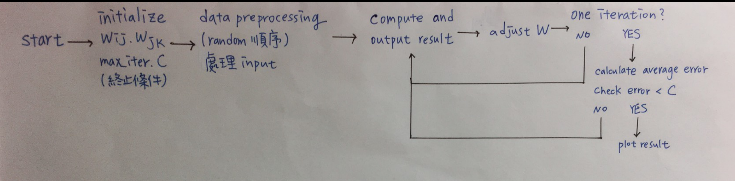
**Project #2(a): Multilayer Perceptron 0516222 許芳瑀**

**Q1:**

**做法:**

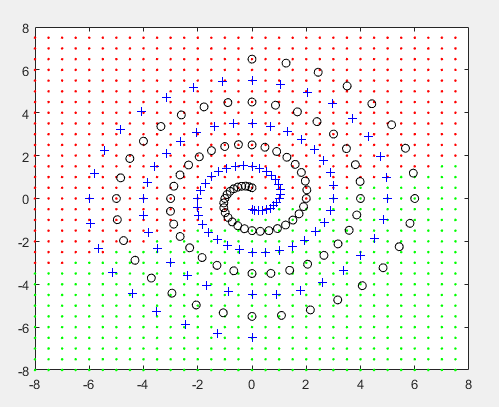
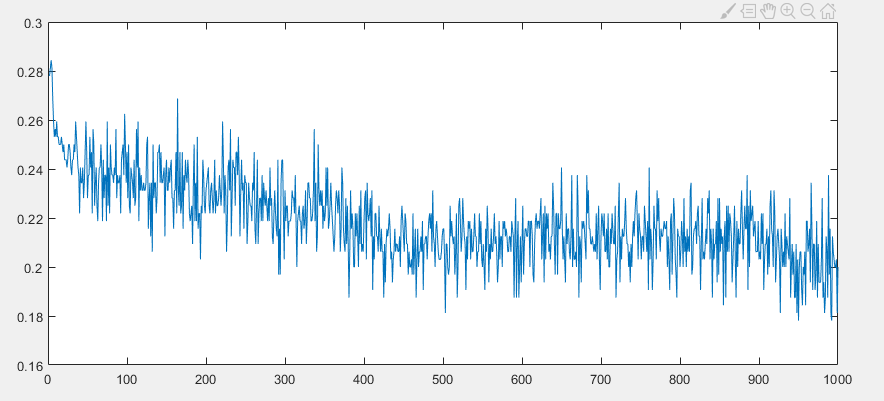
這題我使用了兩層hidden layer ，第一層20個node，第二層10個node，最一開始處理資料時，我先將x y存到oi(加上常數1)，並且將資料順序random，讓同種類的點分散，開始train之後，便是接受input，其中採用的active function是sigmoid function，經過跟weight內積之後，output的輸出是1或0(因為只有要分兩類)，然後調整weight，一個iteration之後計算error，直到我指定的最大次數或是e小於指定的數字。

**Flowchart:**



**結果:**

decision region average error vs. iteration

**討論:**

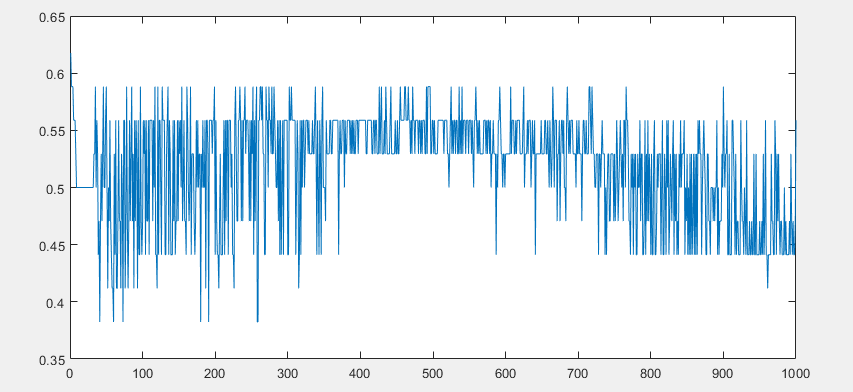
How to determine the hidden node number in each problem?

