COMP 307 Assignment 2
Kandice McLean 300138073

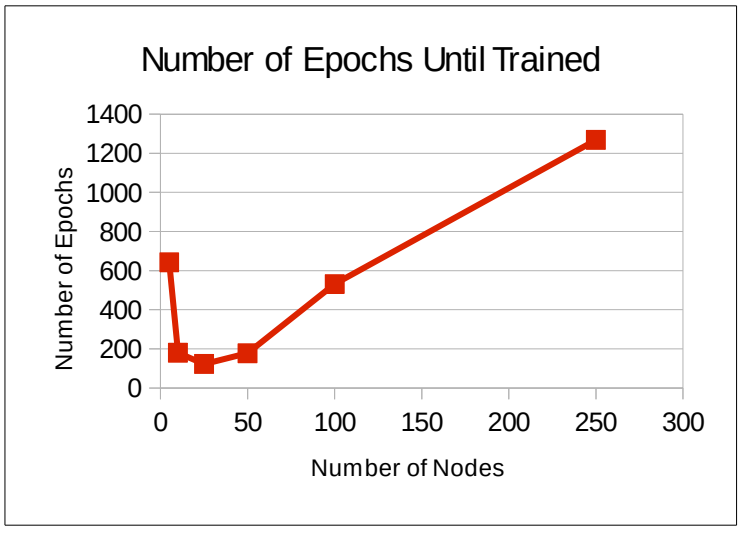
Part One

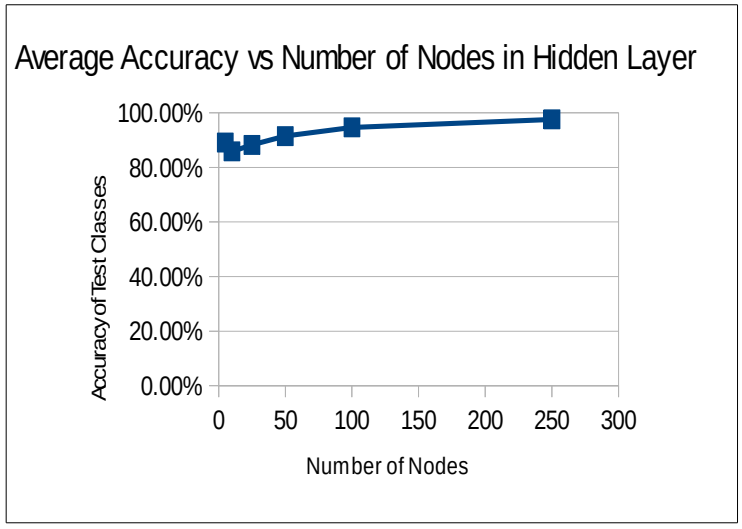
Network Architecture

Number of Input Nodes: 4
Rationale: This is dependant on the number of types of data being fed into the network. In this experiment, the number of inputs was four as there are four numeric attributes: sepal length in cm, sepal width in cm, petal length in cm, and petal width in cm.

Number of Output Nodes: 3
Rationale: The number of outputs correlates to the number of classes the data may be classified as. In the Iris data-set there are three different classes: Iris-setosa, Iris-versicolor and Iris-virginica. These are represented in the outputs as 100, 010 and 001 respectively.

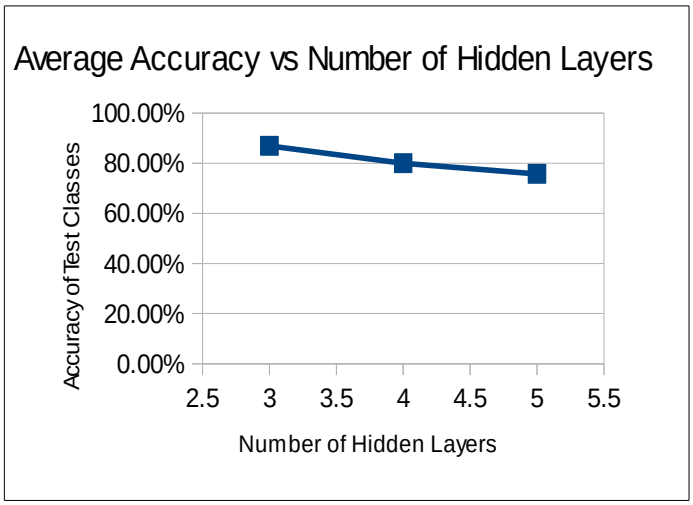
Number of Hidden Nodes: 5
Rationale: An experiment was run testing the accuracy and total number of epochs required to train the network when different amounts of nodes are used in the hidden layer. This experiment assumed only one hidden layer.
The results showed that although it does increase the accuracy to increase this number, the accuracy changes are only very slight whereas complexity is hugely increased. Not only are the number of epochs increased, but increase in nodes each epoch is more complex. There are other ways of increasing accuracy without adding as much complexity, so this number has been kept at 5.

