ME 5404 Neuron Networks Homework #1

A0263252L Shen Xiaoting

Q1:

Considering the signal-flow graph of the perceptron of the figure, we can get the input vector:

$$x(n) = [+1, x_1(n), x_2(n), \cdots, x_m(n)]^T$$

the weight vector:

$$w(n) = [b_k, w_{k1}, w_{k2}, \cdots, w_{km}]^T$$

the induced local field:

$$v(n) = \sum_{i=1}^{m} w_{ki} x_i(n) \; + \; b_k$$

Considering the different activation functions, we can get different outputs.

(1) The activation function : $\varphi(v) = av + b$

Let $\varphi(v) = \xi$, we can get $av + b = \xi$, $v = \frac{\xi - b}{a}$

If $v > \frac{\xi - b}{a}$, it can be classified as class 1. If $v < \frac{\xi - b}{a}$, it can be classified as class 2. There is a hyper-plane decision boundary for classification.

(2) The activation function : $\varphi(v) = \frac{1}{1+e^{-2v}}$

Let $\varphi(v) = \xi$, we can get $\frac{1}{1+e^{-2v}} = \xi$. The figure of $\varphi(v)$ is shown in figure 1.1.

①$0 < \xi < 1, if\ v > -\frac{1}{2} ln(\frac{1}{\xi} - 1)$, it can be classified as class 1. If $v < -\frac{1}{2}\ln(\frac{1}{\xi} - 1)$, it can be classified as class 2. There is a hyper-plane decision boundary for classification.

②$\xi < 0$, $\varphi(v) > \xi$. There is no decision boundary for classification.

③$\xi > 1$, $\varphi(v) < \xi$. There is no decision boundary for classification.

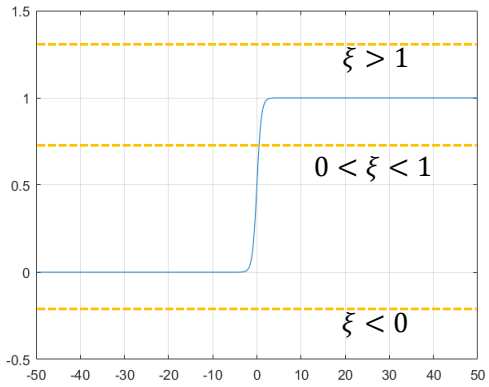


Figure 1.1 $\varphi(v) = \frac{1}{1+e^{-2v}}$

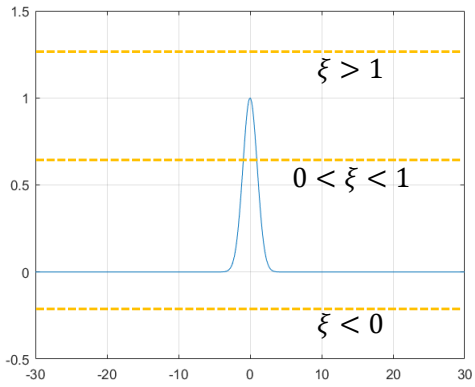


Figure 1.2 $\varphi(v) = e^{-\frac{v^2}{2}}$

(3) The activation function : $\varphi(v) = e^{-\frac{v^2}{2}}$

Let $\varphi(v) = \xi$, we can get $e^{-\frac{v^2}{2}} = \xi$. The figure of $\varphi(v)$ is shown in figure 1.2.

①$0 < \xi < 1, if\ -\sqrt{-2ln\xi} < v < \sqrt{-2ln\xi}$, it can be classified as class 1and othres can be classified as class 2. There are two hyper-plane decision boundaries for classification.

②$\xi < 0,\ \varphi(v) > \xi$. There is no decision boundary for classification.

③$\xi > 1,\ \varphi(v) < \xi$. There is no decision boundary for classification.

