Available online at www.qu.edu.iq/journalcm
JOURNAL OF AL-QADISIYAH FOR COMPUTER SCIENCE AND MATHEMATICS
ISSN:2521-3504(online) ISSN:2074-0204(print)

A Fully Online Clustering Approach for Enhanced Performance of Health Information System

Ahmed Al-Shammari
Department of Computer Science, College of Computer Science and Information Technology, University of Al-Qadisiyah, Al Diwaniyah, 58002, Iraq

ahmed.alshammari@qu.edu.iq

ARTICLE INFO

Article history:
Received: 30 /04/2023
Rrevised form: 08 /06/2023
Accepted : 14 /06/2023
Available online: 30 /06/2023

Keywords:

Data Mining

Clustering

Data stream

Maintenance

Algorithms

Healthcare

ABSTRACT

Health Data clustering is a significant research direction aiming to extract knowledge from a continuous health data flow to support online health decisions. However, processing health data clusters is still a challenging task. Existing clustering approaches are subject to various limitations in terms of considering the neighbor clusters and conducting multiple operations during the maintenance process. In this paper, we model, design and implement a novel framework called IClustMaint for efficiently clustering and maintaining health data clusters incrementally. A two-phase algorithm is embedded in the framework. We first employ the Principal Component Analysis (PCA) method to efficiently reduce the high costs of the initial clustering phase. Next, in the maintenance phase, we propose the incremental Cluster maintenance (ICM) approach for managing the generated cluster during a period of time. Technically, when the data clusters are evolving over time and need to be maintained frequently, the ICM approach improves the performance of cluster maintenance by only tracking the edge points. The experimental results on a real medical dataset verify the efficiency of the proposed approaches.

https://doi.org/10.29304/jqcm.2023.15.2.1239

1. INTRODUCTION

Healthcare data are managed through a system called a health information system (HIS). Systems that collect, store, manage, and communicate a patient's electronic data are called "EMRs" for medical records [1], [2]. health data stream mining is a relatively new field of study that helps online medical decisions by improving information extraction from data streams about medical applications [3], [4]. Due to their high frequency, vast volume, and indefinite sequence, data streams can be extremely difficult to cluster [5], [6]. Large volumes of data must be sent between patients and healthcare facilities quickly using medical web apps [7], [8]. The enormous volume of requests and the amount of data, however, result in a considerable network delay. Consider the case where a group of patients send their information from different measurement devices to the medical center to get instructions from a doctor. These devices are used to measure vital signs including blood pressure, temperature, and ECG information. The doctor or expert must be aware of how each request (case) is changing over time to reply to all of the cases promptly. As a result, finding clusters of risk cases (such as elevated blood pressure or temperature) allows the medical staff to quickly identify the risk cases and respond to any interested patients. In this study, we discuss the clustering and preservation of medical data to effectively identify high-risk patients. Solve the issue, there are a lot of difficulties. The first challenge is how to perform data clustering effectively for initializing a set of clusters. To preserve the cluster attributes, dynamic cluster maintenance algorithms [9], [10] have been presented. These methods have also demonstrated certain drawbacks. For instance, monitoring the impact of nearby clusters and performing several tasks while doing cluster maintenance. As a result, an incremental maintenance strategy is required to successfully track the clusters of medical data. We proposed a new framework called "*IClustMaint*" for incrementally maintaining the existing clusters to overcome the aforementioned restrictions. Clustering and maintenance are the two key parts of this framework. To generate a set of clusters for the incoming data points, we develop the initial clustering technique. Then, we proposed the cluster maintenance technique to progressively maintain the data clusters. The second component includes two approaches for cluster maintenance: (1) Naive maintenance approach, and (2) Incremental Cluster Maintenance (ICM) approach. These approaches use two sets of maintenance operations: (1) data insertion, (2) data deletion, (3) merging and (4) splitting. The main contributions of this paper are summarized as follows:

- We proposed a fast approach for clustering health data based on the Principal Component Analysis (PCA) method. This method is employed to reduce the computations of pairwise distance measurements.
- We further proposed an efficient maintenance approach for improving the performance of cluster maintenance by tracking only the border (unsteady) points.
- We validate the proposed clustering and maintenance approaches with extensive experiments on a real medical dataset.

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 presents the problem definition, which is followed by the proposed framework. The experimental results are presented in Section 4. and the paper is finally concluded in Section 5.

