ภาคผนวก

อ้างอิงสมการที่ใช้ปรับค่าน้ำหนัก ตามที่ Least-Cross Correlation เป็นดังต่อไปนี้

$$\mathbf{w_{k+1}} = \mathbf{w_k} - \mu \cdot \Delta\varepsilon \cdot v\Delta\mathbf{P} \quad (1)$$

เมื่อ

$$\Delta\varepsilon = \varepsilon(\mathbf{w}_+) - \varepsilon(\mathbf{w}_-) \quad (2)$$

โดยที่สัญลักษณ์ในสมการมีความหมายดังที่แสดงไว้ดังต่อไปนี้

$\varepsilon(\mathbf{w_k})$ คือ ผลรวมความผิดพลาดกำลังสอง เมื่อค่าน้ำหนัก คือ $\mathbf{w}_k$

$\varepsilon(\mathbf{w}_+)$ คือ ผลรวมความผิดพลาดกำลังสอง เมื่อค่าน้ำหนัก คือ $\mathbf{w}_+$

$\varepsilon(\mathbf{w}_-)$ คือ ผลรวมความผิดพลาดกำลังสอง เมื่อค่าน้ำหนัก คือ $\mathbf{w}_-$

i คือ จำนวนสมาชิกในเวกเตอร์ค่าน้ำหนัก

K คือ ครั้งที่ปรับค่าน้ำหนัก 1,2,3,....

$\mathbf{w}_k$ คือ เวกเตอร์ค่าน้ำหนักก่อนปรับค่า $\mathbf{w}_k = [w_{k1} \quad w_{k2} \ldots \quad w_{ki}]^{\mathrm{T}}$

$\mathbf{w_{k+1}}$ คือ เวกเตอร์ค่าน้ำหนักหลังถูกปรับค่า

$\mathbf{w}_+$ คือ เวกเตอร์ค่าน้ำหนักที่ถูกรบกวนด้านบวก

$$\mathbf{w}_+ = \mathbf{w}_k + v \cdot \Delta\mathbf{P} = [w_{+1} \quad w_{+2} \ldots w_{+i}]^{\mathrm{T}}$$

$\mathbf{w}_-$ คือ เวกเตอร์ค่าน้ำหนักที่ถูกรบกวนด้านลบ

$$\mathbf{w}_- = \mathbf{w}_k - v \cdot \Delta\mathbf{P} = [w_{-1} \quad w_{-2} \ldots w_{-i}]^{\mathrm{T}}$$

t_p คือ ค่าเอาท์พุทที่ต้องการ ของข้อมูลอินพุทฝึกสอน ลำดับที่ p ค่าน้ำหนัก คือ $\mathbf{w}_{+k}$ ฝึกสอนโครงข่าย

$y_{p\,w_+}$ คือ ค่าเอาท์พุทโครงข่ายของข้อมูลอินพุทฝึกสอน ลำดับที่ p เมื่อค่าน้ำหนัก คือ $\mathbf{w}_+$

$y_{p\,w_-}$ คือ ค่าเอาท์พุทโครงข่ายของข้อมูลอินพุทฝึกสอน ลำดับที่ p เมื่อค่าน้ำหนัก คือ $\mathbf{w}_-$

$y_{p\,w_k}$ คือ ค่าเอาท์พุทโครงข่ายของข้อมูลอินพุทฝึกสอน ลำดับที่ p เมื่อ ค่าน้ำหนัก คือ $\mathbf{w}_k$

N คือ จำนวนอินพุทฝึกสอนทั้งหมด

v คือ ค่าคงที่ควบคุมลักษณะการกระจายตัวแปรสุ่ม

$\Delta\mathbf{P}$ คือ เวกเตอร์การรบกวน มีลักษณะการกระจายตัวแปรสุ่มแบบเกาส์เซียนปรกติ (Normal Gaussian Distribution) มีค่าเบี่ยงเบนมาตรฐานเท่ากับ 1.0

