圖形識別概論Project #2(b)

0516218軒轅照雯

1. Results

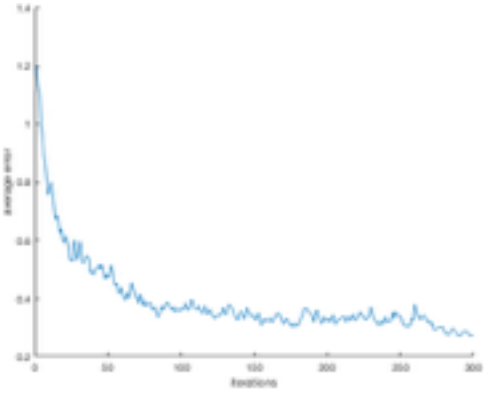
(a) activation function: sigmoidal function

(1) parameters: number of hidden layers: 1
number of hidden nodes: 32
learning rate parameter: 0.01
stop criterion: average error < 0.05 or iterations > 300

(2) CPU time in learning

```
Elapsed time is 7689.099661 seconds.
```

(3) average error vs. iteration

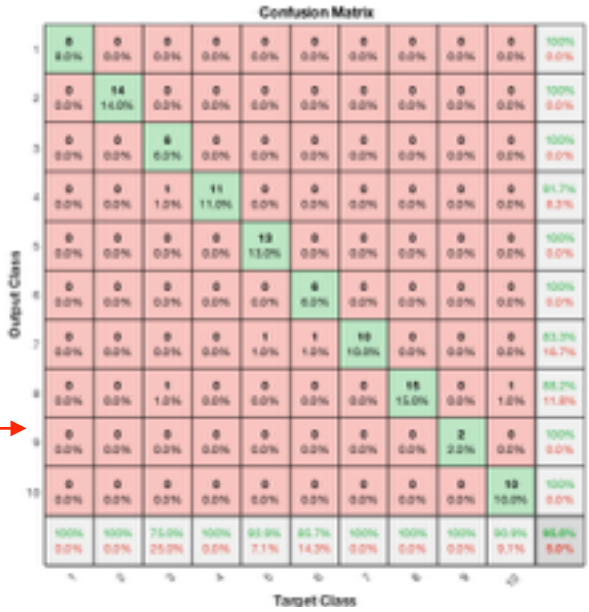


(4) 100 testing patterns with predictions and targets

the patterns with wrong predictions are highlighted

target class and output class i = the class of digit i-1

(5)confusion matrix



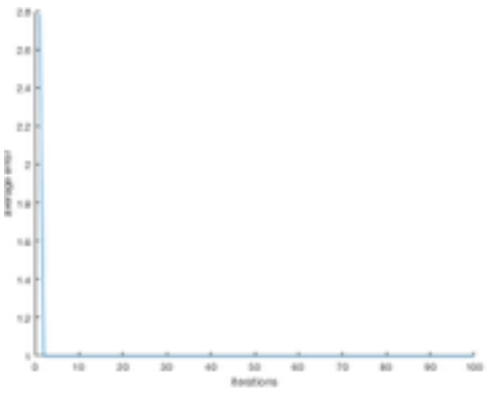
(b) activation function: ReLu function

(1) parameters: number of hidden layers: 1
number of hidden nodes: 250
learning rate parameter: 0.01
stop criterion: average error < 0.1 or iterations > 100

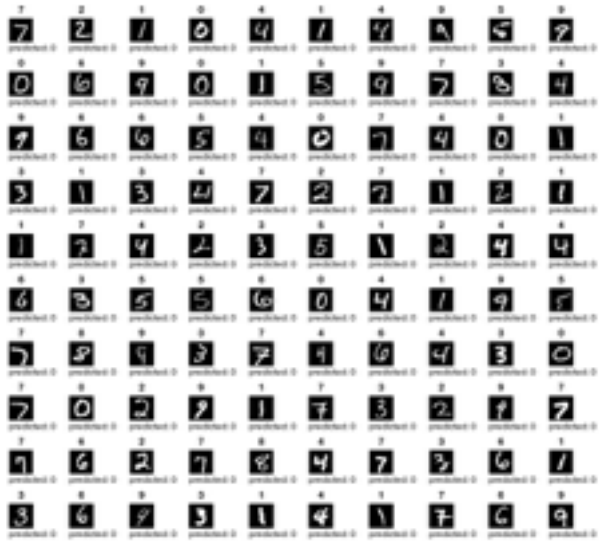
(2) CPU time in learning

```
Elapsed time is 12412.712837 seconds.
```

