https://doi.org/10.31449/inf.v50i6.9905

CB-DBSCAN: A Core Ball-Enhanced Grid-Based Density Clustering Method for High-Dimensional Financial Data

Xiaohe Li*, Linwei Wan
Management School, Henan University of Urban Construction, Pingdingshan, 467036, China
E-mail: xiaohe-li@outlook.com

Keywords: financial big data, high-dimensional data, clustering algorithm, DBSCAN, core ball

Received: June 18, 2025

To optimize the allocation of financial resources, improve decision-making efficiency and enhance market competitiveness, this paper proposes a financial big data clustering method based on Core Ball (CB) and Density-Based Spatial Clustering of Applications with Noise (DBSCAN). First, the algorithm improves DBSCAN using a grid tree. This divides the data space into non-empty grids and constructs a grid tree structure, achieving a fast search of adjacent grids. At the same time, an angle-based pruning strategy is introduced to eliminate redundant trivial points using geometric angle relationships, reduce the number of distance calculations, and lower the computational complexity of the algorithm. On this basis, the concept of CBs is further introduced to divide the dataset into two categories: CBs and non-CBs. The CBs are merged into the same category through density correlation judgment, and the non-CBs are distinguished between boundary points and trivial points to improve the algorithm's processing ability and clustering quality for high-dimensional data. The findings denote that, in the KDD-CUP-99 dataset, the average accuracy of the noise application spatial clustering algorithm based on CB and density is 99.1%. The recall rate shall not be lower than the minimum of 98.2%. The average precision and F1-Score are 98.2% and 98.7%, respectively, and the silhouette coefficient is not less than 0.982. In the Choice Stock Trading Comprehensive Dataset, the accuracy of applying spatial clustering algorithms based on CBs and density noise is not less than 98%. The recall rate shall not be less than 97%. The accuracy and F1-Score are both not less than 98%, which is higher than other algorithms. The average silhouette coefficient and average probability Rand index are 0.986 and 0.927, respectively. The clustering quality of this algorithm is superior to other algorithms. The above results indicate that the noise-based spatial clustering algorithm based on CBs and density can achieve efficient and accurate clustering when processing high-dimensional financial data, and has good application prospects.

Povzetek: Predstavljena je metoda združevanja za finančne velike podatke izboljša DBSCAN z mrežnim drevesom in kotnim "pruningom" ter uvede Core Ball (CB) za boljšo obravnavo visoko-dimenzionalnih podatkov, pri čemer na KDD-CUP-99 in podatkih o trgovanju z delnicami dosega zelo visoko natančnost (≈98–99%+) in boljšo kakovost gručenja kot primerjalni algoritmi.

1 Introduction

With the acceleration of digital transformation in the financial industry, financial institutions have accumulated massive amounts of customer data, transaction records, and market information. These data have the characteristics of complexity and diversity, and traditional data analysis methods are difficult to effectively explore their potential value. Cluster analysis, as an unsupervised learning method, can help financial institutions discover potential patterns and relationships from massive data, providing data support for decision-making [1-2]. For example, through cluster analysis, financial institutions can segment customers, identify the behavioral patterns of different customer groups, and provide more personalized services. In the financial field, cluster analysis has a wide range of application scenarios, including risk assessment, fraud detection, market segmentation, investment portfolio optimization, etc. [3-4]. However, in the face of high-dimensional and large-scale financial data, traditional clustering algorithms often have problems such as high computational complexity and poor clustering performance. For example, the K-means clustering algorithm is sensitive to the selection of initial center points and difficult to handle non spherical clusters. Hierarchical clustering algorithm has low computational efficiency when processing large-scale data [5]. Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is algorithm that can identify clusters of any shape and is robust to noisy data. However, it has high computational complexity when processing large-scale datasets and performance bottlenecks in high-dimensional data. Therefore, to improve the clustering efficiency and quality of high-dimensional data (⩾30 dimensions), a financial big data clustering method based on Core Ball (CB) and DBSCAN is proposed. This method first utilizes a grid tree to improve DBSCAN by dividing the data space into non-empty grids and constructing a grid tree structure to achieve fast search of adjacent grids. At the same time, an angle-based pruning strategy is introduced to eliminate redundant trivial points using geometric angle relationships, reduce the number of distance calculations, and lower the computational complexity of the algorithm. On this basis, the concept of CBs is further introduced, and the dataset is divided into two categories: CBs and non-CBs. CBs are merged into the same category based on density correlation, while non-CBs distinguish between boundary points and trivial points to improve the algorithm's processing ability and clustering quality for high-dimensional data. The innovations of the study are: firstly, DBSCAN is improved by utilizing a grid tree, while an angle-based pruning strategy is introduced to enhance the computational efficiency of the algorithm. Unlike traditional tree structure indexing, CB-DBSCAN utilizes a grid tree structure to optimize the search process, quickly locates adjacent grids through hierarchical spatial segmentation, and adopts an angle based pruning strategy to further reduce the number of distance calculations. Then CB is introduced to enhance the algorithm's ability to handle high-dimensional data. The contribution of the research lies in breaking through the limitations of low efficiency in traditional DBSCAN high-dimensional search, achieving fast positioning of adjacent grids through grid tree hierarchical indexing, and combining geometric judgment of angle pruning to eliminate redundant calculations, providing a new paradigm of "low complexity search" for high-dimensional density clustering. Meanwhile, to address the issue of bias in estimating the density of high-dimensional data, a CB partitioning and density direct connection rule is proposed to solve the clustering bias of traditional algorithms (such as fuzzy C-means and Persistent Homology-based Clustering (PHBC)) in non-spherical clusters and noisy data, and to improve the theoretical system of high-dimensional data density clustering.

1.1 Related works

Financial institutions have accumulated massive amounts of customer transaction records, market data, credit records, and so on. In massive data, valuable information often only accounts for a small portion, so effective analysis methods are needed for clustering analysis. Maulana D J et al. proposed data clustering methods using K-means and synthetic few oversampling techniques to address the issue of data imbalance in financial distress classification. The findings denoted that after introducing the above method, the accuracy of support vector machine could reach 99.1%, and the F1-Score increased to 99.91%, indicating that this method can effectively balance data and significantly improve classification performance [6]. Hunag X et al. proposed a hybrid prediction method based on fuzzy C-means clustering algorithm and convolutional neural network for the clustering and prediction of corporate financial performance. This method first performed fuzzy C-means clustering and preprocessed the clustering labels. Next, the processed data and labels were introduced into a convolutional neural network to predict financial performance levels. The results showed that this method outperformed other machine learning models in clustering and prediction abilities, proving that the model can be applied to predict corporate financial performance [7]. Wang H. proposed a feature mining model for financial product purchasing behaviour data, based on the perspective of consumer behaviour. This was developed to address the problems of low accuracy, poor precision, and low recall rate, which have been identified in traditional purchasing behaviour data feature mining models. This method first extracted and preprocesses financial product purchasing behavior data, then constructed consumer behavior preference functions and used regression functions for quadratic processing, and finally applied the term frequency reverse document frequency algorithm and K-means clustering analysis method to extract features. The findings denoted that the recall rate of this method reached 98.74%, the accuracy rate was 96.51%, and the mining accuracy rate was 98.65%. The obtained mining results have high reliability [8].

