



Assignment 1  
Computational Intelligence  
Dr. S. Hajipour

Mohammad Hossein Shafizadegan  
99104781

October 26, 2023

## Contents

1	Question 1	2
2	Question 2	2
3	Question 3	3
4	Question 4	5
5	Question 5	6
6	Question 6	7
7	Question 7	8
8	Question 8	10
9	Question 9	10
10	Question 10	10
11	Question 11	11

## 1 Question 1

**a**

We can define each pixel of the input image as an input for our neural network. Therefore, the total number of input neurons for our network equals **100**.

**b**

The number of valid classes that can be assigned as output is **10** (as for each number from 0 to 9).

If we code the outputs using one-hot encoding, we will need **10 neurons** for output (for each output class only one neuron has output value of 1).

It is also possible to use simple binary coding for output classes using 4 bits. In this case we only need **4 output neurons**.

**c**

The number and size of hidden layers depend on the complexity and non-linearity of the problem. We note that if the number of samples and test data are relatively small it's not recommended to have several hidden layers since the convergence won't be achieved. The more hidden layer we have, the more complexity we deal with. Also note that here the inputs can be sparse as some pixels of hand written numbers in the corners are the same for all inputs. The concept of dimensionality reduction can be seen here.

**d**

In this case, each output of our neural network is value of a particular pixel. The range of this input value is quite vast from 0 to 255. Normalizing input values can make the optimization process faster and more efficient, by preventing the gradients from becoming too large or too small, and by making the error surface smoother and more symmetric.

There are several ways to normalize the input. One simple way is called the min-max normalization which transforms the input range from 0 to 255 into 0 to 1. This process can be done by dividing the input by its maximum value.

The other popular method of normalizing is z-score which transfer in a way that the average is equal to 0 and the standard deviation will be 1. This can be achieved by subtracting each data from the average and dividing by the standard deviation.

$$\text{min-max normalization : } \frac{x}{\max\{x\}} \quad , \quad \text{z-score : } \frac{x - \mu}{\sigma}$$

## 2 Question 2

**a**

We can use each bit of the input sequence as an input for our neural network hence, we will need **5 input neurons**. We want to see at the end whether the hamming distance is greater than 2 or not so only **one output neuron** is enough which produce a binary value indicating the result.

**b**

A linearly separable function is one that can be computed by a single TLU by dividing the input space into two regions with a hyperplane.

Although the hamming code can be calculated using XOR which is not linearly separable, in this problem the second code is fixed so there is no need for XOR. In the following section we will show why this problem is linearly separable by providing a TLU as a proof.

**c**

We assume there is a TLU with weights of  $w_i$  and threshold of  $\theta$  that can solve this problem.

$$\text{if } x = 10101 \Rightarrow \sum_{i=1}^5 w_i x_i = w_1 + w_3 + w_5 \geq \theta, \quad \text{if } x = 10100 \Rightarrow \sum_{i=1}^5 w_i x_i = w_1 + w_3 \geq \theta$$

$$\text{if } x = 10001 \Rightarrow \sum_{i=1}^5 w_i x_i = w_1 + w_5 \geq \theta, \quad \text{if } x = 00101 \Rightarrow \sum_{i=1}^5 w_i x_i = w_3 + w_5 \geq \theta$$

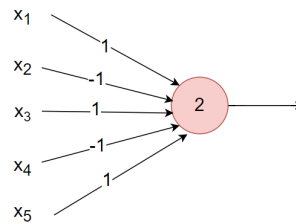
$$\text{if } x \text{ has only one "1"} \Rightarrow \sum_{i=1}^5 w_i x_i = w_i \geq \theta, \quad \text{if } x = 11111 \Rightarrow \sum_{i=1}^5 w_i x_i \geq \theta$$

$$\text{if } x = 01010 \Rightarrow \sum_{i=1}^5 w_i x_i = w_2 + w_4 < \theta$$

assume that  $\theta = 2$  based on the above inequalities we set the following values for  $w_i$  :

$$w_1 = w_3 = w_5 = 1, \quad w_2 = w_4 = -1$$

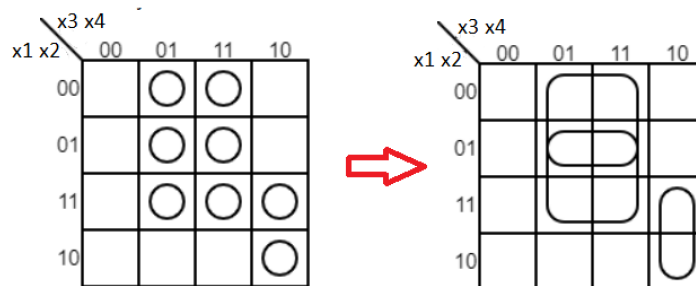
we have checked all possible inputs for this network and it can be seen that this TLU works properly.