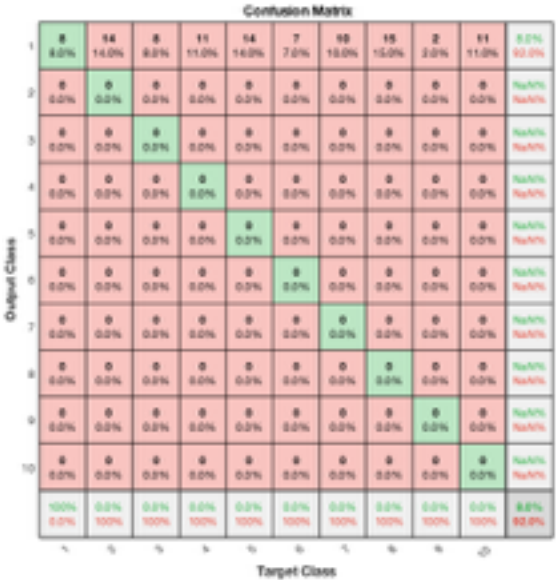
(3) average error vs. iteration



(4) 100 testing patterns with prediction and target (predictions were all 0)



(5) confusion matrix

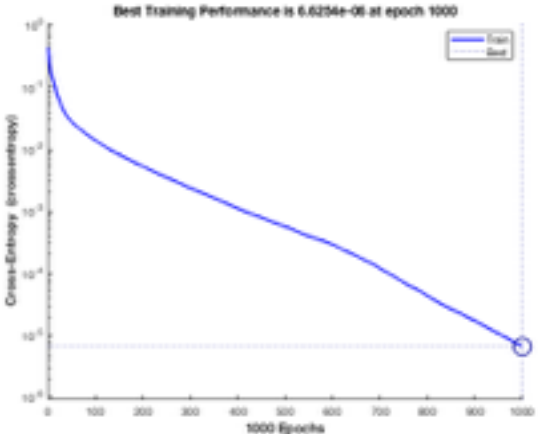


(c) MATLAB neural networks toolbox

(1) parameters: stop criterion: average error=0 or iterations>1000 or gradient< 1.00e-6
number of hidden layers: 1
number of hidden nodes: 3
learning rate parameter: 0.01

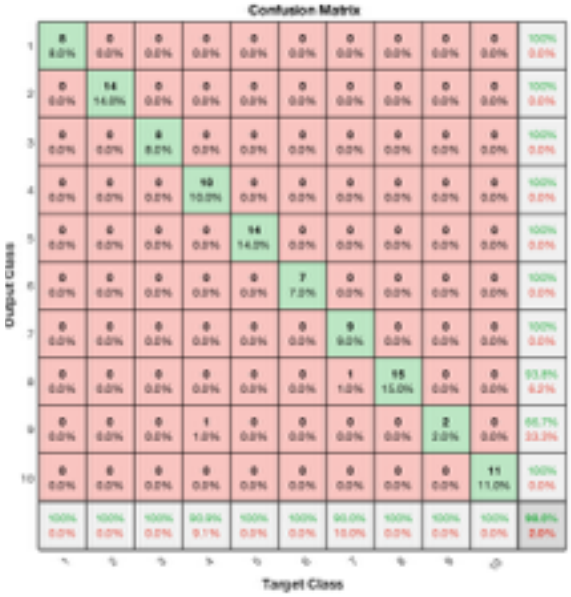
(2) CPU time in learning

(3) cross entropy vs. iteration



(4) 100 testing patterns with predictions and targets

(5)confusion matrix



(d) comparison

-CPU time in learning: MATLAB neural networks toolbox was much faster.

