Nicholas Tran, 300296259

COMP307 - Assignment 2:

Neural and Evolutionary Learning

Part 1: Classifying Pingu with a Neural Network

Initial Weights:

w15	w16	w25	w26	w35	w36	w45	w46	w57	w58	w59	w67	w68	w69
-0.28	-0.22	0.08	0.2	-0.30	0.32	0.1	0.01	-0.29	0.03	0.21	0.08	0.13	-0.36

Output and Prediction after the first feed-forward using the initial weights:

First instance has label Adelie, which is [0] as an integer, and [1. 0. 0.] as a list of outputs.

Predicted label for the first instance is: ['Chinstrap']

Updated weights after applying a single back-propagation to the first instance:

w15	w16	w25	w26	w35	w36	w45	w46	w57	w58	w59	w67	w68	w69
-0.28	-0.22	0.08	0.2	-0.30	0.32	0.1	0.01	-0.28	0.02	0.20	0.09	0.12	-0.37

Final weights and test accuracy after 100 epochs:

w15	w16	w25	w26	w35	w36	w45	w46	w57	w58	w59	w67	w68	w69
0.93	-9.81	-7.29	5.2	2.39	-1.41	2.47	1.43	-9.68	-2.44	3.24	4.91	-2.87	-11.65

test accuracy: 53/65 = 81.54%

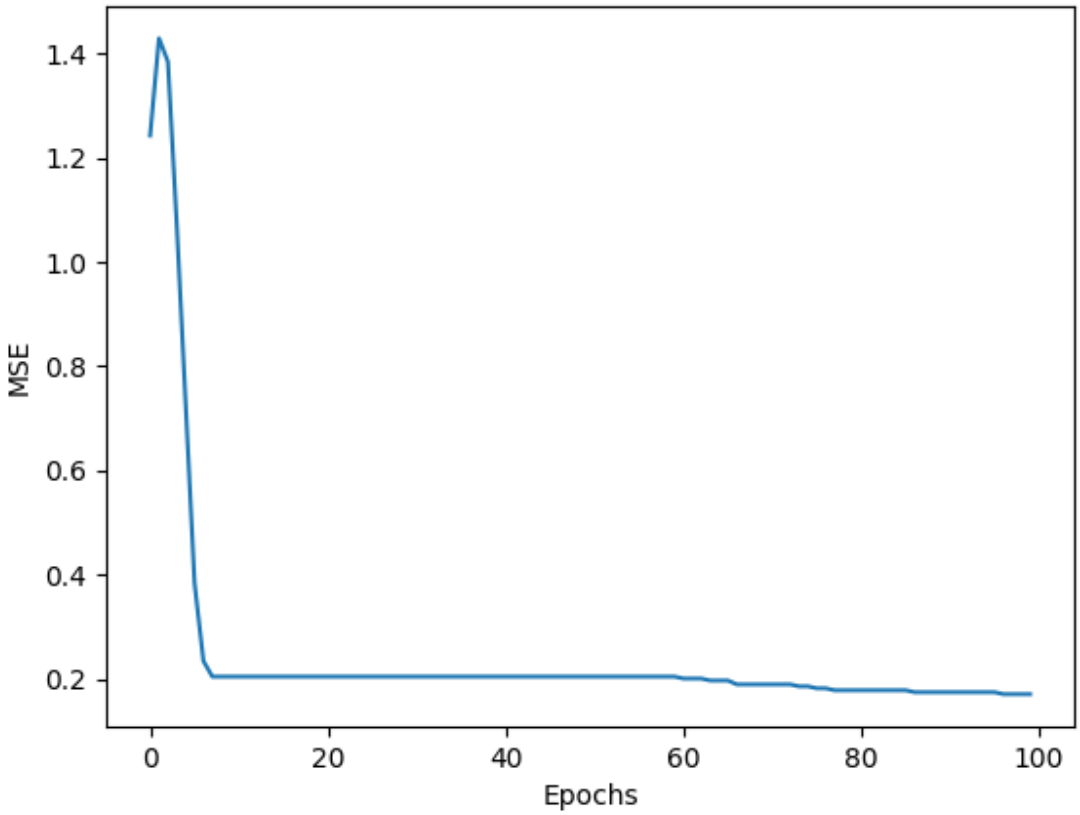
The simple network model without biases correctly identifies 53 instances whilst fails to correctly identify 12 instances. From the first back propagation of the first instance the weights don't change by much but after 100x268 (epochs times training-instances) iterations the weights have changed dramatically and are better at capturing the variance of the data.

