Home Assignment 2 - FFR135 Artificial Neural Networks

Table of Contents

Author: Arshad Nowsath(nowsath)

Loading the dataset from MNIST

```
clc;
clear;
clear all;

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

```
Preparing MNIST data...
MNIST data preparation complete.
```

Convolutional neural network 1

```
% Options to train the network using stochastic gradient descent

options = trainingOptions('sgdm', ...
    'Momentum', 0.9, ...
    'MaxEpochs',60, ...
    'InitialLearnRate',0.001, ...
    'MiniBatchSize',8192, ...
    'Shuffle', 'every-epoch', ...
    'ValidationData',{xValid,tValid}, ...
    'ValidationFrequency',30, ...
    'ValidationPatience',5, ...
    'Plots','training-progress');

% Layout of the layers
layers = [
    imageInputLayer([28 28 1])


 convolution2dLayer(5,20,'stride',1,'Padding',1,'WeightsInitializer','narrow-
normal')

    reluLayer
```

```
    maxPooling2dLayer(2,'Stride',2)

    fullyConnectedLayer(100,'WeightsInitializer','narrow-normal')

    reluLayer

    fullyConnectedLayer(10,'WeightsInitializer','narrow-normal')
    softmaxLayer

    classificationLayer];

% Training the network
network_1 = trainNetwork(xTrain, tTrain, layers, options);

% Computing the scores
[prediction_train_1, scores_train_1] = classify(network_1, xTrain);
[prediction_valid_1, scores_valid_1] = classify(network_1, xValid);
[prediction_test_1, scores_test_1] = classify(network_1, xTest);

% Classification errors  obtained on the training, validation, and
 test sets sets
Classification_train_1 = classification_error(tTrain,
 prediction_train_1);
Classification_valid_1 = classification_error(tValid,
 prediction_valid_1);
Classification_test_1 = classification_error(tTest,
 prediction_test_1);

Training on single CPU.
Initializing input data normalization.
|
=========================================================================================
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation
 |  Mini-batch  |  Validation  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |   Accuracy
 |     Loss     |     Loss     |      Rate       |
|
=========================================================================================
|       1 |           1 |       00:00:06 |        7.68% |        8.57%
 |       2.3240 |       2.3136 |          0.0010 |
|       5 |          30 |       00:01:30 |       87.79% |       88.05%
 |       0.4298 |       0.4352 |          0.0010 |
|       9 |          50 |       00:02:20 |       90.94% |
 |       0.3035 |              |          0.0010 |
|      10 |          60 |       00:02:52 |       92.59% |       92.30%
 |       0.2588 |       0.2639 |          0.0010 |
|      15 |          90 |       00:04:22 |       94.02% |       94.19%
 |       0.2020 |       0.1978 |          0.0010 |
|      17 |         100 |       00:04:48 |       94.95% |
 |       0.1749 |              |          0.0010 |
|      20 |         120 |       00:05:40 |       95.72% |       95.49%
 |       0.1478 |       0.1615 |          0.0010 |
|      25 |         150 |       00:07:03 |       96.81% |       96.21%
 |       0.1219 |       0.1372 |          0.0010 |
```

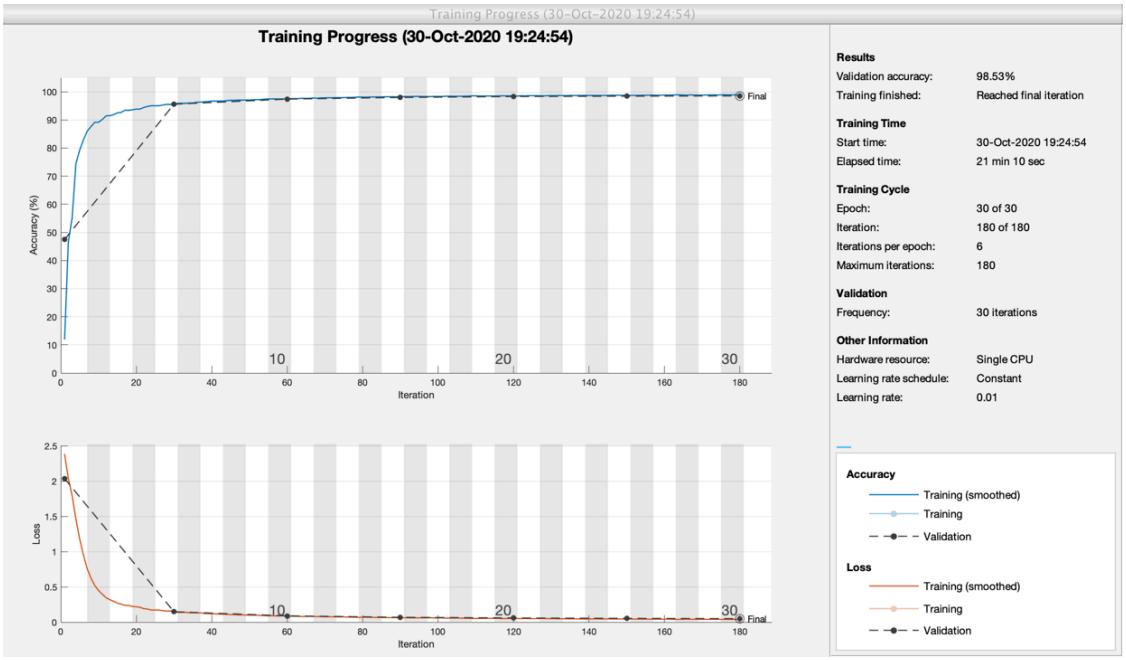
```
|      30 |         180 |       00:08:32 |       96.90% |       96.79%
 |       0.1095 |       0.1175 |           0.0010 |
|      34 |         200 |       00:09:25 |       97.23% |
 |       0.0952 |              |           0.0010 |
|      35 |         210 |       00:09:54 |       97.29% |       97.17%
 |       0.0942 |       0.1045 |           0.0010 |
|      40 |         240 |       00:11:16 |       97.45% |       97.46%
 |       0.0845 |       0.0947 |           0.0010 |
|      42 |         250 |       00:11:45 |       97.75% |
 |       0.0809 |              |           0.0010 |
|      45 |         270 |       00:12:44 |       98.11% |       97.54%
 |       0.0700 |       0.0882 |           0.0010 |
|      50 |         300 |       00:14:09 |       97.90% |       97.63%
 |       0.0692 |       0.0827 |           0.0010 |
|      55 |         330 |       00:15:36 |       98.35% |       97.80%
 |       0.0644 |       0.0774 |           0.0010 |
|      59 |         350 |       00:16:35 |       98.17% |
 |       0.0588 |              |           0.0010 |
|      60 |         360 |       00:17:05 |       98.24% |       97.78%
 |       0.0583 |       0.0755 |           0.0010 |
|
=========================================================================================
```

Convolutional neural network 1

