

臺灣二〇〇四年國際科學展覽會

科 別：電腦科學科

作品名稱：BP 人工神經網路應用於求解直線方程式

學 校：臺中縣私立弘文高級中學
私立衛道高中(附設國中)

作 者：徐偉哲、陳泓瑋

作者簡介



我的名字是徐偉哲，目前就讀台中縣私立弘文高中一年級。在科學時代的生活中，幾乎每個東西都跟電腦科學、物理科學息息相關，尤其是數學可以說是科學的基礎，所以我從前就對這些學科有濃厚的興趣。接觸 BP 人工神經網路以後，覺得它是一門有人工智慧的學問，確實值得我們研究，又因指導老師在這方面有相當的研究，在加上師長們的鼓勵我們便利用課餘的時間來研究，並參加這一次的科展。



我的名字是陳泓瑋，目前就讀台中市私立衛道高級中學國中部三年級，從小就對小說中機器人之類的東西頗感興趣，而感興趣的原因不外乎就是它擁有類似人類的思考方式。之後指導老師時常跟我們提起類神經網路方面的知識，這對我就像一記當頭棒喝！小說中的機器人所擁有的思考能力，不就跟人工神經網路的功用類似嗎？所以後來在一次難得的機會下，以這個題目參加了這次的科展。

英文摘要(Abstract)

The Linear Equation Using The BP Artificial Neural Networks

Now Artificial Neural Networks using on the basic math is fewer. This paper is to suggest the Linear equation of the basic math using the BP Artificial Neural Networks. The BP Neural Networks have power ability for learning and can approximate any function, and regularity can be found to solve the linear equation. A good sample is one of the important elements for learning of Artificial Neural Networks. Generally, the samples are a lot of amount for the resolution of Linear equation. This paper is to use the principle of two points decide one line for the samples. The experiment shows that this method curtails many samples. Furthermore we also use Artificial Neural Networks to solve the problem of point-slope form. The experiment result is very satisfactory, and it offers some idea for the basic math using Artificial Neural Networks.

中文摘要

目前人工神經網路較少用於基礎數學方面的求解，本文針對基礎數學直線方程式提出 BP 人工神經網路應用於求解直線方程式，運用其很強的學習能力、(輸入向量和其對應的目標向量來訓練網路、逼近函數)，尋求規律來求解直線方程式；而良好的樣本是人工神經網路學習的重要條件之一，一般解決直線方程式需要大量樣本，本文利用二點決定一直線的原理來解決樣本問題，實驗結果顯示，這一方法成功的縮短了可觀的學習樣本，此外我們也運用 BP 人工神經網路來求解點斜式的直線方程式問題，實驗結果是可行的，並且為人工神經網路用於基礎數學提供了一些思考方向。

研究報告

壹、研究動機：由於我們指導老師從事神經網路在非線性及多項式數學的研究，即他所說的神經數學(經驗數學)，也經常和我們聊起關於這方面的話題，讓我們覺得神經網路像是一個神奇的學科，其實它是一門學習能力強又非常人性化的學科，所以我們想把它運用在國、高中的數學上。

研究目的：我們的研究目的在於運用類神經網路具有良好的學習能力及思考能力來求解國、高中的線性方程式問題。

貳、研究方法與過程

一、倒傳遞類神經網路(Back-Propagation Neural Network)簡介

倒傳遞類神經網路由 Rumelhart、Hinton 與 Williams 以及 David Parker[6]於 1986 年所提出。是目前應用最廣泛的一種人工神經網路。

1. 網路架構：

倒傳遞網路的架構如圖 1.2 所示[4]，包括輸入層、隱藏層及輸出層；而網路可以不只一層隱藏。其架構與前授型多層感知機的網路架構類似。

倒傳遞網路中的神經元，其最常用的非線性的轉換函數為雙彎曲函數 (Sigmoid function) $f(x) = \frac{1}{1 + e^{-x}}$ ，如圖 1.1 所示，這種函數當自變數，趨近於正負無窮大時，其函數值趨近於-1 或 1，而此函數值域在(-1, 1)之間[5]。

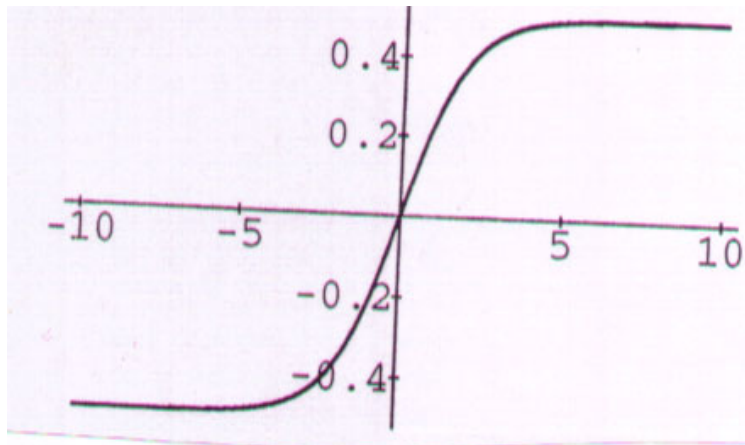


圖 1.1

2.倒傳遞網路演算法：

「倒傳遞演算法則」的網路訓練方式包含兩個部分：

(1)、順向訊號傳遞：當圖樣向量(Pattern Vector)送至輸入層，以前饋(feed forward)方式經由隱藏層，一層一層的向前傳遞至輸出層，並計算出網路輸出值，此時網路的加權值皆是固定的。

