BPNN

0056C203 廖育賢

# objective-the problem and the purpose

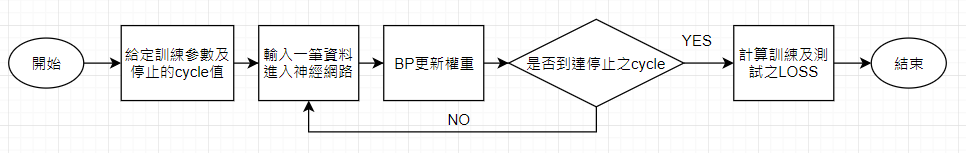
以倒傳遞神經網路學習函數z=5\*sin(pi\*x^2)\*sin(2\*pi\*y)+1。

## method

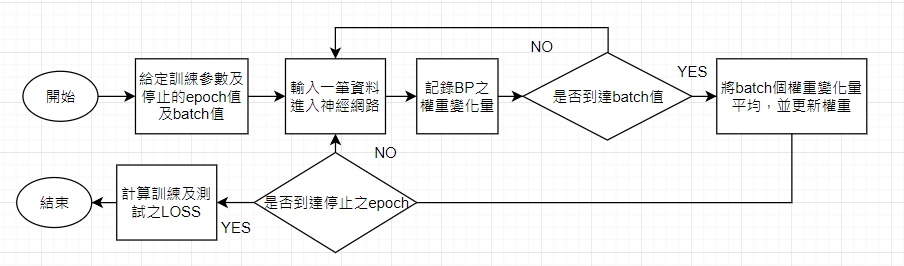
首先隨機取樣400筆資料，分為300筆訓練資料及100筆測試資料，以倒傳遞演算法更新權重，訓練完後，比較Sequential Learning 及Batch Learning 的差異，及神經元個數、學習速率、慣量等參數對於訓練結果的影響。

## program flow chart

### Sequential Mode:



### batch mode:



## equation

### back propagation

wji (n+1)=wji (n)+Δwji , Δwji (n)= η\*δj (n)\* yi (n) ,δj (n)= -

for j is output unit : - =

for j is hidden unit : - , k is output uint

### generalized delta rule

Δwji (n)=η\*δj\*yi (n)+α\*Δwji (n-1)

### batch mode

# simulation results

## program codes

function [ z ] = f( x,y )

z=5\*sin(pi\*x^2)\*sin(2\*pi\*y)+1;

end

function [ y] = normolized\_data(x,a,b)

% enter a matrix x ,size(x) is m by 1

% b:after normolized max value

% a:after normolized min value

y=(x-min(x))\*(b-a)/(max(x)-min(x))+a;

end

function [ y ] = sigmoid( x )

y=1/(1+exp(-x));

end

%========================BPNN.m======================================

format long;

load('data.mat')

d=normolized\_data(data(:,3),0.2,0.8);% map data -->range 0.2~0.8

% Input Data-->1(bias\_input),x,y,z(normolized)

x=zeros(400,4);

x(:,1)=1;

x(:,2:3)=data(1:400,1:2);

x(:,4)=d(1:400);

%======================= Parameters ==============================

iter=200000; %iteration(cycle)

learning\_rate=0.1;

alpha=0.8; %momentum

input\_cell=3; % 1(bias\_input),x,y

hidden\_cell=20;

output\_cell=1; % z=f(x,y)

%hidden layer to input layer:w\_h2i

w\_h2i=rand(hidden\_cell,input\_cell)-0.5; %-0.5~0.5

%output layer to hidden layer :w\_o2h

w\_o2h=rand(output\_cell,hidden\_cell)-0.5; %-0.5~0.5

%========================= Loss ================================

error=zeros(1,300);

E=zeros(1,300);

E\_ave=zeros(1,iter);

%=========================Temp array=================================

yout=zeros(output\_cell);

dw\_o2h=zeros(output\_cell,hidden\_cell);

w\_o2h\_new=zeros(output\_cell,hidden\_cell);

dw\_h2i=zeros(hidden\_cell,input\_cell );

w\_h2i\_new=zeros(hidden\_cell,input\_cell);

%=========================Training=============================

for cycle=1:iter

for num=1:300 % #1~#300 data

%=======================Forward===========================

% the value of x0\*w0+x1\*w1+...

v\_hidden=zeros(hidden\_cell,1); %hidden layer

v\_output=zeros(output\_cell,1); %output layer

% input layer to hidden layer

for j=1:hidden\_cell

for i=1:input\_cell

v\_hidden(j,1)=v\_hidden(j,1)+x(num,i)\*w\_h2i(j,i);

end

end

% hidden layer to output layer

for j=1:output\_cell

for i=1:hidden\_cell

v\_output(j,1)=v\_output(j,1)+sigmoid(v\_hidden(i,1))\*w\_o2h(j,i);

end

yout(j)=sigmoid(v\_output(j,1));

end

error(1,num)=d(num,1)-yout(j);

E(num)=1/2\*(error(1,num))^2;

