Pattern Recognition Final Project

Digit recognition with a Back-Propagation Network

資訊所 黃子睿 P76054088

Topic description

Digit recognition with a Back-Propagation Network

使用倒傳遞神經網路來進行數字辨識

此專題將使用Matlab的Neural Network Toolbox來實現倒傳遞神經網路的演算法，並達到辨識數字的目的，實際架構和參數將依實驗結果好壞作調整。

Objective

希望藉由此次實驗能夠成功辨識出手寫數字。

How to design the classifier

辨識分類器將使用4層神經網路

第一層為輸入圖片，第二層為160個神經元，第三層為10個神經元，第四層為輸出層。

輸出層為的向量，數值最高的索引將會成為辨識結果，因為數字為0~9因此辨識結果為索引值-1。

隱藏層的激發函數都為sigmoid

神經元數目為經過實驗後能夠得出比較良好的分類結果。

由於圖片維度過高，因此使用PCA先對訓練資料做降維前處理，達到節省訓練時間和得到良好分類結果。

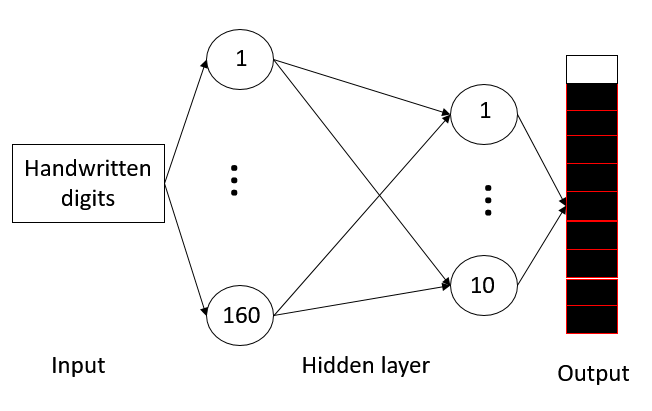


圖1類神經網路架構

sample data

Yann LeCunn – MNIST Digit Recognition

<http://yann.lecun.com/exdb/mnist/>

<http://www.cs.nyu.edu/~roweis/data.html>

how to use your system

執行FinalProject.m

並等待約2分鐘後產生執行結果。

execution results

訓練資料將由10種數字(0~9)各1000筆資料來做訓練

測試資料將由10種數字(0~9)各800筆資料來做測試

神經網路epoch設為3000

使用訓練資料來做測試得到的平均分類正確率為

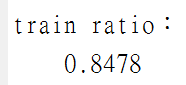


圖2訓練資料分類出的正確比例

以下為的矩陣，用來看每樣數字的分類正確率，舉例來說第一列第一行為數字1分類成數字1的機率0.9800，而第一列第三行則為數字1分類成數字3的機率0.001，其他行以此類推。

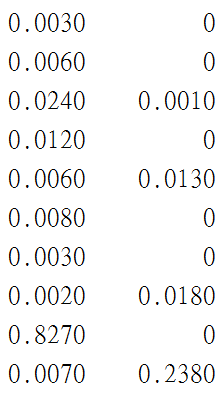
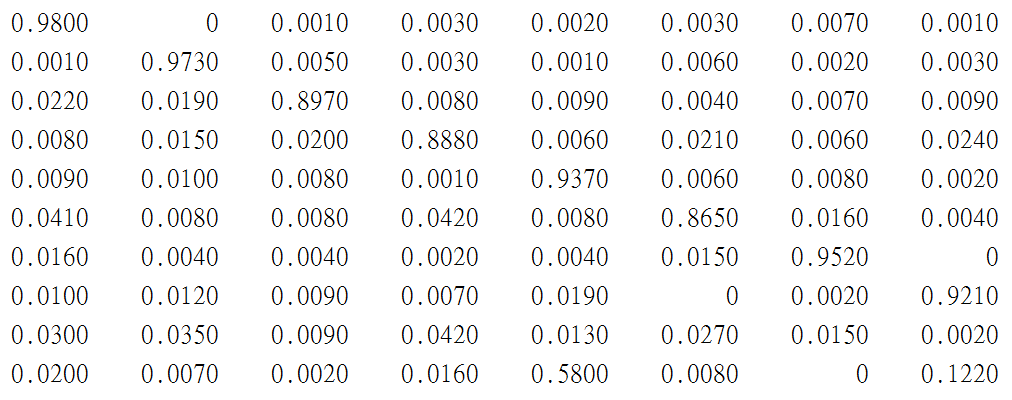


