ITC 1/48
Journal of Information Technology and Control
Vol. 48 / No. 1 / 2019
pp. 18-33
DOI 10.5755/j01.itc.48.1.18753

Clustering Over Multiple Evolving Data Streams of the Traffic Cyber-Physical Systems

Received 2017/08/03 | Accepted after revision 2019/01/08

http://dx.doi.org/10.5755/j01.itc.48.1.18753

# Clustering Over Multiple Evolving Data Streams of the Traffic Cyber-Physical Systems

**Mingyue Cui, Hongzhao Liu, Wei Liu, Shuyi Li, Chongnian Qu**

College of Mechanical and Electronical Engineering, Nanyang Normal University, Nanyang, Henan 473061, China;
Oil Equipment Intelligent Control Engineering Laboratory of Henan Province, Nanyang, Henan 473061, China
email: cuiminyue@sina.com

**Corresponding author:** cuiminyue@sina.com

Extracting and retaining valuable information from multiple data streams of the traffic Cyber-Physical Systems (CPS) has been attracted an increasing amount of attentions by the related researchers. In this paper, the Incremental Clustering framework is proposed for multiple sensor data streams by low rank approximation Matrix Factorization (IC-MF), which can monitor the distribution of clusters over multiple sensor data streams based on their correlation. In the IC-MF, both the low-rank matrix approximation and matrix factorization-based clustering are applied, and IC-MF incorporates the historical results and the relationship between the nodes of the current step and previous step. To improve the accuracy of the increments, a low-rank approximation of the adjacency matrix is obtained at each time step, and makes IC-MF work directly in the low-rank subspace. The main idea of IC-MF is to make use of the similarity between two consecutive time steps to quickly update the approximating subspace. The performance and efficiency of the algorithm are demonstrated by traffic CPS experiments on the real and synthetic data sets. The experimental results show the effectiveness of the proposed algorithm for clustering multiple evolving data streams.

**KEYWORDS:** Cyber-Physical Systems (CPS), Sensor Data Streams, Incremental Clustering, Matrix Decomposition.

## 1. Introduction

The Cyber-Physical Systems (CPS) has been applied in a wide range of domains, such as traffic systems, energy and industrial automation, health and biomedical agriculture, etc. So the CPS has become an active topic in data mining and machine learning[18]. The CPS contains a large amount of sensors that generate abundant data records in real time. In the CPS, large amounts of sensor data need to be efficiently stored and processed to extract useful information [14].

Traffic Cyber-Physical Systems are anticipated to achieve full coordination and optimization of traffic systems via the increased multiple sensor data streams between the transportation cyber systems and traffic physical systems [18]. In this paper, we will focuse on finding out interesting and useful knowledge by analyzing the correlation among multiple sensor data streams. In some cases, the correlation between sensor data streams reports unusual or abnormal events; such a relationship change may imply fundamental changes of the monitored objects and possess high domain significance. Mining multiple sensor data streams is an area of research that attempts to extract useful information by analyzing the correlation among these multiple data streams [26].

The clustering is the most widely used to take data mining automatically. The similar data streams are put together into a group or the dissimilar ones are separated into different groups. By clustering multiple sensor data streams dynamically, we observe the changes of cluster numbers and the members of each cluster, and then the useful information can be extracted to make decision or manage data in various applications. One of the challenges of sensor data streams is the dynamic nature. At different time-steps, the number of data streams (insertion and removal of sensor data streams) or the number of clusters structure (insertion and removal of clusters) might change. It is expected that the algorithm can discover the change over a series of time-steps to clustering multiple sensor data streams.

To discover the changes in the communities at each time-step, there exist some incremental and evolutionary clustering algorithms that are designed to handle dynamic data [3, 17, 24, 25]. However, sensor data streams require the incremental algorithms to handle not only reflecting the corresponding change among data streams. In the practical applications, the monitored processes generating time-stamped data may change drastically over time. This also requires the algorithm to be robust enough as a dramatical change happens from one time-step to the next. The clustering algorithm for multiple sensor data streams also can be used to identify data that deviate from historical patterns. Clustering at a particular time-step should be based on the associations between data streams at the time-step.

Graphs are used in a wide range of complex systems, such as traffic system, sensor network, social network, and so on, since they capture the general notion of an association between two entities [20, 23]. Matrix can be regarded as a common representation of a graph, such as a similarity matrix for a graph. The nodes of the graph are sensor data streams, and an edge is formed among each pair of nodes. The weight of each edge reflects the similarity at one time step between each pair of sensor data streams.

In this paper, the Incremental Clustering algorithm for multiple sensor data streams of traffic CPS is proposed based on low rank approximation Matrix Factorization (IC-MF), which can monitor the distribution of clusters over multiple sensor data streams based on their correlation. Comparing with clustering directly the multiple data streams periodically, an efficient incremental clustering update is applied in this paper. This work is focusing more on multiple sensor data streams based on the correlation evolving over time. To improve the accuracy of the increments, a low-rank approximation of the adjacency matrix is used at each time-step. This algorithm acts directly on the low-rank subspace. The main idea of IC-MF is to apply the similarity between two consecutive time-steps to quickly update the approximating subspace.

The rest of the paper is organized as follows. Related works are given in Section 2, which includes clustering data streams, matrix decom- position and its applications. Preliminaries and similarity in IC-MF are given in Section 3. The detailed procedures for mining multiple sensor data streams are presented in Section 4 and 5. Section 6 presents the experimental results and finally we conclude the paper with a summary and discussion of the future work in Section 7.

