2 Question Description

Part 1: Neural Networks for Classification [30 marks]

Determine and report the network architecture, including the number of input nodes, the number of output nodes, the number of hidden nodes (assume only one hidden layer is used here). Describe the rationale of your choice.

For this classification, I used this .net file

```
lr 0.01

m 0.0

ce 0.002

r 0.5

percent 100

3

4

4

3
```

This declares the the input nodes as 4 as there are 4 features which classify a Iris sepal and petal, lengths and widths. The number of output nodes is 3 as there are 3 possible classifications; Iris-setosa, Iris-virginica, and Iris-versicolor. I used 5 as the number of hidden nodes, as this seems to give the best accuracy in the shortest time, also I didn't want to lower these nodes too over simplify the network, neither increase it as this could lead to overfitting.

Determine the learning parameters, including the learning rate, momentum, initial weight ranges, and any other parameters you used. Describe the rationale of your choice.

Here I used learning rate = 0.01. This seems to give a high accuracy, relatively quickly. Initially the learning I set the learning rate too high, and my neural network was over simplifying and learning too aggressively. Momentum is concerned with how the function can get stuck on a local minima, considering the simplicity and little variance in the data set, it is safe to assume that this can be left at zero.

Determine your network training termination criteria. Describe the rationale of your decision.

My network terminates at 100% accuracy, as there is no point in training beyond this point.

Report your results (average results of 10 independent experiment runs with different random seeds) on both the training set and the test set. Analyse your results and make your conclusions.

Run 1:

epoch = 1171, mse = 0.052, Percentage = 75/75 = 100.00%
Mean squared error for test data: 0.059
Number of incorrect classifications: 1/75

Run2:

epoch = 1184, mse = 0.044, Percentage = 75/75 = 100.00%
Mean squared error for test data: 0.051
Number of incorrect classifications: 1/75

Run3:

Run3:
epoch = 1068, mse = 0.059, Percentage = 75/75 = 100.00%
Mean squared error for test data: 0.065
Number of incorrect classifications: 1/75

Run4:

epoch = 1230, mse = 0.045, Percentage = 75/75 = 100.00%
Mean squared error for test data: 0.052
Number of incorrect classifications: 1/75

Run5:

epoch = 1187, mse = 0.046, Percentage = 75/75 = 100.00%
Mean squared error for test data: 0.053
Number of incorrect classifications: 1/75

Run6:

epoch = 1119, mse = 0.049, Percentage = 75/75 = 100.00%
Mean squared error for test data: 0.055
Number of incorrect classifications: 1/75

Run7:

epoch = 1134, mse = 0.050, Percentage = 75/75 = 100.00%
Mean squared error for test data: 0.057
Number of incorrect classifications: 1/75

Run 8:

epoch = 1017, mse = 0.055, Percentage = 75/75 = 100.00%

```
Mean squared error for test data: 0.061
Number of incorrect classifications: 1/75
```

Run 9:

```
epoch = 1099, mse = 0.051, Percentage = 75/75 = 100.00%
Mean squared error for test data: 0.058
Number of incorrect classifications: 1/75
```

Run 10:

```
epoch = 1297, mse = 0.036, Percentage = 75/75 = 100.00%
Mean squared error for test data: 0.044
Number of incorrect classifications: 1/75
```

Averages:

TRAINING: Ave epoch = 1050.6 Ave mse = 0.0487 Ave percent = 100.00%

TEST: Ave mse on test set = 0.0555 Ave misclassifications = 1/75

(optional/bonus) Compare the performance of this method (neural networks) and the nearest neighbour methods.

The results are similar for these two methods of classification. K nearest neigh does not require training which could be seen as an advantage. However it requires multiple trial runs to find the optimal k value. All the data must be stored too, on the contrary neural networks require training but once the network is trained, the training data can be discarded.

