# Clustering Algorithm for IoT Data Stream Based on K-Dimensional Tree and Self-Organizing Density

[a]Daoqu Geng, [b]Hao Liu

School of Automation / School of Industrial Internet,Chongqing University of Posts and Telecommunications,No. 2, Chongwen Road,Chongqing 400065, China.

[a]gengdq@cqupt.edu.cn, [b]s210301028@stu.cqupt.edu.cn

**Abstract.** With the development of IoT technologies, hundreds of millions of devices are constantly generating sensory data streams that contain a wealth of knowledge. To derive interoperable information from them, effective methods and techniques are needed to process and analyze the data streams. The stream clustering techniques in machine learning have gained increasing attention for its ability to rapidly discover knowledge and extract insights from data streams. In this paper, an IoT data stream clustering algorithm based on K-Dimensional tree and Self-Organizing density (KDSO) is proposed. The algorithm creates new clusters using KD trees to reduce the number of redundant clusters and performs range search quickly. In addition, it follows the idea of competitive learning to absorb new data points to facilitate the merging of micro-clusters. Meanwhile, it dynamically adjusts the clustering parameters for micro-cluster update and evolution. Experimental comparisons are made with other advanced methods. The results show that KDSO outperforms the compared methods in terms of clustering purity and silhouette coefficient, and shortens the clustering processing time, proving its good clustering performance.

**Keywords:** IoT, Discover Knowledge, Data Stream Clustering

## 1 Introduction

IoT infrastructure continuously generates data streams that require rapid analysis and mining of hidden knowledge to help make intelligent decisions in real time [1]. The data stream clustering techniques in the field of machine learning can fulfill this need by quickly discovering and summarizing clustering information to mine the contextual knowledge implicit in the data streams.

There are two types of data stream clustering algorithms, namely two-stage clustering algorithms and fully online clustering algorithms. The two-stage clustering includes online generation of micro-clusters and offline generation of macro-clusters [2] [3]. Compared to two-stage clustering algorithms, fully online clustering is more suitable to cope with the evolutionary characteristics of data streams and noise management [4]. In particular, the density-based stream clustering algorithm is applicable to IoT data

© The Author(s) 2024
T. Yao et al. (eds.), *Proceedings of the 2024 3rd International Conference on Engineering Management and Information Science (EMIS 2024)*, Advances in Computer Science Research 111,
https://doi.org/10.2991/978-94-6463-447-1_24

stream clustering as it is able to adapt to the evolutionary characteristics of data streams and can effectively handle noise in noisy environments [5] [6].

Several recent studies have extended density-based clustering algorithms to data stream. Among them, CODAS [7] is an online data clustering algorithm, but the micro-clusters are not updated with the dynamics of the data stream. On this basis, CEDAS [8] is improved by introducing energy conditions to reflect the data evolution characteristics. However, the CEDAS does not consider the storage of irrelevant micro-clusters and the extraction of relevant micro-clusters, while the fixed micro-cluster radius affects the clustering quality. BOCEDS [9] is an improved algorithm, which can adaptively update the micro-cluster radius and cluster center. Moreover, it introduces a buffering mechanism to achieve the upgrade and downgrade of micro-clusters. However, the BOCEDS may lead to the creation of redundant micro-clusters and ignores the case where data points are absorbed by multiple micro-clusters.

To address the above problems, an IoT data stream clustering algorithm based on K-Dimensional tree and self-organizing density (KDSO) is proposed in this paper to enhance the quality and efficiency of data stream clustering. The algorithm uses the KD tree to determine the generation of new clusters to reduce the number of redundant clusters and performs the range search quickly by exploiting its data structure characteristics. Secondly, the competitive learning idea based on self-organizing maps is used to deal with the process of absorbing newly arriving data points by target clusters, especially in the case where data points are mapped to multiple intersecting clusters, which promotes the merging of clusters through the movement of intersecting clusters to new data points. In addition, the algorithm can adaptively adjust the clustering parameters according to the dynamic changes of the data stream, thus enabling the update and evolution of micro-clusters. The effectiveness of the KDSO is verified by experimentally comparing this method with other advanced methods in three clustering metrics, namely, purity, silhouette coefficient and clustering processing time, using the KDDCup99 dataset.

The remainder of the paper is structured as follows: Section 2 specifies the proposed methodology. Section 3 gives comparative results with analytical discussion. Section 4 concludes the whole paper and provides an outlook for future work.

## 2 Proposed Method

Before describing the KDSO algorithm specifically, the definitions and concepts involved in the algorithm are first introduced.

