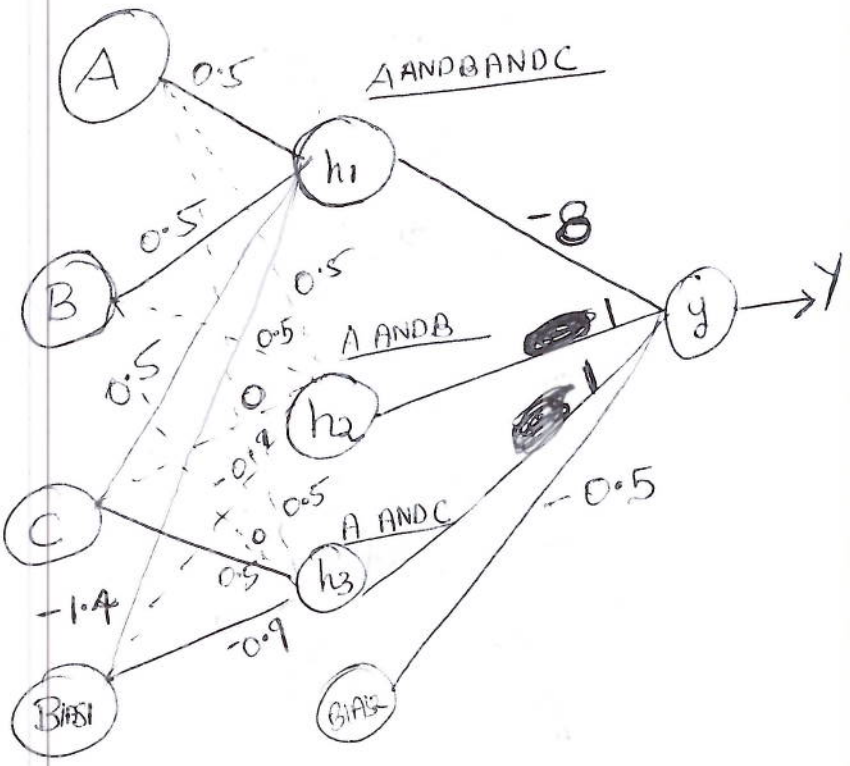
1) B)

A	B	C	Y
1	1	1	0
1	1	0	1
1	0	1	1
1	0	0	0
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0

