Triangulation based Spatial Clustering for Adjacent Data with Various Density

October 24, 2023

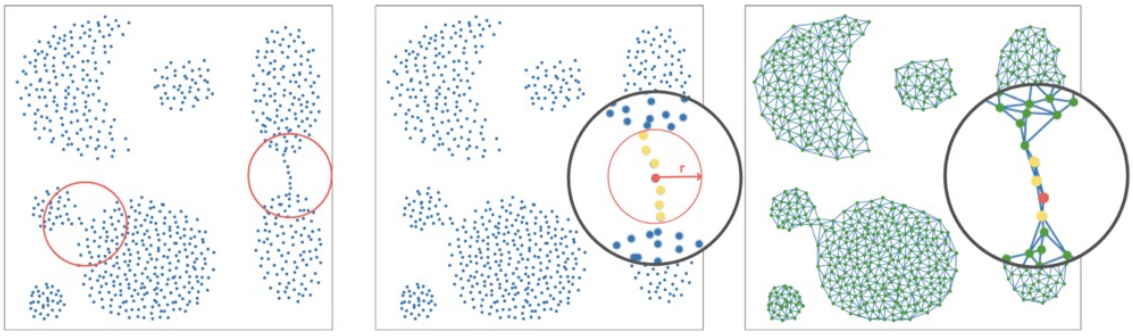
ABSTRACT

Data often exhibits nonlinear relationships and complex structures in many areas, such as geography, bio-informatics, and software engineering. Unfortunately, traditional clustering algorithms tend to identify ellipsoidal, spherical, or other regularly shaped clusters, but encounter difficulties when dealing with complex underlying groups that possess irregular shapes, various densities, and noisy connections between multiple clusters. To handle these complex groups, in this article, we formulate a graph-based, three-step Density and Triangulation based Clustering (DTC) framework: (1) *density-based separation* to create initial clusters based on kernel density estimation on each data point, (2) *Delaunay triangulation-based DBSCAN* to handle the nonlinear shape of data and the touching issue, and (3) *noises handling* according to closest neighbors. The effectiveness and practicability of the DTC are validated through both synthetic shape data and real data application. All the numerical studies demonstrate that the nested and adjacent clusters with disparate density can be separated with high accuracy and meaningful implications.

1 Introduction

Cluster analysis [1] is an unsupervised learning method that groups data points based on their inherent similarities or patterns without labeled data for training. The absence of group information makes this approach challenging but extremely valuable for discovering hidden structures and insights in various applications, such as in geography [2], bio-informatics [3], software engineering [4], and many more. As a specialized branch of cluster analysis, spatial clustering focuses on analyzing geospatial, remote sensing, and image data. Spatial Clustering could be particularly challenging while encountering spatial data with intricate structure, arbitrary shape and varying densities so that traditional clustering methods might not effectively capture its inherent patterns and relationships.

In response to the inability to detect non-globular clusters by traditional partition-based clustering approaches such as K-means [9], a variety of density-based clustering algorithms (DBCLAs) have been developed and became increasingly popular as they facilitated the identification of arbitrarily shaped clusters with varying densities. These DBCLAs all operate on the principle of estimating the density of a region and identifying higher-density areas in the data space as clusters. For instance, for a data point, the Density-based Spatial Clustering of Applications with Noise (DBSCAN) [6] uses the number of data points within a predetermined radius to evaluate the point's density. Then, the points residing in the high-density regions are grouped, while those in the low-density areas are labeled as outliers. This approach is able to identify clusters of arbitrary shapes, filter out noise, and address certain limitations inherent in non-density-based algorithms. These methods are categorized as point-based DBCLAs where the point density is calculated based on the number of points that lie within its immediate neighborhood. Some other point-based DBCLAs like OPTICS and HDBSCAN also build upon this concept, comparing the number of points lie within certain reachability distance to the defined minimum cluster size in order to discern clusters and outliers. Those methods further improved the parameter sensitivity problem by employing an augmented ordering of the dataset. Several other DBCLAs use probabilistic-based density computation to evaluate point density more accurately. For instance, STING [7] computes the confidence interval of probability, and IPCLUS utilizes kernel density estimation (KDE). These methodologies enhance the accuracy of extracting data with irregular shapes across diverse density levels. However, they often fall short in situations where clusters are in close proximity and have uneven densities. This phenomenon, we called the "touching issue," arises when clusters are too adjacent, or some thin or sparse connections exist between the clusters, as illustrated in (a) of Figure 1.



