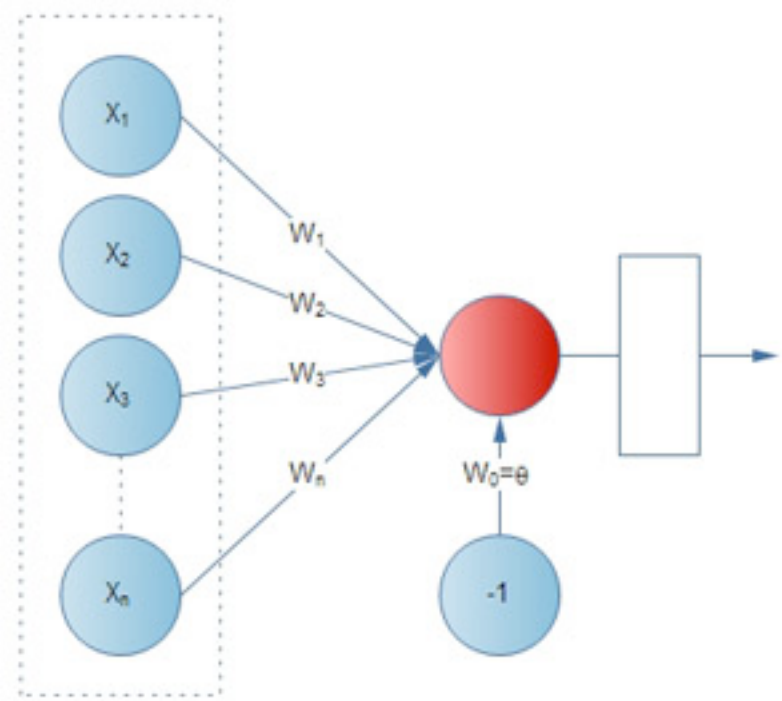
PROGRAMMING ASSIGNMENT 5 Hauschild cs4300 Spring 2018

Overview: In this project you will be implementing a perceptron, a single layer neural network, to perform a classification. A perceptron has n inputs and n weights leading to a threshold function that gives one output. For this project, the output should be either -1 or 1 (indicating the classification



The threshold function for this project is given as:

$$\mathcal{F}(s) = \begin{cases} 1 & \text{if } s > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Essentially, if the total inputs to the function are greater than 0, it indicates it is classified as 1, otherwise it is the other classification.

So that is your network, but now you need to train it. This is an iterative procedure that adjusts the weights over time. You do this by giving a sample to the network. For each weight, the new weight is then calculated by adding a correction to the old value. The dummy weight (w0) is also updated in this way.

$w_i = w_i + L(d-y)x_i$

Where wi is the weight, L is the learning parameter, d is the desired output and y is the actual output from your sample.

Implementation:

In this project you will take in data in the same form that you did for the previous project, except that the inputs are either 1 or -1. So for a set of n data of m variables, you would have each row in the table be m+1 values, with the last one indicating the classification. You would then create a perceptron of m input nodes and m+1 weights (the dummy input included) and then get to training!

When you run the algorithm, you will specify a maximum number of iterations (in case our problem isn't separable). This data will then be used, over the maximum of iterations, to train the perceptron. Each iteration, run your data samples one by one through the training procedure and modify the weights. Repeat this process for the maximum number of iterations. Note you could possibly detect the convergence of your network and end early. Each iteration, then test your network on the data and output an error rate by detecting how many times it does an incorrect classification.

At the end of the training, test this network on all the inputs and again output the error rate.

Once this process is complete, allow me to enter a set of data and get an output classification.

Note that when testing, I suggest to at least try two sets of input data. One that is linearly separable and one that is not. The behavior of your perceptron should be quite different.