-convergence speed: (c)>(b)>(a)

Using ReLu as activation function will result in faster convergence than using sigmoidal function, but not always converge to 0.

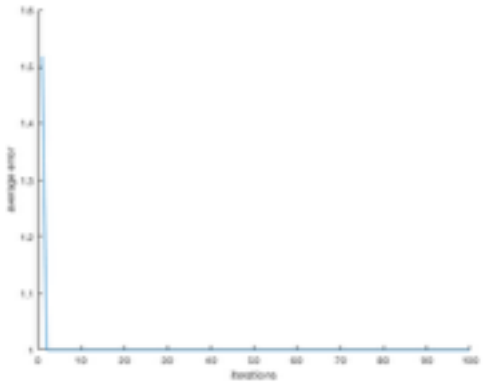
(e)first 100 training patterns

2.Discussion

(1)How to determine the hidden node number in each problem?

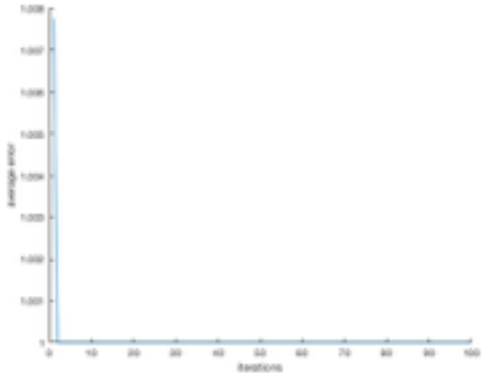
I read lots of references, but it was hard to find examples, since I decided to use only 1 hidden layer. Actually, I only find a reference using 32 hidden nodes and 1 hidden layer, which I applied and got a pretty good performance on problem a and c. However, for problem b, I tried other suggestions of hidden nodes that were used on multiple hidden layers, such as 100, 250, 300,… etc, but all got a bad performance.

(2)adding momentum terms with coefficient=0.9 to problem b



From the result, we can see that there was no improvements.The momentum term is used to prevent the system from converging to a local minimum or saddle point and increase the speed of convergence of the system. However, setting the momentum coefficient too high can create a risk of overshooting the minimum, which can cause the system to become unstable; if too low, can't reliably avoid local minima, and also can slow the training of the system.So the reason of the poor performance may due to the setting of momentum coefficient.

