Neural Networks and Deep Learning – Summer Term 2019

Team - 09	
Name: Rajib Chandra Das Matriculation Number: 1140657 Email: stu218517@mail.uni-kiel.de	Name: M M Mahmudul Hassan Matriculation Number: 1140658 Email: stu218518@mail.uni-kiel.de

Exercise sheet 2

Exercise 1 (Single-layer perceptron and Boolean functions with 2 inputs):

a) We have to show that, the Boolean function XOR cannot be realized by a (single-layer) perceptron.

We have truth tables for XOR:

x1	x2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

We have to calculate output of the neuron using this equation: y = Θ [x1 * w1 + x2 * w2 – θ]

Where,

- Θ: Heaviside Function
- w1, w2: Weights
- θ: Threshold
- x1, x2: Inputs

For x1 = 0 and x2 = 0 output is 0. Therefore, we need: 0 * w1 + 0 * w2 < θ => 0 < θ ------(1)

For x1 = 0 and x2 = 1 output is 1. Therefore, we need: 0 * w1 + 1 * w2 >= θ => w2 >= θ----(2)

For x1 = 1 and x2 = 0 output is 1. Therefore, we need: 1 * w1 + 0 * w2 >= θ => w1 >= θ ----(3)

For x1 = 1 and x2 = 1 output is 0. Therefore, we need: 1 * w1 + 1 * w2 < θ => w1 + w2 < θ----(4)

From equation (1), we have θ is a positive value. From equation (2) and (3) we have w1, w2 are also positive values as well as both weights are greater or equal to threshold. From equation (4) we find a contradiction. Because the summation of both weights cannot be smaller than threshold. Therefore, it's impossible to define values for weights and threshold to satisfy all 4 equations.
So, the Boolean function XOR cannot be realized by a (single-layer) perceptron.

Reference: https://computing.dcu.ie/~humphrys/Notes/Neural/single.neural.html

b) We have to define all Boolean functions with 2 inputs and indicate whether they can be realized by a (single-layer) perceptron. We can consider following facts for this problem.

- For 2 inputs we have 4 different input combinations.
- For 4 different input combinations we have 2ˆ4 = 16 different Boolean functions.

i.

x1	x2	F0 = FALSE	Indication
0	0	0	Possible
0	1	0	
1	0	0	
1	1	0	

ii.

x1	x2	F1 = AND	Indication
0	0	0	Possible
0	1	0	
1	0	0	
1	1	1	

iii.

x1	x2	F2 = x1 ∧ ¬x2	Indication
0	0	0	Possible
0	1	0	
1	0	1	
1	1	0	

iv.

x1	x2	F3 = x1	Indication
0	0	0	Possible
0	1	0	
1	0	1	
1	1	1	

v.

x1	x2	F4 = ¬ x1 ∧ x2	Indication
0	0	0	Possible
0	1	1	
1	0	0	
1	1	0	

vi.

x1	x2	F5 = x2	Indication
0	0	0	Possible
0	1	1	
1	0	0	
1	1	1	

vii.

x1	x2	F6 = XOR	Indication
0	0	0	**Not Possible**
0	1	1	
1	0	1	
1	1	0	

viii.

x1	x2	F7 = OR	Indication
0	0	0	Possible
0	1	1	
1	0	1	
1	1	1	

ix.

x1	x2	F8 = NOR	Indication
0	0	1	Possible
0	1	0	
1	0	0	
1	1	0	

x.

x1	x2	F9 = XNOR	Indication
0	0	1	**Not Possible**
0	1	0	
1	0	0	
1	1	1	

xi.

x1	x2	F10 = ¬ x2	Indication
0	0	1	Possible
0	1	0	
1	0	1	
1	1	0	

xii.

x1	x2	F11 = x1 V ¬x2	Indication
0	0	1	Possible
0	1	0	
1	0	1	
1	1	1	

xiii.

x1	x2	F12 = ¬x1	Indication
0	0	1	Possible
0	1	1	
1	0	0	
1	1	0	

xiv.

