**College of Computer Science and Engineering**

**Department of Computer Science and Artificial Intelligence**

**CCAI-321: Artificial Neural Networks**

**Lab#7 Implementing Backpropagation using Python**

Student id : 1845919

Marks Obtained = / 15 PLO = S1 - AI

**Marks:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Questions** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **Total** |
| **Allocated** | **2** | **2** | **2** | **2** | **2** | **2** | **3** | **15** |
| **Obtained** | **2** | **2** | **2** | **2** | **2** | **2** | **1.5** | **13.5** |
|  |  |  |  |  |  |  |  |  |
| **Allocated** |  |  |  |  |  |  |  |  |
| **Marks** |  |  |  |  |  |  |  |  |

**Weighted Marks:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Allocated** |  |  |  |
| **Obtained** |  |  |  |

Objectives

* Implement the backpropagation algorithm in python
* Train a 2-layer network (implemented in Lab6) in python

Lab Tool(s)

[Download Python | Python.org](https://www.python.org/downloads/)

[Anaconda | Individual Edition](https://www.anaconda.com/products/individual)

Lab Deliverables

Submit a pdf document on Blackboard containing your solution to the lab assessment at the end of this document.

What is the Backpropagation Algorithm?

Backpropagation is a supervised learning algorithm used to train multilayer neural networks. As we have seen in class, there are three main steps in backpropagation.

**Step1:** Forward Propagation

In this step, we compute *a* for all layers, starting from the **first** layer to the **last**.

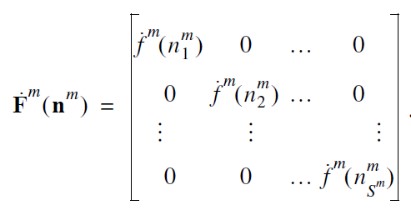
# • 𝑎0 = 𝑝 • 𝑎𝑚+1 = 𝑓𝑚+1(𝑊𝑚+1𝑎𝑚 + 𝑏𝑚+1) 𝑓𝑜𝑟 𝑚 = 0,1, . . 𝑀 − 1 (𝑀 𝑖𝑠 𝑛𝑢𝑚𝑏𝑒𝑟 𝑜𝑓 𝑙𝑎𝑦𝑒𝑟𝑠 𝑖𝑛 𝑛𝑒𝑡𝑤𝑜𝑟𝑘) • 𝑎 = 𝑎𝑀

**Step2:** Backward Propagation

In this step, we compute *s* for all layers, starting from the **last** layer to the **first**. • sM = -2FM (nM) (t-a)

# • 𝑠𝑚 = 𝐹𝑚(𝑛𝑚)(𝑊𝑚+1)𝑇𝑠𝑚+1m = M-1, …..,2,1

Where

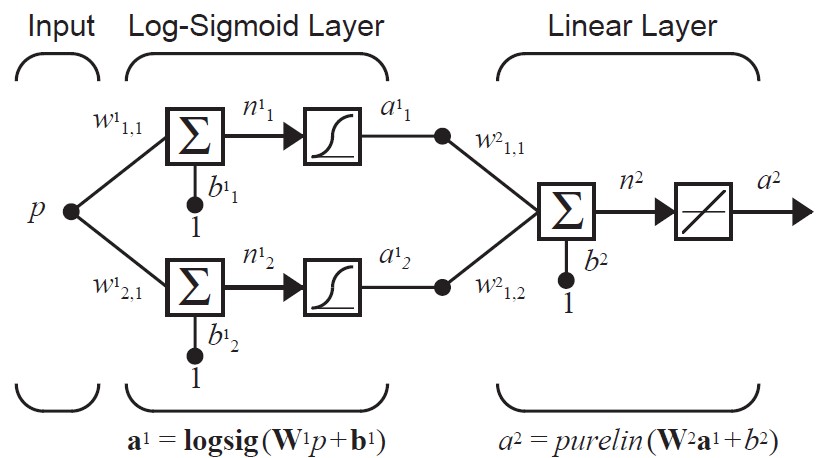


**Step3:** Update W and b

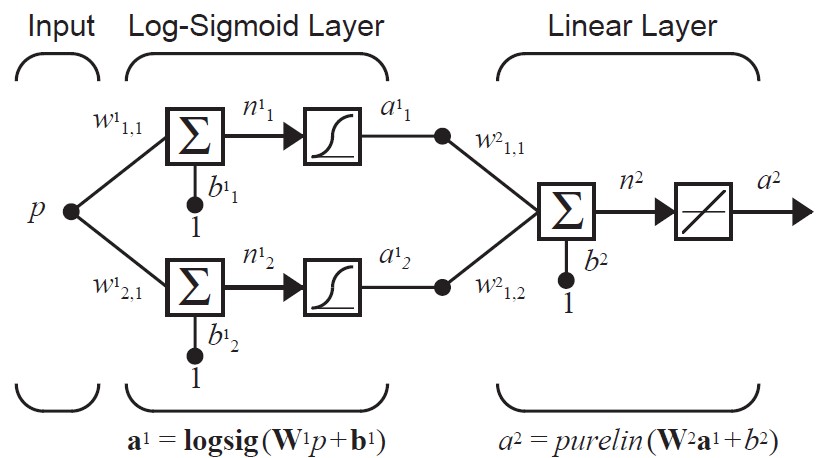
In this step, we update W and b using the values computed in the previous steps.