Nicholas Tran, 300296259

Network Performance, Convergence and Overfitting:

The network performed as expected especially considering this is just the simple network without any bias nodes. Performance can be greatly improved by introducing bias nodes and/or adjusting the learning rate.

The learning rate is responsible for the convergence of a network, whether or not the network converges to a global minima. A high learning rate often means the network never converges to a global minima due to the network overshooting it. In contrast a small learning rate can help the network converge to a global minima but the lower the learning rate the longer the algorithm takes to converge and also makes it more susceptible to getting stuck in local minima. Convergence happens when the performance of the algorithm doesn't significantly change between epochs. The model converged quite quickly, around the 5th epoch or so, as seen in the below graph:



From the 5th epoch onwards the accuracy only increases marginally. With the low number of epochs, the algorithm doesn't actually fully converge as it continues to converge up to the final epoch. The model continues to improve ever so slightly and if the number of epochs were to

increase, the accuracy of the training model will also continue to increase with the trade-off being an increased risk of overfitting.

The performance of training the network is very similar to the performance of the network on the test data (roughly only a 1% difference in accuracy) so there isn't any evidence of overfitting. The low stopping criteria of 100 epochs means that the algorithm doesn't overlearn.

Further Improving the Network with Bias Nodes:

Bias nodes are one way of dramatically increasing the performance of a neural network. This is due to its ability to swing the weights in one direction or the other.

Initial biases:

b_5	b_6	b_7	b_8	b_9
-0.02	-0.20	-0.33	0.26	0.06

Updated biases after applying a single back-propagation to the first instance:

b_5	b_6	b_7	b_8	b_9
-0.02	-0.20	-0.30	0.23	0.03

Final weights and accuracy without biases:

w15	w16	w25	w26	w35	w36	w45	w46	w57	w58	w59	w67	w68	w69
0.93	-9.81	-7.29	5.2	2.39	-1.41	2.47	1.43	-9.68	-2.44	3.24	4.91	-2.87	-11.65

