Alexander Muendesi

Description of Dataset

- The dataset contains information about trips made using Seoul Bike throughout the year 2018.
- It includes the following fields: rental bike number, pickup date and time, pickup station number and address, dropoff date and time, dropoff station number and address, return dock, trip duration in minutes and total distance in metres.
- The dataset also includes latitude and longitude details of the rental bike stations.
- Weather information is also added to the dataset to increase the performance of the prediction models.

Data preprocessing and feature extraction

- No additional data preprocessing or feature extraction was performed on the original data set.
- However a subset of the data set was extracted such that the program execution time was reasonable. A total of 150000 elements were extracted from the dataset.
- 70% of the data set is used as the fitness cases and the remaining 30% of the data is used as the test set.

Description of Representation

- An individual is represented by a linear string of genes. Each individual is composed of a variable number of codons with each codon having 8 bits.
- Each codon is then mapped to a production rule in the grammar.
- The production rules are used to generate a derivation tree which represents the actual program.

Description of Grammar used

- The following BNF grammar is used:
  <start> -> <expr> <op> <function> | <terminal>
  <expr> -> <expr> <op> <function> | <terminal>
  <op> -> + | - | * | /
  <function> -> $\sqrt{< terminal >}$ | sin(<terminal>) | cos(<terminal>) | log(<terminal>)
  <terminal> -> t1 | t2 | . . . | t_n

  Where t represents an input element from a column in the dataset and n is the number of columns in the dataset used
- This grammar allows the formation of expressions using elements from the dataset.
- The expressions can involve arithmetic operators(+,-*,/) or functions(log, sin,cos, square root), with terminals representing specific values from the dataset.
- The <start> symbol represents the beginning of an expression whilst <expr> represents an expression that can be recursively expanded.

Description of initial population generation

- The grow method was used for initial population generation.
- A maximum and minimum number of codons in a chromosome is defined first. Then a chromosome is created with a number of codons randomly in between that range.
- Each codon within a chromosome is then randomly initialised to a random 8 bit string.
- This process is repeated until the population size is met.

Description of fitness function and fitness evaluation

- The raw fitness is used as the fitness function.
- Raw fitness is defined as the sum of the absolute difference between the predicted values and the actual values for each data item.
- Each individual chromosome is mapped to a program using the grammar which is then evaluated to give a result. This result is subtracted from the actual value and the absolute value of the result is used as part of the raw fitness measure.

Description of function and terminal sets

- The terminal set consists of all the column elements in the dataset except the “trip duration” column.
- The function set consists of the following set {+,-,*,/,log,sin,cos,√} with arity {2,2,2,2,1,1,1,1}.
- A protected divide is implemented such that when division by 0 occurs 1 is always returned.
- A protected square root is implemented such that only the absolute value of its argument is used when performing the square root calculation.
- A protected log function is used such that when the argument is 0 it always returns 1, when the argument is negative we use the absolute value of the argument.

Description of selection method

- Tournament selection is used to select individuals to apply genetic operators to.
- A tournament size is first specified as a parameter. Then “tournament size” individuals are selected at random. The fitness of each of the individuals is calculated and the winner of the tournament is the individual with the best fitness and is returned.

Description of control model

- A generational control model is used.
- A population of fixed size is maintained. When it is time to generate a new population, tournament selection is used to select individuals on which genetic operators are applied. This is done until the new population is completely created and the old population is completely replaced.
- This process continues until the maximum number of generations is reached or a solution is found.

Description of genetic operators

- Two genetic operators were used in this implementation, crossover and mutation.
- For mutation, one bit from a randomly selected codon is randomly selected and flipped
- With crossover, 2 point crossover was implemented to promote greater exploration when crossover is performed. 2 point crossover involves using 2 crossover points. All the bits between the two crossover points are replaced by bits from the second parent whilst all the other bits in the individual are kept as they are.
- Should the number of codons in a chromosome exceed the maximum number of codons allowed, the excess is pruned off.

Description of experimental setup

- The following parameters were used:
  1. Population Size: 100
  2. Max number of generations: 50

3. Mutation Rate: 0.1
4. Crossover Rate: 0.9
5. Tournament size: 4
6. Max number of codons in chromosome: 50
7. Minimum number of codons in chromosome: 5

- Specifications of machine were experiments were conducted:
  1. RAM: 8GB
  2. Processor: AMD Ryzen 3 3250U with Radeon Graphics
  3. Number of cores: 2

Results

Train Data

Mean Absolute Error

Seed value										Avg	Best
0	5	6	9	15	18	22	23	42	46		
11.836	12.499	13.596	12.404	12.526	12.517	9.896	13.557	12.478	12.477	12.378	9.896

RSquared

Seed value										Avg	Best
0	5	6	9	15	18	22	23	42	46		
-0.039	-0.133	-0.109	-0.110	-0.133	-0.143	-0.018	-0.109	-0.128	-0.141	-0.106	-0.018

RMSE

Seed value										Avg	Best
0	5	6	9	15	18	22	23	42	46		
20.639	21.549	21.321	21.334	21.551	21.646	20.421	21.315	21.498	21.624	21.289	20.421

Mean Absolute Deviation

Seed value										Avg	Best
0	5	6	9	15	18	22	23	42	46		
6.293	6.271	9.0	6.272	6.272	6.270	3.02	8.753	6.397	5.996	6.454	3.02

