Streaming Algorithms for k-center Clustering with Outliers

Gabriel Beauplet

Graph Mining
15 février 2019

Problem

Given n points and distances between every pair of points, the k-center problem consists in choosing k points to place centers such that the maximum distance of a point to a center is minimized. We present in this report the implementation of two algorithms for this problem. Both algorithm are α approximmation-algorithm, meaning the objective function at the end of these algorithms is at most α times the optimal objective function [1].
All the algorithms are coded in Python on a i5-5300U CPU @ 2.30GHz machine with 16Go of RAM. You can find the code at this github page : `https://github.com/beaupletga/Streaming_Algorithms_for_k-center_Clustering_with_Outliers`

Question 1 : Static Algorithm for k-Center Clustering

This first algorithm [1] is a 3-approximation algorithm, and works with any metrics. This algorithm is static, meaning it needs to put in memory all the points at the same time. This approach works well when the number of points is limited but becomes infeasible as soon as the number of points is too big to put them all in memory at the same time.

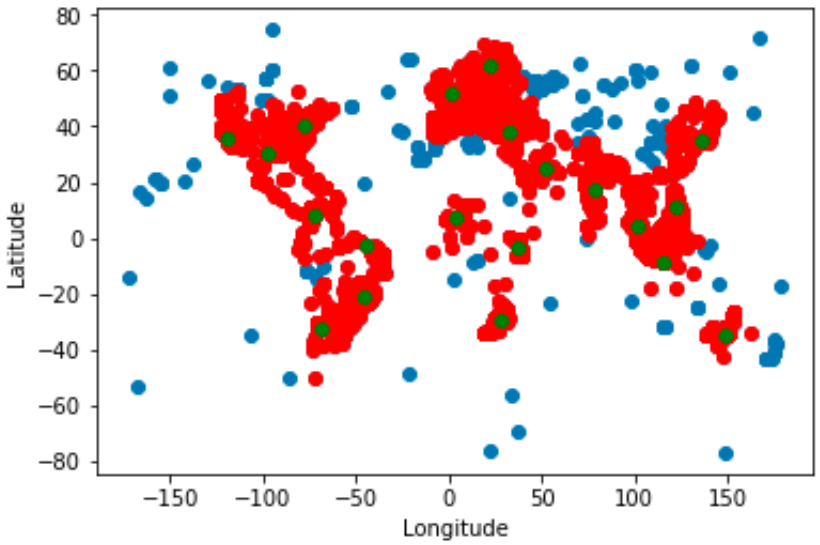

FIGURE 1 – k-Center Clustering using the static algorithm and $r = 5.5$ on a subset of the twitter dataset. red : covered points, blue : uncovered points, green : centers points

1. Because we are in the minimization case

Questions 2,3 : Streaming Algorithm for k-Center Clustering

Evaluate the Approximation Ratio

This second algorithm [2] is also an algorithm for the k-center clustering problem. This time, it's a streaming algorithm which only stores $O(kz)$ points in memory and takes into account z outliers which will be ignored for the clustering. It achieves a $(4+\epsilon)$-approximation to the optimal clustering radius.

Results of the Streaming Algorithm

The results of the streaming algoritm when k=20, z=10 are not good at all (fig 2). The final radius is high and thus almost all the areas of the clusters overlap between eachother.