$v \cdot \Delta\mathbf{P}$ คือ เวกเตอร์ค่าการรบกวน มีลักษณะการกระจายตัวแปรสุ่มเกาซ์เซี่ยน (Gaussian Distribution) มีค่าเบี่ยงเบนมาตรฐาน เท่ากับ v

$\Delta\varepsilon$ คือ ผลต่างระหว่างค่าความผิดพลาดกำลังสองของค่าน้ำหนักที่ถูกรบกวนด้านบวก $(\mathbf{w} + v \cdot \Delta\mathbf{P})$และค่าความผิดพลาดกำลังสองของค่าน้ำหนักที่ถูกรบกวนด้านลบ $(\mathbf{w} - v \cdot \Delta\mathbf{P})$

μ คือ ค่าอัตราการเรียนรู้

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Writer Name : Dolhathai Kannai                         %
% Writing Date : 13 October 2010                         %
% Title : The Least-Cross Correlation Algorithm : LCC    %
% The adjustment equation : w_{k+1} = w_k - μ·Δε·vΔP     %
% where Δε = ε(w_+) - ε(w_-)                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all
close all
clc

%Initial the Standard Deviatoion : v = 0.01
mystd = 0.01;

%Initial the Learning Rate : μ = 70
mu = 70

%Initial the Mean Square Error Acceptable
mymse = 0.001;

%Initial the Maxinum Epoch
epoch = 60000;

% Input-Ouput Training is the AND logic function
input = [-1 -1 ; -1 1 ; 1 -1 ; 1 1];
output = [-1 -1 -1 1];
```

********** ดังที่แสดงในตารางต่อไปนี้ **********

Input X_1	Input X_2	Output y
-1	-1	1
-1	1	-1
1	-1	-1
1	1	1

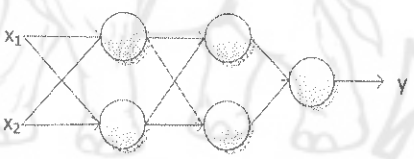
****

```
%Calculate number of input pattern :[4  2]
[numpattern numinput] = size(input);

%Calculate number of output : [1  4]
[numoutput numpattern]= size(output);

%Calculate Node Number at each Hidden layer
numhidden =  [2 2];
layer = size(numhidden);
```

********** ซึ่งการกำหนด ดังกล่าวโครงข่ายจะแสดงดังรูปต่อไปนี้ **********



numhidden = [2 2 1];

****

```
%Calculate the layer number
numlayer = layer(1,2);

%Calculate network : network = [2 2 2 1]
allnet = [numinput numhidden numoutput];

[loopr loopc] = size(allnet);

%Calculate matrix weight each layer : aa = [2 2; 2 2; 2 1]
for j0 = 1:loopc-1
    aa(j0,:)= [allnet(j0+1) allnet(j0)];
end

%Generate the weight using the pw function
allw = pw(aa);

[a b] = size(allw);
lw = a*b;
%Calculate  Weight vector : Wk
```

```
for j1 = 1:lw
    ww{j1} = allw{j1};
end

ip = 1:numpattern;

for ii = 1:epoch

%Calculate perturbation weight vector using the pn Function =
(normdelp : ΔP)
        normdelp = pn(aa);

%Calculate Gaussian perturbation weight vector = (delp : v·ΔP)

        for j1=1:lw
            delp{j1} = mystd*normdelp{j1};
        end

%Calculate Positive - Negative Perturbation Weight vector
(W+ = (Wk + v·ΔP), W_= (Wk − v·ΔP))

       for j2 = 1:lw
            wp{j2} = ww{j2} + delp{j2};
            wn{j2} = ww{j2} - delp{j2};
        end

        for I = 1:numpattern

%calculate network output using W+ : y_pw+

            patinp = input(ip(I),:);
            patout = output(:,ip(I));
            x = 1;
            y = 2;
            for j3 = 1:b-1
                out=[];
                for j4 = 1:numhidden(j3)
                    out(j4) = sig(wp{y}(1,j4) + patinp*wp{x}(j4,:)');
                end
                x = x+2;
                y = y+2;
                patinp= out;
            end
            x;y;
            for j5 = 1:numoutput
                outp(j5) = wp{y}(1,j5) + patinp*wp{x}(j5,:)';
            end
            errorp(I,:) = (patout - outp').^2;