Test Data

Mean Absolute Error

Seed value										Avg	Best
0	5	6	9	15	18	22	23	42	46		
15.086	17.663	21.290	20.006	18.255	19.654	14.607	16.502	16.314	21.678	18.106	14.607

RSquared

Seed value										Avg	Best
0	5	6	9	15	18	22	23	42	46		
0.740	0.677	0.610	0.668	0.646	0.659	0.745	0.723	0.706	0.603	0.677	0.745

RMSE

Seed value										Avg	Best
0	5	6	9	15	18	22	23	42	46		
24.246	27.014	29.707	27.403	28.288	27.785	24.000	25.031	25.786	29.981	26.924	24.000

Mean Absolute Deviation

Seed value										Avg	Best
0	5	6	9	15	18	22	23	42	46		
7.752	10.252	14.0	13.321	10.0	13.0	7.769	10.660	8.426	15.0	11.018	7.769

Runtimes of implementation

- Assignment 1 runtime: 30 minutes
- Assignment 2 runtime: 40 minutes
- Assignment 3 runtime: 18 minutes
- It can be seen that the GE implementation runs the fastest and this is to be expected as there is significantly less computational work to perform. Recursion is used to both map chromosomes to programs and evaluate those programs at the same time. Hence time is saved from not building trees etc.
- Additionally the GE implementation executes for a fewer number of generations when compared to the 2 other implementations from the previous assignments.

Comparison with assignment 1 results

Assignment 1 results

Mean Absolute Error		RMSE		Mean Absolute Deviation		RSquared	
Avg	Best	Avg	Best	Avg	Best	Avg	Best
8.25	6.96	17.04	15.95	3.31	1.91	0.312	0.40

- In terms of predicting accurately we see that the GE implementation significantly outperformed the Assignment 1 GP. This is shown by the RSquared value which is double that of the best and average values of that of the assignment 1 GP.
- In terms of how close the predicted values are together we see that the GE implementation is significantly worse than that of the assignment 1 implementation. The GE's implementation results are very spaced out. This is shown from the very high MedAE values.

- Considering the MAE, we see that the GE implementation also does significantly worse than the assignment 1 implementation. It produces very large errors when it predicts wrong.
- Comparing the RMSE values, we see that the GE implementation produces more errors due to the higher value when compared to assignment 1 implementation.
- From the results we can conclude that the GE implementation is effective at capturing the relationships between inputs and outputs(the trip duration) of the majority of the test data as shown by the high RSquared value.
- However the GE implementation is not particularly good at predicting individual elements or reducing errors on average as shown by the high values in MAE, Median Absolute deviation and RMSE.

Comparison with assignment 2 results

Assignment 2 results

RMSE

Seed value										Avg	Best
10	2	12	0	1	13	19	20	8	22		
16.244	16.276	16.394	16.408	16.518	16.570	16.646	18.368	19.947	20.369	17.374	16.244

RSquared

Seed Value										Avg	Best
0	1	2	8	10	12	13	19	20	22		
0.366	0.358	0.377	0.379	0.367	0.380	0.368	0.378	0.378	0.375	0.373	0.380

Mean Absolute Deviation

Seed Value										Avg	Best
0	1	2	8	10	12	13	19	20	22		
2.141	2.169	7.326	6.596	2.064	2.453	2.303	2.182	5.189	5.171	3.759	2.064

Mean Absolute Error

Seed Value										Avg	Best
0	1	2	8	10	12	13	19	20	22		
7.150	7.199	10.045	11.545	6.971	7.545	7.286	7.409	9.611	11.096	8.586	6.971

- Comparing the RSquared values, we see that the GE implementation does a significantly better job at predicting values than the structure based GP approach.

- However, comparing the other performance measures, the GE implementation performs very badly compared to the structure based GP approach. This shows that the GE implementation has a high error rate as seen from the high values in RMSE and MAE. Also the values produced by the GE implementation tend to be very far apart as shown by the high Mean Absolute Deviation value.
- Overall we see that the GE implementation does not do well in minimising the errors produced on average but it is very good at capturing the relationship between the input data and the output data(trip duration).

Comparison with results in paper

- Comparing the RSquared values, we see that for the training data the GE implementation did worse than all the 4 models in the paper. However for the test data, the GE implementation managed to do better than 3 of the 4 models in the paper.
- This shows that the GE implementation is good at mapping the relationship between inputs and output(trip duration) for the test data but not so much for the training data.
- Comparing the MAE values, we see that the GE implementation only managed to do better than the LR model for the training data but performed worse than all the other 4 models when using the test data. This shows that the GE implementation tends to produce larger errors than the models in the paper.
- Comparing the RMSE values, we see that the GE implementation performed worse than all the other 4 models. The GE implementation has a higher error rate than the other models.
- For the MedAE, we see that the GE implementation was able to perform better than the LR and GBM models only for the training data. However for the test data, the GE implementation performed worse than all the other 4 models. This shows that the GE implementation produces results that are spread out over a wider range compared to the other models.
- Overall we can conclude that the GE implementation performs very well for capturing the relationship between inputs and outputs of the majority of the data set but does poorly when it comes to minimising the errors produced.