test accuracy: 53/65 = 81.54%

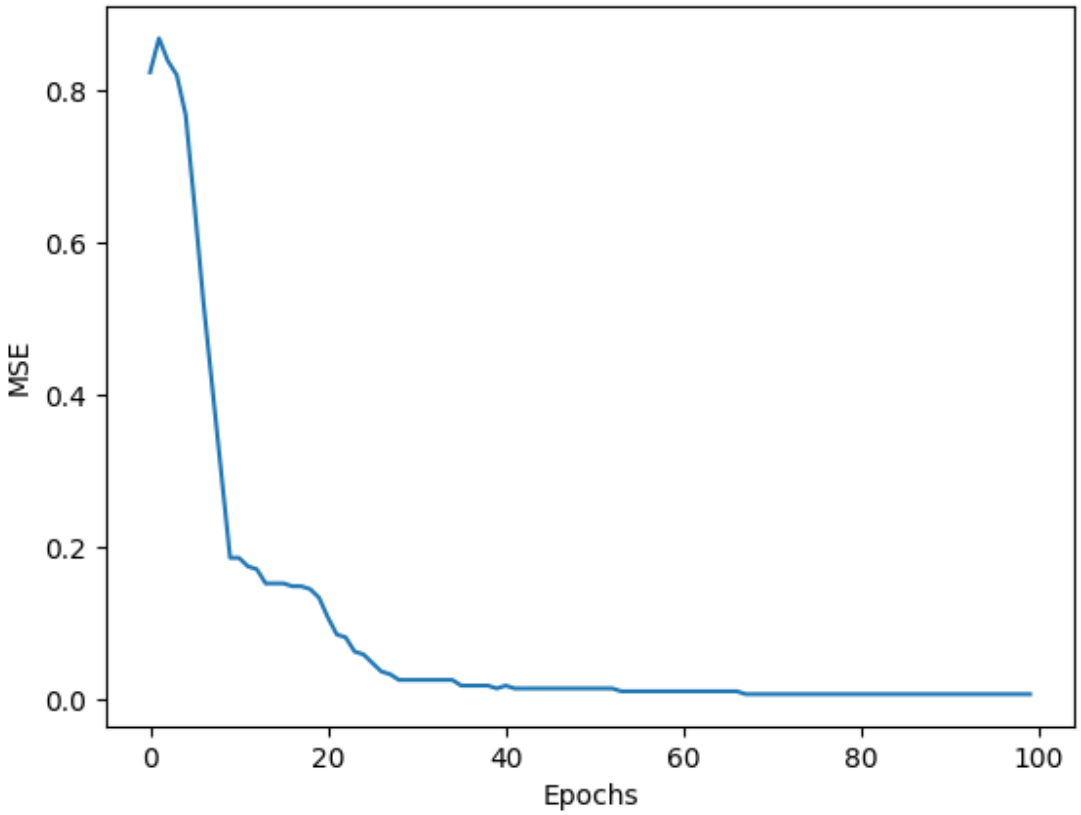
Final weights, biases and accuracy with biases:

w15	w16	w25	w26	w35	w36	w45	w46	w57	w58	w59	w67	w68	w69
-1.42	-11.17	-6.19	5.37	4.2	-0.84	4.65	2.51	-2.71	-7.04	7.55	9.25	-8.18	-5.5

b_5	b_6	b_7	b_8	b_9
-1.35	0.14	-3.75	3.78	-2.78

test accuracy: 65/65 = 100.0%

The neural network with biases shows great improvement, it now correctly classifies all test instances and converging to near perfect training classification as seen by the convergence plot below:



The model converges much slower than the model without biases but it now converges much closer to 100% (99.25%). The introduction of bias nodes allows the model to shift the output to the left or the right by adding a constant to the weighted sums. If we think about the activation function as a graph then the bias nodes allow the curve to be moved left or right, without them then the curve will always go through the origin.

