類神經

LMS Learning

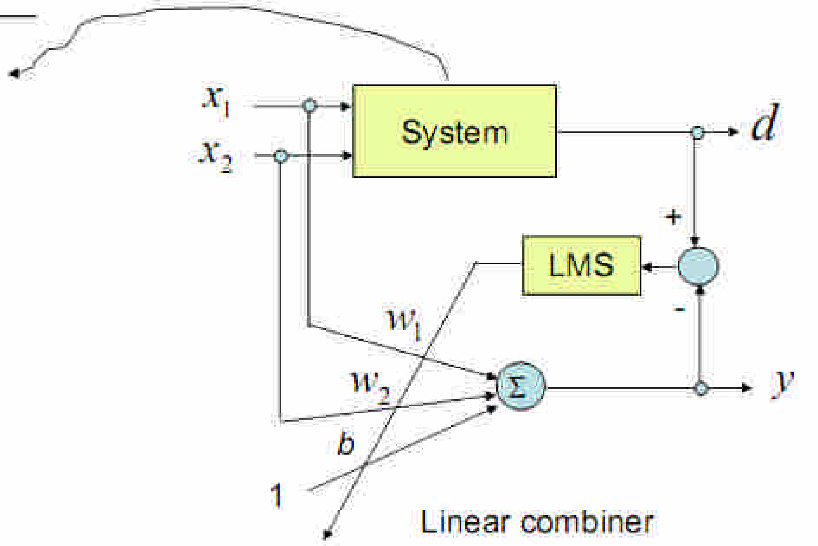
授課教授：趙春棠 教授
學生：黃秉宏
學號：49422019
班級：機電四甲

第一題題目

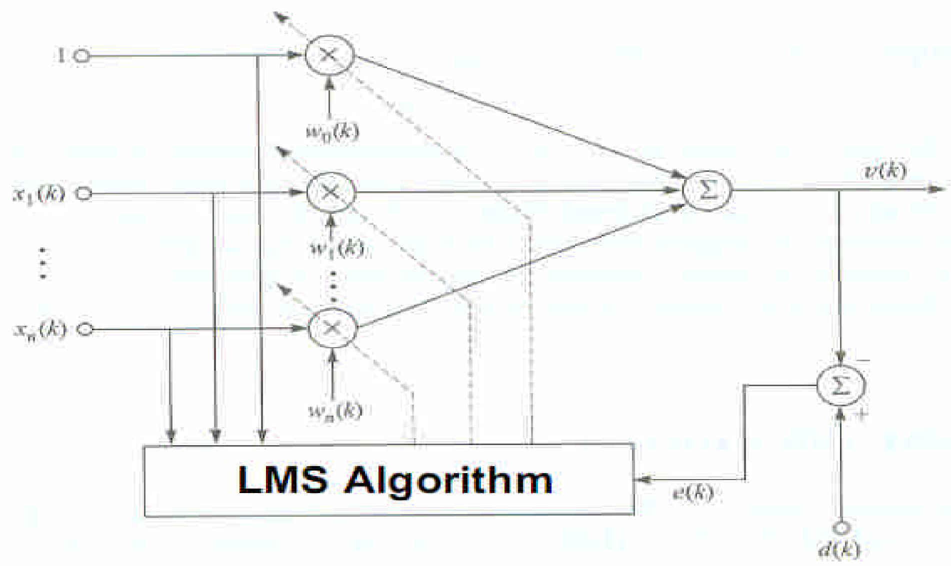
Problem 1

Q1: Use LMS to learn the following weights and simulate it by Matlab

x_1	x_2	d
1	2	2
2	5	1
2	3	2
2	2	2
3	4	1
3	5	3
4	6	2
5	5	3
5	6	4
5	7	3
6	8	4
7	6	2
8	4	4
8	9	3
9	8	4



(1) LMS 架構圖 :



