CS 5600/6600: F20: Intelligent Systems
Assignment 7

Vladimir Kulyukin
Department of Computer Science
Utah State University

October 17, 2020

Learning Objectives

1. Decision Trees and Random Forests
2. Precision, Recall, F1, Confusion Matrices
3. Decision Trees and Random Forrests vs. ConvNets on MNIST

Introduction

In this assignment, we'll use MNIST to experiment with some alternatives to DL models we've discussed in the last 2 lectures. To keep the scope of this assignment reasonable, we'll confine ourselves to decision trees (DTs) and random forests (RFs). If you want to experiment with other alternatives (e.g., SVMs or KNNs) on your own, you can use code samples and pointers from the lecture on 10/14/20 (pdf in Canvas/Announcements). To complete this assignment, you'll need to install `sklearn`, one of the standard Py packages in the scientific computing stack. You can install it from scikit-learn.org

Problem 1 (2 pts)

Write a 1-page analysis of the article "Random Forests" by Dr. Leo Breiman published in *Machine Learning* in 2001. The pdf is included in the zip. This article succinctly presents the basics of the theory of random forests. Dr. Breiman demonstrates not only theoretically but also experimentally that a combination of randomly generated tree classifiers tends to perform better than any individual tree classifier by itself. The classification error of such random forests, as the number of trees in the forest becomes large, converges to a limit. This article is considered to be one of the foundational articles of ensemble-based learning. Save your 1-page analysis of this article in `cs5600_6600_f20_hw07_paper.pdf`.

A touch of terminological irony in the data-driven intelligence literature shouldn't be lost on you. DTs and RFs are often referred to as *standard* or *classical* machine learning (ML) methods although their theory was first developed in the mid/late 1990's and matured in the early 2000's. NNs, on the other hand, almost never fall under the umbrella of standard/classical ML methods although the theory of modern NNs (backprop, gradient descent, etc.) was developed in the early 1980's.

I'd like to mention, in passing, that Dr. Breiman was the major professor and doctoral thesis advisor to Dr. Adele Cutler, our resident random forest star. Dr. Cutler, who is a professor of mathematics and statistics at USU, has done outstanding work on the theory and application of random forests. I've always considered it an honor to do research at the same university with her.

Problem 2 (3 pts)

Recall that in Assignment 5 (posted on 10/4/2020) you built, trained, tested, and validated a few ConvNets on MNIST. In this problem, we'll compare the recognition accuracies of your ConvNets from that assignment with DTs and RFs from this assignment. You don't have to re-train your ConvNets. You'll simply take your best ConvNet accuracy on MNIST from Assignment 5 and compare it with your best DT and RF accuracies from this problem. You'll write your code and comparison comments in `mnist_dt_rf.py`. More on this below. I've also written a few unit tests for you in `mnist_dt_rf_uts.py`.

The file `mnist_dt_rf.py` contains several functions for loading and reshaping MNIST for `sklearn`. You don't need to modify those. All you need to do in order to get MNIST loaded and reshaped is to run `prepare_mnist_data()` and `prepare_mnist_targets()`. The zipped MNIST is in the `data` folder.

If you look at the preparation of the MNIST data, you'll notice that we aren't cutting any slack to DTs or RFs in our experiments by extracting any features from the data. Both classifiers will be evaluated on raw image data in the same way as ConvNets in Assignment 5. In other words, we'll be comparing apples to apples.

OK. Let's get started. Write a function `fit_validate_dt()` that creates a decision tree, fits this tree on the MNIST training data, (i.e., the arrays `mnist_train_data_dc` and `mnist_train_target_dc`), validates the fit tree on the MNIST validation data (i.e., the array `mnist_valid_data_dc`) and prints the classification report and the confusion matrix. See slide 7 in the lecture 12 pdf (posted on 10/12/20 in Canvas/Announcements) on how to fit and validate DTs and generate performance reports and confusion matrices. Make sure that you randomize the state of the DT classifier when creating it. Here's how.

```
clf = tree.DecisionTreeClassifier(random_state=random.randint(0, 1000))
```