(3)normalize data of problem b



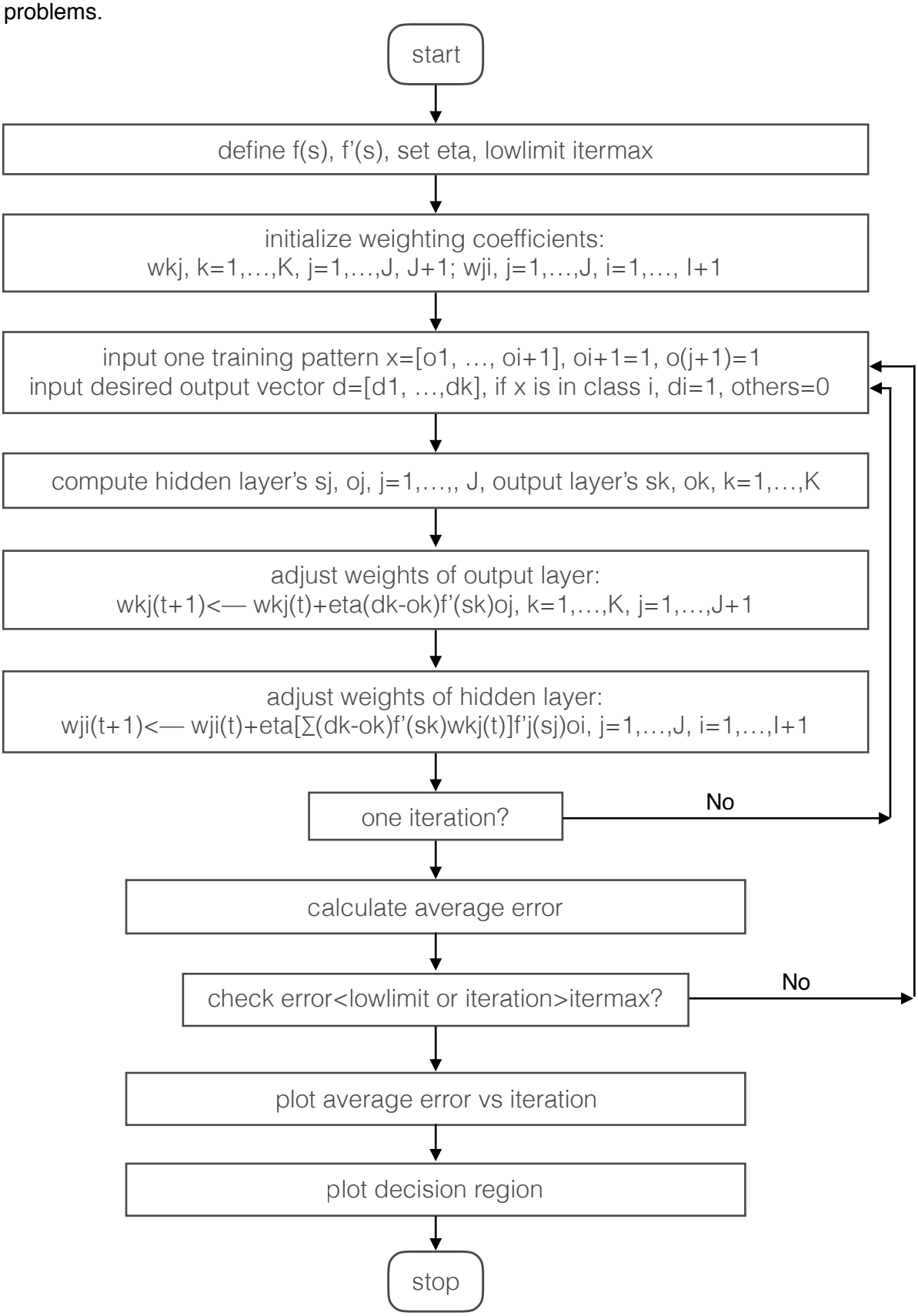
I read a reference that said there was a good performance on MNIST classification after normalizing the data; however, I still got a poor result. I think that the problem is on the activation function (Relu), not on the training data. Therefore, normalizing data didn't help.

(4)changing initial weights

Since I got improvement on project 2a by changing initial weights, I also tried this method (wkj=randn(K, J)+a, wji=randn(J-1, I)+a, a=0~5) on this project but ended up with no improvements. I think it is reasonable because the problem on project 2a was much simpler than this real data, so it may be more difficult to find the proper initial weights.

3. Flowchart of programming

My programs are mostly based on the appendix k of the reference book with some modification on input nodes, output nodes, hidden nodes, activation function and momentum term for different problems.



4. Reference

https://www.mathworks.com/solutions/deep-learning/examples/training-a-model-from-scratch.html
https://medium.com/tebs-lab/how-to-classify-mnist-digits-with-different-neural-network-architectures-39c75a0f03e3
https://datascience.stackexchange.com/questions/18667/relu-vs-sigmoid-in-mnist-example