**Data Stream**: Let the data stream $DS = \{x_1, x_2, \dots x_\infty\}$ be composed of a sequence of $d$-dimensional input vectors, where each data point $x(t) = (x^1, x^2, \dots x^d)$ represents a $d$-dimensional data record arriving at time $t$.

**Cluster Structure**: Define a micro-cluster as a six-tuple, denoted as Eq. 1, where

$$mc = (c, t, \omega, r_{core}, r_{shell}, b) \tag{1}$$

1.The center (c) denotes the cluster's positional core.

2.The time ($t$) indicates when the cluster was last updated.

3.The weight of a cluster ($\omega$) represents the count of data points within the cluster.

4.The core radius ($r_{core}$) measures the area of data concentration within the cluster and is mainly used to determine whether two clusters should be merged. It is defined using Eq. 2 and Eq. 3.

$$r_{core} = \frac{1}{d}\sum_{j}^{d}\sqrt{\frac{1}{\omega}\sum_{i=1}^{\omega}(x_i^j - \mu_i^j)^2} \tag{2}$$

$$\mu^j = \frac{1}{\omega}\sum_{i=1}^{\omega} x_i^j \tag{3}$$

5.The shell radius ($r_{shell}$) denotes the distance from the center of the cluster to the farthest data point, defining the boundary of the cluster and primarily used for operations such as merging and splitting of clusters.

6.The boolean value ($b$) denotes micro-cluster states: active ($b = 1,\ \omega \geq N$) or potential ($b = 0,\ \omega < N$), where $N$ is the weight threshold for creating micro-clusters within an initial radius $r$.

**Initial radius**: The initial radius $r$ serves as the minimum range for constructing micro-clusters, and its value is typically determined based on the specific problem and data distribution. If $r$ is too small it may not be able to cluster neighboring points correctly. Conversely, irrelevant points may be mistakenly grouped into the same cluster, affecting the clustering accuracy.

**Data expiration and deletion**: Every data has its life cycle and with the passage of time any data that reaches its life cycle is deleted. In the design of this algorithm, newly arrived data is added to a buffer queue with a fixed size $TN$. When the size of data in the buffer exceeds threshold $TN$, the data that has been in the buffer for the longest time will be removed according to the idea of FIFO [10].

The flowchart of the steps of the KDSO algorithm is shown in Fig. 1. As long as the stream remains active, the KDSO algorithm can constantly process incoming data streams. Note that it is hard to ascertain when streams are active or not, and expert knowledge is usually required to draw conclusions.

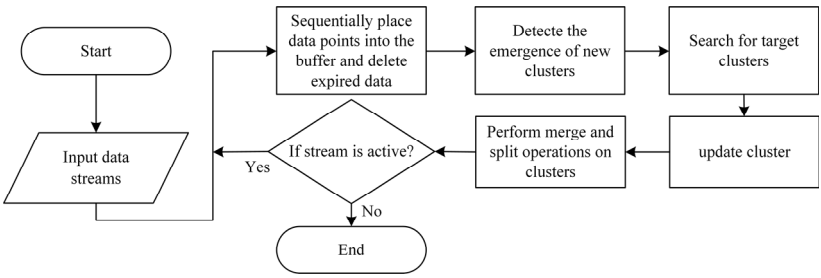


**Fig. 1.** Flowchart of KDSO algorithm steps

Step 1: Initialization and detection of micro-clusters. First, new data is deposited into a buffer of size $TN$, and the earliest data is removed when $TN$ is exceeded. Next, data points not assigned to existing clusters are filtered to construct the KD tree. Subsequently, a range search is performed in the KD tree using an initial radius $r$ and points

within the range are obtained. Finally, if the number of samples obtained surpasses or meets the weight threshold $N$, the cluster centers of the candidate clusters are calculated using Eq. 3, which is used to determine whether the distance between clusters satisfies the conditions for constructing new clusters.

Step 2: Search for target micro-clusters and try to absorb new data points. Determine whether the new data is mapped within the radius of an existing cluster based on Eq. 4. If the condition is satisfied, the idea of competitive learning is implemented by adjusting the cluster center to the vicinity of the new data point using a specific increment based on the Gaussian neighborhood function described in Eq. 6. When the new data point falls within the radius of multiple intersecting clusters, the merging of intersecting clusters is accelerated. After finding the target cluster, update the radius, center and weight of the cluster according to Eqs. 2, 5 and 7.

$$dis(x_{new}, mc) < r_{shell} \tag{4}$$

$$c(t+1) = c(t) + h(x, c(t)) \cdot (x - c(t)) \tag{5}$$

$$h(x, c) = \exp\left(-\| x - c \|^2 / 2r^2\right) \tag{6}$$

$$\omega(t+1) = \omega(t) + 1 \tag{7}$$

Step 3: Merging and splitting clusters. Clusters merge if one's shell radius intersects another's core radius. And before splitting the clusters, make sure that $\omega \geq 2N$ and $r_{core} \geq r$. Then, extract cluster data to build the KD tree and search. A split occurs if distance between candidate cluster (within $r$) and the center of the cluster formed by the remaining data exceeds the sum of the shell radius of these two clusters.

