Large Data Clustering Analysis Based on K-Means and DBSCAN

Zhou Jiarui
1024040918
Nanjing University of Posts and Telecommunications
Jiangsu, Nanjing China

***Abstract*—This paper compares the K-Means and DBSCAN clustering algorithms, analyzing their strengths and limitations in big data analysis. While K-Means is simple and widely used, it is sensitive to initial conditions, requires specifying the number of clusters, and assumes spherical cluster shapes. Improvement methods such as K-Means++, X-Means, Constrained K-Means, k-prototype, and Kernel K-Means address these issues. DBSCAN excels in discovering clusters of arbitrary shapes and handling noise but is sensitive to parameters and struggles with varying densities and high-dimensional data. Improvements include parameter adaptation and multi-density clustering methods, such as OPTICS. This paper introduces HDBSCAN, a robust variant that automatically determines the number of clusters and handles varying densities through hierarchical clustering. Through experimental analysis, we validate the performance of these algorithms on large datasets and illustrate the clustering mechanisms using visualizations such as minimum spanning trees and cluster hierarchy diagrams. These enhancements improve the applicability of clustering algorithms in diverse big data analysis scenarios.**

Keywords—Clustering Algorithms, K-Means, DBSCAN, Big Data Analysis

I. Introduction

With the rapid development of information technology, the volume of data is growing at an exponential rate, driven by advancements in data collection, storage, and processing capabilities. Against this backdrop, extracting valuable information from vast amounts of data has become a hot research topic, as organizations and researchers seek to transform raw data into actionable insights. Cluster analysis, as an important means of data analysis, plays a pivotal role in this process. By grouping similar data points into clusters, it can effectively uncover inherent structures, patterns, and relationships within complex datasets, enabling a deeper understanding of the underlying data distribution. This capability makes cluster analysis indispensable in data mining and knowledge discovery, as it helps identify trends, anomalies, and hidden correlations that might otherwise remain unnoticed. From market segmentation and customer behavior analysis to bioinformatics and social network analysis, clustering techniques are widely applied across diverse domains, driving innovation and decision-making in both academic and industrial settings. As data continues to grow in scale and complexity, the importance of cluster analysis in unlocking the potential of big data will only continue to increase.

[1]The K-Means algorithm, known for its simplicity and ease of use, is widely applied and achieves clustering by minimizing the sum of squared errors (SSE) within clusters. On the other hand, the DBSCAN algorithm is favored for its robustness to noise and adaptability to cluster shapes, enabling it to discover clusters of arbitrary shape in spatial databases with noise.

[2]At its core, DBSCAN relies on density-based spatial clustering, dividing areas with sufficient density into clusters and identifying noise points. Compared to K-Means, DBSCAN does not require the number of clusters to be specified in advance and demonstrates better adaptability to non-spherical datasets and clusters of varying sizes. Additionally, the parameters of the DBSCAN algorithm (eps and min_samples) describe the density of sample distribution in neighborhoods, offering flexibility to the algorithm.

In big data analytics, the application scenarios of clustering algorithms are extensive. For instance, in market analysis, clustering can help businesses identify different customer segments to develop more precise marketing strategies. In bioinformatics, clustering algorithms can be used to analyze gene expression data, discovering genes with similar expression patterns. Moreover, clustering algorithms find applications in various fields such as social network analysis, image segmentation, anomaly detection, and more.

K-Means and DBSCAN are both excellent clustering algorithms, demonstrating outstanding performance on small to medium-sized datasets. However, their efficiency and effectiveness may face challenges when dealing with large-scale datasets. To validate their performance on large datasets, we selected two representative large-scale datasets for experimental analysis: “Product Classification and Clustering” and “Mturk User-Perceived Clusters over Images”. These datasets encompass high-dimensional features and complex structures, providing a comprehensive test of the capabilities of K-Means and DBSCAN in handling large-scale data.

In this paper,we will provide a detailed introduction to the core principles of K-Means and DBSCAN, as well as their improved methods, including K-Means++, X-Means, OPTICS, and HDBSCAN. Subsequently, we will validate the performance of these algorithms on large datasets through experiments, analyzing their computational efficiency, clustering accuracy, and ability to handle noise and outliers. Through these experiments, we aim to offer valuable insights for algorithm selection in practical applications and further promote the optimization and development of clustering algorithms in large-scale data analysis.

II. K-Means

A. The K-Means Algorithm

First, data is read from a given CSV file, which contains two columns labeled V1 and V2. These data are then transformed into a two-dimensional array, with the first column serving as the x-coordinate and the second column as the y-coordinate. For this purpose, the Pandas library is utilized to load the CSV file, and subsequently, the data are converted into a NumPy array. The outcome of these points from the CSV file is illustrated in Figure 1.