(a) Touching issue example (adjacent and bridge problem)

(b) Case example when DBSCAN fail to separate the groups with bridging problem

(c) Triangulation after removal of global effect which successfully solved the bridging problem

Figure 1: Illustration of the touching issue and the effectiveness of triangulation in solving this issue

Recent studies [10, 11] introduced triangulation to address touching issues in clustering problems.

Taking advantage of the Max-Min property [12] of Delaunay triangulation, a more refined proximity relationship between the data can be captured by triangulation. This is evident when comparing the application of a pure point-based DBSCAN to a triangulation-based DBSCAN in handling the bridging problem. By applying DBSCAN based on the number of triangle vertices connected to a point as the clustering criterion, rather than solely relying on the radius, triangulation after the removal of global effects better accommodates the data's shape, thus effectively distinguishing between the clusters (Figure (1)).

However, the without an accurate evaluation of the point density, the existing methods that combine triangulation with DBSCAN couldn't provide a universal solution to further complicated nonlinear data with nested clusters and various densities. To achieve the ultimate goal of generalizing the clustering algorithm to data with all kinds of shape and distance relations, we propose a new Density and Triangulation based Clustering (DTC) algorithm that combines tools like density estimation, Delaunary triangulation, and DBSCAN, to balance both the clustering accuracy and the generalizability of the clustering algorithm.

The rest of the article are organized as follows. In Section 2, we give an overview of the key concepts and steps in our algorithm: KDE, Triangulation, a variant version of DBSCAN, and KNN for noise handling. Section 3 presents simulation results comparing our method with its competitors and a real data application on the PM2.5 air pollution data in the United States to demonstrate our algorithm's efficacy in real life scenarios. In Section 4, we will review the advantages of DTC and propose potential advancements in the future.

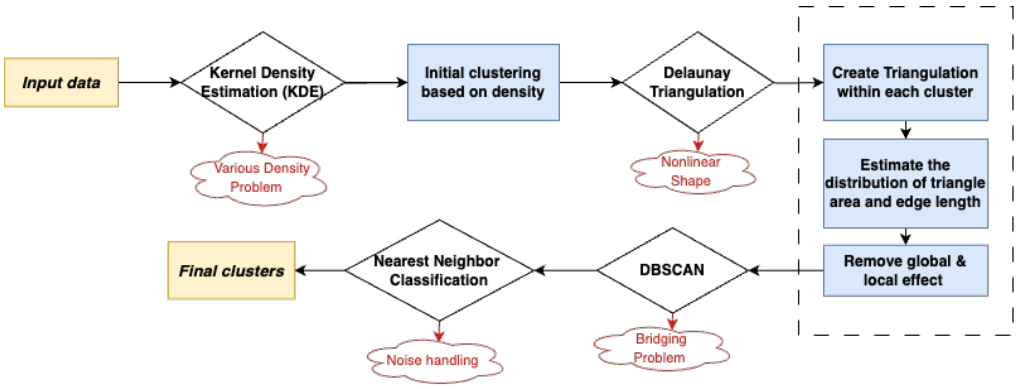


Figure 2: Flowchart of the Density and Triangulation based Clustering Algorithm.

2 Methodology

As shown in Figure 2, the first step is to calculate the kernel density estimation (KDE) for each point in the data set. We use an algorithm to segregate the data into initial clusters based on their

density level. Then we create Delaunay triangulation within each clusters. We remove the triangles with too long edges or too large area based on the estimated distribution of all the triangles. This process is called the removal of global effect. We then remove all the outlying edges based on the distribution of the edges connecting to each vertex, which is the removal of local effect. This two steps collectively helps with separating nonlinear shaped clusters. Upon the final triangle set, a variant of DBSCAN is applied. Rather than utilizing the radius of circle as the epsilon value, we opt for the count of neighboring vertices each point links to — this adjustment is designed for triangulation. Integrating Delaunay triangulation with DBSCAN addresses bridging challenges. Finally, points identified as noise during DBSCAN are relabeled using their closest labeled neighbors, effectively rectifying misclassified boundary points.

2.1 Density-based Separation

Density estimation

Density estimation refers to the process of estimating the probability distribution of a random variable from a set of data points. As a popular density estimation method, kernel density estimation (KDE) [14] techniques have found applications in various fields, including image processing, finance, anomaly detection, and environmental science. In our algorithm, we used Gaussian kernel to calculate KDE of each data point using Python's library *scipy.stats.gaussian_kde*. It includes automatic bandwidth determination and works for both uni-variate and multi-variate data.