(2) 數學方程式：

$$\nabla_w J(w) \approx \frac{1}{2}\frac{\partial e^2(k)}{\partial w}\bigg|_{w=w(k)}$$
$$= \frac{1}{2}\frac{\partial}{\partial w(k)}\left[d^2(k) - 2d(k)x^T(k)w(k) + w^T(k)x(k)x^T(k)w(k)\right]$$
$$= -d(k)x(k) + x(k)x^T(k)w(k) = -d(k)x(k) + w^T(k)x(k)x(k)$$
$$= -[\underbrace{d(k) - w^T(k)x(k)}_{e(k)}]x(k) = -e(k)x(k)$$

Therefore, the learning rule for updating the weights using the steepest-descent method as

$$w(k+1) = w(k) + \mu[-\nabla_w J(w)] = w(k) + \mu e(k)x(k)$$

Where μ is learning rate with $0 < \mu < \frac{2}{\lambda_{max}}$ or $0 < \mu < \frac{2}{\text{trace}\{C_x\}}$

(3) 設定步驟：

Step 1. Set $k = 1$, initialize the synaptic weight vector $w(k = 1)$, and select values for μ_0 and τ.

Step 2. Compute the learning rate parameter as

$$\mu(k) = \frac{\mu_0}{1 + k/\tau}$$

τ : searching time constant

100 <= τ <= 500

Step 3. Compute the error

$$e(k) = d(k) - \sum_{h=1}^{n} w_h(k)x_h(k)$$

Step 4. Update the synaptic weights $w_i(k+1) = w_i(k) + \mu(k)e(k)x_i(k)$, for $i = 1, 2, \ldots, n$.

Step 5. If convergence is achieved, *stop*; else set $k \leftarrow k + 1$, then go to step 2.

(4) Matlab 程式：

```
clc
clear all
x=zeros(2,15);
d=zeros(1,15);
x(1,:)=[1 2 2 2 3 3 4 5 5 5 6 7 8 8 9];
x(2,:)=[2 5 3 2 4 5 6 5 6 7 8 6 4 9 8];
d=[2 1 2 2 1 3 2 3 4 3 4 2 4 3 4];
Cx=x*x'/15;
lr0=0.9/max(eig(Cx));
tau=200;
w=zeros(2,1);
b=zeros(1,1);
t=0;
cw=10000;
a=0:cw-1;
for j=1:cw
    t=t+1;
    lr(t)=lr0/(1+t/tau);
    for i=1:15
        y=x(1,i)*w(1)+x(2,i)*w(2)+b;
        yo(j)=y;
        e=d(i)-y;
        eo(j)=e;
        w=w+lr(t)*e*x(:,i);
        wo(:,j)=w;
        b=b+lr(t)*e;
        bo(j)=b;
    end
end
subplot(2,2,1)
plot(a,yo)
axis([-inf,inf,-7,7])
xlabel('次數');
title('實際輸出值');
subplot(2,2,2)
plot(a,eo)
axis([-inf,inf,-7,7])
xlabel('次數');
title('實際誤差值');

```

說明：

Cx=x*x'/15;

