Monte Carlo Simulation of DBSCAN Parameter Effects on LiDAR Point Cloud

Mohamad Nweder
Department of Computer Engineering,
University *West, Trollhättan,*
Sweden (m.neweder@gmail.com)

***Abstract*—This article presents a Monte Carlo simulation-based study on how DBSCAN parameters *epsilon* and *min_samples* affect LiDAR point-cloud clustering. The simulation was performed by randomly varying these parameters over 2000 iterations, resulting in cluster counts ranging from X-Y and measuring cluster count, cluster size and noise ratio. The results demonstrate that DBSCAN is sensitive to parameter changes and that simulation can be a useful method to evaluate and analyze algorithm behavior without relying on real-world experiments.**

***Index Terms*—monte carlo, dbscan, lidar, clustering, simulation.**

I. Introduction

Simulation methods are often used to understand important tools used today, as they save both money and time. Simulation is often applied to study algorithm behavior, control conditions and observe how algorithms respond to different parameters as input without requiring real-world experiments.

Methods used in simulation are applied in research to study and observe algorithm behavior under complex or uncertain conditions [1], [2]. In this experiment, a Monte Carlo Simulation was used to study the behavior of the DBSCAN clustering algorithm when its parameters change, using LiDAR point-cloud data obtained from PhD research work conducted at Luleå University of Technology (LTU) academic studies on DBSCAN under the supervision of Professor Ramin Karim [1]. DBSCAN algorithm is used to find clusters of varying density and has shown better performance than CLARNAS in benchmark tests [2]. The simulation executed DBSCAN repeatedly while randomly varying the parameters (*epsilon* and *min_samples)* within predefined intervals. Each simulation run generated different values representing number of clusters, size of cluster the largest cluster and ratio of noise points.

Previous studies have discussed DBSCAN parameters sensitivity [2] and the use of Monte Carlo methods for algorithm evaluation [3]. The main goal of this simulation is to demonstrate how Monte Carlo simulation method can be used effectively to examine the sensitivity of changing parameters in DBSCAN algorithm without performing any physical or real-world experiments.

II. Methodology

The simulation was developed to demonstrate how DBSCAN algorithm reacts when its parameters change repeatedly while running the code. Experiment was performed in Python using different libraries such as NumPy, Pandas.

A. Simulation Steps

Simulation followed the Monte Carlo principle, which means that the same processes get repeated lot of times with different randomized data.

Every iteration was executed by DBSCAN algorithm with random parameters value:

- Epsilon (ε): Radius defines how close points must be to be considered neighbors.
- min_samples: Minimum number of core points required to form a cluster.

Both parameters randomized between $\varepsilon = 0.05 – 0.25$ and *min_samples* =3 – 7. These parameters are chosen based on typical values used in previous A Density-Based Algorithm for Discovering Clusters experiments [2] and on the heuristic method described in section 4.2 of Ester et al. (1996).

Each running saved three results :

1- Number of founded cluster.

2- Largest cluster size.

3- Point parts that classed as cluster

B. Material of data

The data used in this simulation originated from LiDAR point-cloud data conducted as PhD research at Luleå University of Technology (LTU) under the supervision of Professor Ramin Karim [1].

The dataset was available to students participating in a course project connected to doctoral research. it includes spatial coordinates from laser scanning of equipment surfaces and has been used in several studies related to Industrial AI and predictive maintenance. Each datapoint had three coordinates (X,Y,Z). Ground level is calculated with a histogram that shows over Z-values to filter underground points. After filtering, the experiment will used only the above ground points.

C. Implementation

A total 2000 iteration were executed using DBSCAN algorithm with different random parameters values. A fixed random seed was ensuring reproducibility across all iterations. The simulation was implemented in Python using NumPy, Pandas, Matplotlib, and scikit-learn, DBSCAN and NearestNeighbors modules with standard modules os, sys, and Json for data handling and management.
The output from each iteration were automatically save for analysis, including overall static and summery data. Figures were generated showing histogram, cumulative distribution, convergent and sensitive measurement for all runed simulations.
Data saved in CSV and JSON format are available in the appendix.

