Assignment 1 – ANN and Perceptron

Kal Radikov

10157529

October 8, 2016

CMPE 452 – Neural Networks

# Simple ANN with existing Linear Separator

The results for this part were fairly clear cut. I used the graph of the data points to create a line that separates the Iris-Setosa plots from the other flowers, and used the x and y intercept points to determine appropriate weights (w1 and w2) for the line. This line is arbitrary, and any line that separates the two plots will work. The points that were used were: (0,15) and (40,0) and a theta of 2 was used. After inputting a petal length of 2 and a petal width of 5, the function determined that the flower is not an Iris-Setosa

# Perceptron

The perceptron was able to classify 10 out of 10 points correctly using the test data. This is because the algorithm was able to clearly find a set of linearly separated data and determine the proper flower classification using the training data set to initialize the correct weights. The function algorithm works by setting random weighting values ranging from 0 to 1 each time it executes. The weights were then adjusted using simple feedback learning and put back into the function until the output matched with the training data set. A global error was defined and the function was able to break out of looping and adjusting weights if the error was 0 or the max iteration count was reached.

To check that the algorithm was learning correctly, the array of test data and algorithm classification were printed one under another and it could then be checked if the last column matched. The corrected weights were then sent to print statement where they were printed with the equation of the linear separator.

# Adaline Learning

Using Adaline learning was a little more difficult than the perceptron especially when using it classify the Iris-Versicolour and Iris-Virginica data plots. This is due to the fact that the two datasets cannot be clearly separated using a straight line equation. The algorithm works similar to the Perceptron algorithm by setting the weights and modifying them through a while loop until given permission to break out. The difference in learning here is creating a function ‘activationFire’ that creates an output of 0 or 1 which modifies a specific weighting. The error correction is multiplied by this output and either nullifies the correction if the neuron fired when supposed to (i.e. the function produced a 0), or allows the weight to be modified by multiplying it by the output of 1. Again, the loop is only allowed to break if the maximum number of iterations has been reached (1000) or the error is 0. The output of this learning varies by producing at most 4 errors in the classification or as low as 1 error. This can be explained by the random values of weights assigned at the start, and limiting the number of times the algorithm is allowed to loop.