**Part 1**

**DISCLAIMER:** I created my own neural network library for this, as Yi allowed me to do so. Because of this, I may get slightly different results in comparison to the results of other people.

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.**

The neural network architecture used for this dataset was (4, 5, 3), 4 input nodes, 1 hidden layer consisting of 5 nodes, and 3 output nodes. I used 4 input nodes since there are only 4 discrete features for each entry in the dataset, therefore we don’t need to worry about prioritization of features (like when training a neural network on a game like Flappy Bird), and we also don’t need to worry about convoluting the dataset like we would for an image input. The output layer contains 3 nodes since this is a classification ternary classification, and therefore only 3 possible outcomes exist. Any more output nodes and there’s the possibility of an unassigned output node. Any less output nodes and you lose the possibility of classifying one of the irises. **HIDDEN NODE EXPLANATION**

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

Learning rate: 0.1; I initially chose a learning rate of 0.2 as per the lecture slides but eventually found that it would at times jump the local minimum entirely. I found that 0.1 helped keep the cost within a minimum more consistently.

Momentum: **add this**

Initial weight range: -1 to 1; I was deciding between 0 to 1 and -1 to 1. I decided on -1 to 1 since it gives the network more persuasion to compute a negative intermediate output, which helps gauge a better starting point for weights that will eventually converge on a negative value.

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

My network training termination criteria had two parts: either if it reached at least 98% or if the it reached the end of the modified dataset. I chose 98 percent because I thought that was a relatively decent accuracy given the dataset. Since I created my own neural network library, I decided to clone and shuffle the iris dataset 7 times, and then adding that to the original dataset, giving my resulting dataset 19200 rows, or x\*2^n rows, where x is the number of rows in the original dataset (150), and n is the number of times I cloned and shuffled it (in this case, 7). Since I thought 19200 rows was enough to train the network on (especially since it I all completely shuffled), I decided to use that as a stopping criteria as well.

1. **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.**

Learning rate **=** 0.1

|  |  |  |
| --- | --- | --- |
| **Training Cycle** | **Cost** | **Correct/Total** |
| 1 | 0.0742 | 141/150 |
| 2 | 0.0728 | 146/150 |
| 3 | 0.0774 | 145/150 |
| 4 | 0.0747 | 146/150 |
| 5 | 0.0801 | 142/150 |
| 6 | 0.0838 | 142/150 |
| 7 | 0.0603 | 147/150 |
| 8 | 0.0654 | 145/150 |
| 9 | 0.0637 | 144/150 |
| 10 | 0.0851 | 140/150 |

I believe the results are quite consistent around an average cost of 0.07. Given that this neural network is self-created, I am satisfied with the results. I believe if I had iterated through the data more it would converge even further on its local minimum, but decided that a slightly lower cost wouldn’t be worth the time it would take to train the network 10 times with even more iterations. It looks as if training cycle 10 may have jumped a minimum towards the end, since it has a considerably worse cost and correct/total value than the remaining 9.

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

**Part 2**

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

I believe a good terminal set is the input x, as well as a random number constant. I decided on this through following the DEAP documentation and understanding that the task at hand wasn’t an overly-complicated one, since there’s only one input and one output for this function. If there were multiple inputs/outputs then I believe it would be more reasonable to use more than just one random constant terminal.

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

I believe a good function set would be the 4 basic operators (+, -, \*, /), as well as square and negation. I chose these operators based on some trial and error using trigonometric functions like sin and cosine, and came to the conclusion that certain operators didn’t add much value to the end result.

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

For this part I decided to go with the mean squared error. I chose this because it always yields a positive value so the sum can never be zero, as well as the fact that squaring always emphasizes larger differences. This can be both a good and bad thing, since the effect that outliers have will be exaggerated. Mean squared error simply takes the difference between the expect output and the actual output, and squares the resulting number. It does this to each individual on all data points, sums the squared differences, and divides it by the number of points to get the average. **MORE ON THIS**

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

Fitness type: Minimal fitness. Since we are trying to reduce the distance from the individual’s guess to the actual answer, it makes sense to have a smaller fitness be better than a bigger fitness.

Program generation type: Half-and-half; I chose this to make the trees more diverse and test a full tree set against a grow tree set

Selection size: 3 (Chooses the 3 best individuals in a given generation and does crossover/mutation if applicable).

Stopping criteria: The only stopping criteria in place is the number of generations that must be completed, which is 100. This would give the program 100 attempts at mating the best individuals and getting a better fitness, which should be enough for the task that this part brings.

1. **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).**
2. **(optional, bonus) Analyse one of the best programs to reveal why it can solve the problem in the task.**

**Part 3**

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

I believe a good terminal set is the input x, as well as a random number constant **MORE ON THIS**

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

I believe a good function set would be the 4 basic operators (+, -, \*, /). **MORE ON THIS**

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

For this part I decided to go with the mean squared error. I chose this because it always yields a positive value so the sum can never be zero, as well as the fact that squaring always emphasizes larger differences. This can be both a good and bad thing, since the effect that outliers have will be exaggerated. Mean squared error simply takes the difference between the expect output and the actual output, and squares the resulting number. It does this to each individual on all data points, sums the squared differences, and divides it by the number of points to get the average. **MORE ON THIS**

1. **Describe the relevant parameter values and the stopping criteria you used.**
2. **Describe your main considerations in splitting the original data set into a training set**

**training.txt and a test set test.txt**

1. **Report the classification accuracy (average accuracy over 10 independent experiment runs with different random seeds) on both the training set and the test set.**
2. **List three best programs evolved by GP and the fitness value of them.**
3. **(optional, bonus) Analyse one of best programs to reveal why it can solve the problem in the task.**