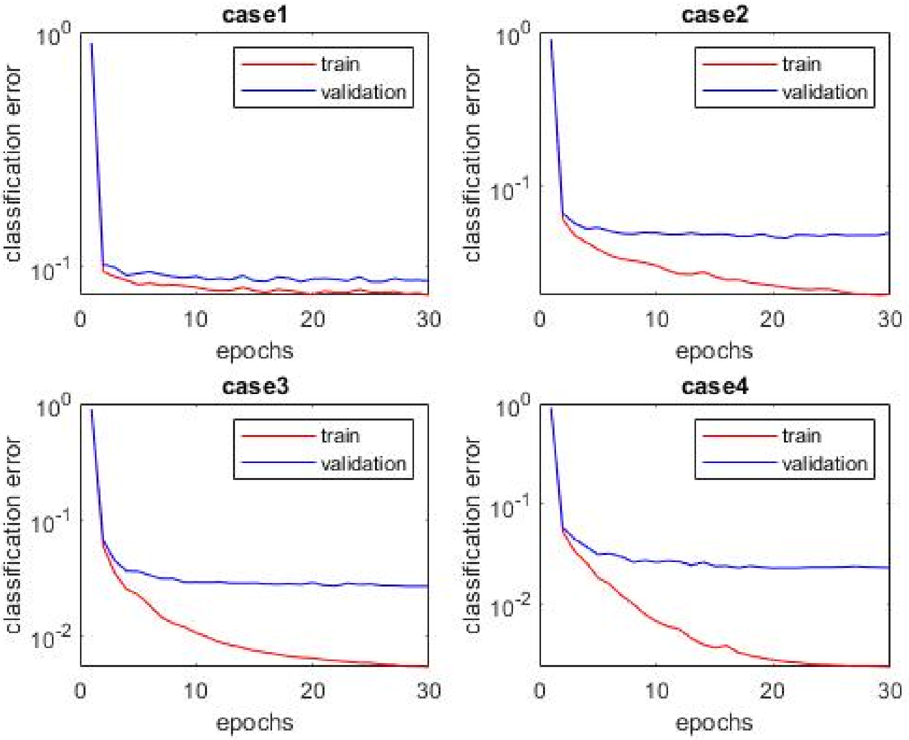
Neural Network Home assignment 3

Min Cheng

October 2018



Network	Epoch number	Train error	Lowest validation error	Test error
case1	19	0.0779	0.0885	0.0828
case2	21	0.02226	0.0462	0.0481
case3	29	0.00538	0.0264	0.0247
case4	20	0.00274	0.0225	0.0238

Table 1: Statistic of the lowest Validation error for each network

Discussion

On the basis of the result we can confer that the more number of layer and neuron in neural network, the better accuracy of prediction can yield. Since the first case Network1 has no hidden layer therefor it has the highest error around 10 percent. When increase the complexity of the network with one hidden layer. The result gets better but also will get some overfitting. The best accuracy is the Network4 with only 2.25 percent Test accuracy error. however it has the largest overfitting as well. Test error is almost 20 times larget than the Training error. Compare the Case3 and Case4, since with one more layer and twice number neuron, the result increase only with 0.09 percent which so little differ can be ignored. So is this worth with so much computation and time consuming to get only this little improvement? I don't think so.

Network	Epoch number	Train error	validation error	Test error
Network1	55	0.00018	0.0266	0.0208
Network2	70	0.00096	0.0275	0.0267
Network3	200	0.0011	0.0212	0.0209

Table 2: Classification errors of all data sets obtained for Networks 1-3

Discussion

Network 2 has one more hidden layers of 100 neurons than Network1 but get less accuracy than Network1 , this means Network2 has overfitting because of the complexity of the network is more complicated. Network 3 uses the L2-regularization weight decay method to prevent the overfitting. Although it has the relative largest training error, but this seems that it does not matter att all. Network3 gets the best accuracy in this three Model, Thus in order to get the best accuracy, besides the learningrate, regularization or some other factor, the complexity of the network should be considered as a import factor as well.

Network	Epoch number	Train error	validation error	Test error
Network1	60	0.0153	0.0207	0.0191
Network2	30	0.0091	0.0163	0.0107

Table 3: Classification errors of all data sets obtained for Networks 1-2

Discussion

Compare with homework 3.3 for FC layer, Since the CNN layer use receptive fields sliding or convolving around the input, thus element from the feature map has the number of the weights which equals to the filters element. This will decrease much more number of weights than the Fully connected layer.The pooling approach will further decrease the number of weight. This series of decreasing number of weight will efficiently prevent the overfitting.

