เอกสารอ้างอิง

1. Baht N.V. and T.J. McAvoy, Use of neural nets for dynamic modeling and control of chemical process systems. *Computers chem. Engng* **14**, 573 - 582 (1990).
2. D.E. Seborg, T.F. Edgar and D.A. Mellichamp, *Process Dynamic and Control*, Wiley , New York (1989).
3. E.P. Nahas, M.A. Henson and D.E. Seborg, Nonlinear internal model control strategy for neural network models. *Computers chem. Engng* **16**, 1039 - 1057 (1992).
4. Jacek M. Zurada, *Introduction to Artificial Neural Systems*. Info Access Distribution Pte Ltd., (1992).
5. K.J. Hunt, D. Sbarbaro, R. Zbikowski and P.J. Gawthrop, Neural Networks for Control Systems - A Survey . *Automatica* **6**, 1083 - 1111 (1992).
6. Marzuki Khalid and Sigeru Omatu, A neural network controller for a temperature control system.. *IEEE Control System Mag.* **6** , 58 - 64 (1992).
7. Philip D. Wasserman , *Advanced Methods in Neural Computing*, Van Nostrand Reinhold, New York. (1993).
8. Phiroz Bhagat, An Introduction to Neural Nets. *Chem. Engng. Prog.*, 55 - 60 (1990).
9. S.Z. Qin, H.T. Zu and T.J. McAvoy, Comparison of Four Neural Net Learning Methods for Dynamic System Identification, *IEEE Trans. Neural Networks* **3**, 122 - 130 (1992).
10. S.Y. Kung, *Digital Neural Networks*, Prentice Hall International Inc., (1993).
11. Ydstie, B.E., Forecasting and control using adaptive connectionist networks, *Computer chem. Engng* **14**, 583 - 599 (1990).
12. Yoh-Ha Pao, *Adaptive Pattern Recognition and Neural Networks*. Addison - Wesley, USA (1989).
13. Wiliam L.Luyben, *Process Modeling, Simulation, and Control for Chemical Engineers*, McGraw - Hill, Singapore (1990).

ภาคผนวก ก

คำอธิบายการเรียนรู้ของข่ายงานนิวรัล

โปรแกรมในตอนท้ายของภาคผนวกนี้ เป็น ส่วนหลักของข่ายงานนิวรัลที่จะเรียนรู้กระบวนการต่างๆ โดยแบ่งเป็น 2 ส่วน คือ

ก.1. ตัวแปรโกลบอล (Global variables) ซึ่งเป็นตัวแปรที่ใช้ตลอดทั้งโปรแกรม ดังนี้

ตัวแปร	คำอธิบาย
inputsize	ขนาดข้อมูลอินพุทในการฝึกข่ายงานนิวรัล
outputsize	ขนาดข้อมูลเป้าหมายในการฝึกข่ายงานนิวรัล
hidlayer	จำนวนชั้นฮิดเดน (1 หรือ 2)
act_num1	ฟังก์ชันกระตุ้นระหว่างชั้นอินพุทกับชั้นฮิดเดน
act_num2	ฟังก์ชันกระตุ้นระหว่างชั้นฮิดเดนกับชั้นเอาท์พุท
layer[ ]	จำนวนหน่วยนิวรอนในแต่ละชั้นของข่ายงานนิวรัล
rawdata	ชนิดแบบจำลองที่ใช้ฝึกข่ายงาน (1 หรือ 2 แบบจำลอง)
nlayer	จำนวนชั้นของข่ายงานนิวรัล (3 หรือ 4)
var[ ]	ค่าพารามิเตอร์อัตราการเรียนรู้, โมเมนตัม, ไบอัส
w1[ ][ ], w2[ ][ ], w3[ ][ ]	ค่าน้ำหนักที่เชื่อมโยงในโครงสร้างของข่ายงานนิวรัล
pattern	จำนวนรูปแบบของข้อมูลที่ใช้ในการเรียนรู้ของข่ายงานนิวรัล
logic1[ ][ ], logic2[ ][ ], logic3[ ][ ]	ตัวแปรตรวจสอบการเชื่อมโยงระหว่างหน่วยนิวรอน

ก.2. ฟังก์ชันในการเรียนรู้ของข่ายงานนิวรัล (backprop) แสดงรายละเอียดในการเรียนรู้ ดังนี้

ก.2.1 การกำหนดแบบจำลองในการเรียนรู้ กำหนดโดยตัวแปร rawdata

ถ้า rawdata = 1 : เลือกแบบจำลองเชิงขนานแบบไม่วกกลับ

rawdata = 2 : เลือกแบบจำลองแบบอนุกรม-ขนานแบบทั่วไป

การเลือกแบบจำลองที่แตกต่างกัน จะกำหนดข้อมูลในการส่งให้ข่ายงานเริ่มต้นการ

เรียนรู้ที่แตกต่างกัน แต่ข้อมูลอินพุทและเอาท์พุทของทั้งสองวิธีจะเก็บไว้ในตัวแปรเดียวกัน คือ netin[ ][ ] และ netout[ ][ ] ตามลำดับ ทั้งนี้ ต้องแปลงข้อมูลให้เท่ากับค่าของฟังก์ชันกระตุ้นที่ใช้ในการแปลง (ถ้าเลือก unipolar ช่วงอยู่ที่ 0.1 ~ 0.9 ถ้าเลือก bipolar ค่าอยู่ในช่วง -1 ~ 1)

ก.2.2 การเริ่มต้นการเรียนรู้

หลังจากที่จัดการข้อมูลอินพุท, เป้าหมาย เริ่มต้นการเรียนรู้โดยการกำหนดค่า RMS = 0 (เป็นตัวแปร error) กำหนดจำนวนรูปแบบการเรียนรู้เป็นจำนวน P รอบ แล้วถ่ายข้อมูลอินพุทให้ตัวแปร Z ข้อมูลเป้าหมายให้ตัวแปร d คำนวณผลรวมค่าอินพุทเก็บไว้ที่ hidden แล้วแปลงด้วยฟังก์ชันกระตุ้น เป็นค่าของหน่วยฮิดเดนเก็บไว้ที่ out[ ] จำนวนชั้นฮิดเดนกำหนดด้วยตัวแปร hidlayer ถ้าจำนวนชั้นฮิดเดนเป็นสองชั้นแล้ว hidlayer = 2 ทำการคำนวณในชั้นฮิดเดนอีกครั้งหนึ่งโดยใช้ข้อมูลจากชั้นฮิดเดนที่ 1 มาเป็นข้อมูลอินพุท ข้อมูลในชั้นฮิดเดนจะส่งไปยังชั้นเอาท์พุทคำนวณค่า output[ ] เป็นผลลัพธ์ออกจากข่ายงานนิวรัล

ก.3 คำนวณค่า RMS error

เปรียบเทียบค่าเป้าหมายกับ d[ ] ค่าผลลัพธ์จากข่ายงานนิวรัล output[ ] คำนวณค่า RMS เก็บไว้ตัวแปร error คำนวณค่าสัญญาณความผิดพลาดในชั้นเอาท์พุทเก็บไว้ใน errout สำหรับค่าสัญญาณความผิดพลาดของชั้นฮิดเดนทั้งสองชั้น คือ errhid1 และ errhid2

ก.4 การปรับค่าน้ำหนัก

เริ่มปรับน้ำหนักในชั้นเอาท์พุทโดยปรับน้ำหนัก w3[ ][ ] ในชั้นฮิดเดนเข้ามีสองชั้น จะปรับน้ำหนัก w1[ ][ ] และ w2[ ][ ]

