TWStream: Three-Way Stream Clustering: Supplementary Material

Jiarui Sun, Mingjing Du, *Member, IEEE,* Zhenkang Lew, and Yongquan Dong

1. Summary of Data Objects by Micro-clusters

In this section, we demonstrate how data objects in streams are summarized into micro-clusters. An example of this process is provided in Fig. S1. It is necessary to maintain two critical attributes to manage a micro-cluster incrementally online: the weight and center. The micro-cluster summary structure (shown in Fig. 2) introduced in the main text is divided into kernel region and shell region. As shown in Fig. S1, when a new data object x_1 arrives at time t, it must first find the nearest micro-cluster, mc_1. Then, the weight and center of mc_1 are updated in accordance with the distance d between x_1 and c_1 (the center of mc_1). As shown in Fig. S1, the weight influence of x_1 on mc_1 varies with respect to distance d (corresponding to Eq. (3)). The influence of x_1 on the weight of mc_1 is 1. Therefore, the weight of mc_1 should be updated according to Eq: $W(mc_1, t) = 2^{-\lambda(t-t_o)}W(mc_1, t_o) + 1$. In addition, since x_1 falls within the kernel region of mc_1, it is also necessary to update c_1 to obtain $c_1^{'}$ according to Eq: $c_1^{'} = (W(mc_1, t_o) \cdot c_1 + x_1)/W(mc_1, t)$. Following the above process, micro-cluster mc_1 finishes summarizing/absorbing data object x_1.

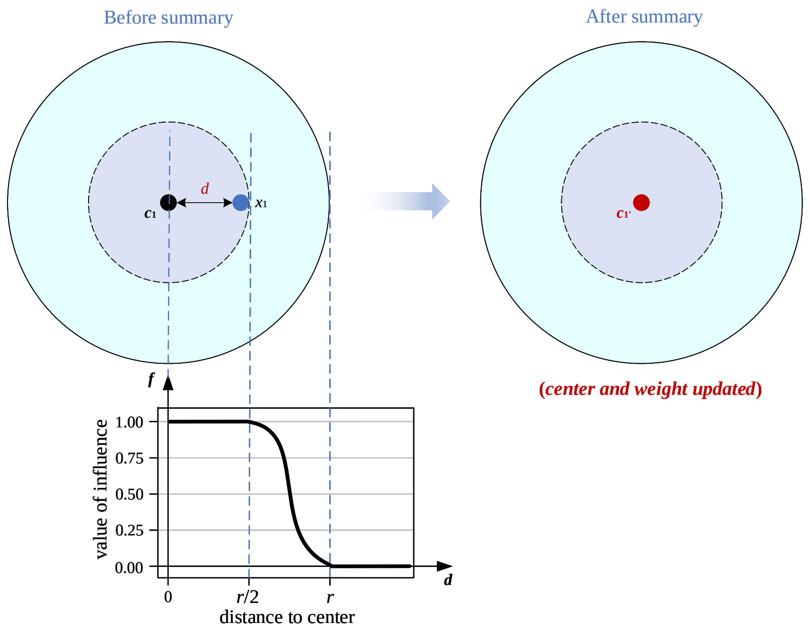


Fig. S1. An example of the micro-cluster mc_1 summarizing the data object x_1 from a stream.

2. Details on the Graph Manager

In a high-speed evolving data stream, the neighbourhood information of each active micro-cluster is continuously updated. To speed up the updating process, we design an *augmented knn graph* to avoid repetitive computations. It maintains a maximum of $2k$ nearest neighbors incrementally for each active micro-cluster. Fig. S2 illustrates the principle of updating. Before time t_1, the $2k$ ($k = 3$) nearest neighbors of the active micro-cluster mc_i are **a**∼**f**, where mc_i is closest to **a** and furthest away from **f**. Assume that at time t_1, an inactive micro-cluster **g** is reactivated. If **g** is closer to mc_i than **c**, insert it after **b**. Accordingly, **f** needs to be removed from the list of $2k$ nearest neighbors. During the data stream evolution, it is assumed that micro-cluster **b** fades from active status to inactive status at time t_2. We will remove it from the list and recalculate the kth nearest neighbor.

This work was supported in part by the National Natural Science Foundation of China under Grant 62006104 and Grant 61872168. *(Corresponding author: Mingjing Du.)*

J. Sun, M. Du, Z. Lew, and Y. Dong are with the School of Computer Science and Technology, Jiangsu Normal University, Xuzhou 221116, China. (e-mail: sunjr@jsnu.edu.cn; dumj@jsnu.edu.cn; 7547461@gmail.com; tomdyq@163.com).

A simple approach to determine the kth nearest neighbor is recalculating and ordering the distances from mc_i to each micro-cluster. It is evident that this approach is inefficient. To overcome the above problem, we maintain up to $2k$ nearest neighbors for mc_i. Fig. S2 shows that after removing **b**, micro-cluster **c** automatically becomes the kth nearest neighbor of mc_i. The micro-clusters that have moved will be deleted and reinserted. By using this redundancy strategy, we are reducing the negative effects of updating the $k$nn graph frequently.

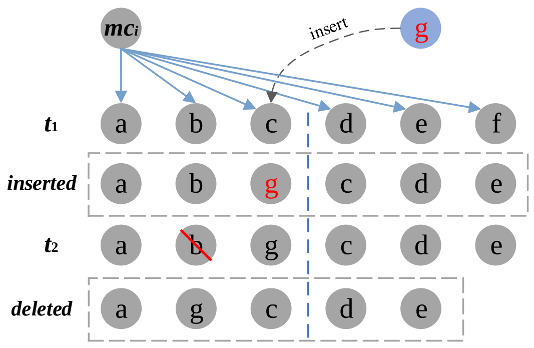


Fig. S2. An example of the augmented $k$nn graph update operation.