(2)、逆向誤差傳遞與加權值學習：網路的推論輸出值藉由加權值的修正，使得趨近於期望的輸出值。即以期望的輸出值與網路推論輸出值之誤差信號，一邊倒傳遞回網路一邊加以修正，因此我們將此種演算法稱之為「倒傳遞演算法」，簡稱為 BP 演算法，茲將結合圖 1.2 倒傳遞神經網路來討論 BP 演算法[1][4]。其中，實線部份表示順向信號傳遞，虛線部份表示逆向誤差傳遞。

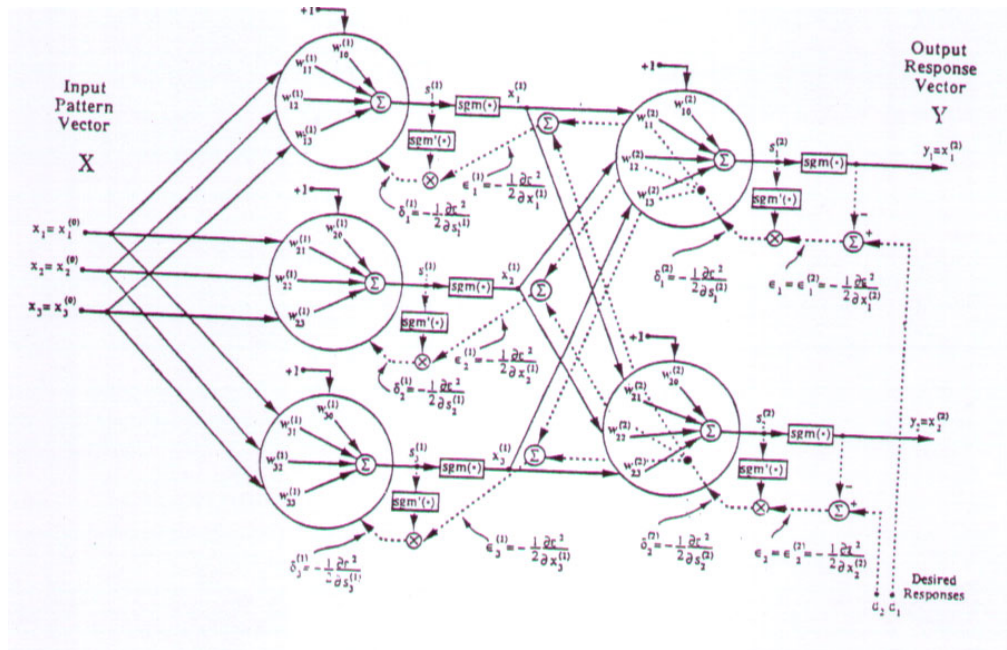


圖 1.2 倒傳遞類神經網路之信號與誤差傳遞方式

二、直線方程式簡介

1. 一個二元一次方程式的每一個解都是一個數對，在座標平面上，每一個解都可以描出一點。二元一次方程式有無限多組解，可以描繪出無限多個點，而這些點連起來就是一條直線。

例：二元一次方程式 $-2x + y = 3$

X	0	1	2	3	4	-1	-2	-3
Y	3	5	7	9	11	1	-1	-3

將(0,3)、(1,5)、(2,7)、(3,9)、(4,11)、(-1,1)、(-2,-1)、(-3,-3)描在座標平面上可得下圖。每一個點都是一個解，事實上， $-2x + y = 3$ 的解都是在一條直線上，故 $-2x + y = 3$ 所有的解所形成的圖形是一條直線，如圖 2.1。[3]