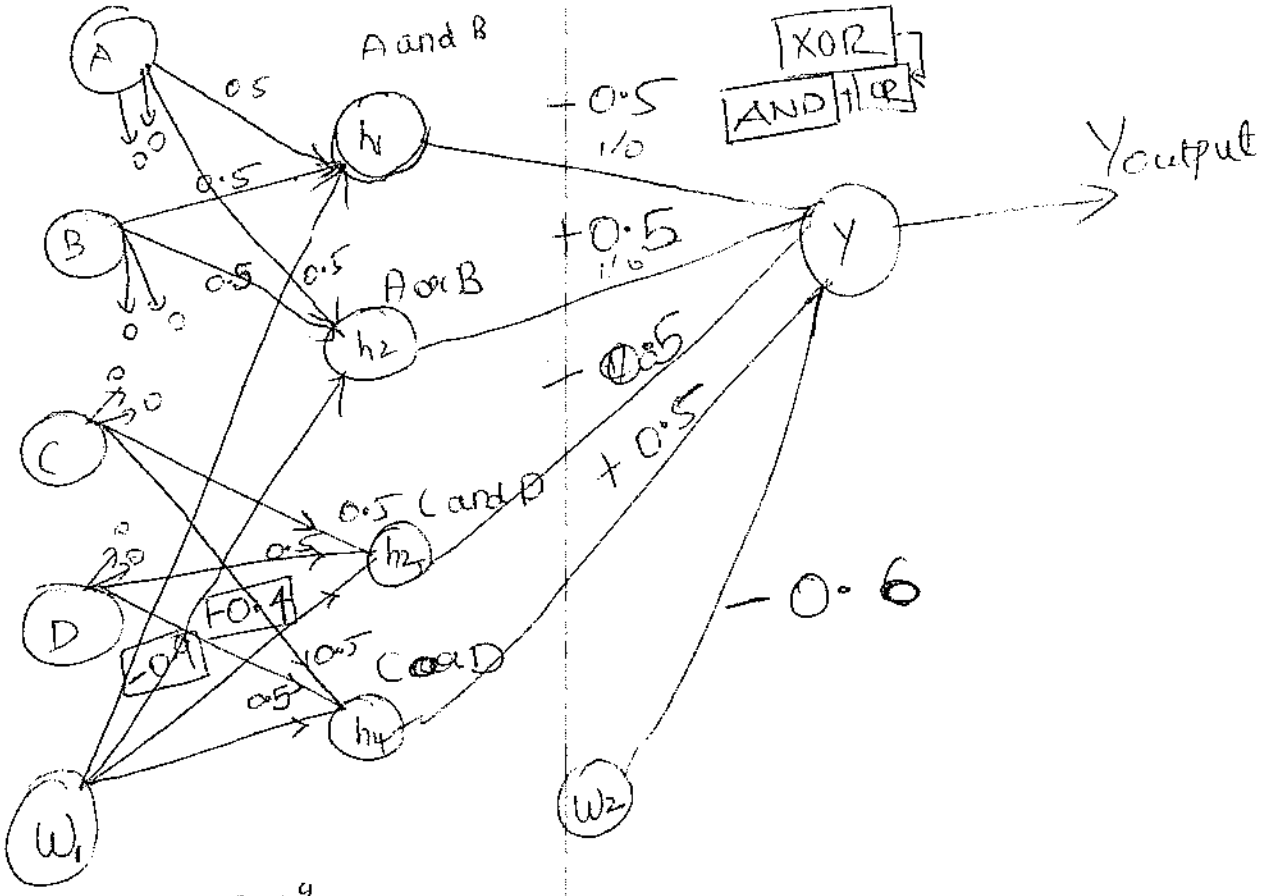


2)

A	B	C	D	Y
1	1	1	1	0
1	1	1	0	0
1	1	0	1	0
1	1	0	0	0
1	0	1	1	0
1	0	1	0	1
1	0	0	1	1
1	0	0	0	0
0	1	1	1	0
0	1	0	1	1
0	1	1	0	1
0	1	0	1	1
0	1	0	0	0
0	0	1	1	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	0

→ 0 0 0 0 0
0 0 0 1 0

1)b) continued



$wh_1 = -0.9$

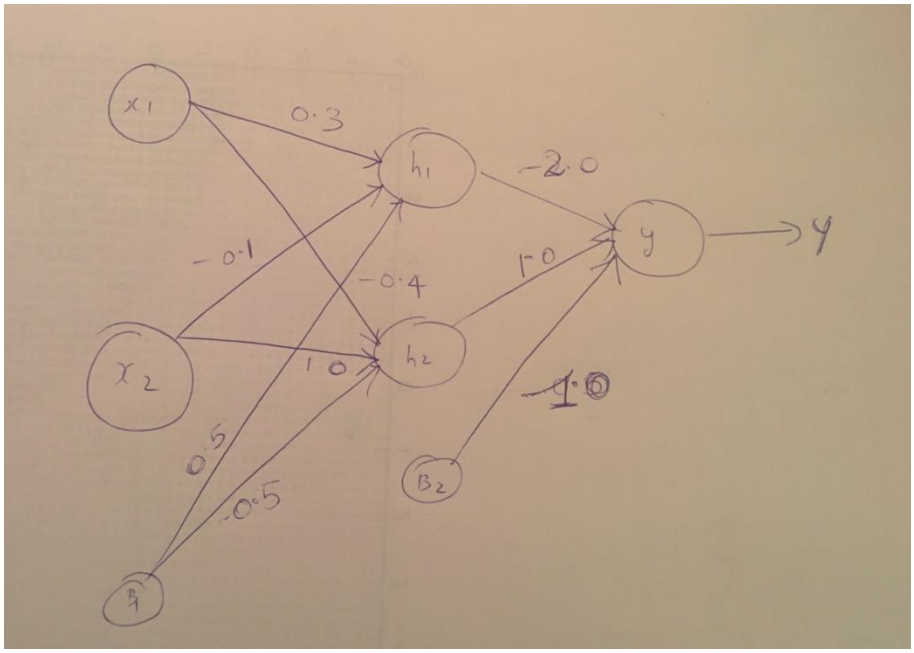
$wh_2 = -0.4$

$wh_3 = -0.9$

$wh_4 = -0.4$

2)

a)



b) Script has been saved as q2.m and valuate.m file in code folder.

REPORT:

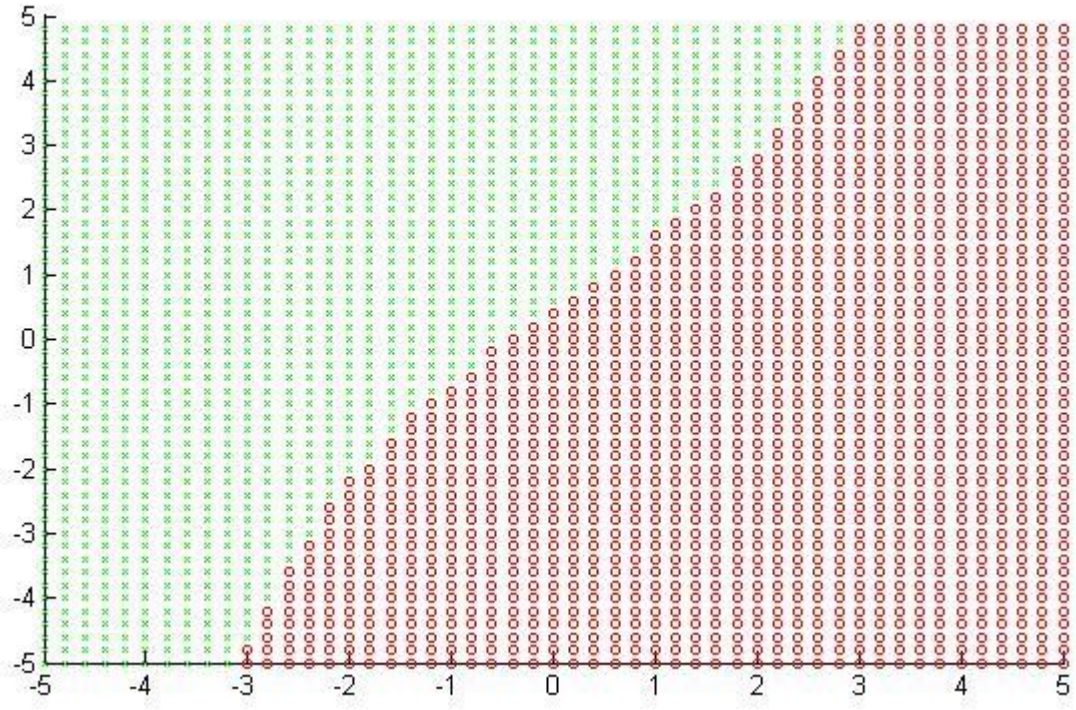
Create an x1 and x2 grid called 'f' that has every possible x1 and x2 varied from -.5 to .5 and create weights W1 and W2

```
x2=-5:.2:5;x2=repmat(x2,51,1);
B = x2(:);
x1=-5:0.2:5;
x1=x1';
x1=repmat(x1,size(x1),1);
f=B;
f(1:end,2)=x1;
f2=f;
```

Search value through valuate function with calculates the product sum of the weights and the inputs , passes the values through the function and repeats the first step with the second set of weights.If the value is <0-→ 0 is returned.If the value is >=0-→ 1 is returned

```
function [ooup,v] =valuate(W1,W2,x,fn)
h1=W1(2,1)*x(1)+W1(3,1)*x(2)+W1(1,1);
h2=W1(2,2)*x(1)+W1(3,2)*x(2)+W1(1,2);
y1=fn(h1);
y2=fn(h2);
Y=y1*W2(2)+y2*W2(3)+W2(1);
if(Y>=0)
    ooup=1;
    v=Y;
else
    ooup=0;
    v=Y;
```

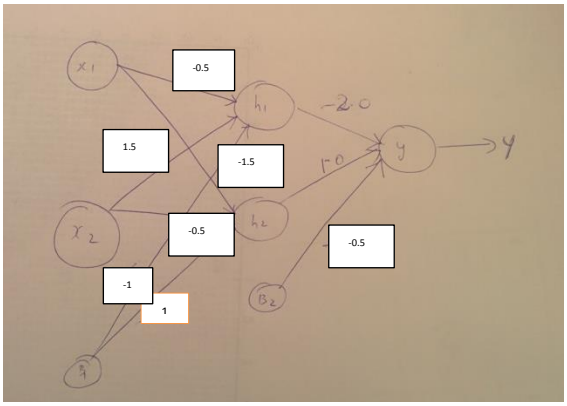
c)



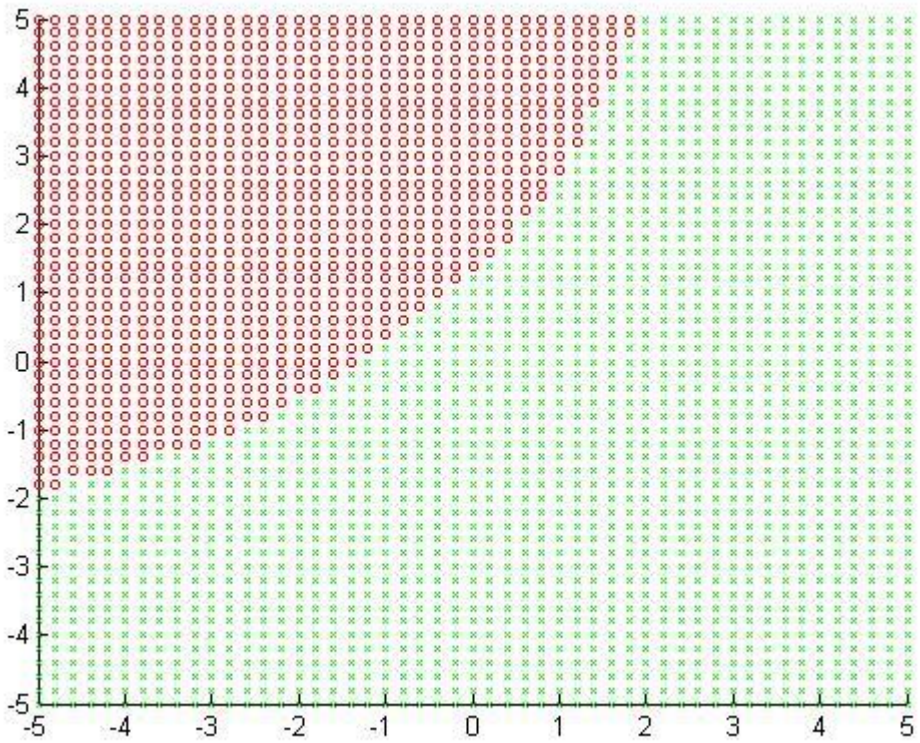
d) d1=-2.44856243751346

d2=3.28366478242260

e)



The script has been saved as q2.m and valuate.m in the code folder



d1= 3.930338252168671

d2= -2.908187623951532

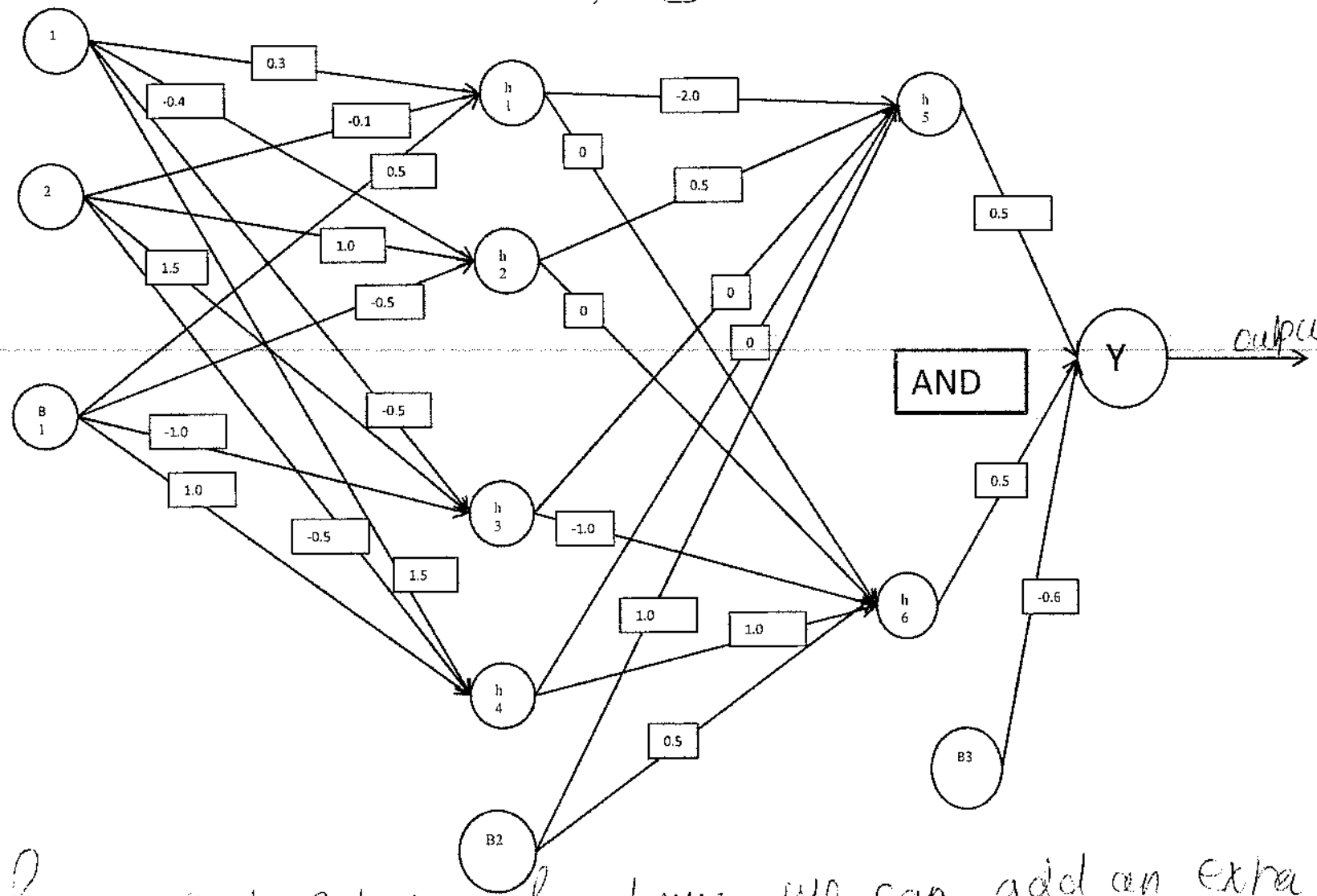
2)

Extra Credit

Activation function

h_1, h_2, h_3, h_4 } $f(x) = a \tanh(bx)$; $a = 1.716$, $b = 2/3$

h_5, h_6, Y } linear threshold @ zero (step function)

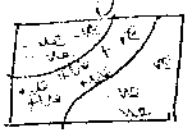


2.Extra credit

Since we need 2 decision boundaries we can add an extra layer (hidden) and use the AND perceptron in the end ∴

-ve	-ve	-ve
+ve	-ve	-ve
-ve	+ve	-ve
+ve	+ve	+ve

}



3)

a) Proof

$$f(x) = \frac{1}{(1+e^{-x})}$$

$$f'(x) = \left(\frac{1}{1+e^{-x}}\right)^2 e^{-x}(-1)$$

$$= \left(\frac{1}{1+e^{-x}}\right)\left(\frac{1}{1+e^{-x}}\right)(-e^{-x})$$

$$= \left(\frac{1}{1+e^{-x}}\right) \cdot \left(\frac{-e^{-x}}{1+e^{-x}}\right)$$

$$= \left(\frac{1}{1+e^{-x}}\right)$$

$$\downarrow$$

$$f(x)$$

$$\downarrow$$

$$1 - f(x) = 1 - \frac{1}{1+e^{-x}}$$

$$= \frac{1+e^{-x}}{1+e^{-x}}$$

$$= \frac{-e^{x}}{1+e^{x}}$$

$$f'(x) = f(x) \times [1 - f(x)]$$

(b) $x_1 = -0.5$ $\quad x_2 = 1$

$f(x) = \frac{1}{1+e^{-x}}$ $\quad h(e) = f'(x) = \frac{e^x}{(1+e^x)^2}$

$h_1 = -6$ $\quad H_1 = f(h_1) = 0.0025$

$h_2 = 6.1$ $\quad H_2 = f(h_2) = 0.9978$ $\quad \eta = 0.1$

$y = 5H_1 + 5.2H_2 = 5.2007$

$Y = \text{Output} = f(y) = \underline{0.9945}$

(c)

(a) $\Delta w_2 = \eta \times (t - Y) \times h(y) \times H2$

$= 0.1 \times (0.9 - 0.9945) \times h(5.2007) \times 0.9978$

$= \underline{-5.1418 \times 10^{-5}}$ Ans

(b) $\Delta w_1 = \eta \times (t - Y) \times h(y) \times H1$

$= 0.1 \times (0.9 - 0.9945) \times h(5.2007) \times 0.0025$

$= \underline{-1.2742 \times 10^{-7}}$

~~(b)~~ Changed weights = $w_2 = 5 + \Delta w_1 = 5.000\ldots$

$w_2 = 5.2 + \Delta w_2 = 5.1999\ldots$

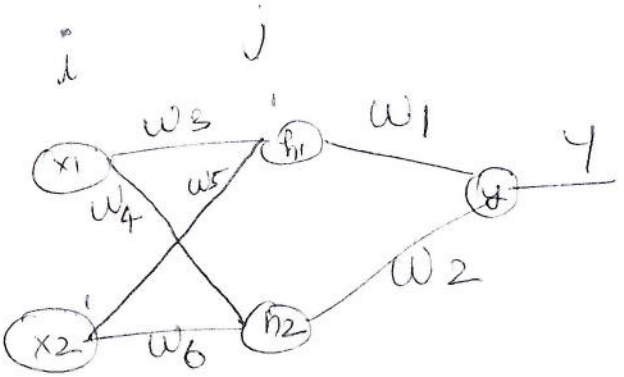
$\therefore \Delta w_3 = \eta \left[\sum_{1}^{2} (t - Y) f'(y) \text{ changed weight } 1 \right] h(h_j)(x_1)$

$= 0.1 \times (0.9 - 0.9945) \times h(y) \times w_{1new} \times h(h_1) \times x_1$

$= \boxed{3.177 \times 10^{-7}}$

Extra Credit

For first Propagation



$\Delta w_1 = -1.2742 \times 10^{-7}$

$\Delta w_2 = -5.148 \times 10^{-5}$

$\Delta w_3 = 3.1777 \times 10^{-7}$

$\Delta w_4 = 2.9917 \times 10^{-7}$

$\Delta w_5 = -6.3554 \times 10^{-7}$

$\Delta w_6 = -5.9834 \times 10^{-7}$

New weights

$w_1 \simeq 5 \quad w_2 \sim 5.199 \quad w_3 \simeq 4 \quad w_4 \simeq -3$

$w_5 \simeq 4 \quad w_6 \simeq 4.6$

Using MATLAB

↳ New Output

$= 0.994520007$

Old Output

$= 0.994517766$

} Error did not decrease significantly.

4.
REPORT

a) Process:

→Load the training and testing data using setupmnist()

→Get the number of hidden nodes from user→100/500/1000

→Do the following for each number of inputs from the 60,000 training(Case 1:100,Case2:1000,Case3:10000,Case4:60000)

→Convert the training labels into 10X1 vectors each

```
for i=1:length(train.labels)
out(i,(train.labels(i)+1))=1;
end

Now out contains the output targets
→Now create the net
net_1000_100=mlp(784,nhidden,10,'logistic');
→Now create the options matrix
→Train the net
[net_1000_100,options] = netopt(net_1000_100,options,train.data',out,'scg');
(net_1000_100=The network having 1000hidden nodes and 100 training samples)
```

→Testing

→Feed forward network for testing samples

```
for i=1:length(testing.data)
y(i,:)=mlpfwd(net_1000_100,testing.data(:,i)');
end
[C,I] = max(y,[],2);
fans_1000_100=I-1;
```

→Feed forward for cross validation (Test using the remaining training samples)

```
for i=101:60000
y(i-100,:)=mlpfwd(net_1000_100,training.data(:,i)');
end
y=round(abs(y));
test2_1000_100=binaryVectorToDecimal(y);
```

→Training error(Run the training matrix itself as input)

```
y=mlpfwd(net_100_60000,training.data');
[C,I] = max(y,[],2);
ftans_100_60000=I-1;
```

→Now find the accuracy of each by comparing with base data(example)

```
ft=[ftans_500_60000, ftans_500_10000, ftans_500_1000, ftans_500_100, ftans_100_60000, ftans_100_10000,
ftans_100_1000, ftans_100_100, ftans_1000_60000, ftans_1000_10000, ftans_1000_1000, ftans_1000_100];
for i=1:size(ft,2)
Y(:,i)=(ft(:,i)==training.labels(:,1));
end
yfinal=sum(Y)/60000;
accuracy.training=yfinal2';
```

##EXAMPLE CODE##

```
clc
clear all
%nhidden = input('input the number of hidden nodes(100/500/1000):');
disp('##########################################')
[training,testing] = setupMNIST();
nhidden = 1000;
ntrain=100;
train.data=training.data(:,1:ntrain);
train.labels=training.labels(1:ntrain,1);
out=zeros(ntrain,10);

for i=1:length(train.labels)
out(i,(train.labels(i)+1))=1;
end
net_1000_100=mlp(784,nhidden,10,'logistic');
options = zeros(1,18);
options(1) = 1; %display iteration values
options(14) = 500; %maximum number of training cycles (epochs)
[net_1000_100] = netopt(net_1000_100,options,train.data',out,'scg');
for i=1:length(testing.data)
y(i,:)=mlpfwd(net_1000_100,testing.data(:,i)');
end
[C,I] = max(y,[],2);
fans_1000_100=I-1;
clear  ntrain train.data train.labels a options y
```

```
disp('##########################################')
ntrain=1000;
train.data=training.data(:,1:ntrain);
train.labels=training.labels(1:ntrain,1);
out=zeros(ntrain,10);
for i=1:length(train.labels)
out(i,(train.labels(i)+1))=1;
end
net_1000_1000=mlp(784,nhidden,10,'logistic');
options = zeros(1,18);
options(1) = 1; %display iteration values
options(14) = 500; %maximum number of training cycles (epochs)
[net_1000_1000,options] = netopt(net_1000_1000,options,train.data',out,'scg');
for i=1:length(testing.data)
y(i,:)=mlpfwd(net_1000_1000,testing.data(:,i)');
end
[C,I] = max(y,[],2);
fans_1000_1000=I-1;
clear  ntrain train.data train.labels a options y
```

```
disp('########################################')
ntrain=10000;
train.data=training.data(:,1:ntrain);
train.labels=training.labels(1:ntrain,1);
out=zeros(ntrain,10);
for i=1:length(train.labels)
out(i,(train.labels(i)+1))=1;
end
net_1000_10000=mlp(784,nhidden,10,'logistic');
options = zeros(1,18);
options(1) = 1; %display iteration values
options(14) = 500; %maximum number of training cycles (epochs)
[net_1000_10000,options] = netopt(net_1000_10000,options,train.data',out,'scg');
for i=1:length(testing.data)
y(i,:)=mlpfwd(net_1000_10000,testing.data(:,i)');
end
[C,I] = max(y,[],2);
fans_1000_10000=I-1;
clear  ntrain train.data train.labels a options y
```

```
disp('########################################')
ntrain=60000;
train.data=training.data(:,1:ntrain);
train.labels=training.labels(1:ntrain,1);
out=zeros(ntrain,10);
for i=1:length(train.labels)
out(i,(train.labels(i)+1))=1;
end
net_1000_60000=mlp(784,nhidden,10,'logistic');
options = zeros(1,18);
options(1) = 1; %display iteration values
options(14) = 500; %maximum number of training cycles (epochs)
[net_1000_60000,options] = netopt(net_1000_60000,options,train.data',out,'scg');
for i=1:length(testing.data)
y(i,:)=mlpfwd(net_1000_60000,testing.data(:,i)');
end
[C,I] = max(y,[],2);
fans_1000_60000=I-1;
clear  ntrain train.data train.labels a options y C I
```

b) net.m has been saved
It was generated using
500 HIDDEN NODES
60000 NUMBER OF TRAINING DATA
0% TRAINING ERROR
1.66% TESTING ERROR

c)

During **training** we adjust the weight of the NN. The training error is actually the error of running through all the trainind data

During **Validation** when verifying if there is an increase in accuracy over data the network hasn't trained on (i.e. validation data set). If the accuracy over the training data set increases, but the accuracy over then validation data set reduces, then we over fit the NN.I used hold-out method to validate the data(ie train the network with some data and test using the remaining data)

During **Testing** we see how the NN performs in actual case.

	Training		Validation		Testing	
	Accuracy	Error	Accuracy	Error	Accuracy	Error
'net_100_60000'	1	0	1	-	0.968	0.032
'net_100_10000'	0.953383	0.046617	0.94406	0.05594	0.9462	0.0538
'net_100_1000'	0.881617	0.118383	0.87961	0.12039	0.8857	0.1143
'net_100_100'	0.6831	0.3169	0.682571	0.317429	0.6766	0.3234
'net_500_60000'	1	0	1	-	0.9834	0.0166
'net_500_10000'	0.96875	0.03125	0.9625	0.0375	0.9621	0.0379
'net_500_1000'	0.880967	0.119033	0.878949	0.121051	0.8841	0.1159
'net_500_100'	0.685767	0.314233	0.685242	0.314758	0.6807	0.3193
'net_1000_60000'	1	0	1	-	0.9843	0.0157
'net_1000_10000'	0.96935	0.03065	0.96322	0.03678	0.9628	0.0372
'net_1000_1000'	0.8733	0.1267	0.871153	0.128847	0.8763	0.1237
'net_1000_100'	0.685533	0.314467	0.685008	0.314992	0.6779	0.3221

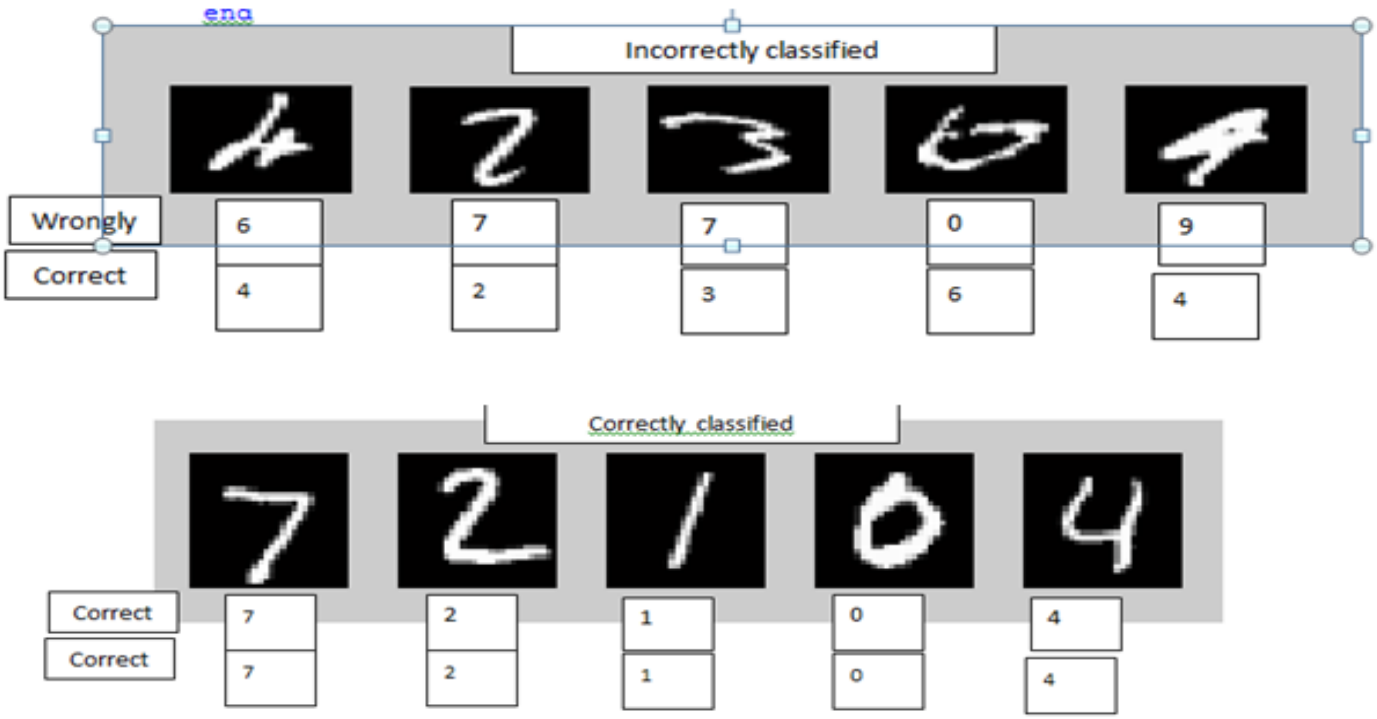
The validation accuracy for the NN with 1000 hidden nodes decreases more than the one with the 500 hidden Nodes(ie 0.96322<0.9625)-→ hence we may be over fitting
Since I am getting the almost same accuracy for test data I am going to choose the one with 500 hidden nodes.

Therefore Net_500_60000 ie NN with 500 hidden nodes and 60000 training data shall be the “best network”.

d) IMAGE

Please run the following code where fans_1000_60000 is the column vector containing the output for the testing the testing data for network with 1000 hidden nodes and 60000 training data.

```
%PICK OUT ROWS OF WRONG ONES
for i=1:size(fans_1000_60000,2)
sl(:,i)=(fans_1000_60000(:,i)==testing.labels(:,1));
end
vl=find(sl==0);
for i=1:5
wrongones(:,i)=testing.data(:,vl(i));
end

%DISPLAY THE IMAGES IN MATLAB
for i=1:5
mat = vec2mat(wrongones(:,i),28);
mat=mat';
K = mat2gray(mat);
subplot(1,5,i)
imshow(K)
end
clear K mat vl sl

%CREATE A NEW FIGURE
Figure

%PICK OUT ROWS OF CORRECT ONES
for i=1:size(fans_1000_60000,2)
sl(:,i)=(fans_1000_60000(:,i)==testing.labels(:,1));
end
vl=find(sl==1);
for i=1:5
correctones(:,i)=testing.data(:,vl(i));
end

%DISPLAY THE IMAGES IN MATLAB
for i=1:5
mat = vec2mat(correctones(:,i),28);
mat=mat';
K = mat2gray(mat);
subplot(1,5,i)
imshow(K)
end

((IMAGE IN NEXT PAGE))
```



4) Extra credit:

→Make sure u have set your default program to open bmp files are MS-PAINT

→ Save the "net" in Matlab

→ Run Following code:

```
matrix=zeros(28, 28)
K = mat2gray(matrix);
imshow(K);
% A black board has been created->you can close it
imwrite(K,'ath.bmp');
system('ath.bmp');
```

→ Now you can zoom in and draw on the board with white paint brush. Please save it after doing so(make sure it is still monochrome/16bit bmp).After saving run the following code.

→
```
KK=imread('ath.bmp' );
%now we have to make KK into 784X1 column matrix
C = KK(:);
C =double(C);
y=mlpfwd(net,A');
[C,I] = max(y,[],2);
finalans=I-1;
```

<<Using the Test image on the right that I drew and using the net I was able to correctly predict the number 2(finalans)