**Project #3: Multilayer Perceptron**

**劉波洲 0540153**

1. Two spiral problem:

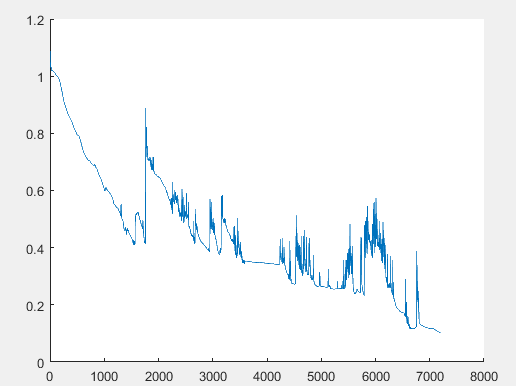
Hiden layer:2

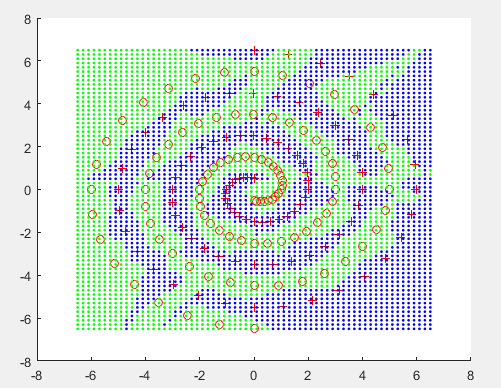
Hiden nodes:2 x 30 x 15 x 2

learning rate parameter ：0.2

stop criterion: error<0.1 or iteration >20000

以下是結果。調整速度比較滿，最後結果還是存在一定的誤差。





Flowchart：



Code:

%Two spiral problem

clear

i=0:1:96;

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

theta=pi.\*i./16;

data(:,1,1)=r.\*sin(theta);

data(:,2,1)=r.\*cos(theta);

data(:,1,2)=-r.\*sin(theta);

data(:,2,2)=-r.\*cos(theta);

ni=30;

nj=15;

nk=2;

wkj=randn(nk,nj+1);

wkj\_tmp=wkj;

wji=randn(nj,ni+1);

wji\_tmp=wji;

wib=randn(ni,3);

wkj\_low=wkj;

wji\_low=wji;

wib\_low=wib;

ob=[0 0 1]';

si=zeros(ni,1);

oi=zeros(ni+1,1);

sj=zeros(nj,1);

oj=zeros(nj+1,1);

sk=zeros(nk,1);

ok=sk;

dk=sk;

lowerlimit=0.1;

itermax=20000;

eta=0.2;

beta=0;

iter=0;

error\_avg=10;

deltak=zeros(1,nk);

deltaj=zeros(1,nj);

deltai=zeros(1,ni);

sumbackkj =zeros(1,nj);

sumbackji =zeros(1,ni);

while (error\_avg >lowerlimit ) && (iter<itermax)

error=0;

iter=iter+1;

for n=1:97

for class=1:2

if class==1

dk=[1 0]';

else

dk=[0 1]';

end

%forward computation

ob=[data(n,1,class) data(n,2,class) 1]';

for i=1:ni

si(i)=wib(i,:)\*ob;

oi(i)=1/(1+exp(-si(i)));

end

oi(ni+1)=1;

for j=1:nj

sj(j)=wji(j,:)\*oi;

oj(j)=1/(1+exp(-sj(j)));

end

oj(nj+1)=1;

for k=1:nk

sk(k)=wkj(k,:)\*oj;

ok(k)=1/(1+exp(-sk(k)));

end

error=error+sum(abs(dk-ok));

%error=error+(dk-ok)'\*(dk-ok)/2;

%backward learning

for k=1:nk

deltak(k)=(dk(k)-ok(k))\*ok(k)\*(1-ok(k));

end

for j=1:nj

for k=1:nk

wkj\_tmp(k,j)=wkj(k,j)+eta\*deltak(k)\*oj(j);

end

end

for j=1:nj

sumbackkj(j)=0.0;

for k=1:nk

sumbackkj(j)=sumbackkj(j)+deltak(k)\*wkj(k,j);

end

deltaj(j)=sumbackkj(j)\*oj(j)\*(1-oj(j));

end

for i=1:ni

for j=1:nj

wji\_tmp(j,i)=wji(j,i)+eta\*deltaj(j)\*oi(i);

end

end

for i=1:ni

sumbackji(i)=0.0;

for j=1:nj

sumbackji(i)=sumbackji(i)+deltaj(j)\*wji(j,i);

end

deltai(i)=sumbackji(i)\*oi(i)\*(1-oi(i));

end

for b=1:3

for i=1:ni

wib(i,b)= wib(i,b)+eta\*deltai(i)\*ob(b);

end

end

wkj=wkj\_tmp;

wji=wji\_tmp;