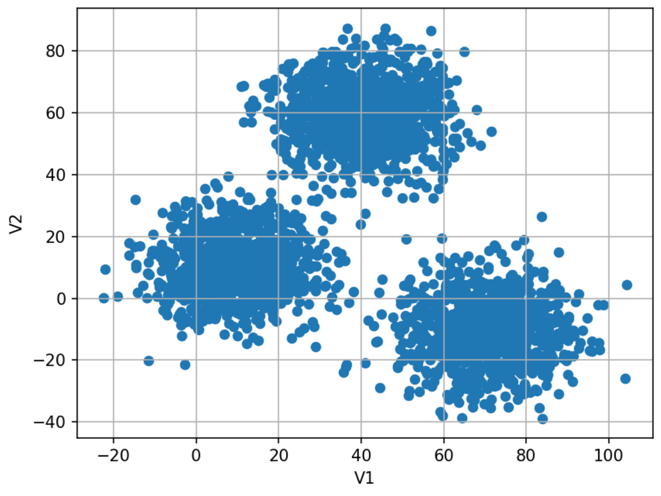

Fig. 1. Original data.

Following this preparation, the specific steps of the K-Means algorithm are implemented as follows:

- Selection of K: Determine the number of clusters, K.
- Initialization of Centroids: Randomly select K initial centroids from the dataset.
- Cluster Assignment: Assign each data point to the nearest centroid, thereby forming K clusters.
- Centroid Update: Recalculate the position of the centroids based on the mean location of the data points within each cluster.
- Iteration: Repeat the cluster assignment and centroid update steps until the centroids no longer change significantly or until a maximum number of iterations has been reached.

After clustering, the results are visualized by plotting each data point using different colors according to their assigned cluster, with the final outcome presented in Figure 2.

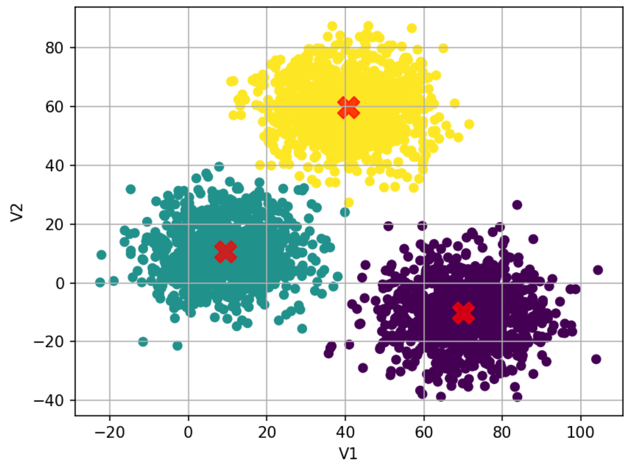

Fig. 2. Clustering Analysis Results with k=3.

B. *Limitations of K-Means*

The K-Means algorithm is widely used in cluster analysis, but it also has some significant drawbacks. Firstly, K-Means is highly sensitive to the selection of initial centroids. Different initial centroids may lead to significant differences in clustering results and may even fall into local optimal solutions. Secondly, K-Means requires users to pre-specify the number of clusters K before the algorithm runs.

However, without prior knowledge, this choice is usually subjective and difficult to determine. In addition, K-Means is very sensitive to noise and outliers. Noisy data or outliers can significantly affect the calculation of centroids, resulting in unsatisfactory clustering results. Meanwhile, K-Means assumes that clusters are spherical and is suitable for uniformly distributed data, which makes it less effective when dealing with data of complex shapes or uneven distributions. Finally, in high-dimensional space, the performance of K-Means will also be affected by the loss of meaning in distance metrics, leading to a decline in clustering effectiveness.

When the value of k is not correct, erroneous results will be obtained, as shown in Figures 3 and 4. Figure 3 represents the result when k is set to 1, and Figure 4 represents the result when k is set to 5.

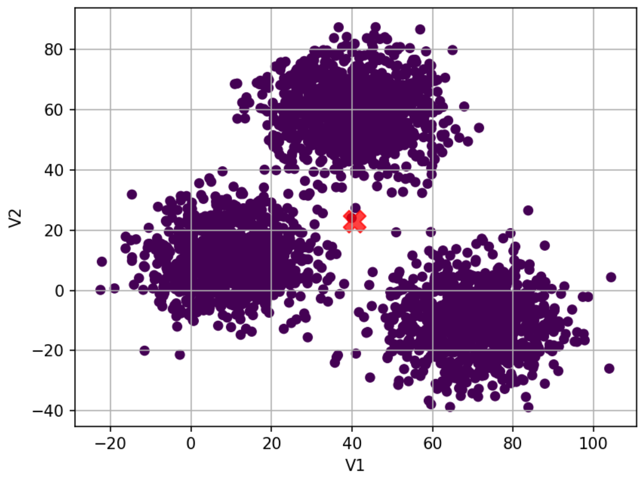

Fig. 3. Clustering Analysis Results with k=1.

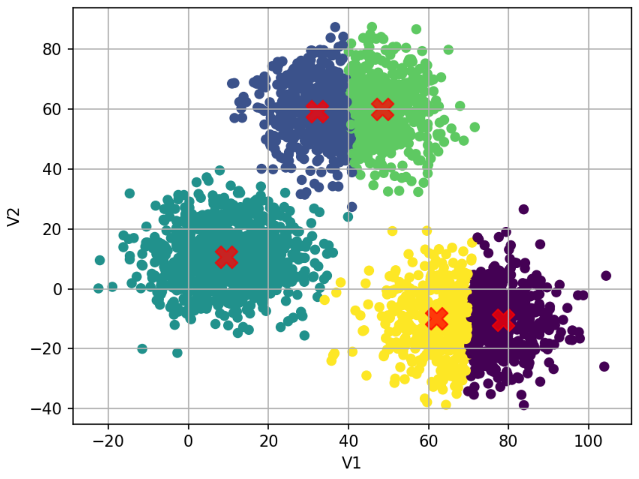

Fig. 4. Clustering Analysis Results with k=5.

