Homework 4
Large-Scale Data Analysis
(individual submission)

Fabian Gieseke & Christian Igel

Department of Computer Science
University of Copenhagen

28 May 2019

New Submission Deadline:
14 June 2019, 23:59

This assignment is an **individual assignment**, i.e., you are supposed to solve the assignment on your own. In particular, the report and the code must be written completely by yourself!

Virtual Machine: For the individiual assignments, we will generally assume that you make use of the LSDA virtual machine, i.e., all instructions provided are tailored to this system. Of course you are free to use your own desktop environment, tools, and Python installation to implement and test your code. Keep in mind, however, that the output might differ depending on the Python version and the installed packages.
Important: You have to make sure that your code runs on the virtual machine!

Deliverables: Please submit *two separate files* via Absalon: (1) a pdf file `answers.pdf` containing your answers and (2) a `code.zip` file containing the associated code as zip file (do

1 Analyzing Airline Data (50 pts)

We will analyze a dataset containing information about flights in the USA that stems from the *Bureau of Transportation Statistics.*[1]. 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. Make use of your virtual machine and the Hadoop cluster that is already installed on it. For both subtasks, keep in mind to first activate your Python3 environment via `source  /.venvs/lsda/bin/activate`.

1. *Data Analysis with Hadoop:* You first need to start Hadoop by executing `./.start_dfs_yarn.sh` from your home directory. 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`. **Also make sure that the first line of your mapper.py/reducer.py files equals "#!/usr/bin/env python3".**
   Hint: You can quickly test the mappers and reducers via Unix Streaming, e.g., via:

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

   (a) *Shortest Flight Distance (5 pts):* Write a mapper and reducer that compute the smallest flight distance per airline ID. The output generated by the reducer should contain, per

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

line, an airline ID along with the associated minimal flight distance. What is minimal distance for airline ID 20436?

(b) *Late Arrival Counts (5 pts):* Write a mapper and reducer that compute the total number of delayed flights per airline ID. Here, a delayed flight is one with a positive arrival delay (treat missing values as 0). The reducer should output, for each airline ID, the number of total flights, the number of delayed flights, and the percentage of delayed flights. What is the percentage of delayed flights for airline ID 20436?

(c) *Mean and Standard Deviation for Arrival Delays (10 pts):* Write a mapper and reducer that compute the mean $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ and the standard deviation $s = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$ for the arrival delays $x_1, \ldots, x_n$ for each airline ID. Consider all delay values including the negative ones (treat missing values as 0). A reducer should output, for each airline ID, the associated mean and standard deviation. What is the mean and standard deviation for airline ID 20436?

(d) *Top-10 of Arrival Delays (10 pts):* Write a mapper and reducer that compute the 10 most delayed flights for each airline ID. The reducer should output, for each airline ID, a list containing the delays for the 10 most delayed flights. What are the 10 most delayed flights for airline ID 20436?

Note: Write efficient mappers/reducers in the sense that only a constant amount of additional main memory is used by each of the workers. For instance, you can define a couple of "counters" (like for the word count example) or lists of constant size. An example for an inefficient reducer would be a reducer that simply collects all the departure delays per airport in a list to subsequently compute the total departure delay (this list could become very large depending on the amount of incoming data!). In doubt, feel free to get in touch with us!

2. *Data Analysis with Spark:* Implement the four jobs for the Airline dataset in Apache Spark! As a starting point, make use of the Jupyter notebook `Airline Stats.ipynb`, which already creates an RDD based on the airline dataset. You are supposed to implement the tasks in a "similar" fashion as before, i.e., you are supposed to (only) make use of `map` and `reduceByKey` (do not simply make use of some Spark functions that yield, e.g., the mean and standard deviation). In doubt, feel free to ask via Absalon ;-).

   Note: You have to execute `source .activate_jupyter_pyspark` before running `pyspark`.

Deliverables

Include answers to the questions raised above to your write-up (`answers.pdf`). Provide, for each subtask, the mapper.py/reducer.py files in a subdirectory (e.g., `1a_shortest_flight_distance`) of your overall source code archive. **Your are also supposed to run all jobs on the Hadoop cluster.** Please add the text output produced by each Hadoop job (e.g., "copy and paste" from the terminal) as well as the generated output files (e.g., `part-00000`) to your `code.zip`. For the Spark task, please add the modified `Airline Stats_HW4.ipynb` notebook to your code.