5. Matlab programs

(a)

```
%load train data
filename1_1 = '/Users/Winnie/Downloads/三上/圖形識別/matlabprac/train-images.idx3-ubyte';
fp1_1=fopen(filename1_1,'r');
magic = fread(fp1_1, 1, 'int32', 0, 'ieee-be');
ntrain = fread(fp1_1, 1, 'int32', 0, 'ieee-be');
rows = fread(fp1_1, 1, 'int32', 0, 'ieee-be');
cols = fread(fp1_1, 1, 'int32', 0, 'ieee-be');
train=zeros(ntrain, rows*cols);
for i = 1:ntrain
    train(i, :)=fread(fp1_1,(rows*cols), 'uint8');
end
%train label
filename1_2 = '/Users/Winnie/Downloads/三上/圖形識別/matlabprac/train-labels.idx1-ubyte';
fp1_2=fopen(filename1_2,'r');
magic = fread(fp1_2, 1, 'int32', 0, 'ieee-be');
ntrain = fread(fp1_2, 1, 'int32', 0, 'ieee-be');
train_tar=zeros(ntrain, 1);
for i = 1:ntrain
    train_tar(i, :)=fread(fp1_2,1, 'uint8');
end
%load test data
filename2_1 = '/Users/Winnie/Downloads/三上/圖形識別/matlabprac/t10k-images.idx3-ubyte';
fp2_1=fopen(filename2_1,'r');
magic = fread(fp2_1, 1, 'int32', 0, 'ieee-be');
ntest = fread(fp2_1, 1, 'int32', 0, 'ieee-be');
rows = fread(fp2_1, 1, 'int32', 0, 'ieee-be');
cols = fread(fp2_1, 1, 'int32', 0, 'ieee-be');
test=zeros(ntest, rows*cols);
for i = 1:ntest
    test(i, :)=fread(fp2_1,(rows*cols), 'uint8');
end
%test label
filename2_2 = '/Users/Winnie/Downloads/三上/圖形識別/matlabprac/t10k-labels.idx1-ubyte';
fp2_2=fopen(filename2_2,'r');
magic = fread(fp2_2, 1, 'int32', 0, 'ieee-be');
ntest = fread(fp2_2, 1, 'int32', 0, 'ieee-be');
test_tar=zeros(ntest, 1);
for i = 1:ntest
    test_tar(i, :)=fread(fp2_2,1, 'uint8');
end
%data
data=zeros(ntrain, rows*cols+10);
for i=1:ntrain
   data(i,1:rows*cols)=train(i, :);
   data(i,rows*cols+1+train_tar(i, :))=1;
end
%train net
I=rows*cols+1;  %input+const
J=32+1;  %hidden+const
K=10;    %class
n=ntrain;
```

```
%initialize
wkj=randn(K, J);
wkj_temp=zeros(size(wkj));
wji=randn(J-1, I);
old_dwkj=zeros(size(wkj));
old_dwji=zeros(size(wji));
oi=[zeros(I-1, 1);1];
sj=zeros(J-1,1);
oj=[sj;1];
sk=zeros(K,1);
ok=zeros(K,1);
dk=zeros(K,1);
lowlimit=0.05;
itermax=300;
iter=0;
%eta=0.7;beta=0.3;   %add momentum term
eta=0.01;beta=0.0;
erroravg=100;
%internal variables
deltak=zeros(1, K);
sumback=zeros(1, J-1);
deltaj=zeros(1, J-1);
tic
