Task 3

Devosmita Chatterjee

910812-7748

Network	C_Training	C_Validation	C_Test	No_Epochs
Network 1	0.4368	0.5140	0.5123	203
Network 2	0.4474	0.5296	0.5247	165
Network 3	0.3920	0.4984	0.4975	368

Table 1: The table presents the classification errors on the training set (C_Training), the validation set (C_Validation), the test set (C_Test) obtained for Networks 1-3 as well as the number of training epochs (No_Epochs) that passed before reaching early stopping.

Table 1 represents the classification errors on all the datasets for networks 1-3 and the number of training epochs (No_Epochs) that passed before reaching early stopping. The classification errors on the training sets are low and the classification errors on the validation set and the test set are slightly higher for all networks. The classification errors on all datasets for network 2 are higher than that for network 1 and the classification errors on all datasets for network 3 are lower than that for network 2 and network 1.

1 Results and Discussion

The results and discussion are the following-

1. In table 1, we see that the classification error on the training set is low and the classification errors on the validation set and the test set are slightly higher for network 1. So we infer that the neural network model is almost a good fit.

2. From table 1, we find that the classification errors on all datasets for network 2 are higher than that for network 1. One reason behind this is overfitting. Since the neural network model is already a good fit, there is no need of increasing a hidden layer of 50 neurons. Another reason might be the increased learning rate 0.03.

3. From table 1, we find that the classification errors on all datasets for network 3 are lower than that for network 2 and even for network 1. This happens due to tuning the L2 regularization to 0.2 from 0 since increasing the L2 regularization reduces the risk of overfitting.

Network	C_Training	C_Validation	C_Test	No_Epochs
Network 1	0.3318	0.3978	0.3992	120
Network 2	0.2611	0.2981	0.2956	120

Table 1: The table presents the classification errors on the training set (C_Training), the validation set (C_Validation), the test set (C_Test) obtained for Networks 1-3 as well as the number of training epochs (No_Epochs) that passed before reaching early stopping.

Table 1 represents the classification errors on all the datasets for networks 1-2 and the number of training epochs (No_Epochs) that passed before reaching early stopping. The classification errors on the training sets are low and the classification errors on the validation set and the test set are slightly higher for all networks. The classification errors on all datasets for network 2 are lower than that for network 1.

1 Results and Discussion

The results and discussion are the following-

1. In table 1, we see that the classification error on the training set is low and the classification errors on the validation set and the test set are slightly higher for network 1. So we conclude that the neural network model is almost a good fit.

2. In comparison to the network 2, the classification errors on all datasets for network 2 are lower than that for network 1 because of inclusion of the layers - convolution2dLayer, maxPooling2dLayer and batchNormalizationLayer.

Adding a convolution2dLayer reduces the number of neurons which regularises the network and hence it reduces the risk of overfitting. It increases the classification accuracies.

maxPooling2dLayer helps in reduction of the feature map which not only reduces the computational cost but also overfitting.

batchNormalizationLayer decreases the amount by which the hidden neuron shift. It reduces the risk of overfitting.

Here, the neural network model is a better fit.

Appendix

ReLU, softmax, early stopping 2019

Network 1

```
clear;clc;
%%
[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',...
'MiniBatchSize', 8192,...
'ValidationData', {xValid, tValid},...
'ValidationFrequency', 30,...
'MaxEpochs',400,...
'Plots', 'Training-Progress',...
'L2Regularization', 0, ...
'Momentum', 0.9, ...
'ValidationPatience', 3, ...
'Shuffle', 'every-epoch', ...
'InitialLearnRate', 0.001);

net = trainNetwork(xTrain, tTrain, layers, options);
classify_Train = net.classify(xTrain);
classify_Valid = net.classify(xValid);
classify_Test = net.classify(xTest);

count1 = 0;
for i = 1:size(tTrain,1)
   if (classify_Train(i,:)~=tTrain(i,:))
       count1 = count1+1;
   end
end
C_Train = count1/size(tTrain,1)

count2 = 0;
for i = 1:size(tValid,1)
   if (classify_Valid(i,:)~=tValid(i,:))
       count2 = count2+1;
   end
end
```

```
C_Valid = count2/size(tValid,1)

count3 = 0;
for i = 1:size(tTest,1)
    if (classify_Test(i,:)~=tTest(i,:))
        count3 = count3+1;
    end
end
C_Test = count3/size(tTest,1)

%C_Train = 0.4368


%C_Valid = 0.5140


%C_Test = 0.5123


%Epoch 203 of 400
```