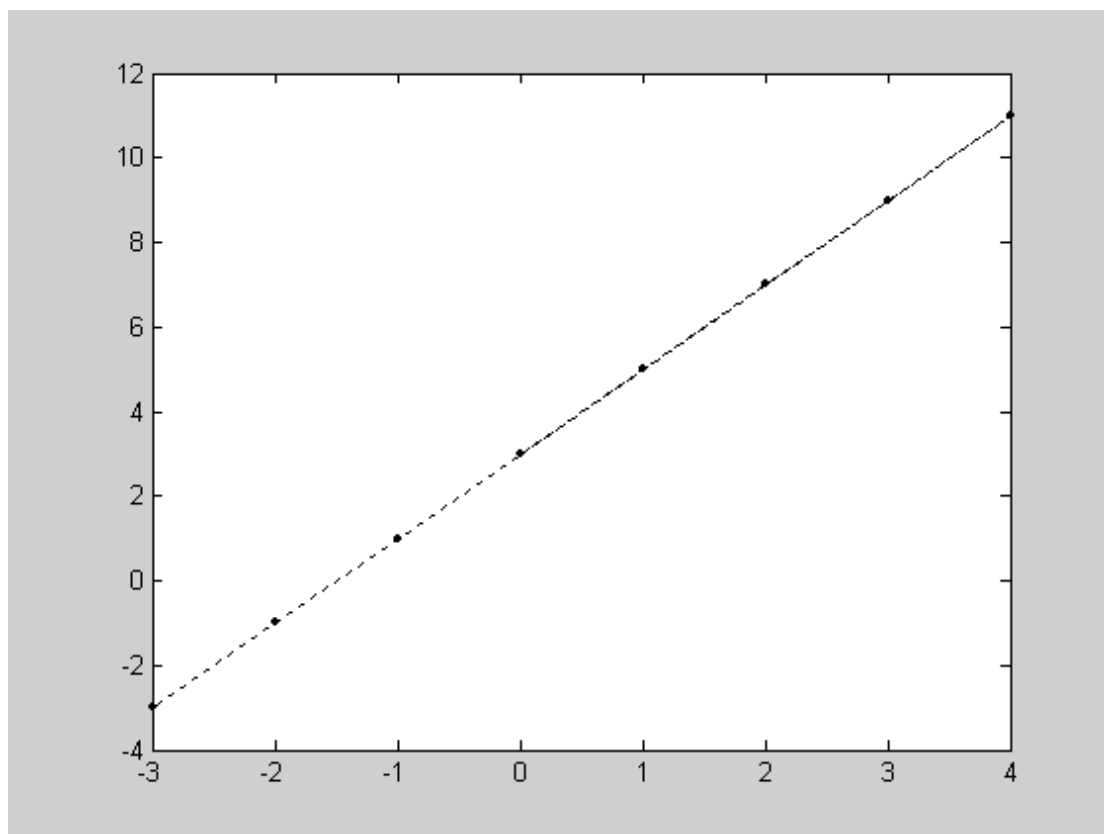


圖 2.1

2. 在日常生活中我們常見到斜坡，而且越陡的斜坡越不好行走，如何衡量陡的程度呢？一個辦法就是看鉛直落差與水平距離之比，比值越大表示坡越陡。斜坡的觀念，在數學上稱為斜率。更確切的說，設 L 為座標平面上的一直線，且不是鉛直線， $M(a, b)$ 與 $N(x, y)$ 為 L 上相異兩點，顯然 $a \neq x$ ，我們稱 $m = \frac{y-b}{x-a}$ 為直線 L 之斜率。[2]

三、樣本製作

1. 直線方程式樣本：直線方程式

$$y = ax + b$$

一般製作樣本輸入 a 、 x 、 b ，輸出 y ，都是由 0 到 9 的數字來製作，如表 3.1-1，但是這種樣本在整數方面就有 $10 \times 10 \times 10 = 1000$ 個樣本，十分的龐大，且會造成人工神經網路訓練的不便。

表 3.1-1

0.000	0.000	0.000	0.000
0.000	0.000	0.010	0.010
0.000	0.000	0.020	0.020
0.000	0.000	0.030	0.030
0.000	0.000	0.040	0.040
0.000	0.000	0.050	0.050
0.000	0.000	0.060	0.060
0.000	0.000	0.070	0.070
0.000	0.000	0.080	0.080
0.000	0.000	0.090	0.090
0.000	0.010	0.010	0.010
⋮	⋮	⋮	⋮

由二元一次方程式性質可知兩點可決定一條直線，根據這性質，在製作樣本時每個未知數僅取 2 個即可，如(表 3.1-2、圖 3.1)

樣本製作

$$y = ax + b$$

$$ax + b = y$$

當 $a = 1, b = 1$

$$x = 1, y = 2$$

$$x = 2, y = 3$$

當 $a = 1, b = 2$

$$x = 1, y = 3$$

$$x = 2, y = 4$$

人工神經網路樣本集
表 3.1-2

0.010	0.010	0.010	0.020
0.010	0.010	0.020	0.030
0.010	0.020	0.010	0.030
0.010	0.020	0.020	0.040

2. 點斜式樣本：

點斜式：過 $p(x_0, y_0)$ ，斜率為 m 的直線方程式如下

$$(y - y_0) = m(x - x_0) \quad (3.2-1)$$

當 $p(1,1)$ $m = 2$ 時

則 $y - 2x = -1$

當 $p(2,3)$ $m = 2$ 時

則 $y - 2x = -1$

當 $p(1,1)$ $m = 3$ 時

則 $y - 3x = -2$

當 $p(2,4)$ $m = 3$ 時

則 $y - 3x = -2$

(3.2-2)

當 $p(1,2)$ $m = 1$ 時

則 $y - x = 1$

當 $p(2,3)$ $m = 1$ 時

則 $y - x = 1$

當 $p(1,3)$ $m = 1$ 時

則 $y - x = 2$

當 $p(2,4)$ $m = 1$ 時

則 $y - x = 2$

人工神經網路樣本集製作如表 3.2

表 3.2-1

+0.020 +0.010 +0.010 +0.010 -0.020 -0.010
+0.020 +0.020 +0.030 +0.010 -0.020 -0.010
+0.030 +0.010 +0.010 +0.010 -0.030 -0.020
+0.030 +0.020 +0.040 +0.010 -0.030 -0.020

表 3.2-2

+0.010 +0.010 +0.020 +0.010 -0.010 +0.010
+0.010 +0.020 +0.030 +0.010 -0.010 +0.010
+0.010 +0.010 +0.030 +0.010 -0.010 +0.020
+0.010 +0.020 +0.040 +0.010 -0.010 +0.020

四、建立 BP 網路模型(如圖 1.2)

1. 訓練樣本

二元一次方程式

訓練次數:30000

訓練樣本:4

輸入層: 3

隱藏層:3

輸出層:1

學習率 η :0.3

動量係數 α :0.8

點斜式

訓練次數:30000

訓練樣本:4

輸入層: 3

隱藏層:3

輸出層:3

學習率 η :0.9

動量係數 α :0.8

2. 樣本集測試

二元一次方程式

表 3.1

$T[j] = 0.0200$	$Y[j] = 0.0200$	$mse = 0.0000$
$T[j] = 0.0300$	$Y[j] = 0.0300$	$mse = 0.0000$
$T[j] = 0.0300$	$Y[j] = 0.0300$	$mse = 0.0000$
$T[j] = 0.0400$	$Y[j] = 0.0400$	$mse = 0.0000$