while (erroravg>lowlimit) && (iter<itermax)
    iter=iter+1;
    error=0;
    %forward computation
    for i=1:n
        oi=[data(i,1:rows*cols) 1]';
        dk=[data(i,rows*cols+1:rows*cols+10)]';
        for j=1:J-1
            sj(j)=wji(j, :)*oi;
            oj(j)=1/(1+exp(-sj(j)));    %sigmoidal
        end
        oj(J)=1.0;
        for k=1:K
            sk(k)=wkj(k, :)*oj;
            ok(k)=1/(1+exp(-sk(k)));
        end
        error=error+sum(abs(dk-ok));
    %backward learning
        for k=1:K
        deltak(k)=(dk(k)-ok(k))*ok(k)*(1.0-ok(k));
        end
        for j=1:J
            for k=1:K
                wkj_temp(k,j)=wkj(k,j)+eta*deltak(k)*oj(j)+beta*old_dwkj(k,j);
                old_dwkj(k, j)=eta*deltak(k)*oj(j)+beta*old_dwkj(k,j);
            end
        end
        for j=1:J-1
            sumback(j)=0.0;
            for k=1:K
                sumback(j)=sumback(j)+deltak(k)*wkj(k,j);
            end
            deltaj(j)=oj(j)*(1.0-oj(j))*sumback(j);
        end
        for i=1:I
            for j=1:J-1
                wji(j,i)=wji(j,i)+eta*deltaj(j)*oi(i)+beta*old_dwji(j,i);
```

```
            old_dwji(j,i)=eta*deltaj(j)*oi(i)+beta*old_dwji(j,i);
        end
    end
    wkj=wkj_temp;
  end
  ite(iter)=iter;
  erroravg=error/n;
  error_r(iter)=erroravg;
end
toc
figure;
hold on;
plot(ite, error_r);
xlabel('iterations');
ylabel('average error');
figure;
hold on;
%test
data_=zeros(100, rows*cols+10);
test_oks=zeros(10, 100);
test_dks=zeros(10, 100);
for i=1:100
  data_(i,1:rows*cols)=test(i, :);
  data_(i,rows*cols+1+test_tar(i, :))=1;
  test_dks(test_tar(i, 1)+1,i)=1;
end
predict=zeros(100, 1);
for i=1:100
  oi=[data_(i,1:rows*cols) 1]';
  dk=[data_(i,rows*cols+1:rows*cols+10)]';
  for j=1:J-1
    sj(j)=wji(j, :)*oi;
    oj(j)=1/(1+exp(-sj(j)));    %sigmoidal
  end
  oj(J)=1.0;
  for k=1:K
    sk(k)=wkj(k, :)*oj;
    ok(k)=1/(1+exp(-sk(k)));
  end
  [m,index]=max(ok);
  test_oks(index, i)=1;
  predict(i,:)=index-1;
  subplot(10, 10, i);
  temp=test(i,:);
  temp=reshape(temp, rows, cols)';
  imshow(temp);
  title([test_tar(i, :)]);
  text(0, 35, sprintf('predicted: %d',predict(i,:)));
end
figure;
plotconfusion(test_dks,test_oks);
```