Separation algorithm

Addressing the challenge of varying density among data clusters is the primary objective of our initial data separation. Clusters with differing density distributions, especially when adjacent, can be difficult to distinguish using only local proximity relationships. Consider Figure 3, where the space within the upper moon-shaped cluster and the interspace between the two clusters are marginally distinct, traditional clustering algorithms would easily misclassify the two moon-shaped clusters as a single cluster. To solve this problem, we employ kernel density estimation. This approach prioritizes global density distribution, ensuring data separation before evaluating local proximity relationships.

Let $\mathcal{P}$ be all the data points, $\mathcal{C}_i$ denotes the set of points that are put in to the same cluster i, and $\mathcal{U}$ denotes the remaining points. Initially, $\mathcal{C} = \emptyset$, $\mathcal{U} = \mathcal{P}$. We first compute the kernel density estimation for each data point. Among all the data points, we find point $p \in \mathcal{U}$ with highest density and put it in $\mathcal{C}$. Then we find a point $p' \in \mathcal{U}$ that is closest to p, and move p' into the set C, record the corresponding step size s. Then within $\mathcal{U}$, we look for the next point closest to any point in S, but only move it into the cluster if the new step size is not an outlier comparing to the distribution of all the past step sizes. To ensure we have a large enough sample size to calculate the statistics, we set a hyperparameter called "min_sample" where you define the minimum cluster size. In the following synthetic data example, we set the "min_sample" as 12, so that after the first 12 points are in the same cluster, once we detect an outlying step size (eg. greater than 3.5 standard deviation),

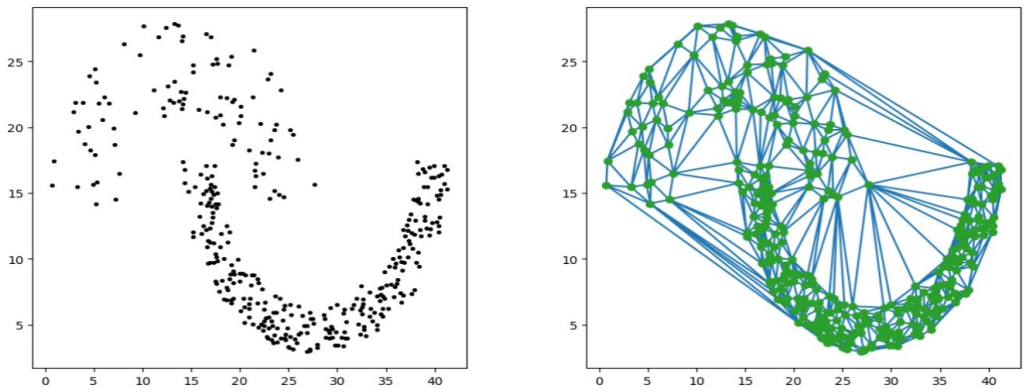
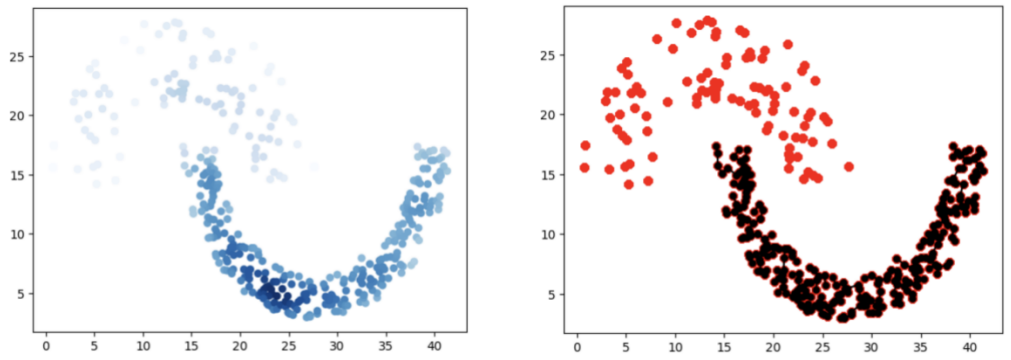


Figure 3: An example of data set with various density

we stop including the new point into the cluster. Then, within all the remaining points in $\mathcal{U}$, we repeat the process until $\mathcal{U} = \emptyset$.

Looking at Figure 4 (a), we can clearly see that the two moon-shaped data groups should be separated into different clusters due to their density difference. By performing the algorithm above, we can distinguish clusters with different densities that otherwise can not be distinguished by solely relying on local proximity relationship. Now, we are ready to solve the touching issue by Delaunay triangulation.