**d**

The error will occur when the hamming distance of the sample is greater than or equal to 2 but our network's output is 1. The error for the train data can be defined as difference of the hamming distance in comparison to 2. We want to keep this distance below 2.

### 3 Question 3

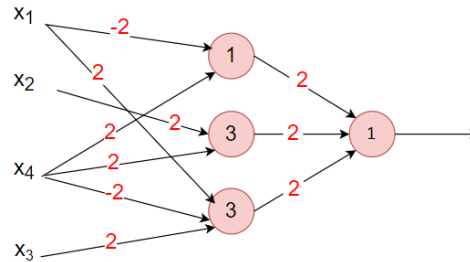
We first create the karnaugh map for this problem and simplify it as follows :



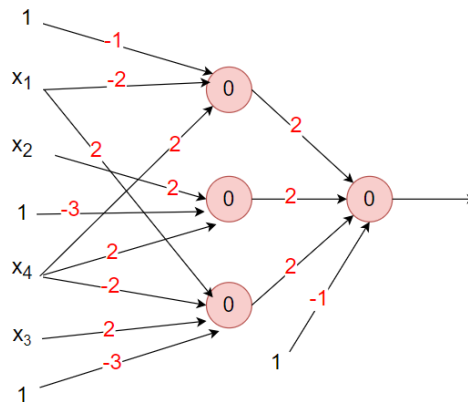
$$y = \overline{x_1} x_4 + x_2 x_4 + x_1 x_3 \overline{x_4}$$

**a**

$$\text{AND : } w_i = \begin{cases} 2 & \text{if } l_i = x_i \\ -2 & \text{if } l_i = \overline{x_i} \end{cases}, \quad \theta = n - 1 + \frac{1}{2} \sum_{i=1}^n w_i, \quad \text{OR : } w_i = 2, \quad \theta = 1$$

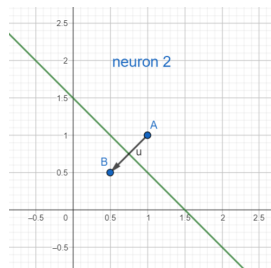
**b**

We set the thresholds to 0 by adding another weigh for the neurons with value of  $-\theta$  and fixed corresponding input of 1.

**c**

$x_1$	$x_2$	$x_3$	$x_4$	$o_1$	$x_2$	$o_3$	$y$
0	0.5	0	0.5	0	0	0	0

Based on the above table and following figure, there previous network will have error for this specific input.

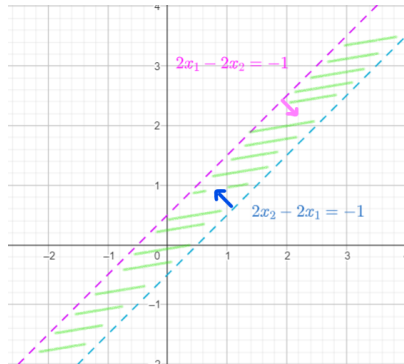


## 4 Question 4

a

$$y = \begin{cases} 1 & 2o_1 + 2o_2 \geq 3 \\ 0 & O.W. \end{cases}, \quad o_1, o_2 \in \{0, 1\} \Rightarrow \text{if } y = 1 \Rightarrow o_1 = o_2 = 1$$

$$\Rightarrow \begin{cases} o_1 = 1 \Rightarrow 2x_2 - 2x_1 \geq -1 \\ o_2 = 1 \Rightarrow 2x_1 - 2x_2 \geq -1 \end{cases} \Rightarrow \text{if } x \in (2x_2 - 2x_1 \geq -1) \cap (2x_1 - 2x_2 \geq -1) \Rightarrow y = 1$$



b

based on the values of weights and thresholds we first create the truth table as follows :