3. Descriptions of Synthetic Datasets and Real-world Datasets

To make experiments easily visualisable, we conduct experiments on three 2-dimensional synthetic datasets, including DS1, DS2 and RBF. We generate datasets DS1 and DS2 (each with 20,000 noise points) to verify the ability of TWStream to identify arbitrarily shaped clusters and multi-density clusters with ambiguous boundaries, respectively. As shown in Fig. 5(a), the DS1 dataset consists of eight clusters divided into four groups and produced sequentially. The DS2 dataset is comprised of two groups of Gaussian-distributed clusters with ambiguous boundaries. Figs. 5(b)-(c) illustrate the first group of clusters with equal radii and the second group of clusters with varying densities, respectively. The RBF[1] dataset is a non-stationary data stream that contains multiple Gaussian-distributed clusters that move rapidly in 2-dimensional space and overlap at certain times. Fig. 5(d) displays the initial segment of the stream extracted from the RBF dataset. In addition, we also use two high-dimensional synthetic datasets Hyperplane and SynEDC. The Hyperplane[2] dataset is generated using a random hyperplane generator. It is a data stream characterized by rapid incremental drift involving a 10-dimensional hyperplane that shifts both its position and orientation. The SynEDC[1] dataset has concept-drift and novel-class. It exhibits clusters that conform to Gaussian distributions with distinct means and variances.

Furthermore, we use some publicly available real-world datasets, including NOAAweather, Powersupply, Adult, Electricity, Insects, Rialto, Airlines, Poker, Covertype and Sensor, in our experiments to further validate the performance of the proposed algorithm. The datasets NOAAweather, Powersupply, Electricity, and Insects are from the USP Data Stream Repository[3]. The NOAAweather dataset comprises weather data gathered in Bellevue, Nebraska, from 1949 to 1999. Its primary objective is to forecast precipitation occurrence on specific dates. The Powersupply dataset encompasses hourly power supply data from the Italian electricity company. Typically, it is employed to predict the specific hour (out of 24) to which the current supply pertains. The Electricity dataset pertains to the electricity market in New South Wales, Australia, where prices are influenced by supply and demand dynamics. The two label values indicate price fluctuations (1 = increase, 0 = decrease). The Insects dataset comprises information collected from sensors installed in mosquito traps, measuring the flight patterns of captured insects. Each sample in this dataset contains 33 features extracted from the acquired signal spectrum. In this paper, we utilize an incrementally drifted and balanced variant of this dataset. Rialto and Poker are two real-world datasets accessible via the driftDatasets repository[4]. The Rialto dataset encompasses 27-dimensional RGB histograms derived from the colorful facades of ten buildings near Venice's renowned Rialto Bridge. These images are captured using a stationary webcam that recorded time-lapse footage over a span of 20 consecutive days in May-June 2016. In the Poker dataset, each entry represents a five-card draw from a standard 52-card deck. Every entry is characterized by ten attributes, as each card is defined by two distinct features: its suit and rank. The Airlines[5] dataset, sourced from Ikonomovska, predicts flight delays by analyzing the scheduled departure times of commercial flights operating within the United States. The Sensor[2] dataset comprises continuous data collected over a span of two months from 54 sensors deployed at the Intel Berkeley Research Lab. Each entry includes information such as

[1]https://github.com/Xicks/DataStream_DataSets/tree/master
[2]https://www.cse.fau.edu/~xqzhu/stream.html
[3]https://sites.google.com/view/uspdsrepository
[4]https://github.com/vlosing/driftDatasets
[5]http://kt.ijs.si/elena_ikonomovska/data.html

temperature, humidity, light levels, sensor voltage readings, and a unique sensor ID. The two remaining datasets, Adult and Covertype, are from the UCI Machine Learning Repository[6]. The Adult dataset is derived from the 1994 Census database and is used to predict whether an individual earns an annual income exceeding \$50,000. The Covertype dataset is employed to predict forest cover types in four wilderness regions within the Roosevelt National Forest in northern Colorado. These predictions are made using cartographic variables as input.

4. Evaluation Metrics

In our experiments, we use three metrics to evaluate the performance of TWStream and comparison algorithms. These metrics include Purity [1], *Normalized Mutual Information* (NMI) [2], and *Adjusted Rand Index* (ARI) [3].

For N distinct samples, $\mathbf{T} = \{T_1, T_2, \ldots, T_K\}$ are the ground-truth classes and $\mathbf{C} = \{C_1, C_2, \ldots, C_K\}$ are the clusters obtained by the stream clustering algorithm.

Purity is defined as:

$$Purity = \frac{1}{N} \sum_{k} \max_{j} |C_k \cap T_j| \tag{1}$$

where C_k is all samples in the kth cluster obtained by clustering. T_j is all samples in the jth ground-truth class. $Purity \in [0,1]$, and a larger value indicates better clustering.

Normalized Mutual Information is defined as:

$$NMI = \frac{MI(\mathbf{C},\mathbf{T})}{\sqrt{H(\mathbf{C})H(\mathbf{T})}} \tag{2}$$

where MI is the mutual information between $\mathbf{C}$ and $\mathbf{T}$, and $H(\cdot)$ is the entropy. $NMI \in [0,1]$, and a larger value indicates better clustering.