DBSCAN, as a clustering algorithm based on spatial proximity, works directly in the data space without the need to calculate the distance matrix between data points like other algorithms. Therefore, it is robust to noisy data and does not require specifying the number of clusters in advance. Kaduskar V et al. proposed a hybrid clustering strategy to address the ambiguity and overlap issues caused by inconsistent diagnoses and differences in patient conditions in medical data mining. This method employed the capacity of fuzzy C-means to manage fuzzy membership degrees, thereby enabling data points to belong to multiple clusters. Additionally, it identified and managed noise through DBSCAN, while detecting clusters of varying densities and shapes. The results showed that the clustering quality of this method was significantly improved, which helped medical professionals better understand the data [9]. Meng Q et al. suggested a debris identification method based on DBSCAN algorithm to address the performance limitations of meshless methods in aerospace protective structure simulation. This method encompassed both geometric discrimination (spatial coordinates) and physical discrimination (velocity distribution), and involved in-depth discussions and validations on the key parameters of the DBSCAN algorithm. The findings denoted that this fragment recognition method was crucial for meshless methods, as it can prevent overly conservative estimates of fragment damage [10]. Yuan G and Ma Z designed an improved multi-scale dense crowd detection method based on You Only Look Once version 5 to address the issue of insufficient ability to accurately identify dense areas in multi-scale scenes. This approach enhanced the DBSCAN clustering algorithm, facilitating the identification of densely populated areas and enabling the identification of target populations at various scales.

The findings denoted that the accuracy of the detection results was about 70%, and by calculating the population density under the same scale conditions and visualizing dense areas, the problem of dividing crowded areas could be more accurately solved [11]. The summary of related works is shown in Table 1.

Table 1: The summary of related works

Literature	Dataset	Clustering accuracy	Limitation
[6]	Enterprise Financial Distress Dataset	99.1%	Unable to handle non-spherical clusters of financial data, significant clustering bias in high-dimensional scenarios
[7]	Financial Performance Dataset of Listed Companies	/	Fuzzy membership degree leads to fuzzy boundaries of financial data clusters
[8]	Financial product purchasing behavior dataset	96.5%	Sensitive to the initial center and unable to adapt to non spherical trading behavior clusters
[9]	Medical Diagnostic Dataset	/	Not targeted for high-dimensional optimization, running slowly on large-scale data
[10]	/	81.8%	Unable to fully explore hidden information
[11]	Crowd density detection dataset	70.0%	Only suitable for dense area recognition, without noise filtering mechanism

In summary, there is currently a lot of research on clustering algorithms for financial big data, but there are still problems with poor clustering quality and low accuracy. Therefore, to improve the clustering quality of financial big data, a clustering algorithm based on CB and DBSCAN is proposed. This algorithm first divides the dataset into CB and non-CB, and uses an angle-based pruning strategy to remove trivial points, thereby improving the clustering efficiency of financial big data.

2 Big data clustering method based on CB and DBSCAN

The complexity and diversity of data make it difficult for traditional data analysis methods to effectively explore its potential value. Therefore, a financial big data clustering method based on CB and DBSCAN is proposed in this study. This method first utilizes grid trees to improve DBSCAN, to enhance the computational efficiency of the algorithm. Then, CB is introduced to improve the algorithm's high-dimensional data processing capability.

2.1 Improved DBSCAN algorithm based on grid tree

In the financial field, cluster analysis can help financial institutions discover potential patterns and relationships from massive amounts of data, providing data support for decision-making. For example, through cluster analysis, financial institutions can segment customers, identify the behavioral patterns of different customer groups, and provide more personalized services. DBSCAN, as a density-based clustering algorithm, can recognize clusters of any shape and is robust to noisy data, effectively handling data points in high-dimensional space [12-13]. For DBSCAN, the definition of its ε neighborhood is shown in equation (1).

$$N_{\varepsilon}(m)=\{n\in D \mid d(m,n)\le \varepsilon\} \tag{1}$$

In equation (1), N_{ε} represents the ε neighborhood; ε represents the distance threshold; D represents the dataset; $d(m,n)$ denotes the distance between points m and n. The core point calculation formula of DBSCAN algorithm is shown in equation (2).

$$|N_{\varepsilon}(m)|\ge DT \tag{2}$$

In equation (2), DT represents the density threshold. Due to the high computational complexity of DBSCAN when processing large-scale datasets, to solve the above problems, research is conducted to improve it using grid trees. The improved DBSCAN divides the data space into non-empty grids, identifies core points and classifies the grids, then merges similar core grids, and finally assigns non core points [14-15]. The grid calculation formula is shown in equation (3).

$$g_i(j)=\left\lfloor \frac{c(j)-\min_{c\in C} c(j)}{\varepsilon/d} \right\rfloor \tag{3}$$

In equation (3), $g_i(j)$ represents the grid; $c(j)$ refers to the coordinate value of the data point in the j th dimension; C represents the dataset; d represents the dimensionality of the data. The core parameters of the grid tree in GrT DBSCAN include grid size (s) and grid non-empty decision threshold (Tgrid). The selection of parameters needs to balance "search efficiency" and "clustering accuracy". The specific selection criteria and optimal values are determined as follows: the selection of grid size (s) directly determines the granularity of data space partitioning, which needs to be determined in conjunction with the data dimension (d) and distance threshold (ε) (DBSCAN core parameters). In high-dimensional data, the sample distance is easily affected by the "curse of dimensionality" and tends to be consistent. If the grid is too large, it will cause multiple core points to be divided into the same grid, increasing the redundancy of neighborhood search; If the grid is too small, a large number of empty grids will be generated, reducing the space utilization of the grid tree. The selection of the Tgrid, which defines the minimum number of samples for a "non-empty grid", needs to be combined with the density threshold setting. The grid in two-dimensional space is shown in Figure 1.

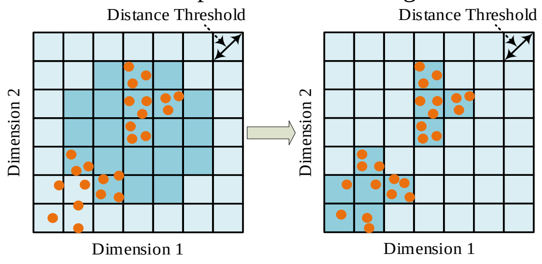


Figure 1: Grid in two-dimensional space

As shown in Figure 1, when searching for non empty grids, the identifiers of each point corresponding to the grid are first calculated and sorted accordingly. Specifically, the points will be sorted in ascending order based on the value of the grid identifier. At this point, it is only necessary to traverse the sorted data once to find all non empty grids [16]. For two grids, if their distance is less than the distance threshold, they are neighbors to each other. The distance calculation formula for the grid is denoted in equation (4).

$$d_g(g_i,g_j)=\sqrt{\sum_{t=1}^{d}\left(\frac{\varepsilon}{\sqrt{d}}\max\left\{|g_i(t)-g_j(t)|-1,0\right\}\right)^2} \tag{4}$$

In equation (4), $d_g\left(g_i,g_j\right)$ represents the distance between grids g_i and g_j. Due to the significant influence of data dimension on the number of neighboring grids, grid trees are introduced in the study to achieve fast search for neighboring grids. Grid tree is a data structure designed to efficiently manage and query grids in multidimensional spaces. The grid tree structure is shown in Figure 2.

