

**A
REPORT
ON
#LAB- IV**

SUBMITTED BY:

Shova Thapa

071/BCT/540

SUBMITTED TO:

**DEPARTMENT OF ELECTRONICS AND COMPUTER
ENGINEERING
PULCHOWK CAMPUS
LALITPUR**

Neural Networks

ANNs are computing systems inspired by the biological neural networks that constitute animal brains. based on a collection of connected units called artificial neurons, (analogous to axons in a biological brain). . Such systems learn (progressively improve performance) to do tasks by considering examples, generally without task-specific programming. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as "cat" or "no cat" and using the analytic results to identify cats in other images.

Each connection (synapse) between neurons can transmit a signal to another neuron. The receiving (postsynaptic) neuron can process the signal(s) and then signal downstream neurons connected to it. Neurons may have state, generally represented by real numbers, typically between 0 and 1. Neurons and synapses may also have a weight that varies as learning proceeds, which can increase or decrease the strength of the signal that it sends downstream. Further, they may have a threshold such that only if the aggregate signal is below (or above) that level is the downstream signal sent.

Typically, neurons are organized in layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first (input), to the last (output) layer, possibly after traversing the layers multiple times.

The original goal of the neural network approach was to solve problems in the same way that a human brain would. Over time, attention focused on matching specific mental abilities, leading to deviations from biology such as back-propagation, or passing information in the reverse direction and adjusting the network to reflect that information.

Neural networks have been used on a variety of tasks, including computer vision, speech recognition, machine translation, social network filtering, playing board and video games, medical diagnosis and in many other domains.

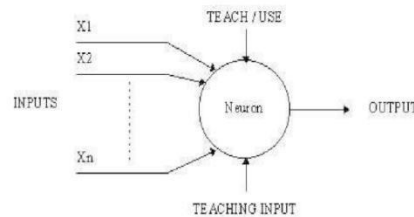


Figure 1.1: A simple neuron

The most common type of Artificial Neural Network consists of three groups, or layers, of units: a layer of "input" units is connected to a layer of "hidden" units, which is connected to a layer of "output" units.

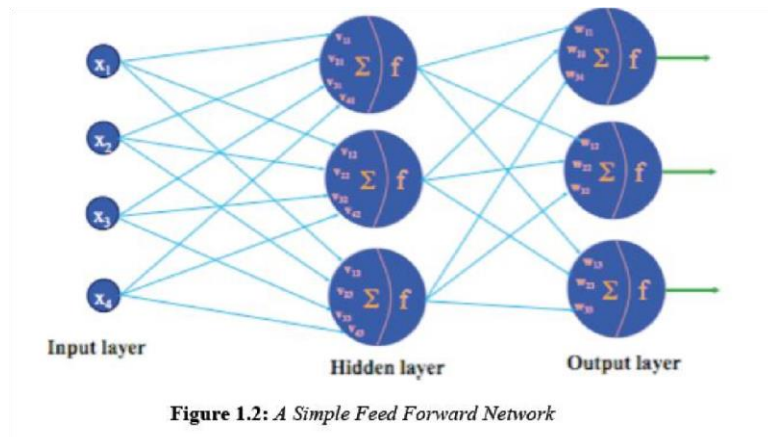
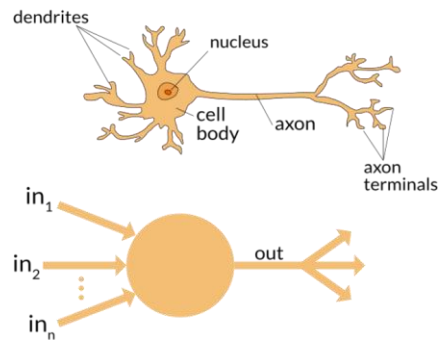


Figure 1.2: A Simple Feed Forward Network

Perceptrons:

The most influential work on neural nets in the 60's went under the heading of 'perceptrons' a term coined by Frank Rosenblatt. The perceptron (See figure 1.3) turns out to be an MCP model (neuron with weighted inputs) with some additional, fixed, pre-processing. Units labeled A1, A2, Aj , Ap are called association units and their task is to extract specific, localized featured from the input images. Perceptrons mimic the basic idea behind the mammalian visual system. They were mainly used in pattern recognition even though their capabilities extended a lot more.



