Big Data Computing - HW 2 - G24

Description of the generator file and table

May 26, 2025

Jianan E 2144167
Gianfranco Mauro 2160477
Davide Volpi 2140728

1 HDFS Results

The goal of this test is to assess the scalability of the standard and fair implementations. The test was performed on file `artificial4M7D100K.txt` with the following parameters: $L = 16, K = 100, M = 10$.
In the following table, we report our running times (in milliseconds).

SCALABILITY WITH RESPECT TO NUMBER OF EXECUTORS			
Number of executors	**Spark's Lloyd's implementation**	**MRFair Lloyd's**	**MRComputeFairObjective**
2	22808	1237060	24902
4	11016	578262	12865
8	6424	301370	6832
16	3224	150972	3393

2 Points Generation Function

The main idea behind the `points_generation` function is to create a scenario with:

- K vertically stacked and well separated clusters.
- Inside each main cluster, split the points into two "sub-clusters" spatially separated and with distinct labels ("A" on the right and "B" on the left) but with an inside distance much smaller than the vertically stacked clusters.

In this way, we can catch the difference between the standard Lloyd's implementation and the Fair Lloyd's one. We expect to observe that the standard Lloyd's algorithm will focus on minimizing the geometric part of the problem, while on the other hand, the Fair algorithm will go beyond that part and will attempt to mitigate the unfairness in the classification.
The function will take as input $N(int)$ points and $K(int)$ centers.

Algorithm 1 points_generation(N, K)

Require: $N \geq 10,\ 1 \leq K < N$

Initialize empty lists X, Y, L ▷ storage for coordinates & labels
$\sigma \leftarrow 1,\ d_x \leftarrow 10\sigma,\ d_y \leftarrow 100\sigma$ ▷ base std. dev. & offsets
$b \leftarrow \lfloor N/K \rfloor,\ r \leftarrow N \bmod K$ ▷ evenly distribute points
for $i = 1, \ldots, K$ **do**
  $m \leftarrow b + [i \leq r]$ ▷ adjust for remainder
  $n_A \leftarrow \text{round}(0.1\, m),\ n_B \leftarrow m - n_A$ ▷ 10% label A, 90% label B
  $y_c \leftarrow (i-1)d_y,\ x_B \leftarrow -d_x/2,\ x_A \leftarrow d_x/2$ ▷ cluster centers
  Draw n_B pts $\sim \mathcal{N}((x_B, y_c), (\sigma/3)^2)$, label B ▷ generate B subcluster
  Draw n_A pts $\sim \mathcal{N}((x_A, y_c), (\sigma/3)^2)$, label A ▷ generate A subcluster
  Append new points to X, Y, L ▷ accumulate results
end for
return DataFrame(X, Y, L)

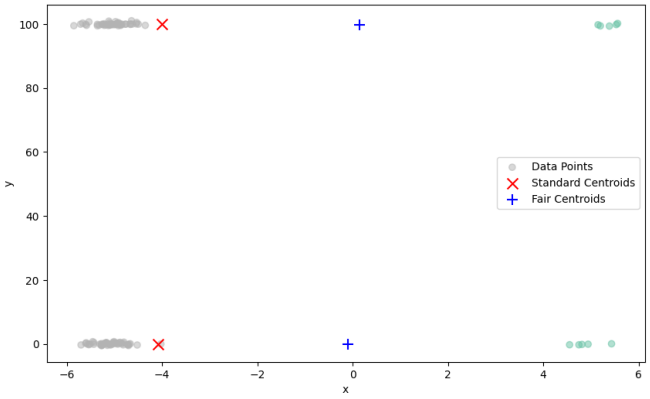

Figure 1: Example of an iteration of the `points_generation` function with $N = 100$ and $K = 2$