The network 2 is a deeper CNN network. With convolution method the first layer can recognize basic things like edges. The second layer can train itself to recognize the collection of edges and so on, thus the more high level layer the more advance feature it can recognize, which means it has the better classification. Therefore the Network 2 has better accuracy since it has more CNN layer.

```
clc
clear all
% [xTrain, tTrain, xValid, tValid, xTest, tTest] = LoadMNIST(1);
% save('xTrain.mat')
% save('tTrain.mat')
% save('xValid.mat')
% save('tValid.mat')
% save('xTest.mat')
% save('tTest.mat')
load('xTrain.mat')
load('tTrain.mat')
load('xValid.mat')
load('tValid.mat')
load('xTest.mat')
load('tTest.mat')
lr=0.003;
mb=10;
mu=size(xTrain,2);
xtrain=zeros(size(xTrain,1),1);
xval_1=zeros(size(xValid,1),1);
xval=zeros(size(xValid));
xtest_1=zeros(size(xTest,1),1);
xtest=zeros(size(xValid));
x_shuffle=zeros(size(xTrain));
t_shuffle=zeros(size(tTrain));
epoch=30;
error_train=zeros(epoch,4);
error_val=zeros(epoch,4);
error_test=zeros(epoch,4);
for i=1:size(xTrain,2)
    xtrain=xtrain+xTrain(:,i);
end
mean=xtrain/size(xTrain,2);
for i=1:size(xTrain,2)
    xTrain(:,i)=xTrain(:,i)-mean;
end
for i=1:size(xValid,2)
     xval_1=xval_1+xValid(:,i);
end
mean=xval_1/size(xValid,2);
for i=1:size(xValid,2)
    xval(:,i)=xValid(:,i)-mean;
end

for i=1:size(xTest,2)
    xtest_1=xtest_1+xTest(:,i);
end
mean=xtest_1/size(xTest,2);
for i=1:size(xTest,2)
    xtest(:,i)=xTest(:,i)-mean;
end
```

```
for bigloop=1:4
    switch bigloop
        case 1
            layer_size=2;
            w =cell(layer_size-1,1);
            b =cell(layer_size-1,1);
            v =cell(layer_size,1);
            w1=normrnd(0,1/sqrt(784),10,784);
            theta1=zeros(10,1);
            theta={theta1};
            w={w1};
            w_old={w1};
            delta=cell(layer_size-1,1);
            delta_theta=cell(layer_size-1,1);
            delta_w=cell(layer_size-1,1);
        case 2
            layer_size=3;
            w =cell(layer_size-1,1);
            b =cell(layer_size-1,1);
            v =cell(layer_size,1);
            w1=normrnd(0,1/sqrt(784),30,784);
            w2=normrnd(0,1/sqrt(30),10,30);
            w={w1 w2};
            w_old={w1 w2};
            theta1=zeros(30,1);
            theta2=zeros(10,1);
            theta={theta1 theta2};
            delta=cell(layer_size-1,1);
            delta_theta=cell(layer_size-1,1);
            delta_w=cell(layer_size-1,1);
        case 3
            layer_size=3;
            w =cell(layer_size-1,1);
            b =cell(layer_size-1,1);
            v =cell(layer_size,1);
            w1=normrnd(0,1/sqrt(784),100,784);
            w2=normrnd(0,1/sqrt(100),10,100);
            w={w1 w2};
            w_old={w1 w2};
            theta1=zeros(100,1);
            theta2=zeros(10,1);
            theta={theta1 theta2};
            delta=cell(layer_size-1,1);
            delta_theta=cell(layer_size-1,1);
            delta_w=cell(layer_size-1,1);
        case 4
            layer_size=4;
            w =cell(layer_size-1,1);
            b =cell(layer_size-1,1);
            v =cell(layer_size,1);
            w1=normrnd(0,1/sqrt(784),100,784);
            w2=normrnd(0,1/sqrt(100),100,100);
            w3=normrnd(0,1/sqrt(100),10,100);
            w={w1 w2 w3};
            w_old={w1 w2 w3};
            theta1=zeros(100,1);
            theta2=zeros(100,1);
            theta3=zeros(10,1);
            theta={theta1 theta2 theta3};
            delta=cell(layer_size-1,1);
            delta_theta=cell(layer_size-1,1);
            delta_w=cell(layer_size-1,1);
    end
    %%%%%%%%%%%%%%%%%
    for k=1:epoch
        start=1;
        shuffle = randperm(mu);
        for L = 1:mu
            x_shuffle(:,L) = xTrain(:,shuffle(L));
            t_shuffle(:,L) = tTrain(:,shuffle(L));
        end
        %%%%%%%%%%%%%%%%%
        for z1=1:size(x_shuffle,2)
            v{1}=x_shuffle(:,z1);