2. RELATED WORK

This section highlights the basic concepts and limitations in the studies related to data stream clustering. To the best of our knowledge, there has not been much work on clustering medical data streams. Thus, we highlight the traditional approaches for clustering data streams. There are three main categories for data stream clustering: (1) partitioning-based approaches, (2) hierarchical-based approaches, and (3) density-based approaches. The partitioning approaches distribute the incoming data points into a set of k clusters [11], [12]. Aggarwal et al. [13] proposed a CluStream algorithm for evolving data stream clustering. A micro cluster is used as a temporal extension of the clustering feature vector in BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [14]. CluStream follows the online/offline approach. In the online phase, CluStream continually maintains a set of n micro-clusters in the data stream. When a new micro-cluster is created, an outlier micro-cluster is deleted or two neighbor micro-clusters are merged. In the offline phase, it performs k-means to cluster the stored n micro-clusters. Zhou et al. [15] introduced SWClustering for tracking the evolution of clusters over sliding windows. They proposed an Exponential Histogram of Cluster Features (EHCF) as a novel data structure to handle cluster evolution. This approach tracks the clusters efficiently

without overlapping with other clusters based on the EHCF data structure. The hierarchical algorithms aim to group data objects into a hierarchical tree of clusters called a dendrogram. This kind of algorithm does not require a predefined number of clusters. Kranen et al [16] introduced ClusTree for indexing micro-clusters for anytime stream mining. ClusTree uses an R*-tree structure to index micro-clusters at different levels of granularities. It applies a depth-first decent strategy for node insertion, which iteratively evaluates for better decisions as long as time permits. Cao et al. [17] proposed the DenStream approach for data stream clustering. This algorithm uses micro- clusters to capture synopsis information of data streams. The online phase updates the micro-clusters collection. For the offline components, a cluster is created as a group of micro-clusters that are dense and close to one another by applying the DBSCAN technique which completely depends on two main parameters, and MinP ts to finalize the clustering process. However, there are some limitations to the proposed clustering approaches. For instance, partitioning-based approaches need to specify the number of clusters k in advance and do not consider the effects of the neighbor clusters [18]. As a result, it is quite difficult to adopt these approaches for evolving data stream clustering. Moreover, hierarchical-based clustering approaches are time-consuming and restricted by computation capacities. Therefore, the main objective of our framework is to propose an efficient approach for incrementally maintaining the medical data clusters promptly.

3. PROBLEM DEFINITION AND PROPOSED FRAMEWORK

First, we formally define the problem of medical data clustering and dynamic cluster maintenance. Then we give an overview of our proposed framework.

3.1 Problem Definition

Assume a stream of data points D={x_1, x_2,x_3,...x_n}, where xi is the ith of arrived data points at the timestamp T={t_1,t_2,t_3,...t_n}. Each data point xi:tj represents a patient's record, the input is then a set of medical data records. The main task is to generate a set of initial clusters C= {c_1, c_2, c_3,...c_n} for the arriving data points, and then incrementally maintain the existing clusters when we receive new data points. We first transform the representation of incoming data points to data vectors V= {v_1, v_2, v_3,...v_n} to reduce the high dimensionality of data.

Definition 1: Data Vector: A data point x is represented as a vector v={m_1,m_2,m_3,...m_n}, where m_i reflects the mean value of a specific segment for the corresponding vector. Then, we calculate the pairwise distance for each vector to find the closest pairs of vectors that have high similarity.

The initial clusters are created for the incoming data points based on their similarity. Because health data streams are evolving over time, the generated clusters are to be updated frequently when cluster c receives new data points. Moreover, it is infeasible to keep the entire data vectors of cluster c due to memory restrictions. Thus, we create a data slot for the cluster.

Definition 2: Data Slot: A data slot *St* is defined as a queue data structure that maintains a cluster c during a predefined time interval. The size of the data slot St size is an adjustable parameter that represents the range of the time interval, i.e., 30 days.

Cluster maintenance requires high computational costs for tracking the cluster evolution. Also, the ground truth of cluster maintenance within a small window size depends on the re-computation concept, which is completely inefficient. Therefore, to improve the performance of cluster maintenance, there is a need for an efficient maintenance approach that is capable of maintaining the data clusters incrementally by tracking a specific part of the cluster.