## 2. Related Works

### 2.1. Clustering Data Streams

Data streams mining has become an active area in data mining community [12]. The goal is to process the incoming data efficiently without recalculation from scratch and without buffering from much historical data. Because of its importance and expressiveness, various problems are studied under data streams min-

ing, such as clustering, classification, and so on.

Clustering multiple data streams or grouping data streams is supposed to process at each time stamp. Various research works have been reported [2, 5, 6, 25]. Joint clustering framework of multiple networks includes an online component that periodically computes and stores detailed summary statistics and an off-line component which responds to a wide variety of inputs [5]. Beringer and Hullermeier discuss clustering over parallel data streams [2]. The Discrete Fourier transformation (DFT) is used to summarize the data streams, and an online version of the classical K-means clustering algorithm is proposed. An incremental update mechanism was used to avoid the recalculation of DFT coefficients, and the data processing method that minimizes time was proposed in reference [18]. A COD (Clustering on Demand) framework was provided to dynamically clustering multiple data streams [9]. In [25], Yeh et al. proposed the COMET-CORE framework for online clusters over multiple evolving streams by correlations and events. DGClust (Distribute Grid Clusting) algorithm is applied in clustering distributed sensor data streams, which reduces both the dimensionality and the communication burdens [19]. The core of DGClust focuses on online discretization of data, frequent state monitoring, and online partition clustering. For exploring data correlation in sensor networks, $\alpha$ – local spatial clustering algorithm constructs a dominating set as the sensor network [16]. The data aggregation is based on the performance of the dominators in terms of their information description/ summarization.

However, these methods are applied in clustering data streams over a period of time, and the correlation between data streams can change in that period. An incremental approach to clustering for data streams over time is a relatively new topic, such as incremental spectral clustering, evolutionary clustering. Incremental algorithms on web data gain more and more attention [8] with the success of Google. Evolutionary clustering has been a relatively new topic and was first formulated by Chakrabarti et al. [3]. Evolutionary clustering simultaneously optimizes two potentially conflicting criteria, i.e., the clustering should fit the current data as much as possible, meanwhile it can not deviate dramatically from the historic context. It is important for the clustering algorithm to adapt to the recent changes in the evolving data. So Wang et al. proposed ECKF (Evolutionary Clustering based on low rank Kernel matrix Factorization) framework for evolutionary clustering large-scale data based on low-rank kernel matrix factorization [24]. Chi et al. [4] stepped further to evolutionary spectral clustering by incorporating temporal smoothness. In [17], an incremental approach is presented by extended standard spectral clustering to handle evolving data. The exigent-system and the cluster labels are incrementally updated as data points are inserted/ deleted or similarity changes occur.

## 2.2. Matrix Factorization and Applications

Graphs are applied in a wide range, such as transportation networks, sensor networks, social networks, and so on. The various problems are researched under graph mining. To find clusters in graphs is a new challenge if the graph is evolving over time. The matrix is a usual representation of a graph. The relationships between matrix factorization and k-means clustering have been explored [9].

The SVD (Singular Value Decomposition) has been served as a building block for many important applications, such as PCA (Principal Component Analysis) and LSI (Latent Semantic Indexing) [13]. SVD factorizes a matrix with the general form of $A \approx SVD^{\mathrm{T}}$, where S is a unitary basis consisting of left-singular vectors of $A$, $D$ is a unitary basis consisting of right-singular vectors of $A$, and $V$ is a diagonal matrix with singular values on the diagonal. In SVD, since matrices $S$ and $D$ are allowed to have negative eigenvalues, the projected data might have negative values in spite of the original data being positive. This can prevent the clustering results from be intuitively applied, such as documents or images that have a positive data input.

Nonnegative Matrix Factorization (NMF) [6,15] is a linear and non-negative approximate data representation technique. NMF focuses on the analysis of data matrices whose elements are nonnegative, a common occurrence in data sets derived from text and images. The non-negative data matrix $A$ is factorized into matrices $F$ and $G$ as $A \approx FG^{\mathrm{T}}$, with the constraints that $F \in \mathbb{R}^{d \times k}$ and $G \in \mathbb{R}^{n \times k}$ are non-negative. It is distinguished from the other methods by using non-negativity constraints. Semi-NMF and Convex-NMF algorithms are to expand the range of application of NMF [10]. Semi-NMF usually offers a low-dimensional representation of data points which lends itself

to a convenient clustering interpretation. The matrix $G$ is restricted to be nonnegative while placing no restriction on the signs of $F$ and allowing data matrix $X$ to have mixed signs. Convex-NMF restricts the columns of $F$ to be convex combinations of data points in $X$. The NMF clustering method has been proposed [10], where F is considered to be a centroid matrix as every column represents a cluster center, and $G$ is the cluster indicator matrix.

In particular, the result of the K-means clustering run can be written as a matrix factorization $A = FG^{T}$, where $A$ is a data matrix, $F$ contains the cluster centroids, and $G$ contains the cluster membership indicators. Nonnegative matrix factorization focuses on the analysis of data matrices whose elements are nonnegative [15].