註：T[j]為目標值、Y[j]為測試值、mse 為誤差

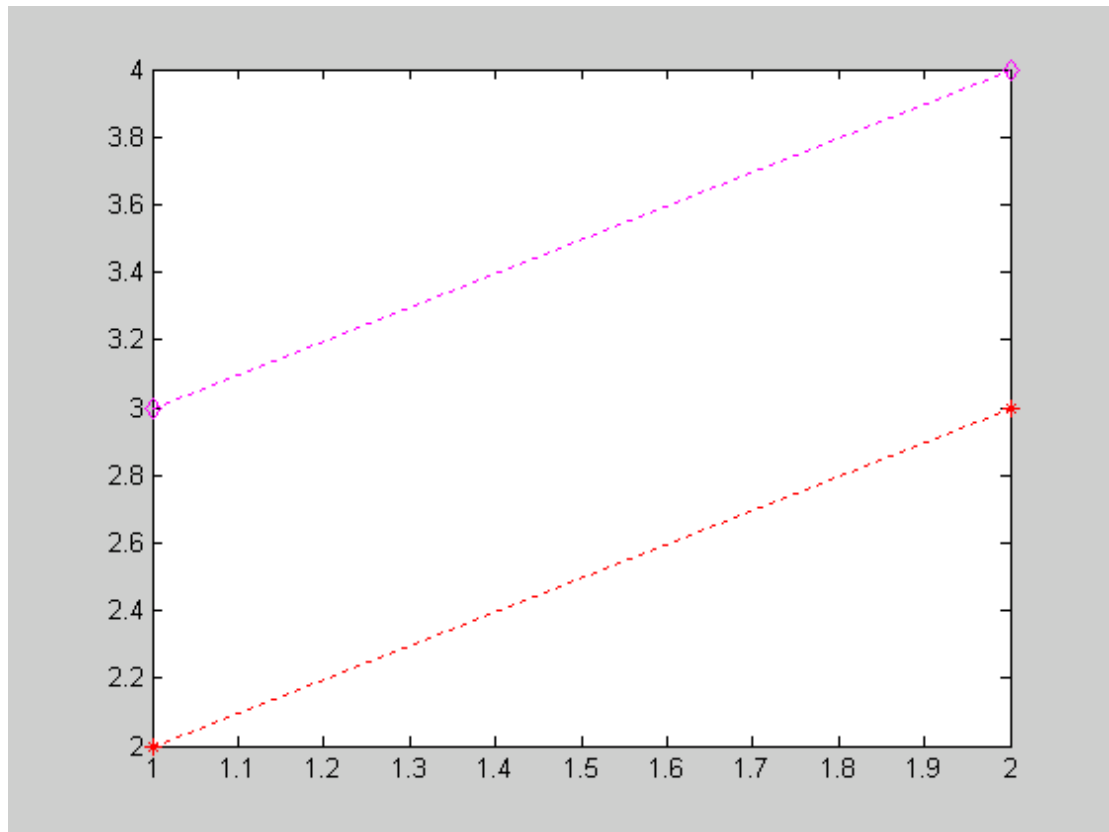


圖 3.1

點斜式

表 3.2-1

T[j]=0.010	-0.0200	-0.010	Y[j]=0.0101	-0.0201	-0.010	mse= 0.0000
T[j]=0.010	-0.0200	-0.010	Y[j]=0.0100	-0.0200	-0.0100	mse= 0.0000
T[j]=0.010	-0.0300	-0.0200	Y[j]=0.0101	-0.0300	-0.0200	mse= 0.0000
T[j]=0.010	-0.0300	-0.0200	Y[j]=0.0101	-0.0300	-0.0200	mse= 0.0000

表 3.2-2

T[j] =0.010	-0.010	0.010	Y[j] =0.010	-0.0101	0.010	mse= 0.0000
T[j] =0.010	-0.010	0.010	Y[j] =0.0100	-0.010	0.010	mse= 0.0000
T[j] =0.010	-0.010	0.0200	Y[j] =0.010	- 0.0101	0.0200	mse= 0.0000
T[j] =0.010	-0.010	0.0200	Y[j] =0.0100	- 0.0100	0.0200	mse= 0.0000

3. 測試下列樣本集外的直線方程式

二元一次直線方程式

當 $a = 1, b = 1$

(1) $x = 3, y = ?$

(2) $x = 4, y = ?$

當 $a = 1, b = 2$

(3) $x = 3, y = ?$

(4) $x = 4, y = ?$

結果如下表 4.2

表 4.2

(1) $T[j] = 0.9990$ $Y[j] = 0.0400$

(2) $T[j] = 0.9990$ $Y[j] = 0.0500$

(3) $T[j] = 0.9990$ $Y[j] = 0.0500$

(4) $T[j] = 0.9990$ $Y[j] = 0.0600$

註：0.999:未知數。

0.010~0.990 代表 1~99。

測試值	實際值
(1) $Y[j] = 0.0400$	$y = 4$
(2) $Y[j] = 0.0500$	$y = 5$
(3) $Y[j] = 0.0500$	$y = 5$
(4) $Y[j] = 0.0600$	$y = 6$

點斜式-1

(1)當 $p(3,5)$ $m = 2$

直線方程式=?

(2)當 $p(3,7)$ $m = 3$

直線方程式=?

測試方程式如表 4.2-1

點斜式-2

(1)當 $q(6,4)$, $m=1$

直線方程式=?

(2)當 $q(1,7)$, $m=1$

直線方程式=?