x1	x2	F13 = ¬x1 V x2	Indication
0	0	1	Possible
0	1	1	
1	0	0	
1	1	1	

xv.

x1	x2	F14 = NAND	Indication
0	0	1	Possible
0	1	1	
1	0	1	
1	1	0	

xvi.

x1	x2	F14 = TRUE	Indication
0	0	1	Possible
0	1	1	
1	0	1	
1	1	1	

Reference: http://mathworld.wolfram.com/BooleanFunction.html

c) We have to give values for synaptic weights w1, w2 and threshold θ for three different Boolean functions:

i.

x1	x2	F1 = AND	Indication
0	0	0	Possible
0	1	0	
1	0	0	
1	1	1	

We have to calculate output of the neuron using this equation: $y = \Theta\,[x1 * w1 + x2 * w2 - \theta]$

Let's assume $w1 = 1.0$, $w2 = 1.0$, $\theta = 1.5$

For x1 = 0 and x2 = 0 output is 0. We have: $0 * 1.0 + 0 * 1.0 - 1.5 = -1.5$. Therefore, $\Theta[-1.5] = 0$

For x1 = 0 and x2 = 1 output is 0. We have: $0 * 1.0 + 1 * 1.0 - 1.5 = -0.5$. Therefore, $\Theta[-0.5] = 0$

For x1 = 1 and x2 = 0 output is 0. We have: $1 * 1.0 + 0 * 1.0 - 1.5 = -0.5$. Therefore, $\Theta[-0.5] = 0$

For x1 = 1 and x2 = 1 output is 1. We have: $1 * 1.0 + 1 * 1.0 - 1.5 = 0.5$. Therefore, $\Theta[0.5] = 1$

So, AND function is realized by a single-layer perceptron.

ii.

x1	x2	F8 = NOR	Indication
0	0	1	Possible
0	1	0	
1	0	0	
1	1	0	

We have to calculate output of the neuron using this equation: $y = \Theta [x1 * w1 + x2 * w2 – \theta]$

Let's assume w1 = -1.0, w2 = -1.0, θ = -0.5

For x1 = 0 and x2 = 0 output is 0. We have: 0 * -1.0 + 0 * -1.0 + 0.5 = 0.5. Therefore, Θ[0.5] = 1

For x1 = 0 and x2 = 1 output is 0. We have: 0 * -1.0 + 1 * -1.0 + 0.5 = -0.5. Therefore, Θ[-0.5] = 0

For x1 = 1 and x2 = 0 output is 0. We have: 1 * -1.0 + 0 * -1.0 + 0.5 = -0.5. Therefore, Θ[-0.5] = 0

For x1 = 1 and x2 = 1 output is 1. We have: 1 * -1.0 + 1 * -1.0 + 0.5 = -1.5. Therefore, Θ[-1.5] = 0

So, NOR function is realized by a single-layer perceptron.

iii.

x1	x2	F7 = OR	Indication
0	0	0	Possible
0	1	1	
1	0	1	
1	1	1	

We have to calculate output of the neuron using this equation: $y = \Theta [x1 * w1 + x2 * w2 – \theta]$

Let's assume w1 = 1.0, w2 = 1.0, θ = 0.5

For x1 = 0 and x2 = 0 output is 0. We have: 0 * 1.0 + 0 * 1.0 - 0.5 = -0.5. Therefore, Θ[-0.5] = 0

For x1 = 0 and x2 = 1 output is 0. We have: 0 * 1.0 + 1 * 1.0 - 0.5 = 0.5. Therefore, Θ[0.5] = 1

For x1 = 1 and x2 = 0 output is 0. We have: 1 * 1.0 + 0 * 1.0 - 0.5 = 0.5. Therefore, Θ[0.5] = 1

For x1 = 1 and x2 = 1 output is 1. We have: 1 * 1.0 + 1 * 1.0 - 0.5 = 1.5. Therefore, Θ[1.5] = 1

So, OR function is realized by a single-layer perceptron.

d) We know from the definition that, Single-layer perceptron linearly separates the input space into 2 regions.

i. 1st Geometrical Representation: This Graph Linearly Separates into two different regions.

