CS 5600/6600: Intelligent Systems
Assignment 7

Vladimir Kulyukin
Department of Computer Science
Utah State University

October 12, 2019

Learning Objectives

1. Decision Trees
2. Random Forests
3. Convolutional Networks

Introduction

In this assignment, we'll compare decision trees and random forests with convolutional networks on MNIST. You'll need to install `sklearn`. Go to `https://scikit-learn.org/stable/install.html` to install it. When I was writing this assignment, I discovered that `sklearn` now requires Python 3.5 or newer. Python 2 doesn't seem to be supported any longer. All good things come to an end.

Problem 1 (2 pts)

The file `mnist_digits_random_forest.py` contains several functions for loading and reshaping the MNIST data for `sklearn`. You don't have to modify them. To get MNIST loaded and reshaped, you need to run `prepare_mnist_data()` and `prepare_mnist_targets()`. Save your definition in `mnist_digits_random_forest.py`.

Write a function `test_dt()` that creates a decision tree, trains and tests this tree on the MNIST training and testing data, i.e., the arrays `mnist_train_data_dc` and `mnist_train_target_dc`. Then the function evaluates the trained random forest on the MNIST validation data (i.e., the array `mnist_valid_data_dc`) and prints the classification report and the confusion matrix. Here's a sample run.

```
>>> test_dt()

              precision    recall  f1-score   support

           0       0.92      0.94      0.93       980
           1       0.94      0.96      0.95      1135
           2       0.89      0.84      0.86      1032
           3       0.83      0.85      0.84      1010
           4       0.87      0.87      0.87       982
           5       0.81      0.82      0.82       892
           6       0.89      0.89      0.89       958
```

```
           7       0.89      0.91      0.90      1028
           8       0.82      0.79      0.81       974
           9       0.83      0.85      0.84      1009

   micro avg       0.87      0.87      0.87     10000
   macro avg       0.87      0.87      0.87     10000
weighted avg       0.87      0.87      0.87     10000

[[ 918    2    5    2    6   18   13    4    5    7]
 [   1 1087    6    8    1    6    4    7   13    2]
 [  13   13  862   39   11   14   18   25   26   11]
 [   6    5   21  856    5   45    1   14   28   29]
 [  10    6   14    8  852    4   21   11   15   41]
 [  10    9    9   40   10  730   23   13   23   25]
 [  15    6   12    7   18   24  855    2   15    4]
 [   2   12   15   16   15    3    1  932   10   22]
 [  13   13   20   39   21   36   18   15  767   32]
 [  11    3    7   19   38   19    4   23   28  857]]
```

Write a function `test_rf(num_trees)` that takes an integer that specifies the number of trees in a random forest, trains and tests this random forest on the MNIST training and testing data, i.e., the arrays `mnist_train_data_dc` and `mnist_train_target_dc`. Then the function evaluates the trained random forest on the MNIST validation data (i.e., the array `mnist_valid_data_dc`) and prints the classification report and the confusion matrix. Here's a sample run for testing a random forest with 5 trees.

```
>>> test_rf(5)

              precision    recall  f1-score   support

           0       0.91      0.98      0.94       980
           1       0.96      0.99      0.97      1135
           2       0.89      0.92      0.91      1032
           3       0.89      0.89      0.89      1010
           4       0.91      0.92      0.92       982
           5       0.89      0.89      0.89       892
           6       0.95      0.93      0.94       958
           7       0.96      0.93      0.95      1028
           8       0.91      0.86      0.88       974
           9       0.92      0.87      0.90      1009

   micro avg       0.92      0.92      0.92     10000
   macro avg       0.92      0.92      0.92     10000
weighted avg       0.92      0.92      0.92     10000

[[ 963    0    2    2    1    4    4    0    3    1]
 [   1 1121    2    4    1    1    2    0    2    1]
 [  16    6  949   12    5    6    7   11   16    4]
 [  13    8   26  903    0   28    1    7   19    5]
 [  10    3    7    5  908    1    7    3    9   29]
 [  17    6    7   40    9  791    9    1    7    5]
 [  18    4    8    1    9   19  891    1    7    0]
 [   2    8   33    8    5    1    3  957    2    9]
```

```
[  10    7   14   27   19   28   13    3  836   17]
[  10    8   14   18   37    9    3   13   22  875]]
```

Use the function `test_rf()` to implement the function `test_rf_range(low_nt, high_nt)` that specifies a range for numbers of trees in random forests and calls `test_rf()` on each number in the range `[low_nt, high_nt]`. For example, a call `test_rf_range(5, 10)` calls `test_rf` on 5, 6, 7, 8, 9, and 10. Save both definitions `mnist_digits_random_forest.py`.

Use this function to train, test, and validate random forests with the number of trees in `[5, 200]`. Write a multi-line comment at the beginning of `mnist_digits_random_forest.py` that gives the top 3 random forests with their reports and confusion matrices. Write down how much time it took you to train, test, and validate all your random forests. Recall Problem 1 from the previous assignment where you designed, trained, and evaluated several ConvNets on MNIST. In your comments, compare the performance of your random forests with your ConvNets and the time it took you to train the random forests and the ConvNets.

Problem 1 (1 pt)

Write a one-page analysis of "Neural Architecture Search with Reinforcement Learning" by Zoph and Le. Save your writeup in `paper_report.pdf`.

What to Submit

1. `mnist_digits_random_forest.py`;
2. `paper_report.pdf`;

Happy Hacking, Reading, and Writing!