這題花了蠻久的時間，一開始我只有用一層layer，50個node，發現效果蠻差的，所以多設計一層，以及把iteration的次數也加高到10000，但效果還是不好。

Any experiment

我只用了八成的資料training，剩下的兩成用於test，再每次iteration之後都算一次答案的正確性。

Accuracy



**Code:**

sl = [];

oj = [];

ok = [];

ol = [];

% last result

accuracy=[];

e=[];

% regulize

oi(:,1) = (oi(:,1)-mean(oi(:,1)) ) / std(oi(:,1));

oi(:,2) = (oi(:,2)-mean(oi(:,2)) ) / std(oi(:,2));

while iter<=max\_iter

disp(iter);

error = 0;

for i = 1:1:train\_total

% foward

sj = oi(i,:)\*wij;

oj = [sj 1];

sk = oj\*wjk;

ok= [sk 1];

sl = ok\*wkl;

ol = round(sigmf(sl,[1,0]));

prev\_wjk = wjk;

prev\_wkl = wkl;

% update wkl

f = sigmf(sl,[1,0]);

for kk=1:num\_k+1

wkl(kk,1) = wkl(kk,1) + rate \* (ans(i)-ol)\* f \* (1-f) \* ok(kk);

end

% update wjk

for jj = 1:num\_j+1

for kk = 1:num\_k

f = sigmf(sk(kk),[1,0]);

fl = sigmf(sl,[1,0]);

sum = (ans(i,1) - ol ) \* fl \* (1-fl) \* prev\_wkl(kk,1);

wjk(jj,kk) = wjk(jj,kk) + rate\* sum \* f \* (1-f) \* oj(jj);

end

end

for jj = 1:num\_j

i=[0:96]

theta\_i = i \* pi / 16;

theta\_j = i \* pi / 16;

r\_i = 6.5 .\* (104 - i) / 104;

r\_j = 6.5 .\* (104 - i) / 104;

N=250;

theta1 = linspace(-180,180, N)\*pi/360;

r = 8 ;

x1 = r\_i .\* sin(theta\_i);

y1 = r\_i .\* cos(theta\_i);

x2 = -x1;

y2 = -y1;

% set ans 0/1 class

oi = [x1.' y1.' zeros(97,1)+1 ; x2.' y2.' zeros(97,1)+1 ];

ans = [zeros(97,1) ;zeros(97,1)+1 ];

% rearrange

total=194;

train\_total=160;

test\_total = 34;

r = randperm(total);

oi(:,1) = oi(r,1);

oi(:,2) = oi(r,2);

ans = ans(r);

test=oi([train\_total+1 :total], :);

test\_ans = ans([train\_total+1 :total],:);

% set initial value

num\_j = 20;

num\_k = 10;

wij = rand(3,20);

wjk = rand(21,10);

wkl = rand(11, 1);

i\_wij = wij;

i\_wjk = wjk;

i\_wkl = wkl;

rate = 0.001;

C = 0.01;

max\_iter = 15000;

iter = 1;

sj = [];

sk = [];

% update wij

sum=0;

for ii = 1:3

for jj = 1:num\_j

fj = sigmf(sj(jj),[1,0]);

for kk=1:num\_k

fk=sigmf(sk(kk),[1 0]);

sum = sum + (ans(i,1) - ol)\*fk\*(1-fk)\*wjk(jj,kk);

end

wij(ii,jj) = wij(ii,jj) + rate\* sum \* fj \* (1-fj) \* oi(i,ii);

end

end

error = error + (ans(i) - ol).^2/2;

end

% 畫圖

for ix=-16:16

for iy=-16:16

dx=0.5\*(ix-1);

dy=0.5\*(iy-1);

oi=[dx dy 1];

sj = oi\*wij;

oj = [sj 1];

sk = oj\*wjk;

ok= [sk 1];

sl = ok\*wkl;

ol = round(sigmf(sl,[1,0]));

if ol==1

figure(3);

axis([-8 8 -8 8]);

plot(dx,dy, 'g .',x1,y1,'ko',x2,y2,'b+');

hold on;

elseif ol==0

figure(3);

axis([-8 8 -8 8]);

plot(dx,dy, 'r .',x1,y1,'ko',x2,y2,'b+');

hold on;

end

end

end

e(iter) = error/train\_total;

if error < C

break;

end

acc=0;

for t=1:test\_total

sj = test(t,:)\*wij;

oj = [sj 1];

sk = oj\*wjk;

ok= [sk 1];

sl = ok\*wkl;

ol = round(sigmf(sl,[1,0]));

if ol== test\_ans(t)

acc = acc+1;

end

end

acc = acc/test\_total;

accuracy(iter) = acc;

iter = iter + 1;

end

figure(1);

plot(e);

figure(2);

