



VRIJE  
UNIVERSITEIT  
BRUSSEL



# BIG DATA PROCESSING REPORT

Gérard Lichtert  
0557513  
[gerard.lichtert@vub.be](mailto:gerard.lichtert@vub.be)

30 december 2024

Sciences and Bio-Engineering Sciences

# Inhoudsopgave

<b>1</b>	<b>Design decisions</b>	<b>2</b>
<b>2</b>	<b>GMM</b>	<b>2</b>
<b>3</b>	<b>Benchmarking</b>	<b>2</b>
3.1	Local machine . . . . .	2
<b>4</b>	<b>Questions</b>	<b>2</b>
4.1	Have you persisted some of your intermediate results? Can you think of why persisting your data in memory may be helpful for this algorithm? . . . . .	2
4.2	In which parts of your implementation have you used partitioning? Did it help for performance? If so, why? . . . . .	3
4.3	Did you use RDDs, DataFrames or Datasets for your implementation? Why? . . .	3

# 1 Design decisions

To implement this assignment I used the sample dataset as an example and I started implementing the required algorithm. I started off using RDD's and followed the Lab sessions' examples as well as the pseudocode left in the assignment description. When deciding when to persist an RDD I paid notice to when the RDD is reused, if it is reused. For example, in the initial reading of the txt file, the RDD is reused to compute the initial as well as the updated weights, means and variances. It is also used to compute the gammas with each set of weights, variances and means. This means it is reused a lot and thus beneficial to be persisted. I also decided to persist the gamma RDD because it is also used to compute the new weights, variances and means. Lastly I also decided to pre-compute the sum of the gamma RDD so there's no need to recompute it when using it during the recomputation of the weights, variances and means.

## 2 GMM

## 3 Benchmarking

### 3.1 Local machine

Initial setup:  $K = 3$ ,  $\text{Epsilon} = 0,01$  Elapsed time: 00h 00m 34s 958ms 000849800ns Weights: 0.30135819500681843, 0.3190340426348462, 0.37960776235833354 Means: 168.5851121666961, 163.85208936111448, 179.82230525830337 Variances 54.039394803851245, 32.951466889706, 52.863466362238

Setup:  $K = 5$ ,  $\text{Epsilon} = 0,01$  Elapsed time: 00h 01m 09s 774ms 000631000ns Weights: 0.1589171892439168, 0.24175949298891342, 0.364581200817622, 0.08930261876976954, 0.14543949817978694 Means: 164.74029436057168, 170.48954064721514, 179.6908146382836, 162.65908846246577, 164.36757418803285 Variances 33.4559895779154, 61.8866268633007, 54.48191424245265, 33.963078614790966, 33.395280960298955

## 4 Questions

### 4.1 Have you persisted some of your intermediate results? Can you think of why persisting your data in memory may be helpful for this algorithm?

Yes I persisted some of my intermediate results. As mentioned in 1 I persisted the initial RDD resulted from reading the txt file. I also persisted the RDD as a result from applying the gamma function to it. As mentioned in class, apache spark lazily evaluates the operations done on the RDD's. Which means that when an eager operation is invoked, all the previous lazy operations get applied to it's latest saved state. When a lazy operation is invoked it just remembers the transformation and waits till an eager operation is called. Consequently if its latest state is far away, it has to recompute a lot. This means that if a intermediate result is reused in multiple results derived from the intermediate results, it is beneficial to persist that intermediate result, to not recompute it in each result derived from it. Testing it with and without the persistence, showed that persisting the intermediate results yielded a big speedup. For example going from 4 minutes to 1 minute with a setup of  $K = 3$  and  $\text{epsilon} = 0,01$

#### **4.2 In which parts of your implementation have you used partitioning? Did it help for performance? If so, why?**

None, I did not make use of partitioning. I figured since the dataset is not categorised or requires 'grouping' partitioning is not needed. However, afterwards I thought perhaps I could have partitioned the dataset to group identical values together in the same partition to help with computing the GMM. Perhaps instead of summing the results of the algorithm I could have applied the algorithm once to a value in the partition and then multiplied it by the amount of occurrences of the value in the partition and then sum the results. Regardless, I did not trust my math skills enough to try this.

#### **4.3 Did you use RDDs, DataFrames or Datasets for your implementation? Why?**

Truth be told I just followed the pseudocode where they used RDD's and did not know I had to 'defend' my choice afterwards. I thought I was free to use whichever I felt comfortable with using as the structure of the dataset did not prompt me to use a specific one. I knew DataFrames and DataSets were more specialized for structured data, but since the dataset just had one column of numbers, I felt like all of the classes could be used. Come to think of it, I did look up the other classes afterwards and thought it might have been better to use the DataFrames class, since it's more optimized than RDD's, according to the apache spark website.