測試方程式如表 4.2-2

表 4.2-1

(1) $T[j] = 0.9990 \ 0.9990 \ 0.9990$ $Y[j] = 0.010 \quad -0.0199 \ -0.0100$
即方程式 $y - 2x = -1$

(2) $T[j] = 0.9990 \ 0.9990 \ 0.9990$ $Y[j] = 0.0101 \ -0.0301 \ -0.0199$
即方程式 $y - 3x = -2$

註：0.9990:未知數

0.010~0.990: 1 ~ 99

實際方程式

(1) $y - 2x = -1$

(2) $y - 3x = -2$

表 4.2-2

$$(1) T[j] = 0.9990 \ 0.9990 \ 0.9990 \ Y[j] = 0.0102 \ -0.0097 \ -0.200$$

$$\text{即方程式 } y - x = -2$$

$$(2) T[j] = 0.9990 \ 0.9990 \ 0.9990 \ Y[j] = 0.010 \ -0.0101 \ 0.0599$$

$$\text{即方程式 } y - x = 6$$

實際方程式

$$(1) \ y - x = -2$$

$$(2) \ y - x = 6$$

參、研究結果與討論

由上面的實驗發現，把神經網路運用在數學的線性問題上是行得通的，實際上它是百分之百的可實行(如圖 3.1)，因為在圖 3.1 上任一直線上任一點都可解，所以我們在樣本上用 2 點可解直線上無數的點。其次在解決斜率上，我們只給它一點及斜率便可求出方程式，在解決數學直線方程式上它是很好的工具。

其次，我們發現，如果我們把實驗的範圍加大時，誤差也會隨之變大，其原因可能是在於 Sigmoid 函數，因此，若我們將其改為線性轉移函數，當可以降低推論值的誤差，而這個部分，也是我們即將繼續的工作。

肆、結論與運用

神經網路運用在數學上目前是很少人在這方面研究，我們在這裡已經初步的解決直線方程式問題，且直線方程式是基礎數學的重要部分，希望以後能由直線方程式發展到基礎數學的其他領域。而我們也期望，能將這項實驗的結果，發展成一套實用的軟體。

伍、參考文獻

- [1]王進德、蕭大全，“類神經網路與模糊控制理論入門”，台北、台灣，全華圖書公司，pp.26-46，1994。
- [2]余文卿，高中數學課本<一>，台中縣，龍騰文化事業股份有限公司，pp.73-74，2003。
- [3]陳光明，“國中學習標竿〈數學 2〉”，台南市，南一書局企業股份有限公司，pp.72，2000。
- [4]馮天瑾，“神經網路技術”，青島市，青島海洋大學出版社，pp.72-77，1994。
- [5]James A. Freeman. Simulating Neural Networks with Mathematica. Addison-Wesley, Reading, MA. pp.16-19. 1994
- [6] Haykin S.. Neural Networks: A Comprehensive Foundation, Prentice Hall, Upper Saddle River, New Jersey. pp.43. 1999

<附件> 軟體工具

警告：

/*本附件程式(其中學習、回想程式由參考文獻[1]修改使用)，僅為實驗、參考用，

* 如因執行本程式造成系統毀損，概不負責

*/

學習程式

(訓練程式)

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <conio.h>

#define Ncycle

#define Ntrain

#define Ninp

#define Nhid

#define Nout

#define eta

#define alpha

#define train_file "c:1"

#define weight_file "c:1.wei"

#define mse_file "c:1.mse"

float random_value(void);

main()

{

FILE *fp1, *fp2, *fp3;

float X[Ninp], T[Nout], H[Nhid], Y[Nout];

float W_xh[Ninp][Nhid], W_hy[Nhid][Nout];

float dW_xh[Ninp][Nhid], dW_hy[Nhid][Nout];

float Q_h[Nhid], Q_y[Nout];

float dQ_h[Nhid], dQ_y[Nout];

float delta_h[Nhid], delta_y[Nout];

float sum, mse;

int Icycle, Itrain;

int i, j, h;

```
long int now;
```

```
clrscr();
```

```
/*-----open files-----*/
fp1=fopen(train_file,"r");
fp2=fopen(weight_file,"w");
fp3=fopen(mse_file,"w");
if (fp1==NULL)
{
puts("File not exist !!");
exit(1);
}
/*-----initialize weights-----*/
srand(time(&now) % 1);
for (h=0;h<Nhid;h++)
for (i=0;i<Nhid;i++)
{
W_xh[i][h]=random_value();
dW_xh[i][h]=0;
}
for (j=0;j<Nout;j++)
for (h=0;h<Nhid;h++)
{
W_hy[h][j]=random_value();
dW_hy[h][j]=0;
}
for (h=0;h<Nhid;h++)
{
Q_h[h]=0;
dQ_h[h]=0;
delta_h[h]=0;
}
for (j=0;j<Nout;j++)
{
Q_y[j]=random_value();
```

```

dQ_y[j]=0;
delta_y[j]=0;
}

/*-----start Learning-----*/
for (Icycle=0;Icycle<Ncycle;Icycle++)
{
mse=0.0;
/*... input one training example...*/
fseek(fp1,0,0);
for (Itrain=0;Itrain<Ntrain;Itrain++)
{
for (i=0;i<Ninp;i++)
fscanf(fp1,"%f",&X[i]);
for (j=0;j<Nout;j++)
fscanf(fp1,"%f",&T[j]);

/*....comput H,Y.....*/
for (h=0;h<Nhid;h++)
{
sum=0.0;
for (i=0;i<Ninp;i++)
sum=sum+X[i]*W_xh[i][h];

H[h]=(float)(exp(sum-Q_h[h])-exp(-(sum-Q_h[h])))/(exp(sum-Q_h[h])+exp(-(sum-Q_h[h])));
}
for (j=0;j<Nout;j++)
{
sum=0.0;
for (h=0;h<Nhid;h++)
sum=sum+H[h]*W_hy[h][j];

Y[j]=(float)(exp(sum-Q_y[j])-exp(-(sum-Q_y[j])))/(exp(sum-Q_y[j])+exp(-(sum-Q_y[j])));
}