end

end

ite(iter)=iter;

error\_avg=error/194;

error\_r(iter)=error\_avg;

end

figure(2)

hold on

plot(ite,error\_r);

figure(3)

hold on

plot(data(:,1,1),data(:,2,1),'r+');

plot(data(:,1,2),data(:,2,2),'ro');

for x=-6.5:0.2:6.5

for y=-6.5:0.2:6.5

ob=[x y 1]';

for i=1:ni

si(i)=wib(i,:)\*ob;

oi(i)=1/(1+exp(-si(i)));

end

oi(ni+1)=1;

for j=1:nj

sj(j)=wji(j,:)\*oi;

oj(j)=1/(1+exp(-sj(j)));

end

oj(nj+1)=1;

for k=1:nk

sk(k)=wkj(k,:)\*oj;

ok(k)=1/(1+exp(-sk(k)));

end

if ok(1,1)> 0.5

plot(x,y,'b.');

elseif ok(1,1)<0.5

plot(x,y,'g.');

end

end

end

2. Double-moon problem:

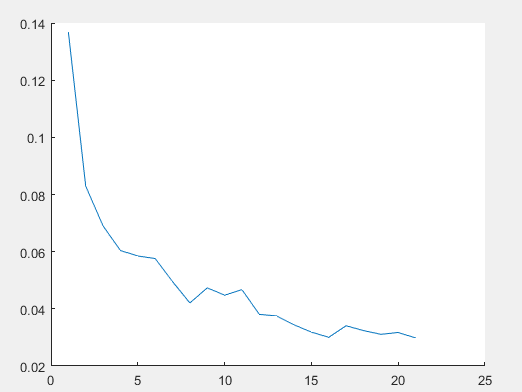
Hiden layer:1

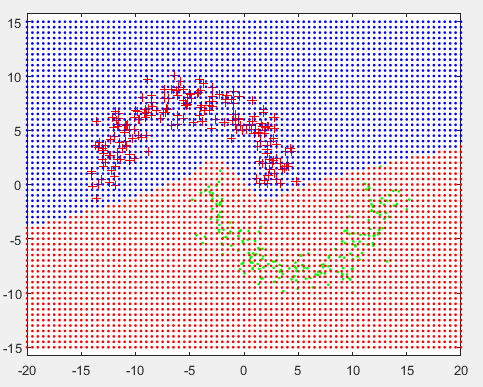
Hiden nodes:34

learning rate parameter eta and beta:在這裏采用慣性刹車的方式，每一次的調整量等于上次調整量加上當前計算出來的調整量，係數分別是0.3，0.7.

stop criterion:所有pattern計算完之後計算平均錯誤量，錯誤率門檻值為0.03在這個錯誤率上效果最好。最大iteration次數為5000.

以下是結果。看得出來，調整次數比較少，所有樣本點均準確識別。





Flowchart：



Code：

clear

N=200;

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);

data(:,1,1)=x1';

data(:,2,1)=y1';

data(:,1,2)=x2';

data(:,2,2)=y2';

nj=34;

nk=2;

wkj=randn(nk,nj+1);

wkj\_tmp=wkj;

wji=randn(nj,3);

wkj\_low=wkj;

wji\_low=wji;

olddelwkj=zeros(size(wkj));

olddelwji=zeros(size(wji));

oi=[0 0 1]';

% sj=[0 0 0 0 0 0 0 0]';

% oj=[0 0 0 0 0 0 0 0 1]';

sj=zeros(nk,1);

oj=zeros(nk+1,1);

sk=[0 0]';

ok=[0 0]';

dk=[0 0]';

lowerlimit=0.03;

itermax=5000;

eta=0.7;

beta=0.3;

iter=0;

error\_avg=10;

count=0;

minerror=10000;

deltak=zeros(1,nk);

sumback =zeros(1,nj);

deltaj=zeros(1,nj);

while (error\_avg >lowerlimit ) && (iter<itermax)

error=0;

iter=iter+1;

count=count+1;

for i=1:N

for class=1:2

if class==1

dk=[1 0]';

else

dk=[0 1]';

end

%forward computation

oi=[data(i,1,class) data(i,2,class) 1]';

for j=1:nj

sj(j)=wji(j,:)\*oi;

oj(j)=1/(1+exp(-sj(j)));

end

oj(nj+1)=1;

for k=1:nk

sk(k)=wkj(k,:)\*oj;

ok(k)=1/(1+exp(-sk(k)));

end

error=error+sum(abs(dk-ok));

%error=error+(dk-ok)'\*(dk-ok)/2;