To extend the applicable range of the NMF method, when the data matrix is unconstrained, Semi-NMF is motivated from the perspective of clustering [10]. The NMF restricts $G$ to be a nonnegative while placing no restriction on the signs of $F$. For reasons of interpretability, the Convex-NMF constrains the basis vectors $F = (f_1, \cdots, f_k)$. The vector defining $F$ lies within the column space of $A$: $f_l = w_{1l}a_1 + \cdots + w_{nl}a_n = Aw_l$, or $F = AW$. It can be applied to both nonnegative and mixed-sign data matrices. This constraint could interpret the columns $f_l$ as weighted sums of certain data points and these columns would capture a notion of the centroid.

However, the traditional NMF, Semi-NMF, and Convex-NMF are linear models and they may fail to discover the nonlinearities of data streams. In real world, the data streams have potential nonlinear structure. The kernel method is a powerful technique in dealing with nonlinear correlations. To achieve linearity of the nonlinearity, the kernel method is to map the data nonlinearly into a kernel feature space. The Convex-NMF method can be accomplished in the kernel feature space to process the nonlinear data.

Although these methods are successfully applied, the graphs require the huge amounts of space. Indeed, several important applications can be modeled as large sparse graphs, such as traffic network analysis, social network analysis. Low rank approximation for the matrix of a graph is essential in finding patterns and detecting anomalies. It can extract correlations and remove noise from matrix structured data. This has led to the development of these methods, such as Column and Row (CUR) decomposition [12], Compact Matrix Decomposition (CMD) [22], and the family of Colibri [23].

The matrix CUR decomposition [12] is a powerful technique for low rank matrix decomposition. It can be regarded as an approximation of a matrix $A \approx CUR$, where $A \in \mathbb{R}^{m \times n}$, $C$ is an $m \times c$ matrix consisting of $C$ randomly picked columns of $A$, $R$ is an $r \times n$ matrix consisting of $r$ randomly picked rows of $A$, and $U$ is a $c \times r$ matrix computed from $C$ and $R$. The CUR decomposition is operated by first selecting the representative column and row exemplars as the left and right matrices according to their probability distributions and then computes the middle matrix based on these two matrices. CUR decomposition both reveals information about the structure of the matrix and allows computations to be performed more efficiently.

The CMD [22] can be employed to obtain sparse low rank approximations. Each unique sample is scaled up based on square root of the number of times in the initial subspace. The CMD is often used to detect anomalies and monitor time-evolving graphs. In comparison to CUR, CMD not only achieves equal accuracy as CUR, but also takes less space and computational. Moreover, CMD not only can analyze static graphs, but can handle with dynamic graphs. The family of Colibri [23] is low rank method for static and dynamic graphs, respectively. Colibri-S saves space and time by eliminating linearly dependent columns while iterating over sampled columns to construct the subspace used for low rank approximation. Colibri-D builds on Colibri-S, and performs incremental updates efficiently, by exploiting the "smoothness" between two consecutive time steps.

## 3. Preliminaries and Formulations

### 3.1. Problem Statement

For a highway traffic monitoring system, the records of the traffic flow are generated in real time. We assume that data items arrive synchronously. That means that all data streams can be updated simultaneously. The data streams will be updated each time new blocks of values arrive. It is reasonable to assume that the sensors are the points in the graph and an edge is formed between each pair of sensors. These data records can be used to construct graphs of traf-

fic networks periodically. The details are depicted as follows.

Let $G^{(t)} = (S^{(t)}, E^{(t)}, W^{(t)})$ be a graph associated with the sensor network of traffic system at time $t$, where $s_i^{(t)} \in S^{(t)}$ represents the set of sensors associated with graph $G^{(t)}$, $e_{ij}^{(t)} \in E^{(t)}$ represents the set of edges between $s_i^{(t)}$ and $s_j^{(t)}$, and $\rho_{ij}^{(t)} \in W^{(t)}$ denotes the edge weight of $e_{ij}^{(t)}$. We assume that $G^{(t)}$ are $n$ sensors generating data streams simultaneously. Without loss of generality, we use the adjacency matrix $X^{(t)} \in \mathbb{R}^{n\times n}$ to represent a graph with weights $G^{(t)} = (S^{(t)}, E^{(t)}, W^{(t)})$. The weight $\rho_{ij}^{(t)}$ on each edge is a function of the similarity between sensors $s_j^{(t)}$ and $s_j^{(t)}$ at $t$. $X^{(t)}(i, j)$ is the element at the $i$th row and $j$th column of the matrix $X^{(t)}$, $X^{(t)}(:, j)$ is the $j$th column of $X^{(t)}$. Every row or column in $X^{(t)}$ corresponds to a node in $S^{(t)}$. If there is an edge from node $s_i^{(t)} \in S^{(t)}$ to node $s_j^{(t)} \in S^{(t)}$ with similarity $\rho_{ij}^{(t)}$, we set the value of $X^{(t)}(i, j)$ to $\rho_{ij}^{(t)} \in W^{(t)}$. Otherwise, we set it to zero.

At time $t$, the clustering algorithm partitions $G^{(t)}$ into $k$ clusters. The mode of cluster is defined by $\mathbb{C}^{(t)} = \{\Delta_1^{(t)}, \Delta_2^{(t)}, \cdots, \Delta_k^{(t)}\}, \forall i \in \{1, \cdots, k\}$. $\Delta_i$ should meet the following conditions:

(a) $\bigcup_{i=1}^{k} \Delta_i^{(t)} = \{s_1^{(t)}, s_2^{(t)}, \cdots, s_n^{(t)}\}$;

(b) $\bigcap_{i=1}^{k} \Delta_i^{(t)} = \varnothing$;

(c) The similarity of $\forall s_i^{(t)}, s_j^{(t)} \in S^{(t)}$ is determined by $\rho_{ij}^{(t)}$.