%calculate network output using W_: y_pw_

            patinp = input(ip(I),:);
            patout = output(:,ip(I));
            x = 1;
```

```
            y = 2;
            for j6 = 1:b-1
                outn = [];
                for j7 = 1:numhidden(j6)
                    outn(j7) = sig(wn{y}(1,j7) +
patinp*wn{x}(j7,:)');
                end
                x = x+2;
                y = y+2;
                patinp= outn;
            end
            x;y;
            for j8 = 1:numoutput
                outnn(j8) = wn{y}(1,j8) + patinp*wn{x}(j8,:)';
            end
            errorn(I,:) = (patout - outnn').^2;
        end
%calculate the sum of squared error of the W+ : ε(W+)

        sumerrorp = sum(sum(errorp));

%calculate the sum of squared error of the W- : ε(W-)
        sumerrorn = sum(sum(errorn));

%calculate the difference of sum squared error of the W+ and W- : Δε
        delsumerror = sumerrorp - sumerrorn;

%update the weight the following equation : W(k+1) = Wk − μ · Δε · vΔP
        for j9 = 1:lw
            ww{j9} = ww{j9} - (mu*delsumerror).*delp{j9};
        end

%calculate the MSE acceptable for stop the training
        for i = 1:numpattern
            patinp = input(ip(i),:);
            patout = output(:,ip(i));
            x = 1;
            y = 2;
            for j10 = 1:b-1
                outs = [];
                for j11 = 1:numhidden(j10)
                    outs(j11) = sig(ww{y}(1,j11) +
patinp*ww{x}(j11,:)');
                end
                x = x+2;
                y = y+2;
                patinp= outs;
            end
            x;y;
            for j12=1:numoutput
                outss(j12) = ww{y}(1,j12) + patinp*ww{x}(j12,:)';
                Y{A,i} = outss;
            end

            error(i,:) = (patout - outss').^2;
        end
```

```
        sumerror = sum(error);

        mse(ii) = sum(sumerror)/numpattern;

        if (mse(ii) < mymse)
            ii
            break;
        end
    end


% plot the surface to show the network output
[x2,x1] = meshgrid([-2:0.1:2],[-2:0.1:2]);
patterns = size(x1);

for jj = 1:patterns
    for kk = 1:patterns

        itest = [x1(jj,kk),x2(jj,kk)];

        x = 1;
        y = 2;
        for k10 = 1:b-1
            outs = [];
            for k11 = 1:numhidden(k10)
                outs(k11) = sig(ww{y}(1,k11) + itest*ww{x}(k11,:)');
            end
            x = x+2;
            y = y+2;
            itest= outs;
        end
        x;y;
        for k12=1:numoutput
            outss(k12) = ww{y}(1,k12) + itest*ww{x}(k12,:)';
        end
        A(jj,kk) = outss ;
    end
    %         yy = sim(net,itest);
end

z = A;
figure(2); mesh(x1,x2,z);
xlabel('x_1'); ylabel('x_2');zlabel('output');
axis vis3d
```

Function

% Function name : pw for generating the weight : (w_k)

```
function out = pw(n)
[a b] = size(n);
for j1= 1:a
    ww{j1} = randn(n(j1,:)) ;
    bb{j1} = randn(1,n(j1,1));
end
out = [ww ; bb];
```

% Function name : pn for generating the perturbation weight : (ΔP)

```
function out = pn(n)
[a b] = size(n);
for j1= 1:a
    ww{j1} = randn(n(j1,:)) ;
    bb{j1} = randn(1,n(j1,1));
end
out = [ww ; bb];
```

*** bold letter to show the vector value ***