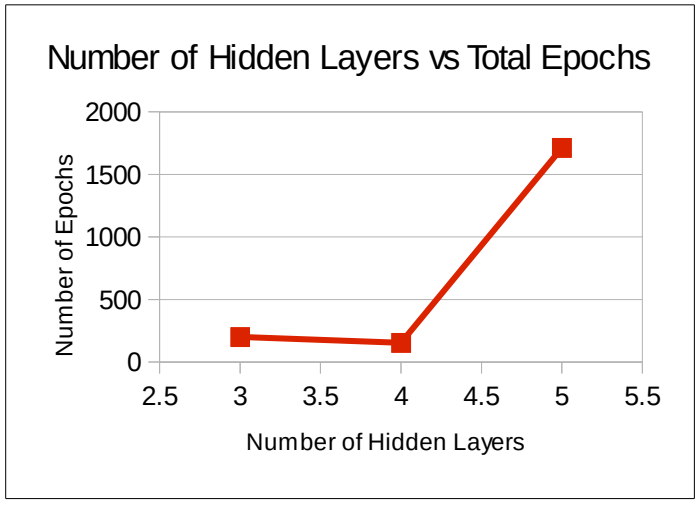




Number of Layers: 3
Rationale: The number of layers will be at least 2, as at a minimum an input and output layer is required. Any number above this indicates hidden layers, so deciding on three layers means an input layer, an output layer and one hidden layer, and any number more than three increases only the hidden layers. To decide the number of hidden layers, some tests were run to compare accuracy vs complexity on 3, 4, and 5 layers, using 5 nodes in each layer.
The test actually showed that the accuracy decreases when adding multiple hidden layers, and the complexity greatly increases.

COMP 307 Assignment 2
Kandice McLean 300138073





Learning Parameters

Learning Rate: 0.1
Rationale: The learning rate should be a small number, as it meant to scale the change of weights.

Momentum: 0.1
Rationale: Like the Learning Rate, this is a scalar value used when taking the previous instances weight change into account, so should be a small value. Setting this to 0 would mean not having the previous output affect the change in the current one.

Initial Weight Ranges: [-0.5,0.5]
Rationale: The weights need to be initialised to a value close to 0 to prevent any bias before training, but they cannot all be 0 as the weights need variation as they are being initialised at the same time. If we initialise weights at the same time to be the same value, then they will have the same gradient during back-propagation so will all be adjusted identically, so must have some variations between them.