ก.5 การสิ้นสุดการเรียนรู้

การเรียนรู้ถูกกำหนดตามจำนวนรอบที่ต้องการให้เรียนรู้, iterate เมื่อครบกำหนดแล้วถ้าต้องการให้เรียนรู้ต่อไปแล้วจึงป้อนจำนวนรอบที่ต้องการให้เรียนรู้เข้าไป หลังจากเสร็จสิ้นการเรียนรู้ค่าน้ำหนักที่ได้ (ซึ่งได้แค่แบบจำลองภายในของข่ายงานนิวรัล) จะถูกบันทึกเก็บลงสู่แฟ้มข้อมูลซึ่งสามารถกำหนดชื่อแฟ้มข้อมูลโดยการป้อนชื่อไฟล์ที่ต้องการลงไป

```
 /* :::: Global variable :::: */
int  maxx, maxy;          /* The maximum resolution of the screen */
int  MaxColors;           /* The maximum # of colors available    */
int  inputsize, outputsize , hidlayer = 1;
int  act_num1 = 1, act_num2 = 1; /* 1 : hidden ,2: output activation func*/
int  layer[4] = {6,5,5,1};
int  pattern,rawdata=1,pastout ;
int  globax,globbx,globcx,globdx ;
int  exit_status=0;
int  nlayer = 4;
float learn,momentum,bias,cycle;
float var[6] = {1,0.85,1,0.65,1,500};
float w1[30][20],w2[20][20],w3[20][20],wr[20][10],model1[250],model2[250];
float logic1[30][20],logic2[20][20],logic3[10][20];
float dw1[30][20],dw2[20][20],dw3[20][20],dwr[20][10];
float diff,error;
```

```
void backprop(float x[250],float desire[250])
{
 char  msg;
 int   c,ans,nx,repeat;
 char  filename[10],str[5];
 char  check,cont;
 int   i,j,k,m,n,p,q,r,s,shift,grid_n;
 int   minx=0,x1,x2,xy1,xy2;
 int   target,input,hid,hid1,hid2,status;

 float iterate,loop,lx,ly,rx,ry,dx,dy,xd1,yd1;
 float px,py,excess,y;
 float errhid1[30],errhid2[30];
 float model1[150],model2[150],d[30],ym[250],yp[250];
 float neto,sumw,out1[30],out[30],z[30],px1,py1,py2,plx[500],ply[500];
 float hidden,power,maxin,base,lamda ;
 float errhid[30],errout[30];
 FILE  *fp;
 float slope1,slope,max,max1,min,min1,inter,inter1;
 float wlayer1[150],wlayer2[150];
 float output[30],netin[250][10],netout[250][10] ;
do{
        nx = 0;q = 0; loop = 0;
        input=layer[0]; hid1=layer[1]; hid2=layer[2]; target=layer[3];
        learn = var[3]; momentum = var[1];grid_n = 5;
        lamda = var[2]; bias = var[4];  iterate = var[5];
 i= 0;
 max   = x[i]; max1  = desire[i];
 min   = x[i]; min1  = desire[i];
```

```
ym[i] = x[i]; yp[i] = desire[i];
for(i = 1;i<inputsize;i++)
{
 for(j=0;j<layer[0];j++)
 {
 ym[i] = x[i];
 if(x[i]>max)
  max = x[i];
 if(x[i]<min)
  min = x[i],
 }
}
for(i=1;i<outputsize;i++)
{
 for(j=0;j<layer[3];j++)
 {
   yp[i] = desire[i];
 if(desire[i]>max1)
       max1 = desire[i];
 if(desire[i]<min1)
  min1= desire[i];
 }
}
/* :::::::: change scale into the range of 0.1 ~ 0.9 :::::::::::: */
slope  = (0.9-0.1)/(max-min);
slope1 = (0.9-0.1)/(max1-min1);
inter  = 0.9-slope*max;
inter1 = 0.9-slope1*max1;
```

```
k = 0;
if(rawdata ==1)    // ::::::::: Nonrecurrent parallel :::::::::
{
    k =0; m =0;
    for(p=0;p<pattern;p++)
    {
        for(i =0;i<layer[0];i++)
        {
         netin[p][i] = x[k]*slope + inter;
         k++;
        }
        for(j=0;j<layer[3];j++)
        {
         netout[p][j] = desire[m]* slope1 + inter1;
         m++;
        }
    }
}
// ::::::::::::::: General-series parallel ::::::::::::::::::::::::
else if(rawdata == 2)              /* for window data -> see baht */
{                                  //              -> see seborg
  pattern = inputsize-layer[0]/2 -1;
  for(p=0;p<=pattern;p++)
  {
   for(i=0;i<layer[0]/2;i++)
   {
        netin[p][i] = (x[k+i] * slope + inter) ;
        netin[p][i+layer[0]/2] = desire[k+i] * slope1 + inter1;
```

```
    }
    for(j=0;j<layer[3];j++)
    {
          netout[p][j]=(desire[layer[0]/2+k+j]*slope1 +inter1);
    }
    k++;
  }
}

repeat = 0;

do{
  error = 0.00;
  for(p =0;p<pattern;p++)
  {
          for(m=0;m<input;m++)
            z[m] = netin[p][m];      /* input neuron of each pattern */
          z[input] = bias;
          for(n=0;n<target;n++)
            d[n] = netout[p][n];
          for(j=0;j<hid1;j++)       {
                hidden = 0;
                for(i=0;i<=input;i++)
                  hidden = hidden + z[i]*w1[j][i];
                out1[j] = activate(hidden,lamda,act_num1);
                out[j]  = out1[j];
          }
          out1[hid1] = bias;   out[hid1] = out1[hid1];
```

```
        hid = hid1;                     // hidden node var. = hid
if(hidlayer == 2)                       // check for 2 hidden layer
{
        for(j=0;j<hid2;j++)
        {
            hidden = 0;
            for(i=0;i<=hid1;i++)
              hidden = hidden + out1[i]*w2[j][i];
            out[j] = activate(hidden,lamda,act_num1);
        }
        out[hid2] = bias;
        hid = hid2;                     // hidden node var. = hid
}
         for(k=0;k<target;k++)
         {
            neto = 0;
            for(j=0;j<=hid;j++)
            {
              neto = neto + out[j] * w3[k][j];
            }
            output[k] = activate(neto,lamda,act_num2);
        }
    lx = 30; rx = 615; ly = 80; ry = 415; shift = 30;
    lx = lx + shift+20;ly = ly+shift+15;rx= rx-20; ry = ry-shift-20;
    dx = rx-lx; dy = ry - ly;
    xd1 = dx/grid_n ;yd1 = dy/grid_n;
            /***********************/
            /* compute Error value */
```

```
                /************************/
    diff = 0;
    for(k = 0;k<target;k++)
        diff = diff+( (d[k]-output[k])*(d[k]-output[k]) );
    error = error + 0.5 * diff;
          if(error >=0 && error <= 1)    /* ::: plot pixel to display ::: */
          {
           py = (ry-yd1)-(error*(dy-yd1));
           px = lx+(dx*loop/500);
           setcolor(p+2);
           circle(px,py,0.5);
          }
       /* :::: for output layer :::: */
    for(k = 0;k< target;k++)
        errout[k]=eout(d[k],output[k],act_num2);

/* :::: for hidden layer :::: */
          for(j=0;j<hid2;j++)                    // for 2 hidden layer
          {
                sumw = 0;
                for(k = 0;k<target;k++)
                    sumw = sumw + errout[k] * w3[k][j];
                errhid2[j] = ehid(out[j],act_num1) * sumw;
          }
          for(j=0;j<hid1;j++)
          {
                sumw = 0;
                for(k = 0;k<hid2;k++)
```

```
            sumw = sumw + errhid2[k] * w2[k][j];
        errhid1[j] = ehid(out1[j],act_num1) * sumw;
}
        /*******************************/
        /* :::: weight adjustment :::: */
        /*******************************/
        /* for output layer */
for(k=0;k<target;k++)
{
  for(j=0;j<=hid2;j++)
  {
        dw3[k][j] = dw3[k][j]*momentum + (learn*out[j]*errout[k]);
        w3[k][j]  = w3[k][j] + dw3[k][j];
  }
}
        /* for hidden layer */
for(j=0;j<hid2;j++)
{
        for(i=0;i<=hid1;i++)
        {
           dw2[j][i] = dw2[j][i]*momentum + (learn*out1[i]*errhid2[j]);
           w2[j][i]  =  w2[j][i] + dw2[j][i];
        }
}
for(j=0;j<hid1;j++)
{
        for(i=0;i<=input;i++)
        {
```

```
                dw1[j][i] = dw1[j][i]*momentum + (learn*z[i]*errhid1[j]);
                w1[j][i]  =  w1[j][i] + dw1[j][i];
              }
          }
          q=q+1; loop=loop+1;
          cycle = cycle + 1;
          gotoxy(48,25); printf("%6.0f",cycle);
          if(loop >500)
          {
           loop = 1; nx = 0;
           bar(lx,ly,rx,ry);
           setcolor(DARKGRAY);
           for(i=1;i<grid_n;i++)              /* label grid line  */
                line(lx,(ly+yd1*i),rx,(ly+yd1*i) );
           for(i=1;i<grid_n;i++)
                line(lx+xd1*i,ly,lx+xd1*i,ry);
          }
/* ::: plot RMS line to screen ::: */
       ply[nx] = (ry-yd1)-(error*(dy-yd1) );
       plx[nx] = lx+(dx*loop/500);
       if(error >=0 && error <= 1)     /* ::: plot RMS line to screen ::: */
       {
          if(nx==0)                  /*  Initial point  */
           nx++;
          else
          {
           setcolor(BLUE);
           line(plx[nx-1],ply[nx-1],plx[nx],ply[nx]);
```

```
            nx = nx+1;
           }
        }
  }
  gotoxy(24,25);printf("%6.4f",error); status = status + 1;
} while( cycle < iterate);
   setcolor(BLUE);
   outtextxy(lx,460,"Do you want to continue <Y/N> :ฐาฐ ");
   cont = toupper(getch());
   setfillstyle(SOLID_FILL,CYAN);
   bar(minx+10,maxy-27,minx+400,maxy-8);
   shadow1(minx+10,maxy-27,minx+400,maxy-8,RED);
   if(cont =='Y')
   {
        repeat = 1;
        setfillstyle(SOLID_FILL,LIGHTBLUE);
        bar(lx+20,ly+90 ,rx-40,ry-90);
        mainbox(lx+30,ly+100,rx-50,ry-100,MAGENTA,LIGHTGRAY);
        outtextxy(lx+60,ly+120,"Enter the iteration number that you want ฑฐฐ");
        gotoxy(60,16);scanf("%f",&var[5]);
        setfillstyle(SOLID_FILL,LIGHTGRAY);
           bar(lx,ly,rx,ry);
           setcolor(DARKGRAY);
           for(i=1;i<grid_n;i++)              /* label grid line  */
                line(lx,(ly+yd1*i),rx,(ly+yd1*i) );
           for(i=1;i<grid_n;i++)
                line(lx+xd1*i,ly,lx+xd1*i,ry);
   }
```

```
}while(repeat ==  1);
cycle = 0;
    /*  model weight            */
    n = 0;
    for(k=0;k<target;k++)
     for(j=0;j<=hid1;j++)
          model2[n] = w2[k][j];
    m = 0;
    for(j=0;j<hid1;j++)
     for(i=0;i<=input;i++)
          model1[m] = w1[j][i];
    strcpy(filename,"modelly1.dat");
    fp = fopen(filename,"wt");
    write(model1,sizeof model1,1,fp);
    fclose(fp);
    strcpy(filename,"modelly2.dat");
    fp = fopen(filename,"wt");
    fwrite(model2,sizeof model2,1,fp);
    fclose(fp);
}
```