            for n1=1:size(w,2)
                b{n1}= w{n1}*v{n1}-theta{n1};
                v{n1+1}=1./(1+exp(-b{n1}));
            end
            pred=find(v{end}==max(v{end}));
            target=find(t_shuffle(:,z1)==max(t_shuffle(:,z1)));
            if pred~=target
               error_train(k,bigloop)=error_train(k,bigloop)+1;
            end
        end
        %%%%%%%%%%%%%%%%%%
        for z2=1:size(xval,2)
            v{1}=xval(:,z2);
            for n=1:size(w,2)
                b{n}= w{n}*v{n}-theta{n};
                v{n+1}=1./(1+exp(-b{n}));
            end
            pred=find(v{end}==max(v{end}));
            target=find(tValid(:,z2)==max(tValid(:,z2)));
            if pred~=target
               error_val(k,bigloop)=error_val(k,bigloop)+1;
            end
        end
        for z3=1:size(xtest,2)
            v{1}=xtest(:,z3);
            for n=1:size(w,2)
                b{n}= w{n}*v{n}-theta{n};
                v{n+1}=1./(1+exp(-b{n}));
            end
            pred=find(v{end}==max(v{end}));
            target=find(tTest(:,z3)==max(tTest(:,z3)));
            if pred~=target
               error_test(k,bigloop)=error_test(k,bigloop)+1;
            end
        end
        error_train(k,bigloop)=error_train(k,bigloop)/size(xTrain,2);
        error_val(k,bigloop)=error_val(k,bigloop)/size(xval,2);
        error_test(k,bigloop)=error_test(k,bigloop)/size(xtest,2);
        %%%%%%%%%%%%%%%%%%%%%
        if k>1
            if error_val(k-1,bigloop)>error_val(k,bigloop)
               w=w_old;
            end
        end
        for j=1:size(xTrain,2)/mb
            for i2=1:size(delta,1)
                delta{i2}=[];
                delta_theta{i2}=[];
                delta_w{i2}=[];
            end

            for i=start:start+mb-1
                v{1}=x_shuffle(:,i);
                for n=1:size(w,2)
                    b{n}= w{n}*v{n}-theta{n};
                    v{n+1}=1./(1+exp(-b{n}));
                end
                for h=layer_size-1:-1:1
                    if h==layer_size-1
                        g_prim=(exp(-b{h}))./((1+exp(-b{h})).^2); % ds
                        delta{h}=(t_shuffle(:,i)-v{end}).*g_prim;
                    else
                        g_prim=(exp(-b{h}))./((1+exp(-b{h})).^2);
                        delta{h}=((delta{h+1}'*w{h+1}))'.*g_prim;
                    end
                end
                for H=1:size(delta,1)
                    if i==start
                       delta_theta{H}=zeros(size(delta(H)));
                       delta_w{H}=zeros(size(delta{H}*v{H}'));
                    end
                    delta_theta{H}=delta_theta{H}+delta{H};
                    delta_w{H}=delta_w{H}+delta{H}*v{H}';
                end
            end
            for g=1:size(delta_w,1)
                w{g}=w{g}+lr*delta_w{g};          %%%%%%%%%%%%%%
                theta{g}=theta{g}-lr*delta_theta{g};
            end
            start=start+mb;
        end
        w_old=w;
    end
    subplot(2,2,bigloop)
    if bigloop==1
    semilogy([1:1:30]',error_train(:,bigloop),'r',
[1:1:30]',error_val(:,bigloop),'b'),axis([0 30 0 1]),title('case1')
    xlabel('epochs');
    ylabel('classification error');
    legend('train','validation');
    elseif bigloop==2
    semilogy([1:1:30]',error_train(:,bigloop),'r',
[1:1:30]',error_val(:,bigloop),'b'),axis([0 30 0 1]),title('case2')
    xlabel('epochs');
    ylabel('classification error');
    legend('train','validation');
    elseif bigloop==3
    semilogy([1:1:30]',error_train(:,bigloop),'r',
[1:1:30]',error_val(:,bigloop),'b'),axis([0 30 0 1]),title('case3')
    xlabel('epochs');
    ylabel('classification error');
    legend('train','validation');
    elseif bigloop==4
    semilogy([1:1:30]',error_train(:,bigloop),'r',
[1:1:30]',error_val(:,bigloop),'b'),axis([0 30 0 1]),title('case4')
    xlabel('epochs');
    ylabel('classification error');
    legend('train','validation');
    end
end
```