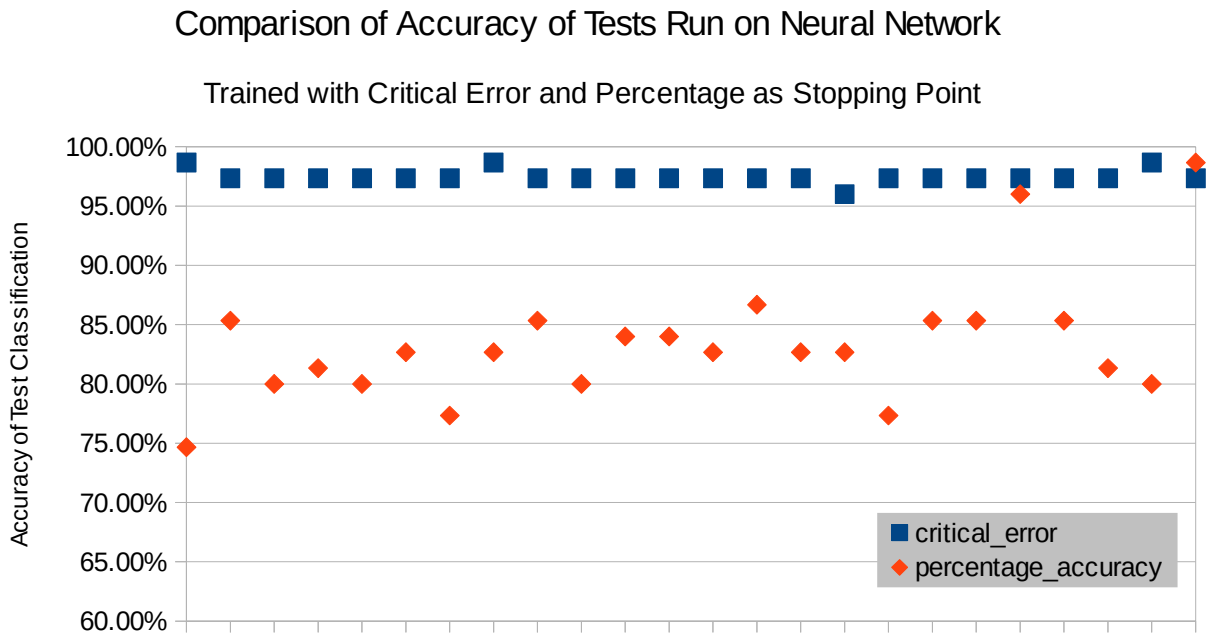
Range: 7.3
This is the range of values across all data used for both testing and training. This value normalises the data, which helps to guarantee stable convergence of weights.

Termination Criteria

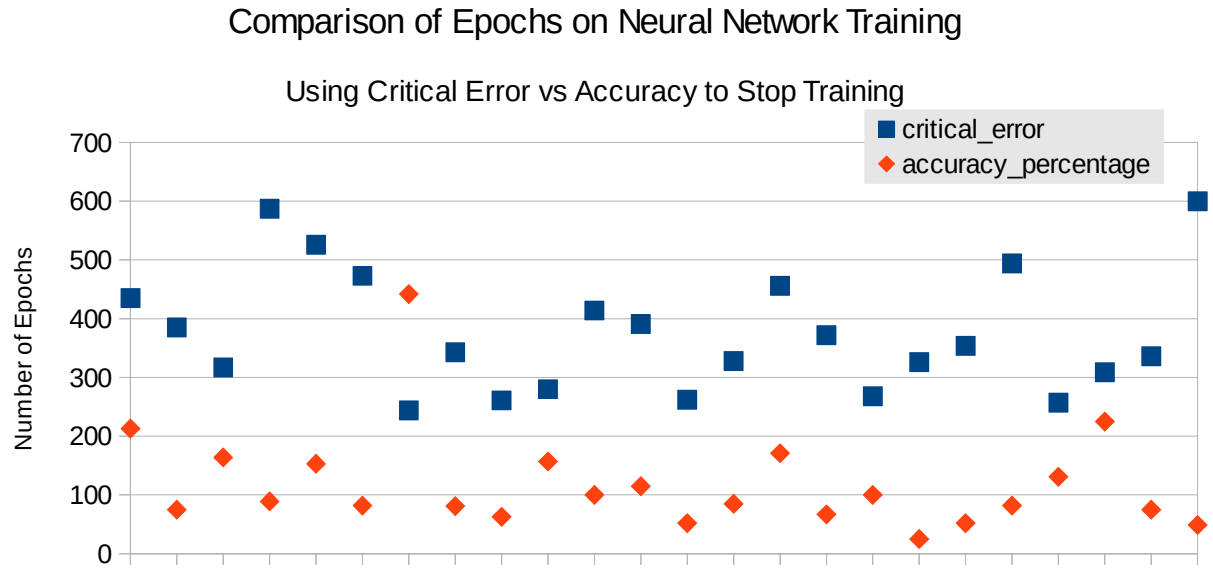
Critical Error: 0.01
The critical error is one of the success conditions for training the network. If the classification error dips below this point, then it can consider its training complete.