Q2:

The decision boundary of a linear separatable problem can be described as the equation

$$w_1x_1 + w_2x_2 + b = 0$$

Through the EXCLUSIVE OR table, we can get

$$w_1 \cdot 0 + w_2 \cdot 0 + b < 0 \quad (1)$$
$$w_1 \cdot 1 + w_2 \cdot 0 + b > 0 \quad (2)$$
$$w_1 \cdot 0 + w_2 \cdot 1 + b > 0 \quad (3)$$
$$w_1 \cdot 1 + w_2 \cdot 1 + b < 0 \quad (4)$$

Adding the inequalities (1) and (4), we can get

$$w_1 + w_2 + 2b < 0$$

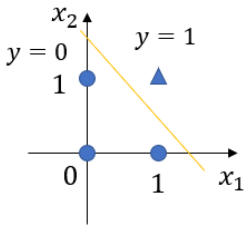
Adding the inequalities (2) and (3), we can get

$$w_1 + w_2 + 2b > 0$$

There is a contradiction of the above two inequalities. This is the proof that the XOR problem is not linearly separable.

Q3:

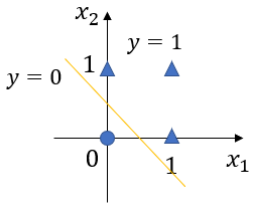
a) The input patterns belong to two classed are marked with circle and triangle. The decision boundary is straight yellow line by the following equations.



AND

Decision boundary: $x_1 + x_2 - 1.5 = 0$

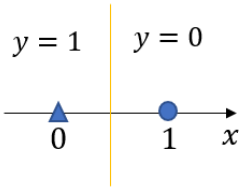
$w = [-1.5,1,1]$



OR

Decision boundary: $x_1 + x_2 - 0.5 = 0$

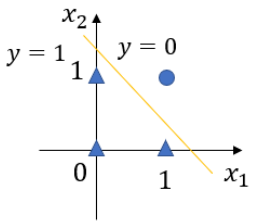
$w = [-0.5,1,1]$



COMPLEMENT

Decision boundary: $x_1 - 0.5 = 0$

$w = [-0.5,1]$



NAND

Decision boundary: $-x_1 - x_2 + 1.5 = 0$

$w = [1.5, -1, -1]$

b) We use the perceptron learning algorithm to get the decision boundary by iteration. We start with a randomly chosen weight vector $w(1) = [-0.5,1,-1]$ for logic functions AND, OR and NAND and weight vector $w = [-1.5,1]$ for logic function COMPLEMENT. We can update the weight vector with the equation

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

Where $\eta = 1$ and $d = \begin{cases} 1, & if\ x\ belongs\ to\ class\ c1 \\ 0, & if\ x\ belongs\ to\ class\ c2 \end{cases}$

Then we can draw the decision boundary of these logic function and plot the trajectories of the weights for each case in figure 3.1-3.4. The weight vector will be invariant in several epochs. After the learning process, we will get the decision boundary which is different from off-line calculation, but it can also classify the points correctly.

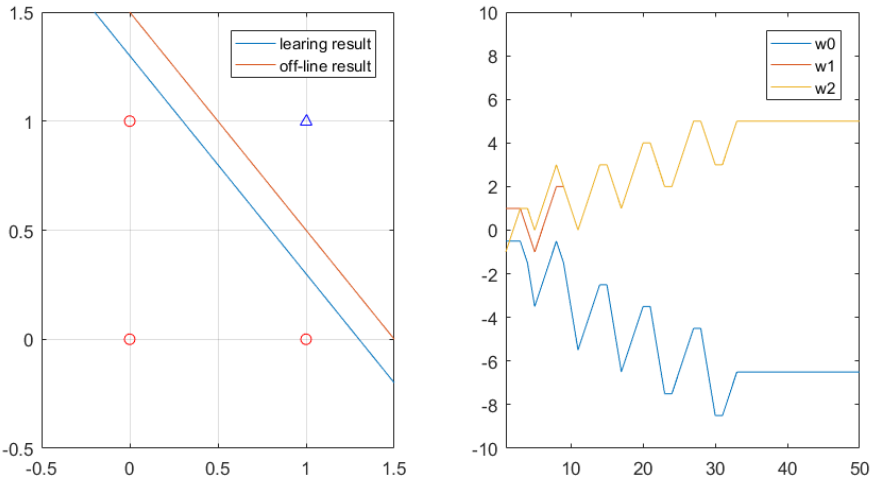


Figure3.1 The decision boundary and the trajectories of the weights for logic function AND