Here's my output of running `test_ut01()` in `mnist_dt_rf_uts.py`.

```
Fitting DT complete...
              precision    recall  f1-score   support

           0       0.92      0.94      0.93       980
           1       0.94      0.96      0.95      1135
           2       0.88      0.83      0.85      1032
           3       0.83      0.85      0.84      1010
           4       0.87      0.87      0.87       982
           5       0.82      0.82      0.82       892
           6       0.90      0.89      0.90       958
           7       0.90      0.90      0.90      1028
           8       0.81      0.79      0.80       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

[[ 920    0    8    4    9   15    8    7    4    5]
 [   0 1087    4    9    1    5    6    7   13    3]
 [   9   14  857   37   18   10   15   26   33   13]
 [   5    6   23  858    7   40    2   12   33   24]
```

```
 [   2    5   12    5  850    8   25   14   22   39]
 [  17    9    2   46   13  733   22   12   20   18]
 [  14    4   12    7   12   22  856    2   22    7]
 [   2   13   21   12   11    6    4  923    8   28]
 [  16   13   27   34   23   32   12    7  773   37]
 [  10    4    9   21   37   19    3   19   30  857]]
.
----------------------------------------------------
Ran 1 test in 21.710s

OK
```

Running this function repeatedly will produce slightly different results, especially the confusion matrices. The important score for us is the weighted average F1 in the row right above the confusion matrix, which is equal 0.87. Since F1 combines both precision and recall, we'll use it as our measure of classification accuracy of DTs and RFs.

Write a function `fit_validate_dts(num_dts)` that takes a positive integer `num_dts` and calls `fit_validate_dt` `num_dts` times. Effectively, this function creates `num_dts` DTs with randomized initial states, fits and validates them on MNIST, and prints the performance report and confusion matrix for each DT.

Write a function `fit_validate_rf(num_dts)` that takes a positive integer `num_dts` that specifies the number of DTs in an RF, creates such an RF, fits it on the MNIST training data (i.e., on `mnist_train_data_dc` and `mnist_train_target_dc`), validates the fit RF on the MNIST validation data (i.e., `mnist_valid_data_dc`), and prints the performance report and the confusion matrix. Make sure that you randomize the initiale state of each RF before fitting it and validating it. Here's how.

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators=num_trees,
                             random_state=random.randint(0, 1000))
rf = clf.fit(mnist_train_data_dc, mnist_train_target_dc)
valid_preds = rf.predict(mnist_valid_data_dc)
```

Here's my output of running `test_ut03()` that creates, fits, and validates an RF of 5 DTs. Note that F1 is now 0.92.

```
Fitting of RF with 5 trees completed...
Validation of RF with 5 trees completed...
              precision    recall  f1-score   support

           0       0.93      0.98      0.95       980
           1       0.95      0.99      0.97      1135
           2       0.88      0.93      0.91      1032
           3       0.88      0.89      0.88      1010
           4       0.90      0.92      0.91       982
           5       0.90      0.89      0.90       892
           6       0.96      0.93      0.95       958
           7       0.94      0.92      0.93      1028
           8       0.92      0.84      0.88       974
           9       0.92      0.88      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

[[ 964    3    2    0    2    1    4    2    2    0]
 [   0 1119    6    1    0    3    2    3    1    0]
 [  18    5  964   10    3    2    3   11   12    4]
 [   7    9   30  901    2   27    0   10   17    7]
 [   3    2   14    5  906    2    7    3    7   33]
 [  12    6    5   44    3  793    7    2   12    8]
 [  14    3   10    3   19   13  890    0    5    1]
 [   3   10   35    8   10    1    0  945    3   13]
 [  15    6   24   37   14   28   10   10  818   12]
 [   5    9    2   20   49   10    0   15   16  883]]
.
----------------------------------------------------------------------
Ran 1 test in 2.258s

OK
```

Use the function `fit_validate_rf()` to implement the function `fit_validate_rfs(low_nt, high_nt)` that specifies a range for numbers of DTs in RFs and calls `fit_validate_rf()` on each number in the range `[low_nt, high_nt]`. For example, `fit_validate_rfs(5, 10)` calls `fit_validate_rf()` on 5, 6, 7, 8, 9, and 10. Use this function to fit and validate RFs with the number of DTs in `[10, 500]` in increments of 10. This run took ≈30 minutes on my laptop.

After you've done your tests. Write a multi-line comment at the beginning of `mnist_dt_rf.py` that includes the following points.

1. The performance report and confusion matrix of your top performing DT;
2. The performance report and confusion matrix of your top performing RF. Be sure to specify the number of DTs in your RF;
3. Your brief answer to the question: Do you think that ensemble learning makes a difference in going from single DTs to RFs?
4. Your brief answers to these two questions. How does the accuracy of your best MNIST ConvNet compare to the accuracies of your best MNIST DT and RF (i.e., their average weighted F1 scores)? Have you drawn any conclusions from this comparison?

What to Submit

1. `cs5600_6600_f20_hw07_paper.pdf` with your 1-page analysis of Dr. Breiman's article on RFs;
2. `mnist_dt_rf.py` with your answers to all the questions in Problem 2.

Have fun working as a random forester!