Homework 5

Large-Scale Data Analysis

Fabian Gieseke

30 May 2017

Submission Deadline:
6 June 2017, 10:00

Note: For this assignment, you will need, in general, at least 50 out of the 100+20 points.*

1 K-D Trees & Feature Selection (50 pts)

In this exercise, you will use k-d trees to speed up the nearest neighbor computations for the real-world task you have already worked with in the context of Homework 1 (distant galaxies). This time, however, we will use larger dataset instances! Download the zip file provided on Absalon and consider the `train.csv`, `validation.csv`, and `test.csv` files in the `neighbors` subdirectory, which contain 100,000 lines for both `train.csv` and `validation.csv` and 1,000,000 lines for `test.csv`, respectively.

1. *K-D Trees (20 pts):* Adapt the initial `nn.py` file that was provided for Homework 1 (Part 3.1) such that a k-d tree is used instead of the brute-force approach (i.e., change the parameter `brute`). You should be able to execute the script without running into memory problems. What is the error on the test set measured in terms of the mean squared error? What is the runtime needed for the testing phase? Can you briefly explain (2-5 sentences) why no memory error occurs anymore?

2. *Feature Subset (10 pts):* Next, only use a subset of the 15 features for both training and testing. More precisely, only consider the first 5 features. What is the induced test error? Compare the runtime needed for the testing phase with the previous on (that was based on all features). What might be the reason for the differences?

3. *Forward Selection (20 pts):* This part involves some more coding: Instead of using the, say, first 5 features, a popular strategy is to incrementally select "good" features. More precisely, in the first round, one selects the feature out of all available features that leads to the smallest error on a validation set. In the second round, another feature is selected out of the remaining features, which leads, *together* with the first feature that was already selected in the first round, to the lowest validation error. This incremental process, called *forward feature selection*, continues until a pre-defined number of features is selected.[1]

   - (15 pts) Implement this scheme for the nearest neighbor model at hand to incrementally select 5 features! Use the `validation.csv` dataset to measure the errors of the intermediate models. Which features are selected in the end? What is the induced testing error? What is the runtime for the testing phase?
   - (5 pts) Plot the validation errors obtained during this process (i.e., the lowest valdiation error after each round). How does the resulting curve look like in case you continue this process until all features are selected?

What to submit?

Provide all source code files (but not the datasets) in the `code.zip` file. Include answers to the questions raised in your write-up (`answers.pdf`):

[1]For details, see Hastie, Tibshirani, and Friedman. *The Elements of Statistical Learning*, pages 57–60, `https://statweb.stanford.edu/~tibs/ElemStatLearn/printings/ESLII_print10.pdf`

2 Hadoop & MapReduce (50 + 20* pts)

This part of the homework is about Hadoop. Make use of your virtual machine and the Hadoop cluster that is already installed on it. After having started your virtual machine, you first need to start the Hadoop cluster by executing `./.start_hadoop` (from your home directory), see the lecture slides for details. Afterwards, you will be able to submit jobs to the Hadoop cluster![2]

Hint: You can quickly test the mappers and reducers via Unix Streaming, e.g., via:

```
cat 2016_1.csv | ./mapper.py | sort | ./reducer.py
```

Note that both the `mapper.py` and the `reducer.py` need to be executable, which you can achieve via `chmod 755 mapper.py` and `chmod 755 reducer.py`.

1. *Word Count (10 pts):* Adapt the word count example shown in the lecture such that the following punctuation marks are ignored (as if not being there): ".", ",", "!", ":", "?". Execute the code on the Hadoop cluster, i.e., after having started the cluster, copy the file `lsda.txt` to HDFS and execute the MapReduce job. Once the cluster execution is finished, copy the output file `part-00000` back from HDFS to the local file system. Add all the output produced by the MapReduce job and the content of the file `part-00000` to your write-up.

2. *Airline Statistics (40 pts):* Next, we will analyze a dataset containing information about flights in the USA that stems from the *Bureau of Transportation Statistics.*[3]. In the zip file provided, you will find a directory `airline_data` containing the files `2016_1.csv`, . . . , `2016_6.csv`. Each line of a file (except the headers) contains the flight date, the airline ID, the flight number, the origin airport, the destination airport, the departure time, the departure delay in minutes, the arrival time, the arrival delay in minutes, the time in air in minutes, and the distance between both airports in miles.

   (a) *Total (10 pts):* Create a directory `airline_data` on the Hadoop cluster and copy all csv files from the local file system to this HDFS directory. Write a mapper and reducer that computes the total departure delay per airport in minutes (ignore all negative values for the delay, which indicate a departure before the scheduled time). What is the total departure delay for airport LAX? Have a look at the output produced by the Hadoop job: What is the number of "Reduce output records"?

   (b) *Max (15 pts):* Write a mapper and reducer that computes the maximal arrival delay per airport in minutes. What is the maximal arrival delay for airport YUM? What is the number of "Reduce output records"?

   (c) *Mean (15 pts):* Write a mapper and reducer that compute the average departure delay per airport (again, ignore all negative delay values). What is the average departure delay for airport SFO? Next, make use of a combiner that is applied after the mapping phase (see lecture). Compare the number of "Reduce input records" between both Hadoop jobs (i.e., without and with combiner).

3. *Landat (20* pts):* Write a mapper and reducer that compute the average misclassification training error for the Landsat data used in Homework 3 (i.e., for the lines given in `landsat_training_subset.csv`). Each mapper should compute predictions for all incoming data by loading the model from the file `model.save`, which is stored on the local file system of the worker.[4] Execute the job on the Hadoop cluster and add the output produced to your write-up.

What to submit?

Provide all source code files (mappers/reducers/combiners) as well as the produced output files (`part-00000`) in the `code.zip` file. Include answers to the questions raised in your write-up (`answers.pdf`):

[2]To simplify this process, you can define an alias `hrun` in `.bash_aliases` (see lecture slides).

[3]`https://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236&DB_Short_Name=On-Time`

[4]A reference solution for Homework 3 is available on Absalon. Executing the first two files should yield the `model.save` file. Since the Hadoop cluster is a pseudo-distributed cluster that runs on a single machine, you can simply specify the absolute path to this file in the mapper (e.g., `/home/lsda/HW5/forests/model.save`).