ภาคผนวก ข

การใช้โปรแกรม "Model Netware"

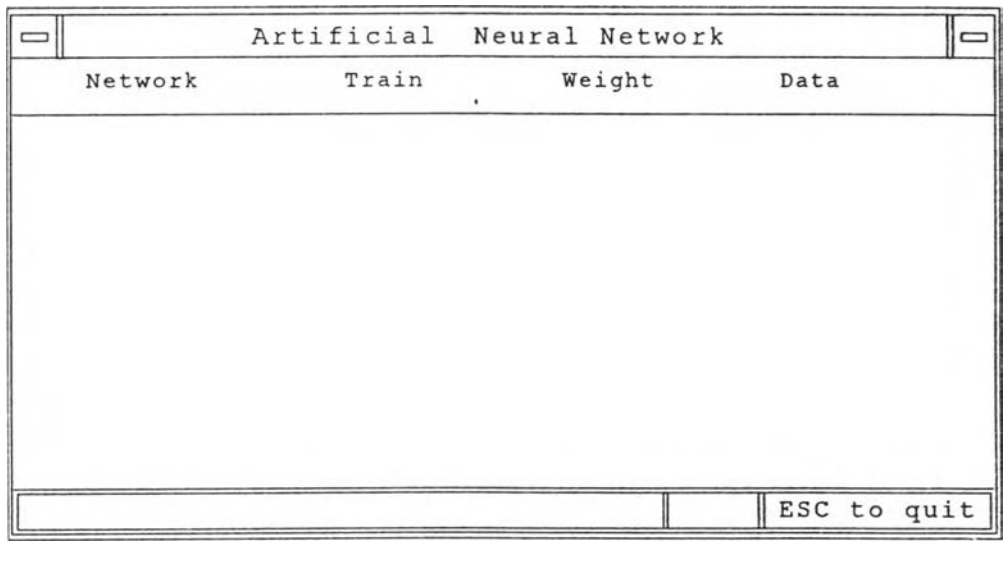
ในการศึกษาการสร้างแบบจำลองของกระบวนการ (Identification) โดยการใช้ข่ายงานนิวรัล มีรายละเอียดอธิบายประกอบดังนี้

ข.1. การเริ่มต้นเข้าสู่โปรแกรม

ที่สภาวะดอส ผู้ใช้เรียกใช้โปรแกรม "Model Netware" จากห้องขับดิสก์ที่มีโปรแกรมอยู่ โดยการป้อน "modelnet" ต่อจากเครื่องหมายพร็อมท์ (prompt) ดังนี้

c:> modelnet

หลังจากที่ผู้ใช้เรียก Model Netware ทำงาน เมนูหลักจะปรากฏบนจอภาพ แสดงดังรูปที่ 1 ประกอบด้วย เมนูหลัก ของข่ายงาน (Network), การฝึกข่ายงาน (Train), น้ำหนัก (Weight) และ การจัดการข้อมูลสำหรับการเรียนรู้ (Data) รายละเอียดของแต่ละเมนูหลัก จะได้อธิบายต่อไป



