce selection pressure promoting slower but surer convergence
Operator Application Rate	crossover = 80%, mutation = 20%	A balance between exploitation and exploration. Higher mutation rate lead to jerky solution searches.
Generations	50	Higher numbers had little to no effect on performance
Fitness Threshold	0.01	Adapted from John Koza's experiments
Random Ephemeral Constant domain	[1,6)	Exclude negative values for more stability in convergence
Train and Test Sets Split	train = 70%, test = 30%	Train set has a majority to allow the algorithm to learn as much as possible.

Table 1: Listing of the parameter for the GP algorithm, against their hand-tuned value and tuning intuition.

5 Experimental Results

The GP algorithm was run 10 times. Table 2 below presents the best program for each run against the mean and standard deviation of the objective values it generates on the test set fitness cases. For a neater presentation, figures

are truncated to 4 decimal places.

Run	**Best Program**	**Fitness(test set)**	**Mean**	**Standard Deviation**
0	$x_1 + x_2 + 4.5150 + \frac{1}{3.5295}$	2.4002	17.0796	4.3300
1	$x1 + x2 + 5.0484$	2.2308	17.1734	4.7087
2	$x1 + x2 + 4.3027$	2.4621	16.5840	4.3300
3	$x1 + x2 + 5.0021$	2.3755	17.2833	4.3300
4	$x1 + x2 + 4.6912$	2.4135	16.9724	4.3300
5	$x1 + x2 + 5.1272$	2.4068	17.4084	4.3300
6	$x1 + x2 + 5.0052$	2.3763	17.2865	4.3300
7	$x1 + x2 + 4.8001$	2.3999	17.0815	4.3300
8	$x1 + \frac{x2}{2.5853} + 11.4238$	2.3320	18.0138	2.8181
9	$x1 + x2 + 5.0151$	2.3787	17.2964	4.330

Table 2: Listing best program per run against test set fitness, mean and standard deviation of objective values evaluated on test set fitness cases.

5.1 Discussion

From Table 2, it is evident that for the majority of the runs, the GP algorithm converges to the same area of the search space given the resemblance of the regressor equations. Therefore, it is not surprising that the majority of the programs in the table have nearly similar objective value descriptive statistics. Additionally, the regressors are all linear equations.