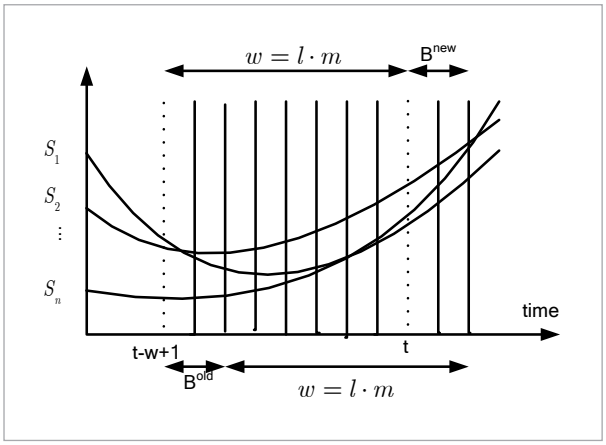
The cluster $\Delta_i$ is a set of similar data streams such that $\Delta_i^{(t)} = \{s_1^{(t)}, \cdots, s_{|\Delta_i^{(t)}|}^{(t)}\}$ ($|\Delta_i^{(t)}|$ is the total number of streams in $\Delta_i^{(t)}$). The set of all clusters found is represented by $\mathbb{C}^{(t)}$.

### 3.2. Correlation Coefficient

In traffic CPS, the data source is assumed to generate a large volume of real time event records for massive sensor data streams. Comparing with static time series, it is desirable to track a time-evolving correlation that captures their changing similarity.

Since the sensor data streams can grow infinitely and arrive continuous, it is often difficult to buffer and process all data. We consider readings within a predefined time window of length $w$. The most common type of time window is sliding window that has the fixed length and comprises the $w$ most recent observations. To support efficient processing and evolving for sensor data streams, we divide the data streams of length $w$ into $m$ blocks. The length of each block equals $l$, which means that $w = l \cdot m$. This time segment is called a "block". The data streams will be updated in a "block-wise" manner. Whenever a new block of length $l$ accumulates, the algorithm builds a new sliding window by inserting the new block into the beginning of the sliding window, and removing the old block from the end. Then, the algorithm clusters the data in the new window. The process of multiple sensor data streams over a sliding window is depicted in Figure 1.

**Figure 1**
Sliding window $w$ divided into $l$ blocks of size $m$




In Figure 1, $B_1$, $B_2$,⋯, $B_m$ denote a sequence of blocks, $B_1 = \{S_i^{(0)}, S_i^{(1)}, \cdots, S_i^{(l-1)}\}$, $B_2 = \{S_i^{(l)}, S_i^{(l+1)}, \cdots, S_i^{(2l-1)}\}$, $B_m = S_i^{(m-1)l}, S_i^{(m-1)l+1}, \cdots, S_i^{(w-1)}$. For monitoring clusters among multiple sensor data streams evolving over time, it is not desirable to compute the similarities $\rho_{ij}$ from scratch at each time point $t$. The observations of data streams usually change rapidly over time. It implies that the rise or fall of one data stream impacts the behavior of the other data streams. To discover potential anomalies and capture various trend or pattern types based on the time-evolving nature of these relationships, we discuss lagged correlation between two sliding windows for two sensor data streams. If $X$ and $Y$ are two random variables, the Pearson correlation is defined as follows

$$\rho(\mu, \nu) = \frac{E(\mu\nu) - E(\mu)E(\nu)}{\sigma_\mu \sigma_\nu}, \quad (1)$$

where $\sigma_X$ and $\sigma_Y$ are the standard deviations of $\mu$ and $\upsilon$, respectively, and $E(\mu\upsilon)$, $E(\mu)$ and $E(\upsilon)$ are the expectations of random variables $\mu$, $\upsilon$, and $\mu\upsilon$.

Given the sliding window $w$, the lag $\varepsilon$ of data stream $S_i^{(w)} = (s_{i,1}, s_{i,2}, \cdots, s_{i,w})$ is denoted as $S_i^{(w)}(\varepsilon) = (s_{i,w+1}, s_{i,w+2}, \cdots, s_{i,w+\varepsilon})$. Lag $\varepsilon$ is a positive integer. The lagged correlation $\rho_{ij}^{(w)}(\varepsilon)$ between two data streams $S_i^{(w)}$ and $S_j^{(w)}$ at lag $\varepsilon$ is computed by the following equation:

$$\rho_{ij}^{(w)}(\varepsilon) = \frac{\sum_{\tau=t-w+1}^{t-\varepsilon} (s_{i,\tau+\varepsilon} - \overline{s}_i)(s_{j,\tau} - \overline{s}_j)}{\sigma_i \sigma_j}, \quad (2)$$

where $\overline{s}_i$ and $\overline{s}_j$ are the mean values in the shifted sliding window $[t-w+1+\varepsilon, t]$ and $[t-w+1, t-\varepsilon]$, respectively, and $\sigma_i$ and $\sigma_j$ are the standard deviations in $[t-w+1+\varepsilon, t]$ and $[t-w+1, t-\varepsilon]$, respectively. Once $\varepsilon = 0$, $\rho_{ij}^{(w)}(0)$ is the local Pearson's correlation. In a stream context, the lagged correlation can be computed as follows[26]