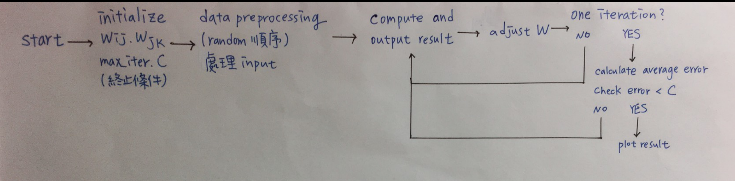
plot(accuracy);

**Q2:**

**做法:**

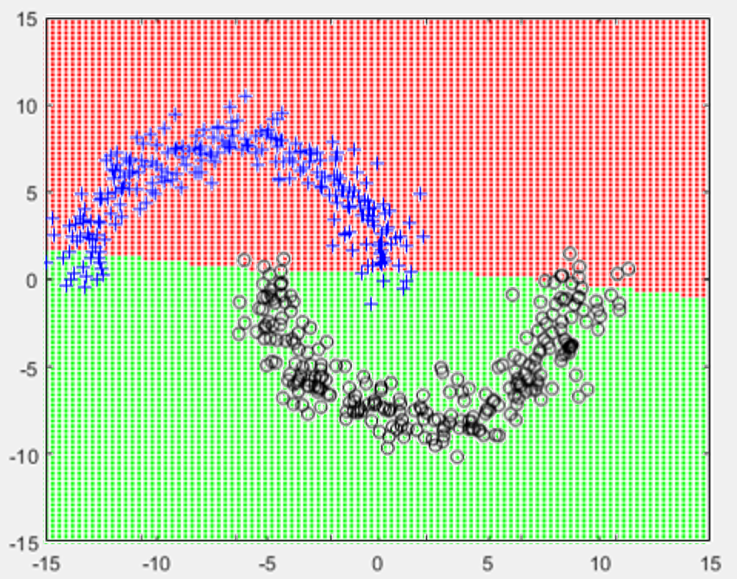
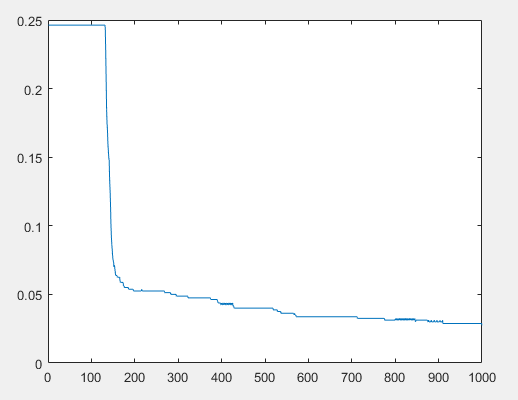
這題我使用了一層hidden layer、四個node，最一開始處理資料時，我先將x y存到oi(加上常數1)，並且將資料順序random，讓同種類的點分散，開始train之後，便是接受input，其中採用的active function是sigmoid function，經過跟weight內積之後，output的輸出是1或0(因為只有要分兩類)，然後調整weight，一個iteration之後計算error，直到我指定的最大次數或是e小於指定的數字。

**Flowchart:**



**結果:**

decision region average error vs. iteration

**討論:**

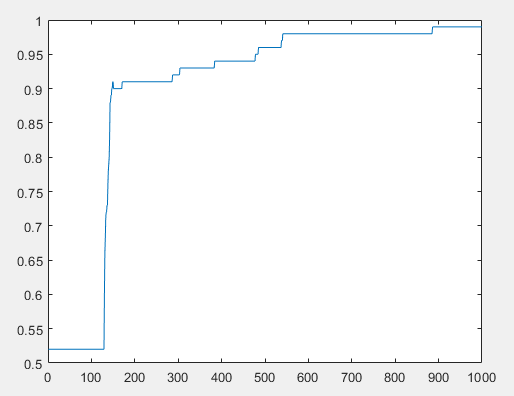
How to determine the hidden node number in each problem?