```

```

/*...compute delta.....*/
for (j=0;j<Nout;j++)
delta_y[j]=(1.0+Y[j])*(1.0-Y[j])*(T[j]-Y[j]);

for (h=0;h<Nhid;h++)
{
sum=0.0;
for (j=0;j<Nout;j++)
sum=sum+W_hy[h][j]*delta_y[j];
delta_h[h]=(1.0+H[h])*(1.0-H[h])*sum;
}

/*....compute dW,dQ.....*/
for (j=0;j<Nout;j++)
for (h=0;h<Nhid;h++)
dW_hy[h][j]=eta*delta_y[j]*H[h]
+alpha*dW_hy[h][j];

for (j=0;j<Nout;j++)
dQ_y[j]=-eta*delta_y[j]+alpha*dQ_y[j];

for (h=0;h<Nhid;h++)
for(i=0;i<Ninp;i++)
dW_xh[i][h]=eta*delta_h[h]*X[i]
+alpha*dW_xh[i][h];

for(h=0;h<Nhid;h++)
dQ_h[h]=-eta*delta_h[h]+alpha*dQ_h[h];

/*....compute new W,Q.....*/
for (j=0;j<Nout;j++)
for (h=0;h<Nhid;h++)
W_hy[h][j]=W_hy[h][j]+dW_hy[h][j];

for (j=0;j<Nout;j++)
Q_y[j]=Q_y[j]+dQ_y[j];

```

```

for (h=0;h<Nhid;h++)
for (i=0;i<Ninp;i++)
W_xh[i][h]=W_xh[i][h]+dW_xh[i][h];
for (h=0;h<Nhid;h++)
Q_h[h]=Q_h[h]+dQ_h[h];

/*...compute the mean_square_error....*/
for (j=0;j<Nout;j++)
mse+=(T[j]-Y[j])*(T[j]-Y[j]);

} /* end of 1 learning cycle */

/*...write the mse_value to mse_file ...*/
mse=mse/Ntrain;
if ((Icycle % 10) ==9)
{
printf("nIcycle=%03d mse=%-8.4f\n",Icycle,mse);
fprintf(fp3,"%03d %-8.4f\n",Icycle,mse);
}

} /* end of total learning cycle */

/*....write the weights to weight_file.....*/
printf("\n");
for (h=0;h<Nhid;h++)
{
for (i=0;i<Ninp;i++)
{
printf("W_xh[%03d][%03d]=%-8.4f",i,h,W_xh[i][h]);
fprintf(fp2,"%-8.4f",W_xh[i][h]);
}
printf("\n");
fprintf(fp2,"\n");
}

printf("\n");
fprintf(fp2,"\n");

```



```

for (j=0;j<Nout;j++)
{
for (h=0;h<Nhid;h++)

{
printf("W_hy[%3d][%3d]=%-8.4f",h,j,W_hy[h][j]);
fprintf(fp2,"%-8.4f",W_hy[h][j]);
}
printf("\n");
fprintf(fp2,"\n");
}
printf ("\n");
fprintf (fp2,"\n");
for (h=0;h<Nhid;h++)
{
printf ("Q_h[%3d]=%-8.4f",h,Q_h[h]);
fprintf(fp2,"%-8.4f",Q_h[h]);
}
printf("\n\n");
fprintf(fp2,"\n\n");
for (j=0;j<Nout;j++)
{
printf("Q_y[%3d]=%-8.4f",j,Q_y[j]);
fprintf(fp2,"%-8.4f",Q_y[j]);
}
printf("\n\n");
fprintf(fp2,"\n\n");

fclose(fp1);
fclose(fp2);
fclose(fp3);
getch();

} /* end of the program */

```

```
float random_value(void)
```

```
{  
    return(((double)rand())/52767.0-0.1);  
}
```

回想程式

(測試程式)

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>

#define Ntest 14
#define Ninp  3
#define Nhid  3
#define Nout  3
#define weight_file "c:1.wei"
#define test_file   "c:1.64"
#define recall_file "c:1.rec"

main()
{
FILE *fp1,*fp2,*fp3;
float  X[Ninp],T[Nout],H[Nhid],Y[Nout];
float  W_xh[Ninp][Nhid],W_hy[Nhid][Nout];
float  Q_h[Nhid],Q_y[Nout];
float sum,mse;
int    Itest;
int    i,j,h;

clrscr();

/*-----open files-----*/
fp1=fopen(weight_file,"r");
fp2=fopen(test_file,"r");
fp3=fopen(recall_file,"w");
if (fp1==NULL)
{
puts("File not exit !!");
exit(1);
}
```

```

if (fp2==NULL)

{
puts("File not exist!!");
exit(1);
}

/*-----input weights from weight_file----*/
fseek(fp1,0,0);
for (h=0;h<Nhid;h++)
for (i=0;i<Ninp;i++)
fscanf(fp1,"%f",&W_xh[i][h]);

for (j=0;j<Nout;j++)
for (h=0;h<Nhid;h++)
fscanf(fp1,"%f",&W_hy[h][j]);

for (h=0;h<Nhid;h++)
fscanf(fp1,"%f",&Q_h[h]);

for (j=0;j<Nout;j++)
fscanf(fp1,"%f",&Q_y[j]);

