**ECE4870/7870 CS 4770/7770 F’18 Computer Assignment 1 Part A Due 9/27/2018**

**Backpropagation Training of a MLP**

This assignment will be given out, and graded, in two separate parts. The idea is that if you can’t get the correct first part, it will guarantee wrong results later. So we will have a check point at the beginning.

Part A:

For this experiment you are to implement a multi-layer perceptron containing a single hidden layer. Download the cross dataset from Canvas. The zip file contains the dataset and initial weights you should use for this experiment. The file “cross\_data (3inputs – 2 outputs).xlsx” contains 314 two-dimensional samples, each with a target values of (0,1) or (1,0). The initial weights and biases for this network are listed in the tables below, and are also given in the files “w1 (3 inputs – 11 nodes).xlsx”, “b1 ((11 nodes).xlsx”, “w2 (from 11 to 2).xlsx”, and “b2 (2 output nodes).xlsx”.

You must write your own code. Do not use any neural network package!

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Hidden layer  weights | | **From Input Node** | |  | | *x1* | *x2* | *x3* | | **To Hidden Node** | *w1(1)* | 0.4033 | -1.0562 | 0.2306 | | *w2(1)* | 0.39 | 0.6544 | -0.9077 | | *w3(1)* | 0.6376 | -0.0601 | -0.833 | | *w4(1)* | 0.0064 | -0.0462 | -0.6638 | | *w5(1)* | 0.0782 | 0.2728 | -0.8454 | | *w6(1)* | -0.2115 | 1.0252 | 0.4156 | | *w7(1)* | 0.7298 | -0.5047 | -0.0307 | | *w8(1)* | -0.7109 | 0.349 | -0.5411 | | *w9(1)* | -0.9315 | 0.9867 | -0.3898 | | *w10(1)* | 0.8441 | 0.4276 | -0.4924 | |  | *w11(1)* | 0.6372 | -0.3299 | 0.6848 | | |  |  | | --- | --- | | Hidden layer biases | | | **Node** | **Bias** | | *b1(1)* | -0.122 | | *b2(1)* | 0.9401 | | *b3(1)* | 0.4271 | | *b4(1)* | -0.1775 | | *b5(1)* | -0.7019 | | *b6(1)* | -0.3326 | | *b7(1)* | -0.6961 | | *b8(1)* | -0.9316 | | *b9(1)* | -0.3681 | | *b10(1)* | 1.0695 | | *b11(1)* | -0.2099 | | |  |  | | --- | --- | | Output biases | | | **Node** | **Bias** | | *b1(2)* | 0.1131 | | *b2(2)* | -0.3824 | |

Remember,  (i = 0 is for bias) and 

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Output layer  weights | | **From Hidden Node** | | | | | | | | | | |  | |
| *y1(1)* | *y2(1)* | *y3(1)* | *y4(1)* | *y5(1)* | *y6(1)* | *y7(1)* | *y8(1)* | *y9(1)* | *y10(1)* | *y11(1)* | |
| **To Output Node** | *w1(2)* | 0.0511 | 0.1611 | 0.0238 | -0.0267 | 0.1089 | 0.2381 | 0.0784 | 0.003 | 0.1646 | -0.1779 | 0.05324 | |
|  | *w2(2)* | -0.3082 | -0.0625 | 0.0129 | -0.1011 | 0.0123 | 0.1153 | 0.2984 | 0.3232 | -0.1580 | -0.0077 | 0.09536 | |

Use the following parameters in your network:

* Sigmoid activation function for all nodes:
* Learning rate:
* Momentum term:

1. Perform **one epoch** of on-line backpropagation training (update weights after each sample) on the cross dataset **in the order provided**. Do not randomize the sample presentation order for the first epoch. List all of the network weight and bias terms after the first epoch using the same table format as used above. Limit your reported precision to 4 decimal places. Also, calculate and report the sum-of-squared-errors of all samples after the first epoch.
2. Print off and attach your code with the report. We will grade this part before proceeding.
3. Here’s part of what the subsequent project (Part B) will contain. If you feel confident in your implementation, you can try this, but don’t include it in the Part A report. Continue to train the network until the change in sum-of-squared-errors is less than 0.001. Randomize the presentation order of the samples for each epoch after the first. Plot the sum-of-squared-errors per epoch. Then test your network in the following way. Sample the square containing the first 2 coordinates (roughly [-2.1, 2.1] x [-2.1, 2.1]) by increments of 0.01 (or 0.001 for finer resolution). Set the 3rd coordinate for all such vectors to some very small random number around 0.0. Create a visual display of the classification labels using colors and/or different symbols for all the points in the square to estimate the decision regions formed by the network on the cross dataset. To get a smoother boundary, you can display only the output from one of the output nodes as a continuous variable between 0 and 1, or the difference between out values. You can use a function to plot 0.5 (or 0.0) isocurves What would happen if you just eliminated the 3rd coordinate for all the training data? Does this dimension help in classification?

**Even though the experiment in Part A deals with a specific 3:11:2 MLP, your program should be general and parameterized to be able to handle up to 20 dimensional input data, 20 hidden neurons, and 10 output neurons.**