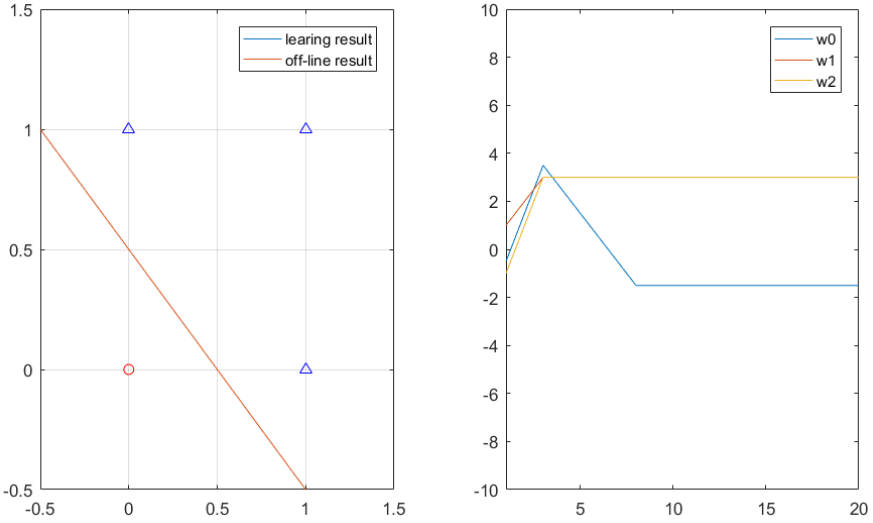


Figure3.2 The decision boundary and the trajectories of the weights for logic function OR

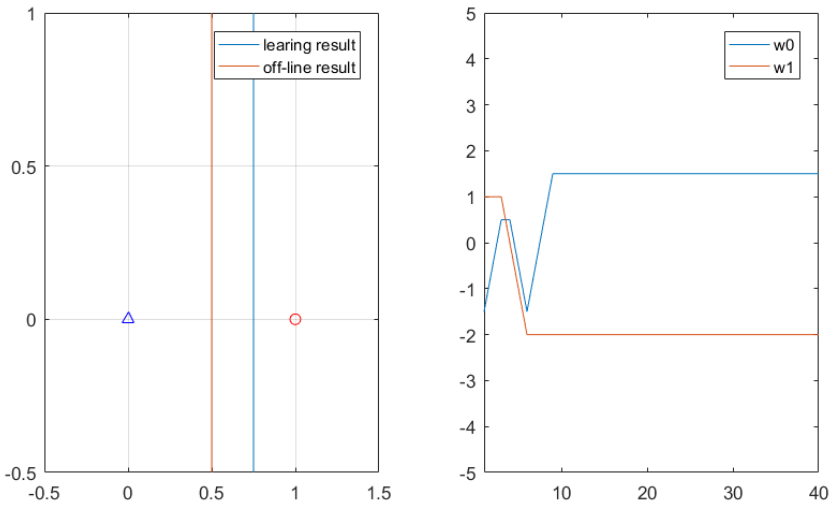


Figure3.3 The decision boundary and the trajectories of the weights for logic function COMPLEMENT