Part 2: Genetic Programming for Symbolic Regression [30 marks]

1. **Determine a good terminal set for this task.**

   Terminal sets have no arguments and form the leaves of the tree. Terminals represent the Input feature's of a GP program. In my program we use the X variable as the terminal set. We also included randomly generated constants in the terminal set. Between the range of -5 and 5. This helps ensure sufficiency - the principle that "there must be a combination of terminals and function symbols that can actually solve the problem". (Mengjie Zhang, Lecture 11) It is possible that our equation includes a constant, thus we need to prepare for this. The range, although may seem small actually can be manipulated in evolutions after being processed by functions in the function set to give a wide range of possible

constants. These are generated randomly, which means every time the behaviour of the program will vary, even if given the same input.

2. **Determine a good function set for this task.**

"The terminal set and function set should be selected so as to satisfy the requirements of closure and sufficiency" (Mengjie Zhang, Lecture 11). Sufficiency, ensure that the must be a combination of terminals and function symbols that can solve the desired problem. Where as closure, is the idea that any function can accept any output value from another a function in the tree, as an input. For this problem I used the following functions: multiply, divide, add, subtract, and to the power of. Initially I Also had the sign function as well as some more complicated math functions, however I believe this was over complicating the problem, and often lead to a best solution having a fitness function rating of around 13.00.

3. **Construct a good fitness function and describe it using plain language (and mathematical formula, or other formats you think appropriate).**

We assign fitness to each function tree, and choose the best one. For every input compute the difference between output and real y values. This can be average it over theses over our training set eg math.abs (result - Y expected value) * 1/n. However for this problem I just implemented absolute error.

4. **Describe the relevant parameter values and the stopping criteria you used.**

I defined the population size at 1024 as this is usual for most GPNN systems. The cross over rate, reproduction rate, and mutation rate must all sum to 1. I used 0.7 for cross over rate, this means that there 70% of chromosomes in the next generation are created by crossing over. And 0.2 mutation rate, again meaning that there is a 2% chance that a given chromosome will be mutated. This works like momentum in the BPNN, where a higher mutation rate would help the network "get out off" local optima. For the reproduction rate I used a value of 0.1, this represents the rate at which programs form the current generation are cloned into the new generation. It enables 'good' programs to survive. **For the selection process we can either choose roulette wheels. Where the lower your error rate i.e higher fitness means your more likely to get selected (larger segment of the pie chart), this is evaluated across the whole set. Or tournaments. Where are randomly selected sets compete (in terms of fitness) where the lowest fitness rate is the winne**r. The problem was set up with tournament selection. Stopping criteria is enforced in a number of ways: First we can have the number of generations reached. Or we can give the network a fitness threshold. E.g if error <

= 0.001/0.0 we stop. This error rate can be low, because our desired outputs (y values) are relatively small numbers. If these values were larger we could be more lenient with error being higher.

5. **List three different best programs evolved by GP and the fitness value of them (you need to run your GP system several times with different random seeds and report the best programs of the runs).**

Best 1 : `Best solution fitness: 0.0`

`Best solution: ((X + −2.0) ∗ (X ∗ X ∗ X)) + (1.0 + (X ∗ X))`

Best 2 : `Best solution fitness: 0.0`

`Best solution: X + (((1.0 - X) * (1.0 - X) * (X * X)) + (1.0 - X))`

Best 3 : `Best solution fitness: 0.0`

```
Best solution: (4.0 / 4.0) + ((X * X) + ((X - (4.0 / 2.0)) * (X * X *
X)))
```

6. **(optional, bonus) Analyse one of the best programs to reveal why it can solve the problem in the task.**

Part 3: Genetic Programming for Classification [40 marks]

1. **Determine a good terminal set for this task.**
   For this problem the terminals are all the attributes that describe a patients health. Then I also added a terminal of a random number between 5 and -5. This was incase there is a constant in our formula, that would do the classification.

2. **Determine a good function set for this task.**
   For the function set we like to keep it simple . I used addition, subtraction, multiplication and division. These were recommended in tutorial 6. And seems to work well for the given data set.