Adjusted Rand Index is defined as:

$$ARI = \frac{2(ad-bc)}{(a+b)(b+d)+(a+c)(c+d)} \tag{3}$$

where a is the number of pair of data objects which are in the same cluster in $\mathbf{C}$ and in the same class in $\mathbf{T}$. where b is the number of pair of data objects which are in different clusters in $\mathbf{C}$ and in the same class in $\mathbf{T}$. where c is the number of pair of data objects which are in the same cluster in $\mathbf{C}$ and in different classes in $\mathbf{T}$. where d is the number of pair of data objects which are in different clusters in $\mathbf{C}$ and in different classes in $\mathbf{T}$. $ARI \in [-1,1]$, the higher the ARI value, the better the clustering quality.

5. Parameters of All Algorithms

TABLE S1
Parameters of all algorithms.

Algorithm	#Num	#Params
TWStream	5	$r \in [0.001, 1)$, λ=0.0028, β=0.0021, $k \in [4, 16]$, $\tau \in [0.5, 0.8]$
EDMStream	5	$r \in [0.001, 1)$, a=0.998, λ=1, β=0.0021, $\delta \in [0.001, 1]$
CEDAS	3	$r \in [0.001, 1)$, *decay*=0.001, $minPts \in \{5, 10\}$
DBSTREAM	5	$r \in [0.001, 1)$, λ=0.0028, t_{gap}=100, $\alpha \in [0.001, 0.3]$, w_{min}=0.01
D-Stream	5	$len \in [0.01, 1)$, λ=0.998, C_m=3, C_l=0.8, β=0.3
DenStream	4	$\epsilon \in [0.001, 1)$, λ=0.0028, $\beta \in [0.5, 0.9]$, $\mu \in [2, 10]$
HDDStream	6	$\epsilon \in [0.001, 1)$, λ=0.0028, $\beta \in [0.5, 0.9]$, $\mu \in [2, 10]$, $\pi \in [2, 20]$, $\delta \in [2, 10]$
MR-Stream	9	$\epsilon \in [0.001, 1)$, a=1, λ=1.002, C_H=3, C_L=0.8, $H \in [2, 10]$, $h \in [2, 10]$, $\beta \in [1, 10]$, $\mu \in [1, 10]$
MuDi-Stream	4	$len \in [0.01, 1)$, λ=0.0028, $\alpha \in [0.5, 0.9]$, $minPts \in [3, 20]$
ESA-Stream	3	$len \in [0.01, 1)$, λ=0.998, *upbound*=1

6. Clustering Results of Algorithms on DS1 and DS2

Figs. S3-S4 illustrate the clustering results of all algorithms on synthetic datasets DS1 and DS2, respectively. Note that the clustering results are presented as micro-clusters or grids.

7. Ablation Experiments

Fig. S5 provides a visual representation of the results of ablation experiments.

[6]http://archive.ics.uci.edu/ml/datasets.php

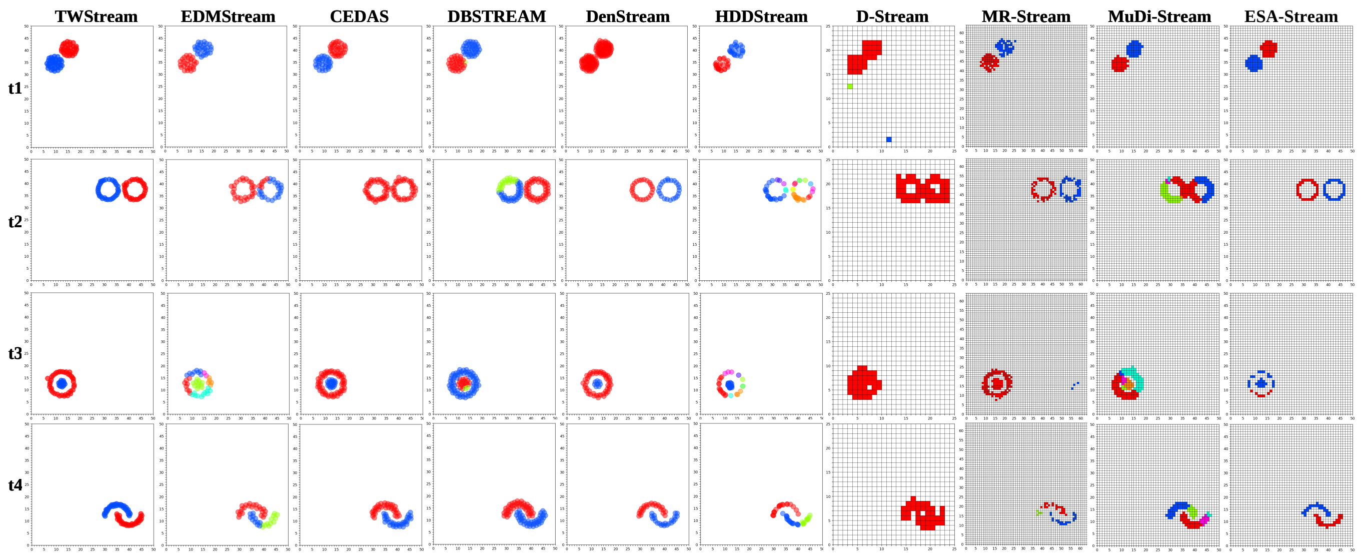


Fig. S3. Clustering results of algorithms on DS1.

