FFR135, Artificial Neural Networks
Home Problem 3

October 24, 2019

Ella Guiladi
930509-0822
guiladi@student.chalmers.se

3.3 ReLU, softmax, early stopping

Table (1) and (2) displays the classification errors and number of training epochs for each of the three networks.

Table 1: Classification errors for each network

Data set	Network 1	Network 2	Network 3
Training set	0.4335	0.4405	0.3981
Validation set	0.5173	0.5244	0.4976
Test set	0.5132	0.5217	0.4962

Table 2: Number of training epochs and iterations for each network

Networks	Number of epochs
Network 1	195
Network 2	165
Network 3	353

It is clear that Network 3 is the best in terms of accuracy from table (1), since it has the lowest classification error. However, it is the network with highest number of epochs, meaning that it took the longest time to run. The network that performs worst is network 2 since it has the highest classification error, even tough it has the lowest number of epochs.

The reason for why network 3 outperforms network 1 and 2 is due to its architecture. Since the networks are deep networks, they will encounter problems with overfitting due to having more neurons. Overfitting can be reduced by using regularisation schemes, such as the L_2-regularisation that is introduced to network 3 with its L_2-regularisation parameter. By reducing overfitting network 3 will be able to learn faster, leading to a lower classification error.

The reason for why network 1 outperforms network 2 is due to having fewer neurons in the hidden layers, since network 1 have one less hidden layer with 50 neurons than network 2. This results in network 2 having more problems with overfitting and a lower accuracy, i.e higher classification error. Another explanation for why network 2 performs less accurate is due to unstable gradients. The ReLU function makes the vanishing gradient problem less severe since the gradient of the activation function does not decrease exponentially, as in the case for the sigmoid function. However the ReLU function does not saturate, thereby resulting in an increase of the weights which slows down the learning of the network. In order for the weights not to grow it is necessary to implement a regularisation scheme such as in network 3.

It is also noted that from table (2) that early stopping occurred for all networks, since the maximum number of epochs were set to 400.

3.4 Convolutional networks

Table (3) and (4) presents the classification errors and number of training epochs for each of the two networks.

Table 3: Classification errors for each network

Data set	Network 1	Network 2
Training set	0.3271	0.2537
Validation set	0.3892	0.2953
Test set	0.3855	0.2862

Table 4: Number of training epochs for each networks

Networks	Number of epochs
Network 1	120
Network 2	120

From table (3) it is clear that network 2 performs best, due to having lower classification error than network 1. Since network 1 has one more fully connected hidden layer with 50 neurons than network 2, network 1 might experience problems with overfitting due to having more neurons. Overfitting the data will in turn decrease the learning of the network and in turn result in a higher classification error. Network 2 also contains two more convolutional layers than network 1, which is beneficial since convolutional networks has fewer neurons than fully connected networks. This decreases the risk of overfitting the data.

Another difference between the two networks is that network 2 has one more Max-pooling layer than network 1. The Max-pooling layer simplifies the outputs of the convolution layers by summarising the outputs of close feature maps, by taking the maximum over the feature maps outputs. The Max-pooling layers thereby reduce the dimensionality and the of amount of parameters and computational work in the network, which increases the performance.

Network 2 also contains batch-normalisation layers, in contrast to network 1. In fully connected networks the batch-normalisation layers have a role in reducing the unstable gradient problem, by preventing local fields of hidden neurons to grow. However, it is unclear to which extent the batch-normalisation layers reduce the unstable gradient in the the convolutional networks with the ReLU function. From the lectures in the Artificial Neural Network course, it has been taught that the batch-normalisation layers may have a reducing effect to the unstable gradient problem, even though it is not known exactly how. Thereby this results in yet another reason to why network 2 performs better than network 1.

Finally it has been noted from table (4) that early stopping did not occur for any of networks, since the maximal number of epochs were set to 120.

