IE541 Symbolic Regression

Jaswanth Mucherla and Saichandar Reddy Naini

November 30, 2021

1 Question 1

We explore Genetic Programming using Python's DEAP library and vary various parameters over the data sets provided to observe the algorithm's behaviour.

We built a Genetic Programming algorithm to fit a regression line to data sets provided. Our GP used One Point crossover, Tournament selection with size 3, and a mutation function which randomly replaces a part of the original tree with a tree of height ranging from 0 to 3. We have experimented with the following function sets (denoted as pset in the code).

	Function set	Description
1	pset1	[x, add, sub, mul, protectedDiv, neg, EphemeralConstant]
2	pset2	[x, mul, EphemeralConstant]
3	pset3	[x, protectedDiv, EphemeralConstant]

Function sets used for experimentation

To evaluate the fitness of each program generated by the GP we used sum of square errors, given below.

$$\frac{(\sum_{i=1}^{n} \hat{y_i} - y_i)^2}{n}$$

$\hat{y_i}$ is the output for ith point provided by the program.
y_i is the target value provided in the data set.

The following table provides justification for the parameters used in the analysis.

	Parameters	Justification
1	NGEN	We chose 100 generations for our analysis as a good compromise between computational costs and effectiveness of results
2	POP	Population is varied from 300 to 500 in the steps of 100
3	CXPB	Crossover probability is varied from 0.4 to 0.8 in the steps of 0.2
4	MUTPB	Mutation probability is varied from 0.1 to 0.5 in the steps of 0.2
5	Crossover Type	We experimented with different crossover types. Onepoint crossover gave the best results for lower computational times
6	Mutation Type	We opted for uniform mutation with a max tree size of 3
7	Selection Type	During initial experimentation Tournament selection gave better results compared to picking the best for every generation

Justification for the parameters used

1.1 Part (a)

We perform Genetic Programming on data set 1.

1.1.1 Results/ insights

We performed statistical analysis by varying population $\in [300, 400, 500]$, crossover probability $\in [0.4, 0.6, 0.8]$, mutation probability $\in [0.1, 0.3, 0.5]$ over three different function sets. Each combination of parameters are run for 100 generations and for 30 runs. The results from the analysis are tabulated below.

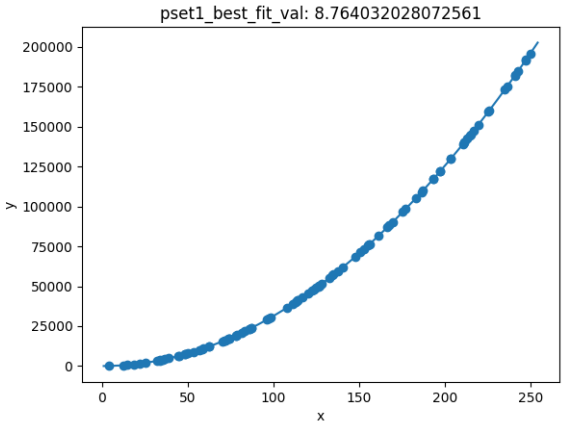
	Population(N)	300	400	500
	Function sets	Best fitness values		
1	pset1	8.764032028072561	845.4550826717076	46.60023593866738
2	pset2	5471.327372221466	197410.13628684886	4761726.262703939
3	pset3	0.1457319880741799	1.3076923252394317	3.3996327959840653

Best fitness values

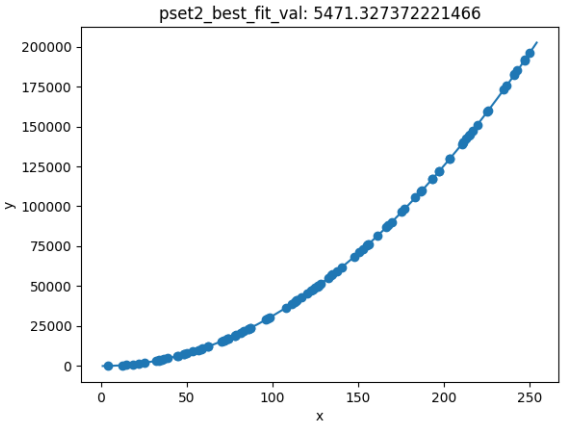
	Function sets	best fitness	equation
1	pset1	8.76	$(-5921.245 * (x \div (-1884.34)) * (x + (x \div (-4864.65)))$
2	pset2	5471.33	$(3.14 * (x * x))$
3	pset3	0.145	$(x \div (((-2196.47) \div (-6900.44)) \div x))$