因為這題的分布看起來沒有很複雜，所以我決定用一層layer就好，然後一開始選擇了三個node，發現結果蠻普通的，所以改用四個node。

Any experiment

我只用了八成的資料training，剩下的兩成用於test，再每次iteration之後都算一次答案的正確性，發現結果達到了大於九成的準確率。

Accuracy



如何決定初始的weight

一開始我的weight都是用random決定，後來發現這樣有可能會產生很極端的結果，所以我就挑了一次最好的結果，並改用那次初始的weight。

**Code:**

clear;

N=250;

theta1 = linspace(-180,180, N)\*pi/360;

r = 8 ;

x1 = -5 + r\*sin(theta1)+randn(1,N);

y1 = r\*cos(theta1)+randn(1,N);

x2 = 5 + r\*sin(theta1)+randn(1,N);

y2 = -r\*cos(theta1)+randn(1,N);

% set ans 0/1 class

oi = [x1.' y1.' zeros(250,1)+1 ; x2.' y2.' zeros(250,1)+1 ];

ans = [zeros(250,1) ;zeros(250,1)+1 ];

% rearrange

total=500;

train\_total=400;

test\_total = 100;

r = randperm(total);

oi(:,1) = oi(r,1);

oi(:,2) = oi(r,2);

ans = ans(r);

test=oi([train\_total+1 :total], :);

test\_ans = ans([train\_total+1 :total],:);

% set initial value

wij = [0.749120729449102,0.127013893617675,0.113789807451064,0.492415695453145;0.147735486618356,0.916921352125118,0.960227492817496,0.361313401952203;0.00288774509561229,0.672121707253300,0.119383120967445,0.470000490547994];

wjk =[0.0587797688209679;0.330011701173118;0.264520477657796;0.245917666364952;0.947676359396337];

rate = 0.0001;

C = 0.0025;

max\_iter = 1000;

iter = 1;

sj = [];

sk = [];

oj = [];

ok = [];

% last result

accuracy=[];

e=[];

while iter<=max\_iter

error = 0;

for i = 1:1:train\_total

% foward

sj = oi(i,:)\*wij;

oj = [sigmf(sj,[1,0]) 1];

sk = oj\*wjk;

ok= round(sigmf(sk,[1,0]));

prev\_wjk = wjk;

% update w2

f = sigmf(sk,[1,0]);

for j=1:5

wjk(j,1) = wjk(j,1) + rate \* (ans(i)-ok)\* f \* (1-f) \* oj(j);

end

% update w1

for ii = 1:3

for jj = 1:4

f = sigmf(sj(jj),[1,0]);

fk = sigmf(sk,[1,0]);

sum = (ans(i,1) - ok ) \* fk \* (1-fk) \* prev\_wjk(jj,1);

wij(ii,jj) = wij(ii,jj) + rate\* sum \* f \* (1-f) \*prev\_wjk(jj) \* f \* (1-f) \* oi(i,ii);

end

end

error = error + (ans(i) - ok).^2/2;

end

e(iter) = error/train\_total;

if error < C

break;

end

acc=0;

for t=1:test\_total

sj = test(t,:)\*wij;

oj = [sigmf(sj,[1,0]) 1];

sk = oj\*wjk;

ok= round(sigmf(sk,[1,0]));

prev\_wjk = wjk;

if ok== test\_ans(t)

acc = acc+1;

end

end

acc = acc/test\_total;

accuracy(iter) = acc;

iter = iter + 1;

end

figure(1);

plot(e);

figure(2);

plot(accuracy);

for ix=-50:50

for iy=-50:50

dx=0.3\*(ix-1);

dy=0.3\*(iy-1);

oi=[dx dy 1];

sj = oi\*wij;

oj = sigmf(sj,[1,0]);

oj = [oj 1];

sk = oj\*wjk;

ok= round(sigmf(sk,[1,0]));

if ok==1

figure(3);

axis([-15 15 -15 15]);

plot(dx,dy, 'g .');

hold on;

elseif ok==0

figure(3);

axis([-15 15 -15 15]);

plot(dx,dy, 'r .');

hold on;

end

end

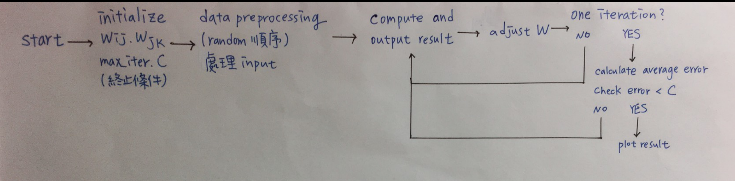
end

**Q3:**

**做法:**

這題我使用了一層hidden layer、五個node，最一開始處理資料時，我先將x y存到oi(加上常數1)，並且將資料順序random，讓同種類的點分散，開始train之後，便是接受input，其中採用的active function是sigmoid function，經過跟weight內積之後，output的輸出是四個數值，哪一個數值為1就是哪一類(其他數值為0)，然後調整weight，一個iteration之後計算error，直到我指定的最大次數或是e小於指定的數字。