2 Distributed Linear Regression with Apache Spark (25 pts)

We have briefly discussed how to conduct gradient descent for fitting a logistic regression model in a distributed fashion via Spark. You can find the correponding notebook on Absalon (`Logistic Regression Spark.ipynb`). This part of the exercise is about immplementing linear least-squares regression in a similar fashion!

1. Warm-Up: Follow the code given in the `Logistic Regression Spark.ipynb` notebook.

2. Next, have a look at the first part of the `Distributed_Linear_Least_Squares.ipynb` notebook. The class `LinearLeastSquaresRegression` computes a standard linear regression model (no distributed computations) and resorts to the "black box" optimizer `fmin_l_bfgs_b`

of the `scipy.optimize` package. Note that it is also possible to only provide a function that computes the function values (i.e., the gradient is not necessarily needed). Thus, it is enough to provide a function to the optimizer that computes

$$F(\mathbf{w}) = \frac{1}{n}(\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \mathbf{w}^T \mathbf{w} \tag{1}$$

for a given $\mathbf{w} \in \mathbb{R}^d$, see Slide 19 of Lecture 4.

3. Implement this approach such that the linear least-squares models are computed in a distributed fashion via `map` and `reduce`. More precisely, finalize the implementation of `_function` of the class `LinearLeastSquaresRegressionSpark`, which should return, for a given $\mathbf{w} \in \mathbb{R}^d$, the function value $F(\mathbf{w})$. Note that you can assume that the single vector $\mathbf{w}$ is small enough to be handled by the driver node that executes the optimizer. However, you implementation should be able to deal with a very large number n of data points stored in matrix $\mathbf{X}$ and $\mathbf{y}$, respectively. Briefly describe the individial steps of your distributed function evaluation (5-10 lines). Add the final figure that is generated to you write-up.

3 Spam Filter with Apache Spark (25 + 15* pts)

Your task is to develop a spam filter in Apache Spark. You can find the file `spam_ham.zip` on Absalon, which contains emails as text files stored in two directories ('spam' and 'ham').[2] You can also find a jupyter notebook `SpamClassifier.ipynb` on Absalon, which you are supposed to extend.

1. *Basic Classifier (25 pts):* For the first part, you are supposed to follow the steps sketched below to design an initial simple baseline classifier.

   (a) *Data (5 pts):* Move the data to your distributed file system (you can use the command `hadoop fs -copyFromLocal` for this). Execute the first cell. Afterwards, collect the first entry for each RDD via a `take(1)` action. How do they returned entries look like?

   (b) *Features (5 pts):* Split each email into a list of words and make use of Spark's `HashingTF` transformer to convert each email to a (sparse) feature vector.[3] Use feature vectors of length 10000 (i.e., `numFeatures=10000`).

   (c) *Labelled training/test data (5 pts):* Convert each feature vector into a labeled data point using Spark's `LabeledPoint` class.[4] Combine both RDDs to a single RDD via `union`. Make use of `randomSplit` to partition your data into a training and a test RDD (80% training, 20% test, seed=0).

   (d) *Model (5 pts):* Fit a `LogisticRegressionWithLBFGS` (default parameters) model on the training RDD. Apply the model to the test RDD to generate a new RDD containing both the true labels as well as the associated predictions.

   (e) *Evaluation (5 pts):* Use `MulticlassMetrics` to compute the accuracy on the test RDD. What is the test accuracy of your model?

2. *Surprise Us (15* pts):* In the second part, you are supposed to extend the first part by, e.g., using other feature extraction schemes and/or other machine learning models. Your implementation should make use of Spark DataFrames! Can you get a better classification accuracy? The quality of your approach (e.g., who sophisticated your modifications are) as well as the quality of your write-up (up to one page) will be taken into account. Thus, this part can be seen as a "mini project", where you are basically allowed to make use of any Spark concepts and tools.[5]

[2] Subset of the dataset provided here: `https://github.com/zrz1996/Spam-Email-Classifier-DataSet`

[3] `https://spark.apache.org/docs/latest/ml-features.html`

[4] `https://spark.apache.org/docs/latest/mllib-data-types.html`

[5] Note that by doing this part, you can reach up to 115 points for this homework; these additional points will be taken into account for the final grade! For instance, if you manage to receive 80, 85, 100, 115 points for HW1-HW4, the average over your best three assignments will be 100 points.

Deliverables

(1) Include answers to the questions raised above to your write-up (`answers.pdf`). Add your extended notebook `SpamClassifier.ipynb` to your code. (2) Describe your project/approach in detail by commenting on the individual steps (up to one page). Provide all your code.