Best solutions

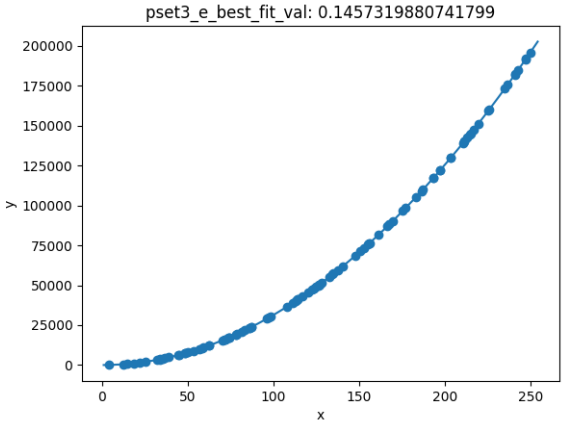
We observe the best result for function set 3 with only ProtectedDiv operator and, interestingly, for a lower population size of 300.



(a) Best fit figure for function set 1



(b) Best fit figure for function set 2



(c) Best fit figure for function set 3

1.1.2 Best final result

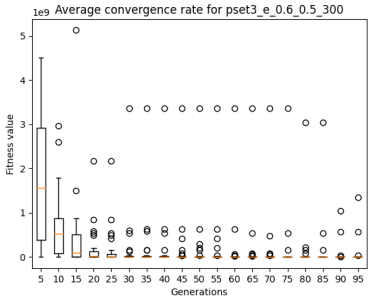
Best final result for data set 1 is obtained with function set pset3 with a fitness value of 0.1457319880741799. The equation for the solution is as follows:

$$y = 3.141606460961125 * x^2$$

The equation obtained is close to $y = \pi * x^2$

1.1.3 Convergence rate

The following figure shows average convergence over 30 runs and for 100 generation. The plot shows a good average convergence rate, however the GP randomly jumps out of the best solutions for individuals runs as shown by outliers.



Convergence plot pset3, cxpb = 0.6, mupb = 0.5, pop = 300

1.2 Part (b)

We perform Genetic Programming on data set 2.

1.2.1 Results/ insights

Similar to part a we have experimented by varying population $\in [300, 400, 500]$, crossover probability $\in [0.4, 0.6, 0.8]$, mutation probability $\in [0.1, 0.3, 0.5]$ over three different function sets. Each combination was run for 100 generations and 30 runs. The best fitness values for each function set are described in the table below.

	Population(N)	300	400	500
	Function sets	Best fitness values		
1	pset1	7.63e-07	4.565e-06	3.1e-06
2	pset2	0.011	0.0071	0.0074
3	pset3	2.76e-06	2.6e-07	2.53e-08

Best fitness values

	Function sets	best fitness	equation
1	pset1	7.63e-07	$((x_1 - 2602.53) \div (2602.53 - (x_1 + x_3))$
2	pset2	0.0071	$(((-0.045) * (-0.045)) * ((-0.045) * (-0.045)))$
3	pset3	2.53e-08	$(((1733.55 \div (x_3 \div x_2))) \div ((x_3 \div x_2) \div (x_1 \div x_2)))$

Best solutions

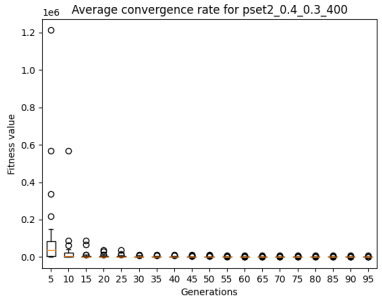
1.2.2 Best final result

We observe from the table the best possible result occurs at pset3 and 500 population. The solution for the corresponding function set and population is as follows.

$$y = 9.8155 * \frac{x_1 * x_2}{x_3^2}$$

1.2.3 Convergence rate

Average fitness values for the 30 runs and various iterations is plotted below. The problem has good convergence rate without outliers at individual runs.



Convergence plot pset2, cxpb = 0.4, mupb = 0.3, pop = 400