**Flowchart:**



**討論:**

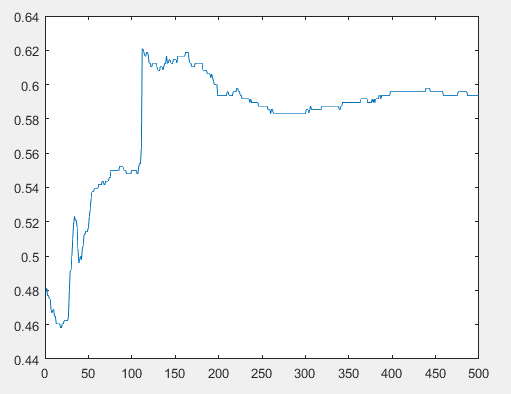
How to determine the hidden node number in each problem?

因為這題的分布看起來沒有很複雜，所以我決定用一層layer就好，然後一開始選擇了三個node，發現結果蠻普通的，所以改用五個node。

Any experiment

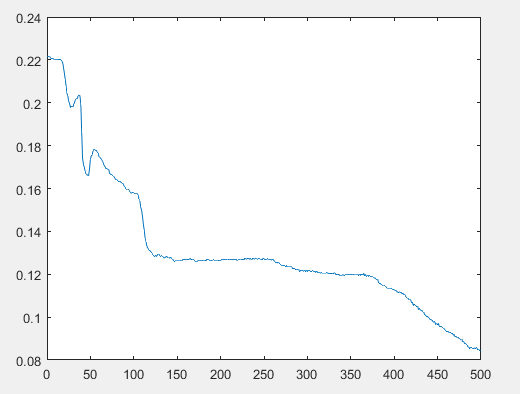
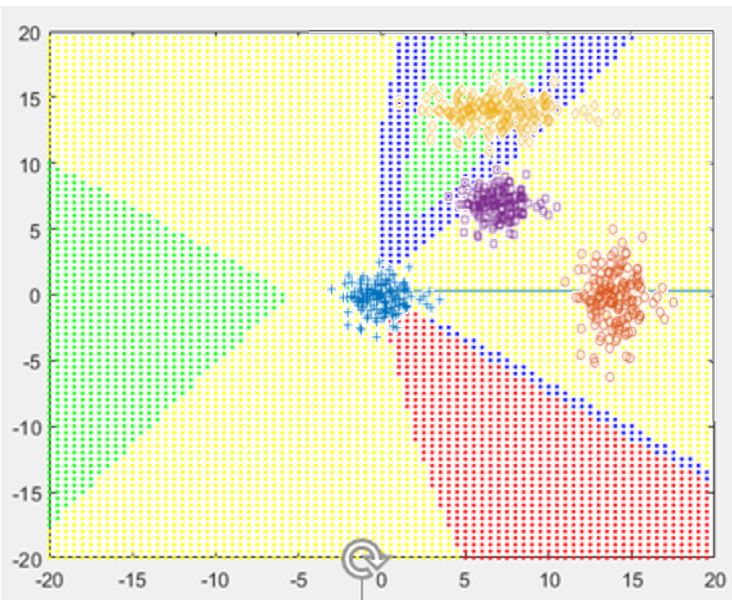
我只用了八成的資料training，剩下的兩成用於test，再每次iteration之後都算一次答案的正確性。

accuracy



**結果:**

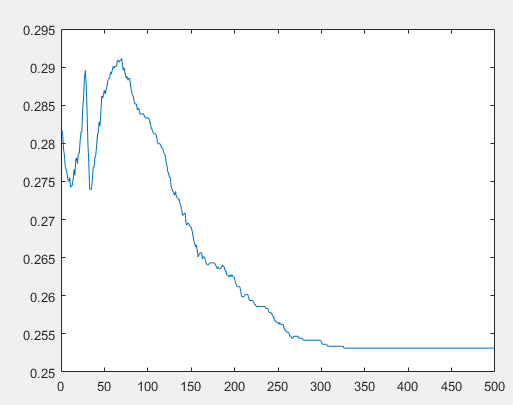
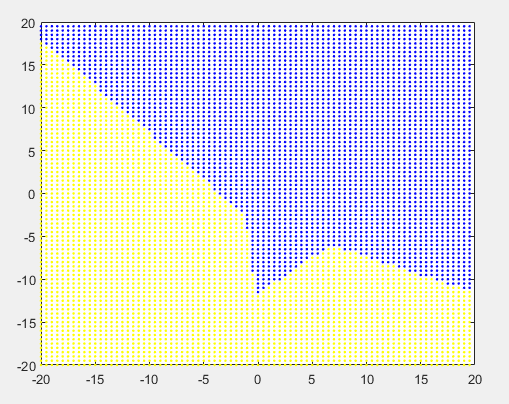
decision region average error vs. iteration