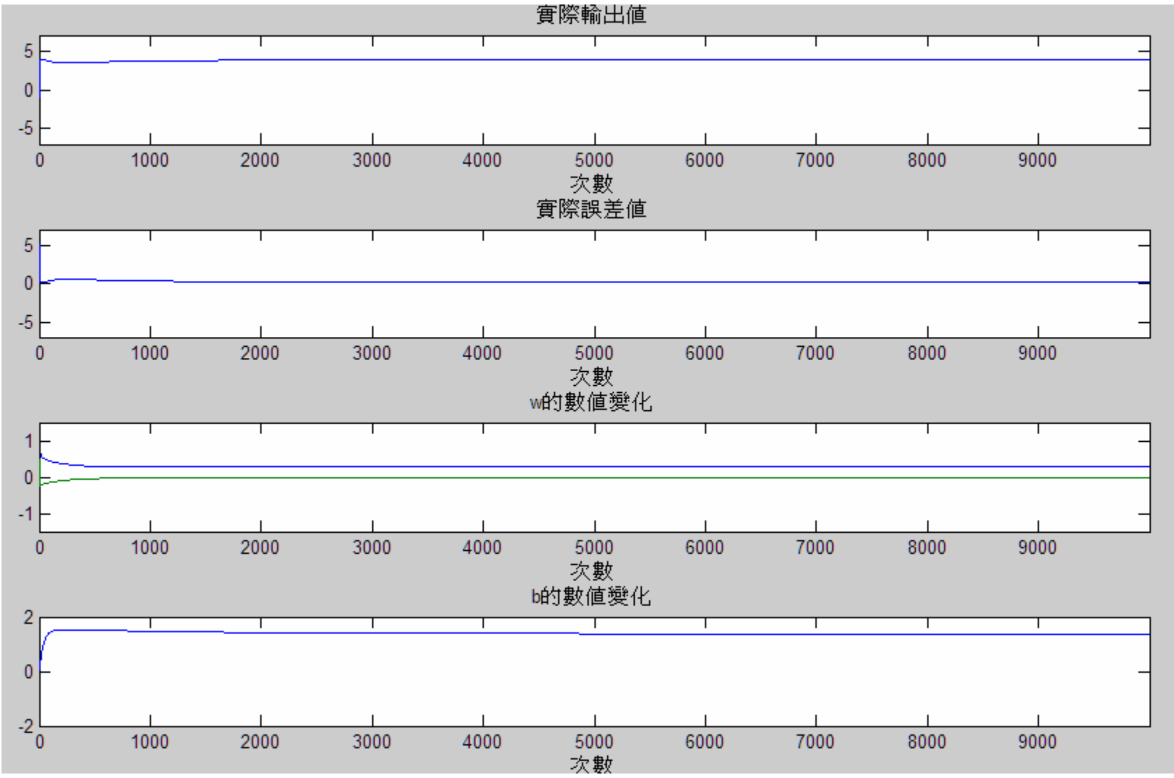
x*x'是求出共變異矩陣，而除以15則是要算出平均值，此項計算便於下個步驟求出eigen value所做的矩陣計算

lr0=0.9/max(eig(Cx));

lr0做的計算則是初始學習速率，eig(Cx)則是將共變異矩陣求出eigen value，再利用max函數求得最大之eigen value。lr0所計算出來之數值為下列公式之u_0。

$$\mu(k) = \frac{\mu_0}{1 + k/\tau}$$

(5) 模擬結果：

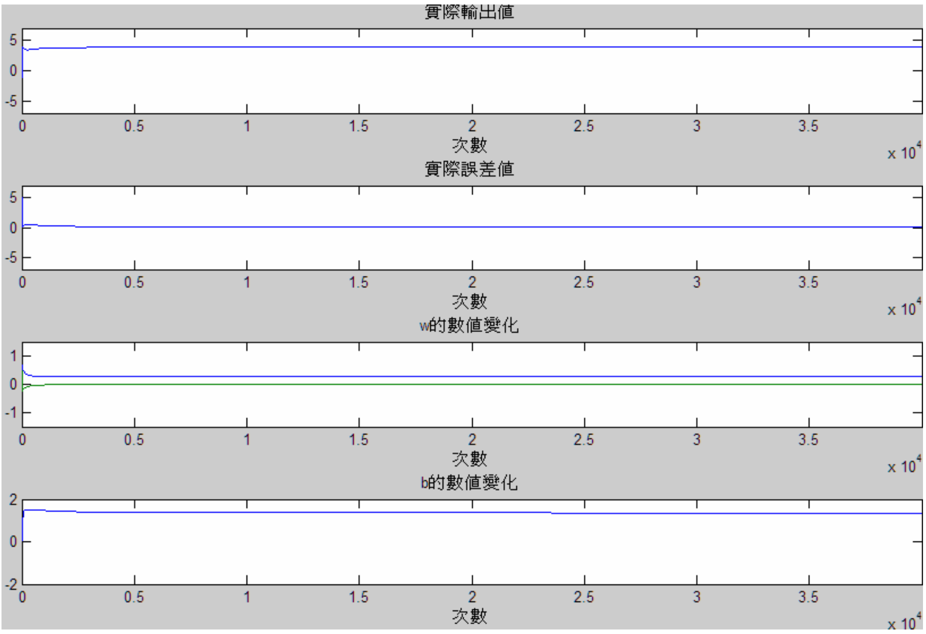



```
Command Window
>> w

w =

    0.2839
   -0.0052

>> b

b =

    1.3688

>> e

e =

    0.1236

>> y

y =

    3.8764

>> 
```