%===================== Back Propagation ==============================

for j=1:output\_cell

delta\_o2h=error(j,num)\*(yout(j)\*(1-yout(j)));

for i=1:hidden\_cell

dw\_o2h(j,i)=alpha\*dw\_o2h(j,i)+learning\_rate\*delta\_o2h\*sigmoid(v\_hidden(i,1));

w\_o2h\_new(j,i)=w\_o2h(j,i)+dw\_o2h(j,i);

end

end

for j=1:hidden\_cell

delta\_h2i=sigmoid(v\_hidden(j,1))\*(1-sigmoid(v\_hidden(j,1)))\*delta\_o2h\*w\_o2h(1,j);

for i=1:input\_cell

dw\_h2i(j,i)=alpha\*dw\_h2i(j,i)+learning\_rate\*delta\_h2i\*sigmoid(x(num,i));

w\_h2i\_new(j,i)=w\_h2i(j,i)+dw\_h2i(j,i);

end

end

w\_o2h=w\_o2h\_new;

w\_h2i=w\_h2i\_new;

end

E\_ave(cycle)=mean(E);

end

%%

%======================== Predict data ==========================

% output cell data=1; % z=f(x,y)

predict\_data=zeros(100,1);

for num=301:400

% the value of x0\*w0+x1\*w1+...

v\_hidden=zeros(hidden\_cell,1); %hidden layer

v\_output=zeros(output\_cell,1); %output layer

% input layer to hidden layer

for j=1:hidden\_cell

for i=1:input\_cell

v\_hidden(j,1)=v\_hidden(j,1)+x(num,i)\*w\_h2i(j,i);

end

end

% hidden layer to output layer

for j=1:output\_cell

for i=1:hidden\_cell

v\_output(j,1)=v\_output(j,1)+sigmoid(v\_hidden(i,1))\*w\_o2h(j,i);

end

yout(j)=sigmoid(v\_output(j,1));

predict\_data(num-300,1)=yout(j);

end

error(1,num)=d(num,1)-yout(j);

E(num)=1/2\*(error(1,num))^2;

end

%%

figure(1)

subplot(211),plot(1:cycle,E\_ave),

title(['Training Loss:',num2str(E\_ave(cycle)) ,' Hidden cell :',num2str(hidden\_cell),' \eta:',num2str(learning\_rate),' \alpha:',num2str(alpha)]);

xlabel('cycle'),ylabel('E ave')

subplot(212),stem(301:400,E(301:400),'.'),

title(['Predict Loss :',num2str(mean(E(301:400)))]);

xlabel('number of data'),ylabel('E')

suptitle(['E predict - E train = ',num2str(mean(E(301:400))-E\_ave(cycle))]);

figure(2)

subplot(121)

a=data(301:400,1);

b=data(301:400,2);

c=normolized\_data(predict\_data(1:100,1),min(data(:,3)),max(data(:,3)));% z--> map to original range

scatter3(a,b,c,'.','r');

xlabel('x'),ylabel('y'),zlabel('z'),title('100 predict data');

subplot(122)

a=data(1:300,1);

b=data(1:300,2);

c=data(1:300,3);

scatter3(a,b,c,'.','b')

xlabel('x'),ylabel('y'),zlabel('z'),title('300 training data')

figure(3)

a=data(301:400,1);

b=data(301:400,2);

c=data(301:400,3);

scatter3(a,b,c,'.','r');

xlabel('x'),ylabel('y'),zlabel('z');

title('100 testing true data')

%========================BPNN\_epoch.m======================================

format long;

load('data.mat')

d=normolized\_data(data(:,3),0.2,0.8);% map data -->range 0.2~0.8

% Input Data-->1(bias\_input),x,y,z(normolized)

x=zeros(400,4);

x(:,1)=1;

x(:,2:3)=data(1:400,1:2);

x(:,4)=d(1:400);

%======================= Parameters ==============================

batch=300;

iter=300000; %iteration(epoch)

learning\_rate=0.2;

alpha=0.9; % momentum

input\_cell=3; % 1(bias\_input),x,y

hidden\_cell=15;

output\_cell=1; % z=f(x,y)

%hidden layer to input layer:w\_h2i

w\_h2i=rand(hidden\_cell,input\_cell)-0.5; %-0.5~0.5

%output layer to hidden layer :w\_o2h

w\_o2h=rand(output\_cell,hidden\_cell)-0.5; %-0.5~0.5

%========================= Loss ================================

error=zeros(1,300);

E=zeros(1,300);

E\_ave=zeros(1,iter);

%=========================Temp array=================================

yout=zeros(output\_cell);

dw\_o2h=zeros(output\_cell,hidden\_cell);

dw\_o2h\_epoch=zeros(output\_cell,hidden\_cell);

dw\_h2i=zeros(hidden\_cell,input\_cell );

dw\_h2i\_epoch=zeros(hidden\_cell,input\_cell);

%=========================Training=============================

for epoch=1:iter

for num=1:batch % #1~#batch data

%=======================Forward===========================

% the value of x0\*w0+x1\*w1+...