รูปที่ 1 เมนูหลักของ Model Netware

ข.2. การกำหนดโครงสร้างของข่ายงานนิวรัล

จากเมนูหลัก กำหนดโครงสร้างของข่ายงานนิวรัล โดยการป้อน "N" จะปรากฏเมนูย่อยแสดงดังรูปที่ 2 อันประกอบด้วย โดยสามารถ กำหนดชั้นฮิดเดนได้ 2 ชั้น

Input neurons : จำนวนนิวรอนในชั้นอินพุท

Hidden1 neurons : จำนวนนิวรอนในชั้นฮิดเดนที่ 1

Hidden2 neurons : จำนวนนิวรอนในชั้นฮิดเดนที่ 2

Output neurons : จำนวนนิวรอนในชั้นเอาท์พุท

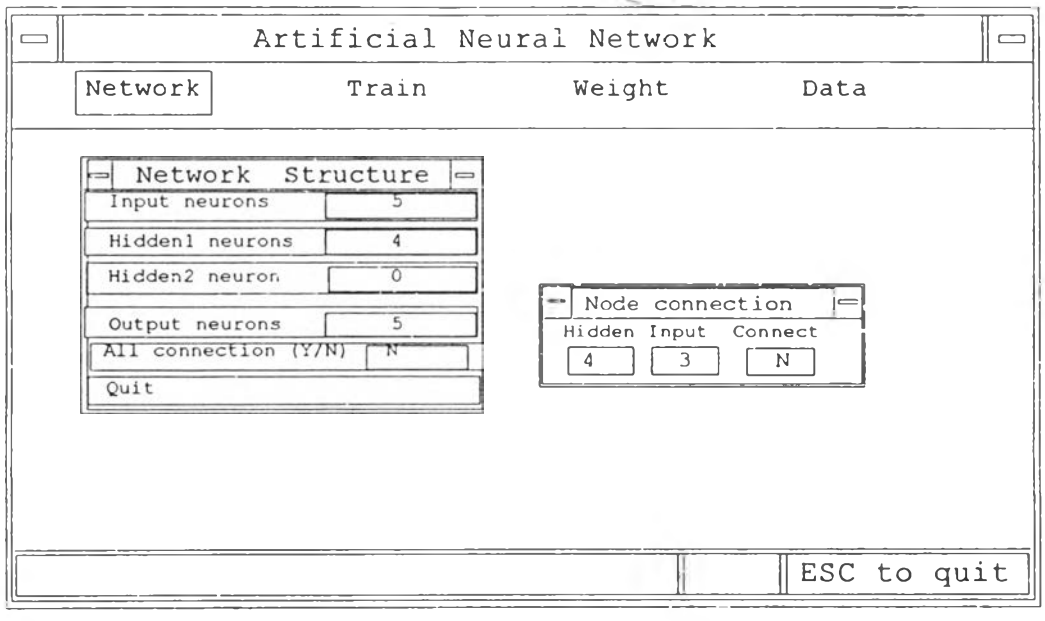
All connection (Y/N) : ความต้องการให้มีการเชื่อมโยงทั้งหมดของนิวรอนระหว่างชั้นหรือไม่

เมื่อเลือก 'Y' หมายถึง ตอ้งการให้มีการเชื่อมโยงทั้งหมดของนิวรอน

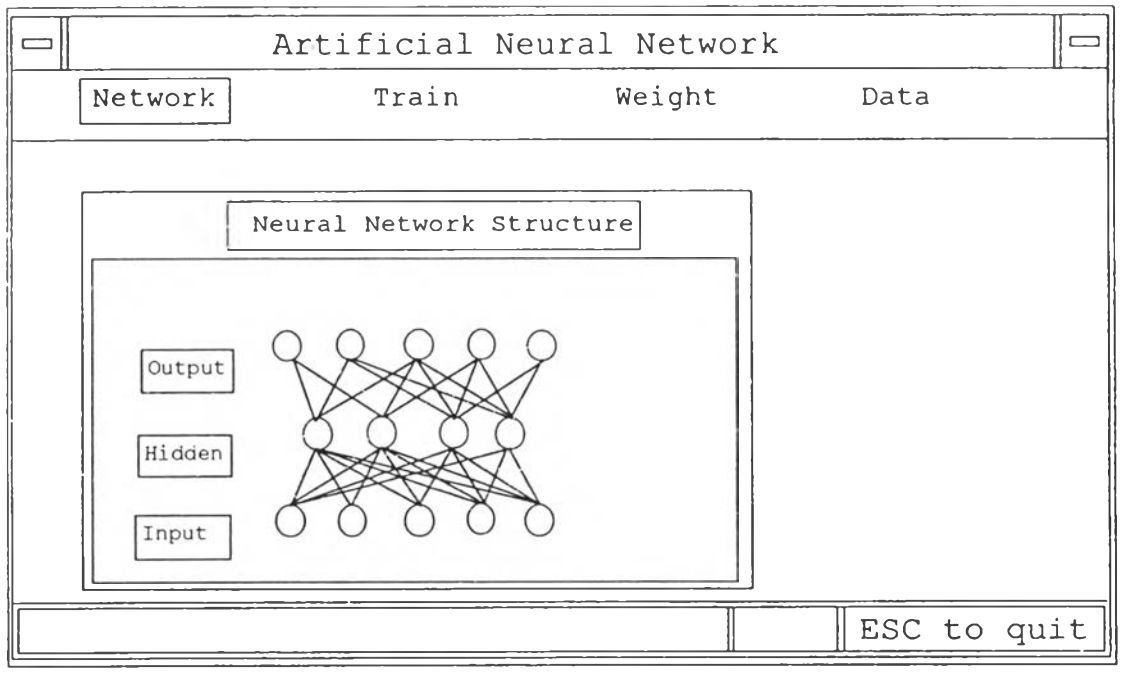
เมื่อเลือก 'N' หมายถึง ตอ้งการให้มีการเชื่อมโยงของนิวรอนบางส่วน

Quit : ออกจากการกำหนดโครงสร้างขายงานนิวรัล

ในการกำหนดให้มีการเชื่อมโยงระหว่างนิวรอนเป็นบางส่วน จะปรากฏเมนูย่อยให้เลือกการเชื่อมโยง ถ้าต้องการให้มีการเชื่อมโยงระหว่างนิวรอน ให้ป้อน 'Y' ถ้าไม่ต้องการให้ป้อน 'N' ผลลัพธ์ของโครงสร้างข่ายงานนิวรัลที่ได้ แสดงดังในรูปที่ 3