3. **Construct a good fitness function and describe it using plain language (and mathematical formula, or other formats you think appropriate).**
For the fitness function we used a threshold function, with a threshold of 0. This is because there are 2 possible classifications. If the result is above 0 it will be declared as a 4 aka malignant. And 2, if result is under 0, implying it was classed as benign.

4. **Describe the relevant parameter values and the stopping criteria you used.**

The program uses an initial population of 1000. The mutation rate is 0.1 and the reproduction rate is 0.1 too. The population runs for 800 generations, however this could be reduced as best solution is usually found quite a bit earlier than this.

5. **Describe your main considerations in splitting the original data set into a training set training.txt and a test set test.txt.**

The data is split by data splitter. The data itself is split into malignant and benign instances it than takes the first 70 percent of these two sets, and randomises the to be used as training instances. The remaining 30 of the two sets is randomised and used as the test set file.

6. **Report the classification accuracy (average accuracy over 10 independent experiment runs with different random seeds) on both the training set and the test set.**

```
RUN1
[JGAP][17:54:17] INFO GPGenotype - Best solution fitness: 97.55
[JGAP][17:54:17] INFO GPGenotype - Best solution: (((5 - (USz * bn)) + ((4 - CT) + (4 -
Ushp))) + (4 - nn)) - m ==> ((Arg(0) + Arg(0) + Arg(2)) + Arg(0) + Arg(2)) + (Arg(0) +
(Arg(1) + Arg(0) + Arg(2)) + Arg(0)) + Arg(0)
[JGAP][17:54:17] INFO GPGenotype - Depths of chroms: 5 / 3
number of correct classifications = 203/209
accuracy = 97.1291866028708

RUN2
[JGAP][19:20:52] INFO GPGenotype - Best solution fitness: 97.95
[JGAP][19:20:52] INFO GPGenotype - Best solution: (nn * (((5 - Ushp) + ((5 - (CT - USz))
- bn)) - USz)) + (5 - USz) ==> Arg(2) + Arg(0) + Arg(1)
[JGAP][19:20:52] INFO GPGenotype - Depths of chroms: 7 / 1
number of correct classifications = 203/209
accuracy = 97.1291866028708

RUN3
[JGAP][19:22:41] INFO GPGenotype - Best solution fitness: 97.95
[JGAP][19:22:41] INFO GPGenotype - Best solution: (((3 - bc) + ((4 - USz) / USz)) + ((5 -
USz) + (5 - CT))) - bn ==> (Arg(1) + Arg(2) + Arg(0)) + Arg(1) + Arg(0)
[JGAP][19:22:41] INFO GPGenotype - Depths of chroms: 5 / 2
number of correct classifications = 202/209
```

accuracy = 96.65071770334929

RUN4
[JGAP][19:24:00] INFO GPGenotype - Best solution fitness: 97.95
[JGAP][19:24:00] INFO GPGenotype - Best solution: ((((5 * ((5 - -2) - -2)) / CT) - USz) - (bn * USz)) - bc ==> Arg(2) + Arg(0) + Arg(0)
[JGAP][19:24:00] INFO GPGenotype - Depths of chroms: 7 / 1
number of correct classifications = 201/209
accuracy = 96.17224880382776

RUN5
[JGAP][19:25:21] INFO GPGenotype - Best solution fitness: 97.75
[JGAP][19:25:21] INFO GPGenotype - Best solution: (((5 - Ushp) + (5 - (m - (4 - (Ushp * bn))))) - USz) + (3 - bc) ==> Arg(2)
[JGAP][19:25:21] INFO GPGenotype - Depths of chroms: 7 / 0
number of correct classifications = 202/209
accuracy = 96.65071770334929

RUN6
[JGAP][19:41:35] INFO GPGenotype - Best solution fitness: 97.55
[JGAP][19:41:35] INFO GPGenotype - Best solution: ((((4 - (5 - (4 - m))) - (USz / ma)) / USz) - (bn - 5)) - Ushp ==> Arg(2) + Arg(0) + (Arg(2) + Arg(0) + Arg(0))
[JGAP][19:41:35] INFO GPGenotype - Depths of chroms: 7 / 2
number of correct classifications = 198/209
accuracy = 94.73684210526315

