# Assignment3

COS 314

Mr. TM Whitaker
U22744968

# Artificial Neural Network:

- Parameters:
    - Activation Function: Sigmoid function - I used the sigmoid function as it squishes an output to a value between 0 and 1 which can then be used to compare to the required output.
    - 
    - Learning Rate: 0.8. - Taken from the textbook and found that it worked well.
    - Epoch limit: 50 – Found through testing that this value was all that was needed and was in fact hardly ever reached.
    - Error Change Threshold: 0.0001 – Found this to be a good value when testing.
    - Stopping Condition: Epoch limit reached or if the error change between epochs is less than the specified error change threshold. This ensures that the program only continues to run if its error change is changing a lot otherwise it has reached a plateau and therefore more epochs are not needed. This is done to save time making it more efficient.
- Neuron and Layer Structure: The code first defines a Neuron and Layer class. Each neuron has an error, value, bias, and weights. Each layer consists of multiple neurons.
- Neural Network Structure: The NeuralNetwork class is defined with a learning rate, a vector of layers, and several methods. The constructor initializes an input layer, a hidden layer, and an output layer with random weights and biases.
- Activation Function: The sigmoid and sigmoidDerivative methods are used as the activation function and its derivative.
- Forward Propagation: The forwardPropagate method calculates the output of the neural network for a given input. It does this by passing the input through each layer, calculating the weighted sum of the inputs for each neuron, and applying the sigmoid function.
- Back Propagation: The backPropagate method adjusts the weights and biases of the neurons based on the error of the output. It does this by calculating the error of each neuron, calculating the gradient of the error with respect to the weights, and adjusting the weights in the direction that reduces the error.
- Training: The train method trains the neural network on a set of inputs and outputs. It does this by performing forward and back propagation for each input and output, calculating the mean squared error.
- Prediction: The predict method calculates the output of the neural network for a set of inputs. It does this by performing forward propagation for each input and returning the outputs.
- Metrics Calculation: The calculateMetrics function calculates performance metrics of the neural network, including accuracy, specificity, sensitivity, and F-measure.
- Main Function: The main function reads in training and testing data, initializes a neural network, trains the network, predicts the outputs for the testing data, calculates performance metrics, and prints out the results.
- The model is used to predict the output based on the inputs provided. The output is a binary classification (1 or 0) and the performance of the model is evaluated using metrics like accuracy, specificity, sensitivity, and F-measure.

## Test Run:

Seed: 123456

| Epoch | Error |
|---|---|
| **1** | 0.0229541 |
| **2** | 0.0309381 |
| **3** | 0.0324351 |
| **4** | 0.0269461 |
| **5** | 0.0304391 |
| **6** | 0.0329341 |
| **7** | 0.0369261 |
| **8** | 0.0409182 |
| **9** | 0.0444112 |
| **10** | 0.0374251 |
| **11** | 0.0219561 |
| **12** | 0.0279441 |
| **13** | 0.0224551 |
| **14** | 0.0224551 |

Error change less then threshold therefore early termination.

Time taken: 102ms

Accuracy: 58.8723%

Specificity: 0

Sensitivity: 1

F-measure: 0

## Genetic Programming:

- Parameters:
  - Population size: 100 – Given.
  - Number of generations: 50 – Given.
  - Tree depth: 6 – Found this to be a good balance between time efficiency and accuracy.
  - Functional symbols: “+, -, *, /” - Found these functional symbols to be adequate for the problem given.
  - Terminal symbols: “0-9” – worked well for this problem.
- Node Structure: The code first defines a Node structure which represents a node in the GP tree. Each node can be either a function (an operation like addition, subtraction, multiplication, or division) or a terminal (a constant value).
- Fitness Function: The fitness function calculates the fitness of a tree (a potential solution). It does this by evaluating the tree for each set of inputs in the data, comparing the tree's output to the expected output, and calculating the mean squared error. The lower the error, the better the tree's fitness.
- Population Initialization: The initializePopulation function generates a population of random trees. Each tree is generated by the generateRandomTree function, which recursively creates nodes until it reaches a specified maximum depth.
- Selection: The tournamentSelection function selects a tree from the population to be a parent for the next generation. It does this by randomly selecting a few trees from the population and choosing the one with the best fitness.

- Crossover: The crossover function combines two parent trees to produce a child tree. It does this by recursively copying nodes from the parents, with a 50% chance of choosing a node from either parent.
- Mutation: The mutate function randomly alters a tree to introduce variation. It can change a function node to a different function, change a terminal node to a different value, or change a terminal node to a function node (and vice versa – creating a growing and shrinking effect).
- Evolution: The evolve function evolves the population over a number of generations. In each generation, it calculates the fitness of each tree in the population, selects parents, performs crossover and mutation to produce a new population, and replaces the old population with the new one.
- Main Function: The main function reads in training and testing data, initializes a population of trees, evolves the population over a number of generations, and then prints out the best tree and its performance metrics.
- The model is used to predict the output based on the inputs provided. The output is a binary classification (1 or 0) and the performance of the model is evaluated using metrics like accuracy, specificity, sensitivity, and F-measure.

## Test Run:

Seed: 123

| Generation | Training Accuracy "%" |
|---|---|
| **1** | 64.7705 |
| **2** | 64.7705 |
| **3** | 64.7705 |
| **4** | 64.7705 |
| **5** | 64.7705 |
| **6** | 64.7705 |
| **7** | 35.2295 |
| **8** | 35.2295 |
| **9** | 35.2295 |
| **10** | 35.2295 |
| **11** | 64.7705 |
| **12** | 64.7705 |
| **13** | 64.7705 |
| **14** | 64.7705 |
| **15** | 64.7705 |
| **16** | 64.7705 |
| **17** | 64.7705 |
| **18** | 64.7705 |
| **19** | 64.7705 |
| **20** | 64.7705 |
| **21** | 64.7705 |
| **22** | 64.7705 |
| **23** | 64.7705 |
| **24** | 64.7705 |
| **25** | 64.7705 |
| **26** | 64.7705 |
| **27** | 64.7705 |
| **28** | 64.7705 |

| | |
|---|---|
| **29** | 64.7705 |
| **30** | 64.7705 |
| **31** | 64.7705 |
| **32** | 64.7705 |
| **33** | 64.7705 |
| **34** | 64.7705 |
| **35** | 64.7705 |
| **36** | 35.2295 |
| **37** | 64.7705 |
| **38** | 64.7705 |
| **39** | 64.7705 |
| **40** | 64.7705 |
| **41** | 64.7705 |
| **42** | 35.2295 |
| **43** | 64.7705 |
| **44** | 64.7705 |
| **45** | 64.7705 |
| **46** | 64.7705 |
| **47** | 64.7705 |
| **48** | 64.7705 |
| **49** | 64.7705 |
| **50** | 64.7705 |

Best tree: (/ (+ (* 9 1) (- 0 3)) (- (+ 3 6) (- 1 0)))

Time taken: 2541ms

Accuracy: 58.8723%

Specificity: 0

Sensitivity: 1

F-measure: 0