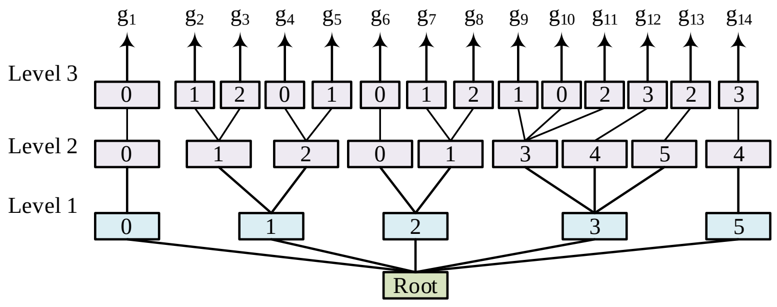


Figure 2: Structure of the grid tree

As shown in Figure 2, the grid tree has a hierarchical organizational structure, with each node representing a grid or a group of grids. It is constructed by recursively dividing the space, with each node corresponding to a specific spatial region. A grid tree consists of root nodes and leaf nodes, and the path identifier from the root node to the leaf node is composed of the keys of each layer node. When using the grid tree to find the nearest neighbor grid, if the child node is greater than the minimum number threshold, the child node with the smallest key value is selected; otherwise, all child nodes are traversed. For each node, the offset calculation formula is shown in equation (5).

$$o_{n'}=o_n+\max\{|KEY(n')-g(j)|-1,0\}^2 \tag{5}$$

In equation (5), $o_{n'}$ and o_n represent the offsets of child nodes and their neighboring nodes; n and n' represent different child nodes; $KEY(.)$ represents the key value. If the offset of a node is greater than or equal to the shortest distance, its descendants that do not include neighboring grids of the given grid can be removed. By using the above method, GrT-DBSCAN is constructed, which divides non-empty grids into core grids and non-core grids, enabling the identification of data core points. Next, it is necessary to merge the core grids of the same class to obtain the final category. For the task of merging grids, the key lies in how to remove trivial points through pruning. A pruning strategy based on angles is proposed for this study, with the core idea of utilizing geometric angle relationships to reduce the number of distance calculations. The formula for calculating the maximum angle of any point in the core point set that does not belong to the ε neighborhood is shown in equation (6).

$$\varphi_y = \arcsin\frac{\varepsilon}{d(p,y)} + \arccos\frac{\overrightarrow{pq}\cdot\overrightarrow{pq}}{d(p,q)\times d(p,y)} \quad (6)$$

In equation (6), ϕ_y represents the maximum angle; $d(p,y)$ represents the distance between two points; $\overrightarrow{pq}$ represents the tangent vector from point p to a circle centered on y; $\overrightarrow{py}$ represents the vector from point p to point y. If the angle between the vector from point p to point q and the vectors from point p to other points in its ε neighborhood is greater than ϕ_y, then point q is considered a non-nearest neighbor point, i.e. a trivial point, and should be deleted. Here, ϕ_y is calculated based on the distribution of points within the ε neighborhood of point p, and is used to guide the deletion decision of point q. The formula for calculating the cutting point is shown in equation (7).

$$q = \arg\min_{y\in s_j} d(p,y) \quad (7)$$

In equation (7), q represents the tangent point between $\overrightarrow{pq}$ and the circle centered on y; s_j represents the set of core points in the j th grid. The merging of similar grids can be achieved through the above method. The GrT-DBSCAN process is shown in Figure 3.

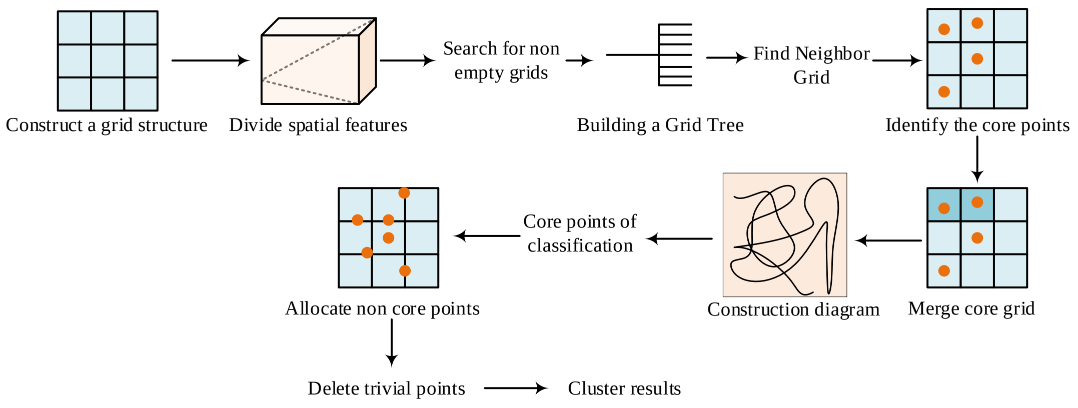


Figure 3: Process of GrT-DBSCAN

As shown in Figure 3, the algorithm first constructs a grid structure, dividing the entire data space into multiple grid cells. After constructing the grid structure, the data space is then finely divided to ensure that each data point is assigned to the corresponding grid. Next, the algorithm will search and identify each non empty grid one by one, and establish a grid tree structure based on the distribution and hierarchical relationship of the non-empty grids. Next, the nearest neighbor grids of each grid are found and the core points are identified. Then, the types of core points are identified and the core grids of the same type are merged. Next, each non-core point is assigned to determine whether it is a boundary point or a trivial point, and trivial points are removed. Finally, the clustering results are output. The pseudocode of GrT DBSCAN is shown in Table 2.

Table 2: The pseudocode of GrT DBSCAN

Pseudocode
Input: Dataset D (sample size N, dimension d), distance threshold ε, density threshold DT, grid size s, non empty grid threshold Tgrid Output: Cluster result C (including core cluster and noise) Step 1: Grid Data Space 2. Initialize the grid set G=∅ 3. For each sample p ∈ D: 4. Calculate the grid coordinates of p in each dimension: grid_comrd (p)=(⌊p₁/s⌋, ⌊p₂/s⌋,..., ⌊pud/s⌋) 5. Assign p to the corresponding grid g-grid_comod (p) 6. Count the number of samples in each grid and screen for non empty grids with G_nonempty={g | count (g) ≥ Tgrid} 7.//Step 2: Build a grid tree 8. Initialize the grid tree T (with the root node being the entire data space) 9. Divide the Gnonempty into grid coordinate levels (from the 1st dimension to the d-th dimension): 10. If the current node contains more than 2 * Tgrid samples: 11. Split the node into 2 child nodes based on the median of the current dimension 12. Repeat step 10 until all leaf nodes correspond to a single non empty grid g ∈ G_nonempty 13. Assign a path identifier KEY (a combination of coordinate key values from the root node to the leaf node) to each leaf node 14.//Step 3: Angle pruning and core mesh determination 15. Initialize the core grid set with G_core=∅ 16. For each grid g ∈ G_nonempty: 17. Randomly select one sample p from g as the reference point 18. Calculate the ε neighborhood N_ ε (p) of p, if count (N_ ε (p)) ≥ DT: 19.//Angle pruning: Remove trivial points in the non p neighborhood of g 20. Calculate the tangent vector pq from p to the CB

```
(Formula 7), and the maximum angle φ _max (Formula 6)
21. For each sample q ∈ g and q Δ N_ ε (p):
22. Calculate the angle between vector pq and py (the vector from p to q)
23. If φ>φ _max: mark q as a trivial point and remove it from g
24. Add g to the G_core
25.//Step 4: Core grid merging and non core point allocation
26. Initialize the visited grid collection with G_visited=∅, cluster C=∅
27. For each grid g ∈ G_core and g Δ G_visited:
28. Initialize the current cluster c={g}, queue Q=[g]
29. Mark g as visited (add G_visited)
30. When Q is not empty:
31. Take out the grid g-cur ∈ Q
32. Search for the neighboring grids of g2cur through the grid tree T (formula 4: grid distance<ε)
33. If g.neighbor ∈ G_core and g.neighbor Δ G_visited:
34. Add gneighbor to c and Q, mark as visited
35. Add c to C
36.//Step 5: Noise determination
37. For each sample p ∈ D:
If p is in a grid Δ G_core and p does not belong to any ε neighborhood of the core grid:
39. Mark p as noise
40. Return clustering result C
```

2.2 Financial big data clustering method based on CB-DBSCAN

Although the GrT-DBSCAN method mentioned above can effectively reduce computational complexity and improve data clustering efficiency, it still has performance bottlenecks when facing high-dimensional financial data. Therefore, to enhance the clustering ability of the algorithm for high-dimensional data, CB is introduced to improve the above algorithm and a financial big data clustering method based on CB-DBSCAN is constructed. The CB-DBSCAN algorithm divides high-dimensional datasets into CB and non-CB categories based on their relationships, thereby achieving distance measurement of high-dimensional data. The CB-based dataset segmentation method is shown in Figure 4.