v\_hidden=zeros(hidden\_cell,1); %hidden layer

v\_output=zeros(output\_cell,1); %output layer

% input layer to hidden layer

for j=1:hidden\_cell

for i=1:input\_cell

v\_hidden(j,1)=v\_hidden(j,1)+x(num,i)\*w\_h2i(j,i);

end

end

% hidden layer to output layer

for j=1:output\_cell

for i=1:hidden\_cell

v\_output(j,1)=v\_output(j,1)+sigmoid(v\_hidden(i,1))\*w\_o2h(j,i);

end

yout(j)=sigmoid(v\_output(j,1));

end

error(1,num)=d(num,1)-yout(j);

E(num)=1/2\*(error(1,num))^2;

%===================== Back Propagation ==============================

for j=1:output\_cell

delta\_o2h=error(j,num)\*(yout(j)\*(1-yout(j)));

for i=1:hidden\_cell

dw\_o2h(j,i)=alpha\*dw\_o2h(j,i)+learning\_rate\*delta\_o2h\*sigmoid(v\_hidden(i,1));

%w\_o2h\_new(j,i)=w\_o2h(j,i)+dw\_o2h(j,i);

end

end

for j=1:hidden\_cell

delta\_h2i=sigmoid(v\_hidden(j,1))\*(1-sigmoid(v\_hidden(j,1)))\*delta\_o2h\*w\_o2h(1,j);

for i=1:input\_cell

dw\_h2i(j,i)=alpha\*dw\_h2i(j,i)+learning\_rate\*delta\_h2i\*sigmoid(x(num,i));

%w\_h2i\_new(j,i)=w\_h2i(j,i)+dw\_h2i(j,i);

end

end

dw\_o2h\_epoch = dw\_o2h\_epoch + dw\_o2h;

dw\_h2i\_epoch = dw\_h2i\_epoch + dw\_h2i;

end

dw\_o2h\_epoch = dw\_o2h\_epoch / batch;

dw\_h2i\_epoch = dw\_h2i\_epoch / batch;

w\_o2h = w\_o2h + dw\_o2h\_epoch;

w\_h2i = w\_h2i + dw\_h2i\_epoch;

E\_ave(epoch)=mean(E);

end

%%

%======================== Predict data ==========================

% output cell data=1; % z=f(x,y)

predict\_data=zeros(100,1);

for num=301:400

% the value of x0\*w0+x1\*w1+...

v\_hidden=zeros(hidden\_cell,1); %hidden layer

v\_output=zeros(output\_cell,1); %output layer

% input layer to hidden layer

for j=1:hidden\_cell

for i=1:input\_cell

v\_hidden(j,1)=v\_hidden(j,1)+x(num,i)\*w\_h2i(j,i);

end

end

% hidden layer to output layer

for j=1:output\_cell

for i=1:hidden\_cell

v\_output(j,1)=v\_output(j,1)+sigmoid(v\_hidden(i,1))\*w\_o2h(j,i);

end

yout(j)=sigmoid(v\_output(j,1));

predict\_data(num-300,1)=yout(j);

end

error(1,num)=d(num,1)-yout(j);

E(num)=1/2\*(error(1,num))^2;

end

%%

figure(1)

subplot(211),plot(1:epoch,E\_ave),

title(['Training Loss:',num2str(E\_ave(epoch)) ,' Hidden cell :',num2str(hidden\_cell),' \eta:',num2str(learning\_rate),' \alpha:',num2str(alpha)]);

xlabel('epoch'),ylabel('E ave')

subplot(212),stem(301:400,E(301:400),'.'),

title(['Predict Loss :',num2str(mean(E(301:400)))]);

xlabel('number of data'),ylabel('E')

suptitle(['E predict - E train = ',num2str(mean(E(301:400))-E\_ave(epoch))]);

figure(2)

subplot(121)

a=data(301:400,1);

b=data(301:400,2);

c=normolized\_data(predict\_data(1:100,1),min(data(:,3)),max(data(:,3)));% z--> map to original range

scatter3(a,b,c,'.','r');

xlabel('x'),ylabel('y'),zlabel('z'),title('100 predict data');

subplot(122)

a=data(1:300,1);

b=data(1:300,2);

c=data(1:300,3);

scatter3(a,b,c,'.','b')

xlabel('x'),ylabel('y'),zlabel('z'),title('300 training data')

figure(3)

a=data(301:400,1);

b=data(301:400,2);

c=data(301:400,3);