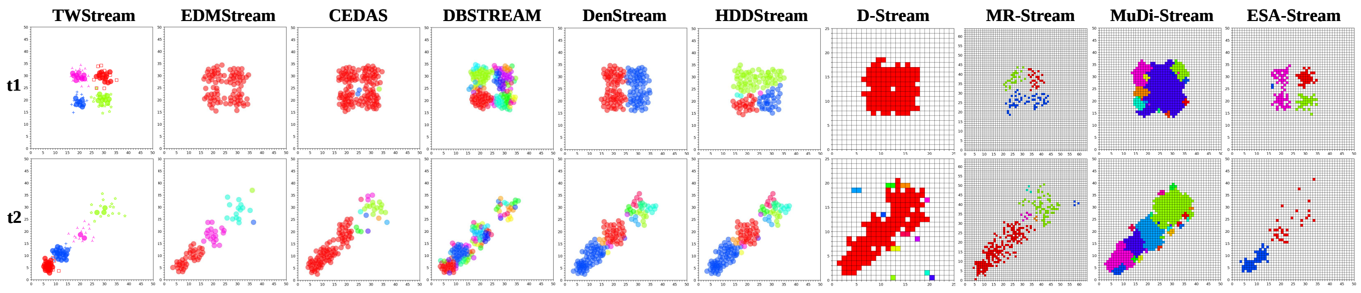


Fig. S4. Clustering results of algorithms on DS2.

8. Parameter Analysis

In this section, we investigate the clustering performance impact of all five parameters of the proposed algorithm: the number of nearest neighbors k, the boundary percentage factor τ, the micro-cluster radius r, the decay factor λ, and the weight threshold factor β. The experiments are conducted on three synthetic datasets, namely DS2, RBF, and Bench[7]. When analyzing one parameter, the other parameters are fixed.

Firstly, we examine the impact on clustering performance of two key parameters, namely, the number of nearest neighbors k and the boundary percentage factor τ. On both DS2 and RBF datasets, we fix τ to be 0.7 and choose k value in the range [4, 16]. From Fig. S6, we can see that as k increases, the Purity of the proposed algorithm decreases monotonically. In contrast, NMI and ARI first increase and then decrease. We speculate that the high Purity at the beginning may be caused by the over-partitioning of clusters due to the small k value. As k increases gradually, the local useful information decreases, decreasing the algorithm's performance. Overall, the proposed algorithm performs smoothly as the value of k increases. Similarly, k is fixed at 8 and τ is selected in the range of [0.3, 0.9] on both DS2 and RBF datasets. Fig. S7 demonstrates that the performance of the proposed algorithm generally shows an increasing and decreasing trend as τ increases. When the τ value is small, the algorithm cannot detect all potential boundary objects. Consequently, clusters with ambiguous or overlapping boundaries may be incorrectly merged. The clustering performance is high and stable when τ is between 0.5 and 0.8. However, as τ approaches 1, objects in the core regions are mistakenly identified as boundary objects, leading to a gradual decline in clustering quality. It can be observed that on the RBF dataset, the growth of τ has a low impact on the performance of the proposed algorithm. However, the growth of τ has a high impact on the algorithm on the DS2 dataset. This may be a result of the DS2 dataset's more clusters with ambiguous or overlapped boundaries. Therefore, TWStream must set a higher τ value to detect more boundary objects.

According to the experimental results on both the DS2 and RBF datasets, the parameters k and τ affect the clustering performance in a predictable manner. Generally, the recommended range for the number of nearest neighbors k is [4, 16], and the recommended range for the boundary percentage factor τ is [0.5, 0.8]. Further settings should be determined based on the

[7]https://github.com/Xicks/DataStream_DataSets

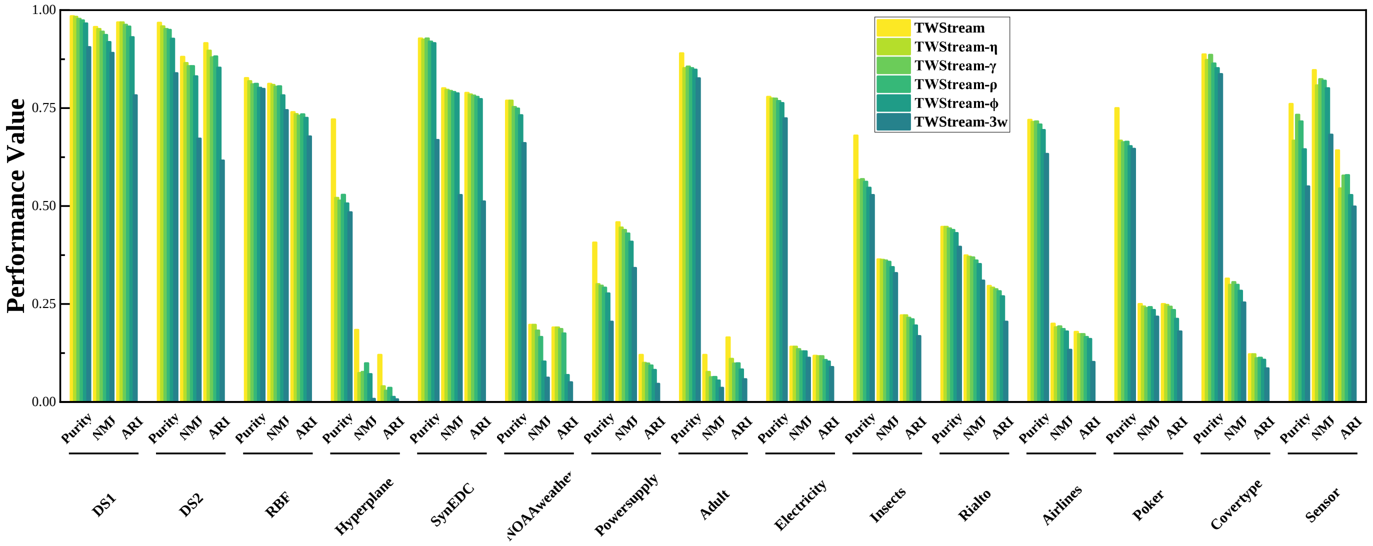


Fig. S5. A histogram of ablation experiment results.