(a) kernel density estimation on each data point

(b) the initial clusters obtained after the density-based separation

Figure 4: Perform density-based separation on the same data set

Algorithm 1 Desnity-based separation

Input: data points P with their corresponding kernel density D
Initialization: $\mathcal{C}_i = \emptyset$, $i = 0$, $\mathcal{U} = P$, $\mathcal{L} = \emptyset$
while $\mathcal{U} \neq \emptyset$ **do**
    Find the point $p \in \mathcal{U}$ with the maximum kernel density in D and the closest point $p' \in \mathcal{U}$.
    Move p, p' from $\mathcal{U}$ to $\mathcal{C}$. Record the step size between p and p' as l in $\mathcal{L}$.
    Find the next closest point $p' \in \mathcal{U}$ to the set C and calculate the step size l
    if $l > (mean(\mathcal{L}) + 3.5sd(\mathcal{L}))$ **then**
        $i+ = 1$
    else
        Move p' from $\mathcal{U}$ to $\mathcal{C}$. Record the corresponding l into $\mathcal{L}$.
    end if
end while

2.2 Delaunay triangulations-based DBSCAN

Delaunay triangulations

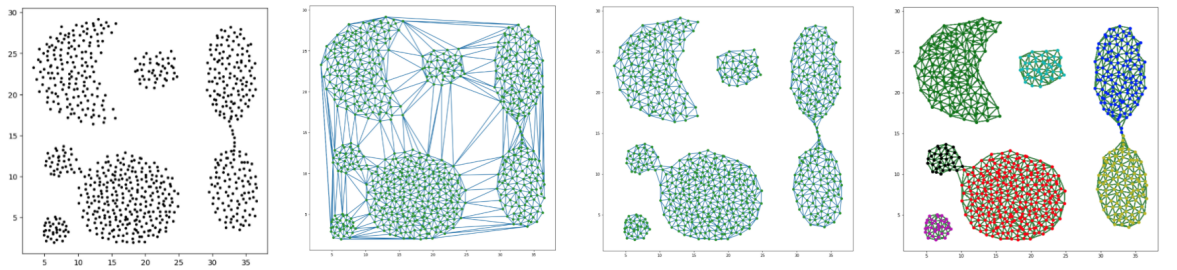
Given a set of points, Delaunay triangulation generates triangles from them such that no point is inside the circumcircle of any triangle. One of the most prominent properties of Delaunay triangulations is that they maximize the minimum angle of all the angles of the triangles in the triangulation, avoiding skinny triangles when unnecessary.

In our algorithm, we first create Delaunay triangulations Tri based on a set of points $S = \{s_1, \cdots, s_n\}$ in the plane. Let T be the set of all the triangle simplices, V be the set of all vertex indices for all triangles $t(x, y, z) \in T$ where x, y, z is the vertex indices of the triangle. Let $E(i, j)$ be the set of all edges for all triangles $t \in T$ that connects vertices i and j.

Removal of Global & Local Effect

We first estimate the distribution of the area and the edge length for all the triangles, globally. Due to the property of triangulation, we can safely remove the triangles with too long edges or too small area to reveal the potential clusters. We used 3.5 standard deviation away from mean as the threshold for outlying edges and area. Since when data points are relatively evenly distributed in a cluster, their triangulation plot should be mostly constituted by nearly equilateral triangles, it follows that we should also remove the outlying edges among all the edges connecting one vertex. It is the global and local removal as part of the preprocessing of the triangulation.

Figure 5 (c) shows the resulting set of triangles after we remove the global and local effect followed the distribution of areas and edge length of triangles in (b). Now we could start performing DBSCAN by choosing minimum points, a value of points required to form a cluster, as the hyperparameter. A detailed algorithm is demonstrated below.



(a) Original data with adjacent problem and touching issue

(b) Delaunay Triangulation on the data

(c) Set of triangles after removing global effect

(d) DBSCAN clustering result after KNN

Figure 5: Example of using Triangulation-based DBSCAN to solve adjacent problems

Algorithm 2 Triangulation-based DBSCAN

Input: data points X, a set of triangles T, and the hyperparamter $minPts$.
Initialization: Create a stack $\mathcal{C} = \emptyset$, a list of unvisited points $\mathcal{U} = X$. Set cluster index $i = 0$.
while $\mathcal{U} \neq \emptyset$ **do**
    Randomly choose a point from $\mathcal{U}$ and add it into $\mathcal{C}$.
    while $\mathcal{C} \neq \emptyset$ **do**
        Pop a point p from $\mathcal{C}$, add all the vertices connected to p into $\mathcal{C}$.
        Count the number vertices connected to p, compare it with $minPts$.
        if number of vertices $> minPts$ **then**
            mark p as a member of cluster i
        else
            mark p as a noise
        end if
    end while
    i += 1
end while