(6) 討論：

1. 學習次數對輸出結果之影響(上述測試以 10000 次學習討論)

學習次數調整為 40000 次之結果




```
Command Window
>> w

w =

    0.2856
   -0.0044

>> b

b =

    1.3573

>> e

e =

    0.1091

>> y

y =

    3.8909

>> |
```

結論：學習次數越高，可讓 w 和 b 值學習更精準。但對收斂速度無幫助。

2. 學習率、tau 值的關係：

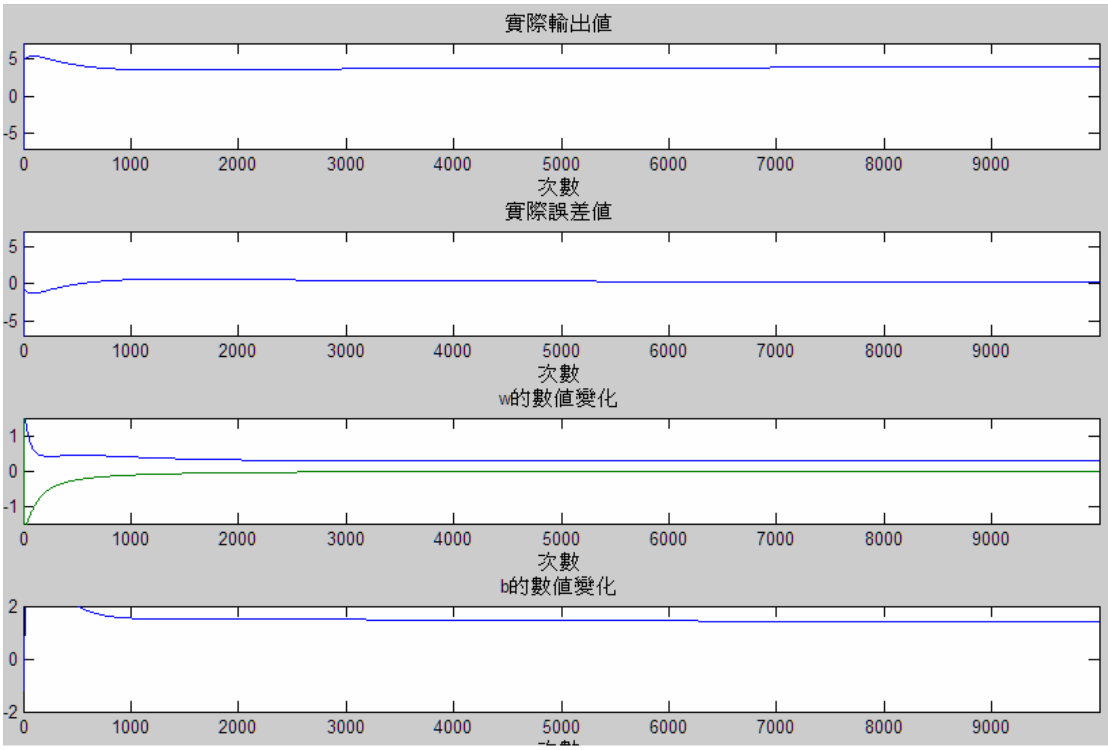
$$\mu(k) = \frac{\mu_0}{1 + k/\tau}$$,

Where μ is learning rate with $0 < \mu < \frac{2}{\lambda_{max}}$ or $0 < \mu < \frac{2}{\mathrm{trace}\{C_x\}}$

$$w(k+1) = w(k) + \mu[-\nabla_w J(w)] = w(k) + \mu e(k) x(k)$$

所求得之 lr0 為 u_0，w 每次更新為前一次數值+學習率*誤差值*輸入值，但因學習率會受到 k 的執行次數影響，而降低學習率，因此 w 所更新的值也越來越小，而 tau 也會直接影響至學習率，tau 越大，學習率越大，學習率越大，w 的更新值越大，也較難收斂。

下列數值以tau=500，lr0設為lr0=1.5/max(eig(Cx))