Transfer Functions

The behavior of an ANN depends on both the weights and the input-output function (transfer function) that is specified for the units. This function typically falls into one of three categories:

- For *linear* (or ramp) the output activity is proportional to the total weighted output.
- For *threshold units*, the output is set at one of two levels, depending on whether the total input is greater than or less than some threshold value.
- For *sigmoid units*, the output varies continuously but not linearly as the input changes. Sigmoid units bear a greater resemblance to real neurons than do linear or threshold units, but all three must be considered rough approximations.

ASSIGNMENT 1:

Output:

```

Command Window
New to MATLAB? See resources for Getting Started.

Net is not learning enter another set of weights and Threshold value
weight w1=.1
weight w2=-.1
theta=.1
Output of Net
    0    0    1    0

McCulloch-Pitts Net for ANDNOT function
Weights of Neuron
    0.1000
   -0.1000

Threshold value
    0.1000

fx >>
  
```

ASSIGNMENT 2:

NAND:

```
clear; clc;
```

```
w1=input
```

```
('Weight
```

```
w1 = ');
```

```
% for nand gate use wt as: -0.1 -0.1 -0.1 i.e w1,w2 and theta works on bipolar as well
```

```
w2 = input('Weight w2 = '); disp('Enter Threshold Value'); theta = input('theta='); y =
```

```
[0 0 0 0]; x1 = [0 0 1 1]; x2 = [0 1 0 1]; z = [1 1 1 0]; % this is the output so change
```

```
accordingly for and,or,nand and nor con = 1; while con
```

```
    zin = x1*w1 + x2*w2;
```

```
        for i=1:4
```

```
            if zin(i) >= theta
```

```
                y(i) = 1;
```

```
            else
```

```
                y(i) = 0;
```

```
            end
```

```
        end
```

```
    disp('output of net');
```

```
    disp(y);
```

```
    if y == z
```

```
        con = 0;
```

```
    else
```

```
        disp('Net is not learning enter another set of weights and Thershold value');
```

```

w1 = input('weight w1=');

w2 = input('weight w2=');

theta = input('theta=');

end

end

disp('Mcculloh-Pitts Net

for NAND function');

disp('Weights of Neuron');

disp(w1);

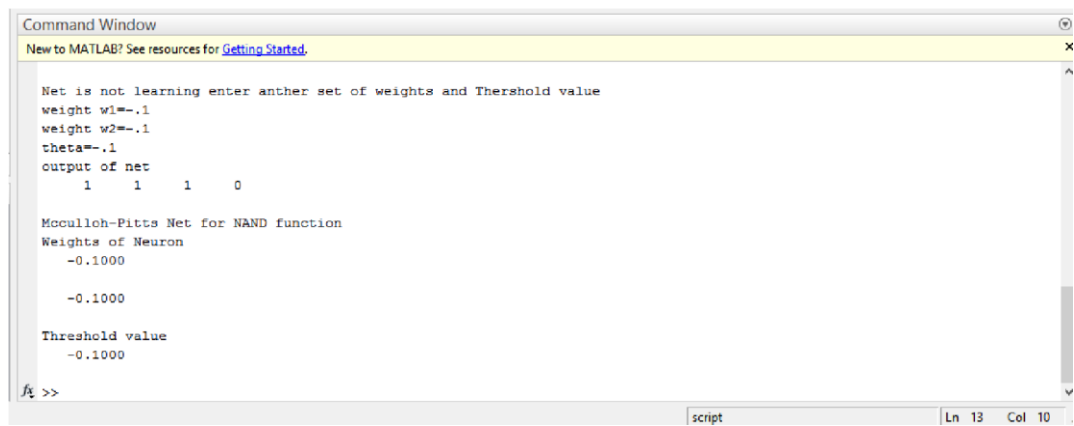
disp(w2);

disp('Threshold value');

disp(theta);

```

Output:



```

Command Window
New to MATLAB? See resources for Getting Started.

Net is not learning enter anther set of weights and Thershold value
weight w1=-.1
weight w2=-.1
theta=-.1
output of net
    1    1    1    0

Mcculloh-Pitts Net for NAND function
Weights of Neuron
    -0.1000

    -0.1000

Threshold value
    -0.1000

f1 >>
script Ln 13 Col 10