Network 2

```
clear;clc;
%%
[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',...
'MiniBatchSize', 8192,...
'ValidationData', {xValid, tValid},...
'ValidationFrequency', 30,...
'MaxEpochs',400,...
'Plots', 'Training-Progress',...
'L2Regularization', 0, ...
'Momentum', 0.9, ...
'ValidationPatience', 3, ...
'Shuffle', 'every-epoch', ...
'InitialLearnRate', 0.003);

net = trainNetwork(xTrain, tTrain, layers, options);
classify_Train = net.classify(xTrain);
classify_Valid = net.classify(xValid);
classify_Test = net.classify(xTest);

count1 = 0;
for i = 1:size(tTrain,1)
   if (classify_Train(i,:)~=tTrain(i,:))
      count1 = count1+1;
   end
end
C_Train = count1/size(tTrain,1)

count2 = 0;
for i = 1:size(tValid,1)
   if (classify_Valid(i,:)~=tValid(i,:))
      count2 = count2+1;
   end
end
C_Valid = count2/size(tValid,1)
```

```
count3 = 0;
for i = 1:size(tTest,1)
    if (classify_Test(i,:)~=tTest(i,:))
        count3 = count3+1;
    end
end
C_Test = count3/size(tTest,1)


%C_Train = 0.4474

%C_Valid = 0.5296

%C_Test = 0.5247


%Epoch 165 of 400
```

Network 3

```
clear;clc;
%%
[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',...
'MiniBatchSize', 8192,...
'ValidationData', {xValid, tValid},...
'ValidationFrequency', 30,...
'MaxEpochs',400,...
'Plots', 'Training-Progress',...
'L2Regularization', 0.2, ...
'Momentum', 0.9, ...
'ValidationPatience', 3, ...
'Shuffle', 'every-epoch', ...
'InitialLearnRate', 0.001);

net = trainNetwork(xTrain, tTrain, layers, options);
classify_Train = net.classify(xTrain);
classify_Valid = net.classify(xValid);
classify_Test = net.classify(xTest);

count1 = 0;
for i = 1:size(tTrain,1)
    if (classify_Train(i,:)~=tTrain(i,:))
        count1 = count1+1;
    end
end
C_Train = count1/size(tTrain,1)

count2 = 0;
for i = 1:size(tValid,1)
    if (classify_Valid(i,:)~=tValid(i,:))
        count2 = count2+1;
    end
end
C_Valid = count2/size(tValid,1)
```

```
count3 = 0;
for i = 1:size(tTest,1)
    if (classify_Test(i,:)~=tTest(i,:))
        count3 = count3+1;
    end
end
C_Test = count3/size(tTest,1)

%C_Train = 0.3920


%C_Valid = 0.4984


%C_Test = 0.4975


%Epoch 368 of 400
```

Convolutional networks 2019

Network 1

```
clear;clc;
%%
[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',...
'MiniBatchSize', 8192,...
'ValidationData', {xValid, tValid},...
'ValidationFrequency', 30,...
'MaxEpochs',120,...
'Plots', 'Training-Progress',...
'L2Regularization', 0, ...
'Momentum', 0.9, ...
'ValidationPatience', 3, ...
'Shuffle', 'every-epoch', ...
'InitialLearnRate', 0.001);

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

classify_Train = net.classify(xTrain);
classify_Valid = net.classify(xValid);
classify_Test = net.classify(xTest);

count1 = 0;
for i = 1:size(tTrain,1)
    if (classify_Train(i,:)~=tTrain(i,:))
        count1 = count1+1;
    end
end
C_Train = count1/size(tTrain,1)

count2 = 0;
for i = 1:size(tValid,1)
    if (classify_Valid(i,:)~=tValid(i,:))
        count2 = count2+1;
    end
```

```
end
C_Valid = count2/size(tValid,1)

count3 = 0;
for i = 1:size(tTest,1)
    if (classify_Test(i,:)~=tTest(i,:))
        count3 = count3+1;
    end
end
C_Test = count3/size(tTest,1)

%C_Train = 0.3318

%C_Valid = 0.3978

%C_Test = 0.3992

%epoch 120 of 120
```

Network 2

```
clear;clc;
%%
[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',...
'MiniBatchSize', 8192,...
'ValidationData', {xValid, tValid},...
'ValidationFrequency', 30,...
'MaxEpochs',120,...
'Plots', 'Training-Progress',...
'L2Regularization', 0, ...
'Momentum', 0.9, ...
'ValidationPatience', 3, ...
'Shuffle', 'every-epoch', ...
'InitialLearnRate', 0.001);

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

classify_Train = net.classify(xTrain);
classify_Valid = net.classify(xValid);
classify_Test = net.classify(xTest);

count1 = 0;
for i = 1:size(tTrain,1)
    if (classify_Train(i,:)~=tTrain(i,:))
        count1 = count1+1;
    end
end
C_Train = count1/size(tTrain,1)

count2 = 0;
```

```
for i = 1:size(tValid,1)
    if (classify_Valid(i,:)~=tValid(i,:))
        count2 = count2+1;
    end
end
C_Valid = count2/size(tValid,1)

count3 = 0;
for i = 1:size(tTest,1)
    if (classify_Test(i,:)~=tTest(i,:))
        count3 = count3+1;
    end
end
C_Test = count3/size(tTest,1)

%C_Train = 0.2611

%C_Valid = 0.2981

%C_Test = 0.2956

%epoch 120 of 120
```