$$\overline{\rho}_{ij}^{(w)}(\varepsilon) = \frac{\psi_{ij}^{(w)}(\varepsilon) - \frac{\sum_{t-w+1+\varepsilon}^{t} s_i \sum_{t-w+1}^{t-\varepsilon} s_j}{w-\varepsilon}}{\sigma_i \sigma_j}, \quad (3)$$

where $\psi_{ij}^{(w)}(\varepsilon)$ is the inner product between the shifted windows $S_i$ $[t-w+1+\varepsilon, t]$ and $S_j$ $[t-w+1, t-\varepsilon]$, $\sum_{t-w+1+\varepsilon}^{t} s_i$ and $\sum_{t-w+1}^{t-\varepsilon} s_j$ are the sum over the two shifted windows, respectively, and $\sigma_i$ can be computed as follows

$$\sigma_i = \sqrt{\sum_{t-w+1+\varepsilon}^{t} (s_i)^2 - \frac{\left(\sum_{t-w+1+\varepsilon}^{t} s_i\right)^2}{w-\varepsilon}}, \quad (4)$$

where $\sum_{t-w+1+\varepsilon}^{t} (s_i)^2$ denotes the sum of the squares of the shifted window $[t-w+1+\varepsilon, t]$. The value of $\sigma_j$ can be computed similarly in the shifted window $[t-w+1, t-\varepsilon]$.

In the case of multiple data streams, two different data streams are regarded as two independent random variables. The recent data are usually more important than the aged data. To reflect the bias toward data, the weighted correlation measurement for incremental updating will be chosen by this works[25]. Given a fading function $f(t)$, at one time step $t$, the correlation coefficient between two data streams $S_i$ and $S_j$ based on Pearson correlation can be defined as follows

$$\overline{\rho}_{ij}^{(w)}(\varepsilon) = \frac{\psi_{ij}^{(w)}(\varepsilon) - \frac{\sum_{t-w+1+\varepsilon}^{t} f(t) s_i \sum_{t-w+1}^{t-\varepsilon} f(t) s_j}{w-\varepsilon}}{\sigma_i' \sigma_j'}, \quad (5)$$

where $f(t)$ is a monotonically non-decreasing function of time index $t$ [25], and $\sigma_i'$ can be computed as follows

$$\sigma_i' = \sum_t f(t)(s_i)^2 - \frac{\left(\sum_t f(t) s_i\right)^2}{\sum_t f(t)}. \quad (6)$$

At different time steps, the data size or the data structure might change. We observe a set of new edges, with associated edge weights. The adjacency matrix $X$ can be built by incrementally updating its data records in a "block-wise" manner.

### 3.3. Low Rank Approximation

In this paper, the problem for clustering multiple sensor data streams is treated as a matrix decomposition problem. It is assumed that the large graphs are constructed by the multiple sensor data streams which can generate a large volume of event records at time $t$. At each time step, by exploiting the "smoothness" between two consecutive time steps, the adjacency matrix $X$ will be incrementally updated. Once the adjacency matrix $X \in \mathbb{R}^{n \times n}$ is constructed, the next step is going to be matrix decomposition and error measure.

To find patterns and detect anomalies, low rank approximation is a good indicator to identify the community in the graph. Such as SVD, CUR, CMD, and the family of Colibri can be applied to the adjacency matrix $X$ for generating a low-rank approximation. A unique and independent subspace $C$ is formed with an iterative procedure, and subspace $C$ is constructed by using the biased sampling method [12]. The family of Colibri is to eliminate linearly dependent columns to construct the subspace used for low rank approximation from the data matrix $X$ [23]. The subspace $C$ is initialized with $C = C_0(:,1)$ and the core matrix $U$ is written as $U = (C^{\mathrm{T}}C)^{-1}$. The approximation matrix $\tilde{X}$ can be written as $\tilde{X} = C_s U R_s^{\mathrm{T}}$, where $C_s \in \mathbb{R}^{n \times c'}$ ($R_s \in \mathbb{R}^{n \times c}$) contain $c(r)$ scaled columns (rows) sampled from $X$, and $U \in \mathbb{R}^{c' \times c'}$ is a small dense matrix which can be computed from $C_s$ and $R_s$. The core matrix $U = (C^{\mathrm{T}}C)^{-1}$ is

the Moore-Penrose pseudoinverse of the square matrix $C^{\mathrm{T}}C$. In the next iterative process, the $i$th column of matrix $C_0$ is checked to see if it is linearly dependent on the current columns of $C$. If not, this column is appended to $C$ and the core matrix U is updated; else, this sample is discarded. In the end, $C$ is obtained by eliminating all the redundant columns from $C_0$.

If the subspace matrix $C$ is given, the approximation of the original adjacency matrix $X$ can be written as $\tilde{X} = C(C^{\mathrm{T}}C)^{-1}C^{\mathrm{T}}X$. The core matrix $U$ satisfies $U = (C^{\mathrm{T}}C)^{-1}$. $\mathrm{R}^{\mathrm{T}}$ is defined as $C^{\mathrm{T}}X$. The final approximation of $X$ can be written as $\tilde{X} = CUR^{\mathrm{T}}$.