$x_1$	$x_2$	$o_1$	$o_2$	$y$
0	0	1	1	1
0	1	1	0	0
1	1	1	1	1
1	0	0	1	0

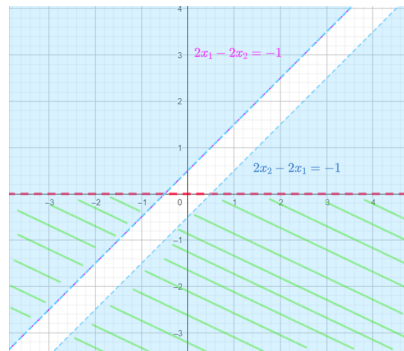
Using SOP format  $\Rightarrow y = \overline{x_1} \overline{x_2} + x_1 x_2$

It can be seen that this network implements **XNOR** logical function

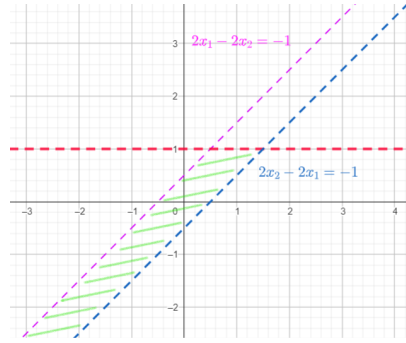
c

$$z = \begin{cases} 1 & y - x_2 \geq 0 \\ 0 & O.W. \end{cases} \Rightarrow x_2 \leq y$$

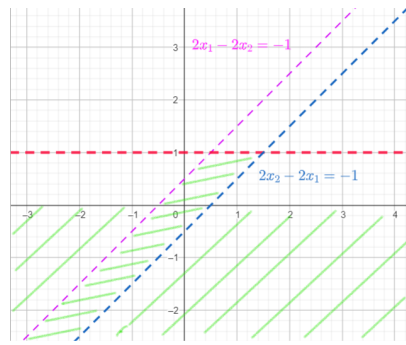
$$\text{if } y = 0 \Rightarrow x_2 \leq 0, \quad x \in (2x_2 - 2x_1 < -1) \cup (2x_1 - 2x_2 < -1)$$



if  $y = 1 \Rightarrow x_2 \leq 1$  ,  $x \in (2x_2 - 2x_1 \geq -1) \cap (2x_1 - 2x_2 \geq -1)$

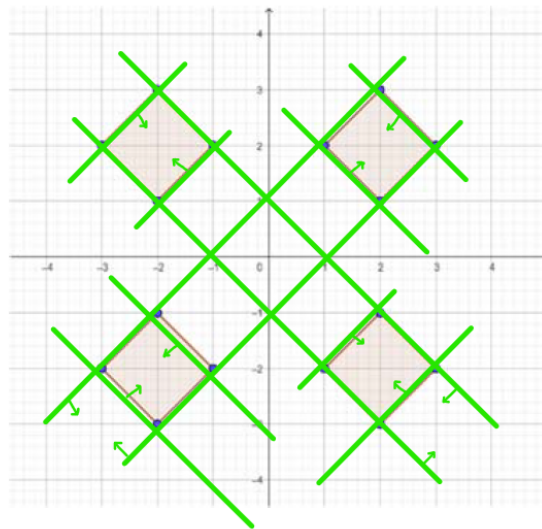


The final answer is aggregation of the above two answers.



## 5 Question 5

a



Regarding the above figure, we need 12 TLUs to create each of these lines. Then we need 4 more neurons to perform the AND process to generate each squares. Finally one more TLU is needed to aggregate these 4 sections. The final

result for the number of required TLUs is **17**

TLUs :  $u_1 : -x_1 - x_2 \geq -1$  ,  $u_2 : x_1 + x_2 \geq -1$  ,  $u_3 : -x_1 - x_2 \geq 3$   
 $u_4 : x_1 + x_2 \geq -5$  ,  $u_5 : x_1 + x_2 \geq 3$  ,  $u_6 : -x_1 - x_2 \geq -5$   
 $u_7 : x_1 - x_2 \geq -5$  ,  $u_8 : -x_1 + x_2 \geq 3$  ,  $u_9 : x_1 - x_2 \geq -1$   
 $u_{10} : -x_1 + x_2 \geq -1$  ,  $u_{11} : x_1 - x_2 \geq 3$  ,  $u_{12} : -x_1 + x_2 \geq -5$   
 $u_{13}, u_{14}, u_{15}, u_{16} : \text{AND TLUs} \Rightarrow w_1 = \dots = w_4 = 2$  ,  $\theta = 7$   
 $u_{17} : \text{OR TLU} \Rightarrow w_1 = \dots = w_4 = 2$  ,  $\theta = 1$