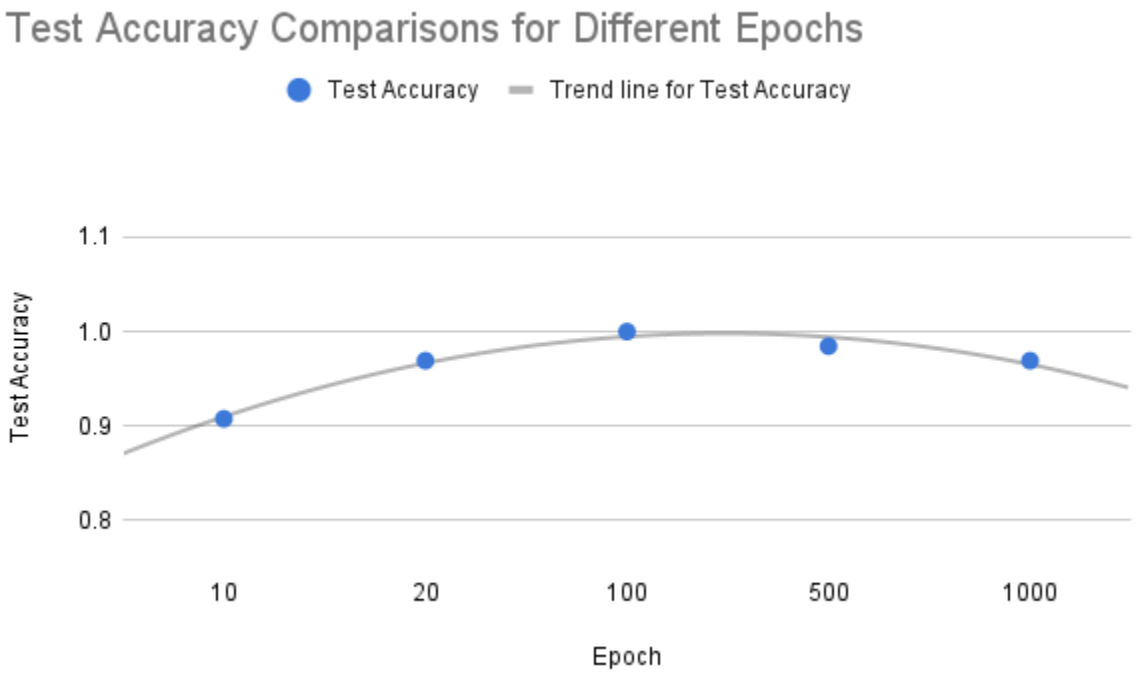
Sensitivity Testing of Hyper Parameters:

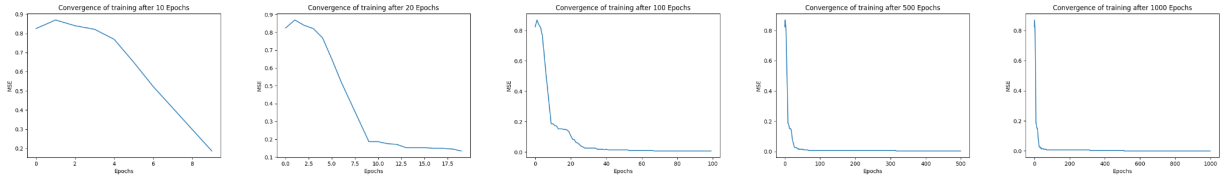
Sensitivity Testing of number of epochs:

Epochs are an important way of adjusting the algorithm, it tells the algorithm when to stop. In the default algorithm the number of epochs chosen is 100 which yields a perfect test accuracy of 100% but what happens if the number of epochs is increased or decreased? In theory a small epoch doesn't sufficiently learn from the data and a large epoch learns too well and doesn't generalise well to the unseen data - it will overfit. Adjusting the number of epochs yields the following results:

Nicholas Tran, 300296259

Epoch	Test Accuracy
10	0.9077
20	0.9692
100	1
500	0.9846
1000	0.9692





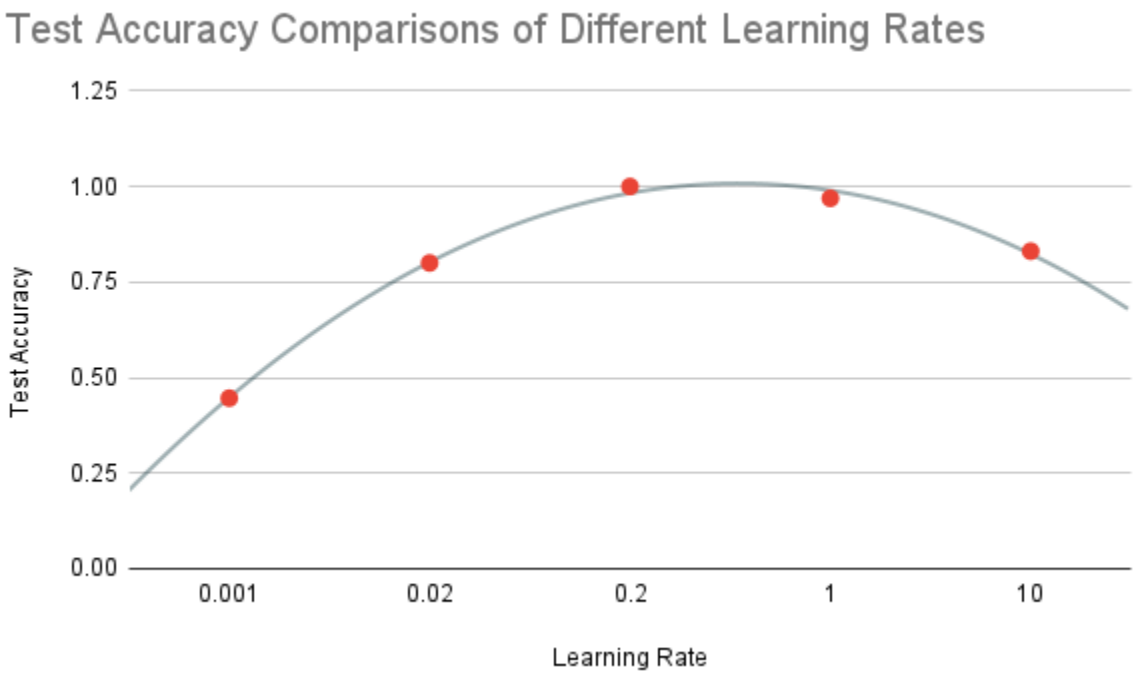
The results are precisely what was to be expected. When the epoch is as low as 10 it's not as accurate as 100 epochs and then when it increases to 20 the test accuracy improves but still not 100% then at 100 it perfectly classifies the instances, however when we further increase the epochs to 500 and 1000 the testing accuracy actually decreases. The higher epochs even though having lower test accuracy has higher training accuracy. A textbook example of overfitting. Looking at the convergence graphs, the algorithm at 10 epochs fails to converge. 20 epochs is on the way to converging, 100 epochs converges close to 100% (0 error) but 500 and 1000 epochs fully converge to 100. There's likely an epoch between 100 and 500 that correctly classifies both all test instances and all training instances (answer: there isn't).

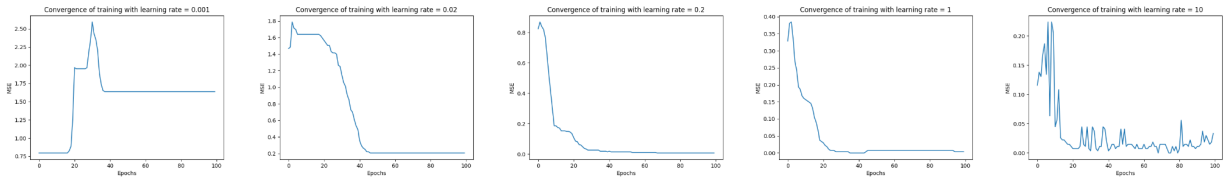
Nicholas Tran, 300296259

Sensitivity Testing of the learning rate:

The role of the learning rate in the algorithm is to adjust the rate of learning. A small learning rate can help the algorithm to get out of local optimas at the cost of time and computation, however a large learning rate can quickly find an optima but can overshoot the true global optima due to its big swings.

Learning Rate	Test Accuracy
0.001	0.4462
0.02	0.8
0.2	1
1	0.9692
10	0.8308





Common values for the learning rate are 0.2 and 1. In this instance 0.2 is the perfect learning rate and 1 actually decreases the performance slightly. A lower learning rate, say 0.02, underfits the

data slightly and a much higher learning rate of 10 has a jagged pattern when looking at the training epochs. The jagged pattern is likely due to the high learning rate causing the weight changes to constantly overshoot/undershoot the optimum weights. A really small learning rate of 0.001 doesn't learn fast enough and can't explain the data well. The training epochs for the learning rate of 0.001 is weird, it starts at the lowest error of 0.75 before spiking up to 2.5 then converging somewhere between 1.5 and 1.75 MSE (mean squared error) .