The metric for approximation error is useful for anomaly detection, where a sudden large error may suggest structural changes in the multiple sensor data streams. The square of the Euclidean distance and the generalized Kullback-Leibler divergence are common metric. In this study, we minimize the distance function $D(X \,||\, \tilde{X})$ between the adjacency matrix $X$ and the product of latent factors $\tilde{X}$. The distance function is defined as follows

$$D(X \,||\, \tilde{X}) = \left\| X - \tilde{X} \right\|_F^2 = \sum_{ij} (X_{ij} - \tilde{X}_{ij}) . \tag{7}$$

Note that if $\sum_{ij} X_{ij} = \sum_{ij} \tilde{X}_{ij} = 1$, the distance function reduces to Kullback-Leibler divergence measure.

## 4. Matrix Factorization-Based Incremental Clustering Framework (IC-MF)

### 4.1. Clustering Framework for Multiple Sensor Data Streams

To achieve an incremental clustering in a sliding window $w$, the graphs can be constructed periodically (e.g., one graph per block). For example, we can construct graph $G^{(m)}$ and $G^{(m-1)}$ for time block $m$ and $m$–1, respectively. Let $X^{(m)}$ and $X^{(m-1)}$ be the adjacency matrix associated with graphs $G^{(m)}$ and $G^{(m-1)}$, respectively. In addition, it is assumed that there is a third set of edges connecting the nodes between $G^{(m)}$ and $G^{(m-1)}$. We denote the adjacency matrix for these edges as $X^{(m-1,m)}$. When a new time block $m$–1 is coming, we can construct the adjacency matrix $X$ by updating its entries as data records are coming in. Each new record triggers an update on an entry $(i, j)$ with a value increase of $\Delta\rho$, i.e., $X^{(t)}(i, j) = X^{(t)}(i, j) + \Delta\rho$, we can get adjacency matrix $X^{(m+1)}$. Then, we will update the matrix $C_0^{(m+1)}$ and try to identify those linearly independent columns $C^{(m+1)}$ within $C_0^{(m+1)}$ as well as the core matrix $U^{(m+1)}$. The core matrix $U^{(m+1)}$ is updated simultaneously with more columns added in $X^{(m+1)}$. At the end, $R^{(m+1)}$ is computed by the following equation: $(R^{(m+1)})^{\mathrm{T}} = (C^{(m+1)})^{\mathrm{T}} X^{(m+1)}$.

To consider the similarity between two consecutive time steps, the previous results and the correlation between two blocks will be embedded in the model with the cost of deviating from an accurate representation of the current data. We can minimize the objective function in equation (8) with respect to $R^{(m)}$, $R^{(m-1)}$, $U^{(m)}$, $U^{(m-1)}$, and $U^{(m-1,m)}$:

$$\begin{aligned} J = {} & D(\tilde{X}^{(m)} \,||\, C_X^{(m)} U^{(m)} R^{(m)^{\mathrm{T}}}) + \\ & D(\tilde{X}^{(m-1)} \,||\, C_X^{(m-1)} U^{(m-1)} R^{(m-1)^{\mathrm{T}}}) + , \\ & D(\tilde{X}^{(m-1,m)} \,||\, R^{(m)} U^{(m-1,m)} R^{(m-1)^{\mathrm{T}}}) \end{aligned} \tag{8}$$

where $\tilde{X}^{(m)}$ is the approximation of the adjacency matrix $X^{(m)}$, $C_X^{(m)}$ is the subspace of the adjacency matrix $X^{(m)}$, $U^{(m)}$ is the weight matrix at $m$, $R^{(m)}$ is the cluster indicator matrix at $m$, and similar definitions are made for $\tilde{X}^{(m-1)}$, $C_X^{(m-1)}$ and $U^{(m-1)}$. Next, we interpret the role of $\tilde{X}^{(m-1,m)}$ and $U^{(m-1,m)}$ matrix in the objective function. $\tilde{X}^{(m-1,m)}$ is the approximation of the adjacency matrix $X^{(m-1,m)}$. $U^{(m-1,m)}$ reflects the correspondence between the subgroups derived from $X^{(m-1,m)}$. Indeed, the objective function can be rewritten as follows

$$\begin{aligned} J = {} & \\ & \min \left\{ \left( \sum_{ij} \tilde{X}_{ij}^{(m)} \log \frac{\tilde{X}_{ij}^{(m)}}{[C_X^{(m)} U^{(m)} R^{(m)^{\mathrm{T}}}]_{ij}} - \tilde{X}_{ij}^{(m)} + (C_X^{(m)} U^{(m)} R^{(m)^{\mathrm{T}}})_{ij} \right) + \right. \\ & \left( \sum_{ks} \tilde{X}_{ks}^{(m-1)} \log \frac{\tilde{X}_{ks}^{(m-1)}}{[C_X^{(m-1)} U^{(m-1)} R^{(m-1)^{\mathrm{T}}}]_{ks}} - \tilde{X}_{ks}^{(m-1)} + [C_X^{(m-1)} U^{(m-1)} R^{(m-1)^{\mathrm{T}}}]_{ks} \right) + \\ & \left. \left( \sum_{ik} \tilde{X}_{ik}^{(m-1,m)} \log \frac{\tilde{X}_{ik}^{(m-1,m)}}{[R^{(m)} U^{(m-1,m)} R^{(m-1)^{\mathrm{T}}}]_{ks}} - \tilde{X}_{ik}^{(m-1,m)} + [R^{(m)} U^{(m-1,m)} R^{(m-1)^{\mathrm{T}}}]_{ks} \right) \right\} \end{aligned} \tag{9}$$