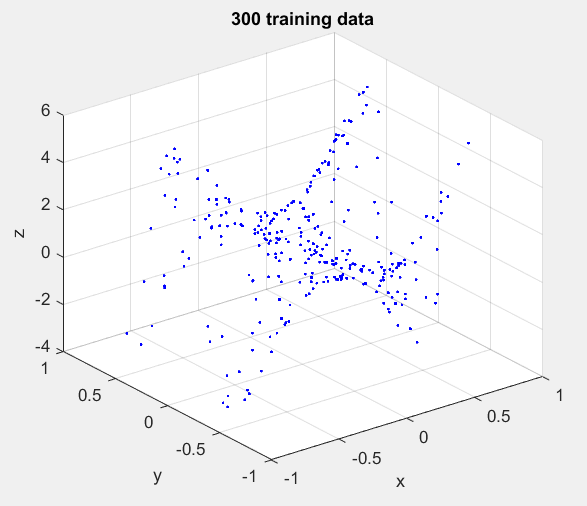
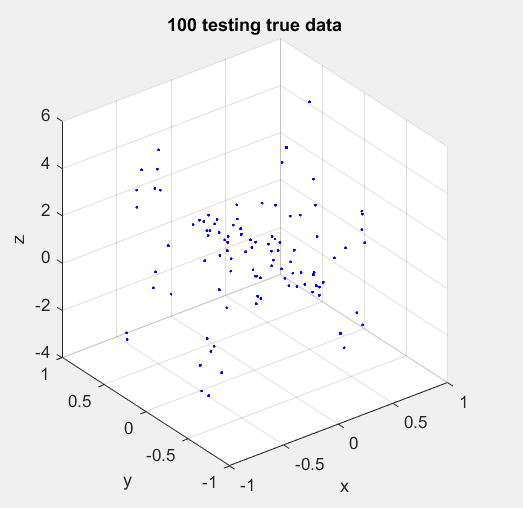
scatter3(a,b,c,'.','r');

xlabel('x'),ylabel('y'),zlabel('z');

title('100 testing true data')

## graph

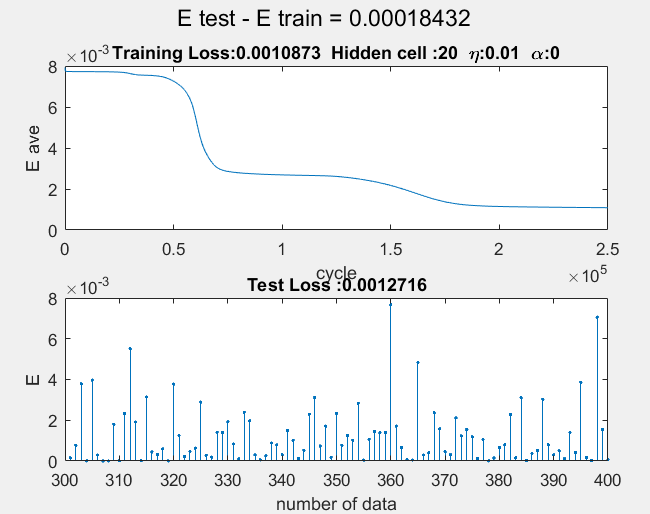
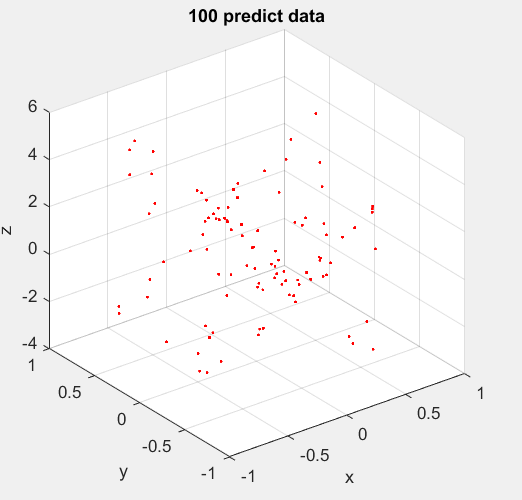
### 400 data (300 training & 100 testing):

圖一

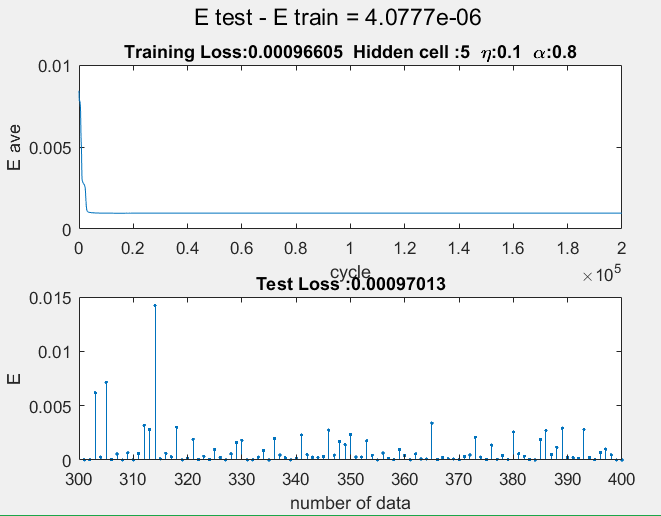
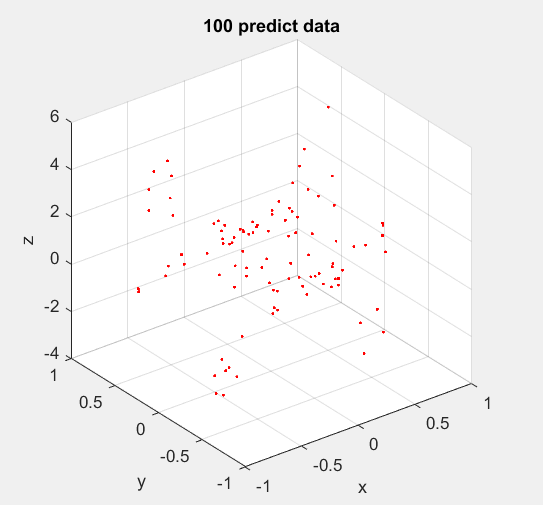
### Sequential learning:

#### ( hidden cell , learning rate , momentum )= ( 20 ,0.01 , 0 )

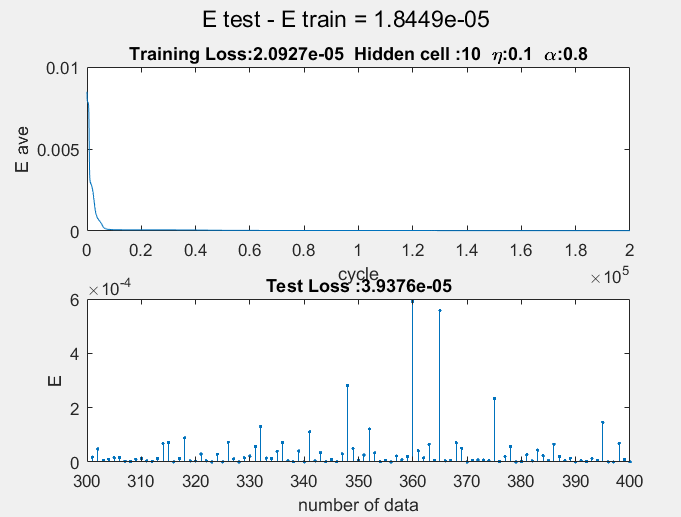
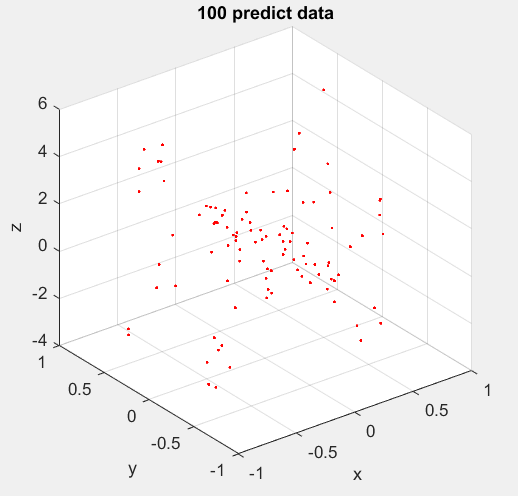
圖二

#### ( hidden cell , learning rate, momentum )=( 5 ,0.1 , 0.8 )

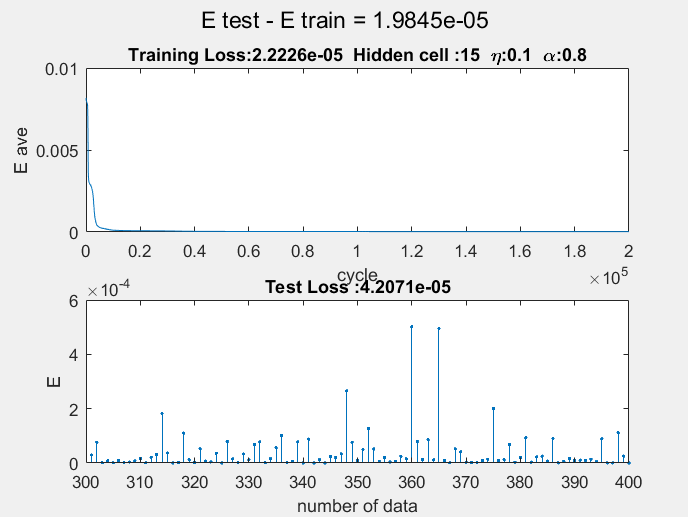
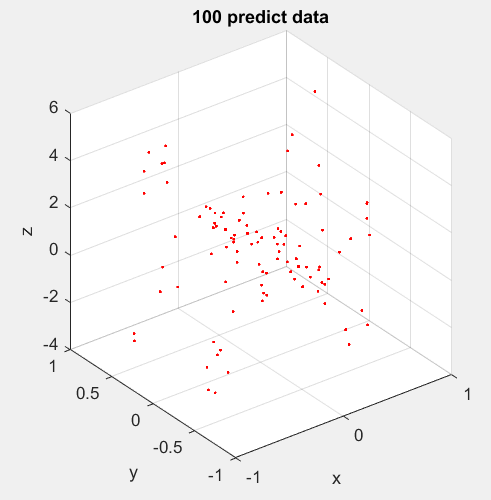
圖三

