F20BC - Coursework Report

# Implementation

The approach we took was to create classes for the following: Artificial Neural Network, Particle Swarm Optimisation, Particle, and Hyperparameters.

## Hyperparameters Profile

We created hyperparameter profiles to store all the hyperparameters we plan to adjust. These include the number of layers and number of neurons per layer, activation functions for each layer, and the adjustable parameters for the PSO, which are the inertia, cognitive, social, and global component as well as the number of iterations and the number of particles. All of this helped us to make the testing of different configurations of the PSO and ANN easier.

## Artificial Neural Network

The ANN class stores layers sizes, activation functions, weights and biases. The weights are initialised randomly while the biases are just filled with 0’s, the weights could also be initialised to 0’s as we do not work with the ANN directly as the weights and biases come from the particles. Activation. Functions and layers sizes come from the Hyperparameter profiles. As we are using PSO to optimise the weights and biases we only need the ANN class to have forward propagation method to get the output. The forward propagation works by; for each hidden layer, multiplying the input matrix and the weight matrix together, then adding the resultant matrix with the bias matrix then finally applying the activation function of the layer to the output. After getting the outputs for each hidden layer we then do the same on the output layer consisting of 1 neuron as this is a binary classification problem, applying our activation function for the output layer we get the prediction.

## Particle

The particle class stores the velocity and position of the particle and handles the position update of the particle. It also stores the particles best position and best fitness value. The method that calculates the new position has been completed with boundaries of the min and max values found in the dataset, which turned out to be min: -13 & max: 18. The update of position is simply adding the current position and the velocity matrices together, if the new position falls outside the boundary we randomly reset the particles position within the boundary.

## Particle Conversion

\*\*Explain how Converting a particle into Neural network works, while also explaining how the helper functions that you made help you do that

## Particle Swarm Optimisation

Particle swarm optimisation has 4 functions, the PSO algorithm itself, finding informants of a particle, accessing the fitness of the particle and updating the velocity of the particle. All these functions are explained in more detail in the sections below. Initialising the PSO is done by first extracting all the relevant hyperparameters from the profile that has been chosen, as well as the data to be used and the labels from the data.

### PSO algorithm

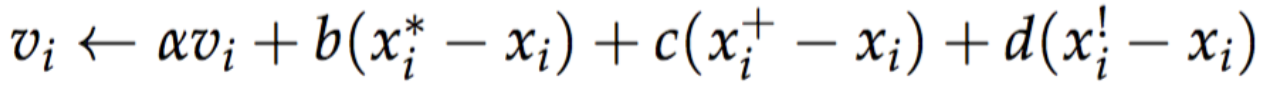
The algorithm consists of first updating the particles position as described in the Particle section, then checking the fitness of the particle, replace the global and personal best positions accordingly if the fitness of the particle is higher than previously set positions, lastly we update the velocity. We do all of this for a set amount of iterations, in the end getting the global best fitness.

### Getting informants

\*\*Explain how you get informants and why they do or don’t update throughout the PSO

### Updating velocity

The velocity update function was implemented using this equation:



First we take the velocity of each particle and multiply it by the inertia (the amount of the velocity we should keep, alpha), Then we will make sure the particle partially moves towards the best position previously discovered by itself (weight b in the equation), then towards the best position discovered by its informants(weight c in the equation), and finally previously discovered global best(weight d in the equation). This is done for each dimension in the velocity vector. The weights b,c, and d are all a random value between 0 and the value of hyperparameters β, γ, and δ this introduces non linearity into the algorithm allowing the particles to search more space without being influenced by the weights all the time, which could lead to a local maxima instead of the global one. The new velocity will then be used in the next iteration of PSO to update the position of the particle within the search space.

### Accessing fitness

For the fitness of the particle the specification mentioned to just use the accuracy of the artificial neural network for the given data and not worry about overfitting. The fitness function takes in the entire dataset, and the labels separately, then convers the particle into a neural network method for which is described in Particle Conversion section, then we run every datapoint in the dataset through the forward propagation method of our artificial neural network and get the predicted values, which then are converted into the classes by using a simple threshold where any values blow 0.5 are classed as 0 and values above 0.5 are class 1. We then compare the predicted labels against the actual labels and get the accuracy by this simple formula: (number of correctly predicted labels / total number of labels \* 100) giving us the percent accuracy of the network.

# Experimental investigation

# Results

# Discussion and Conclusions

# References