$\text{Ancestor}(u_{13}) = \{u_{11}, u_{12}, u_1, u_2\}$  ,  $\text{Ancestor}(u_{14}) = \{u_9, u_{10}, u_5, u_6\}$   
 $\text{Ancestor}(u_{15}) = \{u_1, u_2, u_8, u_7\}$  ,  $\text{Ancestor}(u_{16}) = \{u_9, u_{10}, u_3, u_4\}$   
 $\text{Ancestor}(u_{17}) = \{u_{13}, u_{14}, u_{15}, u_{16}\}$

.

**b**

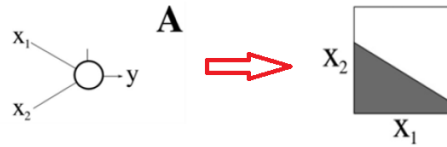
Here we need to rotate and move some of the lines in the previous section by changing the weights and thresholds of the TLUs. Also 4 input TLUs will be removed by setting their weights to 0. The final layer for OR operation is not required anymore.

**c**

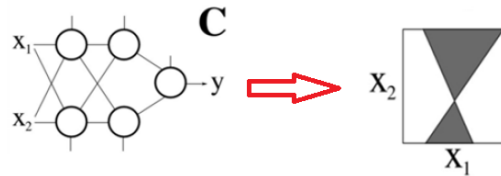
In the general case, we need some TLUs to create the lines used for classification in one layer. Then in the following layer we will need some other TLUs to perform the AND operation and generate each closed section. Since the general case can consist of several closed sections (like in section a) we will need another layer to aggregate these section. Therefor, in the general case will require 3 layers.

## 6 Question 6

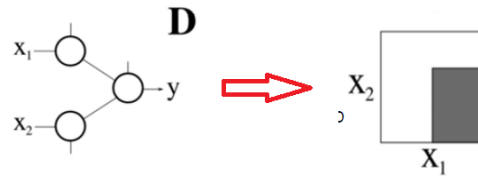
The following neural network consists of only one TLU therefor, it can separate the input space via a straight line into two sections.



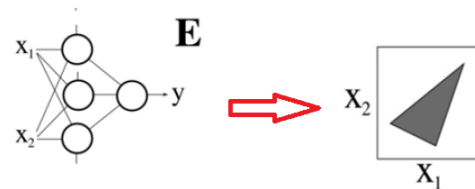
The neural network C, first creates two lines. Each one choose a different section of the space. Then each neuron of the hidden layer defines each of these separate sections and the output layer aggregate them so that the result can be seen as follows



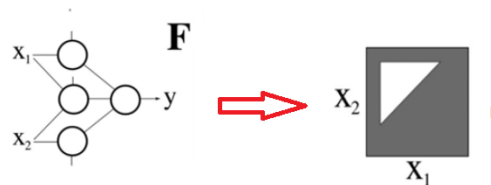
In the neural network D, the first neuron in the first layer creates a vertical line since it only receives  $x_1$ . The second neuron of the first layer creates a horizontal line since it only receives  $x_2$  as input. The output layer choose the common space as it can be seen in the result



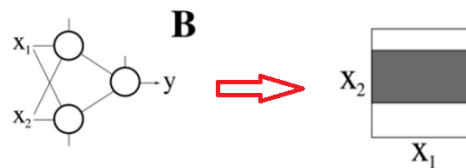
In the neural network E, each input neuron creates a straight line since they receive both inputs. The output neuron then finally choose the common chosen section of all these three lines.



In the neural network F, the first neuron of the input layer defines a vertical line as the last neuron of this layer creates a horizontal line since they receive only one input. The middle neuron of this layer can create the diagonal line. The output neuron defines the final section regarding these lines.



There will be only one image that we haven't assign the corresponding neural network. if we set the wight properly for the neural network B, in a way that both input neurons receive only  $x_2$ , the following image can be accomplished.