3.2. Overview of the Proposed Framework

Figure 1 shows our proposed framework called IClustMaint. The proposed framework includes two main components: (1) initial clustering, and (2) dynamic cluster maintenance. In the initial clustering, we start with generating the data vectors for incoming data points by using Principal Component Analysis (PCA). Then, PCA-based Euclidean distance is used to compute the distance between the generated vectors. Then, Hierarchical Agglomerative Clustering (HAC) algorithm is used to assign similar vectors to their clusters. In cluster maintenance, we propose two maintenance approaches: (1) Naive approach, and (2) Incremental Cluster Maintenance (ICM) approach to incrementally maintain the data cluster.

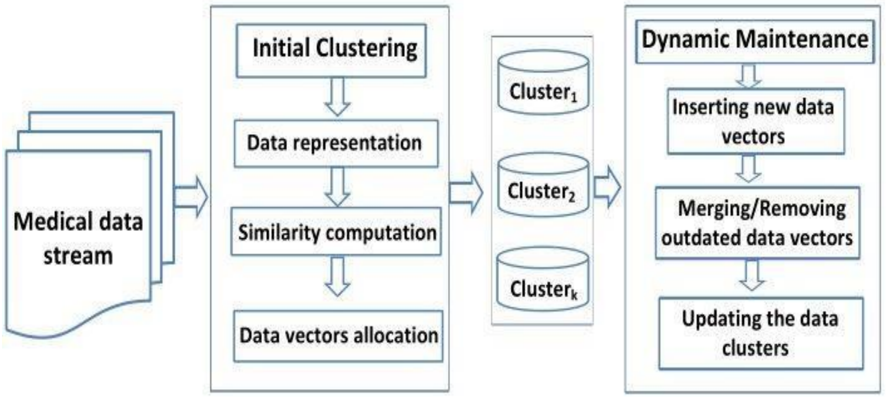


Figure 1 IClustMaint framework

3.2.1 Data Clustering

The first component of the proposed framework is initial clustering. We proposed an improved clustering technique called "HAC+PCA". This component includes three basic steps: (a) data representation, (b) similarity computation, and (c) vector clustering. In the first step, we create the data vectors (V) to efficiently represent the incoming data. Technically, Principal Component Analysis (PCA). The PCA is an unsupervised machine learning technique. PCA is a linear dimensionality reduction approach (algorithm) that converts a collection of correlated variables (p) into a smaller k number of uncorrelated variables called principal components while keeping as much variance in the original dataset as feasible. In the second step, we use the PCA-based Euclidean distance to efficiently capture similar data vectors. It follows the basic properties of the distance metrics. Based on this method, the required number for checking the similarity between a pair of vectors is reduced significantly. Thus, fast computation is performed to finalize the similarity measurement. The final output of this step is the similarity score that refers to each vector against all other vectors. After computing the similarity between the vectors, the last step of the clustering is vector grouping.

To finalize the initial clustering, we use Hierarchical Agglomerative Clustering (HAC) for allocating similar data vectors into their proper clusters. HAC has a powerful visualization in web data clustering. Additionally, it does not require determining the number of clusters k as an initial parameter compared to other approaches like K-means. This approach makes a nested hierarchy of similar groups for a set of data vectors based on the bottom-up process. Initially, each vector v represents a single cluster, then regularly merges the vectors with high similarity (minimum distance).

Algorithm 1 presents the main steps of the initial clustering component. In lines 1-2, we do the data representation by generating the data vectors based on the PCA. In line 3, we calculate the distance between each pair of data vectors. We initialize the data clusters in lines 4-6 based on the minimum distance between the data vectors.

```
Algorithm 1: Data Clustering
Require: Stream of data points $\mathcal{D}=\{x_1,x_2,x_3,\ldots x_n\}$,
         distance threshold $\delta$
Ensure : Set of clusters $\mathcal{C} = \{c_1, c_2,c_3,c_{|\mathcal{C}|}\}$
1 for each $x \in \mathcal{D}$ do
2     $\vec{v}$ ¸ generate a data vector based on PCA
3     for each pair of vectors $\in \mathcal{V}$ do
4         $dist$ ¸ calculate the pairwise distance
5         if $dist(\vec{v}_i, \vec{v}_j) \leq \delta$ then
6             $c_i$ ¸ initialise a cluster
7             $getNext(\vec{v})$
8         $\mathcal{C}$ ¸ $\mathcal{C} \cup c_i$
9     return C
```

3.2.2. Cluster Maintenance

In this section, we focus on the main steps of the cluster maintenance component. Due to the massive number of changes in the underlying data, the clusters need to be maintained effectively. Therefore, we propose two incremental approaches for cluster maintenance: (1) Naive approach, and (2) Incremental Cluster Maintenance approach. The technical details for the proposed maintenance approaches are presented in Sections A and B respectively. The core idea of cluster maintenance is to maintain the existing clusters frequently when new data points arrive. There are two sets of maintenance operations: (1) insertion, (2) deletion, (3) merging and (4) splitting. Once the new data vector is assigned to its nearest cluster. To observe the changes in the existing clusters, each cluster is partitioned into two main spaces: (a) border space, and (b) center space.

a. Border Space (Br): It includes the data vectors located at the boundary of a cluster c.

b. Centre Space (Cr): It includes the data vectors located at the center of a cluster c.

