Incremental Density-Based Clustering with Grid Partitioning (Student Abstract)

Jeong-Hun Kim[*1], **Tserenpurev Chuluunsaikhan**[*1], **Jong-Hyeok Choi**[†2], **Aziz Nasridinov**[†1]

[1] Department of Computer Science, Chungbuk National University, Cheongju, 28644, South Korea, +82-43-261-3597
[2] Bigdata Research Institute, Chungbuk National University, Cheongju, 28644, South Korea, +82-43-261-3597
{etyanue, teo, leopard, aziz}@chungbuk.ac.kr

Abstract

DBSCAN is widely used in various fields, but it requires computational costs similar to those of re-clustering from scratch to update clusters when new data is inserted. To solve this, we propose an incremental density-based clustering method that rapidly updates clusters by identifying in advance regions where cluster updates will occur. Also, through extensive experiments, we show that our method provides clustering results similar to those of DBSCAN.

Introduction

DBSCAN (Ester et al. 1996) is a well-known density-based clustering method for finding arbitrary shape clusters as well as for detecting outliers. In DBSCAN, a cluster is defined as a set of dense objects separated from other clusters by sparse regions, and an object is dense if it has more than μ neighbors within an ϵ radius. The adjacent dense objects propagate the same label to their neighbors. However, DBSCAN performs re-clustering from scratch when new data is inserted. In particular, label propagation caused by new data insertion degrades cluster update performance because it requires executing ϵ-range queries on many other objects (Ouyang and Shen 2022). For example, neighbors to be directly updated by a new object x_p can be easily identified using an ϵ-range query. However, these neighbors propagate labels to their neighbors again, which causes the ϵ-range query to be repetitively executed until no more updates occur. The key to an efficient way is to reduce the complexity of the ϵ-range query and the number of its executions.

To this end, some researchers have attempted to improve the cluster update performance by identifying objects to be updated in advance or by utilizing a data structure in which density information is summarized (Ester et al. 1998; Mai et al. 2020). Such approaches often yield approximate results that differ significantly from those of DBSCAN, with computational costs similar to those of re-clustering from scratch in the worst case.

To address the cluster update issue, we propose an incremental DBSCAN (iDBSCAN) that efficiently updates clusters based on grid partitioning, reducing processing time of the ϵ-range query. Experimental results show that our method achieves cluster update performance up to 163.3 times faster than that of DBSCAN.

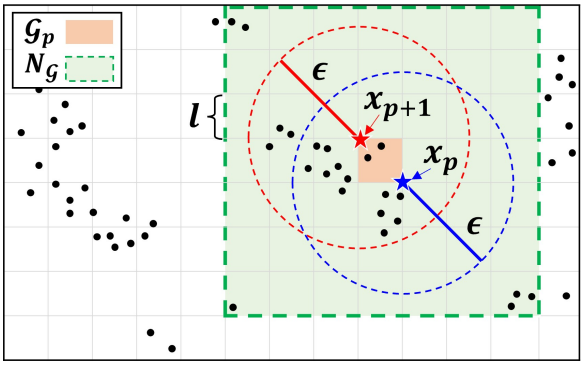

Figure 1: An example of the update localization.

Proposed Method

We describe the proposed method by focusing on cluster updates by a new object, assuming that DBSCAN has been performed. We divide the proposed method into three steps: grid partitioning, update localization, and cluster update.

Grid Partitioning We first partition the data space into grids to build a spatial index. Based on the grids, we accelerate the ϵ-range query by considering only objects in grids adjacent to a target object. Let X be a set of n objects $\{x_1, ..., x_n\}$ in the d-dimensional space, normalized from zero to one. The grid length l is a maximum value among $\{1/2^\rho\}_{\rho=0}^{\infty}$ less than $\epsilon/\sqrt{d}$. Each grid has a unique key, the combination of the grid orders for each dimension.

Update Localization We localize cluster updates by identifying grids adjacent to a new object $x_p = \{x_{p1}, ..., x_{pd}\}$. To this end, we first obtain the grid orders of x_p for each dimension through $s_i = \lfloor x_{pi}/l \rfloor (1 \leq i \leq d)$. Hereafter, we assemble the grid orders to find a grid $\mathcal{G}_p$ containing x_p. Then, we identify the adjacent grid set $N_\mathcal{G}$ included in the $\pm\lceil \epsilon/l \rceil$ range of the grid orders for $\mathcal{G}_p$. In other words, $N_\mathcal{G}$ is the grid set within the hyper-rectangular-shaped range surrounding $\mathcal{G}_p$, as shown in Figure 1. Additionally, we prune

[*]These authors contributed equally.
[†]The corresponding authors.

more grids in $N_{\mathcal{G}}$ that do not reach x_p, utilizing the minimum and maximum distances between grids to identify adjacent grids within the exact ϵ range. We calculate the minimum and maximum distances, $\mathcal{GD}_{min}$ and $\mathcal{GD}_{max}$, between grids as follows:

$$\mathcal{GD}_{min}(\mathcal{G}_o, \mathcal{G}_t) = \sqrt{\sum_{i,j \in \mathcal{G}_o, \mathcal{G}_t} ((|i-j|-1) * l)^2} \quad (1)$$

$$\mathcal{GD}_{max}(\mathcal{G}_o, \mathcal{G}_t) = \sqrt{\sum_{i,j \in \mathcal{G}_o, \mathcal{G}_t} ((|i-j|+1) * l)^2} \quad (2)$$

where i and j are the grid order for each dimension of grids $\mathcal{G}_o$ and $\mathcal{G}_t$, respectively. Considering the location of the grids, we select all $\mathcal{G}_x$ that satisfy $\mathcal{GD}_{min}(\mathcal{G}_p, \mathcal{G}_x) \leq \epsilon$ $(\mathcal{G}_x \in N_{\mathcal{G}})$ as the final adjacent grids; these are denoted by $\tilde{N_{\mathcal{G}}}$. In addition, all $\mathcal{G}_x$ satisfying $\mathcal{GD}_{max}(\mathcal{G}_p, \mathcal{G}_x) \leq \epsilon$ $(\mathcal{G}_x \in \tilde{N_{\mathcal{G}}})$ are fully-connected with $\mathcal{G}_p$.