C. *Improvements to K-Means*

In response to these shortcomings, K-Means can be improved through various methods. [3]Firstly, K-Means++ is an improved initialization method that selects better initial centroids to enhance clustering performance, thereby reducing sensitivity to initial choices.

Secondly, one of the most significant limitations of the K-Means algorithm is its sensitivity to the choice of k, the number of clusters. Selecting an appropriate k value is crucial for obtaining meaningful clustering results. Common methods for determining k include the Elbow Method, [3]the Gap Statistic, and the Silhouette Coefficient method.

1) The Elbow Method: The core metric of the Elbow Method is the Sum of Squared Errors (SSE), which quantifies the compactness of the clusters. As the number of clusters k increases, the data points are partitioned more finely, leading to higher intra-cluster cohesion and consequently a reduction in SSE. When k is less than the true number of clusters, increasing k significantly enhances cluster cohesion, resulting in substantial decreases in SSE. However, once k reaches the true number of clusters, further increments yield diminishing returns in terms of cohesion improvement, causing the rate of decrease in SSE to sharply diminish and eventually level off as k continues to increase. Thus, the relationship between SSE and k forms an elbow-shaped curve, where the "elbow" or inflection point indicates the optimal k value corresponding to the true number of clusters in the dataset. Figure 3 illustrates this relationship for the dataset discussed in the article.

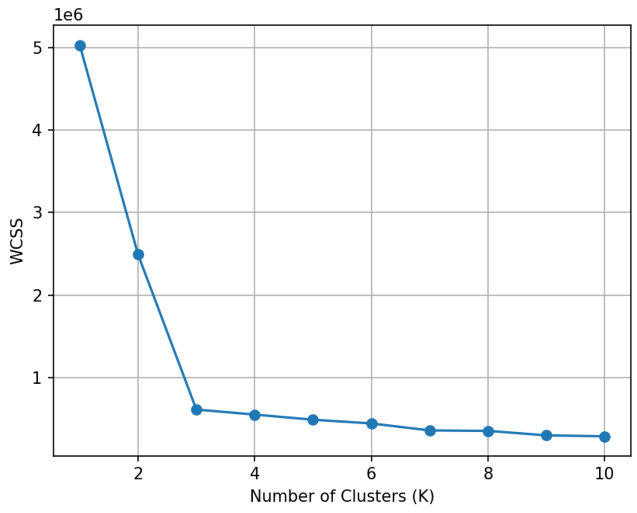

Fig. 5. Elbow Method for Optimal K.

2) The Gap Statistic Method: The drawback of the elbow method is that it requires manual observation and is not sufficiently automated. Therefore, we have thought of the Gap statistic method. This value is typically generated through Monte Carlo simulation. We create random samples in the same number as the original samples within the sample area according to a uniform distribution, and perform K-Means on these random samples to obtain a D_k. Repeating this process many times, usually 20 times, we can get 20 $\log D_k$ values. By averaging these 20 values, we get an approximation of $E(\log D_k)$. Eventually, we can calculate the Gap Statistic. The optimal K corresponding to the maximum Gap Statistic value is the best K. The final visualization result is shown in Figure 4.

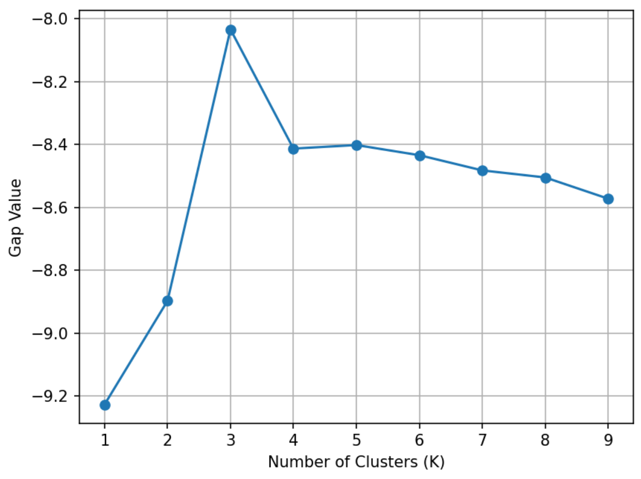

Fig. 6. Gap Statistic for Optimal K.