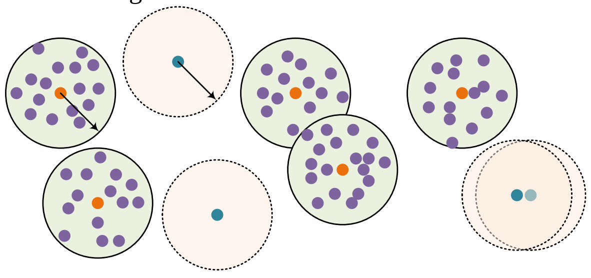


Figure 4: Data set segmentation method based on core ball

From Figure 4, for any point, if it is a core point, a CB containing all its neighboring points is generated; otherwise, a non-CB is generated. For any core point, at least one CB contains the point. It is worth noting that when performing dataset segmentation, the distance between any CBs must be greater than the CB radius [17-18]. Specifically, if there is a sample point $p \in D$ that satisfies the following two conditions, the CB $CB(p)$ is constructed with p as the center: (1) the number of samples contained in the ε neighborhood of point p is not less than the density threshold, that is, $|N_\varepsilon(p)| \geq DT$. Where $N_\varepsilon(p) = \{x \in D | d(p,x) \leq \varepsilon\}$, $d(x,p)$ represents the Euclidean distance between sample point p and x, and $|N_\varepsilon(p)|$ is the count of samples in the ε neighborhood. (2) The radius of the CB $CB(p)$ is $r(p)$ r (p), and it needs to completely cover the ε neighborhood of point p. At this point, the volume of the feature space covering all core points in the dataset is shown in equation (8).

$$V_F = (\max_{i=1}^{d} c_i + \varepsilon)^d \tag{8}$$

In equation (8), V_F represents the volume of the feature space; c_i represents the difference between the max and mini values of the i th dimensional coordinates of the core point. Based on the number of non-core points and the aforementioned feature space volume, the upper limit of CB quantity can be determined. After partitioning the dataset, the categories of each CB can be classified based on the mutual density between CB centers. If there is any CB directly connected by density, then all data points in both belong to the same category. The confirmation rule for direct density connection is as follows: (1) If there is a core point in the intersection of two CBs, then the two CBs are directly density connected. (2) If the distance between the center points of two CBs does not exceed twice the CB radius, the cardinality of set A is greater than 0, and the cardinality of set B is not less than the density threshold, then the two CBs are directly density connected. The sets A and B are shown in equation (9).

$$\begin{cases} A = \{p \mid p \in b_1 \cup b_2, \forall q \in b_1 \cup b_2 \\ d(p,q) \leq \max\left(\delta^2\varepsilon^2 - d_c^2/4, \varepsilon - \left(\delta^2\varepsilon^2 - d_c^2/4\right)\right)\} \\ B = \{p \mid p \in b_1 \cap b_2, \forall q \in b_1 \cap b_2 \\ d(p,q) \leq \left[\varepsilon - \left(\delta^2\varepsilon^2 - d_c^2/4, \varepsilon - \left(\delta^2\varepsilon^2 - d_c^2/4\right)\right)\right]\} \end{cases} \tag{9}$$

In equation (9), b_1 and b_2 represent two different CBs; δ represents the sparsity parameter of CB, the value range is [0.1,1], used to adjust the density judgment fault tolerance of the intersection area - when financial data is sparse, increasing δ can improve fault tolerance and avoid missing density connections; When financial data is dense, reducing δ can improve the strictness of judgment and avoid misjudging density connections; d_c represents the distance between the center points of two CBs. The above method can achieve accurate classification of CB, but due to the existence of a large number of unnecessary calculations, the efficiency of the algorithm is relatively low. Therefore, to further enhance the clustering efficiency of the algorithm, a new algorithm for detecting CB distance is proposed. After removing some trivial points in CB, this algorithm uses the pruning strategy mentioned above to determine whether the shortest distance between CBs does not exceed the distance threshold. If the distance between any point and the center point of CB is greater than the sum of the distance between data points within CB and the distance threshold, then that point is a trivial point [19-20]. The distance calculation formula for data points within CB is shown in equation (10).

$$d_{in} = \max\left\{d(x, c(b_i)) \mid x \in s_i\right\} \tag{10}$$

In equation (10), d_{in} represents the distance between the data points within CB and the center point; x represents any data point within CB. After simply deleting some trivial points, the above pruning strategy can be iteratively used to remove the remaining trivial points. One of the purposes of iterative pruning strategy is to identify and remove trivial points that are not important for forming tight clusters. By calculating d_{in}, it can be determined which points are located on the boundary of the CB, which may have a relatively small impact on the clustering structure. Therefore, it can be considered to mark them as trivial points and remove them in subsequent steps. Optimize the CB: After removing some trivial points, the CB may become more compact. By iteratively calculating new values of d_{in}, the definition of the CB can be further optimized to ensure that the remaining points are more tightly wrapped around the center point, thereby improving the quality of clustering. For the number of iterations required for the above detection algorithm, if the distance between the data points in both sets does not exceed the distance threshold, then the number of iterations required is 1; otherwise, let the upper limit of the distance satisfy equation (11).

$$l = \min_{p \in s_1', q \in s_2} d(p, q) - \varepsilon \tag{11}$$

In equation (11), l represents the upper limit of distance. At this point, the number of iterations is limited by the ratio of the volumes of d-dimensional balls K and H. If the volume ratio of two balls exceeds a certain threshold, they can be considered close enough to merge. The threshold is determined by equation (11). Specifically, if the l value is small, it means that the distance between the two datasets is close, so fewer iterations may be required to achieve the merging condition. On the contrary, if the l value is large, it means that the distance between the two datasets is far and may require more iterations. The formula for calculating the radius of a d-dimensional ball is shown in equation (12).

$$\begin{cases} r_K = \dfrac{\delta\varepsilon + l}{2} \\ r_H = \dfrac{l}{2} \end{cases} \tag{12}$$

In equation (12), r_K and r_H represent the radii of d-dimensional balls K and H, respectively. The above method can be used to construct a financial big data clustering method based on CB-DBSCAN. The process of CB-DBSCAN is shown in Figure 5.