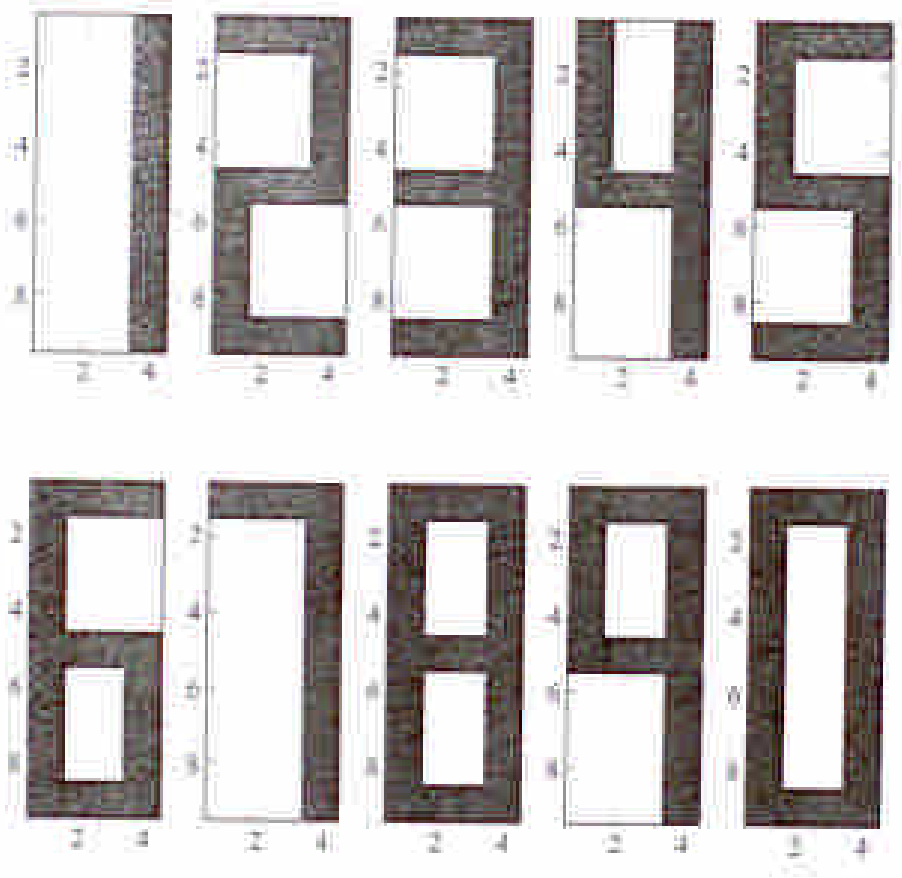
由此圖可看出，tau及lr0越大，學習率較大，需較長時間才能收斂。至於學習率為何會定義右邊公式原因：

$$0 < \mu < \frac{2}{\lambda_{\max}}$$

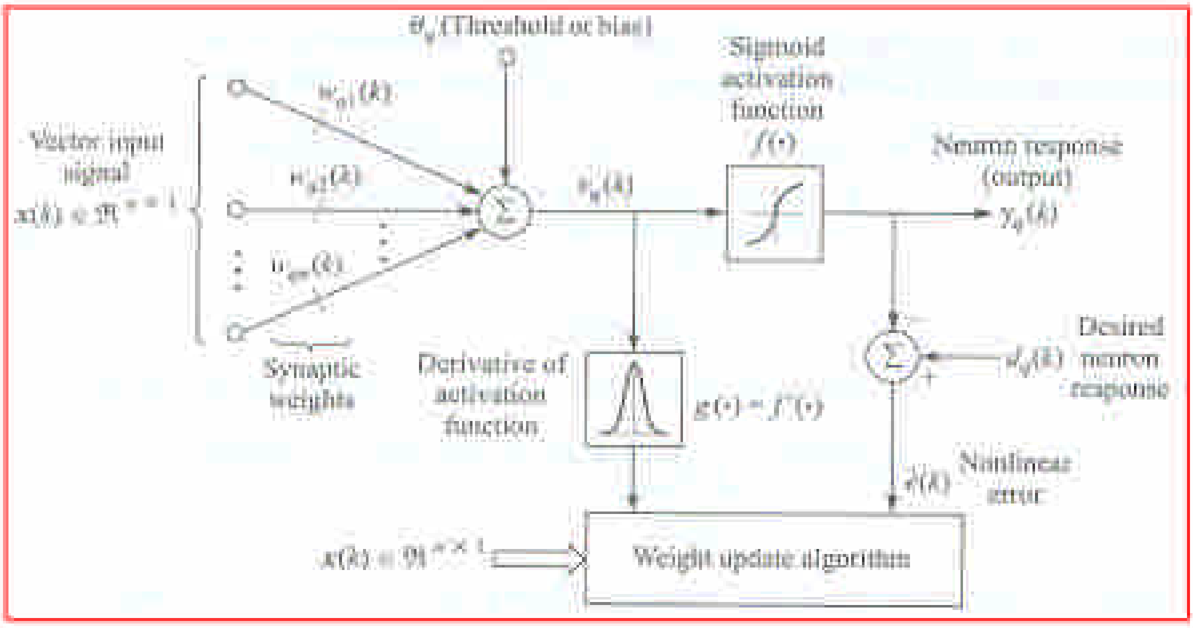
U的大小會決定權重更新步伐，如果u太小則權重改變量也較小，太大則是造成目標發生振盪，因此Haykin建議u在右上之範圍為最好。