Percent: 101
Percent is the other success condition for the network. When it reaches this percentage of accuracy, the network may consider its training complete. The percentage in this network was purposely set to a value it would never reach because after some testing, it was discovered that the accuracy of the network is actually far better if it uses the critical error as a stopping point than percentage. The data

in the following graphs was gathered over 50 tests using all the parameters defined in this report excluding percentage and critical error. For 25 of the tests, critical error was set to 0 and percentage to 100%, so that the training would stop based on accuracy. For the other 25%, percentage was set to 10% and critical error to 0.1, so the network would use the critical error rate as the stopping point.



As you can see using critical error as the termination criteria consistently gives better accuracy on the test than training to percentage does.
This may be the result of under-fitting the data, as using the percentage to signify success in training tends to have lower epochs.



COMP 307 Assignment 2
Kandice McLean 300138073

Part Two

Terminal Set

Aside from the x value inputs, the best terminal set found for this part was using a constant of 1. The original terminal set used was a range from 0.0 to 10.0 with non-integer values allowed. After running the program a few times it was discovered that the selected chromosomes would always have terminals represented as constant/constant, which equals 1. Since the best gene always contained a 1, it was thought best to just pass it a terminal of the constant 1 to speed up the selection process.

Function Set

There were a number of functions to select from, but the set that resulted in the most simplified function and best fit to the test data was to use Add, Subtract, and Multiply.
Using just Add and Subtract did not allow the data to fit correctly, as the function when plotted became a straight line, whereas the final function was a parabola, so needed more dimensions which the multiplication allows for. With the terminal set to a constant of 1, there was no need for divide as all it ever came up for in the best function was for representing another terminal value as 1, so this was removed in order to increase the relevance of the initial population.

Fitness Function

The fitness function used for this part of the project was:

```
for(int i = 0; i < x_Vals.length; i++){

      x_Var.set(x_Vals[i]);
      try {
      double value = program.execute_double(0, NO_ARGS);
      error+= Math.abs(value - y_Vals[i]);


  if (Double.isInfinite(error)) {
      return Double.MAX_VALUE;
    }
  } catch (ArithmeticException ex) {
    System.out.println("x = " + x_Vals[i].floatValue());
    System.out.println(program);
    throw ex;
  }
}

if (error < 0.001) {
  error = 0.0d;
}
return error;
```

For every x value provided in the dataset, calculate the value of the output which results in using x as the input into the function described by the current chromosome. Add the absolute value of the error of the output, which is obtained by subtracting the ideal output from the value returned by the function, to the total error. The returned value after this fitness function is run represents the total distance from the ideal values of the current chromosome.