```

OR:

clear;

```

clc;

w1 = input('Weight w1 = ');
% for or gate use wt as: 0.1 0.1 0.1 i.e w1,w2 and theta works on bipolar as well
w2 = input('Weight w2 = ');

disp('Enter Threshold Value');

theta = input('theta=');

y = [0 0 0 0];

x1 = [0 0 1 1];

x2 = [0 1 0 1];

z = [0 1 1 1];

con = 1;

while con

    zin = x1*w1 + x2*w2;
    for i=1:4

        if zin(i) >= theta
            y(i) = 1;

        else

            y(i) = 0;

        end

    end

    disp('output of net');
    disp(y);

    if y == z

        con = 0;

```

```

else

    disp('Net is not learning enter anther set of weights and Thershold value');

    w1 = input('weight w1=');

    w2 = input('weight w2=');

    theta = input('theta=');

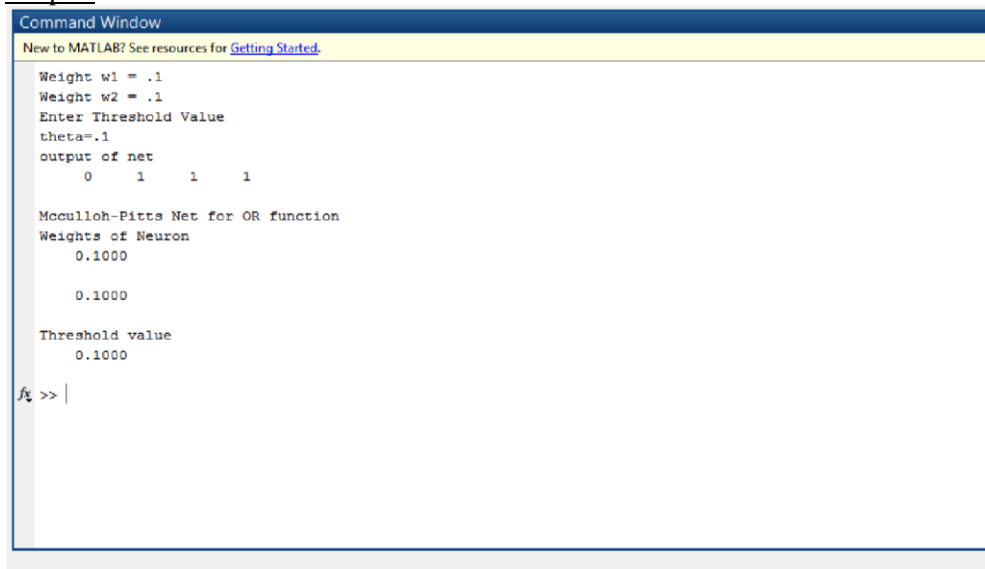
end

end

disp('Mcculloh-Pitts Net for OR function');
disp('Weights of Neuron'); disp(w1);
disp(w2); disp('Threshold value');
disp(theta);

```

Output:



```

Command Window
New to MATLAB? See resources for Getting Started.

Weight w1 = .1
Weight w2 = .1
Enter Threshold Value
theta=.1
output of net
    0    1    1    1

Mcculloh-Pitts Net for OR function
Weights of Neuron
    0.1000

    0.1000

Threshold value
    0.1000

fx >> |

```

NOR:

```

clear; clc;

w1 = input('Weight w1 = ');

```



```

% for nor gate use wt as : -0.1 -0.1 0 i.e w1,w2 and theta works on bipolar as well
w2 = input('Weight w2 = ');

disp('Enter Threshold Value');

theta = input('theta=');

y = [0 0 0 0];

x1 = [0 0 1 1];

x2 = [0 1 0 1];

z = [1 0 0 0]; % this is the output so change accordingly for and,or,nand and nor
con = 1;

while con

    zin = x1*w1 + x2*w2;

    for i=1:4

        if zin(i) >= theta

            y(i) = 1;

```

```

else
y(i) = 0;

end

disp('output of net');
disp(y);

if y == z

con = 0;

else

disp('Net is not learning enter another set of weights and Threshold value');
w1 = input('weight w1=');

w2 = input('weight w2=');

theta = input('theta=');

end

end

disp('Mcculloh-Pitts Net for NOR function');
disp('Weights of Neuron'); disp(w1);
disp(w2); disp('Threshold value');
disp(theta);