Part 2: Genetic Programming for Symbolic Regression

The Terminal Set

The terminal set is a set that contains:

- The external inputs to the program. These are represented by variables such as x or y.
- The constants. These can be specified, created by mutation or randomly generated.
- Any function that has no arguments that has side effects.
-

In this GP, the external inputs are x the numbers being plugged into the functions. The constants are just the ephemeral random constants generated by random.randint which is a set of fixed random constants between -1 and 1.

The Function Set

The function set is a set of primitives, functions that take arguments related to the problem domain and are used to solve them. Our problem involves mapping a series of x values to a series of y values and finding a function that can explain the relationship between the two. In this GP the primitives are the
Basic arithmetic operators:

- Addition
- Subtraction
- Division
- Multiplication

Other mathematical operators:

- Negation
- Square Root
- Natural Logarithm
- Inverse
- Absolute Value
- Exponent

The two important criterias for a function set is **closure** and **sufficiency.** By closure, means It must have type consistency which means that the functions must return the same type and its arguments must also have the same type. Closure also means it must also have evaluation safety,

some operators can produce runtime errors due to its inputs so certain constraints must be put upon certain functions.
The following operations are modified and protected to allow evaluation safety:

- Division: If the numerator or the denominator is close to zero then the function returns zero instead of trying division by zero.
- Square Root: Returns the square root of the absolute value of x so it nevers tries to square root a negative number.
- Natural Logarithm: For very small numbers returns zero else returns the natural logarithm of the absolute value of x.
- Inverse: For values close to zero returns zero.

Other non-mathematical functions were also tried such as Min() and Max() which yielded significantly improved results (MSE of as little as 0.01), however it becomes harder to interpret and much harder to justify considering the problem involves evolving a mathematical function and Min() and Max() are not proper mathematical functions. The trigonometric functions such as sine, cosine and tangent are also popular functions for symbolic regression however the results of including them haven't produced good results.

The Fitness Function

The Fitness function used is MSE, the mean squared errors. The MSE is calculated by squaring the errors which is the actual result minus the desired result, summing them then calculating the mean by dividing them by the number of instances.
MSE = ((f(x) - y)^2)/n, where f(x) is the output of the function, y is the response variable, n is the number of instances. The goal of the GP is to find a mathematical function to minimise the fitness function/the MSE.

The Parameter Values & Stopping Criteria

Grow Type: Ramp Half&Half
There are 3 types of ways to grow the tree, Full, Grow and Ramp half&half.
In Full the nodes are chosen from the function set until a certain depth is reached and it ensures the tree is full and balanced. In Grow the nodes are chosen from both the function and terminal sets, when a terminal is selected then the algorithm moves onto the next non terminal tree. Ramp half&half has the benefits of both grow and full by generating half the population by the grow method and the other half by the full method.
Population: 500
Population should be the largest value possible that the system can handle gracefully, at least 500.
Crossover Rate: 0.9
Traditionally 90% of children are created using crossover.
Mutation Rate: 0.1
If the crossover rate is 90% then the mutation rate should be 10%.
Number of Generations: 300