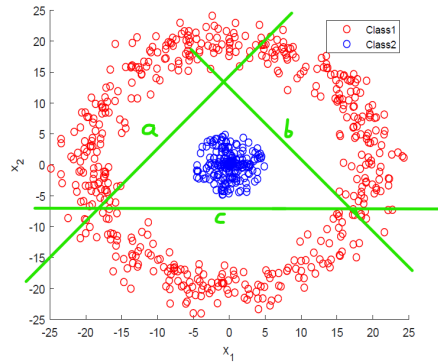
## 7 Question 7

a

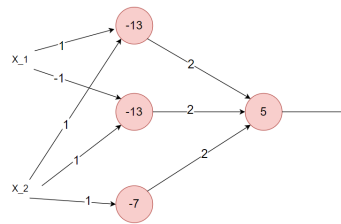
$$a : x_1 + x_2 = 13 \quad , \quad b : x_2 - x_1 = 13 \quad , \quad c : x_2 = -7$$

$$\Rightarrow \begin{cases} \text{blue} = 1 & (x_1 + x_2 \geq -13) \cap (x_2 - x_1 \geq -13) \cap (x_2 \geq -7) \\ \text{red} = 0 & \text{otherwise} \end{cases}$$





We can design the neural network using 3 input neurons and 1 output neuron as follows:



**b**

Only one generalized neuron will be enough.

$$f_{net}(x_1, x_2) = x_1^2 + x_2^2 \quad , \quad f_{act}(net) = \begin{cases} 1 & \text{if } net < 49 \\ 0 & \text{otherwise} \end{cases} \quad , \quad f_{out}(act) = act$$

**c**

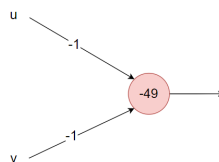
Using a single generalized neuron, we can create a connected geometric region in the feature space via appropriate equations and formulas for network function, activation and output function. If the TLU network creates this connected, cohesive space, it can be modeled using a single generalized neuron but in the cases that the TLU network consists of several cohesive regions (like question 5), several generalized neurons will be needed for each that regions.

**d**

The pre process includes considering the square of the inputs instead of themselves.

$$\text{define } u = x_1^2 \quad , \quad v = x_2^2 \quad \Rightarrow \quad y = \begin{cases} 1 & \text{if } -u - v \geq -49 \\ 0 & \text{otherwise} \end{cases}$$

In this way, only one single TLU will be enough.



## 8 Question 8

$$\begin{aligned}
 \text{neuron 1 : } \quad z_1 &= \exp\left(-\frac{(\max\{|x-3|, |x-5|, |x+2|\})^2}{4}\right) \Rightarrow \\
 f_{net} &= \max\{|x-3|, |x-5|, |x+2|\} \quad , \quad f_{act}(net) = \exp\left(-\frac{net^2}{4}\right) \quad , \quad f_{out}(act) = act \\
 \text{neuron 2 : } \quad z_2 &= \begin{cases} 0 & \sqrt{(x_1+3)^2 + (x_3-4)^2} > 4 \\ 1 & \text{otherwise} \end{cases} \Rightarrow \\
 f_{net} &= \sqrt{(x_1+3)^2 + (x_3-4)^2} \quad , \quad f_{act} = \begin{cases} 0 & \text{if } net \geq 4 \\ 1 & \text{otherwise} \end{cases} \quad , \quad f_{out}(act) = act \\
 \text{neuron 3 : } \quad y &= 6z_1 + 2z_2 + 4 \\
 f_{net} &= 6z_1 + 2z_2 + 4 \quad , \quad f_{act} = net \quad , \quad f_{out}(act) = act
 \end{aligned}$$

## 9 Question 9

Since the  $f_{net}$  can only be a weighted sum function, we have to somehow convert  $x_1^{x_2}$  into linear combination format. Thus we utilize the logarithmic function. In order to implement the max function, we have used the following formula :

$$\max\{a, b\} = \frac{a+b}{2} + \frac{|a-b|}{2}$$

We use 6 general neurons as follows:

$$\begin{aligned}
 \text{neuron 1 : input : } x_2 \quad , \quad f_{net} &= x_2 \quad , \quad f_{act}(net) = \log(net) \quad , \quad f_{out}(act) = act \\
 \text{neuron 2 : input : } x_1 \quad , \quad f_{net} &= x_1 \quad , \quad f_{act}(net) = \log(\log(net)) \quad , \quad f_{out}(act) = act \\
 \text{neuron 3 : input : } out_1, out_2 \quad , \quad f_{net} &= in_1 + in_2 \quad , \quad f_{act}(net) = \exp(\exp(net)) \quad , \quad f_{out}(act) = act \\
 \text{neuron 4 : input : } x_3, out_3 \quad , \quad f_{net} &= in_1 + in_2 \quad , \quad f_{act}(net) = net \quad , \quad f_{out}(act) = act \\
 \text{neuron 5 : input : } x_3, out_3 \quad , \quad f_{net} &= in_1 - in_2 \quad , \quad f_{act}(net) = |net| \quad , \quad f_{out}(act) = act \\
 \text{neuron 6 : input : } out_4, out_5 \quad , \quad f_{net} &= \frac{in_1}{2} + \frac{in_2}{2} \quad , \quad f_{act}(net) = net \quad , \quad f_{out}(act) = act
 \end{aligned}$$