```

Output:

```
Command Window
New to MATLAB? See resources for Getting Started.

Weight w1 = -.1
Weight w2 = -.1
Enter Threshold Value
theta=0
output of net
    1    0    0    0

McCulloch-Pitts Net for NOR function
Weights of Neuron
    -0.1000
    -0.1000
Threshold value
    0

fx >> |
```

ASSIGNMENT 3:

BipolarNAND:

```
clear; clc;
```

```
w1 = input('Weight w1 = ');
```

```
% for nand gate use wt as: -0.1 -0.1 -0.1 i.e w1,w2 and theta works on bipolar as well
```

```
w2 = input('Weight w2 = '); disp('Enter Threshold Value'); theta = input('theta='); y =  
[0 0 0 0]; x1 = [-1 -1 1 1]; x2 = [-1 1 -1 1];
```

```
z = [1 1 1 -1]; % this is the output so change accordingly for and,or,nand and nor
```

```
con = 1;
```

```
while con
```

```
    zin = x1*w1 + x2*w2;
```

```
    for i=1:4
```

```
        if zin(i) >= theta
```

```
            y(i) = 1;
```

```

else

y(i) = -1;

end

disp('output of net');
disp(y);

if y == z

con = 0;

else

disp('Net is not learning enter another set of weights and Thershold value');
w1 = input('weight w1=');

w2 = input('weight w2=');

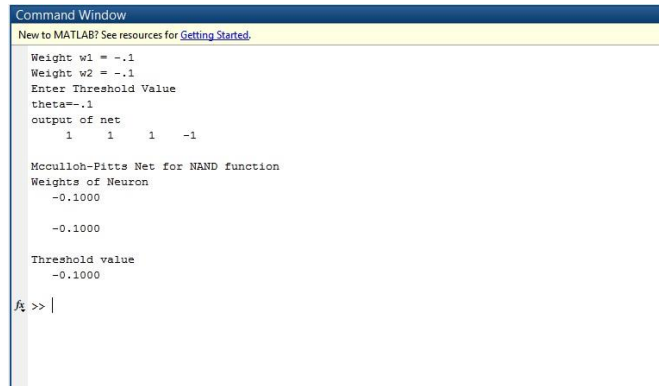
theta = input('theta=');

end
end

disp('Mcculloh-Pitts Net for NAND function');
disp('Weights of Neuron'); disp(w1); disp(w2);
disp('Threshold value'); disp(theta);

```

Output:



```
Command Window
New to MATLAB? See resources for Getting Started.

Weight w1 = -.1
Weight w2 = -.1
Enter Threshold Value
theta=-.1
output of net
    1    1    1   -1

McCulloch-Pitts Net for NAND function
Weights of Neuron
-0.1000

-0.1000

Threshold value
-0.1000

fx >> |
```

