Heidelberg University
Winter semester 2022

Group Parallel and Distributed Systems (PVS)
Artur Andrzejak

Problem Set 1 for lecture Mining Massive Datasets

Due October 31, 2022, 23:59 CET

Exercise 1 **(3 points)**

Consider a cluster of n machines. Each has a probability p of failing in a given period of time T.

a) What is the probability of at least one machine failure during this period of time? Hint: Consider the complementary event that no machine fails during that time.

b) For $0 \leq k \leq n$, what is the probability p_k (in terms of k, n and p), of exactly k machines failing during T? Give an explanation.

c) Show that the probabilites from **b)** satisfy: $p_1 + p_2 + \ldots + p_n = 1 - (1-p)^n$.

Exercise 2 **(3 points)**

In this exercise you will familiarize yourself with RDDs in Spark. For solutions you can use either Python or Java.

a) Describe each of the following transformations: `join(), sort(), groupby().` In particular, what is in each case the type of input and type of output? Give for each one an example ("on paper") with a simple input and output.

b) Give three own programming examples of your choice for transformations (but not for `map()` or `filter()`) and three examples for actions (again, of your choice). Write executable code and test its correctness (either single program or several ones). To generate initial RDDs you can use code from lecture one or from Spark documentation. Submit as solution the source code and results of program runs.

Exercise 3 **(3 points)**

Watch the presentation about *Advanced Spark Features* by Matei Zaharia[1] and answer the following questions:

a) What does broadcast provide? Which other mechanism does it improve and how? Which features of the distributed program determine the number of times the variable will be actually transmitted over the network? Explain the role of tasks and nodes in this.

b) Describe the function of an accumulator. What is the alternative implementation without an accumulator and why is an accumulator a preferred option? Which example application for an acculumator is discussed in the video? What has to be implemented in order to define a custom accumulator? Compare the accumulator mechanism to the `reduce()`-function.

[1] *Advanced Spark Features* by Matei Zaharia: Video

c) Give three examples of RDD operators that result in RDDs with partitioning. Explain the connection between partitioning and network traffic. How does the modification on the PageRank example use partitioning to make the code more efficient? How does Spark exploit the knowledge about the partitioning to save time in task execution? How can you create a custom Partitioner?

Exercise 4 (4 points)

Read the relevant parts of the exercise by Databricks[2] (also download the "adult"[3] dataset). Use PySpark and DataFrames API (but no Spark SQL) to implement the following queries. Submit as your solution the source code and results of program runs.

a) Import "adult" dataset into a dataframe by reading-in RDDs first, and then converting RDDs to dataframes with suitable column names. If necessary, clean up this data so that you can execute the following queries on the dataframe.

b) Compute the ratio of males for each type of `marital_status`. Please also consider the types of `marital_status` with 0 males.

c) Compute the average `hours_per_week` of females who have `income` greater than 50K for each `native_country`.

d) Get the highest and lowest level of `education` for each group of `income`. The highest level of `education` is the level with highest value of `education_num`. To simplify, you can use Python dictionary to translate from `education_num` to `education` for displaying results (derive values from data). The result should be displayed similarly to the table below:

`income`	`highest_education`	`lowest_education`
`<=50K`	<type of `education`>	…
`>50K`	…	…

Exercise 5 (3 points)

Read the relevant parts of the example by Cory Maklin[4] and the relevant parts the exercise by Databricks[2] to perform the following tasks. Submit as your solution the source code and results of program runs. You can reuse the "adult"[??] dataset from the previous exercise. Note: remove the first line ("|1x3 Cross validator") of the file "adult.test" file to ensure that the code runs correctly.

a) In the example by Cory Maklin[4], extract the essential code from the PySpark section into one Python file. Then apply the solution to question 8.1 ("*Print out a table of `feature_name` and `feature_coefficient` from the Logistic Regression model*") from the exercise by Databricks[2] to the Logistic Regression model from Cory Maklin's example.

b) Extend the above code such that it saves the Logistic Regression model trained in **a)** to disk. Also create another Python file which loads this model from disk. Hint: read relevant section(s) in the book *Spark: The Definitive Guide* by Matei Zaharia & Bill Chambers, 2017[5].

[2]Databricks: Spark DF, SQL, ML Exercise
[3]UCI Machine Learning Repository: adult dataset
[4]Cory Maklin, Towards Data Science: Spark MLlib Python Example - Machine Learning At Scale
[5]*Heidi record, click on "Zum Volltext"*: Link