(b) 將 activation function 改為 ReLu function

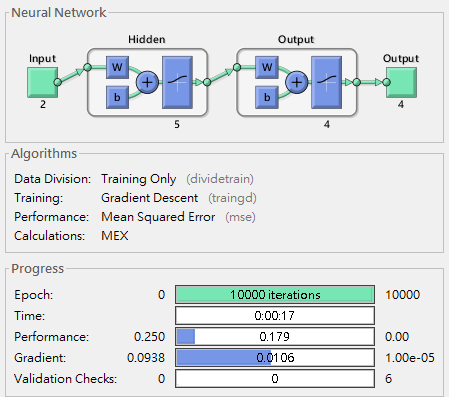
將active function改成ReLu之後，我試過很多其他微調(像是rate之類)，發現效果都很差，可能這題並沒有很適合ReLu funcion當作active function。

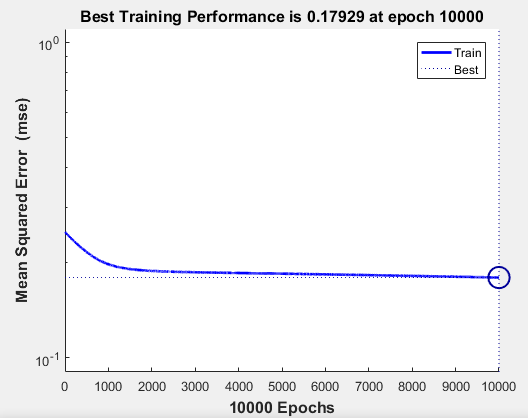
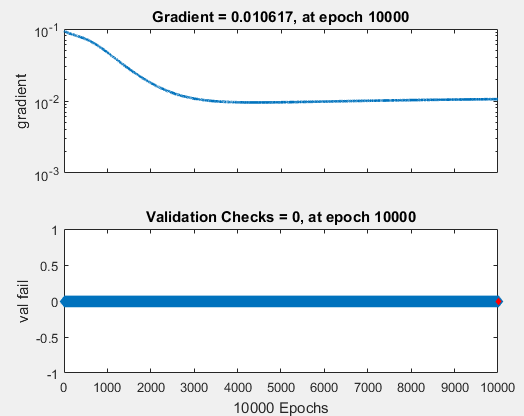
decision region average error vs. iteration

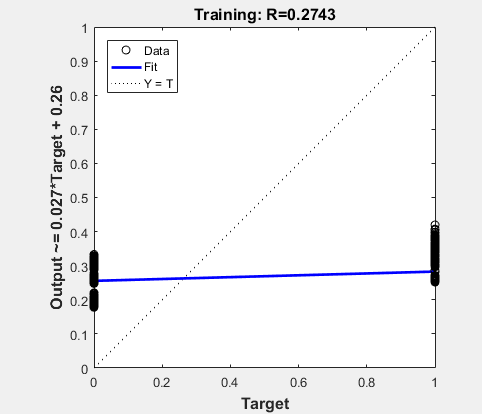
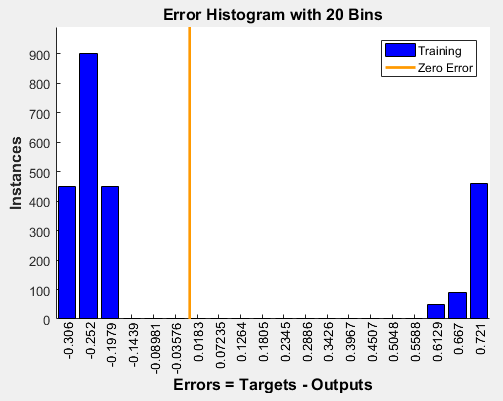


(c) 利用 MATLAB 的 neural networks 的 toolbox

比較:用函式很快就跑出結果了，但是perfoemance和gradient都沒有表現的很好，我也試了其他train function，發現用其他的效果會比gradient decent好很多，像試trainscg之類的