A. Naive Maintenance Approach

We first present the naive maintenance approach to maintain the generated clusters incrementally. Technically, this approach starts with generating new data vectors and then a decision is made to either allocate the new vector to the nearest cluster c by calculating the pairwise distance dist between the newly incoming vector and the existing clusters or initialise new clusters. This approach uses three sets of maintenance operations as follows:

- Enqueue(v; c): this operation inserts the new data vectors into the cluster.
- Dequeue(v; c): this operation removes the expired data vectors from the cluster.
- moveOut(v; Br; c) this operation splits the data vectors from the border space to the outside of the cluster.
- moveIn(v; Br; c) this operation merges the data vectors from the outside to the border space of the core cluster.
- moveOut(v; Cr; c) this operation splits the data vectors from the core space.
- moveIn(v; Cr; c) this operation merges the data vectors into the core space.

The naive approach tracks the entire data vectors in both cluster spaces which incurs high computational costs. Thus, we propose a second maintenance approach to potentially reduce the heavy computations for cluster maintenance.

A. Incremental Cluster Maintenance Approach

cluster maintenance is computationally expensive due to the massive number of observations on the cluster evolution. Therefore, we propose the Incremental Clusters Maintenance approach (ICM) approach to improve the performance of dynamic cluster maintenance. The proposed approach only tracks the data vectors in the border space. This approach does not require checking the data vectors in the core space of the cluster. This approach uses two sets of maintenance operations as follows:

-Enqueue (v; c): this operation inserts the new data vectors to the nearest cluster.

- Dequeue (v; c): this operation removes the expired data vectors from the cluster.

- moveOut (v; Br; c) this operation splits the data vectors from the border space to the outside of the cluster.

- moveIn (v; Br; c) this operation merges the data vectors from the outside of the cluster to the border space of the stable cluster.

In addition, there are several issues might occur during the process of cluster maintenance. For instance, some vectors inside the border space move out of their cluster and some of the neighbor clusters tend to keep these vectors for stability purposes. The ICM approach considers these issues by applying three maintenance rules that are stated as follows:

- Stability: A cluster c is a stable cluster if it exceeds or equals the required number of data vectors to be included in the cluster.

- Sparsity: A sparse vector refers to a vector that is located beyond the border space of a cluster c.

- Observation: A sparse vector merges only to the stable cluster which has a minimum distance center space.

Algorithm 2: Handling Data Cluster

```
   Require: Set of initial clusters $\mathcal{C} = \{c_1, c_2, c_3, \ldots, c_{|C|}\}$, new data points $\mathcal{D}_{new}$,
            cutoff distance threshold $d_c$, number of required vectors $n_c$, $\Delta$ t
   Ensure : Updated set of clusters $\mathcal{C}_{new}$
1  for each $x \in \mathcal{D}_{new}$ do
2      $\mathcal{V}$ ¸ $createVector(\vec{v}_i)$
3      $c$ ←findNearestCluster($\vec{v}_i$,$\mathcal{C}$)
4      $\vec{c}_{t_{new}}$ ←update($\vec{c}_t$)
5  for each $\vec{v}_i \in c_{new}$ do
6      if $dist(\vec{v}_i, \vec{c}_{t_{new}}) > d_c$ then
7          $moveOut(\vec{v}_i, B_r, c_{new})$
8  for each vector $\vec{v}'$ do
9      if $dist(\vec{v}', \vec{c}_{t_{new}}) \leq d_c$, $c_{new} \geq n_c$ then
10         $moveIn(\vec{v}', B_r, c_{new})$
11 $\mathcal{C}_{new}$ ¸ $c_{new} \cup \mathcal{C}$
12 return $\mathcal{C}_{new}$
```

Algorithm 2 presents the cluster maintenance steps for the proposed framework using the Handling Data Clusters maintenance approach. In lines 1-2, we do the maintenance preparation. Lines 3-5 check the distance between the data vectors and the cluster center. In lines 7-9, we assign the vectors to the nearest cluster. In lines 10-15, we track the vectors in the border space by using two sets of maintenance operations.