Problem 2

Q2: Use the simple perception with the sigmoid function to learn the digital 0~9 shown in right figure. (Use one neuron)



(1) 架構圖：



(2) 數學方程式：

Step 1. feed forward computation

$$\bar{e}_q(k) = d_q(k) - y_q(k)$$

$$y_q(k) = f[v_q(k)] = f\left[\sum_{j=1}^{n} x_j(k) w_{qj}(k) + \theta_q\right]$$

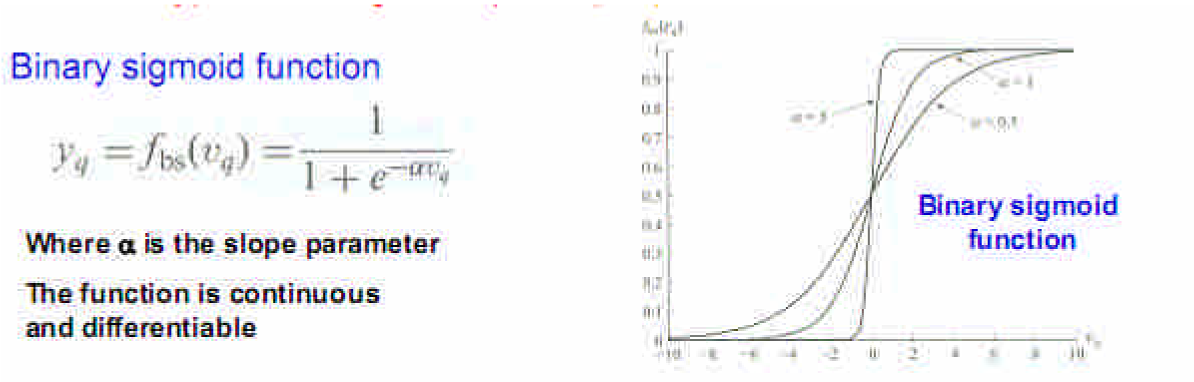
Step 2. feedback learning

$$\boldsymbol{w}_q(k+1) = \boldsymbol{w}_q(k) + \mu\alpha\bar{e}_q(k)[1 - y_q^2(k)]\boldsymbol{x}(k)$$

or

$$w_{qj}(k+1) = w_{qj}(k) + \mu\alpha\bar{e}_q(k)[1 - y_q^2(k)]x_j(k)$$

where $j = 1, 2, \ldots, n$.