BipolarOR:

```
clear; clc;
```

```
w1 = input('Weight w1 = ');
```

```
% for or gate use wt as: 0.1 0.1 0.1 i.e w1,w2 and theta works on bipolar as well
```

```
w2 = input('Weight w2 = '); disp('Enter Threshold Value'); theta =
```

```
input('theta='); y = [0 0 0 0]; x1 = [-1 -1 1 1]; x2 = [-1 1 -1 1];
```

```
z = [-1 1 1 1]; % this is the output so change accordingly for and,or,nand and nor
```

```
con = 1;
```

```
while con
```

```
    zin = x1*w1 + x2*w2; for i=1:4
```

```
        if zin(i) >= theta
```

```
            y(i) = 1;
```

```
        else
```

```
            y(i) = -1;
```

```
        end
```

```
    end
```

```

disp('output of net');
disp(y);

if y == z

con = 0;

else

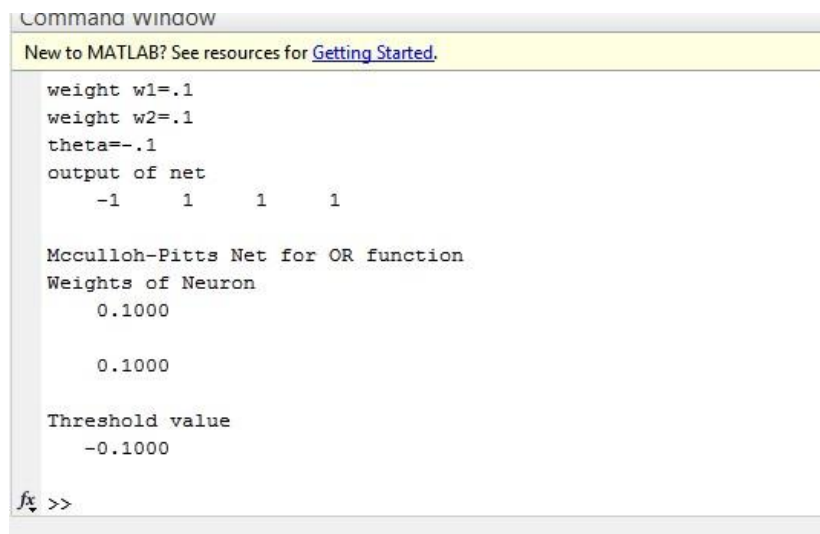
disp('Net is not learning enter another set of weights and Thershold value'); w1 =
input('weight w1='); w2 = input('weight w2='); theta = input('theta=');

end

disp('Mcculloh-Pitts Net for OR function'); disp('Weights of Neuron'); disp(w1); disp(w2);
disp('Threshold value'); disp(theta);

```

Output:



```

Command window
New to MATLAB? See resources for Getting Started.

weight w1=.1
weight w2=.1
theta=-.1
output of net
    -1     1     1     1

Mcculloh-Pitts Net for OR function
Weights of Neuron
    0.1000

    0.1000

Threshold value
    -0.1000

fx >>

```

BipolarNOR:

```
clear; clc;
```

```
w1 = input('Weight w1 = ');
```

```
% for nor gate use wt as : -0.1 -0.1 0.1 i.e w1,w2 and theta works on bipolar as well w2 =  
input('Weight w2 = '); disp('Enter Threshold Value'); theta = input('theta='); y = [0 0 0 0];  
x1 = [-1 -1 1 1]; x2 = [-1 1 -1 1];
```

```
z = [1 -1 -1 -1]; % this is the output so change accordingly for and,or,nand and nor
```

```
con = 1;
```

```
while con
```

```
    zin = x1*w1 + x2*w2; for i=1:4
```

```
    if zin(i) >= theta
```

```
        y(i) = 1;
```

```
    else
```

```
        y(i) = -1;
```

```
    end
```

```
end
```

```
disp('output of net');
```

```
disp(y);
```

```
if y == z
```

```
    con = 0;
```

```
else
```

```
    disp('Net is not learning enter another set of weights and Thershold value');
```

```
    w1 = input('weight w1=');
```

```
    w2 = input('weight w2=');
```

```
    theta = input('theta=');
```

```
end
```

```
end
```

```
disp('Mcculloh-Pitts Net for NOR function'); disp('Weights  
of Neuron');
```

```
disp(w1);
```

```
disp(w2);
```

```
disp('Thresh
```

```
old value');
```

```
disp(theta);
```


Output:

```
Command Window
New to MATLAB? See resources for Getting Started.

Net is not learning enter another set of weights and Thershold value
weight w1=.1
weight w2=.1
theta=-.1
output of net
-1    1    1    1

Net is not learning enter another set of weights and Thershold value
weight w1=-.1
weight w2=-.1
theta=.1
output of net
1    -1    -1    -1

Mcculloch-Pitts Net for NOR function
Weights of Neuron
-0.1000

-0.1000

Threshold value
0.1000
```

Activate Wi
6-20-2020

ASSIGNMENT 4:

clear; clc;

w10 = input('Bias w10= '); %follow the input sequence as %-1 2 2 3 -2 -

2 -3 2 2 as w10,w11,w12, w20, w21, w22, w30, w31,w32 w11 =

input('Weight w11 = '); w12 = input('Weight w12 = '); w20 =

input('Bias w20= '); w21 = input('Weight w21 = '); w22 = input('Weight

w22 = '); w30 = input('Bias w30= '); w31 = input('Weight w31= '); w32

= input('Weight w32= ');

disp('Enter Threshold Value');