รูปที่ 2 เมนูย่อยในการกำหนดโครงสร้างของข่ายงานนิวรัล



รูปที่ 3 ตัวอย่างของโครงสร้างข่ายงานนิวรัลที่มีการเชื่อมโยงบางส่วน

ข.3. การเลือกชนิดข้อมูล

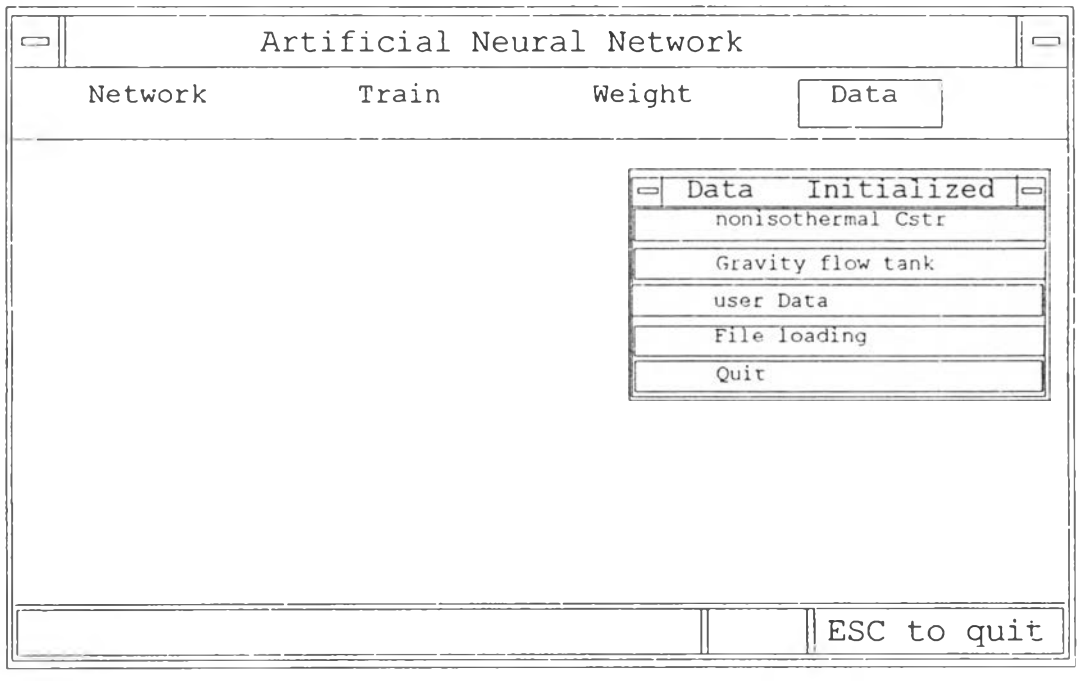
จากเมนูหลักเลือก "D" เพื่อกำหนดการทำ "Data Initialized" อันเป็นการกำหนดข้อมูลเริ่มต้นเพื่อใช้ในการนำไปให้ข่ายงานนิวรัลได้เรียนรู้ ประกอบด้วย

CSTR Unit เป็นข้อมูลพื้นฐานจากหน่วยปฏิบัติการเคมีของระบบถังกวนต่อเนื่องที่มีอุณหภูมิไม่ คงที่ (Non isothermal CSTR) โดยสามารถเปลี่ยนค่าพารามิ เตอร์ในกระบวนการ เพื่อใช้ ให้ข่ายงานนิวรัลได้เรียนรู้และทดสอบการเรียนรู้

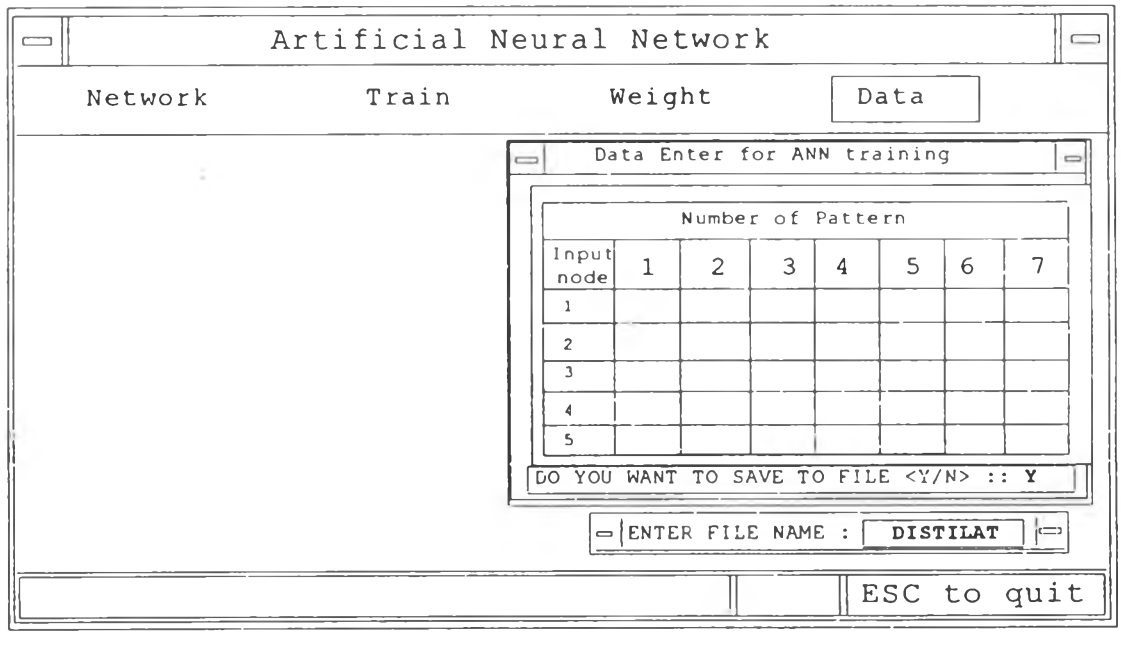
Gravity flow tank ข้อมูลในระบบการไหลของของเหลวออกจากถังโดยแรงโน้มถ่วง โดยสามารถปรับค่าอัตราการไหลของของเหลวได้

User Data เป็นการป้อนข้อมูลผ่านแป้นพิมพ์โดยตรงจากผู้ใช้ ประกอบด้วย ของข้อมูลอินพุทและข้อมูลเป้าหมาย (target) โดยจะสอดคล้อง กับโครงสร้างของข่ายงานนิวรัลที่ได้ กำหนดไว้ตั้งแต่แรก เมื่อป้อนข้อมูลเสร็จแล้ว สามารถเลือกที่จะบันทึกข้อมูลเก็บเข้าแฟ้ม ข้อมูลโดยการตั้งชื่อข้อมูลแล้ว สามารถนำกลับมาใช้งานได้โดยคำสั่ง File_load เพื่อ ใช้ทดสอบการเรียนรู้ใหม่ได้

File_Load เป็นการนำข้อมูลจากแฟ้มข้อมูลที่เคยสร้างไว้จากเมนูย่อย User Data หรือจากการแก้สมการอนุพันธ์ และ จาก Chemical Unit ที่มีการเปลี่ยนค่าพารามิเตอร์ ที่แตกต่างกัน



