Question 1. Show that the decision boundary so constructed is a hyper-plane?

The output signal is defined by

$$y = \varphi(v) = \frac{1}{1+e^{-av}} = \frac{1}{1+e^{-a(b+\sum_{k=1}^{m} x_k)}}$$

Equivalently, we may write

$$v = b + \sum_{k=1}^{m} x_k = -\frac{1}{a} ln^{\left(\frac{1}{y}-1\right)} = \frac{1}{a} ln^{\left(\frac{y}{1-y}\right)}$$

Because the ξ is a threshold, we have:

$$\text{x} = \begin{cases} C_1, & if\ the\ output\ y > \xi \\ C_2, & otherwise \end{cases}$$

So the decision boundary is

$$b + \sum_{k=1}^{m} x_k = \frac{1}{a} ln^{\left(\frac{\xi}{1-\xi}\right)}$$

This is the equation of a hyper-plane.

We are done.

Question 2 Please supply a rigorous mathematical proof for the statement that that the XOR problem is not linearly separable?

Proof:

Suppose there is a linear function $f(x_1, x_2, c) = ax_1 + bx_2 + c$ can separate the two categories $y = \{1,0\}$.

So we must have: $f(x_1, x_2, c) = \begin{cases} ax_1 + bx_2 + c > 0, & when\ y = 1 \\ ax_1 + bx_2 + c < 0, & when\ y = 0 \end{cases}$

We have 4 examples; all of them should follow above the rules.

Substitute the four points into the inequalities, we get:

$$\begin{cases} a*0 + b*0 + c < 0 \\ a*1 + b*0 + c > 0 \\ a*0 + b*1 + c > 0 \\ a*1 + b*1 + c < 0 \end{cases} \Rightarrow \begin{cases} c < 0 & 1 \\ a*1 + c > 0 & 2 \\ b*1 + c > 0 & 3 \\ a*1 + b*1 + c < 0 & 4 \end{cases}$$

By inequalities 1, 2, 3, We can get $a*1 + b*1 + c > 0$. This is in contradiction with the inequalities 4.

So the example can't satisfy general rules. Then the XOR problem is not linearly separable.

We are done.

Question 3 Perform numerous logic functions, such as AND, OR, COMPLEMENT, and EXCLUSIVE OR function

a). Demonstrate the implementation of the logic functions AND, OR, and COMPLEMENT with selection of weights by off-line calculations.

AND:

To implement AND, we can use weights w=[1, 1, -1.5], for the bias, x1,x2 respectively.

These values yield the following truth table:

X1	X2	XW	outcome	y
0	0	-1.5	0	0
0	1	-0.5	0	0
1	0	-0.5	0	0
1	1	0.5	1	1

OR:
To implement AND, we can use weights w=[1, 1, -0.5], for the bias, x1,x2 respectively.
These values yield the following truth table:

X1	X2	XW	outcome	y
0	0	-0.5	0	0
0	1	0.5	1	1
1	0	0.5	1	1
1	1	1.5	1	1

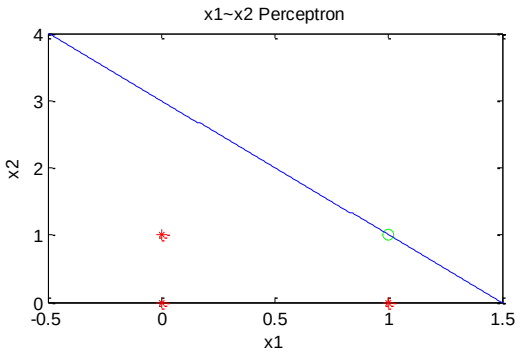
COMPLEMENT:
To implement AND, we can use weights w=[-1, 0.5], for the bias, x1,x2 respectively.
These values yield the following truth table:

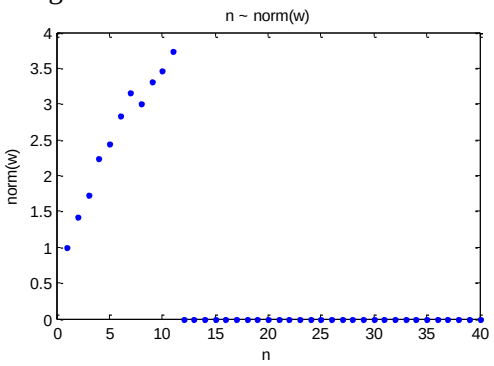
X	XW	outcome	y
0	1	1	1
1	-0.5	0	0

b). Demonstrate the implementation of the logic functions AND, OR, and COMPLEMENT with selection of weights by learning procedure. Suppose initial weights are chosen randomly and learning rate is 1.0. Plot out the trajectories of the weights for each case. Compare the results with those obtained in (a).
Solution :
Implement the logic function by the threshold function:

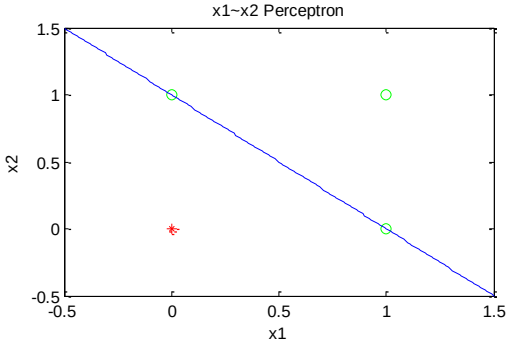
$$A = \begin{cases} 1, & if\ v \geq 0 \\ 0, & otherwise \end{cases}$$

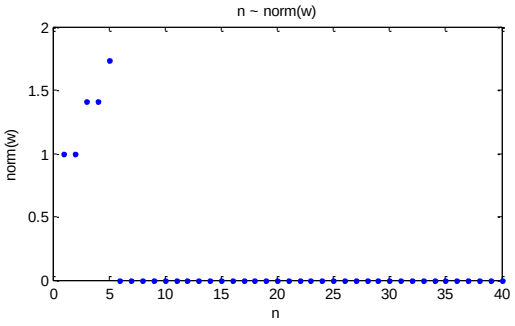
(*the solution code is attached at the end of this report*)
AND: w = [2 , 1 , -3]
The plot of logic function and the trajectories of the weights



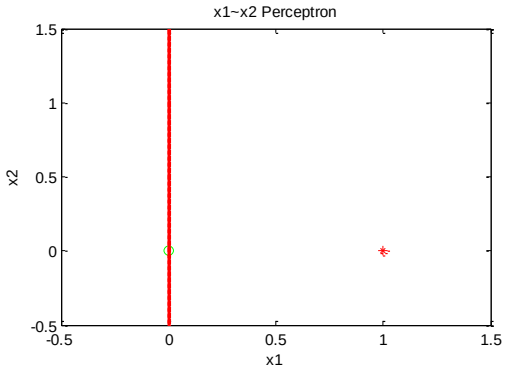


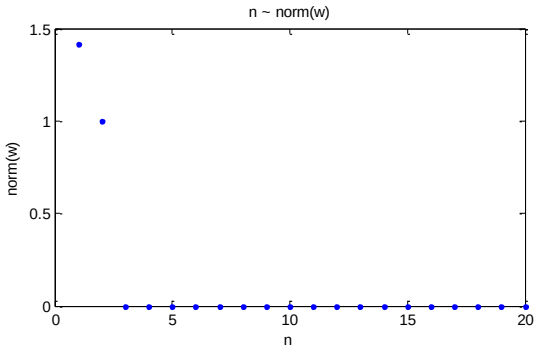
OR: w = [1 1 -1]
The plot of logic function and the trajectories of the weights



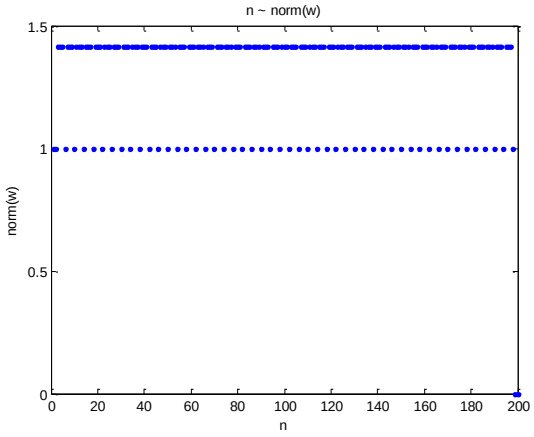


COMPLEMENT: w=[-1 0]



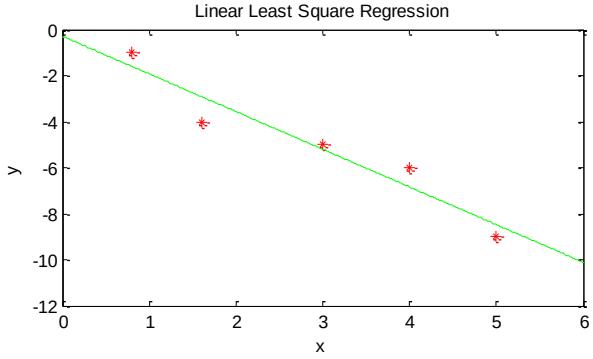


c). What would happen if the perceptron is applied to implement the EXCLUSIVE OR function with selection of weights by learning procedure? Suppose initial weight is chosen randomly and learning rate is 1.0. Do the computer experiment and explain your finding.
Yes, the trajectories of the weights not converge. So EXCLUSIVE OR function is not linearly separable.
The plot of logic function and the trajectories of the weights



Question 4 Single layer perceptron can be used to fit a linear model to a set of input-output pairs.

a). Find the solution of w and b using the standard linear least-squares (LLS) method.
Plot out the fitting result.
W = -1.6371 , b= -0.2851 (*the solution code is attached at the end of this report*)

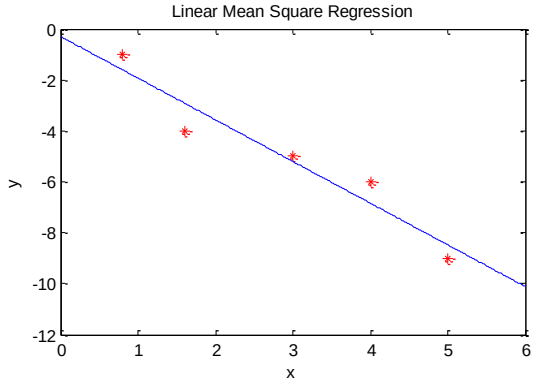


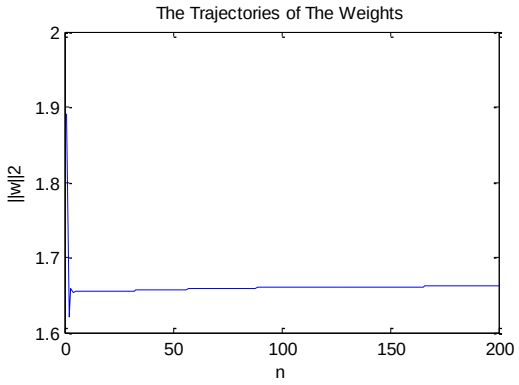
b). Suppose that initial weight is chosen randomly and learning rate is 0.02. Find the solution of w and b using the least-mean-square (LMS) algorithm for 200 epochs. Plot out the fitting result and the trajectories of the weights. Will the weights converge?

Solution:

W = -1.6364, b= -0.2876

Plot out the fitting result and the trajectories of the weights,





According the plot the trajectories of the weights is not converge.

c). Compare the results obtained by LLS and the LMS methods.

The results obtained for the two methods are almost same. In fact, both of them are based on the Least squares error , which means that Least squares solves the problem of fitting a line to data by finding the line for which the sum of the square deviations (or residuals) in the *d* direction (the noisy variable direction) are minimized.

LLS method is solve a linear system like $A\vec{x} = \vec{b}$

LMS method is solve $\min_{\vec{x}\in\Re^n} \phi(\vec{x}) = \min_{\vec{x}\in\Re^n}\left(\frac{1}{2}\vec{x}^T A\vec{x} - \vec{x}^T\vec{b}\right)$

While the two problems are equivalent.

$$A\vec{x} = \vec{b} \quad \Leftrightarrow \quad \min_{\vec{x}\in\Re^n} \phi(\vec{x}) = \min_{\vec{x}\in\Re^n}\left(\frac{1}{2}\vec{x}^T A\vec{x} - \vec{x}^T\vec{b}\right)$$

Solving the linear system is the traditional solution. Solving the optimization problem by steepest descent method is Modern iterative method, which will fast than the previous one when the dimension of A is very large.