2.3 Relabeling of Noise

Up to this point, we already have a roughly accurate clustering results based on kernel density estimation, Delaunay triangulation, and DBSCAN. However, the design of hyperparameter $minPts$ that is passed down from DBSCAN causes data points near the border of a cluster being easily classified as noise. Therefore, we decide to perform a K-Nearest Neighbors (KNN) classification algorithm based on the clustering results we obtained above. In this process, the previously defined noises will be re-evaluated again.

3 Experiments & Interpretation

Before delving into the real dataset, we applied our algorithm onto clustering benchmark datasets and compared with multiple other clustering algorithms.

Figure 6 shows the classical shape dataset named "Aggregation", being used to test the performance of clustering algorithm on complicated spatial data. From the original clusters, we can see this data is nonlinear with both adjacent and bridging problem. Traditional algorithm such as K-means fail to handle the adjacent clusters and other density-based methods fail to separate out the two clusters on the right which are connected by the so-called "bridge". In our DTC approach (with details shown in Figure 5), relatively overly long edges are deleted before we perform DBSCAN, so that the bridge is clearly cut off since very few edges are still connected around the thin connection (the "bridge") between the two groups.

Figure 7 and 8 displayed two other benchmark data sets, all contain adjacent groups with various density distribution. Some clusters are even nested together, which can not be correctly distinguished by the traditional density-based algorithms. But with the aid of clustering based on kernel density estimation, DTC effectively solves the problem and achieves nearly perfect accuracy.

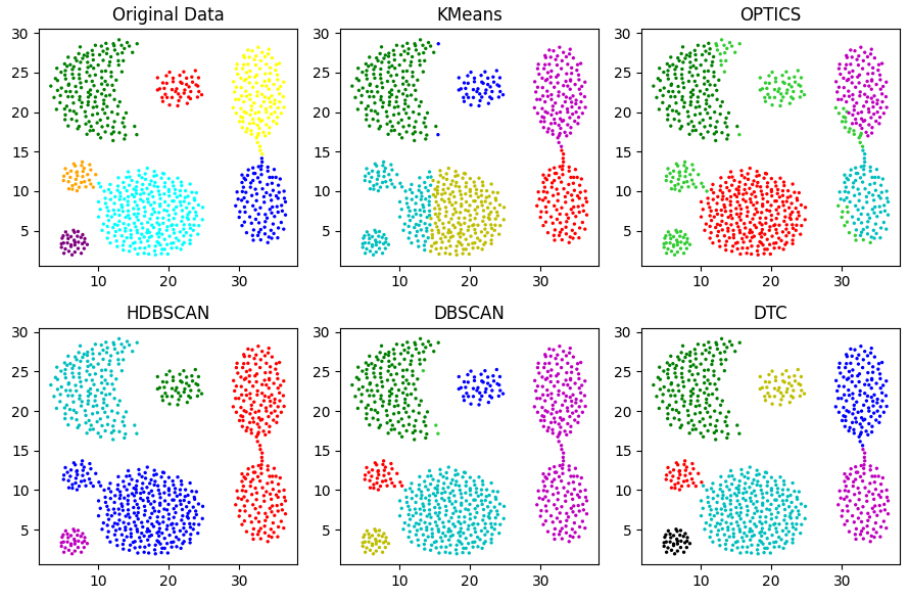


Figure 6: Comparison of the performance on the "Aggregation" data set

	Kernel K-mean	HDBSCAN	DBSCAN	OPTICS	DTC
Aggregation Dataset	86.29%	82.74%	86.42%	84.77%	99.62%
A.K. Jain's Toy Problem	78.55%	80.70%	65.15%	98.93%	100.00%
Zahn's Compound Dataset	57.89%	84.71%	80.95%	91.23%	99.50%