COMP 307 Assignment 2
Kandice McLean 300138073

Parameter Values

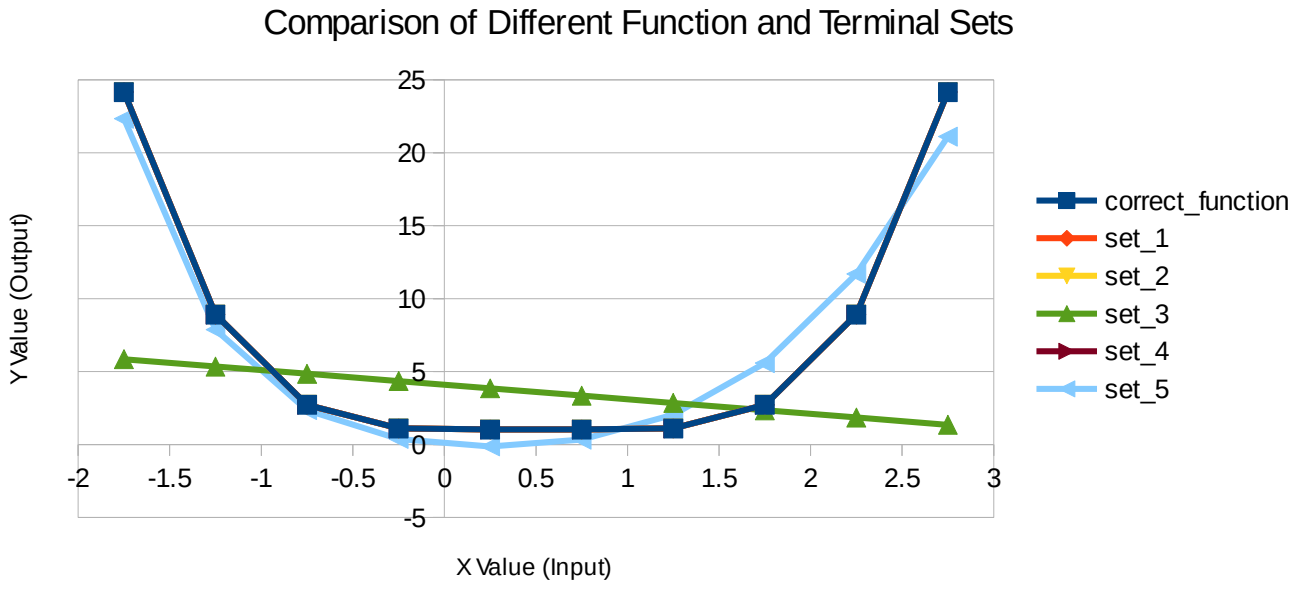
Stopping Criteria
The criteria for success was simply to receive a distance of 0 for a chromosome within 200 generations. If it found a successful chromosome within this amount of generations, it would terminate and return the chromosome with the smallest distance at that point.

GPFitnessEvaluator
The DeltaFitnessEvaluator was used as the fitness function created for the program uses smaller numbers to represent better fit. The DefaultFitnessEvaluator uses larger numbers to represent better fit, so would not work with this particular fitness function.

Population Size
The population size appears to have a large effect on how fast the program runs, with it being noticeably slower with larger initial population sizes, which makes sense as this would mean more chromosomes to check, so would increase the programs complexity. As far as success is concerned, any population >=500 will successfully find a function with a fit of 0 within 200 generations, but anything less than this is inconsistent. Anything higher than 500 would increase the complexity of the program unnecessarily.

Crossover
The default rate was used, which on JGAP is a rate of 35%, which translates to population * 0.35 crossover operations per generation. Each crossover produces 2 individuals, and the crossover point is random.

Mutation
The default JGAP rate of 12 was used. Mutation occurs if a random integer between 0 and 12 is 0, so roughly translates to 1/12 probability. This is applied to each gene in each element of the population that was not produced by crossover, so the likelihood of this occurring is (1/12)*populationSize*chromosomeSize.



[Set 1] << Best Functions and Terminal Sets as per explanations above, but not the only successful one
[Terminal] x, 1.0
[Function] Add, Subtract, Multiply
[Fitness] 0
[Result] ((X * X) * ((X - 1.0) * (X - 1.0))) + 1.0
[Successful] Yes

[Set 2]
[Terminal] x, 0.0 - 10.0
[Function] Add, Subtract, Multiply, Divide
[Fitness] 0
[Result] ((5.0 / 5.0) + (((X * X) - (X + X)) * (X * X))) + (X * X)
[Successful] Yes

[Set 3]
[Terminal] x, 0.0 - 10.0
[Function] Add, Subtract
[Fitness] 68.7
[Result] (4.0 - (4.0 + X)) + 4.0
[Successful] No

[Set 4]
[Terminal] x, 0.00 – 10.0
[Function] Add, Subtract, Multiply, Divide, Power
[Fitness] 0
[Result] ((X * X) * (((X * X) - X) - X)) + (1.0 + (X * X))
[Successful] Yes

[Set 5]
[Terminal] x, 0.00 – 10.0
[Function] Add, Subtract, Multiply, Divide, Power, Exponent
[Fitness] 8.92
[Result] (((X * X) * X) + (((X * X) - X) / (Exp(X)))) – X
[Successful] No

Part Three

Terminal Set
The terminal set must include at least the 9 different attributes, which are Clump Thickness, Uniformity of Cell Size, Uniformity of Cell Shape, Marginal Adhesion, Single Epithelial Cell Size, Bare Nuclei, Bland Chromatin, Normal Nucleoli, Mitoses. Aside from these, a terminal in the range of 1 – 10 was tested (the same as the possible values for each attribute) was tested against not having a terminal to provide alternative coefficients. The alternative coefficients consistently achieved a higher accuracy by 0.25 – 0.5 %.