4. EXPERIMENTS

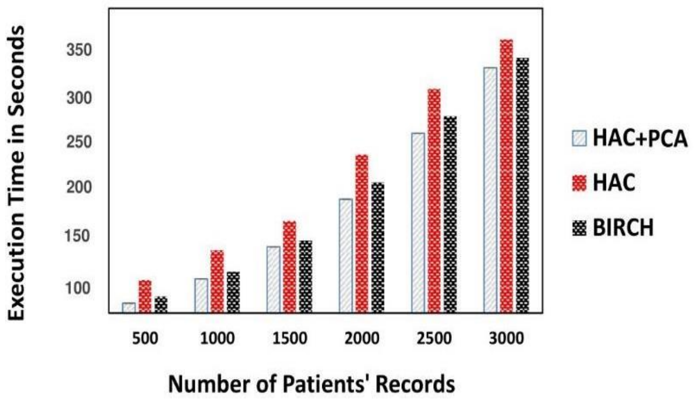
This section highlights the experimental results of the proposed approaches. A brief description of the experimented datasets is presented in subsection A. Then, the comparison between the hierarchical clustering approaches is discussed in subsection B. Finally, the efficiency of the proposed cluster maintenance approaches is presented in subsection C. The experiments are implemented in Python programming language and executed by a processor Intel(R) Core (TM) i7-7700HQ CPU.

4.1. Dataset Description

To verify the performance of the proposed approach, we conducted experiments on a real dataset. Specifically, we use the Cuff-Less Blood Pressure Estimation dataset which includes three main signals for each patient: (1) blood flow, (2) blood pressure, and (3) electrocardiogram. The source of the dataset is available online. on UCI-Centre for Machine Learning and Intelligent Systems. The dataset format is in Matlab's v7.3, and the total size is 3.17 GB. The selected dataset contains 3000 patients' records. We convert the mat files to CSV format and run the experiments on 12000 records.

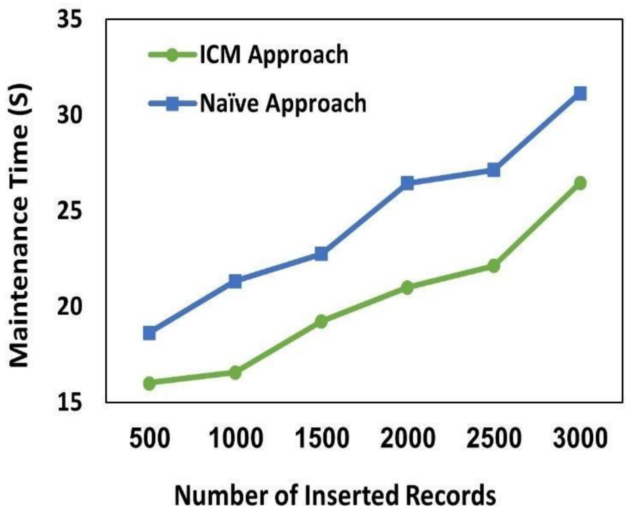
4.2. Comparison Between Clustering Approaches

We validate the efficiency of our initial clustering approach (HAC+PCA) according to the execution time in comparison with two well-known clustering approaches: (1) Hierarchical Agglomerative Clustering (HAC) approach, and (2) Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) approach. In the first approach, we improve the implementation of hierarchical agglomerative clustering. Technically, Principal Component Analysis (PCA) is employed to reduce the extensive pairwise distance calculations of the HAC approach. In the second approach, we use the implementation of incremental hierarchical clustering [16]. Since the BIRCH approach has outperformed (HAC) approach, both approaches require a long clustering time to calculate the similarity among data clusters and finalize the clustering process. The results show that our approach requires less execution time for clustering 3000 patients' records in comparison with competitive approaches as shown in Fig.2.



4.3. Efficiency of Cluster Maintenance

Fig.3 shows the execution time of the proposed data cluster maintenance approaches by varying the number of inserted records. The number of records is set from 500 to 3000 with an increase of 500 records at each iteration. We observe that the ICM maintenance approach requires less execution time for cluster maintenance in comparison with the first maintenance approach. The ICM approach conducts only 3 maintenance operations by tracking only the unstable data of the clusters.