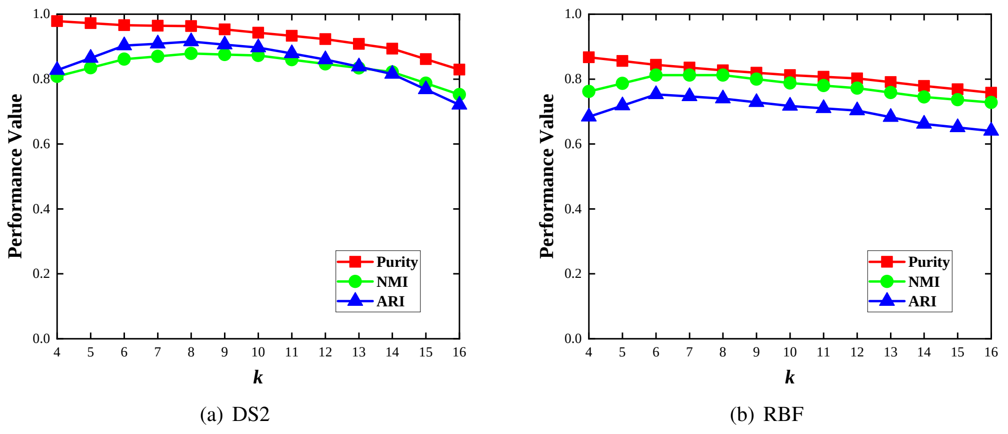


Fig. S6. Performance changes with varying k on DS2 and RBF.

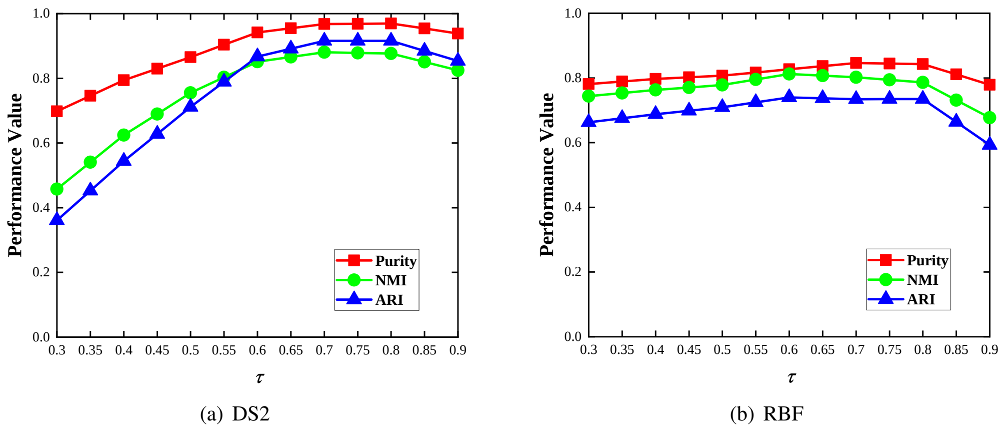


Fig. S7. Performance changes with varying τ on DS2 and RBF.

complexity of the data stream. When the boundaries of the clusters in a stream are more ambiguous or overlapped, a higher τ is required. For data streams without overlapping clusters, τ is set to 0.

Secondly, further analyses are conducted to examine the impact of the remaining three parameters on the performance of the proposed algorithm. On the DS2 dataset, we choose r value in the range [0.01, 0.1]. Fig. S8 illustrates the clustering results of TWStream at two different gap times (t1<t2) with varying r values. In addition, we monitor the clustering performance,

number of micro-clusters, memory usage, and running time of the proposed algorithm in Fig. S9. By observing Fig. S8 and Fig. S9(b), TWStream produces fewer micro-clusters as r increases. When r increases to a certain extent, the possibility of data objects from different clusters being absorbed into the same micro-cluster rises significantly, which leads to a decrease in clustering quality. As shown in Fig. S9(a), the Purity, NMI, and ARI gradually decrease with the increase of r. In addition, Figs. S9(c)-(d) show an overall decreasing trend in memory usage and running time as r increases. Briefly, data stream clustering algorithms must balance clustering quality against clustering cost (e.g., memory and time) to obtain clustering results of the highest possible quality within acceptable memory consumption and waiting time. To select an appropriate micro-cluster radius r, we should consider the complexity of the data stream. In the context of high-dimensional data streams, increasing the radius of micro-clusters is common practice to cope with the sparsity of samples in high-dimensional spaces.

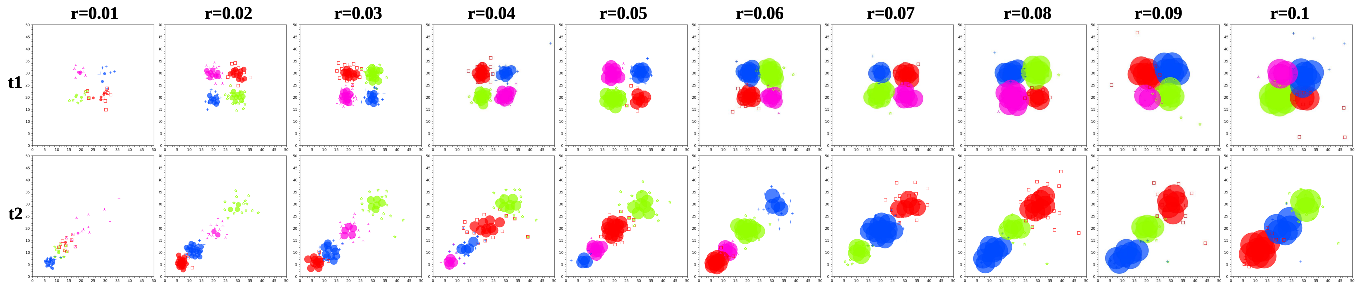


Fig. S8. Clustering results of TWStream with varying r on DS2. Each row shows the clustering results with varying r at the same gap time, where the 2 gap times satisfy t1<t2.