3) The Silhouette Coefficient Method: Calculating the silhouette coefficient involves considering the cohesion (a) and separation (b) of each data point. Cohesion refers to the average distance of a data point to other points in the same cluster, while separation refers to the average distance of a data point to points in the nearest different cluster. The formula for calculating the silhouette coefficient is:

$$s = \frac{b-a}{\max(a,b)} \tag{1}$$

Where s is the silhouette coefficient, a is the average distance of a sample to other samples in the same cluster, and b is the average distance of a sample to samples in the nearest different cluster. The value of the silhouette coefficient ranges from -1 to 1, with values closer to 1 indicating better clustering results, and values closer to -1 indicating worse clustering results. To determine the optimal K value, we need to calculate the silhouette coefficient for different K values. Specifically, for each possible K value, perform K-Means clustering and calculate the silhouette coefficient for each data point, then take the average of all data point silhouette coefficients to obtain the average silhouette coefficient corresponding to each K value. Finally, by plotting the relationship between the average silhouette coefficient and K value, select the K value that maximizes the average silhouette coefficient as the optimal number of clusters. The clustering result corresponding to this K value is generally considered the most reasonable.

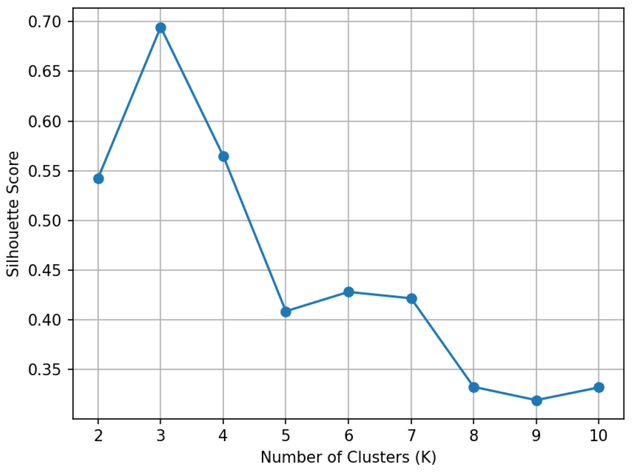

Fig. 7. Silhouette Coefficients for Optimal K.

Finally, for dealing with noise and outliers, density-based clustering algorithms such as DBSCAN (which will be discussed in the next subsection) can be used instead of K-Means for clustering, thereby enhancing the adaptability to complex data.

D. *Variants of the K-Means algorithm*

In response to these shortcomings, K-Means can be improved through various methods. Firstly, K-Means++ is an improved initialization method that selects better initial centroids to enhance clustering performance, thereby reducing sensitivity t initial choices.

1) X-Means: X-Means is an improved k-means clustering algorithm designed to address the issue of having to pre-specify the number of clusters K in the k-means algorithm. [4]X-Means utilizes kd-trees to accelerate each iteration of k-means, thereby reducing computational overhead. Users only need to specify a range of K values (k_min and k_max), and X-Means will automatically select the optimal K value within this range. This is achieved by calculating the Bayesian Information Criterion (BIC) scores for each cluster, with the model having the lowest BIC score being chosen as the optimal clustering. X-Means performs 2-means (dividing each cluster into two) in each iteration because 2-means is less sensitive to local optima, which helps in finding the global optimum solution.

2) Constrained K-Means: Constrained K-Means (CKM) is a variant of the k-means algorithm that allows for the incorporation of hard or soft constraints during the clustering process. [5]These constraints can include minimum or maximum cluster sizes, as well as requirements that certain data points must or must not be assigned to specific clusters. CKM achieves these constraints by transforming the clustering assignment step into a minimum cost flow (MCF) linear network optimization problem. This approach is particularly suitable for scenarios where specific business rules or relationships between data points need to be considered.

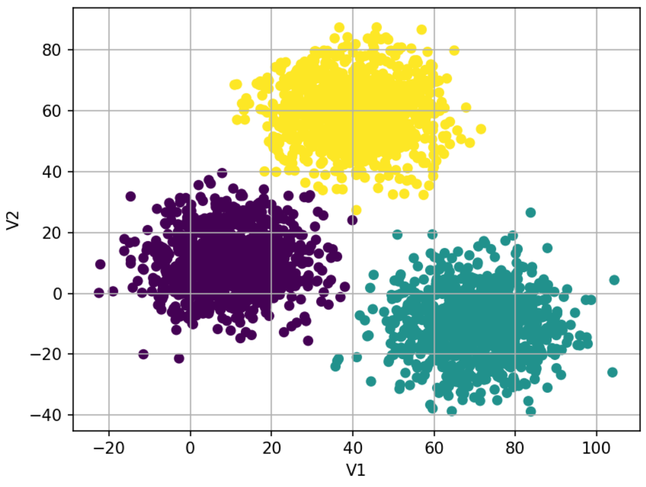

Fig. 8. Clustering Analysis Results with Constrained K-Means.

3) k-prototype: The k-prototype algorithm, introduced by Huang in 1997, integrates the concepts of the k-means and k-modes algorithms, making it suitable for datasets that encompass both continuous and categorical attributes. In the clustering process, the k-prototype algorithm handles numerical and categorical variables separately. For numerical variables, the algorithm employs Euclidean distance to measure the distance between samples; for categorical variables, it utilizes Hamming distance. The outcomes of these two distance calculations are summed to determine the total distance between samples. The k-prototype algorithm can be regarded as an extension of the k-means algorithm, as it defines distinct distance functions for continuous and categorical attributes to manage these two distinct types of data. The algorithm introduces a prototype for mixed-attribute clusters, where the prototype for numerical attributes is the average of all attribute values, and the prototype for categorical attributes is the attribute value with the highest frequency. The k-prototype algorithm establishes an objective function, akin to the SSE (sum of squared errors) of k-means, and iterates continuously until the objective function value stabilizes.