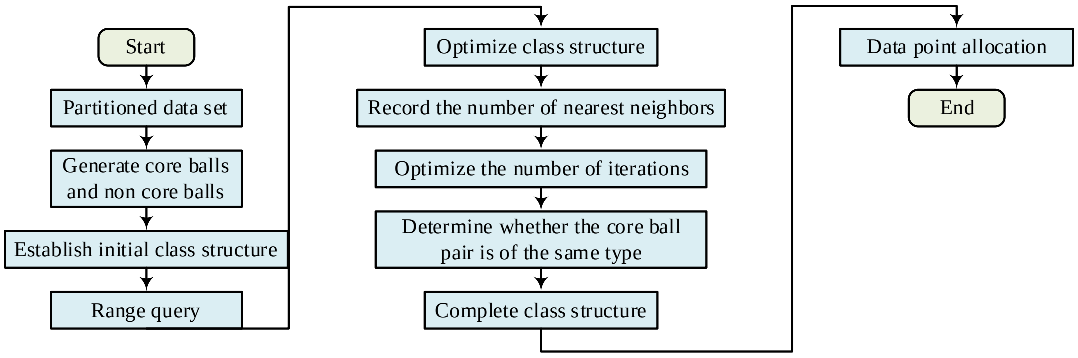

Start
Partitioned data set
Generate core balls and non core balls
Establish initial class structure
Range query
Optimize class structure
Record the number of nearest neighbors
Optimize the number of iterations
Determine whether the core ball pair is of the same type
Complete class structure
Data point allocation
End

Figure 5: Process of CB-DBSCAN

As shown in Figure 5, CB-DBSCAN first divides the dataset into CB and non-CB to preliminarily identify high-density and low-density regions in the data, and based on this, establishes the initial class structure. Next, the algorithm further classifies the structure by performing range queries to ensure that the density within each cluster is sufficiently high. At the same time, the algorithm will record the number of neighbors for each data point to reduce the number of subsequent iterations. Then, for the CB pairs that have not yet been identified as belonging to a category, the above detection algorithm is used to determine whether the two belong to the same category. Finally, the final clustering result can be obtained by assigning all data points. The time complexity of the algorithm is shown in equation (13).

$$O\left(Nf(I) + T_d V_F I\right) = O\left(Nf(I) + T_d h^d I\right) \tag{13}$$

In equation (13), $O(.)$ represents time complexity; N represents the size of the dataset; I represents the input scale; T_d represents the max amount of iterations for the detection algorithm; h represents the hash coefficient; $f(I)$ represents low order functions related to dimensions. From the above equation, as the number of samples increases, the $Nf(I)$ component that constructs the grid tree and executes the hierarchical index will dominate the asymptotic behavior of the algorithm, and the overall asymptotic time complexity of the algorithm is $O(N \cdot I \cdot \log I)$. In the context of financial big data, it exhibits a "linear growth" characteristic and has significant efficiency advantages over traditional algorithms.

3 Analysis of big data clustering test results

3.1 CB-DBSCAN clustering performance test results

To verify the clustering performance of the CB-DBSCAN algorithm, it was tested and compared with the Aggregate Multi-metric Graph Index (AMMGI) algorithm and the Fuzzy Coot Optimization K-Harmonic Means (FCOKHM) algorithm. The dataset used in the experiment was the KDD-CUP-99 dataset, which is a 34-dimensional dataset containing 5 million network connection records, each with 41 features. Although the KDD-CUP-99 dataset was not originally designed for financial data analysis, its rich features and large-scale samples made it an ideal choice for testing algorithm performance when processing high-dimensional data. To meet the needs of financial data analysis, the study performed the following preprocessing steps on the dataset: (1) Feature selection: selected features related to financial transactions, such as service type, protocol type, source address, and destination address. (2) Data cleaning: Removed missing and outlier values to ensure data quality. (3) Feature scaling: The features have been standardized to eliminate the influence of different feature dimensions. (4) Data segmentation: Divide the dataset into a training set and a testing set to evaluate the algorithm's generalization ability. The features of the processed KDD-CUP-99 dataset are shown in Table 3.

Table 3: The features of the processed KDD-CUP-99 dataset

Feature name	Description	Data type
Service type	Network service type	Classification
Protocol type	Network protocol type	Classification
Source address	Source IP address	Continuous
Destination address	Target IP address	Continuous

The approximate parameter of CB-DBSCAN in the experiment was 0.01, the sparsity parameter of CB was 1, the density threshold was 50, and the distance threshold was [1000,7000]. The CPU used in the experiment was Intel core i5 12600, and the operating system was Windows 10. The study used multiple evaluation metrics to measure the performance of the CB-DBSCAN algorithm, including accuracy, F1-Score, and Silhouette Coefficient. The calculation formulas for F1-Score and contour coefficient are shown in equation (14).

$$\begin{cases} F1-Score = 2\times\dfrac{\Pr e\cdot \operatorname{Re} call}{\Pr e+\operatorname{Re} call} \\ Sil = \dfrac{b-a}{\max(a,b)} \end{cases} \tag{14}$$

In equation (14), $\Pr e$ represents accuracy; $\operatorname{Re} call$ represents the recall rate; Sil represents the contour coefficient; a is the average distance from the sample to the nearest sample in the same category; b is the average distance from the sample to the nearest different category. The impact of distance threshold on the performance of CB-DBSCAN is shown in Figure 6.

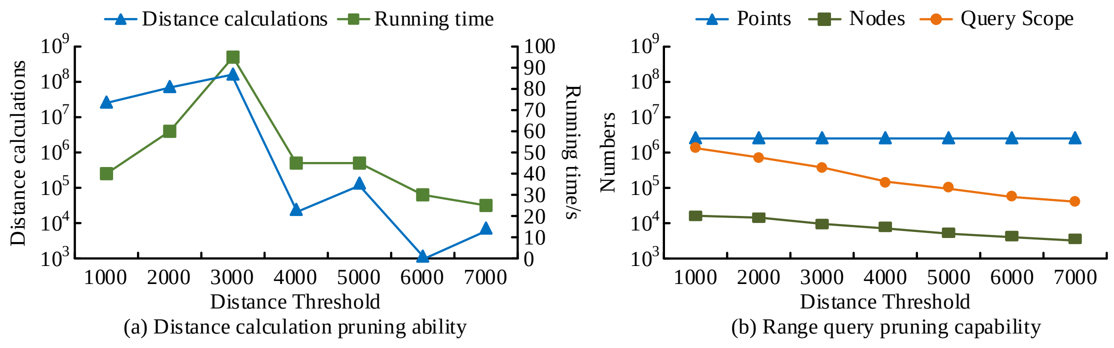


Figure 6: Influence of distance threshold on CB-DBSCAN performance

As shown in Figure 6 (a), with the increase of the distance threshold, the distance calculation times of CB-DBSCAN first increased, then decreased, and then increased again, while the running time first increased and then decreased. When the distance threshold was 6000, the distance calculation times were the least, only 1000 times, and its running time was 30 seconds. As shown in Figure 6 (b), in terms of range query capability, as the distance threshold increased, the point query times of CB-DBSCAN remained unchanged, while the node and range query times gradually decreased. When the distance threshold was 6000, the number of node and range queries was 5000 and 80000 times, respectively. Overall, when the distance threshold was 6000, the comprehensive performance of the model was better. The sensitivity analysis results of density threshold are shown in Table 4.

Table 4: The sensitivity analysis results of density threshold

Density threshold	Accuracy/%	Precision/%	F1-Score/%
30	98.2	98.9	98.2
40	98.7	98.5	98.5
50	99.1	98.7	98.9
60	98.9	98.3	98.6
70	98.5	97.8	98.1

According to Table 4, when the density threshold was 50, the various indicators of the algorithm basically reached their maximum values. At this point, its Accuracy, Precision, and F1-Score were 99.1%, 98.7%, and 98.9%, respectively. The clustering accuracy and recall of different algorithms are shown in Figure 7.