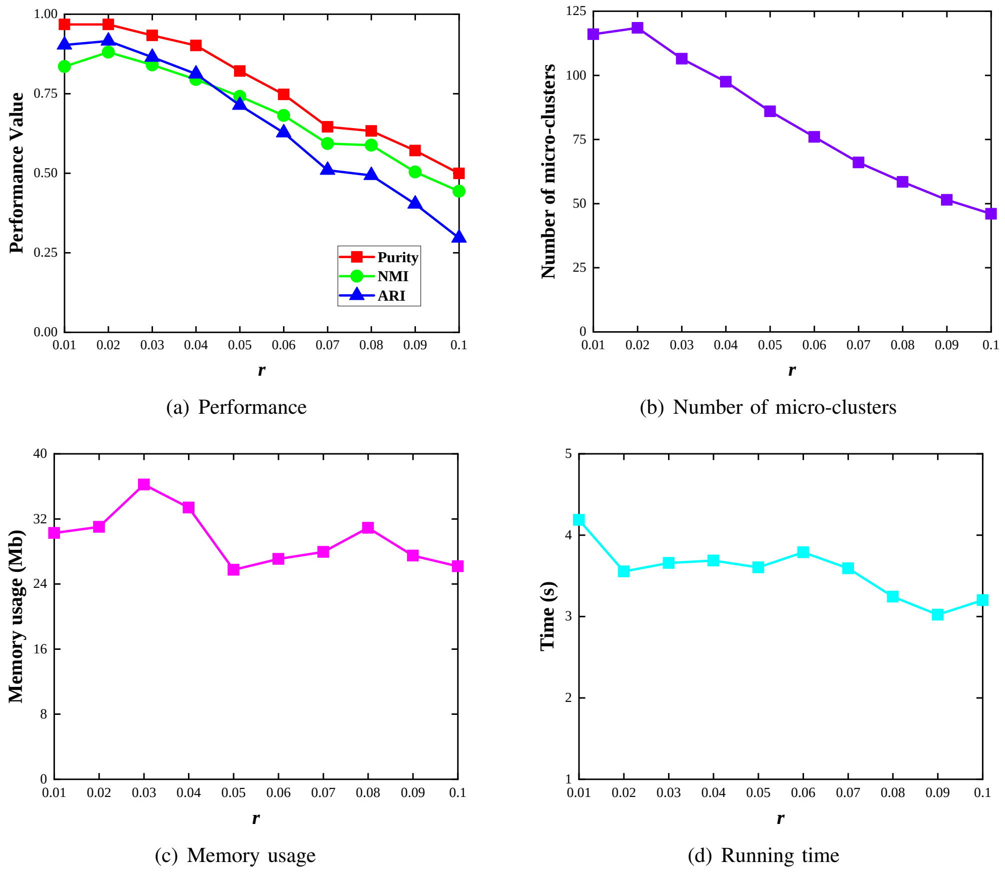


Fig. S9. Performance, number of micro-clusters, memory usage, and running time of TWStream with varying r on DS2.

On the Bench[7] dataset, we examine the impact of the decay factor λ on clustering performance. The data objects of Bench are divided into two classes, which move diagonally in 2-dimensional space, overlapping in the center. λ is selected in the range [0.002, 0.011]. Fig. S10 illustrates the clustering results of TWStream at the gap times t1 with varying λ values. In addition, we monitor the clustering performance, number of micro-clusters, memory usage, and running time of the proposed algorithm in Fig. S11. As shown in Fig. S10, the two clusters intersect at the t1 gap time when $\lambda = 0.002$. In this case, the algorithm incorrectly recognizes the intersecting part as a whole and treats it as a separate cluster. As λ increases, cluster length decreases and cluster boundaries become clearer. It is due to the fact that as λ increases, data weights decay faster, making older data less significant in clustering. According to Fig. S11(a), the clustering performance of the proposed algorithm gradually improves with increasing λ, which is in agreement with our findings in Fig. S10. Additionally, Figs. S11(b)-(d) show that as λ decreases, the number of micro-clusters decreases, memory usage drops, and the proposed algorithm runs more efficiently. Similar to EDMStream [4], we set λ to 0.0028 in this paper to maximize cluster shape identification while capturing concept drifts.

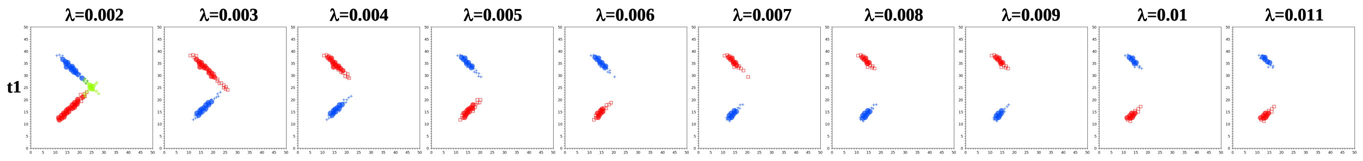


Fig. S10. Clustering results of TWStream with varying λ on Bench at the gap time t1.