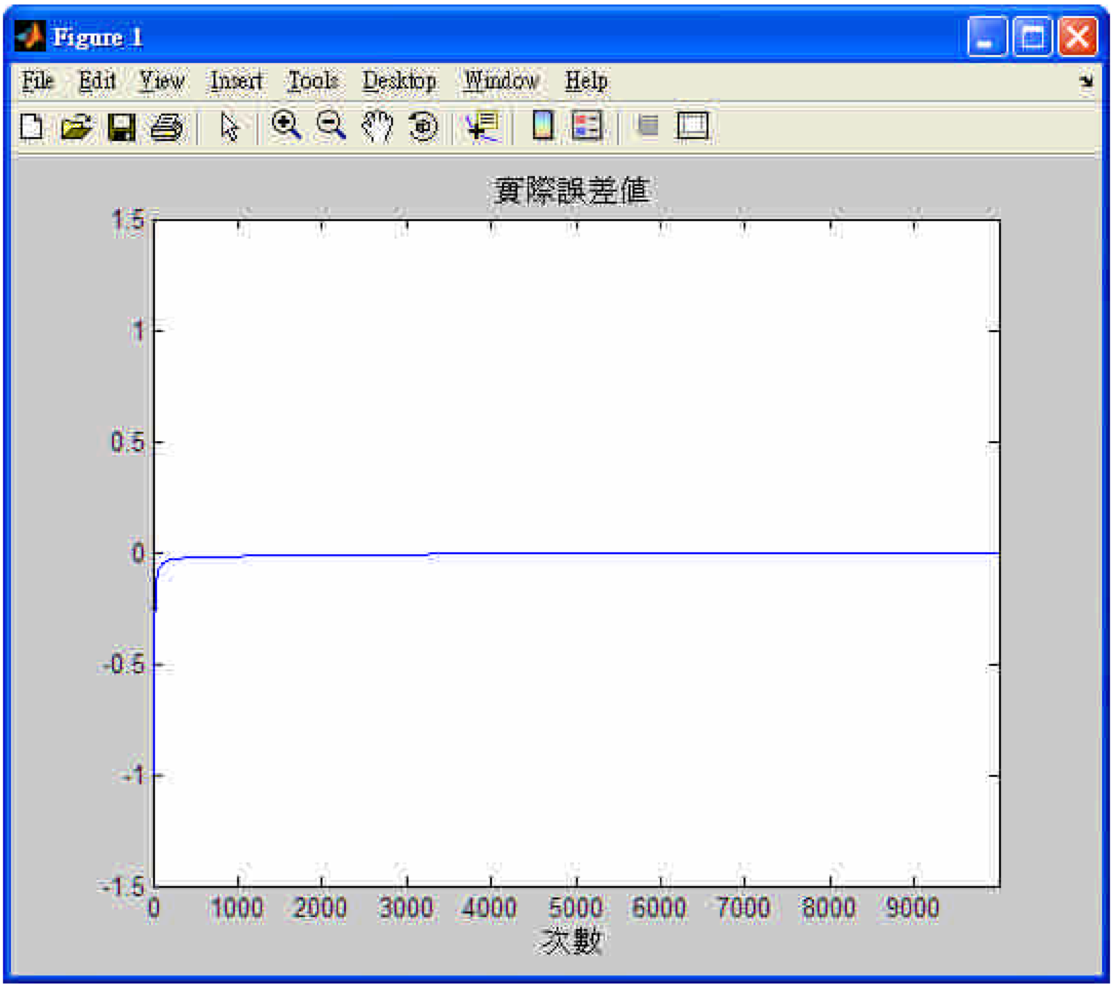
(3) Matlab 程式：

```
clc;
clear all
x=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1;
   1 0 0 0 1 1 1 1 1 1 1 0 0 0 1 0 0 0 0 1 1 0 0 0 1 0 0 0 0 1 1 1 1 1 1 0 0 0 0 1;
   1 0 0 0 1 0 0 0 0 1 1 0 0 0 1 0 0 0 0 1 1 0 0 0 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1;
   1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1;
   1 1 1 1 1 0 0 0 0 1 1 0 0 0 1 0 0 0 0 1 1 0 0 0 1 0 0 0 0 1 1 0 0 0 1 1 1 1 1 1;
   1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0 0 1 1 0 0 0 1 0 0 0 0 1 1 0 0 0 1 1 1 1 1 1;
   1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1;
   1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0 0 1 1 0 0 0 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1;
   1 1 1 1 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1;
   1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1];
w=rand(40,1);
sq=0.05;
d=[0.9 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1 0];
a=0.5;u=0.25;
mh=10000;
xa=0:mh-1;
for j=1:mh
    for i=1:10
        v=x(i,:)*w+sq;
        y=1/(1+exp(-a*v));yo(j)=y;yt(i)=y;
        e=d(i)-y;eo(j)=e;et(i)=e;
        w=w+u*a*e*(1-(y^2))*x(i,:)';
    end
end
plot (xa,eo,'b')
axis([-inf,inf,-1.5,1.5])
xlabel('次數');
title('實際誤差值');
```