รูปที่ 4 เมนูย่อยการกำหนดข้อมูลให้ข่ายงานนิวรัล



รูปที่ 5 การเลือกป้อนข้อมูลโดยตรงให้กับข่ายงานนิวรัล จากเมนูย่อย User Data

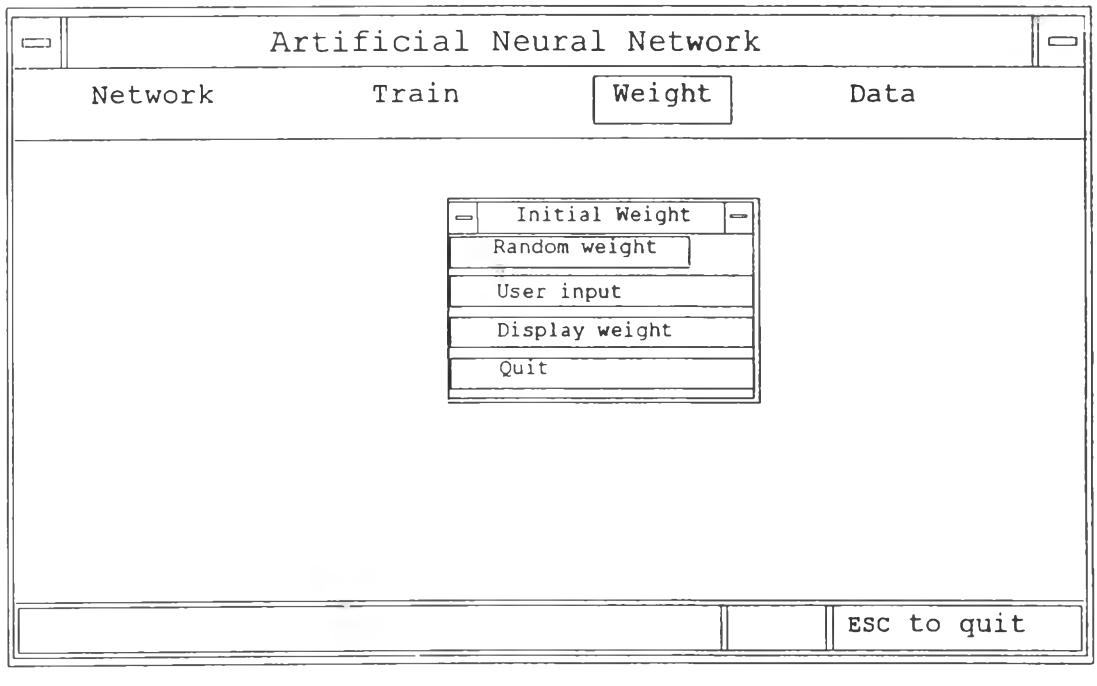
ข.4. การกำหนดค่าน้ำหนักเริ่มต้น ให้กับข่ายงานนิวรัลจากเมนูหลักของโปรแกรม เลือก Weigth เพื่อเริ่มต้นการกำหนดน้ำหนักให้กับข่ายงานนิวรัล จอภาพจะปรากฏเมนูย่อย ดังรูปที่ 6 ซึ่งมีความหมายดังนี้

Random weight : การกำหนดค่าน้ำหนักเริ่มต้นให้ข่ายงานแบบสุ่ม

User input : การกำหนดค่าน้ำหนักเริ่มต้นให้ข่ายงาน โดยผู้ใช้สามารถ ทำการป้อนค่าเข้าไปให้ โครงสร้างของข่ายงานได้โดยตรง

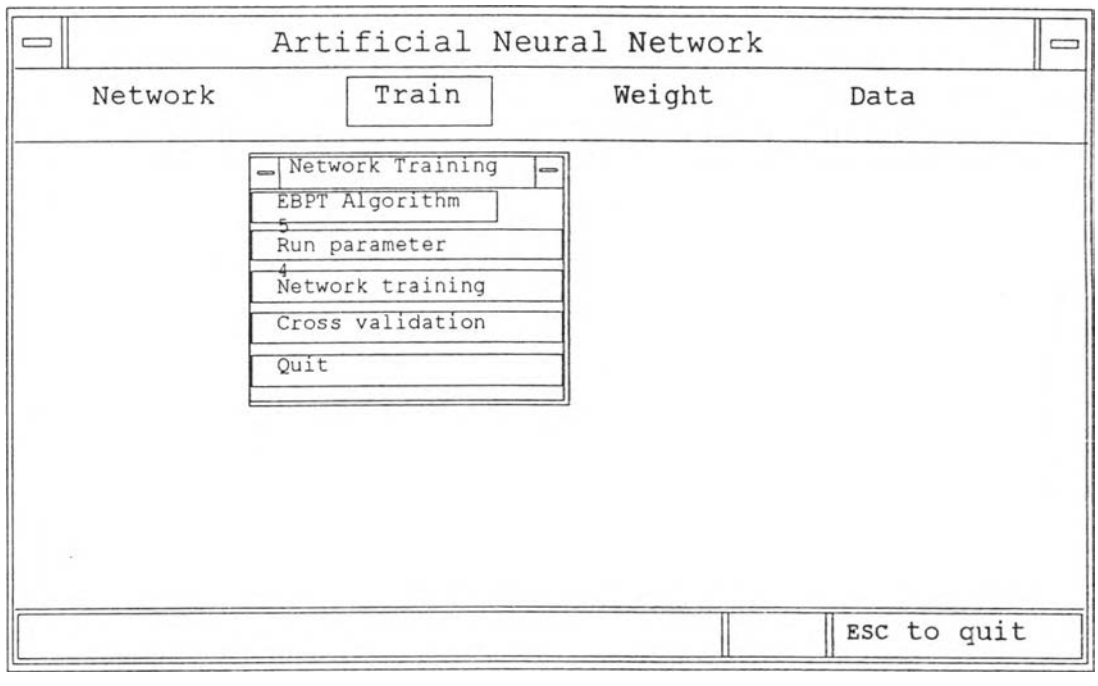
Display weight : แสดงค่าของน้ำหนักในขณะใดๆ ของข่ายงานนิวรัล ไม่ว่าจะเป็นก่อนการเรียนรู้ ขณะเรียนรู้ หรือ หลังการเรียนรู้

Quit : ออกจากการกำหนดค่าน้ำหนักเริ่มต้น



รูปที่ 6 เมนูการกำหนดค่าน้ำหนักเริ่มต้น

ข.5. การเรียนรู้ เมื่อได้กำหนดโครงสร้างของข่ายงานนิวรัล กำหนดชนิดของข้อมูลในการเรียนรู้ และกำหนดค่าน้ำหนักเริ่มต้นในการเรียนรู้แล้ว ขั้นตอนต่อไป คือ การเรียนรู้ของข่ายงานนิวรัลตามโครงสร้างและชนิดข้อมูลตามที่ได้กำหนดไว้ บนเมนูหลักเลือก **Train** จะปรากฏภาพดังใน รูปที่ 7

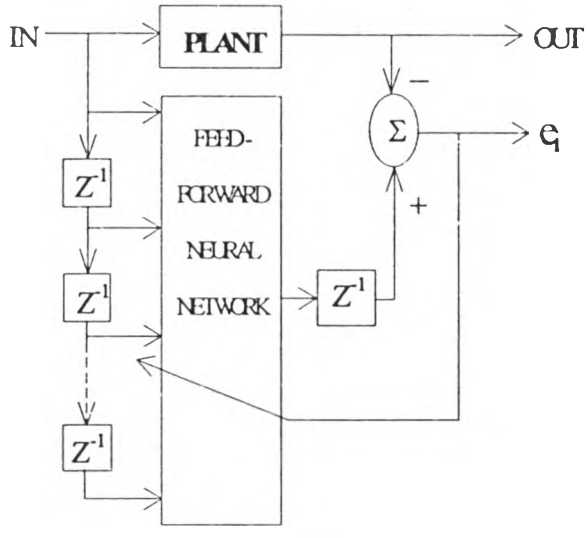


รูปที่ 7 เมนูย่อยในการกำหนดวิธีการเรียนรู้ของข่ายงานนิวรัล

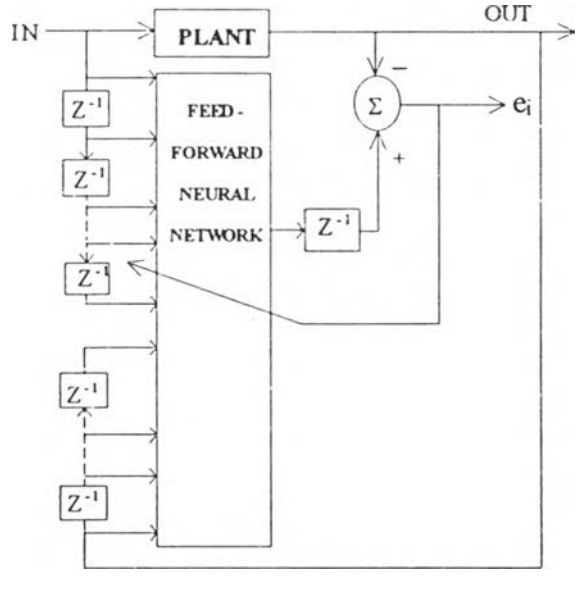
ข.5.1 แบบจำลองในการเรียนรู้ของข่ายนิวรัล เลือกได้จากเมนูย่อย **EBPT Algorithm** ซึ่งมีแบบจำลองให้เลือกในการเรียนรู้ 2 แบบ คือ