III. Simulation Results

Simulation was performed with a total of 2000 iterations using DBSCAN algorithm. Parameters *epsilon (ε) and min_samples* were randomized between given intervals (ε ∈ [0.05, 0.25], min_samples ∈ [3, 7]). Each iteration saved the results locally in two different format, JSON and CSV. This allows experiments opportunity to verify all runs and tests correctly and to present them as figures.

Four main figures will be presented below to exemplify the results of simulation.

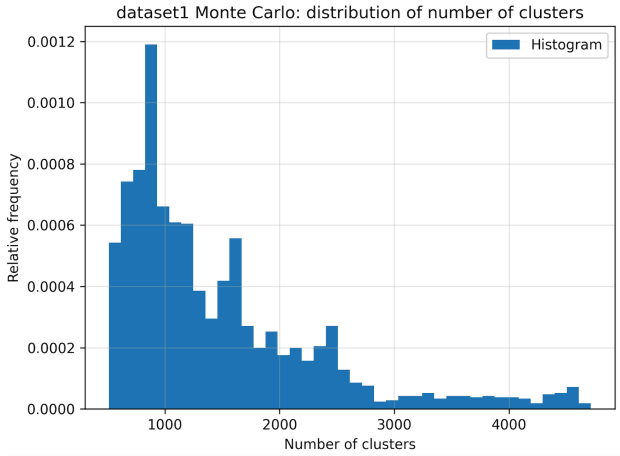

Fig. 1. Histogram of cluster counts.

Histogram in Fig. 1 shows how many cluster that founded between different iterations.

As shown in Fig. 1. The X-axis represent the numbers of clusters detected by DBSCAN in each execute and Y-axis represent the relative frequency of those cluster counts across all iterations.

The distribution stretches from the lower value (fewer cluster) to higher values (many small cluster), showing the random parameter variation worked correctly. Demonstrates that the random variations values of parameters worked correctly and that simulation performed as planned. Most of runs resulted in values within the middle.

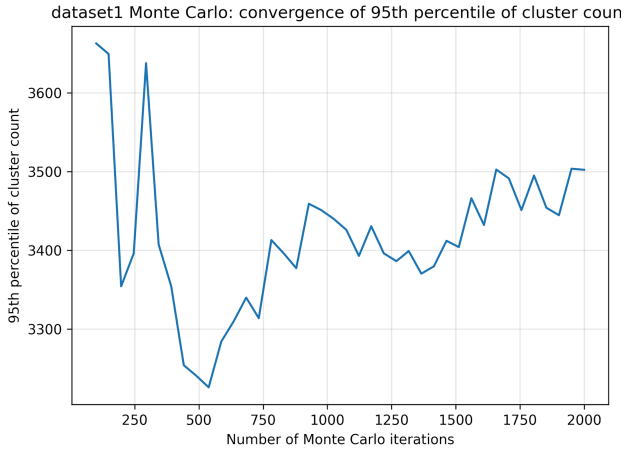

Fig 2. Convergence of cluster count (95th percentile).

As shown in Fig. 2 (calculated as the 95th percentile) , explain how the number of clusters changes and stabilizes as the number of Monto Carlo iterations increases.

The X-axis shows the number of iteration and the Y-axis shows the 96th percental of cluster count.

It also shows that the result become stable after many iterations, which proves that the Carlo simulation has reached convergence around 2000 iteration.

This confirms that the model works as expected and it stabilized around on 2000 iterations in result.