**討論**

**Code:**

clear;

% set data1

mu = [0;0];

sigma = [1 0; 0 1];

rng default % For reproducibility

r = mvnrnd(mu,sigma,150);

temp = zeros(150,1)+1;

oi = [r,temp];

% set data2

mu = [14;0];

sigma = [1 0; 0 4];

rng default % For reproducibility

r = mvnrnd(mu,sigma,150);

oi = [ oi ;r temp];

% set data3

mu = [7;14];

sigma = [4 0; 0 1];

rng default % For reproducibility

r = mvnrnd(mu,sigma,150);

oi = [ oi ;r temp];

% set data4

mu = [7;7];

sigma = [1 0; 0 1];

rng default % For reproducibility

r = mvnrnd(mu,sigma,150);

oi = [ oi ;r temp];

% set ans

ans = [zeros(150,1)+1 ;zeros(150,1)+2 ;zeros(150,1)+3; zeros(150,1)+4];

% rearrange

total=600;

train\_total=total\*0.8;

test\_total = total\*0.2;

r = randperm(total);

oi(:,1) = oi(r,1);

oi(:,2) = oi(r,2);

ans = ans(r);

t=[1:total];

final\_ans = zeros(600,4);

for i=1:600

final\_ans(i,ans(i)) = 1;

end

test=oi([train\_total+1 :total], :);

test\_ans = final\_ans([train\_total+1 :total],:);

% set initial value

wij = rand(3,5).\*2-1;

wjk = rand(6,4).\*2-1;

i\_wij=wij;

i\_wjk=wjk;

rate = 0.0001;

C = 0.005;

max\_iter = 500;

iter = 1;

sj = [];

sk = [];

oj = zeros(600,6);

ok = [];

% last result

accuracy=[];

e=[];

% regulize

oi(:,1) = (oi(:,1)-mean(oi(:,1)) ) / std(oi(:,1));

oi(:,2) = (oi(:,2)-mean(oi(:,2)) ) / std(oi(:,2));

while iter<=max\_iter

error = 0;

for i = 1:1:train\_total

% foward

sj = oi(i,:)\*wij;

oj = [sigmf(sj,[1,0]) 1];

sk = oj\*wjk;

ok= round(sigmf(sk,[1,0]));

prev\_wjk = wjk;

% update w2

for k=1:4

f = sigmf(sk(k),[1,0]);

for j=1:6

wjk(j,k) = wjk(j,k) + rate \* (final\_ans(i,k)-ok(k))\* f \* (1-f) \* oj(j);

end

end

% update w1

for ii = 1:3

for jj = 1:5

f = sigmf(sj(jj),[1,0]);

sum=0;

for kk=1:4

fk = sigmf(sk(kk),[1,0]);

sum = sum + (final\_ans(i,k) - ok(k) ) \* fk \* (1-fk) \* prev\_wjk(jj,kk);

end

wij(ii,jj) = wij(ii,jj) + rate\* sum \* f \* (1-f) \* oi(i,ii);

end

end

for tt=1:4

error = error + (final\_ans(i,tt) - ok(tt)).^2/8;

end

end

e(iter) = error/train\_total;

if error < C

break;

end

acc=0;

for t=1:test\_total

sj = test(t,:)\*wij;

oj = [sigmf(sj,[1,0]) 1];

sk = oj\*wjk;

ok= round(sigmf(sk,[1,0]));

prev\_wjk = wjk;

for temp=1:4

if ok(temp) == test\_ans(t,temp)

acc = acc+1;

end

end

end

acc = acc/test\_total/4;

accuracy(iter) = acc;

iter = iter + 1;

end

figure(1);

plot(e);

figure(2);

plot(accuracy);

for ix=-40:40 %% ±q-15~15

for iy=-40:40

dx=0.5\*(ix-1);

dy=0.5\*(iy-1);

input=[dx dy 1];

sj = input\*wij;

oj = sigmf(sj,[1,0]);

oj = [oj 1];

sk = oj\*wjk;

ok= round(sigmf(sk,[1,0]));

figure(3);

if ok(1)==1

axis([-20 20 -20 20]);

plot(dx,dy, 'g .');

hold on;

end

if ok(2)==1

axis([-20 20 -20 20]);

plot(dx,dy, 'r .');

hold on;

end

if ok(3)==1

axis([-20 20 -20 20]);

plot(dx,dy, 'b .');

hold on;

end

if ok(4)==1

axis([-20 20 -20 20]);

plot(dx,dy, 'y .');

hold on;

end

end

end

%%% relu

wij = rand(3,5)\*2-1;

wjk = rand(6,4)\*2-1;

i\_wij=wij;

i\_wjk=wjk;

rate = 0.0001;

iter=1;

temp2=[];

while iter<=max\_iter

error = 0;

for i = 1:1:train\_total

% foward

sj = oi(i,:)\*wij;

oj = [max(sj,0) 1];

sk = oj\*wjk;

temp2(i,:) = sk;

ok= max(sk,0) > 0;

prev\_wjk = wjk;