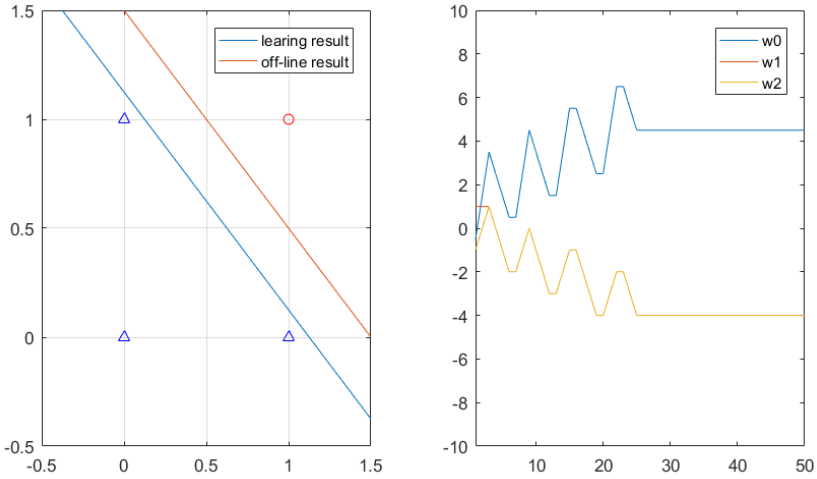


Figure3.4 The decision boundary and the trajectories of the weights for logic function NAND

We tried different learning rate and plot the error ($e(n) = d(n) - y(n)$). We listed the logic function AND as an example. If the learning rate is small ($\eta = 0.04$) , the error will goes to 0 in 46 epochs. If the learning rate is larger ($\eta = 1$), the error will go to 0 in 33 epochs. If we change the learning rate to 100, the error will go to 0 in 28 epochs. The results are shown in figure 3.5-3.7.

The other logic functions have similar results to the logic functions AND. If the learning rate is small, the error will go to zero slower. If the learning rate is large, the error will go to zero faster.

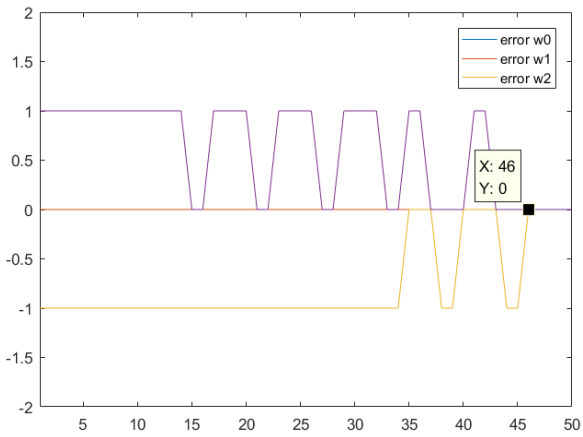


Figure3.5 The iteration error for logic function AND by learning rate $\eta = 0.04$