A Matlab code

A.1 Code for calculating the classification error

```
function classificationError = ClassError(predictionSet,targetSet)
countOfSet=0;

for i=1:size(targetSet,1)
    if(targetSet(i)~=predictionSet(i))
        countOfSet = countOfSet+1;
    end
end
classificationError = (1/size(targetSet,1)) * countOfSet;
end
```

A.2 ReLU network 1

```
clear all
close all

[xTrain, tTrain, xValid, tValid, xTest, tTest] = LoadCIFAR(3);

layers = [imageInputLayer([32 32 3])
fullyConnectedLayer(50)
reluLayer
fullyConnectedLayer(50)
reluLayer
fullyConnectedLayer(10)
softmaxLayer
classificationLayer];

options = trainingOptions( 'sgdm', ...
'Momentum', 0.9, ...
'MaxEpochs',400, ...
'MiniBatchSize', 8192, ...
'InitialLearnRate', 0.001, ...
'ValidationData', {xValid,tValid}, ...
'Shuffle', 'every-epoch',...
'Plots','training-progress',...
'ValidationPatience', 3,...
'ValidationFrequency', 30);


trainedNetwork = trainNetwork(xTrain,tTrain, layers,options);

%Classification Error

% xtrain
predTrain = classify(trainedNetwork,xTrain);
classificationError1 = ClassError(predTrain,tTrain);

%xvalid
predValidation = classify(trainedNetwork,xValid);
classificationError2 = ClassError(predValidation,tValid);

%xtest
predTest = classify(trainedNetwork,xTest);
classificationError3 = ClassError(predTest,tTest);

disp('Classification training error for network 1:')
disp(classificationError1)
disp('Classification validation error for network 1:')
disp(classificationError2)
disp('Classification validation error for network 1:')
disp(classificationError3)
```

A.3 ReLU network 2

```
clear all
close all

[xTrain, tTrain, xValid, tValid, xTest, tTest] = LoadCIFAR(3);

layers = [imageInputLayer([32 32 3])
fullyConnectedLayer(50)
reluLayer
fullyConnectedLayer(50)
reluLayer
fullyConnectedLayer(50)
reluLayer
fullyConnectedLayer(10)
softmaxLayer
classificationLayer];

options = trainingOptions( 'sgdm', ...
'Momentum', 0.9, ...
'MaxEpochs',400, ...
'MiniBatchSize', 8192, ...
'InitialLearnRate', 0.003, ...
'ValidationData', {xValid,tValid}, ...
'Shuffle', 'every-epoch',...
'Plots','training-progress',...
'ValidationPatience', 3,...
'ValidationFrequency', 30);


trainedNetwork = trainNetwork(xTrain,tTrain, layers,options);

%Classification Error

% xtrain
predTrain = classify(trainedNetwork,xTrain);
classificationError1 = ClassError(predTrain,tTrain);

%xvalid
predValidation = classify(trainedNetwork,xValid);
classificationError2 = ClassError(predValidation,tValid);

%xtest
predTest = classify(trainedNetwork,xTest);
classificationError3 = ClassError(predTest,tTest);

disp('Classification training error for network 2:')
disp(classificationError1)
disp('Classification validation error for network 2:')
disp(classificationError2)
disp('Classification validation error for network 2:')
disp(classificationError3)
```

A.4 ReLU network 3

```
clear all
close all

[xTrain, tTrain, xValid, tValid, xTest, tTest] = LoadCIFAR(3);

layers = [imageInputLayer([32 32 3])
fullyConnectedLayer(50)
reluLayer
fullyConnectedLayer(50)
reluLayer
fullyConnectedLayer(10)
softmaxLayer
classificationLayer];

options = trainingOptions( 'sgdm', ...
'Momentum', 0.9, ...
'MaxEpochs',400, ...
'MiniBatchSize', 8192, ...
'InitialLearnRate', 0.001, ...
'ValidationData', {xValid,tValid}, ...
'Shuffle', 'every-epoch',...
'Plots','training-progress',...
'L2Regularization', 0.2, ...
'ValidationPatience', 3,...
'ValidationFrequency', 30);

trainedNetwork = trainNetwork(xTrain,tTrain, layers,options);

%Classification Error

% xtrain
predTrain = classify(trainedNetwork,xTrain);
classificationError1 = ClassError(predTrain,tTrain);

%xvalid
predValidation = classify(trainedNetwork,xValid);
classificationError2 = ClassError(predValidation,tValid);

%xtest
predTest = classify(trainedNetwork,xTest);
classificationError3 = ClassError(predTest,tTest);

disp('Classification training error for network 3:')
disp(classificationError1)
disp('Classification validation error for network 3:')
disp(classificationError2)
disp('Classification test error for network 3:')
disp(classificationError3)
```