5. CONCLUSIONS

In this paper, we introduced a new framework called IClustMaint for efficiently clustering health data streams and incrementally maintaining data clusters. We first design a data clustering component (HAC+PCA) to generate a set of clusters for the incoming health data. Technically, the Hierarchical Clustering and Principal Component Analysis (PCA) method is used to efficiently reduce the pairwise distance calculations which effect to accelerate the clustering process. Thereafter, the existing clusters are evolving over time and need to be maintained frequently. Therefore, we proposed two approaches: (1) Naive maintenance approach, and (2) Incremental Clusters maintenance approach (ICM) for maintaining the data clusters dynamically. Our approaches have outperformed the traditional incremental clustering and maintenance approaches.

REFERENCES

[1] Al-Shammari, C. Liu, M. Naseriparsa, B. Q. Vo, and T. Anwar. A framework for clustering and dynamic maintenance of xml documents. In Advanced Data Mining and Applications: 13th International Conference, ADMA 2017, Singapore, November 5–6, 2017, Proceedings, volume 10604, page 399. Springer, 2017.

[2] Lal, H. C. Ashworth, S. Dada, L. Hoemeke, and E. Tambo. Optimizing pandemic preparedness and response through health information systems: lessons learned from ebola to covid-19. Disaster medicine and public health preparedness, 16(1):333–340, 2022.

[3] L. Sun, J. Ma, Y. Zhang, and H. Wang. Exploring data mining techniques in medical data streams. In Australasian Database Conference, pages 321–332.Springer, 2016.

[4] S. J. Miah, E. Camilleri, and H. Q. Vu. Big data in healthcare research: a survey study. Journal of Computer Information Systems, 62(3):480–492, 2022.

[5] T. Guo, K. Yu, M. Aloqaily, and S. Wan. Constructing a prior dependent graph for data clustering and dimension reduction inthe edge of aiot. Future Generation Computer Systems, 128:381–394, 2022.

[6] T. Guo, K. Yu, M. Aloqaily, and S. Wan. Constructing a prior dependent graph for data clustering and dimension reduction inthe edge of aiot. Future Generation Computer Systems, 128:381–394, 2022.

[7] R. P. Wijayanti, P. W. Handayani, and F. Azzahro. Intention to seek health information on social media in indonesia. Procedia Computer Science,197:118–125, 2022.

[8] P. Zhou, X. Wang, L. Du, and X. Li. Clustering ensemble via structured hypergraph learning. Information Fusion, 78:171–179, 2022.

[9] M. Zhao, A. Jha, Q. Liu, B. A. Millis, A. Mahadevan-Jansen, L. Lu, B. A. Landman, M. J. Tyska, and Y. Huo. Faster mean shift: Gpu-accelerated clustering forcosine embedding-based cell segmentation and tracking. Medical Image Analysis, 71:102048, 2021.

[10] M. Hahsler and M. Bola ˜ nos. Clustering data streams based on shared density between micro-clusters. IEEE Transactions on Knowledge and DataEngineering, 28(6):1449–1461, 2016.

[11] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. de Carvalho, and J. Gama. Data stream clustering: A survey. ACM Computing Surveys (CSUR),46(1):13, 2013.

[12] L. Bai, X. Cheng, J. Liang, and H. Shen. An optimization model for clustering categorical data streams with drifting concepts. IEEE Transactions on Knowledge and Data Engineering, 28(11):2871– 2883, 2016.

[13] C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In Proceedings of the 29th international conference onVery large data bases-Volume 29, pages 81–92. VLDB Endowment, 2003.

[14] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: A new data clustering algorithm and its applications. Data Mining and Knowledge Discovery,1(2):141–182, 1997.

[15] Zhou, F. Cao, W. Qian, and C. Jin. Tracking clusters in evolving data streams over sliding windows. Knowledge and Information Systems, 15(2):181–214, 2008.

[16] P. Kranen, I. Assent, C. Baldauf, and T. Seidl. The clustree: indexing micro-clusters for anytime stream mining. Knowledge and information systems,29(2):249–272, 2011.

[17] F. Cao, M. Estert, W. Qian, and A. Zhou. Density-based clustering over an evolving data stream with noise. In Proceedings of the 2006SIAM international conference on data mining, pages 328–339. SIAM, 2006.

[18] P. Zhou, X. Wang, L. Du, and X. Li. Clustering ensemble via structured hypergraph learning. Information Fusion, 78:171–179, 2022.