/*-----start testing-----*/
fseek(fp2,0,0);
for (Itest=0;Itest<Ntest;Itest++)
{
/*.....input one testing example....*/
for (i=0;i<Ninp;i++)
fscanf(fp2,"%f",&X[i]);
for (j=0;j<Nout;j++)
fscanf(fp2,"%f",&T[j]);

/*.....comput H,Y.....*/
for (h=0;h<Nhid;h++)
{
sum=0.0;

```

```

for (i=0;i<Ninp;i++)
sum=sum+X[i]*W_xh[i][h];

H[h]=(float)(exp(sum-Q_h[h])-exp(-(sum-Q_h[h])))/(exp(sum-Q_h[h])+exp(-(sum-Q_h[h])));
}
for (j=0;j<Nout;j++)
{
sum=0.0;
for (h=0;h<Nhid;h++)
sum=sum+H[h]*W_hy[h][j];
Y[j]=(float)(exp(sum-Q_y[j])-exp(-(sum-Q_y[j])))/(exp(sum-Q_y[j])+exp(-(sum-Q_y[j])));
}
/*.....compute the mean_square_error...*/
mse=0.0;
for (j=0;j<Nout;j++)
mse+=(T[j]-Y[j])*(T[j]-Y[j]);

/*--write the results to recall_file---*/
printf("T[j]=");
fprintf(fp3,"T[j]=");
for(j=0;j<Nout;j++)
{
printf("%-8.4f",T[j]);
fprintf(fp3,"%-8.4f",T[j]);
}
printf("Y[j]=");
fprintf(fp3,"Y[j]=");
for (j=0;j<Nout;j++)
{
printf("%-8.4f",Y[j]);
fprintf(fp3,"%-8.4f",Y[j]);
}
printf(" mse= %-8.4f\n\n",mse);
fprintf(fp3," mse= %-8.4f",mse);
fprintf(fp3,"\n");
} /* end of recalling for total teat examples */

```

