COMP307 Assignment: 2
Name: Karu Skipper
ID: 300417869

Part 1: Neural Networks for Classification [30 marks]

https://github.com/Kayskip/Neural-Networks-for-Classification

1. 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.

I have included an output.txt file of my results from the bpnn.

Number of layers: 3
Units in input layer: 4
Units in hidden layer: 3
Units in output layer: 3

We will assume that only **one hidden layer** is used in this assignment, this results in **1 input layer, 1 hidden layer** and **1 output layer**. Any higher than this will result in a larger amount of epochs, which is unessarily more difficult to handle. I have chosen 3 output nodes based off the 3 different iris flower classes: Veriscolour, Setosa and Virginica. I have chosen **4 input layers** based of the iris flower attributes: **sepal length**, **sepal width**, **pedal length** and **pedal width**. Using all the attributes of the flower as inputs will increase the accuracy of our output.

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

epsilon = 0.2;
momentum = 0.0;
range = 0.1;
ecrit = 0.03;

As described in the lectures, a good starting point for the **epsilon** is 0.2, if the weights seem to diverge then decreasing the epsilon is a good way to obtain the learning rate for your Neural Network. In this case I have defaulted the epsilon at 0.2.

Momentum has been defualted at 0.0. Changing this did not affect the output or accuracy of my results. Momentum helps your Neural Network to find the global minimum, if this momentum is too small then your neural network may settle for a minimum that is not the global minimum. Having this momentum altered did not seem to affect the accuracy of the network.

Range was changed to 0.1 as increasing this allowed for more errors in the output, this meant a random value to be chosen between -0.1 and 0.1. However this change was not as significant as predicted, this maybe due to the fact that the weights change immediately as you train the network.

Ecrit (Critical Error) has been left at its default value as increasing this value increased the room for errors in the output. Decreasing this slightly below 0.03 did not affect any results, therefor I have left it in its current state.

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

Termination classification accuracy: 101.0%
Critical error: 0.03

I have set the classification error (mse) to 101.0% so the criteria is never met. Instead I have set the critical error (ecrit) to 0.03 to be satisfied instead.

4. 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.

1)
Mean squared error for test data: 0.052
Number of incorrect classifications: 4/75

2)
Mean squared error for test data: 0.057
Number of incorrect classifications: 5/75

3)
Mean squared error for test data: 0.055
Number of incorrect classifications: 4/75

4)
Mean squared error for test data: 0.053
Number of incorrect classifications: 4/75

5)
Mean squared error for test data: 0.058
Number of incorrect classifications: 5/75

6)
Mean squared error for test data: 0.057
Number of incorrect classifications: 5/75

7)
Mean squared error for test data: 0.051
Number of incorrect classifications: 4/75

8)
Mean squared error for test data: 0.054
Number of incorrect classifications: 4/75

9)
Mean squared error for test data: 0.056
Number of incorrect classifications: 5/75

10)
Mean squared error for test data: 0.053
Number of incorrect classifications: 4/75

<u>RESULTS:</u>
Mean squared error for all tests: 0.0546
Number of incorrect classifications: 4.4/75

From the results above, the network averaged 4.4/75 incorrect classifications. Which is around **94.133% accuracy**.

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

Using the same iris-test and iris-training data I ran the k-nearest-neighbour algorithm and recieved these results:

Training data accuracy = 93.3%
Testing data accuracy = 92.0%

The key difference is that the Neural Network starts with random weights every time, providing variance between each execution of the algorithm. Overall the Neural Network performed better than the k-Nearest-Neighbour by 2% on the testing data set. The Neural Network on average took around 208 epochs to learn the training data set. Where as for the k-Nearest-Neighbor it has a complexity of O(n^2). This results in the neural network having a much better effeciency than that of the k-nearest-neighbor algorithm.

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

https://github.com/Kayskip/Genetic-Programming-for-Symbolic-Regression

1. Determine a good terminal set for this task.

• Variable 'X'
• Terminal (Random Integer between min and max value, I have set this to two and ten)

(Image of Terminal constructor extracted from JGAP javadoc)

```
Terminal(GPConfiguration a_conf, java.lang.Class a_returnType, double a_minValue, double a_maxValue, boolean a_wholeNumbers)
      Constructor.
```

2. Determine a good function set for this task.

Function Set:
• Power
• Multiply
• Add
• Divide
• Subtract

```
new Pow(config, CommandGene.DoubleClass),
new Multiply(config, CommandGene.DoubleClass),
new Add(config, CommandGene.DoubleClass),
new Divide(config, CommandGene.DoubleClass),
new Subtract(config, CommandGene.DoubleClass),
new Terminal(config, CommandGene.DoubleClass, 2.0d, 10.0d, true)
```

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

To evaluate the performance of a program in Genetic programming, you need a fitness function. I created a fitness function that iterates through all inputs (20). for every input it runs the program and gives a result. This result is then compared with the true result value and the difference between the two is take as the fitness value. This is then average over every input x to give the average fitness value. The lower the fitness value the better.

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

- Max Crossover Depth: (8.0)
- Max Init Depth : (4.0)
- Population Size: (1000)
- Strict Program Creation: (true)
- Reproduction Prob: (0.2)
- Crossover Prob: (0.9)
- Mutation Prob: (35.0)

```
config.setMaxCrossoverDepth(8);
config.setMaxInitDepth(4);
config.setPopulationSize(1000);
config.setStrictProgramCreation(true);
config.setReproductionProb(0.2f);
config.setCrossoverProb(0.9f);
config.setMutationProb(35.0f);
```

I set the maximum initial depth to 4 as this is the default for most GP programs, as it is a solid starting point.

1000 was a good population size for this problem as it is not an entirely large population but is not entirely small either in comparison to other GP programs.

Having an incorrect crossover depth can lead to bad result but I found 8 to work quite well, with the crossover probabilty being 0.9.

A mutation probability of 35 was found by luck, I found there to be not much room for error infinding this value and that having a number slightly higher or lower than 35 yeilded in negative results.

0.2 was a good choice for the reporduction probability as this allows for more variation in the gene pool for the next generation.

Termination Criteria:

Max Evolutions: 1000
Minimum Acceptable Error: 0.001 (0.1%)

The maximum evolutions was never reached, and an error 0.1 percent

Q5) List three different best programs evolved by GP

Trial 1
Evolutions 59
Fitness 0.00
(4/4)+(((X*X)-X)*((*X)-X))

Trial 2
Evolutions 137
Fitness 0.00
(X*X)+((X*(X(X(((X*X)-X)-X)))+(3.0-2.0))

Trial 3
Evolutions 60
fitness 0.00
((X*X)+(4.0/4.0))+((((X*X)-X)-X)*(X*X))

Trial 3 would be the best function with the exanded form y = x4 - 2x3 + x2 + 1