VRIJE
UNIVERSITEIT
BRUSSEL

BIG DATA PROCESSING REPORT

Gérard Lichtert
0557513
gerard.lichtert@vub.be

31st December 2024

Sciences and Bio-Engineering Sciences

Contents

1 Design decisions

To implement this assignment I used the sample dataset as a starting point. I started implementing the required algorithm using RDD's as the pseudocode used RDD's. Whilst I could have also used the other classes, it seemed overkill to use the other classes as the dataset was just a single column of numbers.

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 Benchmarking

2.1 Local machine

When setting up the experiment, I had to choose a K for the amount of gaussians and an Epsilon, which denoted the precision of the results. When doing the experiments, I repeated the experiments a hundred times and took the average elapsed time as well as the average of the results. Switching these values up yielded the following:

Setup: $K = 1$, $\epsilon = 0,01$
Elapsed time: 00h 00m 00s 328ms 000663820ns

Weights:	1
Means:	171.34
Variances:	94.35

Setup: $K = 1$, $\epsilon = 0,001$
Elapsed time: 00h 00m 00s 316ms 000859734ns

Weights:	1
Means:	171.34
Variances:	94.35

Setup: $K = 3$, $\epsilon = 0,01$
Elapsed time: 00h 00m 34s 958ms 000849800ns

Weights:	0.33	0.33	0.34
Means:	172.92	172.31	171.25
Variances:	50.25	52.45	48.34

Weights:	0.30	0.31	0.37
Means:	168.58	163.85	179.82
Variances:	54.03	32.95	52.86

Setup: $K = 3$, $\epsilon = 0,001$ Elapsed time: 00h 02m 05s 487ms 000494461ns

Setup: $K = 5$, $\epsilon = 0,01$
Elapsed time: 00h 00m 51s 443ms 000337634ns

Weights:	0.20	0.20	0.20	0.21	0.19
Means:	170.36	171.29	171.24	171.82	172.85
Variances:	50.59	50.62	49.12	47.38	52.67

Setup: $K = 5$, $\epsilon = 0,01$ In all cases, it meant that when the Epsilon is smaller, more iterations are required to find results. When the chosen K is higher, this also means that the array of results is higher, which means more computations must be done to update the weights, variances and means. Both of which increase the elapsed time.

After testing different starting values, I also tested my implementation by removing or adding persists on RDD's. I found that removing persists on the intermediate results described in 1 leads to a lot longer execution times, especially when having a smaller Epsilon or higher K.

I must admit, I tried this experiment only a few times because it just took too long and it was already mentioned in class that not persisting intermediate results was very inefficient.

3 Questions

3.1 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 persistance, 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 Epsilon = 0,01

3.2 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 outside the default 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 occurences of the value in the partition and then sum the results. Regardless, I did not trust my math skills enough to try this.

3.3 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.