1. **Nonrecurrent parallel identification model**
2. **General series-parallel identification model**

แบบจำลองในแต่ละแบบนั้นแสดงดังในรูปที่ 8 โดยที่ผู้ใช้สามารถเลือกหมายเลขประจำโมเดล โมเดลดังกล่าวจะถูกนำมาใช้สำหรับการเรียนรู้ ซึ่ง **EBPT algorithm** ใช้อัลกอริธึมในการเรียนรู้ของข่ายงานนิวรัลชนิดเกรเดียนท์เดสเซ็นท์ (Gradient descent)



1. Nonrecurrent parallel identification model



2. General series-parallel identification model

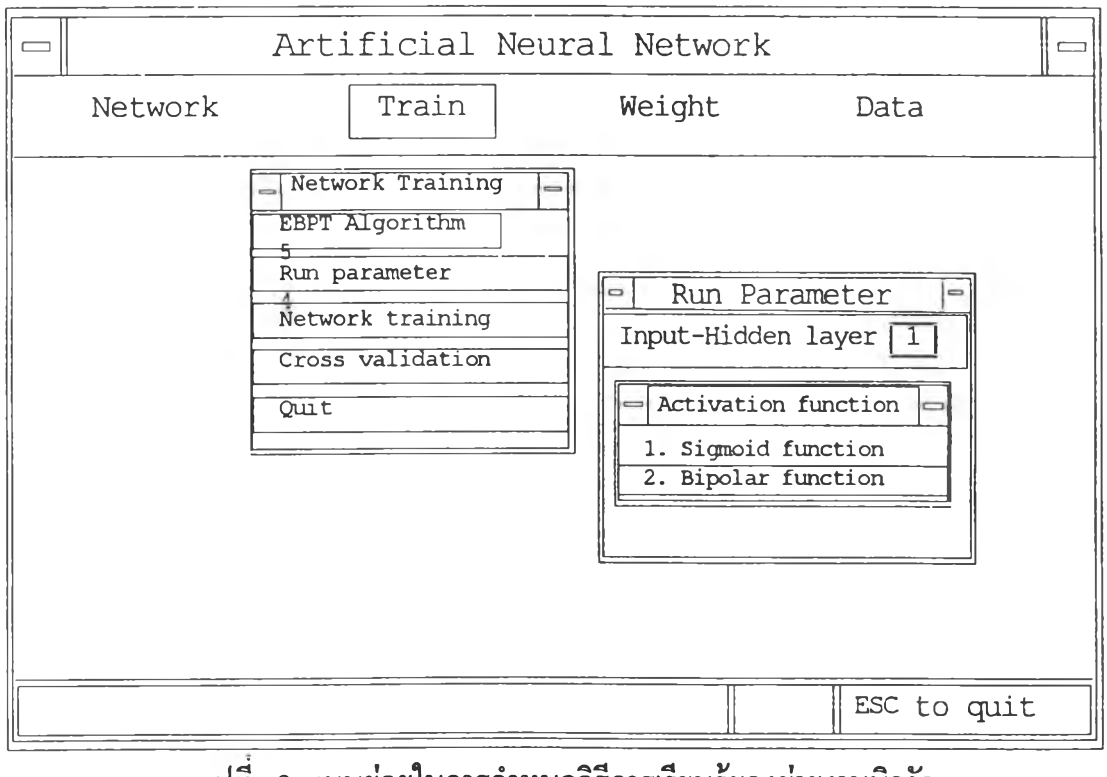
รูปที่ 8 แบบจำลองสำหรับการเรียนรู้ เลือกจากคำสั่ง EBPT Algorithm

ข.5.2 พารามิเตอร์สำหรับการเรียนรู้ (Run parameter)

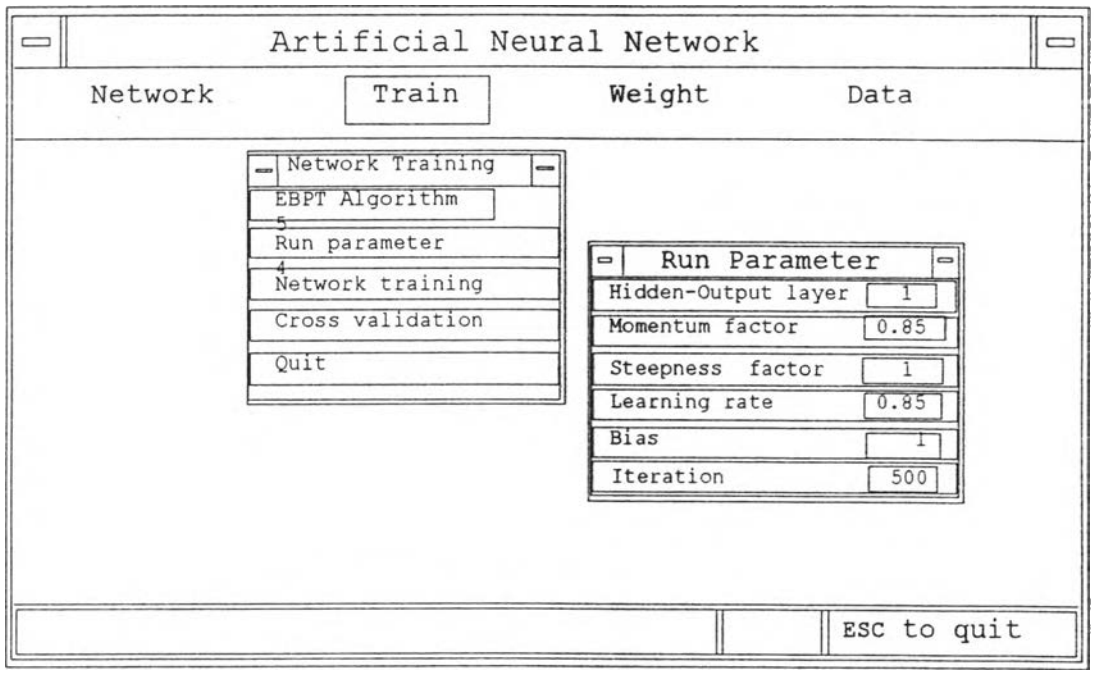
พารามิเตอร์มีผลโดยตรงต่อการเรียนรู้ประกอบด้วย ฟังก์ชันการกระตุ้น ที่ใช้ในการแปลงผลรวมของสัญญาณที่เข้ามายังนิวรอน ออกไปยังนิวรอนที่อยู่ในชั้นอื่น ฟังก์ชันดังกล่าวประกอบด้วย Sigmoidal function และ Bipolar function การกำหนดค่าดังแสดงในรูปที่ 9 สำหรับพารามิเตอร์อื่นๆที่สำคัญได้แก่ อัตราการเรียนรู้, โมเมนตัม เป็นต้น พารามิเตอร์เหล่านี้สามารถกำหนดได้โดยเมนู Run parameter แสดงดังในรูปที่ 10