RUN7
[JGAP][19:47:46] INFO GPGenotype - Best solution fitness: 97.75
[JGAP][19:47:46] INFO GPGenotype - Best solution: (5 - Ushp) - (bn - (((USz + 5) + ((bc / nn) - (bc * USz))) / nn)) ==> Arg(1) + (Arg(0) + Arg(1) + Arg(1)) + Arg(2)
[JGAP][19:47:46] INFO GPGenotype - Depths of chroms: 6 / 2
number of correct classifications = 200/209
accuracy = 95.69377990430623

RUN8
[JGAP][20:36:18] INFO GPGenotype - Best solution fitness: 97.34
[JGAP][20:36:18] INFO GPGenotype - Best solution: ((((1 - bn) * Ushp) + (((5 - USz) + (5 - USz)) - Ushp)) + 5) - USz ==> Arg(2) + Arg(0) + Arg(0)
[JGAP][20:36:18] INFO GPGenotype - Depths of chroms: 6 / 1
number of correct classifications = 204/209
accuracy = 97.60765550239235

RUN9
[JGAP][20:47:19] INFO GPGenotype - Best solution fitness: 97.55
[JGAP][20:47:19] INFO GPGenotype - Best solution: (4 + (2 + (3 + (((USz * bn) / Ushp) - CT)))) - (USz * bn) ==> Arg(1)
[JGAP][20:47:19] INFO GPGenotype - Depths of chroms: 7 / 0
number of correct classifications = 202/209
accuracy = 96.65071770334929

RUN10
[JGAP][20:50:03] INFO GPGenotype - Best solution fitness: 97.55
[JGAP][20:50:03] INFO GPGenotype - Best solution: ((bn + (4 + ((3 - Ushp) - USz))) + (4 + (5 - Ushp))) - (bn * CT) ==> Arg(1)
[JGAP][20:50:03] INFO GPGenotype - Depths of chroms: 6 / 0
number of correct classifications = 199/209
accuracy = 95.21531100478468

AVERAGES:
- ave number of correct classifications = 201.4

- Ave best solution fitness = 97.684
- Ave accuracy on test set = 96.37

7. List three best programs evolved by GP and the fitness value of them.

```
RUN1
[JGAP][17:54:17] INFO GPGenotype - Best solution fitness: 97.55
[JGAP][17:54:17] INFO GPGenotype - Best solution: (((5 - (USz * bn)) + ((4 - CT) + (4 -
Ushp))) + (4 - nn)) - m ==> ((Arg(0) + Arg(0) + Arg(2)) + Arg(0) + Arg(2)) + (Arg(0) +
(Arg(1) + Arg(0) + Arg(2)) + Arg(0)) + Arg(0)
[JGAP][17:54:17] INFO GPGenotype - Depths of chroms: 5 / 3
number of correct classifications = 203/209
accuracy = 97.1291866028708

RUN2
[JGAP][19:20:52] INFO GPGenotype - Best solution fitness: 97.95
[JGAP][19:20:52] INFO GPGenotype - Best solution: (nn * (((5 - Ushp) + ((5 - (CT - USz))
- bn)) - USz)) + (5 - USz) ==> Arg(2) + Arg(0) + Arg(1)
[JGAP][19:20:52] INFO GPGenotype - Depths of chroms: 7 / 1
number of correct classifications = 203/209
accuracy = 97.1291866028708

RUN8
[JGAP][20:36:18] INFO GPGenotype - Best solution fitness: 97.34
[JGAP][20:36:18] INFO GPGenotype - Best solution: ((((1 - bn) * Ushp) + (((5 - USz) + (5
- USz)) - Ushp)) + 5) - USz ==> Arg(2) + Arg(0) + Arg(0)
[JGAP][20:36:18] INFO GPGenotype - Depths of chroms: 6 / 1
number of correct classifications = 204/209
accuracy = 97.60765550239235
```

8. (optional, bonus) Analyse one of best programs to reveal why it can solve the problem in the task.