Homework 3

Large-Scale Data Analysis

Fabian Gieseke, Anders Søgaard, and Dirk Hovy

9 May 2017

Submission Deadline:
16 May 2017, ***15:00 (← More time!)***

Note: There will be four or five regular homework assignments on the LSDA course. They must be passed in order to be eligible for the final exam. Here, "passing an assignment" means that you have to make a serious attempt to solve the corresponding exercises, which can, e.g., be measured in terms of points for the individual subtasks. ***For this second assignment, you will need, in general, at least 45 out of the 90+40* points.***

1 Big Trees (90 pts)

Note: Your code must be executable on the virtual machine using at most 2GB main memory (default setting).

Figure 1: Landsat 8 image (true color)

In this exercise, we will deal with a relatively large dataset that is based on the *Landsat 8 satellite*. The satellite takes images using *multiple bands*, which yields so-called *multispectral images* with multiple values given for each pixel. Using such multispectral images, one can generate "normal" true color images, see Figure 1 for an example. If you are interested, you can check out the details related to this type of data, but it's not needed for doing the assignment.[1]

We will make use of two preprocessed Landsat 8 datasets, which are available via Absalon: `landsat_train.csv` and `landsat_test.csv`. The training set contains $n = 25,667,779$ lines. Each line contains a label (first value) and a pattern with $d = 9$ features (each pattern corresponds to a pixel in an associated multispectral image). The test set only contains patterns that stem from another image. Your task is to train a random forest on the training data and to apply it on the test data!

1. **Random Subset (20 pts):** We will make use of the random forest implementation provided by the *Scikit-Learn* package to train a random forest with 10 trees. Although this implementation depicts one of the fastest ones that are available, it would take too much time to use all the training patterns. For this reason, you will first have to extract a *random* subset of the training data:

   (a) (5 pts): Why is it, in general, a good idea to extract a random subset of the data if one cannot make use of all the training instances? Why shouldn't we simply take, e.g., the first training instances stored in the file?

   (b) (15 pts): Process the `landsat_train.csv` file to extract a subset of 100,000 instances (e.g., line by line or in chunks). Each training instance should have the same probability to be selected (uniform random subset without replacement). Store the resulting subset in a new text file named `landsat_train_subset.csv`

[1] `https://landsat.gsfc.nasa.gov/landsat-data-continuity-mission/`

having the same format as the original one. Store the remaining instances in another file called `landsat_train_remaining.csv`.

Hints: Make use of `subset = sorted(numpy.random.choice(25667779, 100000, replace=False))`. *After the files have been generated, you can count the number of lines by executing, e.g.,* `wc -l landsat_train_subset.csv`.

2. **Building Trees (10 pts):** Next, build a random forest based on the subset extracted in the previous step. Make use of the `RandomForestClassifier` class provided by the *Scikit-Learn* package using `n_estimators=10`, `criterion='gini'`, `max_depth=None`, `min_samples_split=2`, and `max_features=None` as parameters.[2].

   After training, save the random forest to a file `model.save` via the `pickle` module. Report the misclassification error on the training set.

3. **Applying Trees (20 pts):** Validate the quality by resorting to the unused training instances stored in `landsat_train_remaining.csv`. What is the misclassification error on this set? Finally, apply the model to all instances given in `landsat_test.csv` and visualize the predictions (e.g., one color per class). Here, each line of `landsat_test.csv` corresponds to a pixel in a 3000×3000 image, where the first 3000 lines correspond to the first row, the following 3000 ones to the second row, and so on. Add the resulting image to your write-up.

   Hint: Load the fitted model from the file `model.save` *generated before. Store intermediate results to files (e.g., the predictions for the test instances)!*

4. **Training Many Trees (20 pts):** Training a random forest via *Scikit-Learn* using a lot of training instances and/or many trees quickly becomes infeasible. The main problem is the memory consumption, which increases in case more data/trees are used: Firstly, the implementation loads the whole dataset into main memory—causing memory problems in case the dataset becomes too large. Secondly, all trees that are built are kept in memory—hence, building more and more trees will eventually lead to memory problems as well.[3]

   Can you nevertheless train a random forest with a huge number of trees (say, at least 100) using a random subset of at least $1,000,000$ training instances? How does the "test image" look like (add this to your write-up!).

   Hints: Make use of the Scikit-Learn `RandomForestClassifier` *class and write a "wrapper". Make use* `scipy.state.mode` *to efficiently combine predictions!*

5. **Testing Time (20 pts):** Assume that you are given fully-grown random forest for n training instances ($m = 1$, i.e., the construction of each tree is only stopped as soon as the leaves are pure). How much time is needed to obtain a prediction $\mathcal{T}(\mathbf{x})$ for a new test instance $\mathbf{x}$ per single tree $\mathcal{T}$ in (a) the best case and in (b) the worst case? Make use of the $\mathcal{O}$-notation for both cases.

   Hint: ***You can solve this subtask independently from the previous ones!*** *Two short explanations are enough.*

What to submit?

Provide all source code files in the `code.zip` file. Include answers to the questions raised in your write-up (`answers.pdf`):

2 Big Text Data (40*)

Details will be given on Thursday during the lecture.

[2]You can make use of Python 2.7, which is installed by default. The corresponding documentation can be found here: `http://scikit-learn.org/0.17/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier`

[3]If you are curious: `http://datascience.la/benchmarking-random-forest-implementations/`.