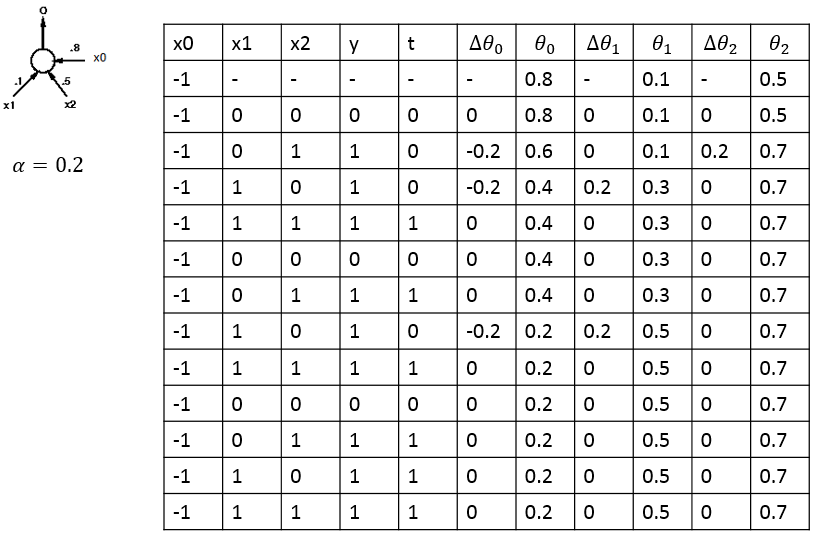
**BENG 420/520: Homework #2 Perceptron**

**Due: Monday, 3/16/2020, by 11:59pm**

*Update: You can work with another student for this assignment. Only one submission is required from each group of TWO. Please specify your partner’s name in the documentation.*

*Instructions: You should submit this assignment as a Word or PDF document on Blackboard. Please also submit your documented Matlab code.*

1. We used the OR gate example in class to understand the perceptron classifier. Using sequential update, the classifier converged after seven steps of updates shown in Table 1. In this assignment, you are asked to apply the batch update algorithm to **determine** the values of the Perceptron weights. Learn the four training data points as many times as needed until the classifier is able to correctly label all of them (i.e. converged). Complete Table 2. You will need to **add** more rows to show all the learning updates.



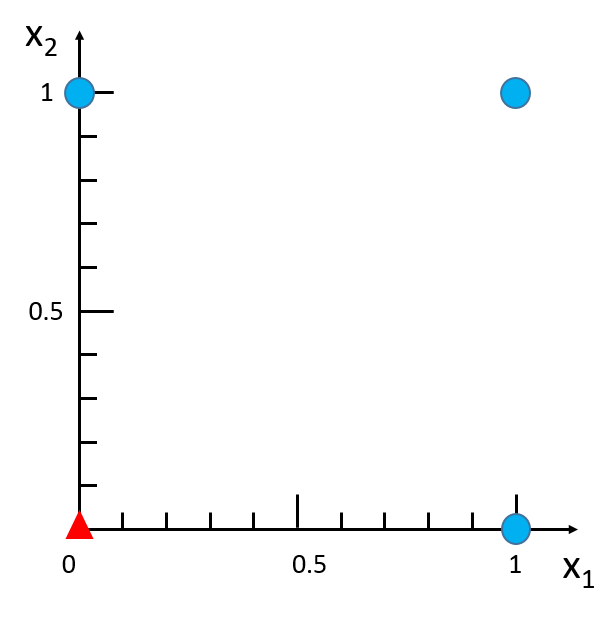
*Table 1: Perceptron weight update table using sequential update*

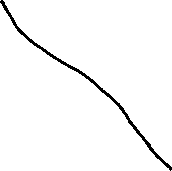
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x0 | x1 | x2 | y | t |  |  |  |  |  |  |
| -1 | - | - | - | - | - | 0.8 | - | 0.1 | - | 0.5 |
| -1 | 0 | 0 | 0 |  |  |  |  |  |  |  |
| -1 | 0 | 1 | 1 |  |  |  |  |  |  |  |
| -1 | 1 | 0 | 1 |  |  |  |  |  |  |  |
| -1 | 1 | 1 | 1 |  |  |  |  |  |  |  |

*Table 2: Perceptron weight update table using batch update*

**Discuss** what you learned from these two perceptron update procedures with this example. Did the two sets of weights converge to the same values? **Draw** the two decision boundaries on the plot blow.

The two update procedures do not necessarily converge to the same values for the weights. The batch update procedure converged after 2 iterations. I believe this means that the batch version updated more rapidly but there were more overall calculations required because each iteration of the batch method accounts for 4 sequential updates.





1. The objective of this question is to enhance your understanding of the **sequential** *delta learning rule* used in Perceptron learning and multi-layer neural network. You are asked to implement the learning and prediction algorithms of (1) a perceptron with threshold transfer function, (2) a logistic neuron and (3) compare their performances. The following Matlab programs are provided:
2. ~~Data generation and plotting routines are given in~~ **~~h2\_neurons.m~~**~~, which is the file you should program to (1) call your perceptron learning function to learn the weights and (2) predict the labels using the learned perceptron. You will need to program at places where there is a “~~**~~YOU~~**~~” indicator in the code comment.~~
3. **~~perceptron\_threshold.m~~**~~: The parameters of the function are defined. You will implement the perceptron learning algorithm with a threshold transfer function here.~~
4. **logistic\_neuron.m**: The parameters of the function are defined. Implement the learning algorithm with a logistic sigmoid transfer function.
5. **~~sig.m~~**~~: Implement a scaled logistic sigmoid function so that the output is between~~ **~~-1 and 1~~**~~.~~
6. **~~dsig.m~~**~~: Implement the derivative of that scaled logistic sigmoid function in sig.m.~~
7. Each time when you run the program, random data points are generated. Use the same set of data to test the perceptron and the logistic neuron, with the same learning rate and sufficiently large number of iterations to make sure both learning processes converge. **Describe** your observation on the accuracy performances of these two neurons on this type of data and **explain** why it occurs. You may want to run it several times confirm your observation. **Include** a snapshot of the plotted results from one run in your report.
8. Note that class labels are +1 and -1, instead of +1 and 0.
9. Submit perceptron\_threshold.m, logistic\_neuron.m, sig.m, dsig.m, and your modified h2\_neurons.m in addition to the **discussion** on your observation. Make sure to **comment** your code.