## 10 Question 10

**a**

The weight matrix will be as follows

$$\begin{bmatrix} 0 & 0 & 2 & -2 \\ 2 & 0 & 0 & -3 \\ 1 & -1 & 0 & 4 \\ 0 & 2 & 1 & 0 \end{bmatrix}$$

**b**

In the input phase the output of the input neurons will be the value of external inputs and the output of the output neurons will be assigned arbitrarily as 0.

$$u_1 = 1 \quad , \quad u_2 = 1 \quad , \quad u_3 = 0 \quad , \quad u_4 = 0$$

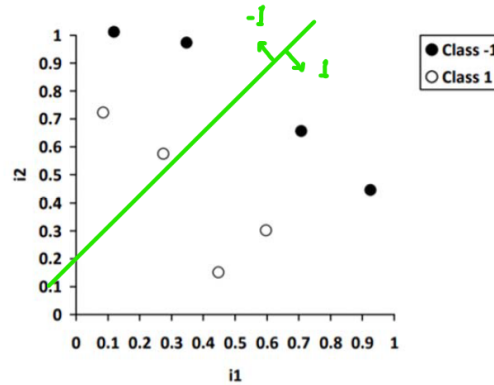
Now we start updating the values of the neurons based on the given order

$net_u$	$u_1$	$u_2$	$u_3$	$u_4$
$net_{u_3} = 2u_1 + u_4 = 2$	1	1	1	0
$net_{u_4} = -2u_1 - 3u_2 + 4u_3 = -1$	1	1	1	0
$net_{u_1} = 2u_2 + u_3 = 2$	1	1	1	0
$net_{u_2} = -u_3 + 2u_4 = -1$	1	0	1	0
$net_{u_3} = 2u_1 + u_4 = 2$	1	0	1	0
$net_{u_4} = -2u_1 - 3u_2 + 4u_3 = 2$	1	0	1	1
$net_{u_1} = 2u_2 + u_3 = 1$	1	0	1	1
$net_{u_2} = -u_3 + 2u_4 = 1$	1	1	1	0

## 11 Question 11

a

$$w_1x_1 + w_2x_2 \geq \theta \Rightarrow x_1 - x_2 \geq -0.2$$

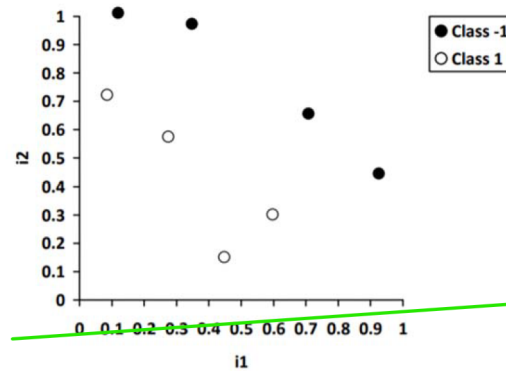


It can be seen that 4 data will be assigned with wrong label

b

$$x = (0.7, 0.65) \quad , \quad w = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \Rightarrow w^{(new)} = w + \eta(o - y)x = \begin{bmatrix} 1 \\ -1 \end{bmatrix} - \frac{1}{2} \times 2 \begin{bmatrix} 0.7 \\ 0.65 \end{bmatrix} = \begin{bmatrix} 0.3 \\ -1.65 \end{bmatrix}$$

$$\theta^{(new)} = \theta - \eta(o - y) = -0.2 + \frac{1}{2} \times 2 = 0.8 \quad \Rightarrow \quad 0.3x_1 - 1.65x_2 \geq 0.8$$



Four data have been labeled incorrectly.