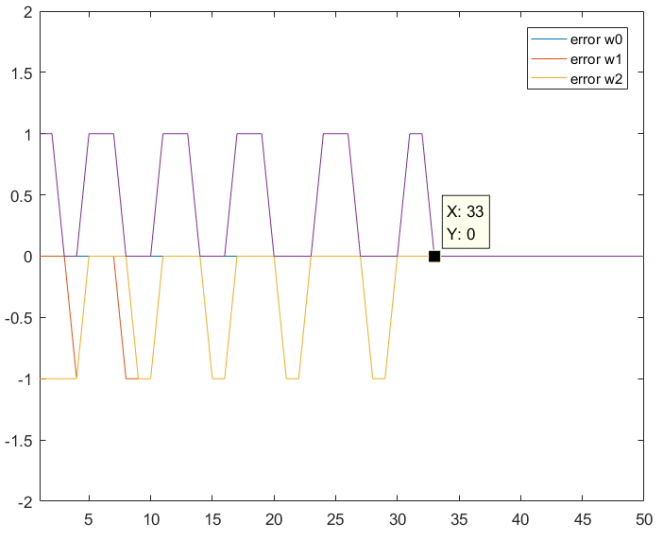


Figure3.6 The iteration error for logic function AND by learning rate $\eta = 1$

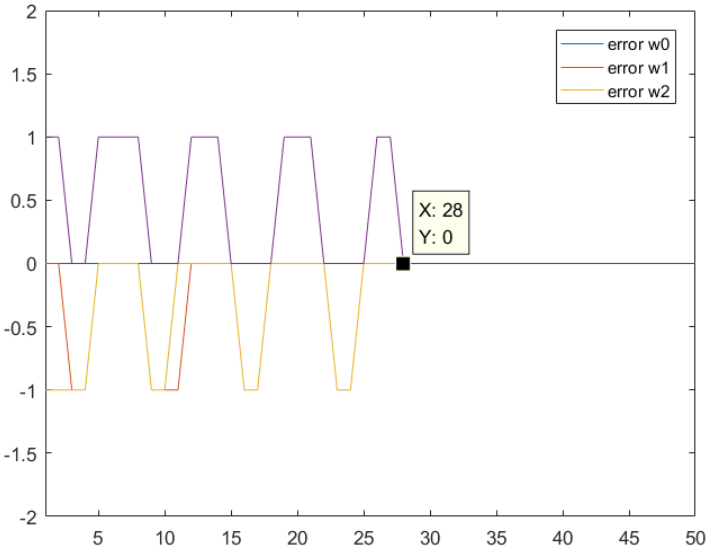


Figure3.7 The iteration error for logic function AND by learning rate $\eta = 100$

c) If we apply the perceptron learning algorithm to the XOR function, we can not get proper

decision boundary and the weight vector will not stop updating. The XOR logic function can not be classified by one line and we have proved it by question 2. The result is shown in figure 3.8.

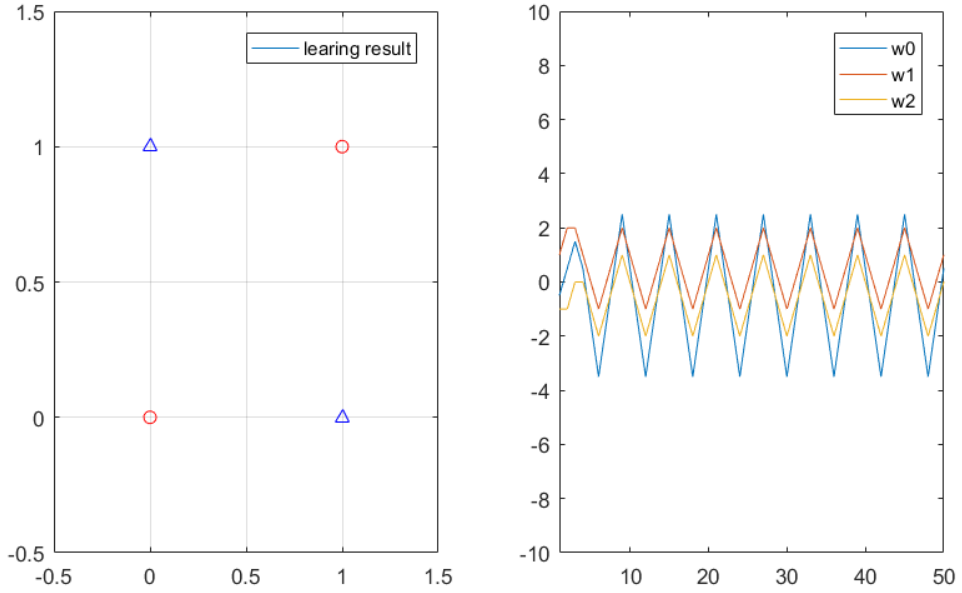


Figure3.8 The decision boundary and the trajectories of the weights for logic function XOR

Q4:

a) According to the single linear neuron figure, we can get the induced local field equation

$$v = b + wx$$

Through the given pairs, we can get the regression matrix

$$X = \begin{bmatrix} x(1)^T \\ x(2)^T \\ \vdots \\ x(6)^T \end{bmatrix}$$

The standard linear least squares method, we can get the weight vector by the equation $w = (X^T X)^{-1} X^T d$. The parameter of $b = 0.3873, w = 1.6094$. The result is shown in figure 4.1.

b) and c)