From the graph, we find the co-ordinates for the straight line,

x1	-0.5	0	1
x2	0	1	3

Therefore, the line equation: $x2 = 2 * x1 + 1 \Rightarrow 2 * x1 - x2 + 1 = 0$

So, this Geometrical Representation can be realized by a single-layer perceptron when:

$w1 = 2.0$, $w2 = -1.0$ and $\theta = -1$

ii. 2nd Geometrical Representation: As it is not straight line, this Geometrical Representation cannot be realized by a single-layer perceptron

iii. 3rd Geometrical Representation: As it is not straight line, this Geometrical Representation cannot be realized by a single-layer perceptron

iv. 4th Geometrical Representation: This graph is as similar as Boolean TRUE function. For any input for x1 and x2 we will get output from neuron. So, this Geometrical Representation can be realized by a single-layer perceptron when: $w1 = 0.0$, $w2 = 0.0$ and $\theta = 0.0$

v. 5th Geometrical Representation: This Graph Linearly Separates into two different regions.

From the graph, we find the co-ordinates for the straight line,

x1	1.0	1.0	1.0	1.0	1.0
x2	-1	0	1	2	3

Therefore, the line equation: $x1 = 1.0 \Rightarrow x1 + 0 * x2 - 1.0 = 0$

So, this Geometrical Representation can be realized by a single-layer perceptron when:

$w1 = 1.0$, $w2 = 0.0$ and $\theta = 1.0$

vi. 6th Geometrical Representation: As it is not straight line, this Geometrical Representation cannot be realized by a single-layer perceptron.

Exercise 2 (Types of neural networks, synaptic weight matrix):

a) Here Explanation is given for the following terms related to neural networks:

i. Boolean Function: In mathematics and logic, a Boolean function is a function of the form $f : \{0,1\}^k \rightarrow \{0,1\}$ where k is a non-negative integer called the arity of the function.

ii. Feedforward neural network: this artificial neural network has following properties:
   - Uni-directional data flow.
   - Layer structure: connections only from lower to higher layers.

iii. Recurrent neural network: This artificial neural network has following properties:
   - Bi-directional data flow.

- Any connections possible, with/ without layers.

iv. Multi-layer perceptron: A multilayer perceptron (MLP) is a class of feedforward artificial neural network. A Multi-Layer-Perceptron consists of at least three layers of nodes: an input layer, a hidden layer and an output layer.

b)

i. It's a Feedforward neural network. Because there is no existence of feedback loops and neurons are connected in the formation of layers to layers.

ii. It's a Recurrent neural Network. Because there are some feedback loops like 1 -> 4 -> 7 -> 1, 1 -> 3 -> 7 -> 1, 2 -> 4 -> 6 -> 3 -> 2 etc.

c)

i.

	1	2	3	4	5	6	7
1							
2							
3	✓						
4	✓	✓					
5	✓	✓					
6				✓	✓		
7			✓	✓	✓		

ii.

	1	2	3	4	5	6	7
1							✓
2			✓				
3	✓					✓	
4	✓	✓					
5	✓	✓					
6				✓	✓		
7			✓	✓	✓		

Exercise 3 (Computing the output of a feedforward neural network):

a) Let's assume,

Inputs: x[i],

Output: y[i],

Weights: w[i][j],

State of neurons: s[i] and

Threshold: θ.

Computing 1st layer:

We are given, x[1] = 3 , x[2] =1, w[1][0] = 1, w[2][0] = 1.