4) The Elbow Method: Kernel K-Means is a nonlinear version of the k-means algorithm that captures nonlinear structures in the data by mapping it into a high-dimensional space using kernel tricks. Kernel K-Means can choose different kernel functions, such as the Radial Basis Function (RBF) kernel and polynomial kernels, etc. Based on the selected kernel function, the kernel matrix between data points is calculated. Initial cluster labels are randomly chosen, or methods such as K-Means++ can be used for initialization. The algorithm iteratively performs cluster assignment and centroid updates through the kernel matrix until convergence criteria are met. Kernel K-Means is capable of identifying nonlinear structures in the data, such as crescent-shaped clusters, which are difficult to achieve with standard k-means. The visualization of the data mapped into the high-dimensional space by Kernel K-Means is shown in Figure 9.

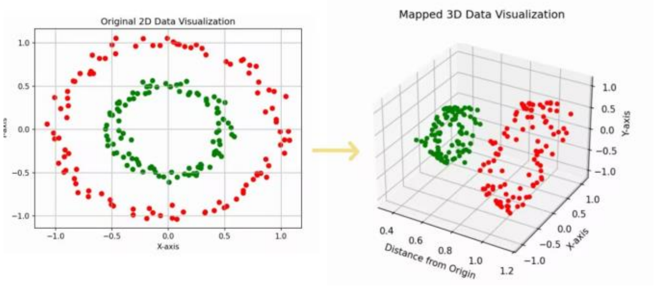

Fig. 9. Clustering Analysis Results with Constrained k-prototype.

E. *Results*

We conducted experimental analysis on X-Means, Constrained K-Means, K-Prototype, Kernel K-Means, as well as K-Means and K-Means++, and compared their performance on the aforementioned metrics. For each experiment, we employed a 10-fold cross-validation scheme to partition the dataset, and the reported results include the average values and standard deviations of these metrics across the 10 validation folds. Table 1 introduces the algorithms dealing with initialization issues, and Table 2 contains methods oriented towards mixed data types. The experimental evaluation was conducted using Accuracy and Adjusted Rand Index (ARI) as metrics.

TABLE I. COMPARISON OF K-MEANS CONSTRAINED K-MEANS ,AND X-MEANS

Metric	Product Classification and Clustering		
	k-means	*Constrained k-means*	*x-means*
Accuracy	0.533	0.860	0.675
ARI	0.824	0.812	0.6

TABLE II. COMPARISON OF K-PROTOTYPE AND KERNEL-K-MEANS

Metric	Mturk User-Perceived Clusters over Images	
	k-prototype	*Kernel-k-means*
Accuracy	0.513	0.618
ARI	0.009	0.136

Through experimental analysis on two large datasets, Product Classification and Clustering and Mturk User-Perceived Clusters over Images, we observed significant differences in the performance of various clustering algorithms. On the Product Classification and Clustering dataset, Constrained K-Means performed the best, achieving an accuracy of 0.860 and an ARI of 0.812, significantly outperforming standard K-Means and X-Means, indicating that the introduction of constraints effectively improved clustering results. On the Mturk User-Perceived Clusters over Images dataset, Kernel K-Means outperformed K-Prototype, with an accuracy of 0.618 and an ARI of 0.136, demonstrating the advantage of kernel methods in handling complex data structures. However, the relatively low ARI values for both algorithms suggest room for improvement when dealing with high-dimensional and complex data. Overall, improved algorithms (such as Constrained K-Means and Kernel K-Means) generally outperform traditional algorithms, but further optimization is needed to address more complex data scenarios. Future research could focus on parameter optimization, hybrid methods, and deep learning-based clustering to enhance the performance of algorithms on large-scale and complex data.

III. DBSCAN

DBSCAN and K-means are two widely used clustering algorithms, differing primarily in their clustering principles and applicable scenarios. DBSCAN, density-based, identifies clusters of arbitrary shapes and handles noise but is sensitive to parameters Eps and MinPts. K-means, distance-based, requires predefining the number of clusters and suits spherical clusters but is noise-sensitive and ineffective for complex shapes. DBSCAN excels with complex, noisy data, while K-means is better for regular-shaped, low-noise datasets.

A. *The DBSCAN Algorithm*

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based clustering algorithm capable of effectively discovering clusters of arbitrary shape and exhibiting strong robustness against outliers. The core idea of this algorithm is to partition clusters based on the density-connected relationships between data instances, without the need to specify the number of clusters in advance. DBSCAN determines density-connected regions through two key parameters: Eps (neighborhood radius) and MinPts (minimum neighborhood density threshold), identifying high-density regions as clusters and low-density regions as noise points.

The algorithm relies on two key parameters: Eps (neighborhood radius) and MinPts (minimum neighborhood density threshold). Eps defines the radius range used to determine whether instances are neighbors, while MinPts specifies the minimum number of instances required to form a dense region. Based on these two parameters, DBSCAN divides the data space into three categories: core objects, border objects, and noise objects. Core objects in high-density regions are connected through density-reachability to form a cluster, while border objects exist as the boundaries of the cluster. Noise objects in low-density regions are identified as outliers since they are not connected to any cluster.