table

```
target=find(t_shuffle(:,z1)==max(t_shuffle(:,z1)));
title ={'case1','case2','case3','case4'};
best_epoch=zeros(4,1);
best_validation=zeros(4,1);
best_test=zeros(4,1);
best_train=zeros(4,1);
for i=1:4
    best_epoch(i,1)=find(error_val(:,i)==min(error_val(:,i)),1);
    best_validation(i,1)=error_val(best_epoch(i,1),i);
    best_test(i,1)=error_test(best_epoch(i,1),i);
    best_train(i,1)=error_train(best_epoch(i,1),i);
end
T=table(title',best_epoch,best_train,best_validation,best_test)

clc
clear all
% [xTrain3, tTrain3, xValid3, tValid3, xTest3, tTest3] = LoadMNIST(3);
% save('xTrain3.mat')
% save('tTrain3.mat')
% save('xValid3.mat')
% save('tValid3.mat')
% save('xTest3.mat')
% save('tTest3.mat')
load('xTrain3.mat')
load('tTrain3.mat')
load('xValid3.mat')
load('tValid3.mat')
load('xTest3.mat')
load('tTest3.mat')

layers1 = [
    imageInputLayer([28 28 1])
    fullyConnectedLayer(100)
    reluLayer
    fullyConnectedLayer(100)
    reluLayer
    fullyConnectedLayer(10)
    softmaxLayer
    classificationLayer];

opts = trainingOptions('sgdm', ...
    'Momentum',0.9,...
    'InitialLearnRate',0.01,...
    'MaxEpochs',200, ...
    'MiniBatchSize',8192,...
    'ValidationPatience',5,...
    'ValidationFrequency',30,...
    'Shuffle', 'every-epoch',...
    'Plots','training-progress', ...
    'Verbose',false, ...
    'ExecutionEnvironment','gpu',...
    'ValidationData',{xValid3,tValid3});
net = trainNetwork(xTrain3,tTrain3,layers1,opts);

error1_test=0;
result=net.classify(xTest3);
for z1=1:size(result,1)
      if tTest3(z1,1)~=result(z1,1)
         error1_test=error1_test+1;
      end
end
error1_test = error1_test/size(result,1)

error1_train=0;
result=net.classify(xTrain3);
for z1=1:size(result,1)
      if tTrain3(z1,1)~=result(z1,1)
         error1_train=error1_train+1;
      end
end
error1_train = error1_train/size(result,1)

layers2 = [
    imageInputLayer([28 28 1])
    fullyConnectedLayer(100)
    reluLayer
    fullyConnectedLayer(100)
    reluLayer
    fullyConnectedLayer(100)
    reluLayer
    fullyConnectedLayer(10)
    softmaxLayer
    classificationLayer];
```

```
opts = trainingOptions('sgdm', ...
    'Momentum',0.9,...
    'InitialLearnRate',0.01,...
    'MaxEpochs',200, ...
    'MiniBatchSize',8192,...
    'ValidationPatience',5,...
    'ValidationFrequency',30,...
    'Shuffle', 'every-epoch',...
    'Plots','training-progress', ...
    'Verbose',false, ...
    'ExecutionEnvironment','gpu',...
    'ValidationData',{xValid3,tValid3});
net = trainNetwork(xTrain3,tTrain3,layers2,opts);

error2_test=0;
result=net.classify(xTest3);
for z1=1:size(result,1)
      if tTest3(z1,1)~=result(z1,1)96
         error2_test=error2_test+1;
      end
end
error2_test= error2_test/size(result,1)

error2_train=0;
result=net.classify(xTrain3);
for z1=1:size(result,1)
      if tTrain3(z1,1)~=result(z1,1)
         error2_train=error2_train+1;
      end
end
error2_train= error2_train/size(result,1)

opts = trainingOptions('sgdm', ...
    'Momentum',0.9,...
    'InitialLearnRate',0.01,...
    'MaxEpochs',200, ...
    'MiniBatchSize',8192,...
    'ValidationPatience',5,...
    'ValidationFrequency',30,...
    'Shuffle', 'every-epoch',...
    'Plots','training-progress', ...
    'Verbose',false, ...