Step 4: Update the state of the micro-cluster. Over time, any data that reaches its lifecycle will be deleted from the buffer. If the deleted data belongs to an existing micro-cluster, the corresponding micro-cluster will also be updated. When the weight of an active cluster falls below the threshold $N$, the $b$ is set to 0, indicating that the cluster is updated to a potential cluster. When the weight of a potential cluster rises to the threshold $N$ and above due to the absorption of new data, reset $b$ to 1 and the cluster is updated to active cluster status.

## 3 Results and Discussions

In this section, the proposed KDSO algorithm is evaluated and its performance is compared with CEDAS and BOCEDS. T The experiments were conducted on a computer featuring 40 GB RAM and an i5-11500 processor, with implementation in the MATLAB 2016b environment. The dataset used is from KDDCUP99 and contains 50,000 samples, which is a frequently used dataset in data stream clustering studies. The experiments are based on three metrics for performance evaluation: Purity, Silhouette Coefficient and clustering processing speed.

Cluster purity is the percentage of dominant samples in each cluster. If there are $n_i$ samples in a cluster, where $n_i^d$ samples are the number of dominant samples with the largest share of the cluster, then the cluster purity can be computed with Eq. 8.

$$Purity = \frac{1}{k}\sum_{i=1}^{k}\frac{n_i^d}{n_i} \tag{8}$$

The silhouette coefficient evaluates the success of clustering by comparing the distance between samples within the same cluster and the distance between samples from different clusters, and its value ranges from $[-1,1]$. Silhouette coefficient nearing -1 indicate poor clustering results, while nearing 1 indicates better clustering results. The silhouette coefficient is expressed as Eq. 9.

$$s_i = \frac{b_i - a_i}{\max\{a_i, b_i\}} \tag{9}$$

Where $b_i$ indicates the mean distance between sample $i$ and all samples within a certain cluster. $a_i$ is the mean distance of sample $i$ from other points in the same cluster.

As shown in Fig. 2, the clustering purity of the three algorithms varies gradually with the continuous influx of data streams and decreases in the later stages. This is due to the fact that the few number of clusters at the initial clustering, but as the number of clusters grows with the continuous influx of data streams, the number of incorrectly assigned samples also increases. However, the KDSO algorithm maintains better and more stable clustering purity than the other two algorithms. This is because the KDSO algorithm not only follows the idea of competitive learning to promote the merging of micro-clusters, but also splits the clusters in time according to the threshold condition, which reduces the clustering of erroneous data points.

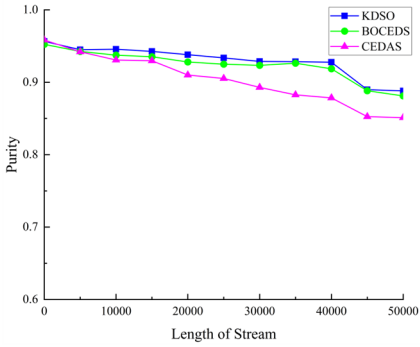


**Fig. 2.** Clustering purity comparison chart

As shown in Fig. 3, a comparison between the KDSO algorithm and the other two algorithms regarding the silhouette coefficients is demonstrated. Compared to the other two algorithms, KDSO possesses better ability to distinguish clusters. This is because the KDSO algorithm reduces the number of redundant clusters by searching through the range of the KD tree and performing the judgment of cluster spacing. In contrast, both CEDAS and BOCEDS adopt the strategy of constructing new clusters instantly, which does not take into account the distance of new clusters from other clusters. Furthermore, in comparison to CEDAS, the KDSO offers the benefit of adaptive updates to the radius and center of clusters, resulting in a more pronounced clustering effect.

As can be seen in Fig. 4, KDSO outperforms CEDAS and BOCEDS in terms of clustering processing time, which is due to the fact that KDSO utilizes the data structure of the KD tree to achieve a fast range finding while avoiding the creation and maintenance of redundant clusters. In contrast, both the CEDAS and BOCEDS algorithms require more time to maintain numerous clusters as the amount of data increases, and the KDSO algorithm reduces the overall clustering processing time by labeling clusters with the concepts of "activated state" and "potential state". Furthermore, some of the high time complexity steps are performed only under specific conditions (e.g., splitting clusters), so the performance will be better than the expected results.