% thershold is 0 for all theta1

= input('theta1='); theta2 =

input('theta2='); theta3 =

input('theta3='); y1 = [0 0 0

0]; y2= [0 0 0 0]; y3 = [0 0 0

0]; x1 = [0 0 1 1]; x2 = [0 1 0

1]; z = [0 1 1 0]; con = 1;

while con

```
z1_in = x1*w11 + x2*w12 + w10;
```

```
z2_in = x1*w21 + x2*w22 + w20;
```

```
for i=1:4
```

```
if z1_in(i) >= theta1
```

```
y1(i) = 1;
```

```
else
```

```
y1(i) = 0;
```

```
end
```

```
end
```

```
for i=1:4
```

```
if z2_in(i) >= theta2
```

```
y2(i) = 1;
```

```
else
```

```
y2(i) = 0;
```

```
end
```

```
end
```

```
z3_in = y1*w31+y2*w32+w30;
```

```
for i=1:4
```

```
if z3_in(i) >= theta3
```

```
y3(i) = 1;
```

```
else
```

```
y3(i) = 0;
```

```
end
```

```

end
disp('output of net');
disp(y3);

if y3 == z

con = 0;

else

disp('Net is not learning enter another set of weights and Thershold value');
w11 = input('Weight w11 = ');      w12 = input('Weight w12 = ');
w21 = input('Weight w21 = ');      w22 = input('Weight w22 = ');
w31 = input('Weight w31 = ');      w32 = input('Weight w32 = ');
disp('Enter Threshold Value');      theta1 = input('theta1=');
theta2 = input('theta2=');          theta3 = input('theta3=');

end
end
disp('Mcculloh-Pitts Net for XOR function');
disp('Biases'); disp(w10); disp(w20);
disp(w30); disp('Weights
of Neuron'); disp(w11);
disp(w12); disp(w21);
disp(w22); disp(w31);
disp(w32); disp('Threshold
values'); disp(theta1);
disp(theta2); disp(theta3);

```

Output:

```
Command Window
New to MATLAB? See resources for Getting Started.

Bias w10= -1
Weight w11 = 2
Weight w12 = 2
Bias w20= 3
Weight w21 = -2
Weight w22 = -2
Bias w30= -3
Weight w31= 2
Weight w32= 2
Enter Threshold Value
theta1=0
theta2=0
theta3=0
output of net
0    1    1    0

McCulloch-Pitts Net for XOR function
Biases
-1
```

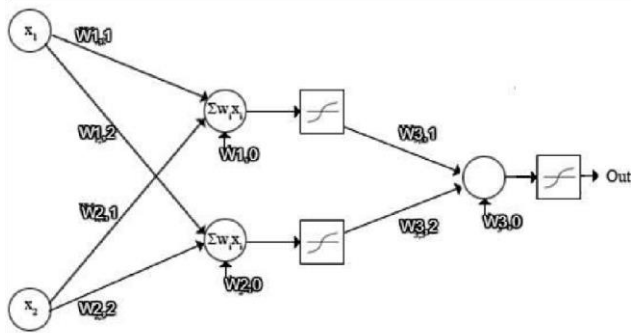


Fig: MLP

ASSIGNMENT 5:

```
clear; clc;
w10= -0.3;
w11 = 0.21;
w12 = 0.15;
w20 = 0.25;
w21 = -0.4;
w22 = 0.1;
w30 = -0.4;
```

w31 = -0.2;

w32 = 0.3;

r = 1 ; y1 = 0;

y2 = 0; y3 = 0;

x1 = [0 0 1 1];

x2 = [0 1 0 1];

z = [0 1 1 0];

d3 = 0; d1 = 0;

d2 = 0;

con2 = 1; while

con2<=100000

for i =1:4

z1_in = x1(i) * w11 + x2(i) * w12 + w10;

y1 = activation_function(z1_in);

z2_in = x1(i) * w21 + x2(i) * w22 + w20;

y2 = activation_function(z2_in);

z3_in = y1 * w31 + y2 * w32 + w30;

y3 = activation_function(z3_in);

d3 = (z(i)-y3) * y3 * (1-y3);

d_in1 = d3 * w31;

d1 = d_in1 * y1 * (1-y1);

d_in2 = d3 * w32;

d2 = d_in2 * y2 * (1-y2);

```

w30 = w30 + (r * d3 * 1);
w31 = w31 + (r * d3 * y1);
w32 = w32 + (r * d3 * y2);