Question 5 Derive the formula to calculate the optimal parameter w* such that the following cost function $J(w)$ is minimized.

Solution:

Based on the cost function:

$$J(w) = \sum_{i=1}^{n} r(i)(d(i) - y(i))^2 = \sum_{i=1}^{n} r(i)(d(i) - w^T x(i))^2$$

1.LMS solution:

The gradient of $J(w)$ is

$$\frac{\partial J(w)}{\partial w} = g(w) = -2X^T R(d - Xw^T)$$

$$X^T = \begin{bmatrix} x(1,1) & .. & x(1,m) \\ .. & .. & .. \\ x(n,1) & .. & x(n,m) \end{bmatrix};$$

$$R = \begin{bmatrix} r(1) & .. & 0 \\ .. & r(i) & .. \\ 0 & .. & r(n) \end{bmatrix};$$

$$d = [d(1), d(2), .. d(n)]^T;$$

Applying the steepest descent method, we have

$$w(n+1) = w(n) - \eta g(w) = w(n) + 2\eta x^T r(d - Xw^T), \qquad \eta \text{ is the learning rate}$$

So, the LMS solution is
For i=1,2, .. compute

$$w(n+1) = w(n) - \eta g(w) = w(n) + 2\eta x^T r(d - Xw^T)$$

Stop when w convergence.

2. LLS solution:

$$J(w) = \sum_{i=1}^{n} r(i)(d(i) - y(i))^2 = \sum_{i=1}^{n} r(i)(d(i) - w^T x(i))^2$$

Set gradient of $J(w)$

$$\frac{\partial J(w)}{\partial w} = g(w) = -2X^T R(d - Xw^T) = 0$$

So

$$X^T Rd - X^T RXw^T = 0 \rightarrow \quad X^T Rd = X^T RXw^T$$

Then

$$w^T = (X^T RX)^{-1} \mathrm{X}^T Rd$$

This is the close formula to the optimal parameter w.

Appendix.

Q3

Code for implement the Perceptron Question 3

```
function homework1_3()
input = [0,0;0,1;1,0;1,1];
output = [0;0;0;1];%% for AND ; output = [0;1;1;1];%% for OR  ; output = [0;1;1;0]; %% for XOR
% input = [0;1]; %% for COMPLEMENT  % output = [1,0];
[m,n]= size(input);
xx = [input,ones(m,1)];
%% preceptron
ita = 1;
w = zeros(n+1,1);
N= 10;
error_his = zeros(N,1);
w_his = zeros(N*m,1);
step = 0;
for k = 1:N
   sumerror = 0;
   for i = 1:m
      error = output(i)-active_function(xx(i,:)*w);
       if error~=0
           w = w + ita*error*xx(i,:)';
           sumerror = sumerror + 1 ;
           step = step + 1 ;
           w_his(step) = norm(w);
       end
   end
   error_his(k)=sumerror;
end
w'

function re = active_function(v)
re = 0;
if v>=0
    re = 1;
end
```

Q5

Code for Least Mean Square Regression and Linear Least Square Regression

```
function homework1_1()
Data = [0.8,-1;1.6,-4;3,-5;4.0,-6;5.0,-9];
[m,n] = size(Data);
%% Linear Least Square Regression
x = [Data(:,1),ones(m,1)];
y = Data(:,2);
w = inv(x'*x)*x'*y
test_x = [0:0.01:6];
test_y = test_x*w(1) + w(2);
plot(Data(:,1),Data(:,2),'r*'); hold on;
plot(test_x,test_y,'g');
xlabel('x');
ylabel('y');
title('Linear Least Square Regression');

%% Least Mean Square Regression
ita = 0.02;
iter_n = 200;
w0 = zeros(n,1)
agti_w0 = zeros(iter_n,1);
for i = 1:iter_n
   error = y - x*w0;
   error;
   w0 = w0 + ita*x'*error
   agti_w0(i) = norm(w0);
end

% figure(2)
test_x = [0:0.01:6];
test_y = test_x*w0(1) + w0(2);
plot(Data(:,1),Data(:,2),'r*'); hold on;
plot(test_x,test_y,'g');
xlabel('x');
ylabel('y');
title('Linear Mean Square Regression');
figure(3)
plot(agti_w0);
xlabel('n');
ylabel('||w||2');
title('The Trajectories of The Weights');
```