* 𝑊𝑚(𝑘+1)=𝑊𝑚(𝑘)−𝛼𝑠𝑚(𝑎𝑚−1)𝑇
* 𝑏𝑚(𝑘+1)=𝑏𝑚(𝑘)−𝛼𝑠𝑚

In this lab, we will implement the backpropagation algorithm and use it to train the two layer network that was implemented in the previous lab. We will start by implementing each step of the algorithm separately then combine them, and use it to train the network.



# Q1. Implement the 2-layer network described in the previous lab. The figure below illustrates the network architecture for your reference. To do this task, follow the steps below [2 marks]



1. Implement a function, name it: **logsiglayer.** Use the same function you implemented in the previous lab.
2. Implement a function, name it: **linearlayer.** Use the same function you implemented in the previous lab.

# Q2. Implement the feedforward step. To do so, follow the steps below [2 points]

1. Implement a new function, name it: **forwardpropagation**.
2. The parameters to this function are: p, W1, b1, W2 and b2.
3. This function returns a0, a1 and a2 (a).
   1. a0 = p
   2. a1 is computed by passing (p,W1, and b1) to logsiglayer
   3. a is computed by passing (a1, W2, and b2) to linearlayer

# Q3. Implement the derivatives of the transfer functions for each layer. To do so, follow the steps below [2 points]

1. Implement a new function, name it: **dlogsig**. The parameter to this function is (n). The function computes the value of the derivative for logsig at point n.

Remember: logsig = 1 / (1+ e^(-n))

And the derivative is: (1-a1) a1

1. Implement a new function, name it dpurelin. The parameter to this function is (n). The function computes the value of the dervative for dpurelin at point n. Remember: purelin = n and the derivative is: 1.

# Q4. Implement the backpropagation step. To do so, follow the steps below [2 points]

1. Implement a new function, name it: **backpropagation**.
2. The parameters to this function are: F2, t, a, F1, W2.
3. This function returns s2 and s1.
   1. s2 = -2F2 (n2) (t-a)
   2. 𝑠1 = 𝐹1(𝑛1)(𝑊2)𝑇𝑠2

# Q5. Implement the update step. To do so, follow the steps below [2 points]

1. Implement a new function, name it: **updateparams**.
2. The parameters to this function are: W1, W2, b1, b2, s1, s2, a0, a1, a2, alpha.
3. This function returns updated W and b, given by the formula below

a. 𝑊𝑚 = 𝑊𝑚 − 𝛼𝑠𝑚(𝑎𝑚−1)𝑇 b. 𝑏𝑚 = 𝑏𝑚 − 𝛼𝑠𝑚

# Q6. Combine the 3 steps in one function. To do so, follow the steps below [2 points]

1. Implement a new function, name it: **backpropagatealgorithm**.
2. The parameters to this function are: W1, W2, b1, b2, p, t, alpha.
3. This function returns updated W and b, by calling the 3 steps in order
   1. First call forwardpropagation to get a0, a1 and a2
   2. Then call dlogsig and dpurelin to get the value of the derivatives (F2 and F1). Note: you might use np.diag to convert the values into a diagonal matrix
   3. Then call backpropagation to get s2 and s1
   4. Lastly, update the parameters by calling updateparams which returns the updated parameters

**Q7. Given the data points below, train the neural network function, implemented in Q1, using the backpropagation algorithm. To do so, follow the steps below.**

# [2+1 points]

|  |  |
| --- | --- |
|  |  |
| **p** | **g(p)** |
| - 2 | 0.0 |
| -1.5 | 0.006 |
| -1 | 0.29 |
| -0.5 | 0.617 |
| 0 | 1.0 |
| 0.5 | 1.38 |
| 1 | 1.707 |
| 1.5 | 1.92 |
| 2 | 2.0 |

1. Randomly initialize the parameters: W1, W2, b1, and b2.
2. Randomly select a data point: t and p.
3. Pass the parameters to backpropagatealgorithm to get the updating weights and biases
4. Check manually if the results you got are correct.
5. Repeat steps 2 and 3, until there is no changes in weights and biases.
6. How many iterations did you do? What are the converged values of Ws and bs?

**Extra. How can you implement a neural network in a general way? In which you do not need to write another customized code for a 3-layer, 4-layer, or even more layers.**