Unlike traditional clustering methods based on distance or prototypes, DBSCAN can automatically discover clusters of arbitrary shape and varying density in complex data distributions while exhibiting strong robustness against noise data. This unique advantage makes DBSCAN widely applicable in anomaly detection, data cleaning, and other domains where discovering abnormal patterns and handling outliers is crucial.

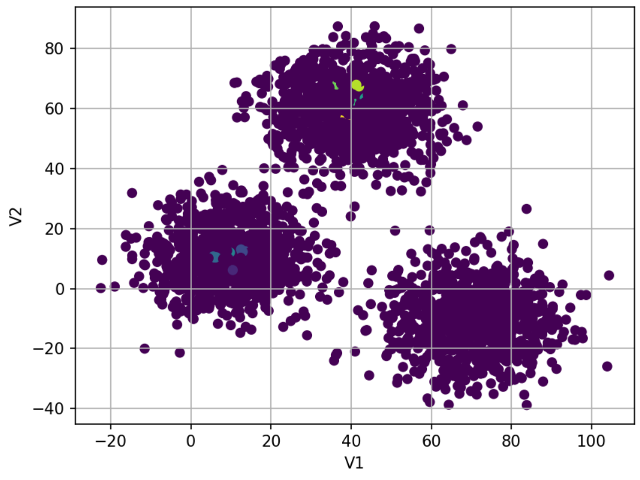

Fig. 10. Clustering Analysis Results of DBSCAN.

B. *Limitations of DBSCAN*

Although DBSCAN (Density-Based Spatial Clustering of Applications with Noise) demonstrates certain advantages in clustering tasks, such as the ability to identify clusters of arbitrary shapes and effectively handle noise, it also exhibits several notable limitations. Firstly, DBSCAN is highly sensitive to the parameters Eps (neighborhood radius) and MinPts (minimum number of points), which significantly influence the clustering results. In practical applications, especially in large-scale data scenarios, determining the optimal parameter combination often requires extensive trial and error, increasing computational costs and potentially leading to unstable clustering outcomes. Secondly, DBSCAN assumes that all clusters have similar densities, making it ineffective for datasets with varying densities. For instance, when a dataset contains both high-density and low-density clusters, DBSCAN may misclassify low-density clusters as noise or fail to accurately identify the boundaries of high-density clusters. Additionally, DBSCAN faces significant memory overhead issues. Since the algorithm requires computing a distance matrix between all data points, its space complexity is $O(n^2)$. [6]When dealing with large-scale datasets, memory requirements grow rapidly, resulting in poor scalability. This issue is particularly pronounced in high-dimensional data scenarios, where distance metrics often lose their effectiveness, further exacerbating the computational burden. Therefore, despite its strengths in certain scenarios, DBSCAN's sensitivity to parameters, inability to adapt to varying densities, and high memory consumption limit its applicability to large-scale and high-dimensional data.

C. *Improvements to the DBSCAN*

1) Parameter Adaptation Methodss: To address DBSCAN's sensitivity to parameters Eps and MinPts, researchers have proposed adaptive methods. For example, the k-distance graph-based approach automatically determines Eps by analyzing data distribution, while local density estimation dynamically adjusts MinPts, reducing reliance on manual tuning and enhancing robustness.

2) Multi-Density Clustering Enhancements: To tackle DBSCAN's inability to handle multi-density data, the OPTICS algorithm introduces reachability distance and reachability plots to identify clusters of varying densities. Additionally, density peak-based methods effectively partition multi-density data by identifying points with high local density as cluster centers.

3) Memory Optimization and Scalability Improvements: To reduce memory overhead, spatial indexing techniques (e.g., R-tree, KD-tree) minimize distance computations. Distributed DBSCAN improves scalability for large-scale data by partitioning data across multiple nodes for parallel processing.

4) Adaptability to High-Dimensional Data: To address distance metric issues in high-dimensional data, subspace clustering selects relevant dimensions to mitigate the "curse of dimensionality." Combining dimensionality reduction techniques (e.g., PCA, t-SNE) as preprocessing steps preserves local structures, further enhancing clustering performance in high-dimensional spaces.

D. *Variants of the DBSCAN*

HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) is a density-based clustering algorithm that combines the advantages of hierarchical clustering and density-based clustering. It can automatically identify cluster structures in data and effectively handle noise points. Unlike traditional DBSCAN, HDBSCAN does not require pre-specifying global density parameters (such as eps). Instead, it adapts to clusters of varying densities by constructing a hierarchical structure of the data. Its core idea is to build a Minimum Spanning Tree (MST) based on mutual reachability distance and generate a hierarchical clustering tree through pruning. Each node in the hierarchical clustering tree represents a potential cluster, and the depth of the tree reflects the density variations within clusters. HDBSCAN automatically selects the optimal cluster structure by optimizing cluster stability, thereby avoiding the complexity of manual parameter tuning. Additionally, HDBSCAN can identify noise points (labeled as -1) and adapt to clusters of different densities in the data, making it highly effective for handling complex datasets.