(b)

```
%load train data
filename1_1 = '/Users/Winnie/Downloads/三上/圖形識別/matlabprac/train-images.idx3-ubyte';
fp1_1=fopen(filename1_1,'r');
magic = fread(fp1_1, 1, 'int32', 0, 'ieee-be');
ntrain = fread(fp1_1, 1, 'int32', 0, 'ieee-be');
rows = fread(fp1_1, 1, 'int32', 0, 'ieee-be');
cols = fread(fp1_1, 1, 'int32', 0, 'ieee-be');
```

```
train=zeros(ntrain, rows*cols);
for i = 1:ntrain
    train(i, :)=fread(fp1_1,(rows*cols), 'uint8');
end

%train label
filename1_2 = '/Users/Winnie/Downloads/三上/圖形識別/matlabprac/train-labels.idx1-ubyte';
fp1_2=fopen(filename1_2,'r');
magic = fread(fp1_2, 1, 'int32', 0, 'ieee-be');
ntrain = fread(fp1_2, 1, 'int32', 0, 'ieee-be');
train_tar=zeros(ntrain, 1);
for i = 1:ntrain
    train_tar(i, :)=fread(fp1_2,1, 'uint8');
end
%load test data
filename2_1 = '/Users/Winnie/Downloads/三上/圖形識別/matlabprac/t10k-images.idx3-ubyte';
fp2_1=fopen(filename2_1,'r');
magic = fread(fp2_1, 1, 'int32', 0, 'ieee-be');
ntest = fread(fp2_1, 1, 'int32', 0, 'ieee-be');
rows = fread(fp2_1, 1, 'int32', 0, 'ieee-be');
cols = fread(fp2_1, 1, 'int32', 0, 'ieee-be');
test=zeros(ntest, rows*cols);
for i = 1:ntest
    test(i, :)=fread(fp2_1,(rows*cols), 'uint8');
end
%test label
filename2_2 = '/Users/Winnie/Downloads/三上/圖形識別/matlabprac/t10k-labels.idx1-ubyte';
fp2_2=fopen(filename2_2,'r');
magic = fread(fp2_2, 1, 'int32', 0, 'ieee-be');
ntest = fread(fp2_2, 1, 'int32', 0, 'ieee-be');
test_tar=zeros(ntest, 1);
for i = 1:ntest
    test_tar(i, :)=fread(fp2_2,1, 'uint8');
end
%data
%train=normalize(train,'range');  %normalize data
data=zeros(ntrain, rows*cols+10);
for i=1:ntrain
   data(i,1:rows*cols)=train(i, :);
   data(i,rows*cols+1+train_tar(i, :))=1;
end
%train net
I=rows*cols+1;  %input+const
J=250+1;  %hidden+const
K=10;    %class
n=ntrain;
%initialize
wkj=randn(K, J);
wkj_temp=zeros(size(wkj));
wji=randn(J-1, I);
old_dwkj=zeros(size(wkj));
old_dwji=zeros(size(wji));
oi=[zeros(I-1, 1);1];
sj=zeros(J-1,1);
oj=[sj;1];
sk=zeros(K,1);
ok=zeros(K,1);
dk=zeros(K,1);
lowlimit=0.1;
itermax=100;
```

```
iter=0;
%eta=0.1;beta=0.9;   %add momentum term
eta=0.01;beta=0.0;
erroravg=100;
besterror=100;
%internal variables
deltak=zeros(1, K);
sumback=zeros(1, J-1);
deltaj=zeros(1, J-1);
tic
while (erroravg>lowlimit) && (iter<itermax)
    iter=iter+1;
    error=0;
    %forward computation
    for i=1:n
        oi=[data(i,1:rows*cols) 1]';
        dk=[data(i,rows*cols+1:rows*cols+10)]';
        for j=1:J-1
            sj(j)=wji(j, :)*oi;
            oj(j)=max(0, sj(j));      %ReLu
        end
        oj(J)=1.0;
        for k=1:K
            sk(k)=wkj(k, :)*oj;
            ok(k)=max(0, sk(k));
        end
        error=error+sum(abs(dk-ok));
    %backward learning
        for k=1:K
          deltak(k)=(dk(k)-ok(k))*(sk(k)>0);   %ok'(k)=0, 1;
        end
        for j=1:J
            for k=1:K
                wkj_temp(k,j)=wkj(k,j)+eta*deltak(k)*oj(j);
            end
        end
        for j=1:J-1
            sumback(j)=0.0;
            for k=1:K
                sumback(j)=sumback(j)+deltak(k)*wkj(k,j);
            end
            deltaj(j)=sumback(j)*(sj(j)>0);   %oj'(j)=0,1
        end
        for i=1:I
            for j=1:J-1
                wji(j,i)=wji(j,i)+eta*deltaj(j)*oi(i);
            end
        end
        wkj=wkj_temp;
    end
    ite(iter)=iter;
    erroravg=error/n;
    error_r(iter)=erroravg;
end
toc
 figure;
 hold on;
 plot(ite, error_r);
 xlabel('iterations');
 ylabel('average error');
```

```
figure;
hold on;
%test
%test=normalize(test,'range');
data_=zeros(100, rows*cols+10);
test_oks=zeros(10, 100);
test_dks=zeros(10, 100);
for i=1:100
    data_(i,1:rows*cols)=test(i, :);
    data_(i,rows*cols+1+test_tar(i, :))=1;
    test_dks(test_tar(i, 1)+1,i)=1;
end
predict=zeros(100, 1);
for i=1:100
    oi=[data_(i,1:rows*cols) 1]';
    dk=[data_(i,rows*cols+1:rows*cols+10)]';
    for j=1:J-1
        sj(j)=wji(j, :)*oi;
        oj(j)=max(0, sj(j));      %ReLu
    end
    oj(J)=1.0;
    for k=1:K
        sk(k)=wkj(k, :)*oj;
        ok(k)=max(0, sk(k));      %ReLu
    end
    [m,index]=max(ok);
    test_oks(index, i)=1;
    predict(i,:)=index-1;
    subplot(10, 10, i);
    temp=test(i,:);
    temp=reshape(temp, rows, cols)';
    imshow(temp);
    title([test_tar(i, :)]);
    text(0, 35, sprintf('predicted: %d',predict(i,:)));
end
figure;
plotconfusion(test_dks,test_oks);
```