#### ( hidden cell , learning rate , momentum )=( 10 ,0.1 , 0.8 )

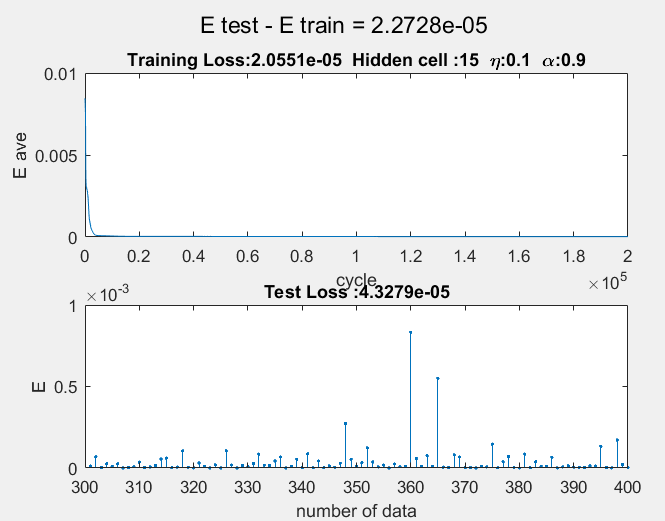
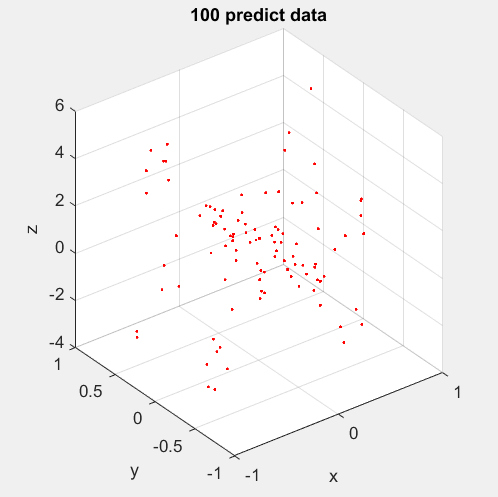
圖四

#### ( hidden cell , learning rate , momentum )=( 15 ,0.1 , 0.8 )

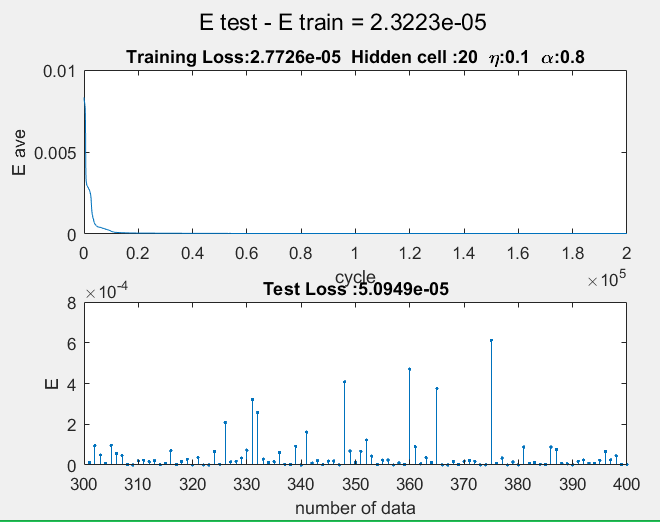
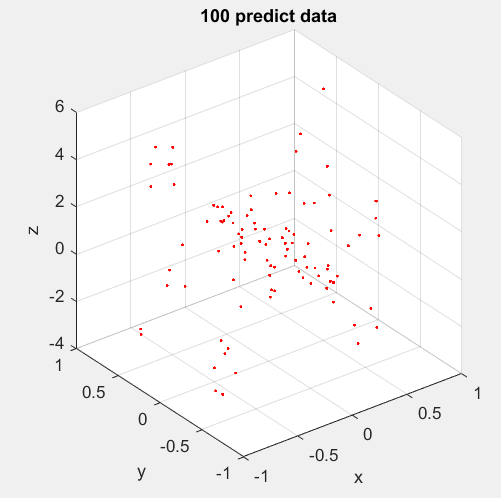
圖五

#### ( hidden cell , learning rate , momentum )= ( 15 ,0.1 , 0.9 )

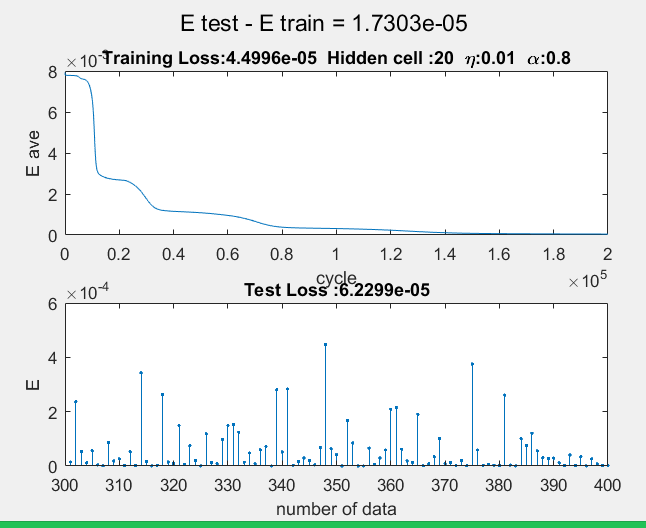
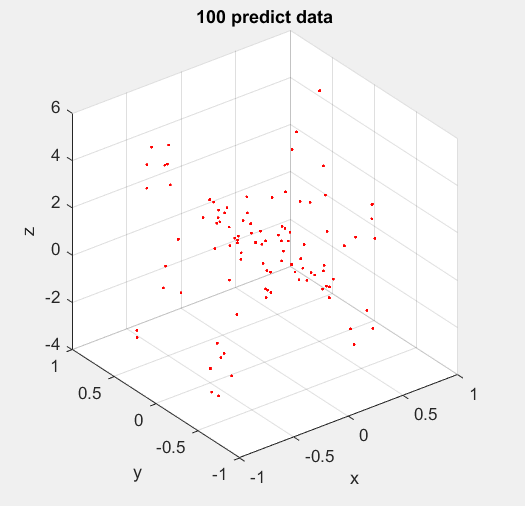
圖六