COMP 307 Assignment 2
Kandice McLean 300138073

Function Set
The function set used was Add, Subtract and Multiply. To find which worked best, first the program was tried using Add, Subtract, Multiply, Power, Exponent, Sine, Cosine and Tangent. After running the program multiple times, it was discovered that the final chromosome only ever contained Add, Subtract and Multiply. Once the other functions were removed, the result lost no accuracy, but the best solution was discovered on average 50 generations earlier. This highlights the benefit of having a population which contains only ideal functions and terminals, as it takes less generations to breed out the unwanted traits.

Fitness Function
The fitness Function used for this part of the assignment:

```
double truePositive = 0, trueNegative = 0, falsePositive = 0, falseNegative = 0;
            for(int i = 0; i < instances.size(); i++){

                  clumpThickness.set(instances.get(i).getAttributes().get(0));
            uniformityOfCellSize.set(instances.get(i).getAttributes().get(1));
            uniformityOfCellShape.set(instances.get(i).getAttributes().get(2));
                  marginalAdhesion.set(instances.get(i).getAttributes().get(3));
      singleEpithelialCellSize.set(instances.get(i).getAttributes().get(4));
                  bareNuclei.set(instances.get(i).getAttributes().get(5));
                  blandChromatin.set(instances.get(i).getAttributes().get(6));
                  normalNucleoli.set(instances.get(i).getAttributes().get(7));
                  mitoses.set(instances.get(i).getAttributes().get(8));

                  try{
                        int classLabel = 0;
                        double result = program.execute_double(0,NO_ARGS);

                        if (result > 0.0) {
                              classLabel = 4;
                        } else {
                              classLabel = 2;
                        }

                         if(classLabel == instances.get(i).getClassLabel()){
                               if(result > 0.0){
                                     truePositive++;
                               }else{
                                     trueNegative++;
                               }
                          }else{
                               if(result > 0.0){
                                     falsePositive++;
                               }else{
                                     falseNegative++;
                               }
                          }

                  }catch(ArithmeticException e){
                        e.printStackTrace();
                  }
            }

            double measure = ((truePositive/(truePositive+falseNegative)) +
(trueNegative/(trueNegative+falsePositive)))/2;
```

```
        return measure * 100;
```

As this program is made to use GP for classification, the fitness function measure is of accuracy rather than error as per the regression function in part 2. This function takes the current chromosome, and sets the variables to the data provided from each instance in a list. It the gets the result from the function using program.execute_double. If the result from this is higher than 0, it will classify the instance as Positive, otherwise it classifies it as Negative. Once it has a classification, it will check with the instance if this is correct. It will then increment the applicable variable: truePositive, falsePositive, trueNegative or falseNegative. Once it has checked all the instances,it divides the correct classifications with themselves + false classifications and divides the entire thing by 2 (as there are two classes) to retrieve a weighted accuracy. The result is multiplied by 100 to turn it into a percentage before it is returned to the main program.

Parameters

Population
The initial population was set to 500, which was high enough to get a good range of chromosomes to select from but small enough so that the program does not run noticeably slowly.

Generations
The generations were set to 200, however it could possibly be set lower, as usually the best chromosome is found within 100 generations.

Crossover
The default rate was used, which on JGAP is a rate of 35%, which translates to population * 0.35 crossover operations per generation. Each crossover produces 2 individuals, and the crossover point is random.

Mutation
The default JGAP rate of 12 was used. Mutation occurs if a random integer between 0 and 12 is 0, so roughly translates to 1/12 probability. This is applied to each gene in each element of the population that was not produced by crossover, so the likelihood of this occurring is (1/12)*populationSize*chromosomeSize.