圖3訓練資料分類出的矩陣

使用測試資料來做測試得到的平均分類正確率為

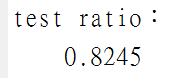


圖4 測試資料分類出的正確比例

以下為的矩陣，用來看每樣數字的分類正確率，舉例來說第一列第一行為數字1分類成數字1的機率0.9800，而第一列第三行則為數字1分類成數字3的機率0.001，其他行以此類推。

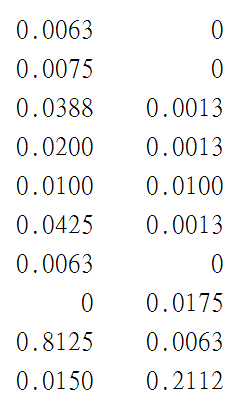
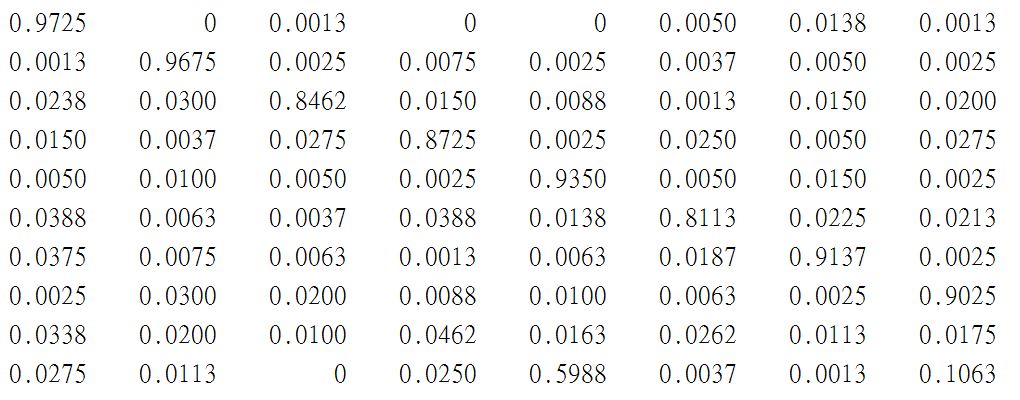