%backward learning

for k=1:nk

deltak(k)=(dk(k)-ok(k))\*ok(k)\*(1.0-ok(k));

end

for j=1:nj

for k=1:nk

wkj\_tmp(k,j)=wkj(k,j)+eta\*deltak(k)\*oj(j)+beta\*olddelwkj(k,j);

olddelwkj(k,j)=eta\*deltak(k)\*oj(j)+beta\*olddelwkj(k,j);

end

end

for j=1:nj

sumback(j)=0.0;

for k=1:nk

sumback(j)=sumback(j)+deltak(k)\*wkj(k,j);

end

deltaj(j)=sumback(j)\*oj(j)\*(1-oj(j));

end

for ii=1:2

for j=1:nj

wji(j,ii)=wji(j,ii)+eta\*deltaj(j)\*oi(ii)+beta\*olddelwji(j,ii);

olddelwji(j,ii)=eta\*deltaj(j)\*oi(ii)+beta\*olddelwji(j,ii);

end

end

wkj=wkj\_tmp;

end

end

ite(iter)=iter;

error\_avg=error/(2\*N);

error\_r(iter)=error\_avg;

if error\_avg<minerror

count=0;

minerror=error\_avg;

wkj\_low=wkj;

wji\_low=wji;

elseif count>20

wkj=wkj\_low;

wji=wji\_low;

break;

end

end

figure(2)

hold on

plot(ite,error\_r);

figure(3)

plot(data(:,1,1),data(:,2,1),'r+');

hold on

plot(data(:,1,2),data(:,2,2),'g.');

for x=-20:0.5:20

for y=-15:0.5:15

oi=[x y 1]';

for j=1:nj

sj(j)=wji(j,:)\*oi;

oj(j)=1/(1+exp(-sj(j)));

end

oj(nj+1)=1.0;

for k=1:nk

sk(k)=wkj(k,:)\*oj;

ok(k)=1/(1+exp(-sk(k)));

end

if ok(1,1)> 0.5

plot(x,y,'b.');

elseif ok(1,1)<0.5

plot(x,y,'r.');

end

end

end

axis equal

3. 4 classes with Gaussian distribution

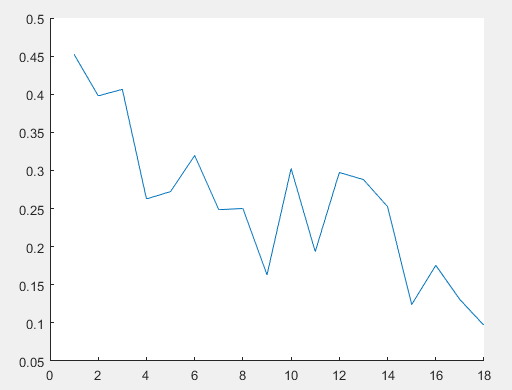
Hiden layer:1

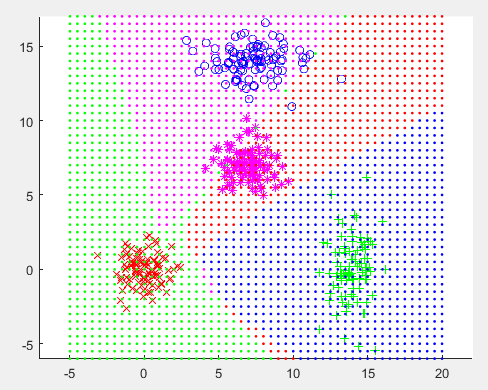
Hiden nodes:34

learning rate parameter eta and beta:在這裏采用慣性刹車的方式，每一次的調整量等于上次調整量加上當前計算出來的調整量，係數分別是0.3，0.7.

stop criterion:所有pattern計算完之後計算平均錯誤量，錯誤率門檻值為0.1，在這個錯誤率上效果最好。最大iteration次數為5000.

以下是結果。看得出來，調整次數比較少，大部分樣本點均準確識別。





Flowchart：



Code：

%four classes with Gaussian distribution

clear

N=100;

mu=[0 0]';

sigma=[1 0;0 1];

data(:,:,1)=mvnrnd(mu,sigma,N);

mu=[14 0]';

sigma=[1 0;0 4];

data(:,:,2)=mvnrnd(mu,sigma,N);

mu=[7 14]';

sigma=[4 0;0 1];

data(:,:,3)=mvnrnd(mu,sigma,N);

mu=[7 7]';

sigma=[1 0;0 1];

data(:,:,4)=mvnrnd(mu,sigma,N);

nj=200;

nk=2;

wkj=randn(nk,nj+1);

wkj\_tmp=wkj;

wji=randn(nj,3);

olddelwkj=zeros(size(wkj));

olddelwji=zeros(size(wji));

oi=[0 0 1]';

sj=zeros(nk,1);

oj=zeros(nk+1,1);

sk=[0 0]';

ok=[0 0]';

dk=[0 0]';

lowerlimit=0.1;

itermax=5000;

eta=0.7;

beta=0.3;

iter=0;

error\_avg=10;

count=0;

minerror=10000;