The strengths of HDBSCAN lie in its robustness and flexibility. It is capable of handling clusters of arbitrary shapes and effectively addressing noise and density variations in data. Unlike distance-based clustering algorithms such as K-Means, HDBSCAN does not require pre-specifying the number of clusters. Instead, it automatically determines the optimal number of clusters by analyzing the stability of the hierarchical clustering tree. Furthermore, HDBSCAN provides rich visualization tools, such as the minimum spanning tree and the condensed tree, which help users gain deeper insights into the hierarchical structure of the data and the clustering process. These features make HDBSCAN highly valuable for real-world data analysis, particularly excelling in high-dimensional data, noisy data, and datasets with uneven density distributions. By integrating the strengths of density-based clustering and hierarchical clustering, HDBSCAN offers a powerful and flexible solution for clustering analysis of complex data.

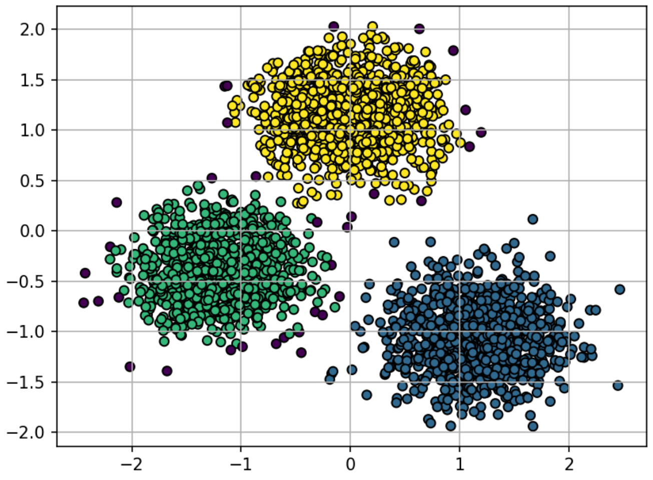

Fig. 11. Clustering Analysis Results of HDBSCAN.

Figure 12, the Minimum Spanning Tree (MST) graph, illustrates the connectivity relationships between data points, providing a clear visualization of how HDSCAN constructs clusters by identifying dense regions and separating them from sparser areas. The MST highlights the essential connections that define the underlying structure of the data, enabling a deeper understanding of how HDSCAN distinguishes clusters from noise and identifies the natural groupings within the dataset. [7]Figure 13, the Cluster Hierarchy graph, showcases the hierarchical structure of clusters, offering insights into the formation process of clusters at different density levels. This graph reveals how HDSCAN organizes data points into a tree-like structure, where clusters are formed by iteratively merging smaller clusters based on their density connectivity. By visualizing the hierarchy, one can better comprehend how HDSCAN captures the nested relationships between clusters and identifies stable clusters that persist across varying density thresholds. Together, these visualizations provide a comprehensive understanding of HDSCAN's clustering mechanism, from the initial connectivity analysis to the final hierarchical cluster formation.

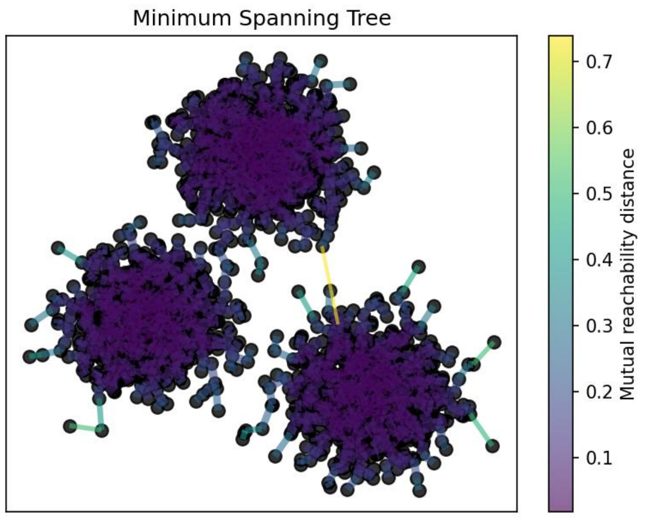

Fig. 12. Minimum Spanning Tree.

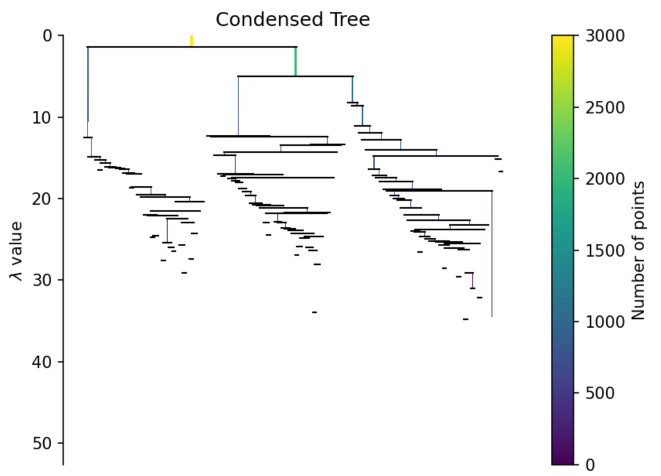

Fig. 13. Condensed Tree.