Table 1: Accuracy comparison table

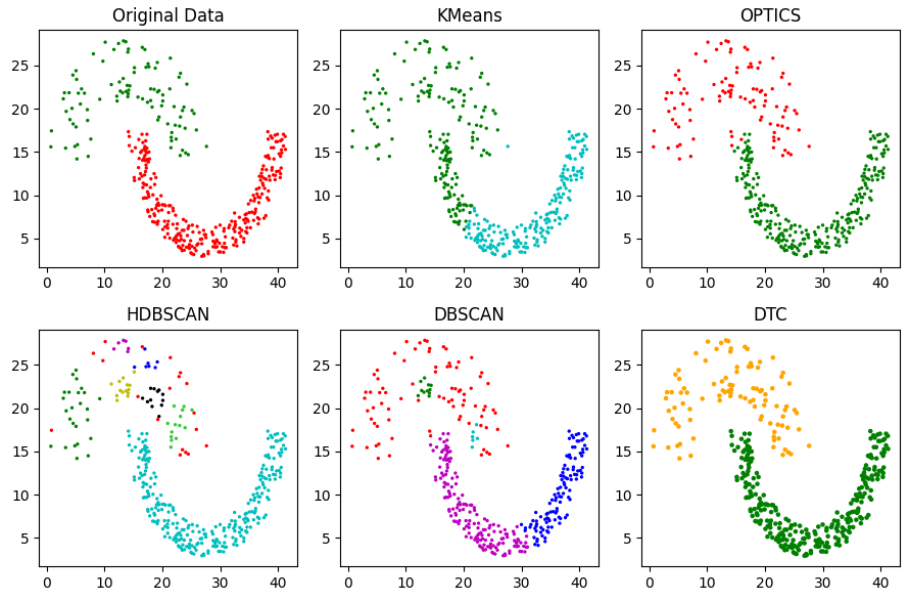


Figure 7: Comparison of the performance on the " A.K. Jain's Toy Problem" data set

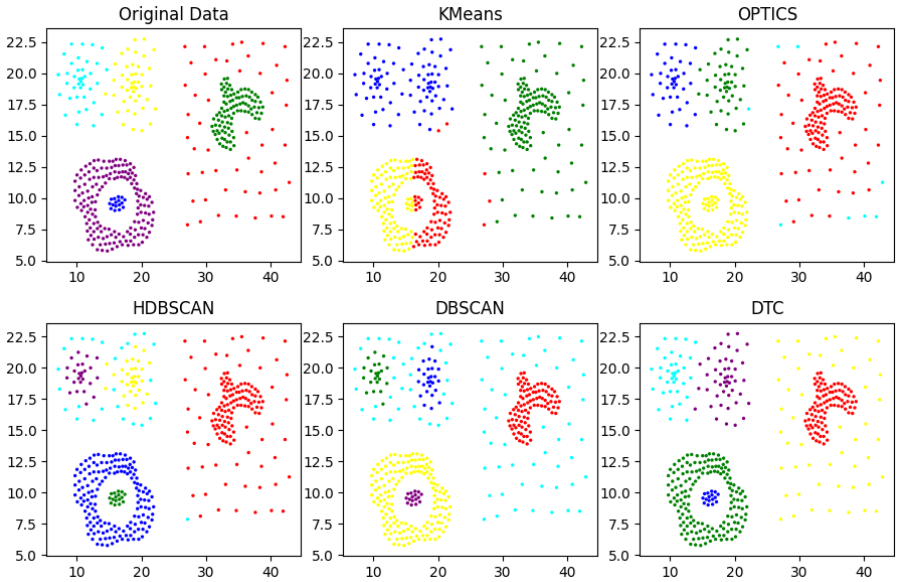
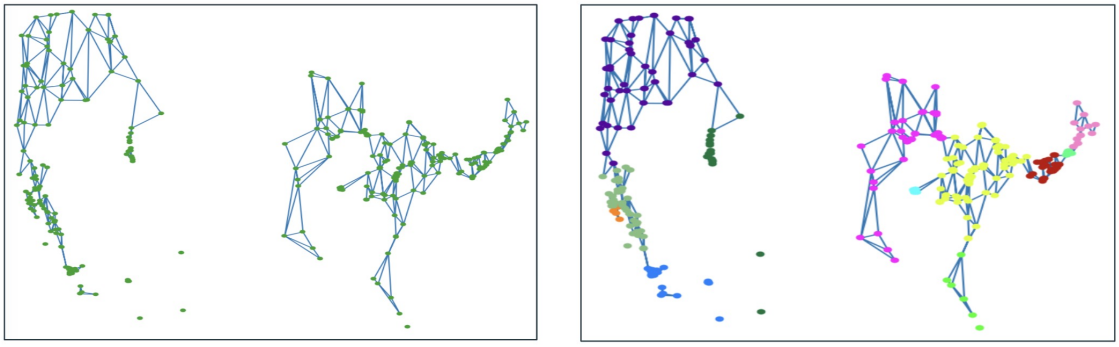


Figure 8: Comparison of the performance on "Zahn's Compound" data set

To further examine the applicability of this algorithm into real data, we applied the proposed algorithm to meteorological data. We used the daily mean surface concentrations of total PM2.5 for the year 2011 obtained from the United States Environmental Protection Agency. Among all the longitude and latitude data points, we selected the location where the corresponding PM2.5 value is greater than $12ug/m^3$, the threshold value given by U.S. Environmental Protection Agency (EPA) to differentiate good and potentially harmful air quality.