deltak=zeros(1,nk);

sumback =zeros(1,nj);

deltaj=zeros(1,nj);

while (error\_avg >lowerlimit ) && (iter<itermax)

error=0;

iter=iter+1;

count=count+1;

for i=1:N

for class=1:4

if class==1

dk=[1 0]';

elseif class==2

dk=[0 1]';

elseif class==3

dk=[1 1]';

elseif class==4

dk=[0 0]';

end

%forward computation

oi=[data(i,1,class) data(i,2,class) 1]';

for j=1:nj

sj(j)=wji(j,:)\*oi;

oj(j)=1/(1+exp(-sj(j)));

end

oj(nj+1)=1;

for k=1:nk

sk(k)=wkj(k,:)\*oj;

ok(k)=1/(1+exp(-sk(k)));

end

error=error+sum(abs(dk-ok));

%error=error+(dk-ok)'\*(dk-ok)/2;

%backward learning

for k=1:nk

deltak(k)=(dk(k)-ok(k))\*ok(k)\*(1-ok(k));

end

for j=1:nj

for k=1:nk

wkj\_tmp(k,j)=wkj(k,j)+eta\*deltak(k)\*oj(j)+beta\*olddelwkj(k,j);

olddelwkj(k,j)=eta\*deltak(k)\*oj(j)+beta\*olddelwkj(k,j);

end

end

for j=1:nj

sumback(j)=0.0;

for k=1:nk

sumback(j)=sumback(j)+deltak(k)\*wkj(k,j);

end

deltaj(j)=sumback(j)\*oj(j)\*(1-oj(j));

end

for ii=1:2

for j=1:nj

wji(j,ii)=wji(j,ii)+eta\*deltaj(j)\*oi(ii)+beta\*olddelwji(j,ii);

olddelwji(j,ii)=eta\*deltaj(j)\*oi(ii)+beta\*olddelwji(j,ii);

end

end

wkj=wkj\_tmp;

end

end

ite(iter)=iter;

error\_avg=error/(4\*N);

error\_r(iter)=error\_avg;

if error\_avg<minerror

count=0;

minerror=error\_avg;

elseif count>50

break;

end

end

figure(2)

hold on

plot(ite,error\_r);

figure(3)

hold on

plot(data(:,1,1),data(:,2,1),'rx')%class1

plot(data(:,1,2),data(:,2,2),'g+')%class2

plot(data(:,1,3),data(:,2,3),'bo')%class3

plot(data(:,1,4),data(:,2,4),'m\*')%class4

for x=-5:0.5:20

for y=-6:0.5:17

oi=[x y 1]';

for j=1:nj

sj(j)=wji(j,:)\*oi;

oj(j)=1/(1+exp(-sj(j)));

end

oj(nj+1)=1;

for k=1:nk

sk(k)=wkj(k,:)\*oj;

ok(k)=1/(1+exp(-sk(k)));

end

if ok(1,1)> 0.5 && ok(2,1)< 0.5

plot(x,y,'g.');

elseif ok(1,1) < 0.5 && ok(2,1)> 0.5

plot(x,y,'b.');

elseif ok(1,1) > 0.5 && ok(2,1)> 0.5

plot(x,y,'m.');

elseif ok(1,1) < 0.5 && ok(2,1)< 0.5

plot(x,y,'r.');

end

end

end

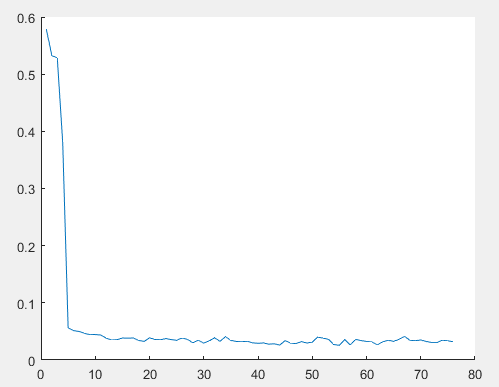
axis equal

討論

確定hidden nodes個數：基本采用兩種策略，逐漸增大和逐漸減小，慢慢找出比較合適的。同時采用一種策略：每種情況都測試一邊，通過平均錯誤率找到最好情況。

在找個數的過程中發現一個現象，并不是個數越多越好，會出現個數增多但效果更差的情況。在試驗2中，個數35和200的情況明顯好於個數為50的情況。原因可能是初始weight隨機生成，增大了不確定性。

在調整過程中爲避免出現下圖中的情況，采用策略：超過一定次數平均錯誤率沒有變小，就停止學習。



參靠文獻：

Reorganizing Neural Network System for Two Spirals and Linear Low-Density Polyethylene Copolymer Problems G. M. Behery,1 A. A. El-Harby,1 and Mostafa Y. El-Bakry2 https://www.hindawi.com/journals/acisc/2009/721370/