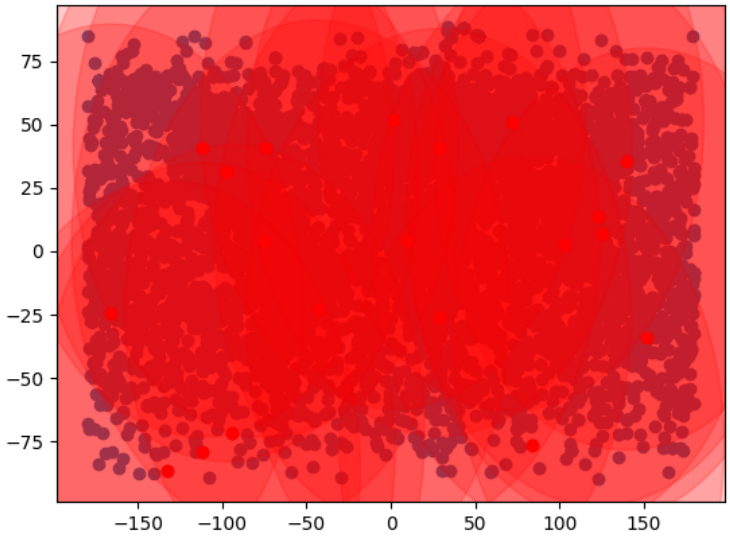

FIGURE 2 – Result of the k-center clustering algorithm for k=20 and z=10. The algorithm finds 20 centers, each of them with a radius $r = 114.04$.

In the instructions of the exercice, it's written : "To this end, you should try to give an upper bound on the optimum solution". And several lines below : "You should also discuss how you computed a lower bound on the optimum solution". I guess, the second one is the valid one but I did the first one before realizing it. In the following, I first propose a way to compute this upper bound and then the lower bound.

Upper bound

A naïve approach would be to cover the range of the dataset by squares :

1. Compute the minimum and maximum of both axes (the dataset has only two columns) in order to get the enclosing rectangle
2. Find the minimum length l to cover the enclosing rectangle by k squares of length l
3. Fix a circle at the center of each square of radius $r' = \sqrt{2}\frac{l}{2}$

This algorithm works because any square of length l can be covered by a circle of radius $\sqrt{2}l$. In this case, we are able to cover the enclosing rectangle without taking care of the outliers with 18 squares of length 60 (fig 3) and hence 18 circles of radius $r = \frac{60}{2}\sqrt{2} \simeq 42.42$ (fig 4).
Allowing z outliers will only reduces the radius of each circle. Then we found an upper bound (not tight at all) for the optimal solution : $r = 42.42$.

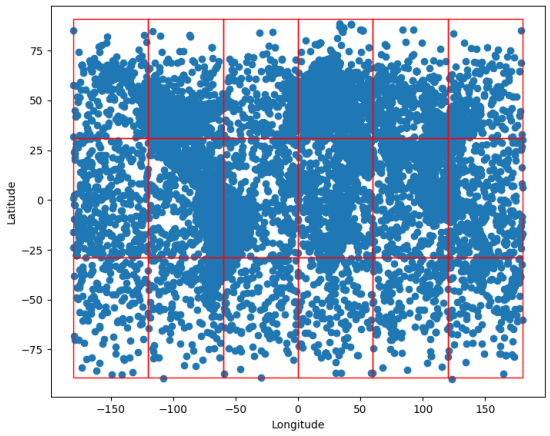

FIGURE 3 – Covering the dataset with squares of length $l = 60$ (it looks like rectangles because both axis don't have the same scale

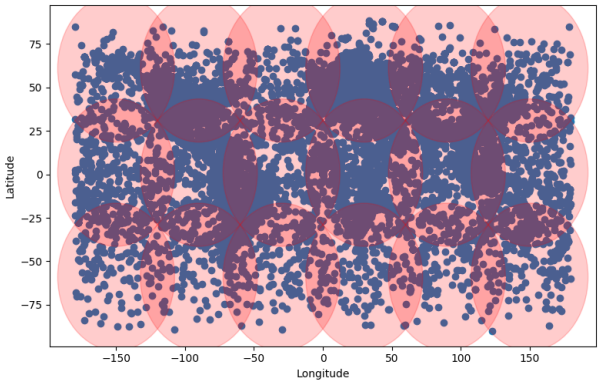

FIGURE 4 – Covering the dataset using circles of radius $r' = 30\sqrt{2}$

The approximation ratio is $A = \frac{\text{Radius Streaming Algorithm}}{\text{Radius from Upper Bound to the Optimal Solution}} = \frac{114.04}{42.42} = 2.68$. This is actually a lower bound on the approximation ratio of the algorithm.