Nicholas Tran, 300296259

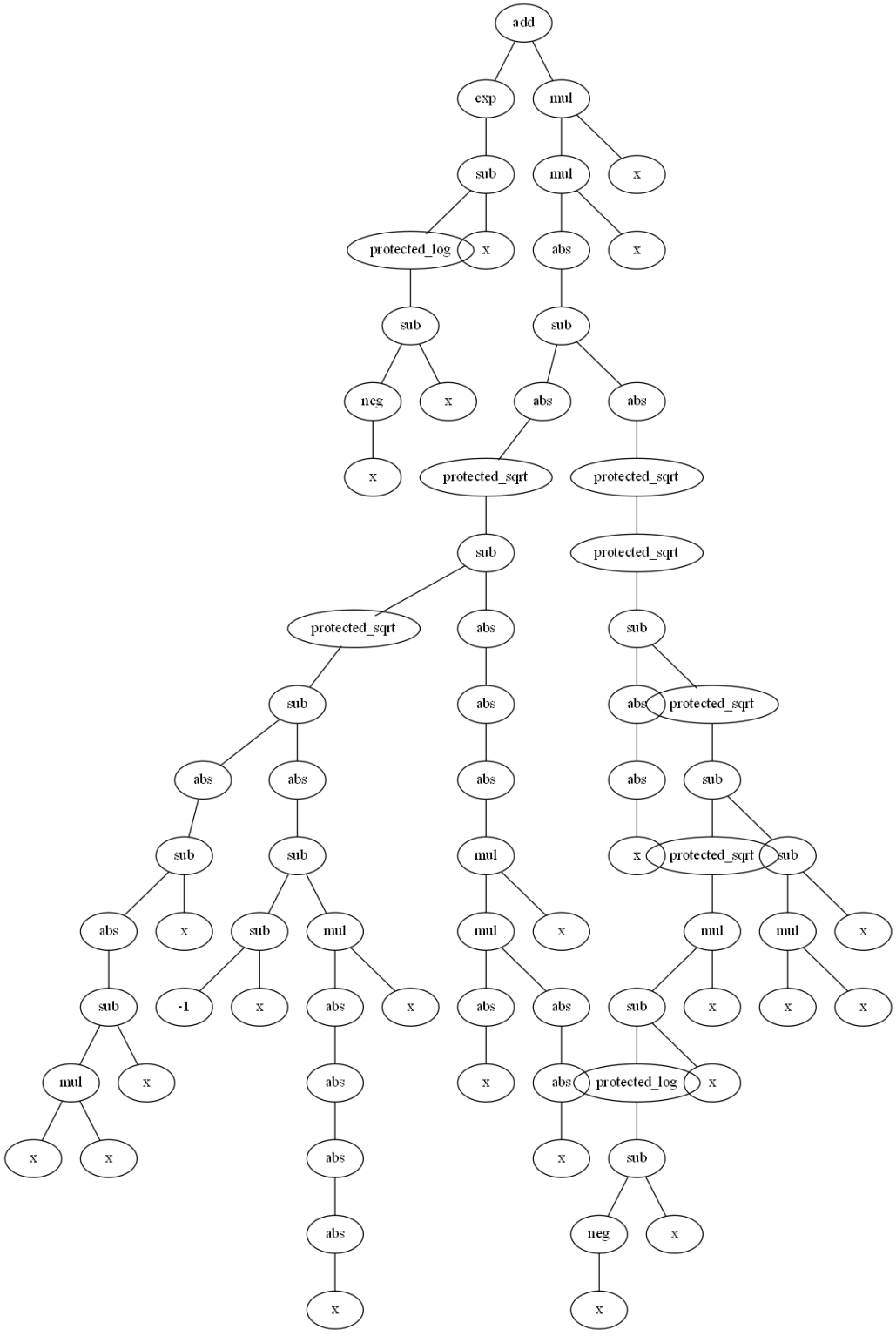
Number of Generations should be large enough to find a sufficient solution but not too large as to needlessly increase runtime. A number of generations of 300 was sufficient enough to be able to break out of local optimas and had a noticeable increase in performance compared to 100 or 200 generations.
Tree Depth: 17
Koza in his book on Genetic Algorithms suggests a max depth of 17.