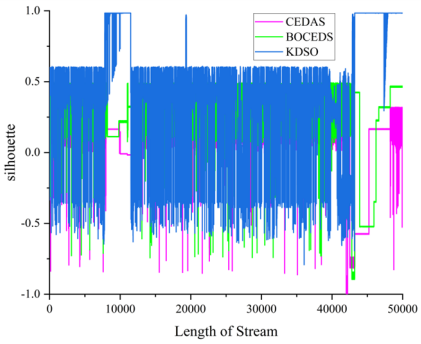


**Fig. 3.** Clustering silhouette coefficient comparison chart

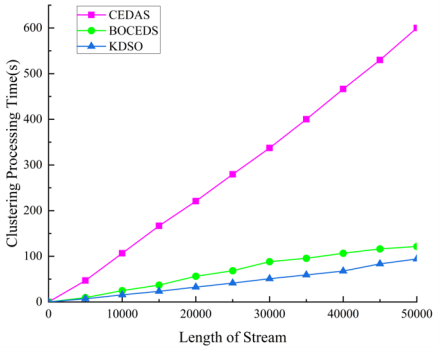


**Fig. 4.** Clustering processing time comparison chart

## 4 Conclusion

In this paper, an IoT data stream clustering algorithm based on KD tree and self-organizing density is proposed, which reduces the number of redundant clusters by using KD trees to detect the creation of new clusters and perform range searches quickly. In addition, it facilitates the merging of micro-clusters through competitive learning. Meanwhile, it dynamically adjusts clustering parameters for micro-cluster update and evolution. The proposed KDSO was experimentally compared with CEDAS and BOCEDS.

The results indicate that it outperforms the other methods in terms of purity, silhouette coefficients, and clustering processing time, demonstrating better clustering performance.

In future work, we will consider incorporating semantic web technologies to interpret the information implicit in the clustering results of IoT data stream and explore how to utilize ontology and inference mechanisms to enhance the knowledge discovery capability of data stream clustering algorithms.

## Acknowlegements

This work was sponsored in part by the National Key R&D Program of China (No.2022YFE0114300), and in part by the Natural Science Foundation of Chongqing, China (No. cstc2021jcyj-msxmX0330).

## References

1. Tu D Q, Kayes A S M, Rahayu W, et al. (2020) IoT streaming data integration from multiple sources. Computing, 102(10): 2299-2329. DOI: 10.1007/s00607-020-00830-9.
2. Aggarwal C C, Philip S Y, Han J, et al. (2003) A framework for clustering evolving data streams. Proceedings 2003 VLDB conference. Berlin, Germany. 81-92. https://doi.org/10.5555/1315451.1315460.
3. Carnein M, Trautmann H. (2018) evoStream – Evolutionary Stream Clustering Utilizing Idle Times. Big Data Research, 14: 101-111. https://doi.org/10.1016/j.bdr.2018.05.005.
4. Dai B R, Huang J W, Yeh M Y, et al. (2006) Adaptive clustering for multiple evolving streams. IEEE Transactions on Knowledge and Data Engineering, 18(9): 1166-1180. https://doi.org/10.1109/TKDE.2006.137.
5. Amini A, Wah T Y, Saboohi H. (2014) On density-based data streams clustering algorithms: A survey. Journal of Computer Science and Technology, 29: 116-141. https://doi.org/10.1007/s11390-014-1416-y.
6. Amini A, Saboohi H, Ying Wah T, et al. (2014) A fast density-based clustering algorithm for real-time internet of things stream. The Scientific World Journal, 2014(2): 6-18. https://doi.org/10.1155/2014/926020.
7. Hyde R, Angelov P. (2015) A new online clustering approach for data in arbitrary shaped clusters. 2015 IEEE 2nd international conference on cybernetics (CYBCONF). Chengdu, China. 228-233. https://doi.org/10.1109/CYBConf.2015.7175937.
8. Hyde R, Angelov P, MacKenzie A R. (2017) Fully online clustering of evolving data streams into arbitrarily shaped clusters. Information Sciences, 382: 96-114. https://doi.org/10.1016/j.ins.2016.12.004.
9. Islam M K, Ahmed M M, Zamli K Z. (2019) A buffer-based online clustering for evolving data stream. Information sciences, 489: 113-135. https://doi.org/10.1016/j.ins.2019.03.022.
10. Alamri A A, Syntetos A A. (2018) Beyond LIFO and FIFO: Exploring an allocation-in-fraction-out (AIFO) policy in a two-warehouse inventory model. International Journal of Production Economics, 206: 33-45. https://doi.org/10.1016/j.ijpe.2018.09.025

**Open Access** This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (http://creativecommons.org/licenses/by-nc/4.0/), which permits any noncommercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.