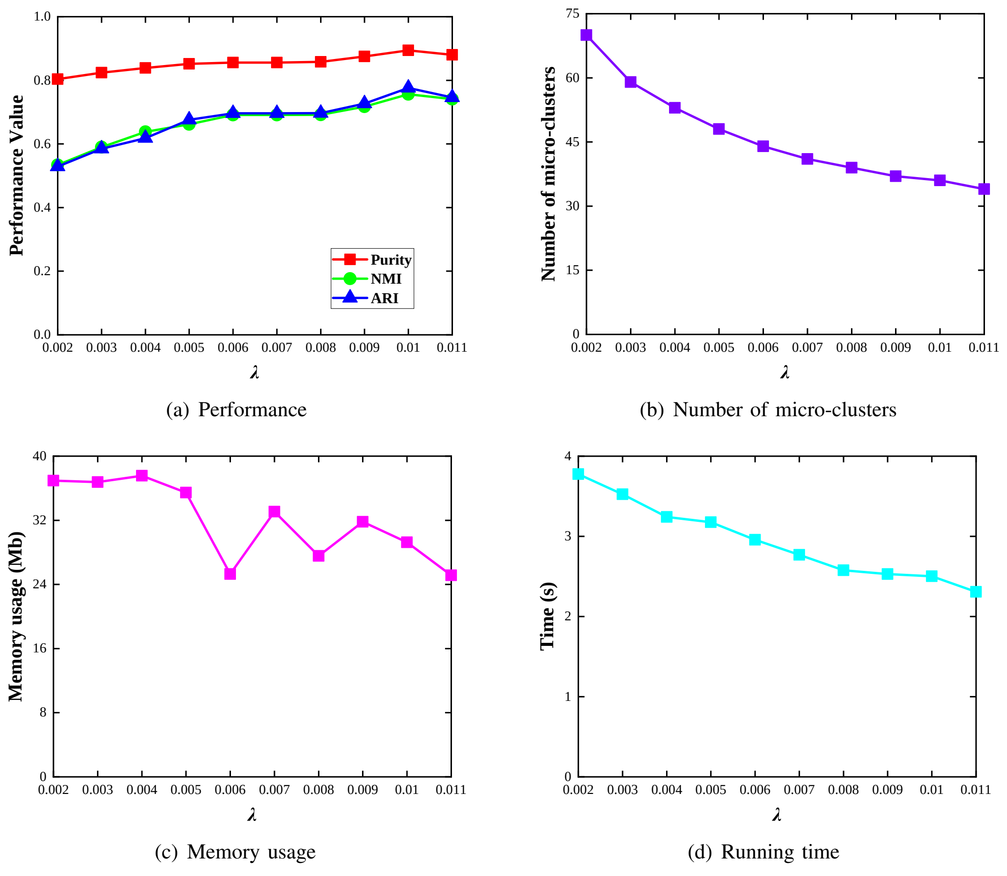
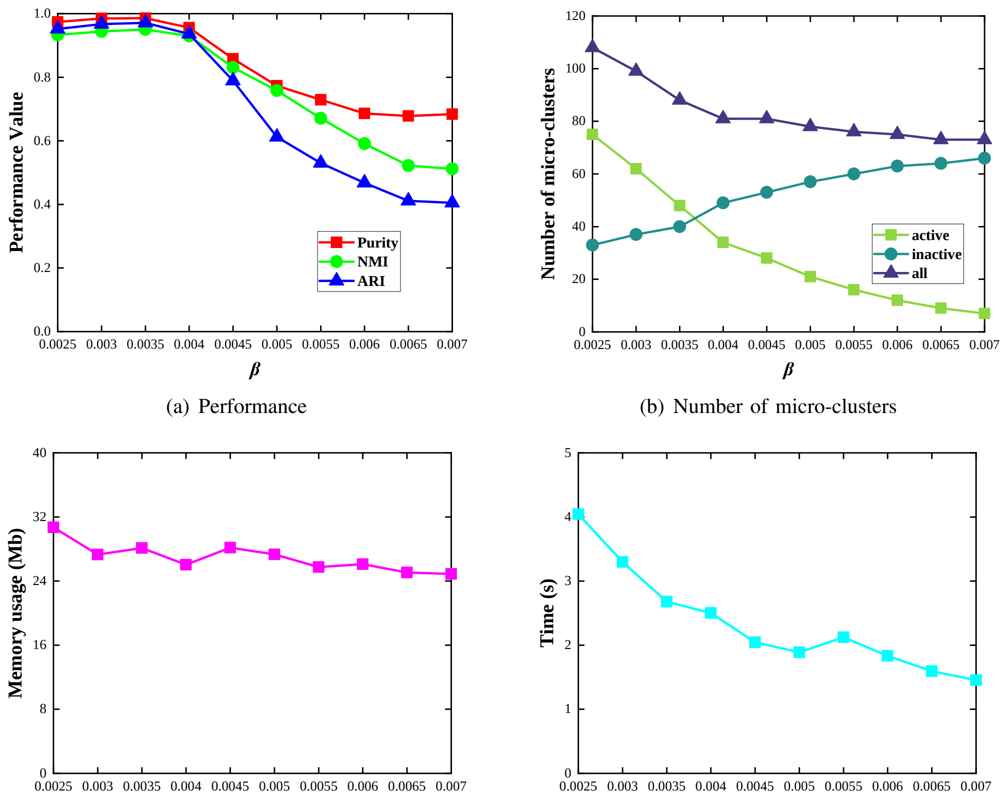


Fig. S11. Performance, number of micro-clusters, memory usage, and running time of TWStream with varying λ on Bench.