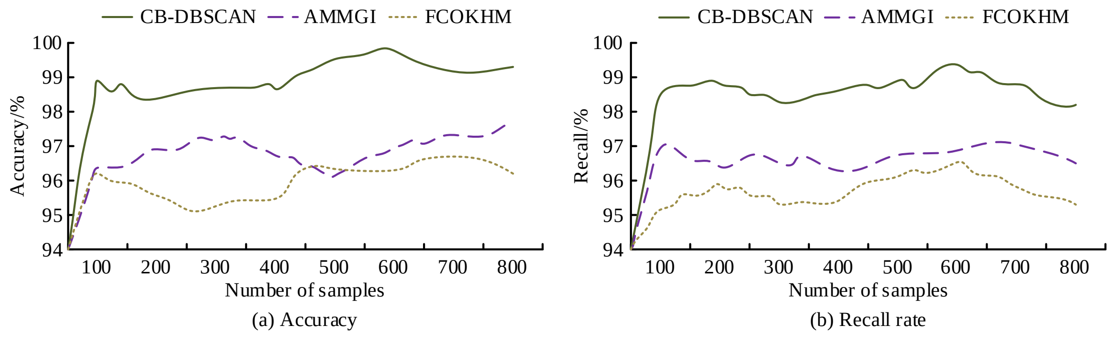


Figure 7: Cluster accuracy and recall rate of different algorithms

According to Figure 7 (a), the average clustering accuracy of AMMGI and FCOKHM was 96.9% and 96.0%, respectively. The clustering accuracy of CB-DBSCAN was always not less than 98%, and its average accuracy was as high as 99.1% (p<0.05). According to Figure 7 (b), the recall rates of AMMGI and FCOKHM were generally not more than 97%, with average recall rates of 96.7% and 95.7%, respectively. The lowest recall rate of CB-DBSCAN was 98.2%, with an average recall rate of 98.7% (p<0.05), which was also higher than other algorithms. This is because the CB-DBSCAN algorithm uses the concept of CBs to estimate the density of data points, which can more accurately measure the spatial density around the data points. Compared with traditional density estimation methods, this CB-based estimation approach can better adapt to data clusters of different shapes and sizes. The clustering accuracy and F1-Score of different algorithms are shown in Figure 8.

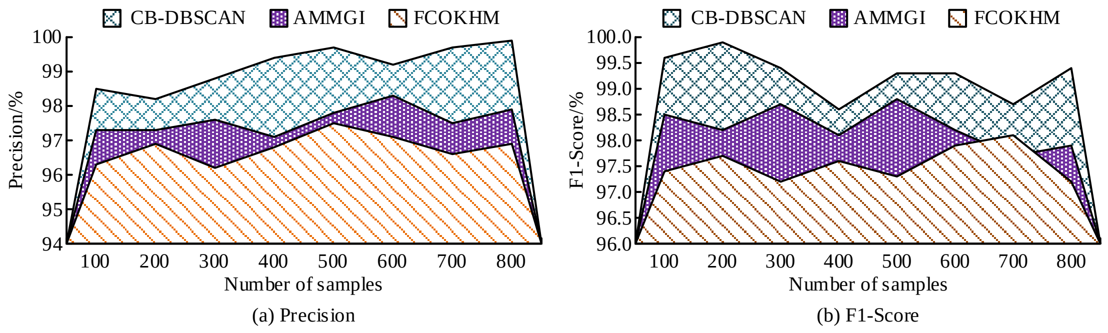
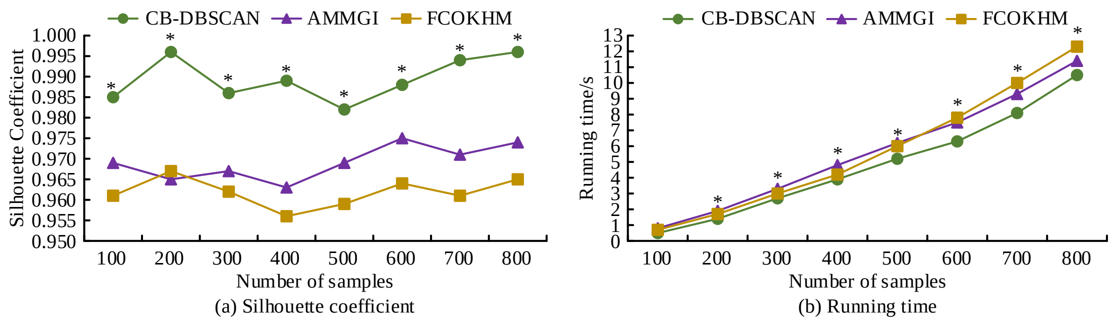


Figure 8: Clustering accuracy and F1-Score of different algorithms

According to Figure 8 (a), the clustering precision of AMMGI and FCOKHM was generally within the range of 96% -98%, with an average precision of 97.6% and 96.8%, respectively. The clustering precision of CB-DBSCAN was always not less than 98%, with an average precision of up to 99.2% (p<0.05). According to Figure 8 (b), the F1-Score ranges for AMMGI and FCOKHM were both 97%-99%, with an average F1-Score of 98.3% and 97.6%, respectively. The F1-Score of CB-DBSCAN was always not less than 98%, with an average F1-Score of 98.7% (p<0.05), which was higher than other algorithms. The above results indicate that CB-DBSCAN can achieve accurate clustering of high-dimensional data. The silhouette coefficients and running time of different algorithms are shown in Figure 9.



Note: * indicates *p*<0.05.

Figure 9: Profile coefficient and running time of different algorithms

According to Figure 9 (a), the silhouette coefficients of AMMGI and FCOKHM were the highest at 0.975 and 0.967, and the lowest at 0.963 and 0.956, respectively. The average silhouette coefficients were 0.969 and 0.962, respectively. The average contour coefficient of CB-DBSCAN was as high as 0.990 (*p*<0.05), consistently higher than other algorithms. As shown in Figure 9 (b), compared to other algorithms, CB-DBSCAN had a shorter running time. When the sample size was 800, the running times of AMMGI, FCOKHM, and CB-DBSCAN were 11.4s, 12.3s, and 10.5s (*p*<0.05), respectively. The above results indicated that CB-DBSCAN had excellent clustering performance and high computational efficiency. This is because the CB mechanism can adapt well to differences in data distribution, and for dense areas, it can identify tight cluster structures. For sparse regions, it can avoid incorrectly clustering isolated points into clusters.

3.2 Financial big data clustering test results

To verify the clustering performance of the CB-DBSCAN algorithm on financial data, the study used the financial dataset provided by the Choice Financial Information Terminal for testing. Choice Financial Information Terminal was a comprehensive financial data service platform that provides rich financial data resources, including trading records and market information of financial products such as stocks, funds, and bonds. In the study, the "Stock Trading Dataset" from Choice Financial Information Terminal was used, which includes stock trading records from 2010 to 2020. The specific dataset is named "Choice Stock Trading Comprehensive Dataset"(https://choice.eastmoney.com/product/datacenter). It was compared with PHBC and Feature Weighted Subspace Clustering (FWSC). The experimental parameters and hardware and software settings are shown in the previous section and will not be repeated. The clustering accuracy and recall of different algorithms are shown in Figure 10.

