**Lab 2 – Neural Network - Perceptron**

***Group members:***

1. *Duc Phi Ngo (Student ID: 616922)*
2. *Huynh Anh Vu Nguyen (Student ID: 618069)*
3. *Thanh Do Nguyen (Student ID: 615941)*
4. *Van Ty Pham (Student ID: 616938)*
5. *Nhien Quang Luu (Student ID: 440386)*

1. Consider a neuron with 2 inputs, 1 output, and a threshold activation function. If the two weights are w1 = 1 and w2 = 1, and the bias is b = −1.5, then what is the output for input (0, 0)? What about for inputs (1, 0), (0, 1), and (1, 1)? Draw the discriminant function for this function, and write down its equation. Does it correspond to any particular logic gate?   (No coding)

A diagram of a circle with arrows and a circle with lines

AI-generated content may be incorrect.

The activation function is a threshold function, meaning:

**y=1** w1\*x1 + w2\*x2 + b >= 0

**y=0** otherwise

The neuron computes the weighted sum:

**z = w1\*x1 + w2\*x2 + b = x1+x2-1.5**

so:

         (0,0) = -1.5 0

         (0,1) = -0.5 0

         (1,0) = -0.5 0

         (1,1) = 0.5 1

This neuron implements the AND gate but with a different threshold

A grid of numbers and symbols

AI-generated content may be incorrect.

The Discriminant Function: **x1+x2 = 1.5**

A graph with blue squares and red line

AI-generated content may be incorrect.

2. Work out the Perceptrons that construct logical NOT, NAND, and NOR of their inputs. (No coding)

**y=1** w1\*x1 + w2\*x2 + b >= 0

**y=0** otherwise

**NOT gate**

**A grid with black squares and black text

AI-generated content may be incorrect.**

The perceptron equation: **y = w1\*x + b**

x=0 output = 1 w1\*0 + b >= 0 b>=0

x=1 output = 0 w1\*1 + b <0   w1+b<0

choosing w1= -1 and b = 0.5

**y = -x + 0.5**

**NAND gate**

**A grid of numbers and letters

AI-generated content may be incorrect.**

The perceptron equation: **y = w1\*x1 + w2\*x2 + b**

(0,0) output=1 b>=0

(0,1) and (1,0) output=1 w1+b>=0

(1,1) output=0 w1+w2+b<0

choosing w1=w2= -1 and b = 1.5

**y = -x1-x2 + 1.5**

**NOR gate**

**A grid of numbers and symbols

AI-generated content may be incorrect.**

The perceptron equation: **y = w1\*x1 + w2\*x2 + b**

(0,0) output=1 b>=0

(0,1) and (1,0) output=0 w1+b<0

(1,1) output=0 w1+w2+b<0

choosing w1=w2= -1 and b = 0.5

**y = -x1-x2 + 0.5**

3. Study Section 3.4.1.  Then prove the same Perceptron Convergence Theorem using ||x|| <= R (instead of using ||x|| <= 1) for some constant R.

A notebook with writing on it

AI-generated content may be incorrect.

4. The Perceptron code on the website (<http://stephenmonika.net/> ) is a batch update algorithm, where the whole of the dataset is fed in to find the errors, and then the weights are updated afterwards, as is discussed in Section 3.3.5. Convert the code to run as sequential updates and then compare the results of using the two versions. (Modify existing Python program )

**Original Batch update code:**

A screen shot of a computer code

AI-generated content may be incorrect.

**Modified Sequential update code:**

A computer screen shot of a program

AI-generated content may be incorrect.

**Key change:**

**Instead of**:

*self.weights -= eta \* np.dot(np.transpose(inputs), (self.activations - targets))*

**Use**:

*for m in range(self.nData):*

*inputs\_seq = np.array([inputs[m]])*

*targets\_seq = np.array([targets[m]])*

*self.activations = self.pcnfwd(inputs\_seq)*

*self.weights -= eta \* np.dot(np.transpose(inputs\_seq), (self.activations - targets\_seq))*

**Batch learning**:

* Process all inputs at once
* More stable updates because updates are based on the entire dataset
* Output:
* Initial weights:
* [[-0.01086659]
* [-0.03923124]
* [ 0.02419855]]
* Iteration 1:
* Updated weights:
* [[ 0.08913341]
* [ 0.06076876]
* [-0.07580145]]
* Iteration 2:
* Updated weights:
* [[-0.01086659]
* [-0.03923124]
* [ 0.22419855]]
* Iteration 3:
* Updated weights:
* [[0.08913341]
* [0.06076876]
* [0.12419855]]
* Iteration 4:
* Updated weights:
* [[0.08913341]
* [0.06076876]
* [0.12419855]]
* Iteration 5:
* Updated weights:
* [[0.08913341]
* [0.06076876]
* [0.12419855]]
* Iteration 6:
* Updated weights:
* [[0.08913341]
* [0.06076876]
* [0.12419855]]
* Iteration 7:
* Updated weights:
* [[0.08913341]
* [0.06076876]
* [0.12419855]]
* Iteration 8:
* Updated weights:
* [[0.08913341]
* [0.06076876]
* [0.12419855]]
* Iteration 9:
* Updated weights:
* [[0.08913341]
* [0.06076876]
* [0.12419855]]
* Iteration 10:
* Updated weights:
* [[0.08913341]
* [0.06076876]
* [0.12419855]]
* Final Outputs:
* [[0]
* [0]
* [0]
* [1]]

**Sequential learning**:

* Update weight after every process input
* More dynamic learning behavior.
* Output
* Initial weights:
* [[ 0.03991313]
* [ 0.01924401]
* [-0.04891021]]
* Iteration 1:
* Step 1, Updated weights:
* [[0.03991313]
* [0.01924401]
* [0.05108979]]
* …
* Iteration 10:
* Step 1, Updated weights:
* [[0.03991313]
* [0.01924401]
* [0.05108979]]
* Step 2, Updated weights:
* [[0.03991313]
* [0.01924401]
* [0.05108979]]
* Step 3, Updated weights:
* [[0.03991313]
* [0.01924401]
* [0.05108979]]
* Step 4, Updated weights:
* [[0.03991313]
* [0.01924401]
* [0.05108979]]
* Final Outputs:
* [[0]
* [0]
* [0]
* [1]]

5. Study Section 3.4.4.  This will give you a good idea about Prime Indian Dataset (<http://archive.ics.uci.edu/ml/datasets.html> - download it), how to plot various parts of it and then using Perceptron to see how well it can classify this data set. You can use the Perceptron code in #4 (or your modified version).  This Section has all the key information so that you can run the Perceptron code on the given dataset.

<https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv>

A screenshot of a computer screen

AI-generated content may be incorrect.

<https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv>

A screen shot of a graph

AI-generated content may be incorrect.

**Run the Perceptron code, analyze the results and write your observation. Show the results – you can submit a plot of the results.**