Upon implementing our algorithm as depicted in Figure 9, the clustering results from Figure 10 clearly demonstrates that the nonlinear-shaped PM2.5 spatial data points are separated into 12 clusters, providing insightful information on the air quality across various regions in the US. The dark green, purple, cyan, and orange clusters specifically pinpoint major urban centers including New York, Philadelphia, San Jose, and St Louis. Despite these cities being represented by closely situated data points that could be prone to misclassification, our algorithm effectively differentiates them.

Furthermore, Figure 11 showcases the varying pollution levels across each cluster. Cities on the west coast, especially in California, display higher pollution levels compared to those in the midland and on the east coast. Among eastern cities, areas encompassing Philadelphia, New York, and St Louis show elevated PM2.5 values. This clustering result also aligns with the "Most Polluted City" rankings by the American Lung Association[13].



(a) PM2.5 data after global & local removal

(b) PM2.5 data's final clustering result

Figure 9: Apply the triangulation algorithm to PM2.5 data

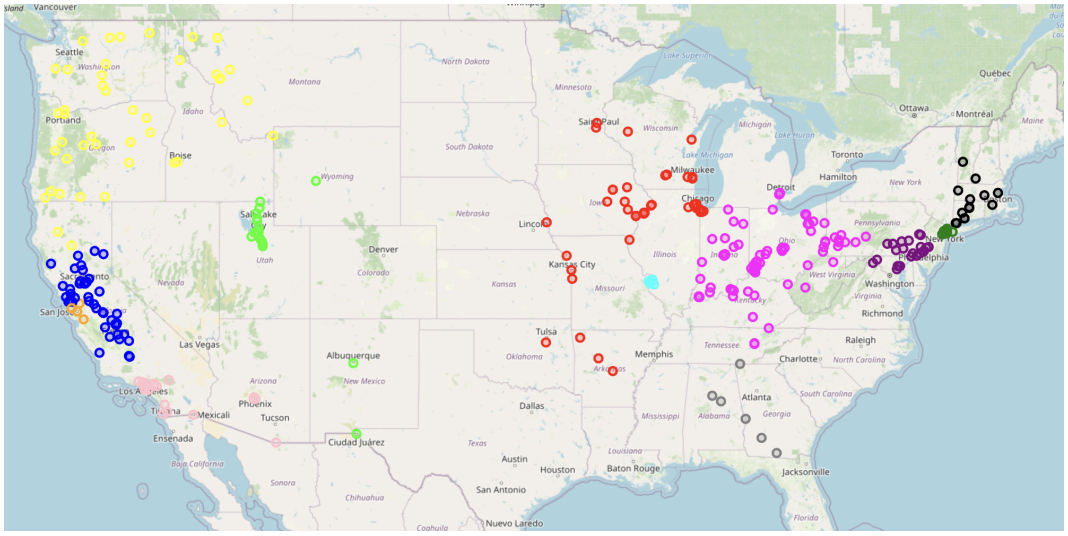


Figure 10: Clustering result on the 2011's PM2.5 data in the United States

4 Conclusion and Future Work

Our discussions and illustrations above have highlighted the main advantages of the proposed DTC algorithm, notably its proficiency in handling adjacent data points with complicated structure and density distribution. Its remarkable performance in tackling challenges related to the "touching" problem makes it favorably compared to other clustering methods.