E. *Results*

HDBSCAN outperforms DBSCAN on both datasets, particularly excelling on the Product Classification and Clustering dataset with significantly higher accuracy and ARI, as well as a notable reduction in noise points. In contrast, DBSCAN struggles with high-dimensional and complex data, especially on the Mturk User-Perceived Clusters over Images dataset, where it achieves a lower ARI and a higher percentage of noise points. HDBSCAN demonstrates stronger robustness and adaptability through hierarchical clustering and adaptive density processing, making it well-suited for handling complex and large-scale datasets. Future research directions could focus on parameter optimization, exploring ways to automatically tune parameters such as Eps and MinPts for DBSCAN and HDBSCAN to enhance clustering performance; investigating hybrid methods that combine DBSCAN and HDBSCAN with other clustering algorithms to leverage the strengths of both density-based and hierarchical clustering; and incorporating deep learning techniques to design clustering algorithms capable of automatically learning data features, thereby addressing more complex data structures.

TABLE III. COMPARISON OF HDBSCAN AND DBSCAN

Metric	**Product Classification and Clustering**	
	DBSCAN	***HDBSCAN***
Accuracy	0.710	0.850
ARI	0.677	0.710

TABLE IV. COMPARISON OF HDBSCAN AND DBSCAN

Metric	**Mturk User-Perceived Clusters over Images**	
	DBSCAN	***HDBSCAN***
Accuracy	0.480	0.620
ARI	0.110	0.250

IV. CONCLUSION

Experimental results indicate that K-Means performs well on low-dimensional and uniformly distributed data but struggles with high-dimensional and complex data, while DBSCAN excels in handling noise and non-spherical clusters but faces challenges with high-dimensional data. Future research could focus on parameter optimization, exploring ways to automatically tune algorithm parameters (such as the number of clusters for K-Means and Eps and MinPts for DBSCAN) to improve clustering performance; investigating hybrid methods that combine K-Means and DBSCAN with other clustering algorithms to leverage the strengths of both distance-based and density-based clustering; and incorporating deep learning techniques to design clustering algorithms capable of automatically learning data features, thereby addressing more complex data structures.

V. SUMMARY

This article aims to comprehensively compare and analyze the two classic clustering algorithms, K-Means and DBSCAN, exploring their strengths, limitations, and improvement methods in big data analysis. By delving into their core principles and applicable scenarios, the paper demonstrates how to overcome their limitations through improved algorithms (such as K-Means++, X-Means, OPTICS, HDBSCAN, etc.), thereby enhancing clustering performance. Future research directions include handling high-dimensional data, adaptive parameter tuning, integration with deep learning, optimization of computational efficiency, and the development of domain-specific customized algorithms. As data scale and complexity continue to grow, the research and optimization of clustering algorithms will provide stronger support for data mining and knowledge discovery.

REFERENCES

After a semester of studying big data analysis courses, I have accumulated a wealth of knowledge in this field and mastered many practical analytical skills. Mr. Zou Zhiqiang's classes are not only rich in content but also filled with valuable practical experience. Through his lectures, I gained a deep understanding of some core concepts and fundamental methods of big data analysis. In this semester's experiment, I chose clustering analysis as my research direction, and through the combination of theory and practice, I further deepened my understanding of K-Means and DBSCAN. Here, I would like to express my heartfelt gratitude to Mr. Zou Zhiqiang for his hard work and dedicated guidance throughout the semester. His professionalism and patient explanations have been immensely beneficial to me. This learning experience has not only broadened my knowledge but also sparked a strong interest in the field of big data analysis.

[1] *Ahmad, A., et al., "A K-mean clustering algorithm for mixed numeric and categorical data," Data Knowl. Eng., 2007.*

[2] Ester M., Kriegel H.-P., Sander J., Xu X., “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,” in Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD-96), 1996.

[3] A. Arthur and S. Vassilvitskii, “K-Means++: The Advantages of Careful Seeding,” in *Proc. 18th Annu. ACM-SIAM Symp. Discrete Algorithms (SODA)*, 2007, pp. 1027–1035.K. Elissa, “Title of paper if known,” unpublished.

[4] D. Pelleg and A. Moore, “X-Means: Extending K-Means with Efficient Estimation of the Number of Clusters,” in *Proc. 17th Int. Conf. Mach. Learn. (ICML)*, 2000, pp. 727–734.

[5] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl, “Constrained K-Means Clustering with Background Knowledge,” in *Proc. 18th Int. Conf. Mach. Learn. (ICML)*, 2001, pp. 577–584.

[6] Y. Li, W. Zhang, and X. Wang, “Adaptive DBSCAN: A Density-Based Clustering Algorithm for Dynamic Data Streams,” *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 8, pp. 1–15, 2023, doi: 10.1109/TKDE.2023.10373228.

[7] J. Liu, Y. Wang, and T. Zhang, “A Comparative Study of DBSCAN and K-Means for Real-Time Clustering in Streaming Data,” in Proc. 32nd ACM Int. Conf. Information and Knowledge Management (CIKM), 2023, pp. 4002–4010, doi: 10.24963/ijcai.2023/445.