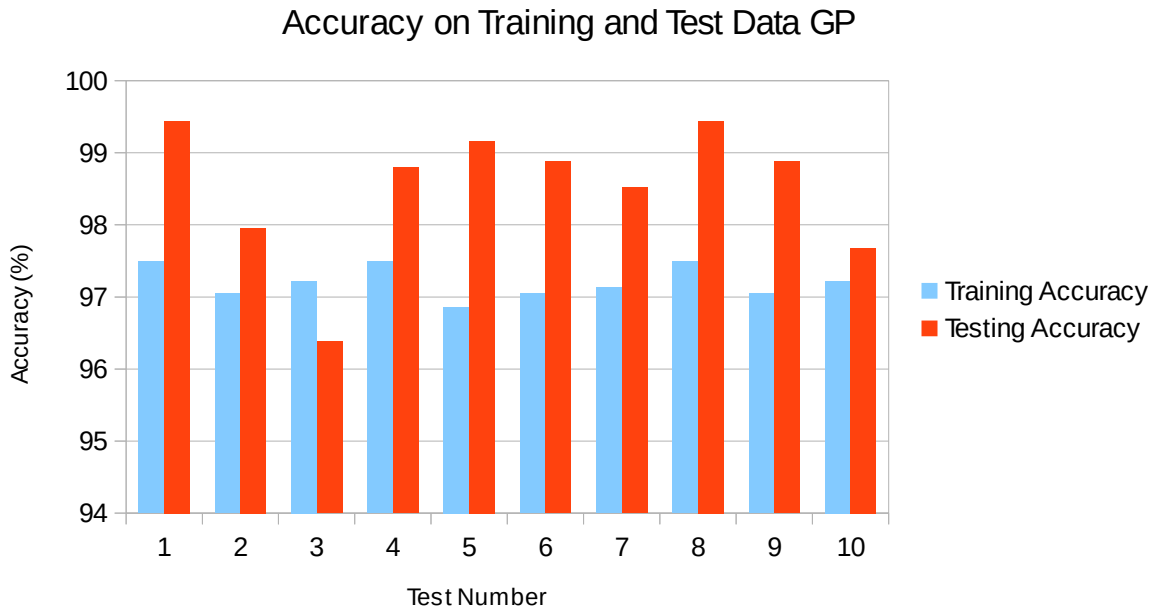
Termination Criteria
The program is considered successful if it finds a function that is >99% accurate on the training set. Otherwise it will stop at generation 200. It is worth noting this success criteria was never achieved on the many tests run.

Data
The data divides evenly into 3 sets, so used two thirds of it as training data and one third to test this. The training set needed to be larger than the test set to prevent underfitting the data. The test set had to be large enough that its accuracy results could be trusted, hence why it was a 2/3 and 1/3 split instead of 3/4 training and 1/4 testing.

COMP 307 Assignment 2

Kandice McLean 300138073

The below graph depicts the Accuracy recorded from running the program with the described parameters, terminals and functions in this report.



Test Number	Formula
1	`(((bn - (5.0 - ct)) * (m * ucsh)) - (7.0 - ucsi)) - (6.0 - ucsi)`
2	`((ucsi - 5.0) + (ucsh * m)) + (ucsh + (((ucsh * bn) - 5.0) - 5.0))`
3	`(bn * ucsi) - (5.0 - (((ct - 5.0) * ct) - 3.0))`
4	`((bn + (ct - 6.0)) * nn) + (ucsi - 5.0)`
5	`((((((ct + ucsi) - 6.0) + (ucsh * bn)) - 6.0) + (bc + ucsh)) - 4.0) - 4.0`
6	`((((m - 1.0) + (ucsi + (ucsh * bn))) + (secs - 7.0)) - 4.0) - 1.0`
7	`(nn + ucsh) + ((bn - 7.0) + ((ucsi - 5.0) - (4.0 - ct)))`
8	`((nn + (((((bn + (ct - 4.0)) * ucsi) - 5.0) - 3.0) * ucsh)) - 3.0) - 3.0`
9	`ct + ((ct + ((((bn * ucsh) + (ucsi - 6.0)) + (nn - 4.0)) - 6.0)) - 5.0)`
10	`(((((((bn * ucsh) + (secs - 3.0)) * m) - 4.0) * ucsh) - 3.0) * nn) - 5.0`