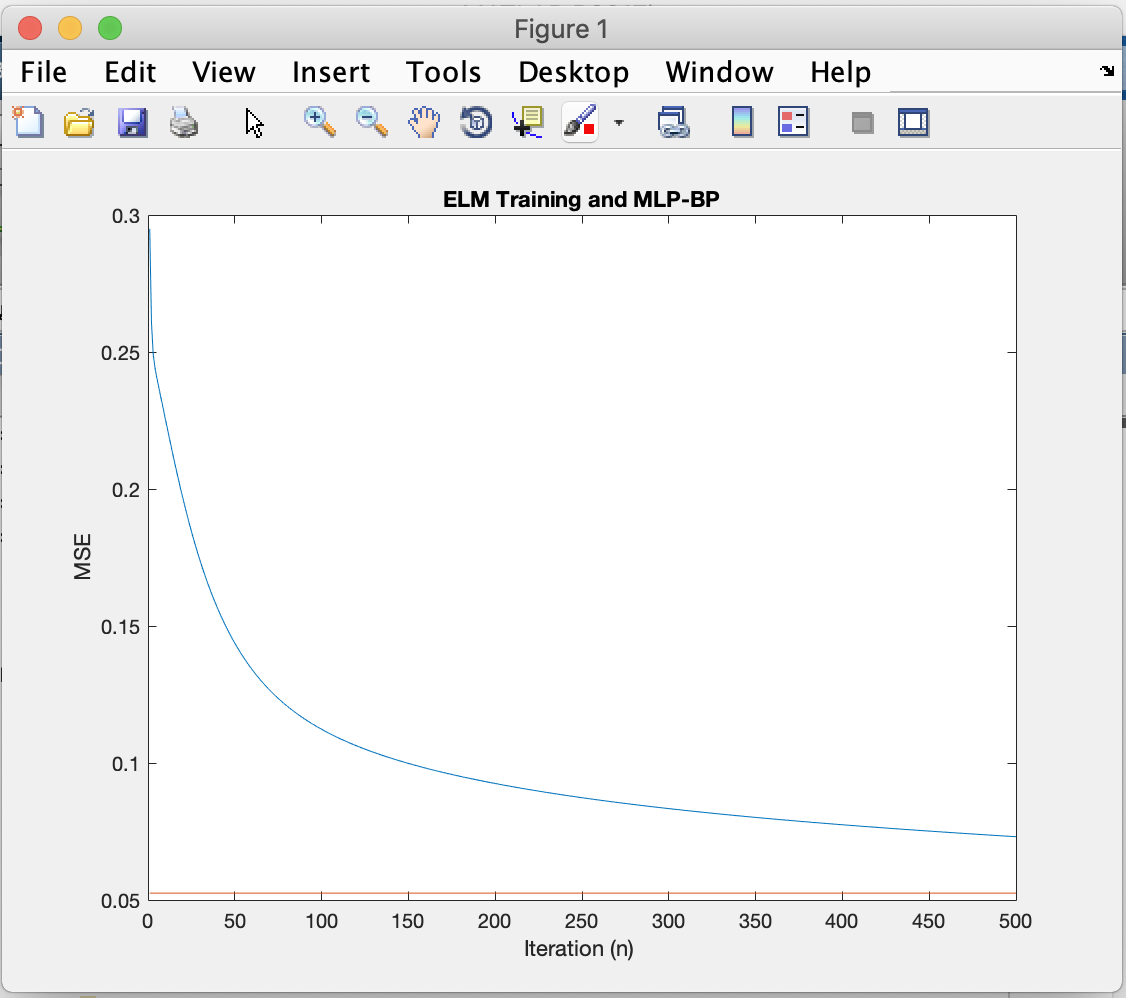
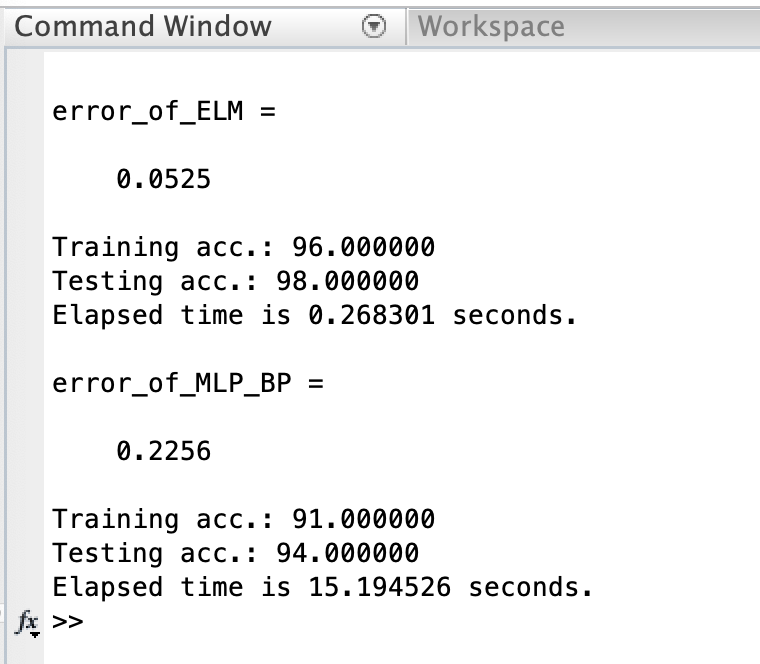
7.8.1 จากตัวอย่างรหัสโปรแกรมข้างต้น ให้ทำการศึกษาและทดลองกับปัญหาจริงอื่นๆ โดยเปรียบเทียบประสิทธิภาพในด้านเวลาและความถูกต้องกับตัวแบบโครงข่ายประสาทเทียมหลายชั้นแบบแพร่ย้อนกลับ