Lower Bound

A simple way to compute the lower bound for this problem is to compute a set of k+z+1 points which have the biggest distances between eachother. Indeed, if the minimum distance between k+z+1 points is d then the algorithm won't be able to have a better solution because as most z of these points will be outliers and the other ones will have to belong some cluster.
Thus a simple algorithm is :

1. Choose n points (the more the better), centers=$\emptyset$
2. Compute the distances between these points and add the 2 furthest points as centers
3. Compute the distances between the points and the centers
4. Add to the centers the point which is the furthest away from the centers
5. Go back to step 3 until k+z+1 points have been added to the centers

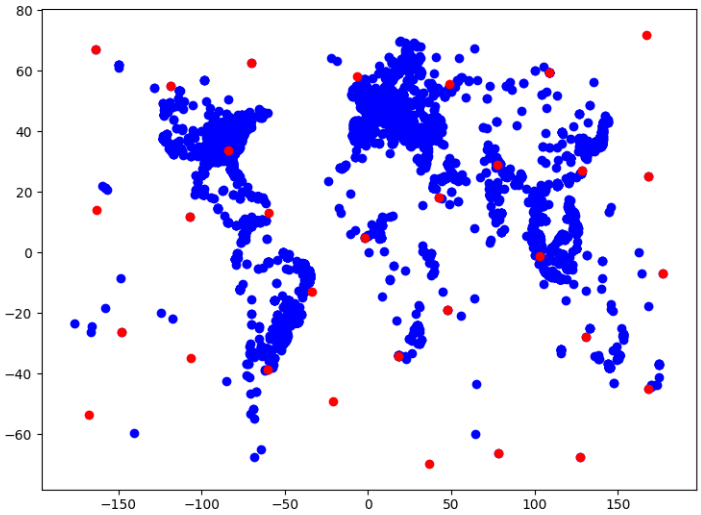

FIGURE 5 – k+z+1-centers with the above algorithm when k=20, z=10. Red :centers,blue :dataset. The minimum distance between all the centers is 33.65.

In the first part, we found a clustering with a radius $r = 114$. We know from fig 5 that the radius of an optimal solution can't be lower than 33.65. Then the approximation ratio is $\leq \frac{114}{33.65} = 3.38$. Thus, the twitter data seems to not be far from the worst case scenario ($O(4 + \epsilon)$).

Evaluate the Streaming Algorithm Depending on z

Approximation Ratio

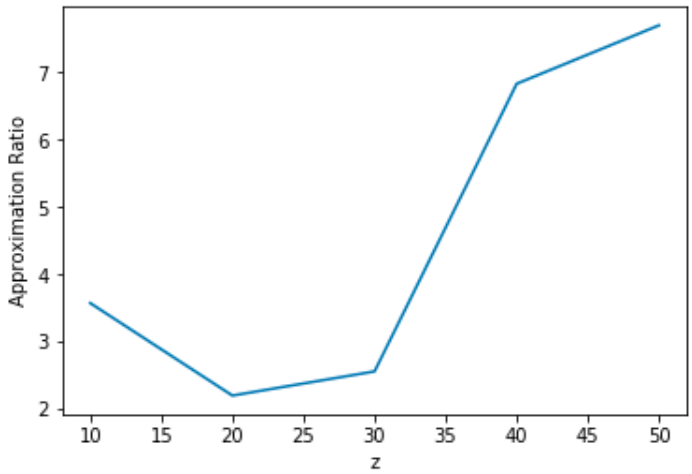

FIGURE 6 – Evolution of the Approximation ratio depending on the value of z={10,20,30,40,50}

The shape of the approximation ratio (fig 6) is strange because allowing more outliers should reduces the value of the radius. Two hypothesis could explain this behaviour :

1. The lower bound on the optimal radius decreases faster than the radius of the streaming algorithm
2. The streaming algorithm doesn't performs well when the batch size is too high

It's actually a bit of those two hypothesis.
Indeed we can see of the fig 7 that the lower bound onto the optimal radius decreases quickly whereas the radius of the streaming algorithm doesn't have a special trend.
Moreover, the batch size is dependent on the number of outliers (because the batch size is $k + z$). At the begining of the algorithm, we compute the value of the inital radius r as the mininmum distance of the first batch divides by 2. We observe on fig 8 that freezing the initial value of r enables the streaming algorithm to decreases its radius when the number of allowed outliers increases (which is the natural behaviour).