On the DS2 dataset, we discuss how the weight threshold factor β impacts the clustering performance of the proposed algorithm. As shown in Fig. S12, we track the clustering performance, the number of micro-clusters, the memory usage, and the running time of the proposed algorithm. Eq. (6) in the main text indicates that β belongs to the range $(1 - 2^{-\lambda}, 1)$, so when $\lambda = 0.0028$, β belongs to the range (0.002, 1). we choose β value in the range [0.0025, 0.007]. Fig. S12(a) shows that as β increases, the clustering performance improves and then gradually reduces. As shown in Fig. S12(b), the number of active micro-clusters gradually decreases with increasing β, while the number of inactive micro-clusters gradually increases. The total number of micro-clusters decreases slowly with increasing β. A positive correlation exists between β and the weight threshold (i.e., W_{min}), which is used to distinguish between active micro-clusters and inactive micro-clusters. Thus, a higher value of β leads to fewer active micro-clusters. The clustering performance will be reduced if there are too many or too few active micro-clusters. Similarly, Fig. S12(c) shows a gradual decrease in memory usage, and Fig. S12(d) shows a gradual reduction in running time. To filter out possible outliers and maintain the appropriate number of active micro-clusters to capture the actual data distribution, we set β to 0.0021, consistent with the EDMStream [4] setting.



(a) Performance

(b) Number of micro-clusters

(c) Memory usage

(d) Running time

Fig. S12. Performance, number of micro-clusters, memory usage, and running time of TWStream with varying β on DS2.

In conclusion, the impact of each parameter on clustering performance is discussed comprehensively. Moreover, recommendations are provided for selecting values for all parameters.

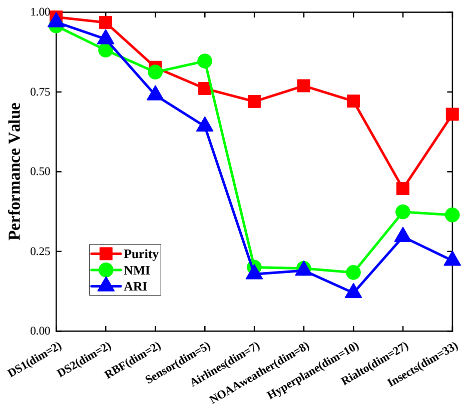
9. Effects of Stream Dimensionality on Clustering Quality

In this section, we discuss how data stream dimensionality affects the clustering quality of the proposed algorithm. The experiments are conducted on nine datasets with increasing dimensionality in Table I in the main text. In Fig. S13, we present the clustering performance and computation rate of TWStream on the above datasets. The computation rate, denoted as the amount of data processed per unit of time, is calculated by dividing the total amount of datasets by the overall running time of the algorithm.

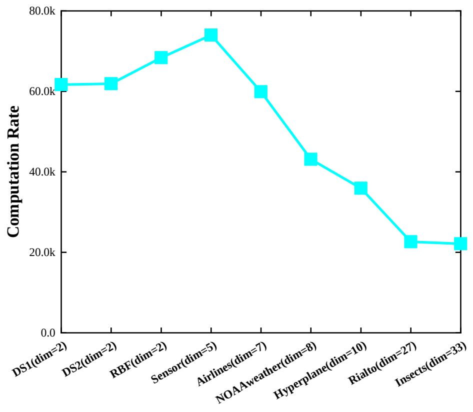
According to Fig. S13(a), clustering performance decreases as dataset dimensionality increases. In particular, NMI and ARI metrics show substantial decreases. Specifically, the quality of clustering decreases significantly when the dimensionality is increased to 7. We attribute this limitation to the manual adjustment of the micro-cluster radius r, which is common among most micro-cluster-based stream clustering algorithms. This approach struggles to cope with sample sparsity in high-dimensional data streams. Recently, some researchers propose a data summary method using the *granular ball* [5], [6], [7], [8]. This technique enables the adaptive determination of the granular ball radius. Therefore, the strategy may be a potential option for the adaptive selection of micro-cluster radius in our future work. According to Fig. S13(b), the computation rate initially increases but subsequently decreases with increasing dataset dimensionality. Specifically, the computation rate decreases continuously when the dimensionality is greater than 5. It may be due to the proposed algorithm utilizing the Euclidean distance as the distance metric, which is time-consuming in spaces with high dimensions. To mitigate this limitation, we plan to explore more efficient distance metrics in our future work.

References

[1] M. Steinbach, G. Karypis, and V. Kumar, “A comparison of document clustering techniques,” 2000.
[2] M. Meilă, “Comparing clusterings—an information based distance,” *Journal of Multivariate Analysis*, vol. 98, no. 5, pp. 873–895, 2007.



(a) Performance



(b) Computation Rate

Fig. S13. Clustering performance and computation rate of TWStream on data streams with varying dimensionality. The horizontal axes of (a) and (b) represent the nine datasets arranged in increasing dimensionality. The vertical axis of (a) represents clustering performance. The vertical axis of (b) represents computation rate. The computation rate is denoted as the amount of data processed per unit of time.

[3] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, pp. 193–218, 1985.

[4] S. Gong, Y. Zhang, and G. Yu, "Clustering stream data by exploring the evolution of density mountain," *Proceedings of the VLDB Endowment*, vol. 11, no. 4, pp. 393–405, 2017.

[5] D. Cheng, Y. Li, S. Xia, G. Wang, J. Huang, and S. Zhang, "A fast granular-ball-based density peaks clustering algorithm for large-scale data," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[6] S. Xia, X. Dai, G. Wang, X. Gao, and E. Giem, "An efficient and adaptive granular-ball generation method in classification problem," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[7] S. Xia, S. Zheng, G. Wang, X. Gao, and B. Wang, "Granular ball sampling for noisy label classification or imbalanced classification," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[8] S. Xia, D. Peng, D. Meng, C. Zhang, G. Wang, E. Giem, W. Wei, and Z. Chen, "Ball k-means: Fast adaptive clustering with no bounds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 87–99, 2020.