圖5測試資料分類出的矩陣

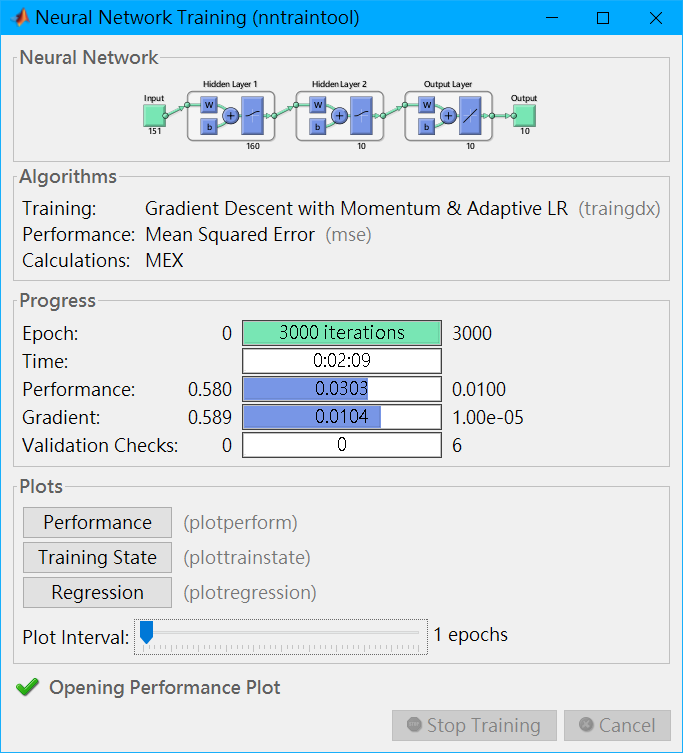


圖6類神經網路架構

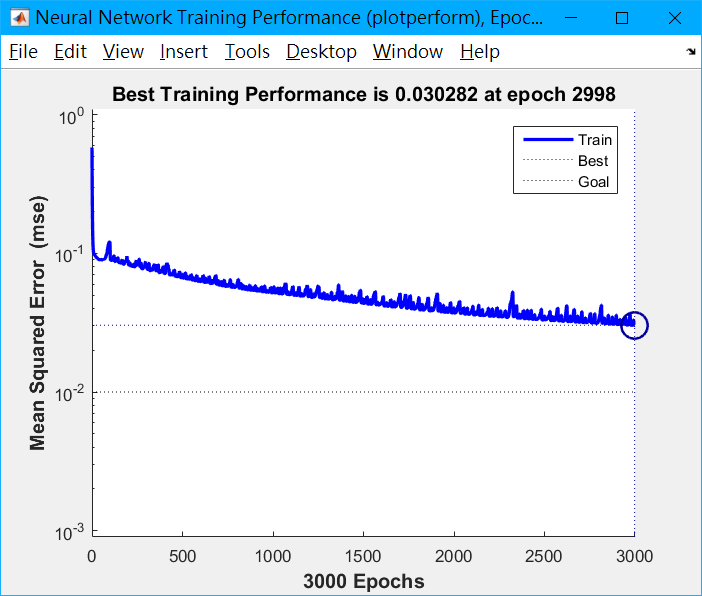


圖7實驗收斂圖

discussion

從實驗結果來看，由訓練資料和測試資料相比訓練資料期望會得到比較好的分類結果，數字也是正確的。訓練和測試結果的分類準確率達到80%以上可以說是很準確，但仍有一些數字沒辦法精準分類，舉例來說觀察圖3和圖5得到的分類矩陣可以知道數字0常常被分成數字4，可能是因為兩者在外型有點類似的緣故，因為中間都含有一個空洞，如果增加epoch和訓練資料筆數預期會得到更好的分類結果，但訓練時間則會顯著提升，epoch原本是設成10,000但訓練太花時間，MSE也沒有明顯下降，所以我將epoch設置成3,000比較能夠在短時間內訓練出來並看到不錯的分類結果，從圖6來看可以知道epoch設為3,000的時候訓練時間約為2分鐘左右，而圖7則可以發現epoch在3,000就已經有良好的收斂，因此我將epoch設為3,000。

source code

clc

clearvars

load mnist\_all.mat

dataRow = [5923 6742 5958 6131 5842 5421 5918 6265 5851 5949 ...

980 1135 1032 1010 982 892 958 1028 974 1009];

dataString = {'train0' 'train1' 'train2' 'train3' 'train4' 'train5' ...

'train6' 'train7' 'train8' 'train9' ...

'test0' 'test1' 'test2' 'test3' 'test4' 'test5' 'test6' ...

'test7' 'test8' 'test9'};

% Merge

trainDataCount = 1000;

trainImage = train0(1:trainDataCount, 1:end);

% Label = 1~10 for convinence

VT(1:trainDataCount) = 1;

for i = 2:10

Data = dataString(i);

eval(['trainImage(1+trainDataCount\*(i-1):trainDataCount\*i, 1:end) = double(', Data{1}, '(1:trainDataCount, 1:end));']);

VT(1+trainDataCount\*(i-1):trainDataCount\*i) = i;

end

testDataCount = 800;

testImage = test0(1:testDataCount, 1:end);

VTt(1:testDataCount) = 1;

for i = 12:20

Data = dataString(i);

eval(['testImage(1+testDataCount\*(i-11):testDataCount\*(i-10), 1:end) = double(', Data{1}, '(1:testDataCount, 1:end));']);

VTt(1+testDataCount\*(i-11):testDataCount\*(i-10)) = i-10;

end

% Label, The highest value is the result

T = zeros(10, trainDataCount\*10);

for i = 1:trainDataCount\*10

T(VT(i), i) = 1;

end

Tt = zeros(10, testDataCount\*10);

for i = 1:testDataCount\*10

Tt(VTt(i), i) = 1;

end

% Transfer to double

trainImage = im2double(trainImage);

testImage = im2double(testImage);

% PCA

[coef, ~, latent] = pca(trainImage);

lat = cumsum(latent)./sum(latent);

a = find(lat > 0.95);

mm = a(1);

P = trainImage \* coef(:, 1:mm);

Pt = testImage \* coef(:, 1:mm);

P = P';

Pt = Pt';

% P = trainImage';

% Pt = testImage';

net = newff(P, T, [160 10], {'logsig','logsig'}, 'traingdx');

net.trainParam.epochs = 3000;

net.trainParam.goal = 1e-2;

net.trainParam.lr = 1;

net.trainParam.mc = 0.1;

net.divideFcn = '';

disp('training...');

net = train(net, P, T);

sim1 = sim(net, P);

[~ , Y1] = max(sim1);

ratio1 = mean(Y1 == VT);

disp('train ratio：'); disp(ratio1);

trainError = zeros(10, 10);

for i = 1:trainDataCount\*10

trainError(VT(i), Y1(i)) = trainError(VT(i), Y1(i)) + 1;

end

trainError = trainError / trainDataCount

sim2 = sim(net, Pt);

[~ , Y2] = max(sim2);

ratio2 = mean(Y2 == VTt);

disp('test ratio：'); disp(ratio2);

testError = zeros(10, 10);

for i = 1:testDataCount\*10

testError(VTt(i), Y2(i)) = testError(VTt(i), Y2(i)) + 1;

end

testError = testError / testDataCount