    'L2Regularization',0.03,...
    'ExecutionEnvironment','gpu',...
    'ValidationData',{xValid3,tValid3});
net = trainNetwork(xTrain3,tTrain3,layers1,opts);

error3_test=0;
result=net.classify(xTest3);
for z1=1:size(result,1)
      if tTest3(z1,1)~=result(z1,1)
         error3_test=error3_test+1;
      end
end
error3_test= error3_test/size(result,1)
```

```
error3_train=0;
result=net.classify(xTrain3);
for z1=1:size(result,1)
      if tTrain3(z1,1)~=result(z1,1)
         error3_train=error3_train+1;
      end
end
error3_train= error3_train/size(result,1)

   clc
clear all
% [xTrain4, tTrain4, xValid4, tValid4, xTest4, tTest4] = LoadMNIST(4);
% save('xTrain4.mat')
% save('tTrain4.mat')
% save('xValid4.mat')
% save('tValid4.mat')
% save('xTest4.mat')
% save('tTest4.mat')
load('xTrain4.mat')
load('tTrain4.mat')
load('xValid4.mat')
load('tValid4.mat')
load('xTest4.mat')
load('tTest4.mat')

layers1 = [
    imageInputLayer([28 28 1])
    convolution2dLayer(5,20,'Stride',1,'Padding',1)
    reluLayer
    maxPooling2dLayer(2,'Stride',2,'Padding',0)
    fullyConnectedLayer(100)
    reluLayer
    fullyConnectedLayer(10)
    softmaxLayer
    classificationLayer];

opts = trainingOptions('sgdm', ...
    'Momentum',0.9,...
    'InitialLearnRate',0.001,...
    'MaxEpochs',60, ...
    'MiniBatchSize',8192,...
    'ValidationPatience',5,...
    'ValidationFrequency',30,...
    'Shuffle', 'every-epoch',...
    'Plots','training-progress', ...
    'Verbose',false, ...
    'ExecutionEnvironment','gpu',...
    'ValidationData',{xValid4,tValid4});
net = trainNetwork(xTrain4,tTrain4,layers1,opts);

error1_test=0;
result=net.classify(xTest4);
for z1=1:size(result,1)
```

```
      if tTest4(z1,1)~=result(z1,1)
         error1_test=error1_test+1;
      end
end
error1_test= error1_test/size(result,1)

error1_train=0;
result=net.classify(xTrain4);
for z1=1:size(result,1)
      if tTrain4(z1,1)~=result(z1,1)
         error1_train=error1_train+1;
      end
end
error1_train= error1_train/size(result,1)

layers2 = [
    imageInputLayer([28 28 1])

    convolution2dLayer(3,20,'Stride',1,'Padding',1)
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,30,'Stride',1,'Padding',1)
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,50,'Stride',1,'Padding',1)
    batchNormalizationLayer
    reluLayer

    fullyConnectedLayer(10)
    softmaxLayer
    classificationLayer];

opts = trainingOptions('sgdm', ...
    'Momentum',0.9,...
    'InitialLearnRate',0.01,...
    'MaxEpochs',30, ...
    'MiniBatchSize',8192,...
    'ValidationPatience',5,...
    'ValidationFrequency',30,...
    'Shuffle', 'every-epoch',...
    'Plots','training-progress', ...
    'Verbose',false, ...
    'ExecutionEnvironment','gpu',...
    'ValidationData',{xValid4,tValid4});
net = trainNetwork(xTrain4,tTrain4,layers2,opts);

error2_test=0;
result=net.classify(xTest4);
```

```
for z1=1:size(result,1)
      if tTest4(z1,1)~=result(z1,1)
         error2_test=error2_test+1;
      end
end
error2_test= error2_test/size(result,1)


error2_train=0;
result=net.classify(xTrain4);
for z1=1:size(result,1)
      if tTrain4(z1,1)~=result(z1,1)
         error2_train=error2_train+1;
      end
end
error2_train= error2_train/size(result,1)
```

Published with MATLAB® R2018a