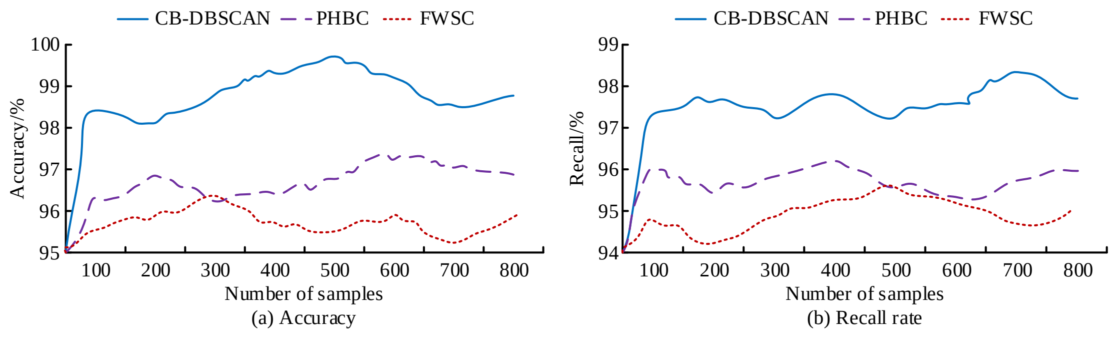


Figure 10: Cluster accuracy and recall rate of different algorithms

According to Figure 10 (a), the clustering accuracy ranges of PHBC and FWSC were 96%-98% and 95%-97%, respectively, with average accuracies of 96.8% and 95.7%, respectively. The clustering accuracy of CB-DBSCAN was the lowest at 98.1%, with an average accuracy of up to 98.9% (*p*<0.05), which was higher than other algorithms. According to Figure 10 (b), the recall rates of PHBC and FWSC were in the range of 95%-97% and 94%-96%, respectively, with average recall rates of 95.8% and 95.0%, respectively. The recall rate of CB-DBSCAN was always not less than 97%, with an average recall rate of 97.6% (*p*<0.05), which was superior to other algorithms. The clustering accuracy and F1-Score of different algorithms are shown in Figure 11.

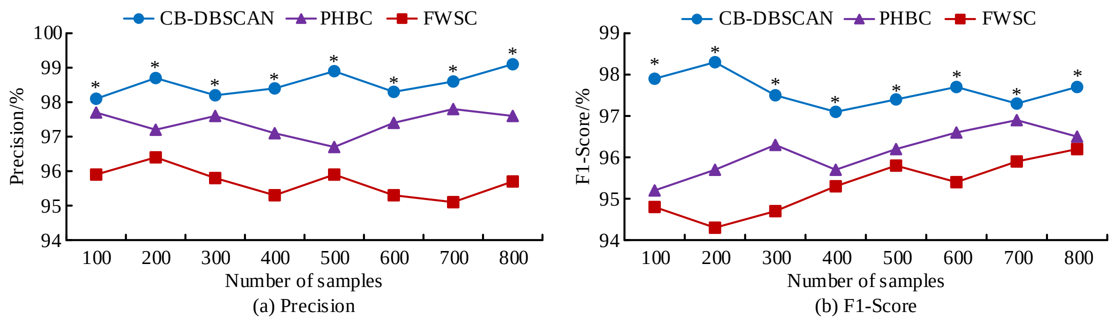


Figure 11: Clustering accuracy and F1-Score of different algorithms

According to Figure 11 (a), the clustering precision ranges of PHBC and FWSC were 97%-99% and 96%-98%, respectively, with average precision of 97.6% and 96.8%, respectively. The clustering precision of CB-DBSCAN was always above 98%, with an average precision of up to 99.2%. From Figure 11 (b), the F1-Score of PHBC and FWSC were the highest at 98.8% and 98.1%, and the lowest at 97.7% and 97.2%, respectively. The average F1-Score was 98.3% and 97.6%, respectively. The F1-Score of CB-DBSCAN was the lowest at 98.2%, with an average F1-Score of 98.7%, which was also higher than other algorithms. The above results indicate that CB-DBSCAN can achieve accurate clustering of financial big data. The silhouette coefficients and probability Rand indices of different algorithms are shown in Figure 12.

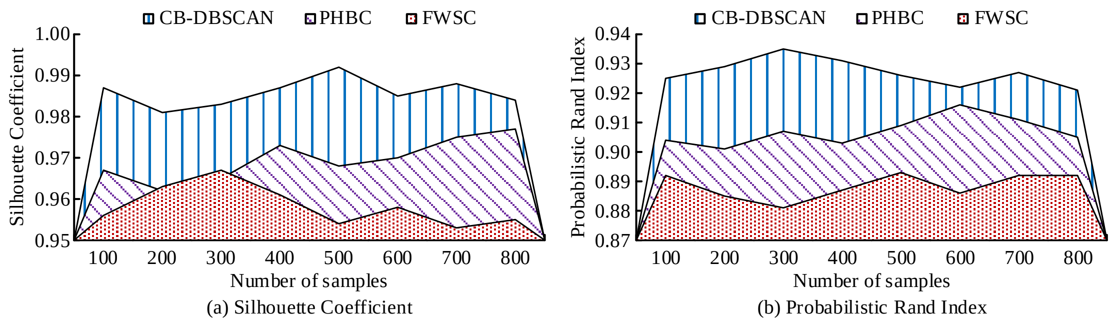


Figure 12: Silhouette coefficients and probability Rand index of different algorithms

According to Figure 12 (a), the silhouette coefficients of PHBC and FWSC were the highest at 0.977 and 0.967, and the lowest at 0.962 and 0.953, respectively. The average silhouette coefficients were 0.970 and 0.958, respectively. The silhouette coefficient of CB-DBSCAN was the lowest at 0.981, and the average silhouette coefficient was as high as 0.986, which was higher than other algorithms. According to Figure 12 (b), the probability Rand indices of PHBC and FWSC were the highest at 0.916 and 0.893, and the lowest at 0.901 and 0.881, respectively. The average probability Rand indices were 0.907 and 0.889, respectively. The lowest probability Rand index of CB-DBSCAN was 0.921, and the average probability Rand index was 0.927, which was higher than other methods. The above results indicate that CB-DBSCAN has high clustering quality for financial big data.

4 Discussion

Current research results show that CB-DBSCAN outperforms State of the Art (SOTA) methods in both high-dimensional data and financial big data scenarios. In the KDD-CUP-99 high-dimensional dataset (34 dimensions, 5 million samples), CB-DBSCAN has an average accuracy of 99.1%, an average recall of 98.7%, and an average F1-Score of 98.7%, which is 2.2-3.1% higher than AMMGI (accuracy of 96.9%, recall of 96.7%) and FCOKHM (accuracy of 96.0%, recall of 95.7%), respectively; The contour coefficient is 0.990, significantly higher than AMMGI's 0.969 and FCOKHM's 0.962, and the running time at a sample size of 800 is 10.5 seconds, which is shorter than AMMGI (11.4s) and FCOKHM (12.3s). In the Choice financial database, its average accuracy is 98.9%, recall rate is 97.6%, and F1-Score is 98.7%, which is superior to PHBC (accuracy 96.8%, recall 95.8%) and FWSC (accuracy 95.7%, recall 95.0%), with an average silhouette coefficient of 0.986 and a probability Rand index of 0.927, also leading the comparison algorithm.

The advantages of CB-DBSCAN stem from two core designs: firstly, the scalability of the grid tree, which divides the high-dimensional space into non-empty grids through hierarchical partitioning, and quickly locates adjacent grids with path identifiers, reducing the search complexity from O (N^2) to O (NlogN), solving the problem of inefficient search in AMMGI and PHBC high-dimensional environments, and reducing the increase in running time when the sample size increases; The second is the robustness of angle pruning, which is based on geometric angle judgment to remove trivial points (such as only 1000 distance calculations when the distance threshold is 6000), reduce redundant calculations while filtering noise, and avoid clustering bias caused by noise interference in FWSC and fuzzy C-means. Financial data still maintains an accuracy of over 98% even when containing 5% abnormal noise.