We use the least mean square method to do linear regression problem. The results of LLS and LMS are shown in figure 4.1. It is clear that the lines are very close. After 100 epochs, learning rate is 0.01, the weights can not converge to constants. If I extend to 500 epochs, we can get the results in figure 4.2. It is obvious that the value of w and b will converge to $b = 0.3873, w = 1.6094$ which is same as the results by using the LLS method.

d)

We change the learning rate $\eta = 0.0001$, $\eta = 0.01$, $\eta = 0.03$ and get the results from figure 4.2-4.4. The larger learning rate, the weight will converge in 100 epochs. The smaller learning rate, the weight will converge in 300 epochs.

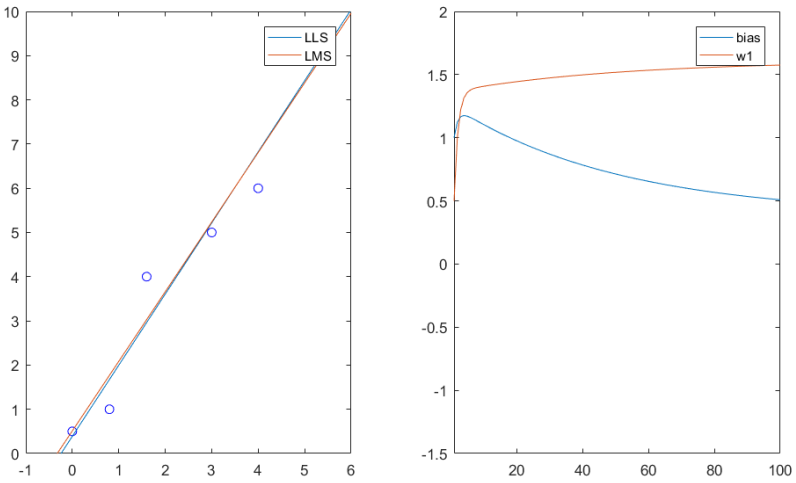


Figure 4.1 The linear regression result using LLS and LMS method and the w and b iteration process by 100 epochs

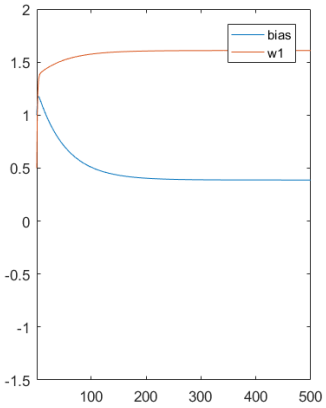


Figure 4.2 The change of parameters of b and w by 500 epochs with learning rate 0.01

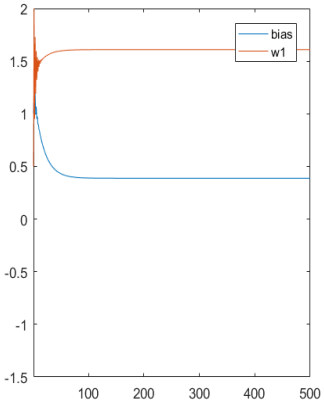


Figure 4.3 The change of parameters of b and w by 500 epochs with learning rate 0.03

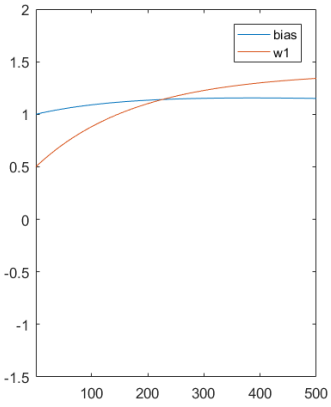


Figure 4.4 The change of parameters of b and w by 500 epochs with learning rate 0.0001

Q5:

The cost function

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

Since $Y = w^T x$, Let $\frac{\partial J(w)}{\partial w} = 0$

$$\frac{\partial J(w)}{\partial w} = -2r(i)(d(i) - w^T x)x = 0$$

$$w^* = (x^{-1})^T d(i)^T$$