```
% Options to train the network using stochastic gradient descent

options = trainingOptions('sgdm', ...
    'Momentum', 0.9, ...
    'InitialLearnRate',0.01, ...
    'MaxEpochs',30, ...
    'MiniBatchSize',8192, ...
    'Shuffle', 'every-epoch', ...
    'ValidationData',{xValid,tValid}, ...
    'ValidationFrequency',30, ...
    'ValidationPatience',5, ...
    'Plots','training-progress');

% Layout of the layers
layers = [
    imageInputLayer([28 28 1])


 convolution2dLayer(3,20,'stride',1,'Padding',1,'WeightsInitializer','narrow-
normal')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)


 convolution2dLayer(3,30,'stride',1,'Padding',1,'WeightsInitializer','narrow-
normal')
```

```
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)


 convolution2dLayer(3,50,'stride',1,'Padding',1,'WeightsInitializer','narrow-
normal')
    batchNormalizationLayer
    reluLayer


    fullyConnectedLayer(10,'WeightsInitializer','narrow-normal')
    softmaxLayer

    classificationLayer];

% Training the network
network_2 = trainNetwork(xTrain, tTrain, layers, options);

% Computing the scores
[prediction_train_2, scores_train_2] = classify(network_2, xTrain);
[prediction_valid_2, scores_valid_2] = classify(network_2, xValid);
[prediction_test_2, scores_test_2] = classify(network_2, xTest);

% Classification errors  obtained on the training, validation, and
 test sets sets
Classification_train_2 = classification_error(tTrain,
 prediction_train_2);
Classification_valid_2 = classification_error(tValid,
 prediction_valid_2);
Classification_test_2 = classification_error(tTest,
 prediction_test_2);

Training on single CPU.
Initializing input data normalization.
|
=========================================================================================
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation
 |  Mini-batch  |  Validation  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |   Accuracy
 |     Loss     |     Loss     |      Rate       |
|
=========================================================================================
|       1 |           1 |       00:00:09 |       11.89% |       47.48%
 |       2.3843 |       2.0371 |          0.0100 |
|       5 |          30 |       00:03:25 |       95.58% |       95.58%
 |       0.1523 |       0.1499 |          0.0100 |
|       9 |          50 |       00:05:30 |       97.07% |
 |       0.1048 |              |          0.0100 |
|      10 |          60 |       00:06:33 |       97.62% |       97.39%
 |       0.0837 |       0.0914 |          0.0100 |
|      15 |          90 |       00:11:29 |       98.40% |       97.99%
 |       0.0655 |       0.0709 |          0.0100 |
```

```
|      17 |         100 |       00:12:41 |       98.29% |
 |       0.0608 |             |          0.0100 |
|      20 |         120 |       00:14:53 |       98.51% |       98.32%
 |       0.0588 |      0.0617 |          0.0100 |
|      25 |         150 |       00:17:54 |       98.61% |       98.45%
 |       0.0474 |      0.0554 |          0.0100 |
|      30 |         180 |       00:21:07 |       98.95% |       98.55%
 |       0.0413 |      0.0514 |          0.0100 |
|
=========================================================================================
```



Saving the network

```
save network_1;

save network_2;
```

Classification error function

```
function C = classification_error(target, outputs)
length_valset = size(target,1);
C = 1 / length_valset * sum(outputs ~= target);
end
```

Published with MATLAB® R2020a