A.5 Convolution network 1

```
clear all
close all

[xTrain, tTrain, xValid, tValid, xTest, tTest] = LoadCIFAR(4);

layers = [imageInputLayer([32 32 3])
convolution2dLayer(5,20,'Padding',1,'Stride',1)
reluLayer
maxPooling2dLayer(2,'Padding',0,'Stride',2)
fullyConnectedLayer(50)
reluLayer
fullyConnectedLayer(10)
softmaxLayer
classificationLayer];

options = trainingOptions( 'sgdm', ...
'Momentum', 0.9, ...
'MaxEpochs',120, ...
'MiniBatchSize', 8192, ...
'InitialLearnRate', 0.001, ...
'ValidationData', {xValid,tValid}, ...
'Shuffle', 'every-epoch',...
'Plots','training-progress',...
'ValidationPatience', 3,...
'ValidationFrequency', 30);


trainedNetwork = trainNetwork(xTrain,tTrain, layers,options);

%Classification Error

% xtrain
predTrain = classify(trainedNetwork,xTrain);
classificationError1 = ClassError(predTrain,tTrain);

%xvalid
predValidation = classify(trainedNetwork,xValid);
classificationError2 = ClassError(predValidation,tValid);

%xtest
predTest = classify(trainedNetwork,xTest);
classificationError3 = ClassError(predTest,tTest);

disp('Classification training error for network 1:')
disp(classificationError1)
disp('Classification validation error for network 1:')
disp(classificationError2)
disp('Classification validation error for network 1:')
disp(classificationError3)
```

A.6 Convolution network 2

```
clear all
close all

[xTrain, tTrain, xValid, tValid, xTest, tTest] = LoadCIFAR(4);

layers = [imageInputLayer([32 32 3])
convolution2dLayer(3,20,'Padding',1,'Stride',1)
batchNormalizationLayer
reluLayer
maxPooling2dLayer(2,'Padding',0,'Stride',2)
convolution2dLayer(3,30,'Padding',1,'Stride',1)
batchNormalizationLayer
reluLayer
maxPooling2dLayer(2,'Padding',0,'Stride',2)
convolution2dLayer(3,50,'Padding',1,'Stride',1)
batchNormalizationLayer
reluLayer
fullyConnectedLayer(10)
softmaxLayer
classificationLayer];

options = trainingOptions( 'sgdm', ...
'Momentum', 0.9, ...
'MaxEpochs',120, ...
'MiniBatchSize', 8192, ...
'InitialLearnRate', 0.001, ...
'ValidationData', {xValid,tValid}, ...
'Shuffle', 'every-epoch',...
'Plots','training-progress',...
'ValidationPatience', 3,...
'ValidationFrequency', 30);


trainedNetwork = trainNetwork(xTrain,tTrain, layers,options);

%Classification Error

% xtrain
predTrain = classify(trainedNetwork,xTrain);
classificationError1 = ClassError(predTrain,tTrain);

%xvalid
predValidation = classify(trainedNetwork,xValid);
classificationError2 = ClassError(predValidation,tValid);

%xtest
predTest = classify(trainedNetwork,xTest);
classificationError3 = ClassError(predTest,tTest);

disp('Classification training error for network 2:')
disp(classificationError1)
disp('Classification validation error for network 2:')
disp(classificationError2)
disp('Classification validation error for network 2:')
disp(classificationError3)
```