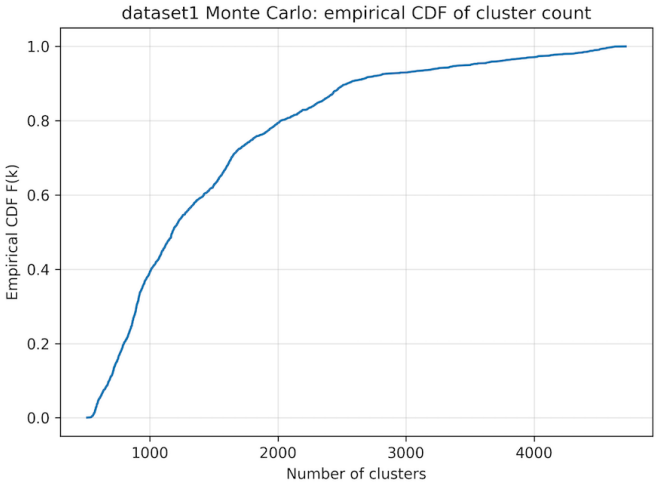

Fig. 3. Empirical cumulative distribution (ECDF) of cluster count.

As shown in Fig. 3. The ECDF represent the cumulative probability that the number of clusters is below a given value, it demonstrates a stable and continuous distribution across all Monte Carlo runs

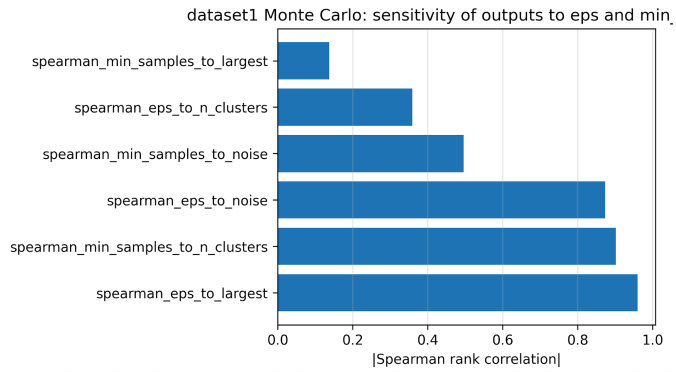

Fig. 4. Sensitivity analysis using Spearman correlation

As fig. 4 prove the sensitivity of outputs to e *epsilon* and also *min_samples* based on spear rank correlation.

The result of fig.4 proved has the strongest influence on the number of clusters, confirming it as the dominant DBSCAN parameter.

On average and depending on our *epsilon* and also *min_samples* on DBSCAN produced between 800 and 3500 clusters per iteration.

Variations decreased after about 2000 iterations. which approved that the model achieved convergence and produced statistically stable results.

IV. DISCUSSION

The simulation demonstrated that the Monte Carlo method is useful for testing the sensitivity of parameter changes in DBSCAN without performing any real physical experiments. The results demonstrated that the algorithm reacts strongly to changes in *epsilon (ε)* and *min_samples*. Small parameters changes had a significant effect on how many clusters were formed and their sizes.

After about 2000 iterations, the result become stable and reproducible, demonstrating that the model reached convergence and produced consistent through all iterations. Different data could give other results depending on noise level or density.

Future work could compare this simulation with other clustering algorithms such as OPTICS or K-Means to study how their behavior under similar conditions.

V. CONCLUSION

The simulation demonstrated that the Monte Carlo method is effective for studying how DBSCAN reacts to parameter changes.
Epsilon(ε) had the strongest effect on cluster number while *min_samples* had a smaller impact.
After around 2000 iterations, the results became stable and reproducible, demonstrating that the simulation worked as expected and planned.
Only one LiDAR dataset was used in this experiment which could limit the generality of the results.
This type of experiment can be used in future studies to obtain results faster without physical experiments and to test other datasets or algorithms.

REFERENCES

[1] Luleå University of Technology, *LiDAR point-cloud data from PhD research supervised by Prof. Ramin Karim*, Luleå, Sweden, 2025.

[2] J. Ester, M. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in Proc. KDD, 1996, pp. 226–231. [Online]. Available : https://file.biolab.si/papers/1996-DBSCAN-KDD.pdf

[3] N. Metropolis and S. Ulam, "The Monte Carlo method," Journal of the American Statistical Association, vol. 44, no. 247, pp. 335–341, 1949. [Online]. Available: https://web.williams.edu/Mathematics/sjmiller/public_html/105Sp10/handouts/MetropolisUlam_TheMonteCarloMethod.pdf .

GITHUB

https://github.com/Nweder/DTA400