說明：x 為 10*40 矩陣，設 i 為圖片數，則 x(I,:)為第 i 張圖片之輸入訊號，而 w 為 40*1 之矩陣，sq 為 sita q、d 為期望值、a 為 afa、u 為

學習率，學習次數為 10000 次。運算式採用 Binary sigmoid function 方式計算。

(4)　模擬結果：



實際誤差圖

```
Command Window
>> d

d =

  Columns 1 through 8

    0.9000    0.8000    0.7000    0.6000    0.5000    0.4000    0.3000    0.2000

  Columns 9 through 10

    0.1000         0

>> yt

yt =

  Columns 1 through 8

    0.9000    0.7999    0.7040    0.6000    0.4972    0.4025    0.2978    0.1949

  Columns 9 through 10

    0.1020    0.0028

>>
```

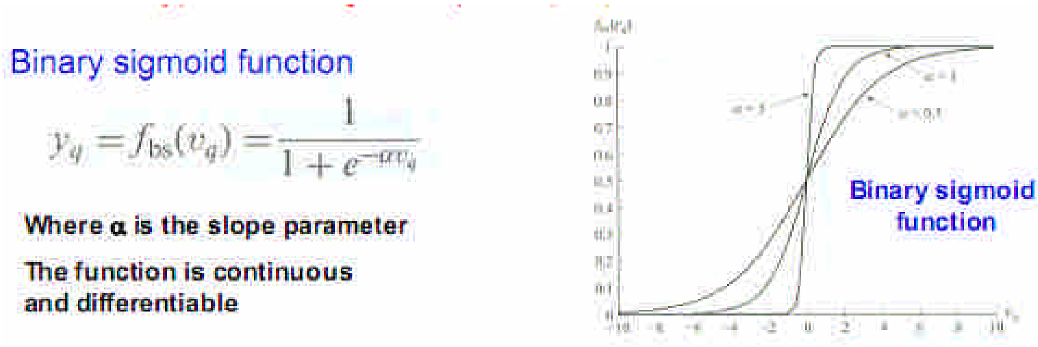
d 為期望值、yt 為記錄最後輸出值

eo

	9990	9991	9992	9993	9994	9995	9996	9997	9998	9999	10000
1	-0.0028428	-0.0028425	-0.0028422	-0.0028419	-0.0028416	-0.0028413	-0.002841	-0.0028407	-0.0028403	-0.00284	-0.002839
2											
3											
4											

誤差值在接近 10000 次運算時的數值

(5) 討論：



Step 1. feed forward computation

$$\bar{e}_q(k) = d_q(k) - y_q(k)$$

$$y_q(k) = f[v_q(k)] = f\left[\sum_{j=1}^{n} x_j(k) w_{qj}(k) + \theta_q\right]$$

Step 2. feedback learning

$$w_q(k+1) = w_q(k) + \mu\alpha\bar{e}_q(k)[1 - y_q^2(k)]x(k)$$

or

$$w_{qj}(k+1) = w_{qj}(k) + \mu\alpha\bar{e}_q(k)[1 - y_q^2(k)]x_j(k)$$

where $j = 1, 2, \ldots, n$.

1. afa 值的影響：

由Binary sigmoid function看出afa越小，則exp(-a*v)越小，但輸出值y會越大。afa越小則會影響到w的更新權值。

2. sita q 的影響：

sita q 越小，vq 也會變小，而 vq 變小後，經過 Binary sigmoid function 運算後，進而影響至輸出值 y 會變大。

3. u 的影響：

u越小則會影響到w的更新權值。

實驗：

將afa設為0.1、sita q=0.05、u=0.01，可從下圖看出收斂速度明顯慢了好幾倍。