Three Different Best Solutions

Random Seed: 123

Nicholas Tran, 300296259

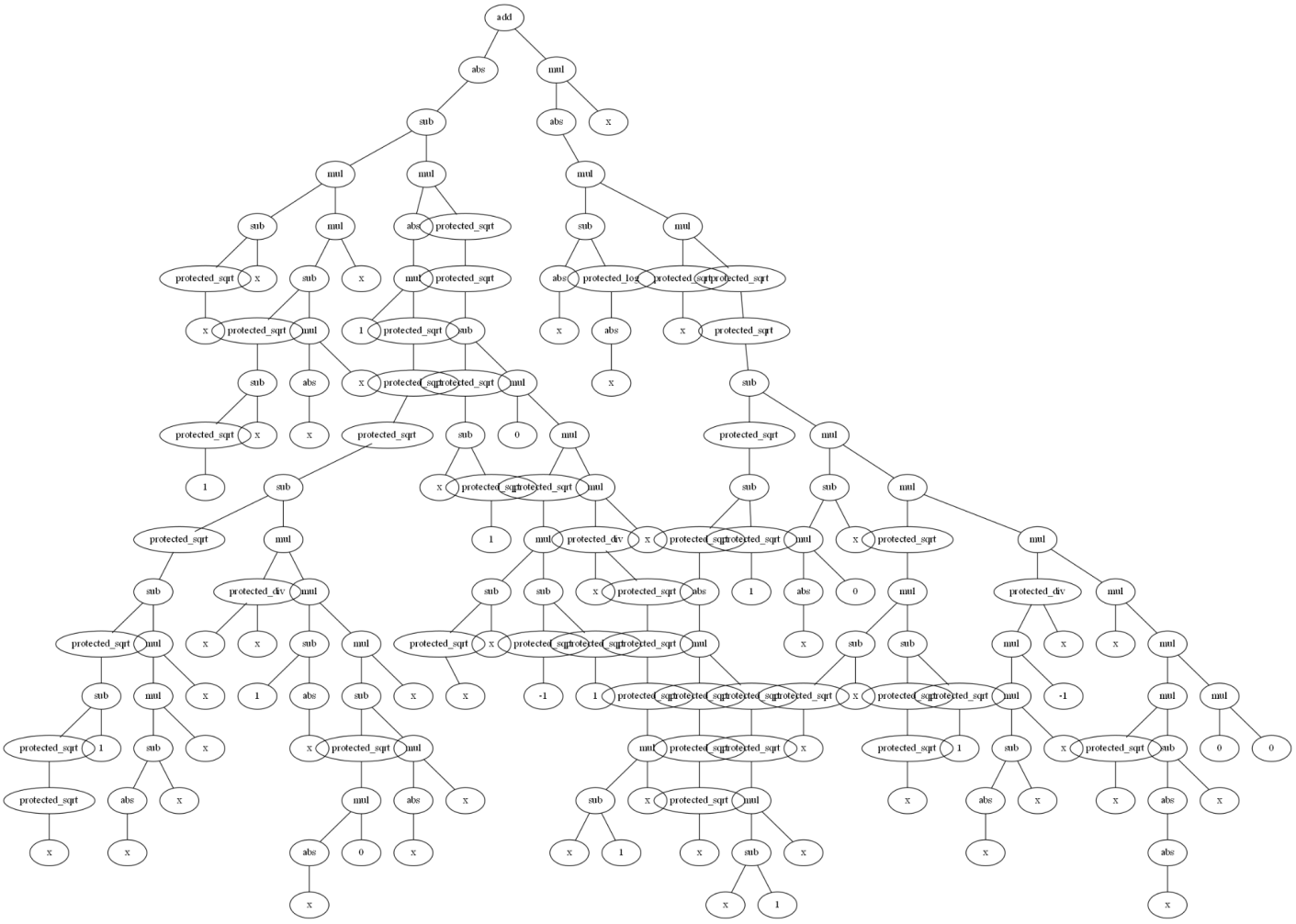
Best Fitness: 0.14397483855410287 MSE
Best Solution:



Random Seed: 318
Best Fitness: 0.033467403378160884 MSE

Nicholas Tran, 300296259

Best Solution:



Random Seed: 12

Nicholas Tran, 300296259

Best Fitness: 0.0824403406296779 MSE
Best Solution: