Name : LUO ZIJIAN

Matric.No： A0224725H

MUSNET： E0572844

Subject： NEURAL NETWORKS

Assignment: HOMEWORK ONE

Solution 1

According to the question.1. that v_n is the induced local field, so it can be written as:

$$v_k = \sum_{i=1}^{m} w_{ki} x_i + b_k \tag{1}$$

The observation vector

$$x = [x_1, x_2, x_3, ..., x_m]^T \tag{2}$$

The weight vector

$$w = [w_{k1}, w_{k2}, w_{k3}, ..., w_{km}] \tag{3}$$

Thus, combine (1)(2)(3), we can get this:

$$v(n) = \sum_{i=1}^{m} w_i(n) x_i(n) + b(n) = w^T(n) x(n) \tag{4}$$

So the output:

$y = \varphi(v) > \xi \rightarrow$ x belongs to class1

$y = \varphi(v) < \xi \rightarrow$ x belongs to class2

The key to estimate whether the following functions can be chosen as the activation function is that function satisfies following hyper-plane equations:

$$v(n) = w^T(n) x(n) = 0$$

Next I will use MATLAB to help me to plot those functions in coordinate axis.

(1) Quadratic function: $\varphi(v) = (v-a)^2 + c$

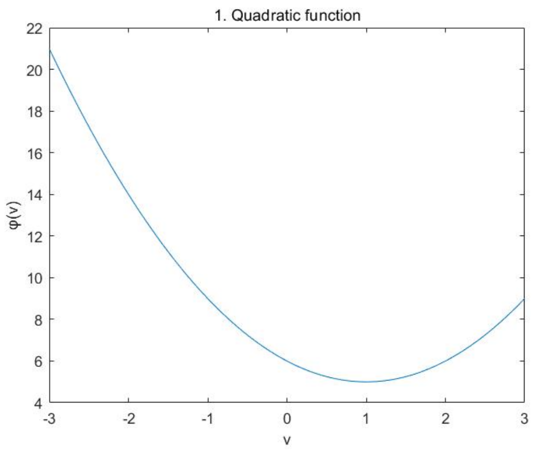
Let $y = \varphi(v) = (v-a)^2 + c = \xi$

So the decision boundary is :

$$v = a \pm \sqrt{\xi - c}$$

$$v = x_1 w_1 + x_2 w_2 + \cdots + x_m w_m + b$$

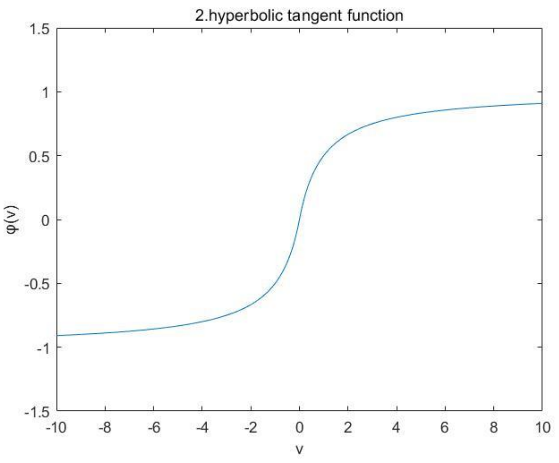
However, in this case it exists 2 hyper-plane. This function can not be chosen as activation function.



(2) Hyperbolic tangent function: $\varphi(v) = \dfrac{1-e^{-v}}{1+e^{-v}}$

Let $y = \varphi(v) = \dfrac{1-e^{-v}}{1+e^{-v}} = \xi$

So the decision boundary is : $v = \ln(\dfrac{1-\xi}{1+\xi})$

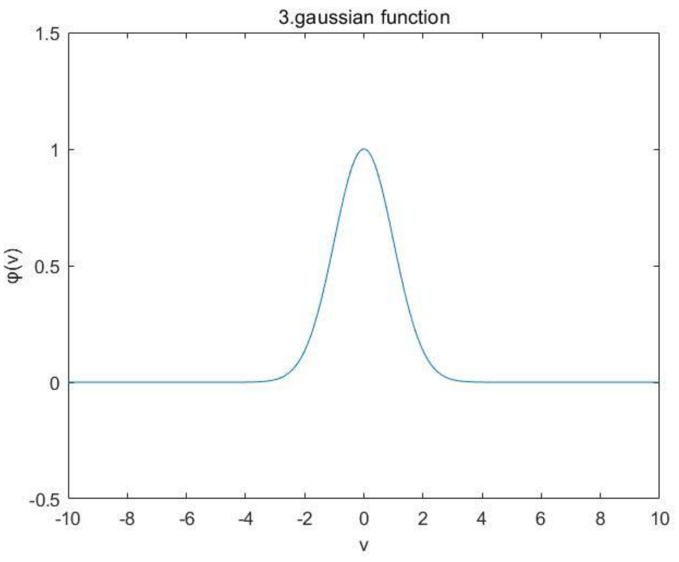


Thus , according to equation (5), while $x_1 w_1 + x_2 w_2 + \cdots + x_m w_m + b > \ln(\dfrac{1-\xi}{1+\xi})$, x belongs to C1(respectively, $x_1w_1 + x_2w_2 + \cdots + x_mw_m + b < \ln(\frac{1-\xi}{1+\xi})$, x belongs to C2)

(3) Bell-shaped Gaussian function : $\varphi(v) = e^{-\frac{(v-m)^2}{2}}$

Let $y = \varphi(v) = e^{-\frac{(v-m)^2}{2}} = \xi$

So the decision boundary is : $v = m \pm \sqrt{-2\ln(\xi)}$



However, in this case it exists 2 hyper-plane. This function can not be chosen as activation function.

Solution 2

It is obvious that XOR problem is not linearly separable, but it is hard to proof directly. So this time I choose proofs by contradiction to proof this problem.

Suppose that XOR is linearly separable, so it exists a hyper-plane to meet those condition:

$$v(n) = w^T(n)x(n) + b$$
$$(x1, x2, y) \in [(1,1,0), (1,0,1), (0,1,1), (0,0,0)]$$

First case: if $wx^T > 0$, the class label will be C1, otherwise the class label will be C2. According to the truth table,

$$[w_1, w_2][1,0]^T > 0$$
$$[w_1, w_2][0,1]^T > 0$$
$$[w_1, w_2][1,1]^T < 0$$
$$[w_1, w_2][0,0]^T < 0$$

From the linear basic, we can easily get: $[1,1]=[1,0]+[0,1]$,

Then $[w_1, w_2][1,1]^T = [w_1, w_2][1,0]^T + [w_1, w_2][0,1]^T > 0$, but it is self-contradictory.

Second case: when if $wx^T < 0$, the class label will be C1, otherwise, the class label will be C2.

Similarly, according to the truth table,

$$[w_1, w_2][1,0]^T < 0$$
$$[w_1, w_2][0,1]^T < 0$$
$$[w_1, w_2][1,1]^T > 0$$
$$[w_1, w_2][0,0]^T > 0$$

From the linear basic, we can easily get: $[1,1]=[1,0]+[0,1]$,

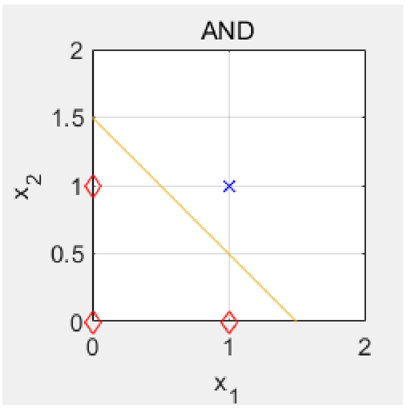
Then $[w_1, w_2][1,1]^T = [w_1, w_2][1,0]^T + [w_1, w_2][0,1]^T < 0$, but it is self-contradictory.

In conclusion, the XOR problem is not linearly separable.

Solution 3

a) I use MATLAB to help me demonstrate the implementation of the logic function AND, OR, COMPLEMENT, NAND with selection of weights by off-line calculations.

   1. AND

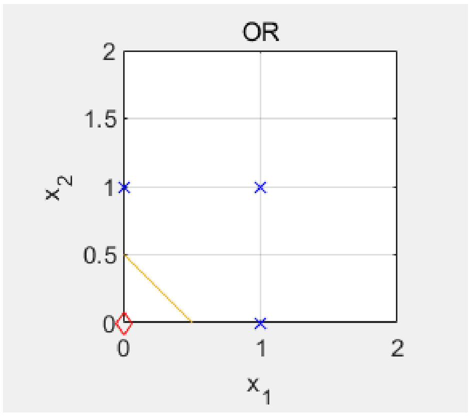


$$v = wx^T = [-1.5,1,1]\begin{pmatrix}1\\x_1\\x_2\end{pmatrix}$$

So the weight vector is [1,1], and the bias is -1.5

If $x_1 + x_2 - 1.5 > 0$, the value of y is 1, otherwise, the value of y is 0.

2. OR

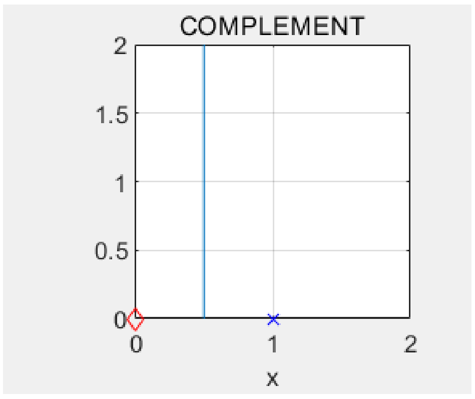


$$v = wx^T = [-0.5,1,1]\begin{pmatrix}1\\x_1\\x_2\end{pmatrix}$$

So the weight vector is [1,1], and the bias is -0.5

If $x_1 + x_2 - 0.5 > 0$, the value of y is 1, otherwise, the value of y is 0.

3. COMPLEMENT

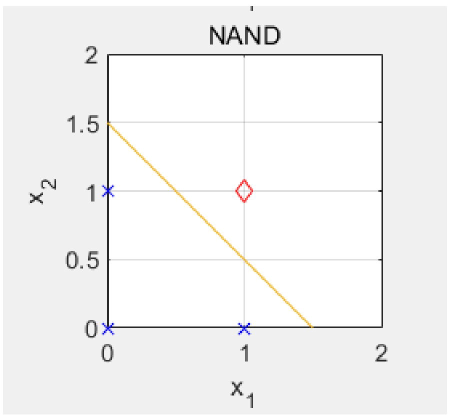


$$v = wx^T = [-0.5,1]\begin{pmatrix}1\\x\end{pmatrix}$$

So the weight vector is [1], and the bias is -0.5

If $x - 0.5 > 0$, the value of y is 1, otherwise, the value of y is 0.

4. NAND



$$v = wx^T = [1.5, -1, -1]\begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}$$

So the weight vector is [-1,-1], and the bias is 1.5

If $-x_1 - x_2 + 1.5 > 0$, the value of y is 0, otherwise, the value of y is 1.

b) Perceptron Learning Algorithm is a learning process, which can be shown as below:

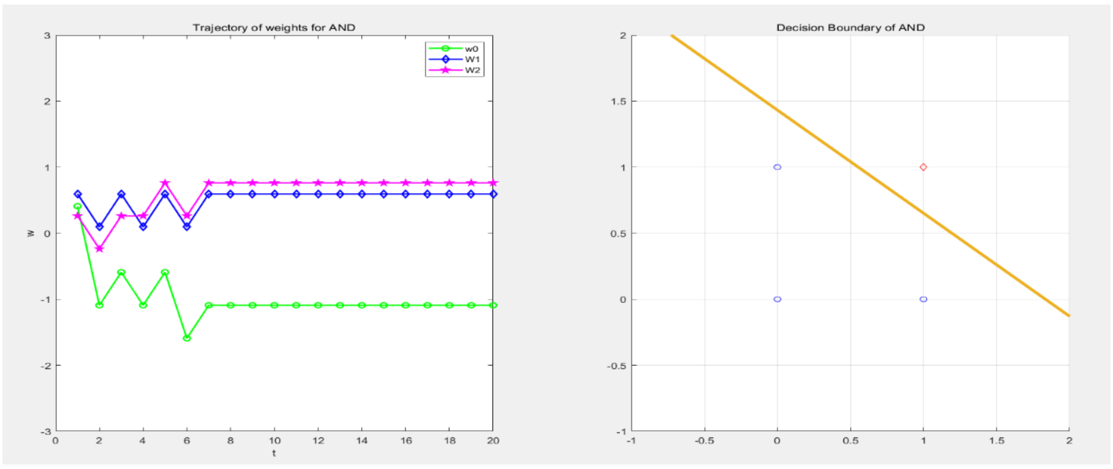
$$w(n+1) = w(n) + \eta e(n)x(n)(\eta > 0)$$
$$e(n) = d(n) - y(n)$$

1. AND

Input for AND:

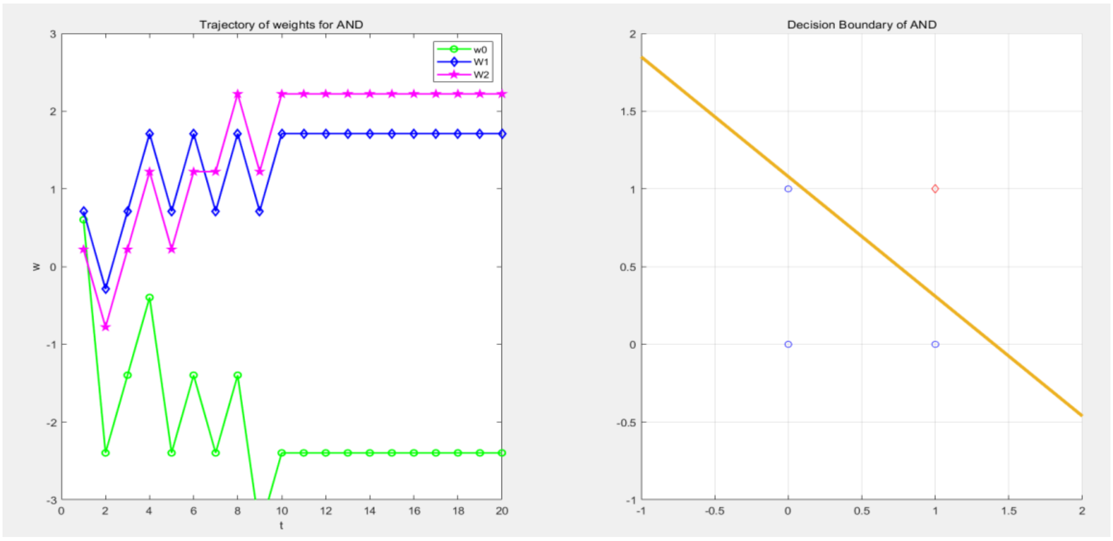
$$x = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} d = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} w = \begin{bmatrix} 0.6256 & 0.7802 & 0.0811 \end{bmatrix}$$

The trajectory of weights in this case is plotted in left, and the obtained decision boundary is plotted in right.



$(\eta = 1)$

We can easily find the epoch of $(\eta = 1)$ is about 7, When we change the learning rate= 0.5, the epoch of $(\eta = 0.5)$ is about 10.

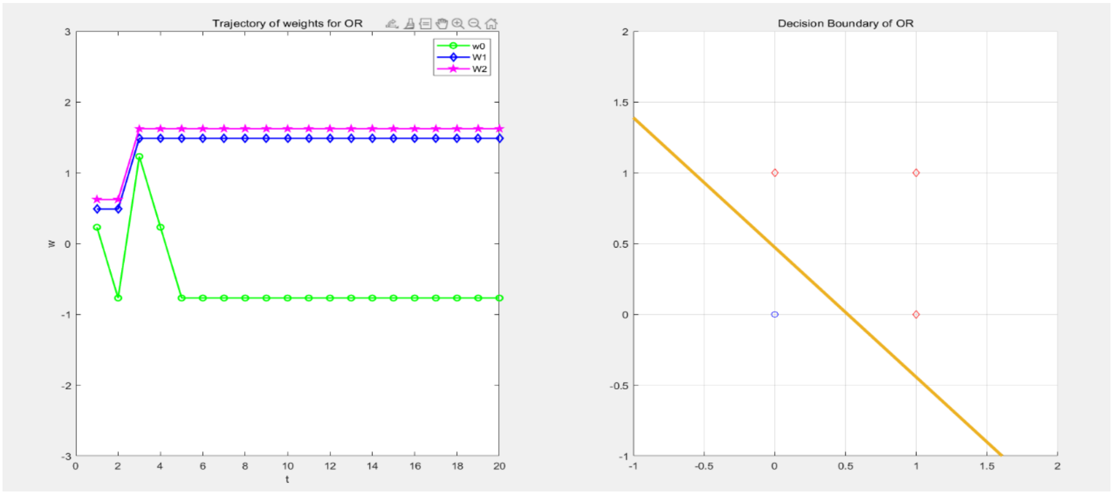


$(\eta = 0.5)$

2. OR

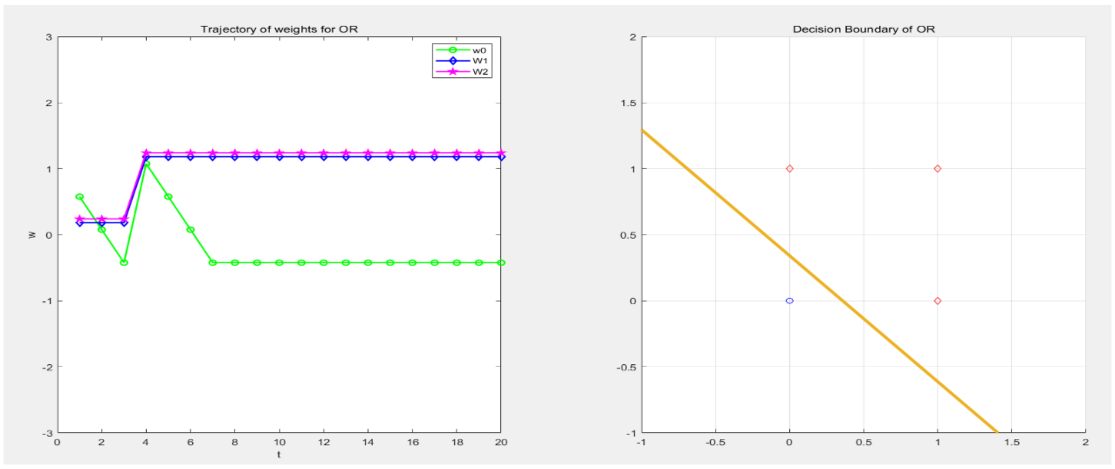
Input of OR: $x = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} d = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} w = \begin{bmatrix} 0.2316 & 0.4889 & 0.6241 \end{bmatrix}$

The trajectory of weights in this case is plotted in left, and the obtained decision boundary is plotted in right.



$(\eta = 1)$

We can easily find the epoch of $(\eta = 1)$ is about 5, When we change the learning rate= 0.5, the epoch of $(\eta = 0.5)$ is about 7.

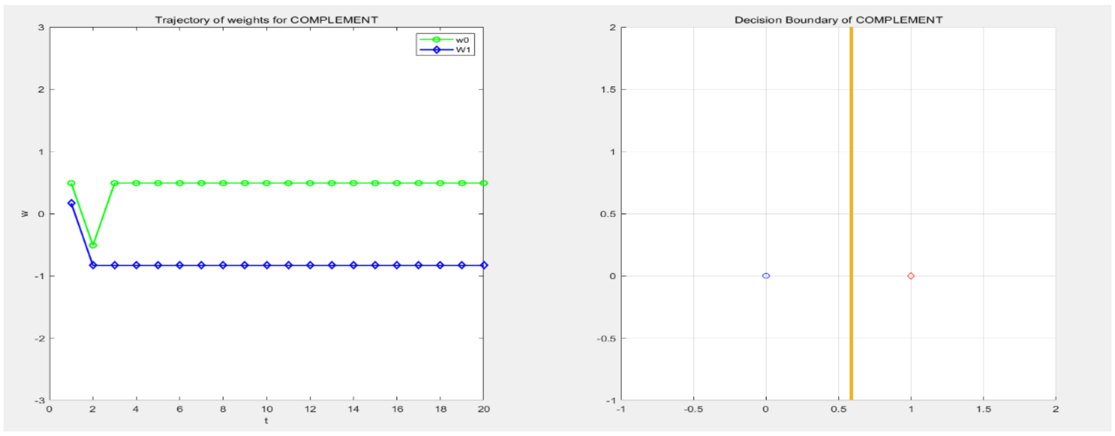


$(\eta = 0.5)$

3. COMPLEMENT

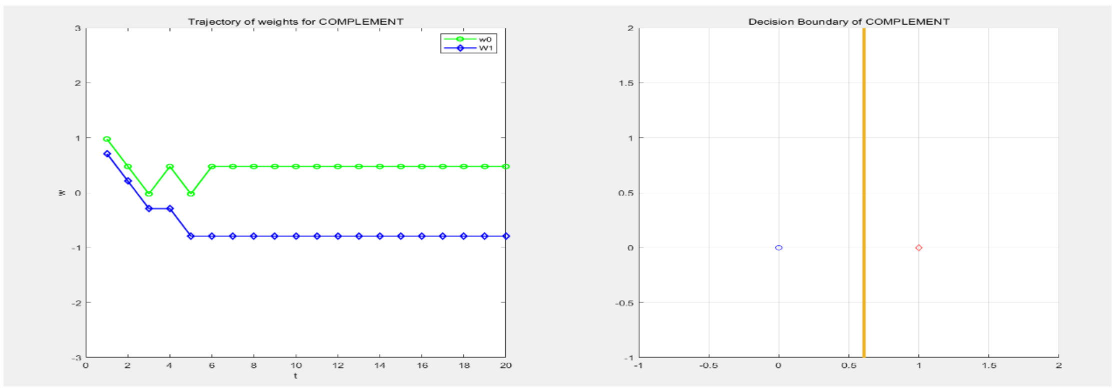
Input of COMPLEMENT: $x = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} d = \begin{bmatrix} 1 \\ 0 \end{bmatrix} w = \begin{bmatrix} 0.4899 & 0.1679 \end{bmatrix}$

The trajectory of weights in this case is plotted in left, and the obtained decision boundary is plotted in right.



$(\eta = 1)$

We can easily find the epoch of $(\eta = 1)$ is about 3, When we change the learning rate= 0.5, the epoch of $(\eta = 0.5)$ is about 7.

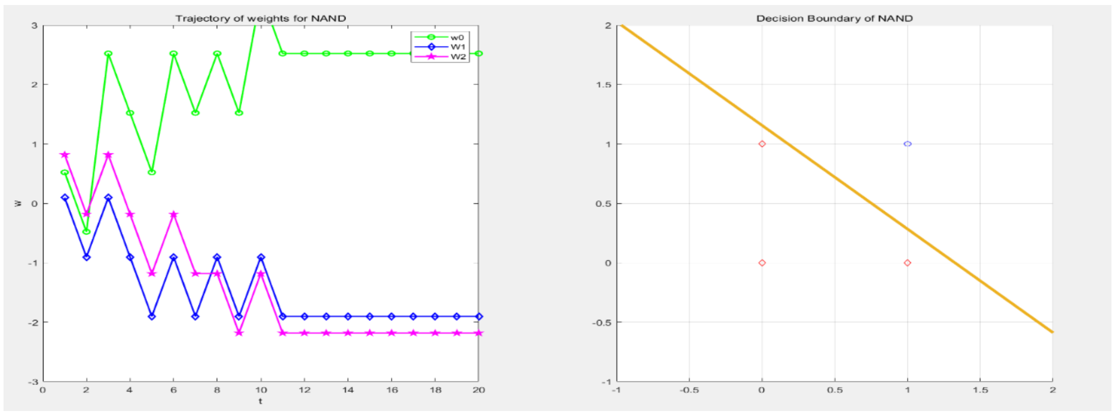


$(\eta = 0.5)$

4. NAND

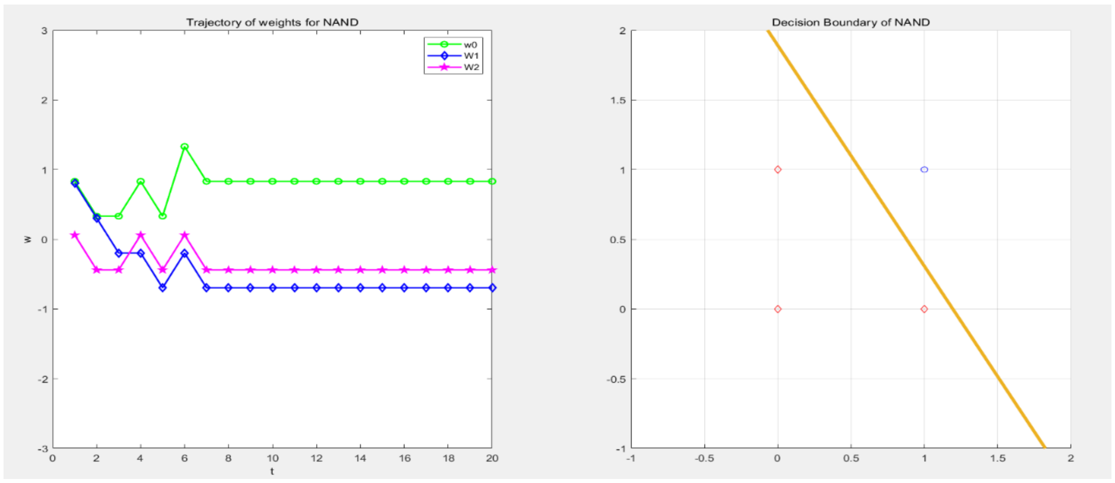
Input of NAND: $x = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} d = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} w = \begin{bmatrix} 0.2316 & 0.4889 & 0.6241 \end{bmatrix}$

The trajectory of weights in this case is plotted in left, and the obtained decision boundary is plotted in right.



$(\eta = 1)$

We can easily find the epoch of $(\eta = 1)$ is about 11, When we change the learning rate= 0.5, the epoch of $(\eta = 0.5)$ is about 7.

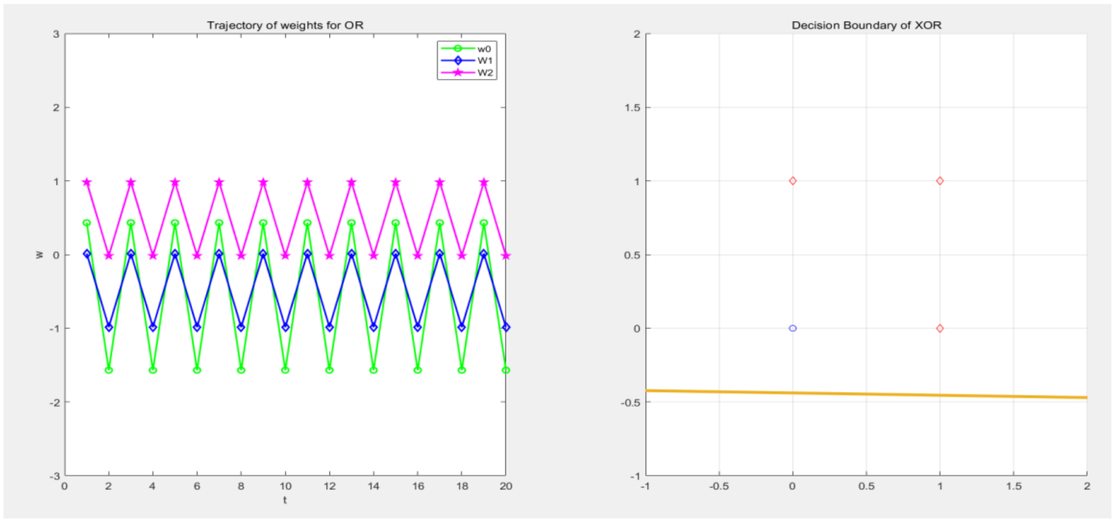


$(\eta = 0.5)$

In conclusion, with the increase of the learning rate, the weight vector converges become faster. However, for each model, it exist one best learning rate, which is essential for training. Just like the NAND model, it seems that useless if we increase learning rate.

c) XOR

When I apply the same perceptron in XOR function, I find the weight can not converge(in left) and no correct decision boundary can be contained(in right). This is because XOR function is not linearly separable.



Solution 4

a) Using the LLS method

For LLS method, the cost function is $E = \sum_{i=1}^{n} e(i)^2 = e^T e$

And the $e(i) = d(i) - y(i) \Rightarrow e = d - n$

Next, use the chain rule, we can get

$$\frac{\partial E}{\partial w} = -2e^T X$$

Let $\frac{\partial E}{\partial w} = 0$, we can get $w = \left(X^T X\right)^{-1} X^T d$

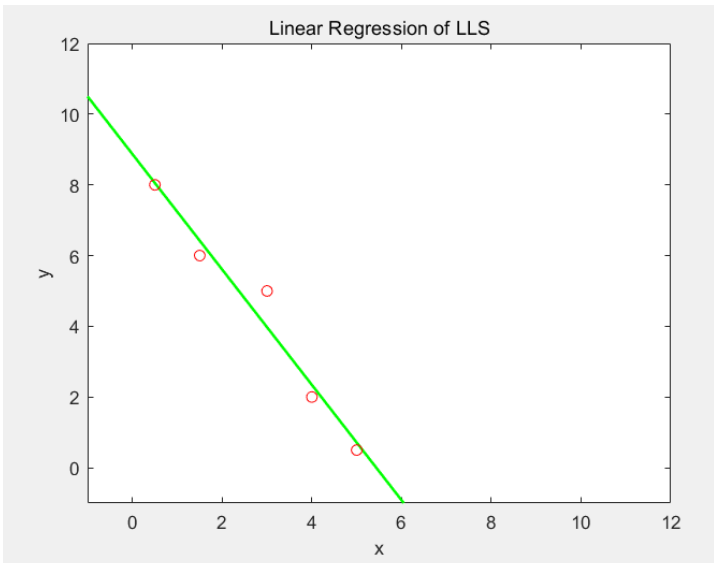
For this input of LLS:

$$x = \begin{bmatrix} 1 & 0.5 \\ 1 & 1.5 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{bmatrix} d = \begin{bmatrix} 8 \\ 6 \\ 5 \\ 2 \\ 0.5 \end{bmatrix}$$

From the above method, we can calculate the weight vector.

$$\begin{bmatrix} b \\ w \end{bmatrix} = \left(X^T X\right)^{-1} X^T d = \begin{bmatrix} 8.8684 \\ -1.6316 \end{bmatrix}$$

To sum up, the result of b is 8.8684 and w is -1.6316



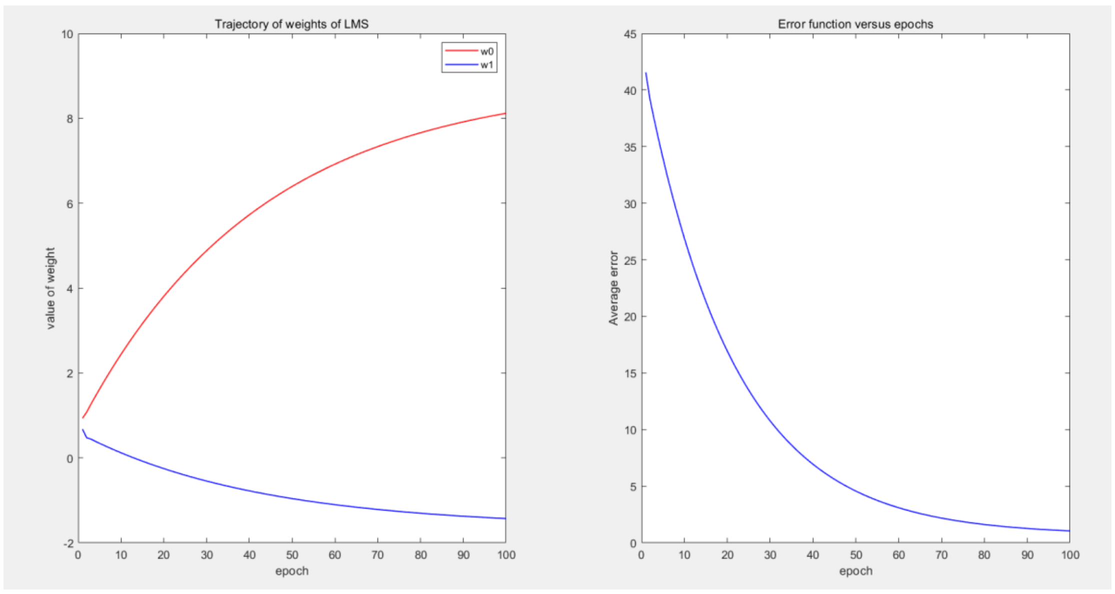
b) Compared to LLS, LMS method focus more on mean error. Cost function is $E = \frac{1}{2} e(n)^2$,

And the gradient of cost function is $\frac{\partial E}{\partial w} = -e(n)x(n)$

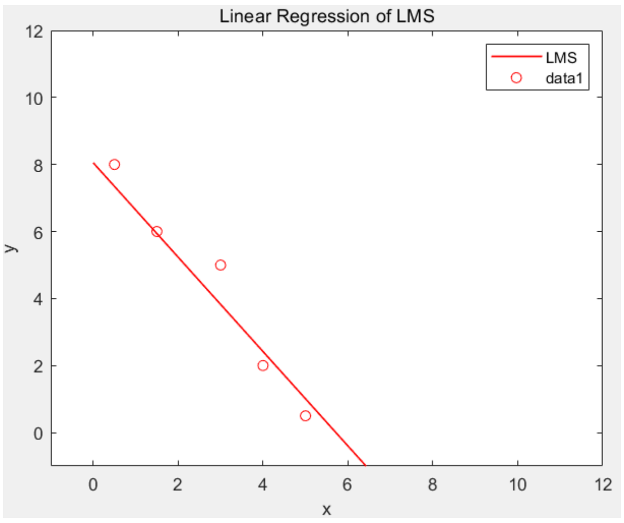
Applying steepest descent method, we have

$$w(n+1) = w(n) + \eta e(n)x(n)$$

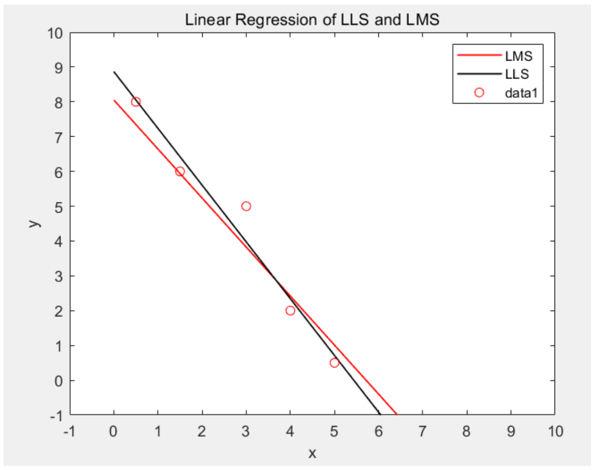
Input of LMS method is similar to LLS, as for initial weight, we set $\begin{bmatrix} 0.4897 \\ 0.3395 \end{bmatrix}$