% update w2

for k=1:4

f=1;

if max(sj(k),0)==0

f=0;

end

for j=1:6

wjk(j,k) = wjk(j,k) + rate \* (final\_ans(i,k)-ok(k))\* f \* (1-f) \* oj(j);

end

end

% update w1

for ii = 1:3

for jj = 1:5

f=1;

if max(sj(jj),0)==0

f=0;

end

sum=0;

for kk=1:4

fk=1;

if max(sk(kk),0)==0

fk=0;

end

sum = sum + (final\_ans(i,k) - ok(k) ) \* fk \* prev\_wjk(jj,kk);

end

wij(ii,jj) = wij(ii,jj) + rate\* sum \* f \* oi(i,ii);

end

end

for tt=1:4

error = error + (final\_ans(i,tt) - ok(tt)).^2/8;

end

end

e(iter) = error/train\_total;

if error < C

break;

end

acc=0;

for t=1:test\_total

sj = test(t,:)\*wij;

oj = [max(sj,0) 1];

sk = oj\*wjk;

ok= max(sk,0)>0;

prev\_wjk = wjk;

for temp=1:4

if ok(temp) == test\_ans(t,temp)

acc = acc+1;

end

end

end

acc = acc/test\_total/4;

accuracy(iter) = acc;

iter = iter + 1;

end

figure(4);

plot(e);

figure(5);

plot(accuracy);

for ix=-40:40 %% ±q-15~15

for iy=-40:40

dx=0.5\*(ix-1);

dy=0.5\*(iy-1);

oi=[dx dy 1];

sj = oi\*wij;

oj = [max(sj,0) 1];

sk = oj\*wjk;

ok= max(sk,0)>0;

figure(6);

if ok(1)==1

axis([-20 20 -20 20]);

plot(dx,dy, 'g .');

hold on;

end

if ok(2)==1

axis([-20 20 -20 20]);

plot(dx,dy, 'r .');

hold on;

end

if ok(3)==1

axis([-20 20 -20 20]);

plot(dx,dy, 'b .');

hold on;

end

if ok(4)==1

axis([-20 20 -20 20]);

plot(dx,dy, 'y .');

hold on;

end

end

end

% use toolbox

clear;

% set data1

mu = [0;0];

sigma = [1 0; 0 1];

rng default % For reproducibility

r = mvnrnd(mu,sigma,150);

oi = [r];

% set data2

mu = [14;0];

sigma = [1 0; 0 4];

rng default % For reproducibility

r = mvnrnd(mu,sigma,150);

oi = [ oi ;r];

% set data3

mu = [7;14];

sigma = [4 0; 0 1];

rng default % For reproducibility

r = mvnrnd(mu,sigma,150);

oi = [ oi ;r];

% set data4

mu = [7;7];

sigma = [1 0; 0 1];

rng default % For reproducibility

r = mvnrnd(mu,sigma,150);

oi = [ oi ;r ];

% set ans

ans = [zeros(150,1)+1 ;zeros(150,1)+2 ;zeros(150,1)+3; zeros(150,1)+4];

% rearrange

total=600;

train\_total=total\*0.8;

test\_total = total\*0.2;

r = randperm(total);

oi(:,1) = oi(r,1);

oi(:,2) = oi(r,2);

ans = ans(r);

t=[1:total];

final\_ans = zeros(600,4);

for i=1:600

final\_ans(i,ans(i)) = 1;

end

test=oi([train\_total+1 :total], :);

test\_ans = final\_ans([train\_total+1 :total],:);

net = feedforwardnet(5);

net.layers{2}.size = 4;

net.inputs{1}.size=2;

net.layers{1}.transferFcn = 'logsig';

net.layers{2}.transferFcn = 'logsig';

net.divideFcn = 'dividetrain';

net.trainFcn='traingd';

net=train(net,oi.',final\_ans.');