(c)

```
%load train data
filename1_1 = '/Users/Winnie/Downloads/三上/圖形識別/matlabprac/train-images.idx3-ubyte';
fp1_1=fopen(filename1_1,'r');
magic = fread(fp1_1, 1, 'int32', 0, 'ieee-be');
ntrain = fread(fp1_1, 1, 'int32', 0, 'ieee-be');
rows = fread(fp1_1, 1, 'int32', 0, 'ieee-be');
cols = fread(fp1_1, 1, 'int32', 0, 'ieee-be');
train_data=zeros(ntrain, rows*cols);
for i = 1:ntrain
    train_data(i, :)=fread(fp1_1,(rows*cols), 'uint8');
end
%train label
filename1_2 = '/Users/Winnie/Downloads/三上/圖形識別/matlabprac/train-labels.idx1-ubyte';
fp1_2=fopen(filename1_2,'r');
magic = fread(fp1_2, 1, 'int32', 0, 'ieee-be');
ntrain = fread(fp1_2, 1, 'int32', 0, 'ieee-be');
train_tar=zeros(ntrain, 1);
for i = 1:ntrain
    train_tar(i, :)=fread(fp1_2,1, 'uint8');
end
%load test data
```

```
filename2_1 = '/Users/Winnie/Downloads/三上/圖形識別/matlabprac/t10k-images.idx3-ubyte';
fp2_1=fopen(filename2_1,'r');
magic = fread(fp2_1, 1, 'int32', 0, 'ieee-be');
ntest = fread(fp2_1, 1, 'int32', 0, 'ieee-be');
rows = fread(fp2_1, 1, 'int32', 0, 'ieee-be');
cols = fread(fp2_1, 1, 'int32', 0, 'ieee-be');
test=zeros(ntest, rows*cols);
for i = 1:ntest
    test(i, :)=fread(fp2_1,(rows*cols), 'uint8');
end
%test label
filename2_2 = '/Users/Winnie/Downloads/三上/圖形識別/matlabprac/t10k-labels.idx1-ubyte';
fp2_2=fopen(filename2_2,'r');
magic = fread(fp2_2, 1, 'int32', 0, 'ieee-be');
ntest = fread(fp2_2, 1, 'int32', 0, 'ieee-be');
test_tar=zeros(ntest, 1);
for i = 1:ntest
    test_tar(i, :)=fread(fp2_2,1, 'uint8');
end
trainFcn = 'trainscg';  % Scaled conjugate gradient backpropagation.
hiddenLayerSize = 32;
net = patternnet(hiddenLayerSize, trainFcn);
net.trainParam.lr=0.01;
net.divideParam.trainRatio = 100/100;
net.divideParam.valRatio = 0/100;
net.divideParam.testRatio = 0/100;
tar=zeros(ntrain, 10);
for i=1:ntrain
   tar(i,1+train_tar(i, :))=1;
end
[net,tr] = train(net, train_data', tar');
view(net);
figure, plotperform(tr);
figure;
hold on;
target=zeros(100, 10);
for i=1:100
   target(i,1+test_tar(i, :))=1;
end
for i = 1:10
   for j=1:10
     subplot(10, 10, (i-1)*10+j);
     img=reshape(test((i-1)*10+j,:), rows, cols)';
     actualLabel = test_tar((i-1)*10+j, :);
     pred((i-1)*10+j,:)=sim(net,test((i-1)*10+j,:)');
     [a, index]=max(pred((i-1)*10+j,:));
     index=index-1;
     imshow(img);
     title(actualLabel);
     text(0, 35, sprintf('predicted: %d',index));
   end
end
figure;
plotconfusion(target', pred');
```