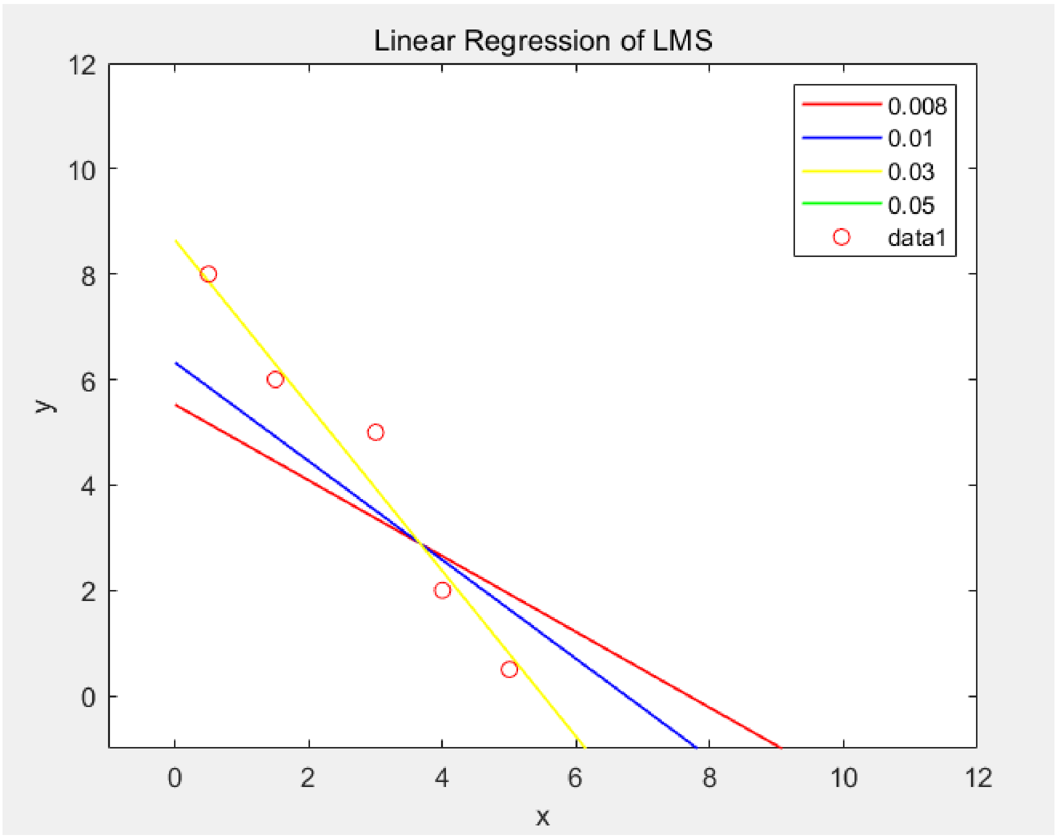
From this result, we can find that the weights and average error show a converging tendency after several learning steps and epochs. However, the weights can not converge to a certain value in 100 epochs.



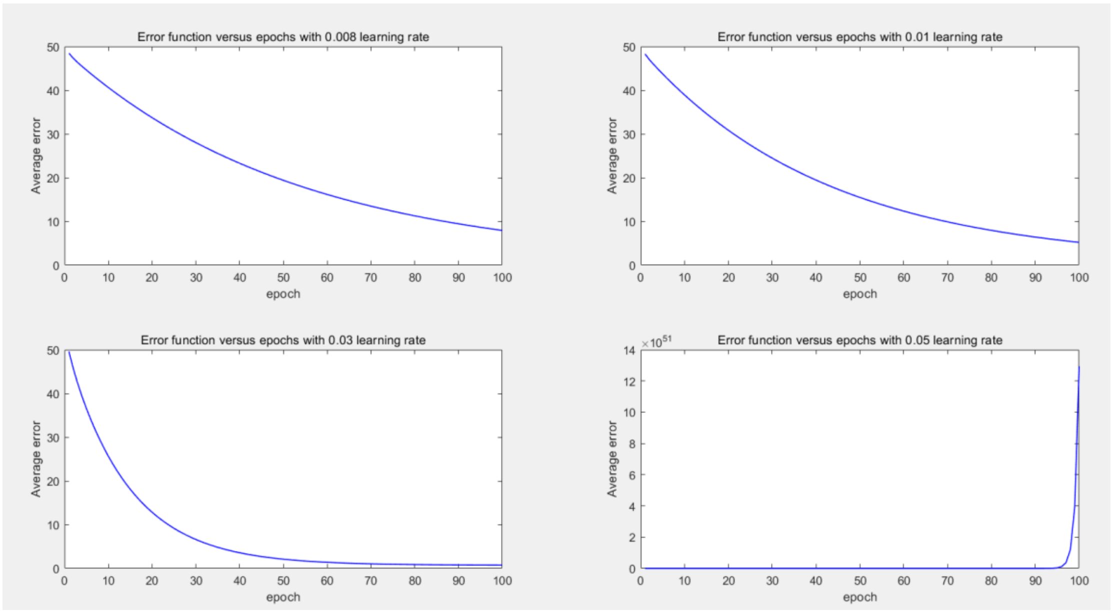
c) We can see clearly find that these two fitting results (both LLS and LMS) are very similar.



d) Using different learning rate, 0.008, 0.01, 0.03, 0.1, we can get different regression.



For the error function:



As shown in the experiments, small learning rate may lead to good performance. While the big learning rate may result in large errors. To sum up, it would be careful to choose right learning rate in different project.

Solution 5

From the above statement,

$$E = \frac{1}{2}e(n)^2 + \frac{1}{2}\lambda \|w(n)\|^2$$

$$e(n) = d(n) - w^T(n)x(n)$$

Firstly , we can get these derivation by using chain rule:

$$\frac{\partial e}{\partial w} = -x^T(n)$$

$$\frac{\partial E}{\partial w} = \lambda w(n) + e(n)\frac{\partial e}{\partial w} = \lambda w(n) - e(n)x^T(n)$$

Secondly, let $g(n) = \lambda w(n) - e(n)x^T(n)$

Lastly, put the value into the equation: $w(n+1) = w(n) - \eta g(n)$

We can get:

$$\begin{aligned} w(n+1) &= w(n) - \eta g(n) = w(n) - \eta\left[\lambda w(n) - e(n)x^T(n)\right] \\ &= (1-\eta\lambda)w(n) + \eta x(n)e(n) \end{aligned}$$

As desired.