We take the partial derivatives of $J$ with respect to $R^{(m)}$ and $R^{(m-1)}$ in equations (10) and (11) (the partial derivatives with respect to $U^{(m)}$ and $U^{(m-1)}$ can be similarly derived).

$$\begin{aligned} \frac{\partial J}{\partial R_{ij}^{(m)}} = {} & \sum_{a=1}^{N} \left[ \frac{-\tilde{X}_{ia}^{(m)} [C_X^{(m)} U^{(m)}]_{aj}}{[C_X^{(m)} U^{(m)} R^{(m)^{\mathrm{T}}}]_{ai}} + [C_X^{(m)} U^{(m)}]_{aj} \right] \\ & + \sum_{a=1}^{M} \left[ \frac{-\tilde{X}_{ia}^{(m-1,m)} [R^{(m-1)} U^{(m-1,m)^{\mathrm{T}}}]_{aj}}{[R^{(m)} U^{(m-1,m)} R^{(m-1)^{\mathrm{T}}}]_{ia}} + [R^{(m-1)} U^{(m-1,m)^{\mathrm{T}}}]_{aj} \right], \end{aligned} \tag{10}$$

$$\frac{\partial J}{\partial R_{ij}^{(m-1)}} = \sum_{a=1}^{M}\left[\frac{-\tilde{X}_{ia}^{(m-1)}[C_X^{(m-1)}U^{(m-1)}]_{aj}}{[C_X^{(m-1)}U^{(m-1)}R^{(m-1)^{\mathrm{T}}}]_{ai}} + [C_X^{(m-1)}U^{(m-1)}]_{aj}\right] + \sum_{a=1}^{N}\left[\frac{-\tilde{X}_{ia}^{(m-1,m)}[R^{(m)}U^{(m-1,m)}]_{aj}}{[R^{(m)}U^{(m-1,m)}R^{(m-1)^{\mathrm{T}}}]_{ia}} + [R^{(m)}U^{(m-1,m)}]_{aj}\right], \quad (11)$$

Given the objective function (9) and its partial derivatives, the updates of the these matrices $R^{(m)}$, $R^{(m-1)}$, $U^{(m)}$, $U^{(m-1)}$ and $U^{(m-1,m)}$ can be solved by using a gradient descent approach. Now we give an iterative matrix factorization with asymptotic convergence based on updating rules:

$$R_{ij}^{(m)} \leftarrow R_{ij}^{(m)} \times \frac{\sum_{a=1}^{N}\frac{\tilde{X}_{ia}^{(m)}[C_X^{(m)}U^{(m)}]_{aj}}{[C_X^{(m)}U^{(m)}R^{(m)^{\mathrm{T}}}]_{ai}} + \sum_{a=1}^{M}\frac{\tilde{X}_{ia}^{(m-1,m)}[R^{(m-1)}U^{(m-1,m)^{\mathrm{T}}}]_{aj}}{[R^{(m)}U^{(m-1,m)}R^{(m-1)^{\mathrm{T}}}]_{ia}}}{\sum_{a=1}^{N}[C_X^{(m)}U^{(m)}]_{aj} + \sum_{a=1}^{M}[R^{(m-1)}U^{(m-1,m)^{\mathrm{T}}}]_{aj}}, \quad (12)$$

$$R_{ij}^{(m-1)} \leftarrow R_{ij}^{(m-1)} \times \frac{\sum_{a=1}^{M}\frac{\tilde{X}_{ia}^{(m-1)}[C_X^{(m-1)}U^{(m-1)}]_{aj}}{[C_X^{(m-1)}U^{(m-1)}R^{(m-1)^{\mathrm{T}}}]_{ai}} + \sum_{a=1}^{N}\frac{\tilde{X}_{ia}^{(m-1,m)}[R^{(m)}U^{(m-1,m)}]_{aj}}{[R^{(m)}U^{(m-1,m)}R^{(m-1)^{\mathrm{T}}}]_{ia}}}{\sum_{a=1}^{M}[C_X^{(m-1)}U^{(m-1)}]_{aj} + \sum_{a=1}^{N}[R^{(m)}U^{(m-1,m)}]_{aj}}, \quad (13)$$

$$U_{ij}^{(m-1,m)} \leftarrow U_{ij}^{(m-1,m)} \times \frac{\sum_{a=1}^{N}\sum_{b=1}^{M}\left(\frac{\tilde{X}_{ab}^{(m-1,m)}}{[R^{(m)}U^{(m-1,m)}R^{(m-1)^{\mathrm{T}}}]_{ab}}[R^{(m)}]_{ai}[R^{(m-1)}]_{bj}\right)}{\left(\sum_{a=1}^{N}\sum_{b=1}^{M}[R^{(m)}]_{ai}[R^{(m-1)}]_{bj}\right)}, \quad (14)$$

$$U_{ij}^{(m)} \leftarrow U_{ij}^{(m)} \times \frac{\sum_{a=1}^{N}\sum_{b=1}^{N}\left(\frac{\tilde{X}_{ab}^{(m)}}{[C_X^{(m)}U^{(m)}R^{(m)^{\mathrm{T}}}]_{ab}}[C_X^{(m)}]_{ai}[R^{(m)}]_{bj}\right)}{\left(\sum_{a=1}^{N}\sum_{b=1}^{N}[C_X^{(m)}]_{ai}[R^{(m)}]_{bj}\right)}, \quad (15)$$