## c & d

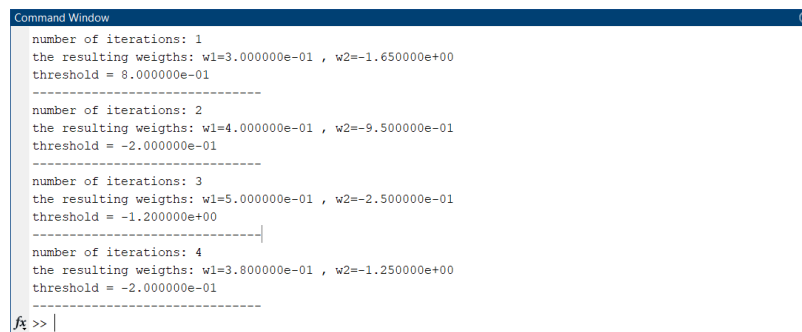
For this section we have developed a MATLAB code to train this simple neural network. All necessary explanations are provided via comment in the code itself. A section of the code can be seen below.

```

1  for i=1:n_iter
2      y = -1*ones(1,length(x));
3      y(w' * x' >= theta) = 1;
4      error(:,i) = out - y;
5      error_labeled = nonzeros(error(:,i)); % get the nonzero elements
6      % stop the process if there is no error :)
7      if isempty(error_labeled)
8          break;
9      else
10         % randomly choose one of the data labeled wrongly
11         idx = randi(length(error_labeled)); % get a random index
12         idx = find(error(:,i) == error_labeled(idx), 1);
13         % save the chosen data
14         selected_data(i, :) = x(idx, :);
15         % update weights and threshold
16         w = w + eta * (out(idx) - y(idx)) * x(idx, :);
17         theta = theta - eta * (out(idx) - y(idx));
18         fprintf('number of iterations: %d \n', i)
19         fprintf('the resulting weights: w1=%d , w2=%d \n', w(1), w(2));
20         fprintf('threshold = %d \n', theta);
21         disp('-----')
22     end
23 end

```

Now we set the number of iterations to 4 and run the code. Here we can see the value of weights and threshold in each iteration:

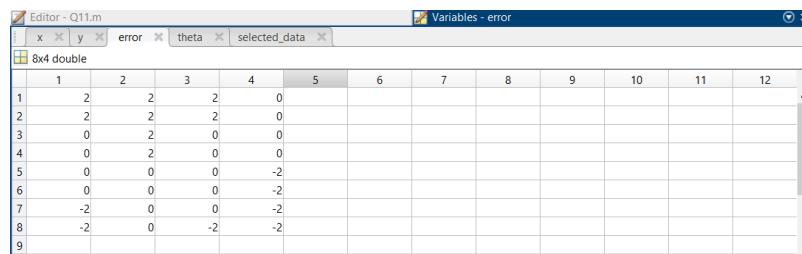


```

Command Window
number of iterations: 1
the resulting weights: w1=3.000000e-01 , w2=-1.650000e+00
threshold = 8.000000e-01
-----
number of iterations: 2
the resulting weights: w1=4.000000e-01 , w2=-9.500000e-01
threshold = -2.000000e-01
-----
number of iterations: 3
the resulting weights: w1=5.000000e-01 , w2=-2.500000e-01
threshold = -1.200000e+00
-----
number of iterations: 4
the resulting weights: w1=3.800000e-01 , w2=-1.250000e+00
threshold = -2.000000e-01
-----
fx >>

```

We have created a matrix in our code in order to save the values of errors in each iteration for outputs of all data



	1	2	3	4	5	6	7	8	9	10	11	12
1	2	2	2	2	0							
2	2	2	2	2	0							
3	0	2	0	0	0							
4	0	2	0	0	0							
5	0	0	0	-2								
6	0	0	0	-2								
7	-2	0	0	-2								
8	-2	0	-2	-2								
9												

We can see that we still facing 4 errors. So now we increase the number of iterations to 100 but we note that in the code, the process will stop whenever the error equals to zero. Here is the final result:

```

Command Window
-----
number of iterations: 13
the resulting weigths: w1=-1.800000e-01 , w2=-1.400000e+00
threshold = -1.200000e+00
-----
number of iterations: 14
the resulting weigths: w1=-8.800000e-01 , w2=-2.050000e+00
threshold = -2.000000e-01
-----
number of iterations: 15 |
the resulting weigths: w1=-7.800000e-01 , w2=-1.350000e+00
threshold = -1.200000e+00
-----
fx >>

```

It can be seen that after 15 iterations we won't have any errors. The final value for the weights and the threshold can be also seen above.