#### ( hidden cell , learning rate , momentum )= ( 20 ,0.1 , 0.8 )

圖七

#### ( hidden cell , learning rate , momentum )= ( 20 ,0.01 , 0.8 )

圖八

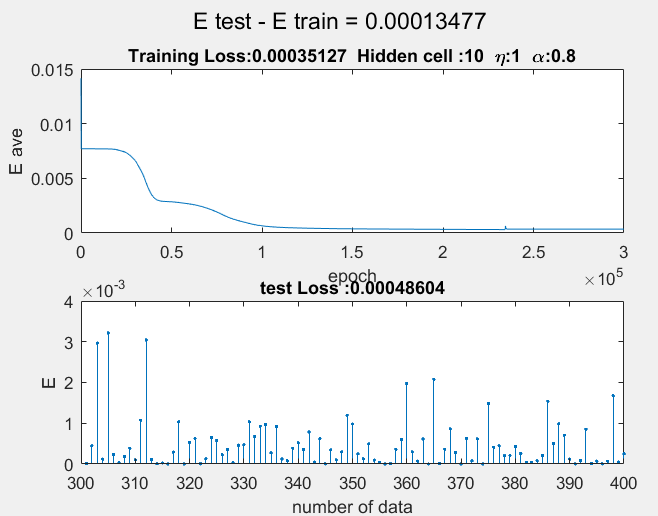
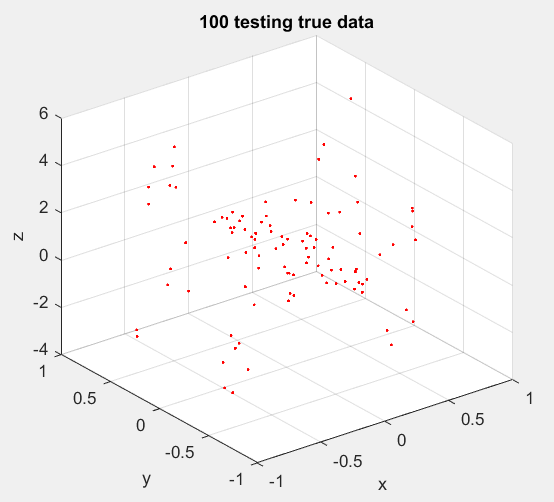
### batch learning:

#### ( hidden cell , learning rate , momentum )= ( 10 ,0.5 , 0.8 )

#### 

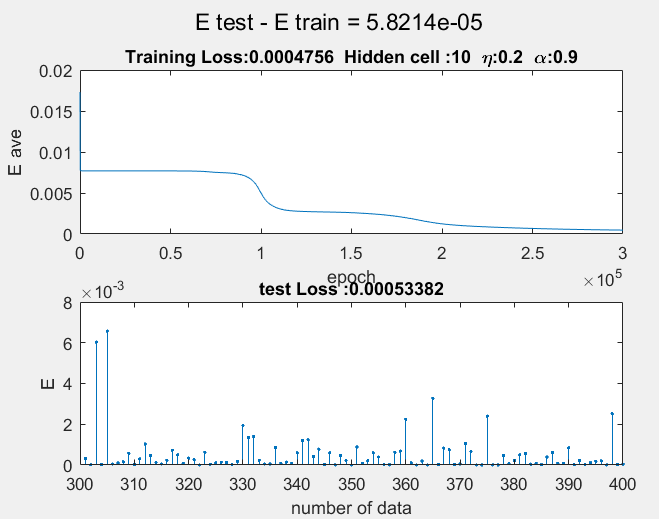
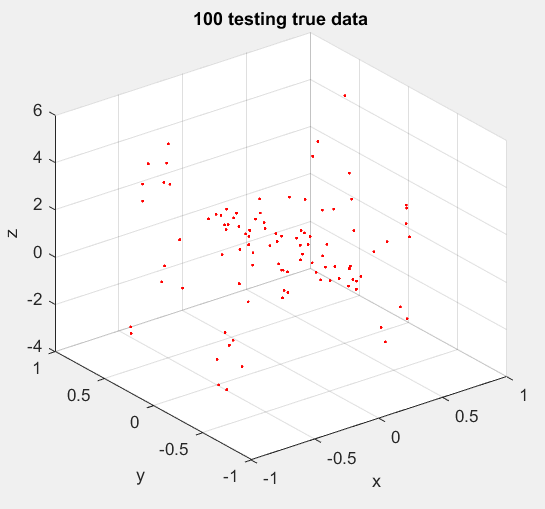
圖九