Cluster Update We introduce an efficient cluster update process utilizing the adjacent grid set $\tilde{N_{\mathcal{G}}}$. First, we find reachable objects through an ϵ-range query on a new object x_p. Since these reachable objects are included in $\mathcal{G}_p$ and $\tilde{N_{\mathcal{G}}}$, we reduce the complexity of the ϵ-range query considering the specific grids and their objects only. In particular, objects included in grids of $\tilde{N_{\mathcal{G}}}$ that are fully-connected with $\mathcal{G}_p$ can be regarded as reachable objects without distance calculations. Hereafter, we determine the cluster label y_p of x_p based on the reachable objects. Let $R_{\mathcal{D}}$, $R_{\mathcal{B}}$, and $R_{\mathcal{O}}$ be sets of dense objects, border objects, and outliers among reachable objects of x_p, respectively; $\mathcal{C}$ is a set of cluster labels $\{1, ..., k\}$. We obtain the cluster label y_p as follows:

$$y_p = \begin{cases} \min_{x \in R_{\mathcal{D}}} y_x, & \text{if } R_{\mathcal{D}} \neq \emptyset \\ k+1, & \text{if } R_{\mathcal{D}} = \emptyset \text{ and } |R_{\mathcal{O}}| \geq \mu \\ outlier, & \text{otherwise} \end{cases} \quad (3)$$

Next, we update the clusters by performing label propagation using x_p. This label propagation does not occur when x_p is an $outlier$, and when x_p forms a new cluster, i.e., $y_p = k+1$, all objects in $R_{\mathcal{O}}$ are assigned to the $k+1$ cluster. On the other hand, when x_p is assigned to an existing cluster, x_p becomes a dense or border object. We assume that the cluster label y_p of x_p is $c \in \mathcal{C}$. If $|R_{\mathcal{D}}|_{y \geq c} + |R_{\mathcal{B}}|_{y \geq c} + |R_{\mathcal{O}}| \geq \mu$, then x_p is a dense object; otherwise it is a border object. When x_p is a dense object, we propagate the cluster label y_p to all objects in $R_{\mathcal{D}}$ and to their neighbors. On the other hand, when x_p is a border object, we traverse $\forall x \in R_{\mathcal{D},\mathcal{B},\mathcal{O}}$ in the order of cluster labels, and repeat the same process as x_p to propagate the cluster label y of each object x. At this time, since the ϵ-range query for the label propagation is executed only in $\forall x \in R$, not in the entire dataset, the number of queries becomes less than that of DBSCAN.

Experiments

We conduct experiments on three synthetic datasets with various shapes, scales (n), dimensions (d), and noises (η). Then, we compare the results of our method with those of DBSCAN to evaluate the cluster update performance. The synthetic datasets are moons, circles, and blobs, and we generate these datasets using the scikit-learn Python library. For each run of the experiments, we first perform DBSCAN on 90% of objects in a dataset and then insert the remaining 10% of objects sequentially. We utilize the same input parameters ϵ and μ for the proposed method and DBSCAN. We use running time (RT), mean update time (MUT), and clustering accuracy (ACC) as evaluation metrics. RT is the accumulated time for all cluster updates, MUT is the mean running time for a singular cluster update, and ACC is a ratio consistent with DBSCAN. The results are shown in Table 1. Our method outperforms DBSCAN by a significant margin on all datasets. These results indicate that our method can significantly reduce the complexity of the ϵ-range query and the number of its executions.

Dataset	Ours		DBSCAN		ACC
	RT (sec)	MUT (ms)	RT (sec)	MUT (ms)	(%)
circles ($n = 10k$)	0.84	0.84	66.10	66.10	99.9
circles ($n = 30k$)	7.76	1.94	924.00	308.00	99.9
blobs ($d = 3$)	8.30	16.60	38.03	76.06	97.7
blobs ($d = 5$)	14.44	28.88	43.26	86.52	97.6
moons ($\eta = 0.1$)	0.19	0.37	12.54	25.09	99.2
moons ($\eta = 0.2$)	0.15	0.30	12.25	24.49	98.7

Table 1: Clustering results on synthetic datasets. The default settings for each dataset are $n = 5k$, $d = 2$, and $\eta = 0.2$.

Conclusion and Future Work

This paper proposes iDBSCAN, which performs DBSCAN by efficiently updating clusters based on grid partitioning. In the future, we will expand the proposed method to update clusters simultaneously for large transactions.

Acknowledgements

This paper was funded by the Basic Science Research Program through the National Research Foundation of Korea (Grant No: 2021R1I1A3042145).

References

Ester, M.; Kriegel, H.-P.; Sander, J.; Wimmer, M.; and Xu, X. 1998. Incremental Clustering for Mining in a Data Warehousing Environment. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, 323–333. San Francisco, CA, USA.

Ester, M.; Kriegel, H.-P.; Sander, J.; Xu, X.; et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, 226–231.

Mai, S.; Jacobsen, J.; Amer-Yahia, S.; Spence, I.; Tran, P.; Assent, I.; and Nguyen, Q. V. H. 2020. Incremental density-based clustering on multicore processors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3): 1338–1356.

Ouyang, T.; and Shen, X. 2022. Online Structural Clustering Based on DBSCAN Extension with Granular Descriptors. *Information Sciences*, 607: 688–704.