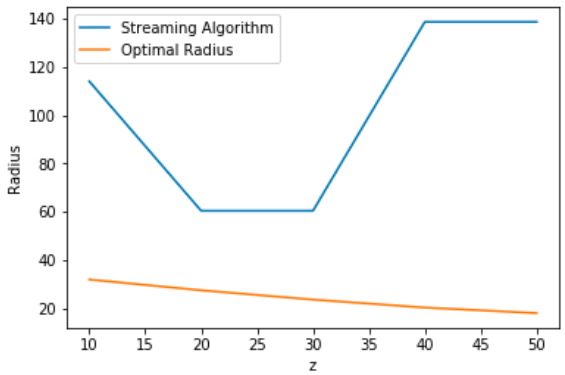

FIGURE 7 – Evolution of the radius of the streaming algorithm and the lower bound of the optimal radius depending on the value of z={10,20,30,40,50}

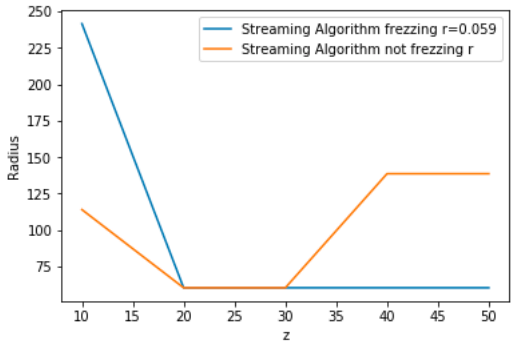

FIGURE 8 – Evolution of the radius depending on the value of z when freezing the initial value of the radius $r = 0.059$ or when not freezing r

0.0.1 Time

The time spent by the algorithm is quite high because the algorithms are coded in Python (which is a slow language). Moreover we observe by comparing the fig 7 and fig 9, that the time spend by the algorithm is negatively correlated with the value of the radius at the end of the algorithm.

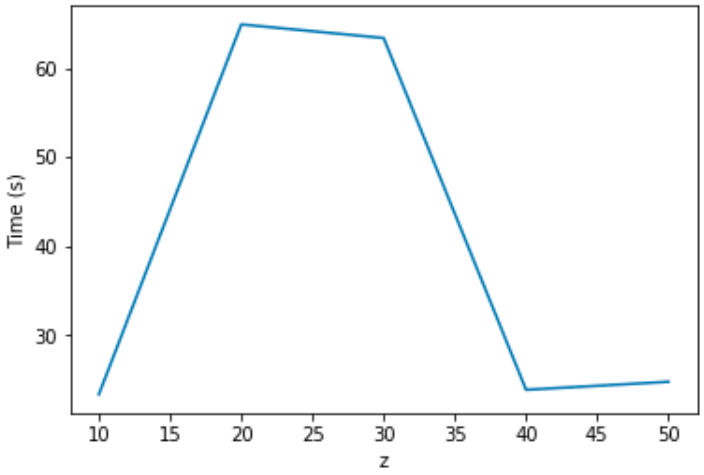

FIGURE 9 – Evolution of the time spent by the algorithm depending on the number of outliers z={10,20,30,40,50}

Conclusion

This report is a bit longer than expected but some things needed to be explained. The static algorithm in the first part performs quite well but is limited by the memory of the computer. On an another side, the streaming algorithm is memory efficient but we can't say that it performs well because tuning some parameters may change totally the value of the final radius (ex : the initial value of r). However, we can minimize these imperfections by reducing the value of α, even if the time spent by the algorithm will be longer. Finally, I accidently computed an upper bound for the optimal radius and it appears to be not bad at all and easy to compute (its radius is 42.42 when the optimal one can't be lower than 33.65).

Références

[1] Moses Charikar, Samir Khuller, David M. Mount, and Giri Narasimhan,*Algorithms for facility location problems with outliers.*

[2] Richard Matthew McCutchen and Samir Khuller,*Streaming algorithms for k-center clustering with outliers and with anonymity.*