```

```

w20 = w20 + (r * d2 * 1);
w21 = w21 + (r * d2 * x1(i));
w22 = w22 + (r * d2 * x2(i));

```

```

w10 = w10 + (r * d1 * 1);
w11 = w11 + (r * d1 * x1(i));
w12 = w12 + (r * d1 * x2(i));
con2 = con2+1; end end

```

```

disp('BackPropagation Net for XOR function');
disp('Weights of Neuron'); disp(w10);
disp(w11); disp(w12); disp(w20);
disp(w21); disp(w22); disp(w30);
disp(w31); disp(w32); disp('Check');
intr = 1; while intr    disp('Enter check
value');    i1 = input('i1=');    i2 =
input('i2=');    k1 = (i1 * w11) + (i2 *
w12) + w10;    k1 =
activation_function(k1);    k2 = (i1 *
w21) + (i2* w22) + w20;    k2 =
activation_function(k2);    k3 = (k1 *
w31) + (k2 * w32) + w30;    k3 =
activation_function(k3);

```

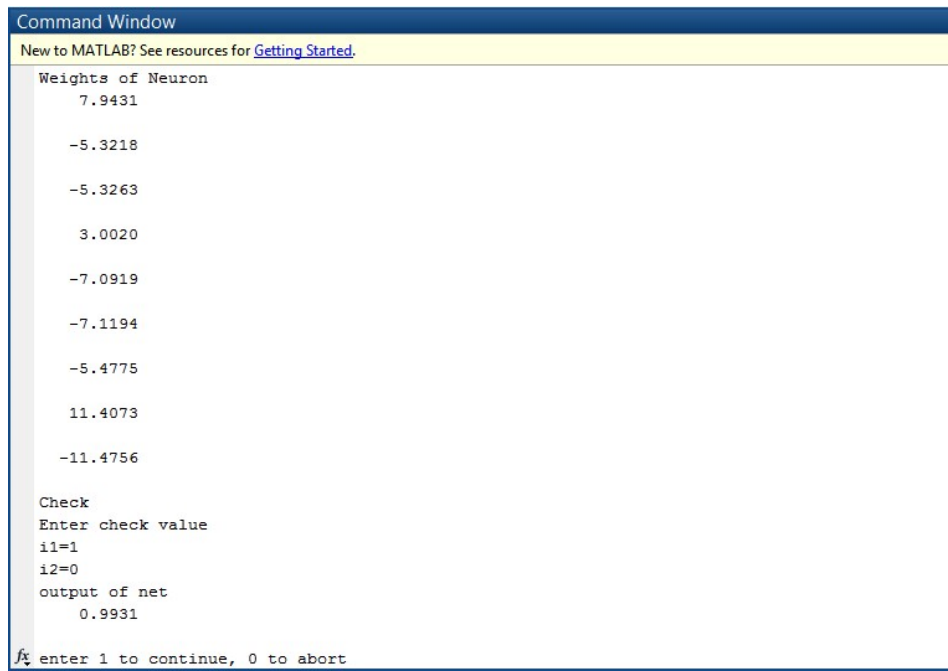
```
disp('output of net');

disp(k3);

disp('enter 1 to continue, 0 to abort');
intr = input('continue=');

end
```

Output:



```
Command Window
New to MATLAB? See resources for Getting Started.

Weights of Neuron
    7.9431
   -5.3218
   -5.3263
    3.0020
   -7.0919
   -7.1194
   -5.4775
   11.4073
  -11.4756

Check
Enter check value
i1=1
i2=0
output of net
    0.9931

> enter 1 to continue, 0 to abort
```

DISCUSSION:

We wrote and run the programs in Matlab to simulate various logical gates and their bipolar variants including XOR with backpropagation. Perceptron was developed for NAND, OR, ANDNOT, NOR and their bipolar variants, except XOR gate. Two neurons were connected serially for XOR gate. For each problem, weights and threshold values were taken as input and then checked to see if they generated desired output. This process was repeated correct output were obtained.

CONCLUSION:

Thus,from this lab session,we became familiarized with the concept of neurons, neural networks and perceptron.