```
fclose(fp1);
```

```
fclose(fp2);
```

```
fclose(fp3);
```

```
getch();
```

```
} /* end of the program */
```

二元一次方程式

這是依照 2 點(取 3 點)決定一直線的原理所做的樣本

$$y = ax + b$$

$$ax + b = y$$

$$\text{當 } a = 1, b = 1$$

$$x = 1, y = 2$$

$$x = 2, y = 3$$

$$\text{當 } a = 1, b = 2$$

$$x = 1, y = 3$$

$$x = 2, y = 4$$

人工神經網路樣本集

$$0.010 \ 0.010 \ 0.010 \ 0.020$$

$$0.010 \ 0.010 \ 0.020 \ 0.030$$

$$0.010 \ 0.020 \ 0.010 \ 0.030$$

$$0.010 \ 0.020 \ 0.020 \ 0.040$$

點斜式樣本：

點斜式：過 $p(x_0, y_0)$ ，斜率為 m 的直線方程式如下

$$(y - y_0) = m(x - x_0) \quad (3-2)$$

$$\text{當 } p(1,1) \ m = 2$$

$$y - 2x = -1$$

$$\text{當 } p(2,3) \ m = 2$$

$$y - 2x = -1$$

$$\text{當 } p(1,1) \ m = 3$$

$$y - 3x = -2$$

$$\text{當 } p(2,4) \ m = 3$$

$$y - 3x = -2$$

(3.2-2)

當 $p(1,2)$ $m=1$ 時

則 $y-x=1$

當 $p(2,3)$ $m=1$ 時

則 $y-x=1$

當 $p(1,3)$ $m=1$ 時

則 $y-x=2$

當 $p(2,4)$ $m=1$ 時

則 $y-x=2$

人工神經網路樣本集

表 3.2-1

+0.020 +0.010 +0.010 +0.010 -0.020 -0.010

+0.020 +0.020 +0.030 +0.010 -0.020 -0.010

+0.030 +0.010 +0.010 +0.010 -0.030 -0.020

+0.030 +0.020 +0.040 +0.010 -0.030 -0.020

表 3.2-2

+0.010 +0.010 +0.020 +0.010 -0.010 +0.010

+0.010 +0.020 +0.030 +0.010 -0.010 +0.010

+0.010 +0.010 +0.030 +0.010 -0.010 +0.020

+0.010 +0.020 +0.040 +0.010 -0.010 +0.020

二元一次方程式

訓練次數:30000

訓練樣本:4

輸入層: 3

隱藏層:3

輸出層:1

學習率 η :0.3

動量係數 α :0.8

點斜式

訓練次數:30000

訓練樣本:4

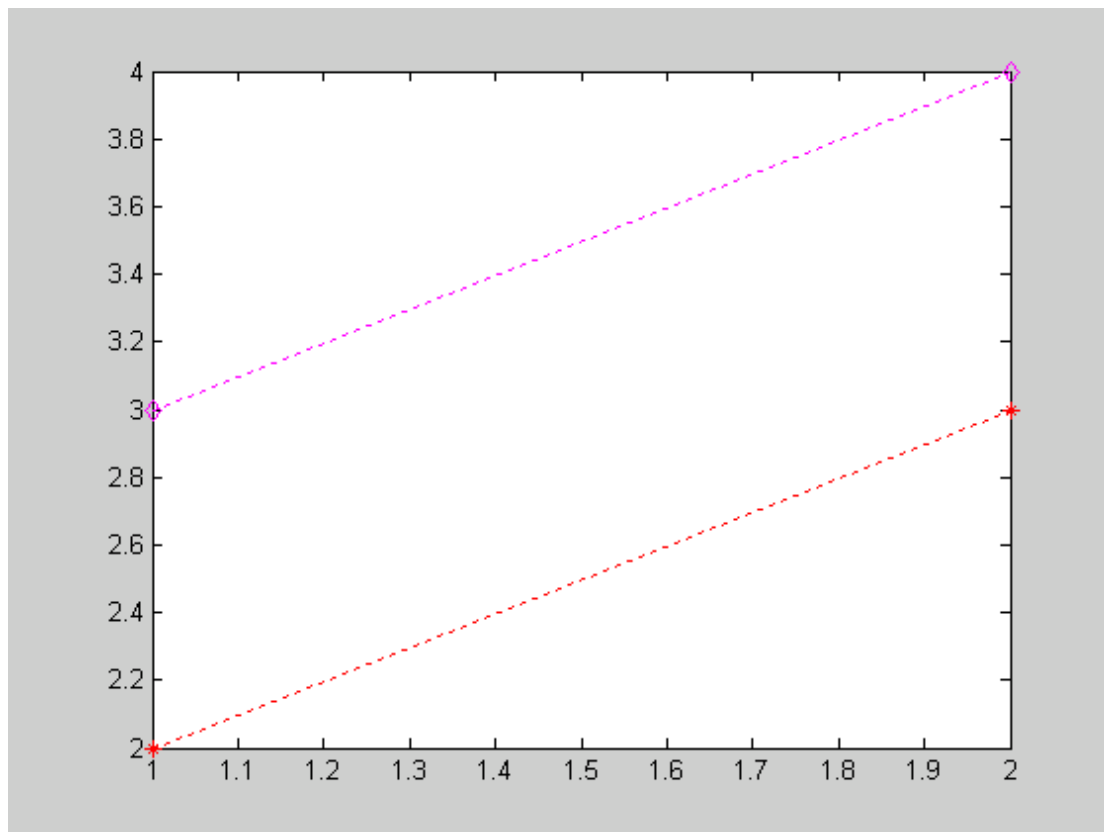
輸入層: 3

隱藏層:3

輸出層:3

學習率 η :0.9

動量係數 α :0.8



樣本集測試

二元一次方程式

T[j]=0.0200	Y[j]=0.0200	mse= 0.0000
T[j]=0.0300	Y[j]=0.0300	mse= 0.0000
T[j]=0.0300	Y[j]=0.0300	mse= 0.0000
T[j]=0.0400	Y[j]=0.0400	mse= 0.0000

點斜式

樣本集測試

表 3.2-1

T[j]=0.010	-0.0200	-0.010	Y[j]=0.0101	-0.0201	-0.010	mse= 0.0000
T[j]=0.010	-0.0200	-0.010	Y[j]=0.0100	-0.0200	-0.0100	mse= 0.0000
T[j]=0.010	-0.0300	-0.0200	Y[j]=0.0101	-0.0300	-0.0200	mse= 0.0000
T[j]=0.010	-0.0300	-0.0200	Y[j]=0.0101	-0.0300	-0.0200	mse= 0.0000

表 3.2-2

T[j] =0.010	-0.010	0.010	Y[j] =0.010	-0.0101	0.010	mse= 0.0000
T[j] =0.010	-0.010	0.010	Y[j] =0.0100	-0.010	0.010	mse= 0.0000
T[j] =0.010	-0.010	0.0200	Y[j] =0.010	- 0.0101	0.0200	mse= 0.0000
T[j] =0.010	-0.010	0.0200	Y[j] =0.0100	- 0.0100	0.0200	mse= 0.0000

樣本集外直線方程式測試

二元一次方程式

當 $a = 1, b = 1$

(1) $x = 3, y = ?$

(2) $x = 4, y = ?$

當 $a = 1, b = 2$

(3) $x = 3, y = ?$

(4) $x = 4, y = ?$

T[j]=0.9990 Y[j]=0.0400

T[j]=0.9990 Y[j]=0.0500

T[j]=0.9990 Y[j]=0.0500

T[j]=0.9990 Y[j]=0.0600

測試值

實際值

(1)Y[j]=0.0399 =4

(2)Y[j]=0.0499 =5

(3)Y[j]=0.0499 =5

(4)Y[j]=0.0599 =6

註：0.0100~0.9900:01~99

樣本集外直線方程式測試

點斜式-1

(1)當 $p(3,5)$ $m = 2$

直線方程式=?

(2)當 $p(3,7)$ $m = 3$

直線方程式=?

測試方程式如表 4.2-1

點斜式-2

(1)當 $q(6,4)$, $m=1$

直線方程式=?

(2)當 $q(1,7)$, $m=1$

直線方程式=?

測試方程式如表 4.2-2

測試方程式

表 4.2-1

(1) $T[j]=0.9990$ 0.9990 0.9990 $Y[j]=0.010$ -0.0199 $-0.0100 = y - 2x = -1$

(2) $T[j]=0.9990$ 0.9990 0.9990 $Y[j]=0.0101$ -0.0301 $-0.0199 = y - 3x = -2$

0.9990:未知數

0.010~0.990: 1 ~ 99

實際方程式

(1) $y - 2x = -1$

(2) $y - 3x = -2$

表 4.2-2

(1) $T[j] = 0.9990$ 0.9990 0.9990 $Y[j] = 0.0102$ -0.0097 -0.200

(2) $T[j] = 0.9990$ 0.9990 0.9990 $Y[j] = 0.010$ -0.0101 0.0599

實際方程式

(1) $y - x = -2$

(2) $y - x = 6$