Neurons in this layer have linear activation function with slope c = 1.

s[1] = (3 * 1) = 3

s[2] = (1 * 1) = 1

Computing 2nd layer:

We are given,

w[3][1] = 1, w[3][2] = -2,

w[4][1] = -1, w[4][2] = 0,

w[5][1] = 3, w[5][2] = 2,

w[6][1] = 0, w[6][2] = 2 and

Neurons in this layer are threshold element with θ = 0.

s[3] = Θ (3 * 1 + 1 * (-2) - 0) = Θ(1) = 1

s[4] = Θ (3 * (-1) + 1 * 0 - 0) = Θ(-3) = 0

s[5] = Θ (3 * (3) + 1 * 2 - 0) = Θ(11) = 1

s[6] = Θ (3 * 0 + 1 * 2 - 0) = Θ(2) = 1

Computing 3rd layer:

We are given,

w[7][3] = 0, w[7][4] = 2, w[7][5] = -3, w[7][6] = 1, w[8][3] = 1, w[8][4] = -2, w[8][5] = 3, w[8][6] = 8,

w[9][3] = 0, w[9][4] = 2, w[9][5] = 3, w[9][6] = -4,

Neurons in this layer have linear activation function with slope c = 1.

s[7] = (1 * 0 + 0 * 2 + 1 * (-3) + 1 * 1 - 0) = (-2)

s[8] = (1 * 1 + 0 * (-2) + 1 * 3 + 1 * 8 - 0) = (12)

s[9] = (1 * 0 + 0 * 2 + 1 * 3 + 1 * (-4) - 0) = (-1)

Therefore, we can get the final output,

y[1] = -2

y[2] = 12

y[3] = -1

From this solution we are sure that,

i. Neuron 1 and 2 can be computed in parallel.
ii. Neuron 3, 4, 5 and 6 can be computed in parallel.
iii. Neuron 7, 8 and 9 can be computed in parallel.
iv. Computation of neurons in a layer have to wait until the computation of the previous layer.

b)

Firstly x1 = 1, x2 = 0, x3 = 1

Computing 1st layer:

w[4][1] = -2 , w[4][2] = 5, w[4][3] = -4

w[5][1] = 1 , w[5][2] = -2

w[6][1] = 3, w[6][2] = -1, w[6][3] = 6

w[7][2] = 7, w[7][3] = 1

θ [4] = 1.5, θ[5] = 0.5, θ[6] = -0.5, θ[7] = 0.5

x4 = Θ (1 * (-2) + 0 * 5 + 1 * (-4) - 1.5) = Θ(-7.5) = 0

x5= Θ (1 * 1 + 0 * (-2) - 0.5) = Θ(0.5) = 1

x6 = Θ (1 * 3 + 0 * (-1) + 1 * 6 + 0.5) = Θ(9.5) = 1

x7 = Θ (0 * 7 + 1 * 1 – 0.5) = Θ(0.5) = 1

Computing 2nd layer:

w[8][4] = -1 , w[8][5] = 4, w[8][6] = -2

w[9][5] = -3 , w[9][6] = 5, w[9][7] = 1

w[10][4] = 8, w[6][5] = 2, w[6][7] = -3

θ [8] = -0.5, θ[9] = 1.5, θ[10] = -1

Neurons in this layer have linear activation function with slope c = 1.

x[8] = 0 * (-1) + 1 * 4 + 1 * (-2) + 0.5 = 2.5

x[9] = 1 * (-3) + 1 * 5 + 1 * 1 – 1.5= 1.5

x[10] = 0 * 8 + 1 * 2 + 1 * (-3) + 1= 0

Computing 3rd layer:

w[11][7] = 6, w[11][8] = 1, w[11][9] = -2

w[12][6] = 1, w[11][9] = -4, w[11][10] = 3

θ [11] = -1.5, θ[12] = 1.5

X[11] = Θ (1 * 6 + 2.5 * 1 + 1.5 * (-2) + 1.5) = Θ(7) = 1

X[12] = Θ (1 * 1 + 1.5 * (-4) + 0 * 3 - 1.5) = Θ(-6.5) = 0

Therefore, output is x[11] = 1 and x[12] = 0

Secondly x1 = 0, x2 = 1, x3 = 1

Computing 1st layer:

w[4][1] = -2 , w[4][2] = 5, w[4][3] = -4

w[5][1] = 1 , w[5][2] = -2

w[6][1] = 3, w[6][2] = -1, w[6][3] = 6

w[7][2] = 7, w[7][3] = 1

θ [4] = 1.5, θ[5] = 0.5, θ[6] = -0.5, θ[7] = 0.5

x4 = Θ (0 * (-2) + 1 * 5 + 1 * (-4) - 1.5) = Θ(-0.5) = 0

x5= Θ (0 * 1 + 1 * (-2) - 0.5) = Θ(-2.5) = 0

x6 = Θ (0 * 3 + 1 * (-1) + 1 * 6 + 0.5) = Θ(5.5) = 1

x7 = Θ (1 * 7 + 1 * 1 – 0.5) = Θ(7.5) = 1

Computing 2nd layer:

w[8][4] = -1 , w[8][5] = 4, w[8][6] = -2

w[9][5] = -3 , w[9][6] = 5, w[9][7] = 1

w[10][4] = 8, w[6][5] = 2, w[6][7] = -3

θ [8] = -0.5, θ[9] = 1.5, θ[10] = -1

Neurons in this layer have linear activation function with slope c = 1.

x[8] = 0 * (-1) + 0 * 4 + 1 * (-2) + 0.5 = -1.5

x[9] = 0 * (-3) + 1 * 5 + 1 * 1 – 1.5= 4.5

x[10] = 0 * 8 + 0 * 2 + 1 * (-3) + 1= -2

Computing 3rd layer:

w[11][7] = 6, w[11][8] = 1, w[11][9] = -2

w[12][6] = 1, w[11][9] = -4, w[11][10] = 3

θ [11] = -1.5, θ[12] = 1.5

X[11] = Θ (1 * 6 + (-1.5) * 1 + 4.5 * (-2) + 1.5) = Θ(-3) = 0

X[12] = Θ (1 * 1 + 4.5 * (-4) + (-2) * 3 - 1.5) = Θ(-24.5) = 0

Therefore, output is x[11] = 0 and x[12] = 0

Reference: https://brilliant.org/wiki/feedforward-neural-networks/

Exercise 4 (Multi-layer perceptron and XOR):

a) Let's consider following Boolean operations.

x1	x2	XOR	x1 ∨ x2	x1 ∧ x2	¬ (x1 ∧ x2)	¬ (x1 ∧ x2) ∧ (x1 ∨ x2)
0	0	0	0	0	1	0
0	1	1	1	0	1	1
1	0	1	1	0	1	1
1	1	0	1	1	0	0

So, we are sure that XOR function can be represented by Boolean function ¬ (x1 ∧ x2) ∧ (x1 ∨ x2)

Now,

x1 and x2 is the input,

We need one perceptron in 2nd layer x3 for ¬ (x1 ∧ x2) which is basically NAND function.

We need one perceptron in 2nd layer x4 for (x1 ∨ x2) which is basically OR function.

We need one perceptron in 3rd layer x5 for ¬ (x1 ∧ x2) ∧ (x1 ∨ x2) which is basically AND function.