ข.5.3 การฝึกข่ายงานนิวรัล (Network training)

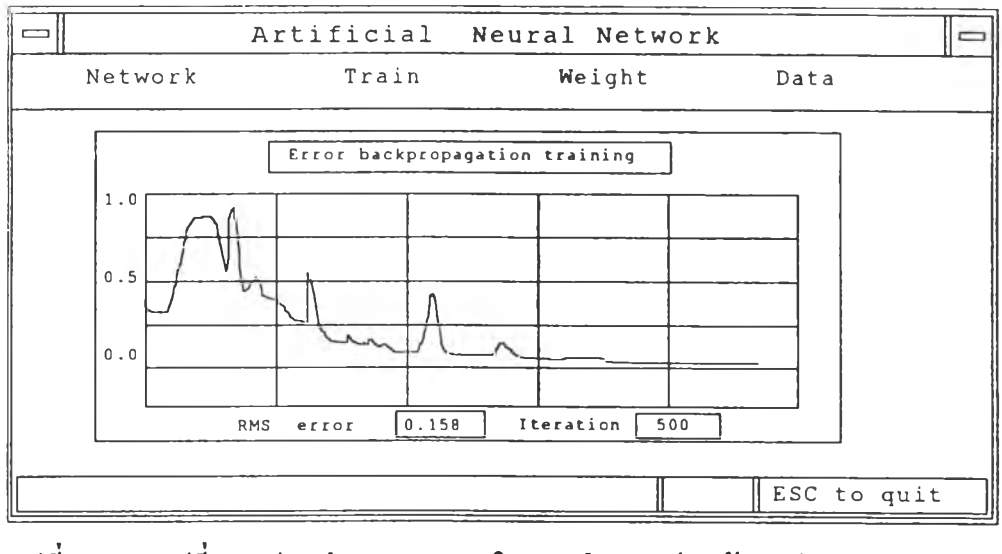
ในเมนูย่อยของ Network training นั้นเป็นการกำหนดให้ข่ายงานนิวรัลได้เรียนรู้กระบวนการตามค่าพารามิเตอร์, ชุดข้อมูลที่สอดคล้องกันระหว่างอินพุตกับเอาท์พุต และ ค่าน้ำหนัก ตามที่ได้กำหนดมาตั้งแต่เริ่ม ในรูปที่ 11 แสดงการเปลี่ยนแปลงค่าของ RMS error ในระหว่างการเรียนรู้



รูปที่ 9 เมนูย่อยในการกำหนดวิธีการเรียนรู้ของข่ายงานนิวรัล



รูปที่ 10 การกำหนดค่าพารามิเตอร์สำหรับใช้ในการเรียนรู้ของข่ายงานนิวรัล

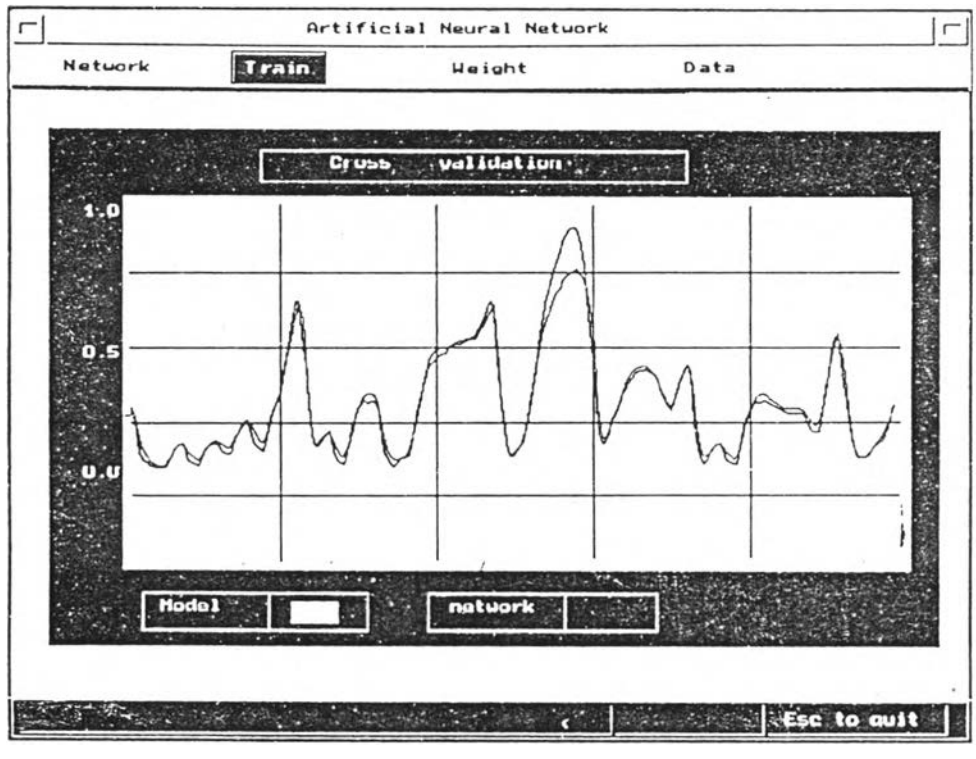


รูปที่ 11 การเปลี่ยนแปลงค่า RMS error ในระหว่างการเรียนรู้ของข่ายงานนิวรัล

ข.5.4 การทดสอบการเรียนรู้ของข่ายงานนิวรัล (Cross validation)

เมื่อการเรียนรู้ของข่ายงานนิวรัลถึงจุดหมายที่ต้องการแล้ว นั่นคือ ข่ายงานนิวรัลได้สร้างแบบจำลองของกระบวนการที่ได้เรียนรู้มา ดังนั้นเพื่อทดสอบแบบจำลองดังกล่าว จึงเลือกเมนู

ย่อย cross validation โดยใช้ข้อมูลที่ข่ายงานไม่เคยได้เรียนรู้มาก่อนมาทำการทดสอบ ผลลัพธ์ที่ได้แสดงในรูปที่ 11



รูปที่ 11 การทดสอบการเรียนรู้ของข่ายงานนิวรัลจาก Cross validdation

ประวัติผู้เขียน

นายสุรพล คำสุภา เกิดเมื่อวันที่ 8 ตุลาคม พ.ศ. 2507 สำเร็จการศึกษาในระดับชั้นมัธยมศึกษาปีที่ 5 จากโรงเรียนตากพิทยาคม เมื่อ พ.ศ. 2525 สำเร็จการศึกษา ปริญญาวิทยาศาสตรบัณฑิต สาขา เคมี จากมหาวิทยาลัยเชียงใหม่ เมื่อ พ.ศ. 2529 เคยทำงานที่บริษัท สยามเรชินและเคมีภัณฑ์ จำกัด เมื่อ พ.ศ. 2529-2532 ทำงาน บริษัท ทุนเท็กซ์ จำกัด เมื่อ พ.ศ. 2532-2533 ปัจจุบันทำงานในตำแหน่ง วิศวกรฝ่ายขาย บริษัท สยามราชธานี จำกัด