#### ( hidden cell , learning rate , momentum )= ( 10 ,1 , 0.8 )

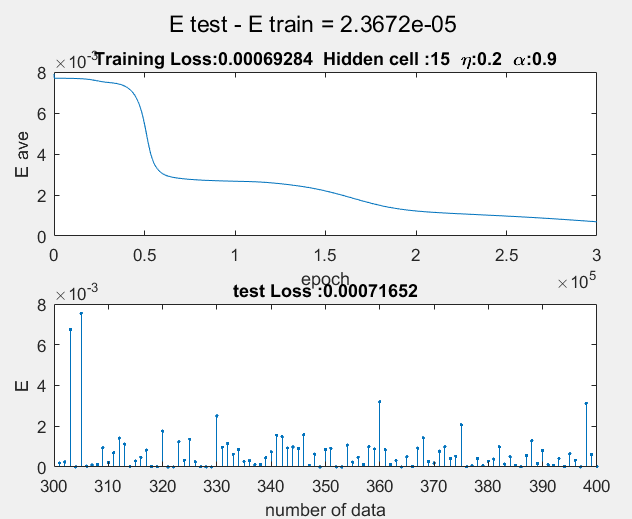
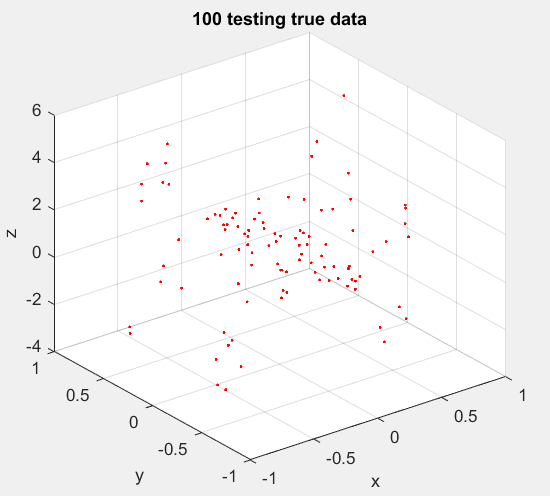
圖十

#### ( hidden cell , learning rate , momentum )= ( 10 ,0.2, 0.9 )

圖十一

#### ( hidden cell , learning rate , momentum )= ( 15 ,0.2, 0.9 )

圖十二

## table

### sequential learning:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Hidden cell | η | α | Training Loss | Test Loss | E\_train-E\_test |
| 20 | 0.01 | 0 | 0.0010873 | 0.0012716 | 0.00018432 |
| 5 | 0.1 | 0.8 | 0.0009660 | 0.000970 | 0.0000040 |
| 10 | 0.1 | 0.8 | 0.000020 | 0.000039 | 0.0000184 |
| 15 | 0.1 | 0.8 | 0.000022 | 0.000042 | 0.0000198 |
| 15 | 0.1 | 0.9 | 0.000020 | 0.000043 | 0.0000227 |
| 20 | 0.1 | 0.8 | 0.000027 | 0.000050 | 0.0000232 |
| 20 | 0.01 | 0.8 | 0.000044 | 0.000062 | 0.0000173 |

表一

### batch learning:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Hidden cell | η | α | Training Loss | Testing Loss | E\_train-E\_test |
| 10 | 0.5 | 0.8 | 0.000368 | 0.000460 | 0.0009130 |
| 10 | 1 | 0.8 | 0.000351 | 0.000486 | 0.0001340 |
| 10 | 0.2 | 0.9 | 0.000475 | 0.000533 | 0.0000582 |
| 15 | 0.2 | 0.9 | 0.000692 | 0.000716 | 0.0000236 |

表二

# conclusion-analysis

以Learning Mode 來看，Batch learning 因為將一批的權重更新量做平均，所以可使學習效果穩定、平滑，適用於處理細緻複雜的問題，缺點是收斂速度緩慢，耗費運算空間；Sequential learning 優點是可以做real-time學習，但因為每一筆資料皆會影響權重，如果資料選得不好，造成陷入局部最佳值，為了解決此情況發生，我們可以加入Momentum將上一次的更新量保留一個比例，加入此次的權重更新量，由圖二Training Loss 0.00108，加入Momentum 0.8後，得到Training Loss 0.00002 ，明顯提升學習效果，也可以發現加入慣量後，收斂速度提高了。

從learning rate的角度來看，理論上learing rate越小，學習之Loss也會變小，而圖七之Training Loss 0.000044 ，將learning rate提高後，得到圖八 Training Loss 0.00002，Loss反而降低了0.00002，此情況也可由圖九圖十觀察到，推測原因可能是此次的訓練目標函數，對於learning rate精度要求不高，故0.1或0.01對函數來說影響不大；從Hidden Layer 神經元個數來看，由表一，固定learning rate 0.1、Momentum 0.8，改變神經元數5,10,15，觀察到10顆的traning loss 最好 ，故神經元太多並不會提高學習表現，我們要選擇最合適的神經元個數。

由表一，此次學習效果最好的是10顆神經元、learning rate 0.1 、M0mentum 0.8，testing loss為0.000039，如何調整參數至最佳結果是我們需要進一步學習的課題。