It is worth noting that the algorithm has three types of applicable boundaries: firstly, extremely sparse financial data (such as niche product customer data, with samples<100), where the number of neighboring samples is below the density threshold and cannot form a CB, making it easy to classify all samples as noise; The second type is mixed clusters with significant density differences (the density difference between ultra dense clusters and extremely sparse clusters is more than 10 times), and a globally unified density threshold can cause extremely sparse clusters to be misjudged as noise; The third is the dynamic financial data flow. Static clustering requires rebuilding the grid tree and CB, which cannot be incrementally updated, resulting in delayed response in real-time scenarios (such as adding 1000 transactions per second). In the future, scene adaptability can be improved through adaptive density threshold and integrated incremental framework optimization.

5 Conclusion

To achieve effective clustering of financial big data, a CB-DBSCAN-based clustering method for financial big data was proposed. The outcomes denoted that the CB-DBSCAN algorithm had significant performance advantages in processing high-dimensional financial data. The performance test results (KDD-CUP-99 dataset) showed that CB-DBSCAN outperformed the existing advanced methods AMMGI and FCOKHM, with an average accuracy and recall of 99.1% and 98.7%, respectively, and an average accuracy and F1-Score of 99.2% and 98.7%, respectively, both higher than other algorithms. In addition, the silhouette coefficient of CB-DBSCAN was the lowest at 0.982, and the average silhouette coefficient was as high as 0.990, indicating high internal consistency of its clustering results, shorter running time, and high computational efficiency. In practical application testing results (Choice Stock Trading Comprehensive Dataset), CB-DBSCAN also performed well. Compared to other algorithms, CB-DBSCAN had an average accuracy of 98.9%, an average recall of 97.6%, an average precision of 99.2%, and an average F1-Score of 98.7%, all of which are higher than other algorithms. The average silhouette coefficient and average probability Rand index were 0.986 and 0.927, respectively, which were also superior to other algorithms. The above findings denote that the CB-DBSCAN algorithm not only achieves accurate clustering when processing high-dimensional financial data, but also has high computational efficiency and clustering quality, making it suitable for clustering analysis of large-scale financial data. Although the CB-DBSCAN algorithm performs well on static datasets, its clustering results have poor quality when dealing with dynamically growing data. Therefore, in the future, it will consider introducing sliding window or CluStream algorithms to achieve high-quality and effective clustering of dynamic data.

References

[1] Du L, An X. An enterprise financial credit risk measurement method based on differential evolution algorithm. International Journal of Information Technology and Management, 2025, 24(1-2): 67-77. DOI:10.1504/IJITM.2025.144106.

[2] Sudha D, Gowri S. Deep FC-IIWO Fuzzy Clustering: An Optimized Fuzzy Clustering Approach for Big Data Clustering. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 2025, 33(03): 329-352. DOI:10.1142/S021848852550014X.

[3] Zhu J, Han X. Big data clustering algorithm of power system user load characteristics based on K-means and SOM neural network. Multimedia Tools and Applications, 2025, 84(10): 7477-7491. DOI:10.1007/s11042-024-19156-1.

[4] Ramaijane T, Hlomani H, Zlotnikova I, & Maupong T. Evaluation of Manifold Dual Contouring Algorithms Based on k-d tree and Octree Data Structures. Informatica, 2024, 48(3):399-418. DOI:10.31449/inf.v48i3.5420.

[5] Agarwal S, Reddy C R K. A smart intelligent approach based on hybrid group search and pelican optimization algorithm for data stream clustering. Knowledge and Information Systems, 2024, 66(4): 2467-2500. DOI:10.1007/s10115-023-02002-5.

[6] Maulana D J, Saadah S, Yunanto P E. Kmeans-SMOTE Integration for Handling Imbalance Data in Classifying Financial Distress Companies using SVM and Nave Bayes. Jurnal RESTI, 2024, 8(1): 54-61. DOI:10.29207/resti.v8i1.5140.

[7] Huang X, Hu Y, Liu H. A hybrid FCM-CNN method to cluster and forecast financial performance of listed companies. Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology, 2023, 44(2): 1991-2006. DOI:10.3233/JIFS-221995.

[8] Wang H. A feature mining model for financial product purchase behaviour data based on consumer behaviour perspective. International Journal of Business Intelligence and Data Mining, 2025, 26(1): 205-217. DOI:10.1504/IJBIDM.2025.143935.

[9] Kaduskar V, Srivastava A, Husain S O, Gupta M, & Patil V. Integrating Fuzzy C-Means and DBSCAN: A Hybrid Approach to Medical Data Mining. Fuzzy Information and Engineering, 2025, 17(1): 108-119. DOI:10.26599/FIE.2025.9270055.

[10] Meng Q, Fan J, Yuan Y, & Su Y. Fragment Identification Method Based on DBSCAN and Its Application in Hypervelocity Impact of Whipple Shield. International Journal of Aeronautical and Space Sciences, 2025, 26(3): 989-1000. DOI:10.1007/s42405-024-00814-5.

[11] Yuan G, Ma Z. Research on Dense Crowd Area Detection Method Based on Improved YOLOv5 and Improved DBSCAN Clustering Algorithm. Journal of Applied Mathematics and Physics, 2024, 12(12): 4206-4212. DOI:10.4236/jamp.2024.1212259.

[12] Regilan S, Hema L K. Optimizing environmental monitoring in IoT: integrating DBSCAN with genetic algorithms for enhanced clustering. International Journal of Computers & Applications, 2024, 46(1): 21-31. DOI:10.1080/1206212X.2023.2277966.

[13] Pan Y R, Xia Y H, Long L J, &Yang M L. Power-Line Extraction and Modelling from 3D Point Clouds Data Based on K-D Tree DBSCAN Algorithm. Journal of Electrical Engineering & Technology, 2024, 19(5): 3587-3597. DOI:10.1007/s42835-023-01641-6.

[14] Mosin V G, Kozlovskii V N, Nenashev M V. Novel Tools for Monitoring Process Quality: Detection of Anomalies in Digital Data on Content Consumption Using the DBSCAN Clustering Method. Russian Engineering Research, 2024, 44(9): 1370-1374. DOI:10.3103/S1068798X24702137.

[15] Yang W, Zhu Z, Zhou M, Li D, Zhang J, & Qin M, et al. Extraction of Tooth Cusps Based on DBSCAN Density Clustering and Neighborhood Search Algorithm. Critical Reviews in Biomedical Engineering, 2024, 52(2): 27-37. DOI:10.1615/critrevbiomedeng.2023050386.

[16] Wenying Z. An improved DBSCAN Algorithm for hazard recognition of obstacles in unmanned scenes. Soft computing: A fusion of foundations, methodologies and applications, 2023, 27(24): 18585-18604. DOI:10.1007/s00500-023-09319-x.

[17] Chen B Y, Luo Y B, Zhang Y, Jia T, Chen H P, & Gong J, et al. Efficient and scalable DBSCAN framework for clustering continuous trajectories in road networks. International Journal of Geographical Information Science, 2023, 37(8): 1693-1727. DOI:10.1080/13658816.2023.2217443.

[18] Ouf S, Ashraf M, Roushdy M. A Proposed Paradigm Using Data Mining to Minimize Online Money Laundering. Informatica, 2024, 48(3):309-328. DOI:10.31449/inf.v48i3.6103.

[19] Xing Z, Zhao W. Block-diagonal guided DBSCAN clustering. IEEE Transactions on Knowledge and Data Engineering, 2024, 36(11): 5709-5722. DOI:10.1109/TKDE.2024.3401075.

[20] Hauseux L. How can we theoretically measure the performance of density-based clustering algorithms. Performance Evaluation Review, 2025, 52(4): 19-20. DOI:10.1145/3725536.3725546.