ผลการประมวลผลโปรแกรมการจำแนกประเภทข้อมูล Iris โดยใช้โครงข่ายประสาทเทียมหลายชั้นแบบแพร่ย้อนกับโดยใช้ชั้นซ่อน 1 ชั้น เมื่อเทียบกับการใช้ Extreme Learning Machine จะสามารถเห็นได้ถึงความแตกต่างของประสิทธภาพในด้านของเวลา ซึ่ง Extreme Learning Machine ทำได้ดีกว่ามาก

**Source Code ของโปรแกรม**

%Chaiwat Kaewmukdasawan 593020413-8 sec1

close all;clear all;clc;

% รับข้อมูลเข้ามา

dataset = load('iris.txt');

% dataset ตั้งแต่ช่วงคอลัม 1-4

x = dataset(:,1:4);

xmax = max(x); %ค่าสูงสุด

xmin = min(x); %ค่าต่ำสุด

% normalize ปรับให้ data ตั้งแต่คอลัม 1-4 อยู่ในช่วง 0-1

Xnorm = (x-xmin)./(xmax-xmin);

% T คือ target 1,0,0 , 0,1,0 , 0,0,1 แยกเป็นชื่อแต่ละ data

T = dataset(:,5:end);

% sz คือ size ของ data ทั้งหมด เท่ากับ 150

sz = size(dataset,1);

% I คือ Random permutation สุ่มค่าจำนวน 150 เป็นการสลับค่าไปมาจนถึง 150

I = randperm(sz);

%แบ่ง data สำหรับ xTrain ตั้งแต่ 1-100

xTrain = Xnorm(I(1:100),:);

% แบ่ง data สำหรับ xTest ตั้งแต่ 101-150

xTest = Xnorm(I(101:end),:);

% แบ่ง data สำหรับ tTrain ตั้งแต่ 1-100

tTrain = T(I(1:100),:);

% แบ่ง data สำหรับ tTest ตั้งแต่ 1-150

tTest = T(I(101:end),:);

clear X T

%%%%%%%%%%%%%%%%%%%%%%%%%% ELM %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

tic;

%Training phase

dim = size(xTrain,2);

hidden\_node = 8;

input\_weight = unifrnd(-1,1,dim,hidden\_node);

bias = unifrnd(-1,1,1,hidden\_node);

hidden\_layer = 1./(1+exp(-xTrain\*input\_weight+repmat(bias,size(xTrain,1),1)));

output\_weight = pinv(hidden\_layer)\*tTrain;

output\_train = hidden\_layer\*output\_weight;

%Test phase

hidden\_layer = 1./(1+exp(-xTest\*input\_weight+repmat(bias,size(xTest,1),1)));

output\_test = hidden\_layer\*output\_weight;

error\_of\_ELM = mse(tTrain-output\_train)

Y = output\_train;

%Performance of Traning

[tmp,Index1] = max(Y,[],2);

[tmp,Index2] = max(tTrain,[],2);

fprintf('Training acc.: %f \n',mean(mean(Index1 == Index2))\*100);

Y = output\_test;

% Performance of Testing

[tmp,Index1] = max(Y,[],2);

[tmp,Index2] = max(tTest,[],2);

fprintf('Testing acc.: %f \n',mean(mean(Index1 == Index2))\*100);

toc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear X T

%Model MLP-BP Learning : 1 hidden layer

tic;

n = 0.01;

L = 50; %Hidden node

wi = rands(size(xTrain,2),L);

bi = rands(1,L);

wo = rands(L,size(tTrain,2));

bo = rands(1,size(tTrain,2));

E = [];

for k = 1:500

for i = 1:size(xTrain,1)

H = logsig(xTrain(i,:)\*wi + bi);

Y = logsig(H\*wo + bo);

e = tTrain(i,:) - Y;

dy = e .\* Y .\* (1-Y);

dH = H .\* (1-H) .\* (dy\*wo');

wo = wo + n \* H'\*dy;

bo = bo + n \* dy;

wi = wi + n \* xTrain(i,:)'\*dH;

bi = bi + n \* dH;

end

H = logsig(xTrain\*wi + repmat(bi,size(xTrain,1),1));

Y = logsig(H\*wo + repmat(bo,size(xTrain,1),1));

E(k) = mse(tTrain - Y);

plot(E); title('ELM Training and MLP-BP');

hold on

test(k) = error\_of\_ELM;

plot(test);

hold off

xlabel('Iteration (n) '); ylabel('MSE');

drawnow;

end

error\_of\_MLP\_BP = E(10)

%Train Pedic

H = logsig(xTrain\*wi + repmat(bi,size(xTrain,1),1));

Y = logsig(H\*wo + repmat(bo,size(xTrain,1),1));

%Performance of Traning

[tmp,Index1] = max(Y,[],2);

[tmp,Index2] = max(tTrain,[],2);

fprintf('Training acc.: %f \n',mean(mean(Index1 == Index2))\*100);

%Testing Pedic

H = logsig(xTest\*wi + repmat(bi,size(xTest,1),1));

Y = logsig(H\*wo + repmat(bo,size(xTest,1),1));

% Performance of Testing

[tmp,Index1] = max(Y,[],2);

[tmp,Index2] = max(tTest,[],2);

fprintf('Testing acc.: %f \n',mean(mean(Index1 == Index2))\*100);

toc;