Furthermore, as the machine learning and pattern recognition fields continue to grow, there emerges a pressing need for robust spatial clustering tools adept at managing intricate non-uniform

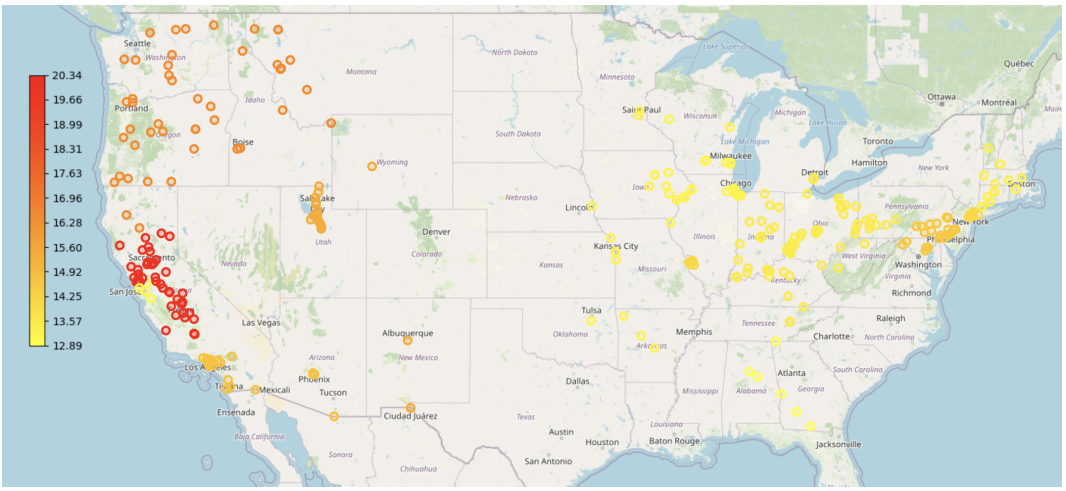


Figure 11: Average PM2.5 level for each clusters

datasets. So looking ahead, we plan to further improve the speed of DTC by implementing the FAISS library. FAISS is designed for efficient neighbor search, which can help us significantly improve KDE-based clustering process. we will develop a Fast-DTC that promises to improve both speed and mememory use. The potential of FDTC in addressing clustering challenges for large-scale, complex datasets will open more potential applications in broader machine learning contexts.

References

[1] D. B. Ramey, Nonparametric clustering techniques, *Encyclopedia of Statistical Science*. 6 (1985), 318–319.

[2] W. Gesler, The uses of spatial analysis in medical geography: A review, *Social Science & Medicine*. 23 (1986), no. 10, 963-973.

[3] D. J. Miller, Y. Wang, and G. Kesidis, Emergent unsupervised clustering paradigms with potential application to bioinformatics, *Frontiers in Bioscience-Landmark*. 13 (2008), no. 2, 677-690.

[4] M. Shtern and V. Tzerpos, Clustering methodologies for software engineering, *Advances in Software Engineering*. 2012.

[5] T. H. Grubesic, R. Wei, and A. T. Murray, Spatial clustering overview and comparison: Accuracy, sensitivity, and computational expense, *Annals of the Association of American Geographers*. 104 (2014), no. 6, 1134-1156. https://doi.org/10.1080/00045608.2014.958389

[6] M. Ester, H. P. Kriegel, J. Sander, and X. Xiaowei, A density-based algorithm for discovering clusters in large spatial databases with noise, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining.* (1996), 226-231.

[7] W. Wang, J. Yang, and R. Muntz, Sting: a statistical information grid approach to spatial data mining, *Proceedings of the 23rd International Conference on Very Large Data Bases.* (1997), 186–195.

[8] R. J. Campello, D. Moulavi, and J. Sander, Density-based clustering based on hierarchical density estimates, *Advances in Knowledge Discovery and Data Mining, 17th Pacific-Asia Conference.* (2013), 160–172.

[9] P. Bhattacharjee and P. Mitra, A survey of density-based clustering algorithms, *Front. Comput. Sci.*. 15 (2021), 151308. https://doi.org/10.1007/s11704-019-9059-3

[10] J. Kim and J. Cho, Delaunay triangulation-based spatial clustering technique for enhanced adjacent boundary detection and segmentation of LiDAR 3D point clouds, *Sensors.* 19 (2019), no. 18, 3926. https://doi.org/10.3390/s19183926

[11] Q. Liu, M. Deng, Y. Shi, and J. Wang, A density-based spatial clustering algorithm considering both spatial proximity and attribute similarity, *Computers & Geosciences.* 46 (2012), 296–309. 10.1016/j.cageo.2011.12.017.

[12] O. R. Musin, Properties of the Delaunay triangulation, Dept. of Cartography and Geoinformatics, Moscow State University. (n.d.).

[13] American Lung Association, Most polluted cities: State of the Air. https://www.lung.org/research/sota/city-rankings/most-polluted-cities (n.d.)

[14] E. Parzen, On estimation of a probability density function and mode, *The Annals of Mathematical Statistics.* 33 (1962), no. 3, 1065-1076. https://doi.org/10.1214/aoms/1177704472.