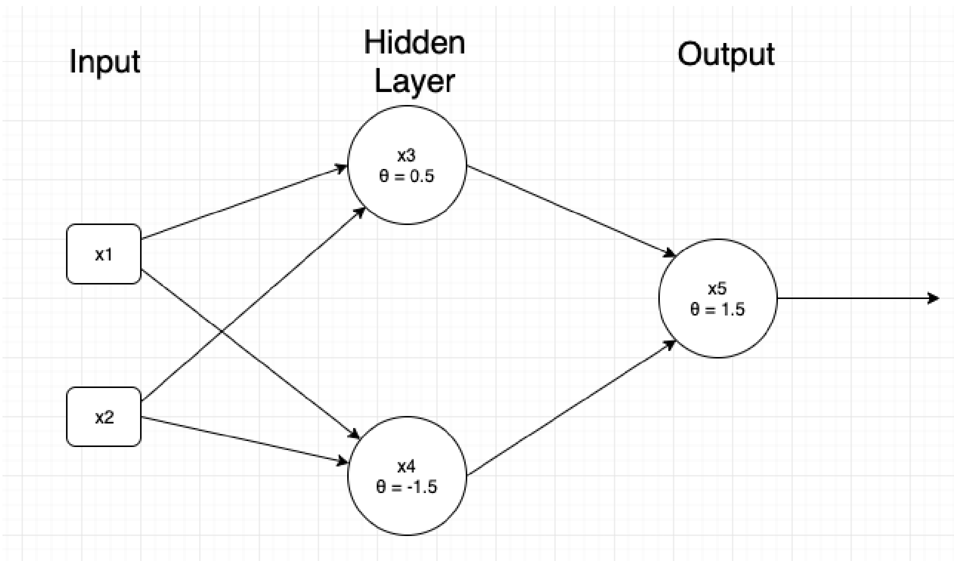

Figure: Multilayer Perceptron for XOR

And here is the weight matrix:

	1	2	3	4	5
1					
2					
3	1	1			
4	-1	-1			
5			1	1	

Now let's try to prove this multilayer perceptron for XOR function.

$w[3][1] = 1$, $w[3][2] = 1$

$w[4][1] = -1$, $w[4][2] = -1$

$w[5][3] = 1$, $w[5][4] = 1$

$\theta[3] = 0.5$, $\theta[4] = -1.5$, $\theta[5] = 1.5$

Firstly x1 = 0, x2 = 0

Computing 2nd Layer:

$x[3] = \Theta(0 * 1 + 0 * 1 - 0.5) = \Theta(-0.5) = 0$

$x[4] = \Theta(0 * (-1) + 0 * (-1) + 1.5) = \Theta(1.5) = 1$

Computing 3rd Layer:

$x[5] = \Theta(0 * 1 + 1 * 1 - 1.5) = \Theta(-0.5) = 0$

Therefore, for x1 = 0 and x2 = 0 output is 0 which is matched with XOR function.

Secondly x1 = 0, x2 = 1

Computing 2nd Layer:

$x[3] = \Theta(0 * 1 + 1 * 1 - 0.5) = \Theta(0.5) = 1$

$x[4] = \Theta(0 * (-1) + 1 * (-1) + 1.5) = \Theta(0.5) = 1$

Computing 3rd Layer:

$x[5] = \Theta(1 * 1 + 1 * 1 - 1.5) = \Theta(0.5) = 1$

Therefore, for x1 = 0 and x2 = 1 output is 1 which is also matched with XOR function.

Thirdly x1 = 1, x2 = 0

Computing 2nd Layer:

$x[3] = \Theta(1 * 1 + 0 * 1 - 0.5) = \Theta(0.5) = 1$

x[4] = Θ (1 * (-1) + 0 * (-1) + 1.5) = Θ(0.5) = 1

Computing 3rd Layer:

x[5] = Θ (1 * 1 + 1 * 1 - 1.5) = Θ(0.5) = 1

Therefore, for x1 = 1 and x2 = 0 output is 1 which is also matched with XOR function.

Fourthly, x1 = 1, x2 = 1

Computing 2nd Layer:

x[3] = Θ (1 * 1 + 1 * 1 - 0.5) = Θ(1.5) = 1

x[4] = Θ (1 * (-1) + 1 * (-1) + 1.5) = Θ(-0.5) = 0

Computing 3rd Layer:

x[5] = Θ (1 * 1 + 0 * 1 - 1.5) = Θ(-0.5) = 0

Therefore, for x1 = 1 and x2 = 1 output is 0 which is also matched with XOR function.

So, the output of this multilayer perceptron is matched with XOR function for every combination of input.

b) F(x1, x2) = if (x1 + x2) == 1 then 1 else 0

Here x1 and x2 is two binary input. Let's get output for different input combination.

x1	x2	x1 + x2	F(x1, x2)
0	0	0	0
0	1	1	1
1	0	1	1
1	1	2	0

This function is working exactly as XOR function. We have already defined a multilayer perceptron for XOR function in part a.