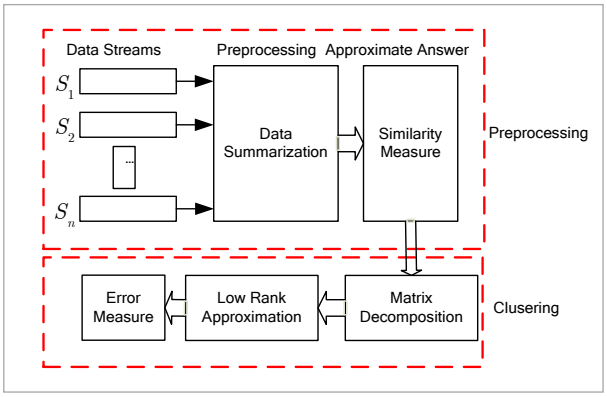
$$U_{ij}^{(m-1)} \leftarrow U_{ij}^{(m-1)} \times \frac{\sum_{a=1}^{M}\sum_{b=1}^{M}\left(\frac{\tilde{X}_{ab}^{(m-1)}}{[C_X^{(m-1)}U^{(m-1)}R^{(m-1)^{\mathrm{T}}}]_{ab}}[C_X^{(m-1)}]_{ai}[R^{(m-1)}]_{bj}\right)}{\left(\sum_{a=1}^{N}\sum_{b=1}^{N}[C_X^{(m-1)}]_{ai}[R^{(m-1)}]_{bj}\right)}. \quad (16)$$

The proofs of correctness and convergence of the update formulas are given in Section 5.

## 4.2. Algorithm Description

At the beginning of this section, the IC-MF framework is shown in Figure 2. It consists of two phases: the preprocessing phase and the clustering phase. In the first maintaining phase, the coming data streams have statistical property are statistics, and will be preprocessed. We will try to reveal the cluster evolutionary in the clustering phase.

**Figure 2**
IC-MF Framework




Algorithm 1 summarizes the whole procedure of the proposed IC-MF. Based on the statistical and similarity measure, suppose the adjacency matrices $X^{(m)}$, $X^{(m-1)}$ and $X^{(m-1,m)}$ are given, as well as the clustering results including $U^{(m-1)}$ and $R^{(m-1)}$. Now we need to obtain $U^{(m)}$ and $G^{(m)}$, as well as the new representative subspace and approximation. The variable factors are initialized by the previous clustering results instead of random values to improve the clustering efficiency [10].

**Algorithm 1:** Iteration Updating

Input: Matrices $X^{(m)}$, $X^{(m-1)}$, $X^{(m-1,m)}$, and maximum iteration $T$

Output: Matrices $R^{(m)}$, $R^{(m-1)}$, and $U^{(m-1,m)}$

1 Initialize $R^{(m)}$, $R^{(m-1)}$, $U^{(m)}$, $R^{(m-1)}$ and $U^{(m-1,m)}$
2 Repeat
3 For $i = 1 : T$ do
4 Update $R^{(m)'}$ via Eq. (12)
5 Update $R^{(m-1)'}$ via Eq. (13) with $R^{(m)}$
6 Update $U^{(m-1,m)'}$ via Eq. (14) with $R^{(m)}$ and $R^{(m-1)}$
7 Update $U^{(m)'}$ via Eq. (15) with $R^{(m)}$
8 Update $U^{(m-1)'}$ via Eq. (16) with $R^{(m-1)}$

Set $R^{(m)} \leftarrow R^{(m)'}$, $R^{(m-1)} \leftarrow R^{(m-1)'}$ $U^{(m)} \leftarrow U^{(m)'}$, $U^{(m-1)} \leftarrow U^{(m-1)'}$ and $U^{(m-1,m)} \leftarrow U^{(m-1,m)'}$;
If (converged) then
Break
End if
End for
Until converged
Return $R^{(m)}$, $R^{(m-1)}$, and $U^{(m-1,m)}$

Since the correlation between data streams will be changed, IC-MF reports cluster results through matrix factorization and low rank approximation for multiple sensor data streams, which is an incremental clustering algorithm. This algorithm can deal with the problem of continuously monitoring clustering structures of multiple sensor data streams from sensor networks. The adjacency matrix is constructed and analyzed based on the data streams generated by the sensors of applications. In Algorithm 2, the clustering result is generated from each sliding window $w$, in which $w$ depends on the user. The number of clusters must be recalculated in each $w$. As soon as the correlation coefficients in $w$ are available for all streams, the iteration updating algorithm is used to obtain the clustering results. As the time goes on, the results $\mathbb{C}^{(m)}$ of cluster at the new time step are compared with the cluster results $\mathbb{C}^{(current)}$ at current time step, so the evolution of event can be mined. These results can be fed into different mining applications such as anomaly detection and state analysis.

**Algorithm 2:** *IC – MF* ($S, s, e, w, l$)

Input: Data set $S$, start time $s$, end time $e$, time horizon $w$, length of block $l$;

Output: Evolution event $\Gamma = \{\mathbb{C}^{(1)}, \mathbb{C}^{(2)}, \ldots, \mathbb{C}^{(m-1)}, \mathbb{C}^{(m)}\}$

While $(m = 1 : \lfloor w/l \rfloor)$ {
Construct the adjacency matrices $X^{(m)}$, $X^{(m-1)}$, $X^{(m-1,m)}$;
Determine the cluster number $k^{(m)}$;
Associate the nodes and clusters between two blocks, and $R^{(m)}$, $R^{(m-1)}$, and $U^{(m-1,m)}$ are obtained by Algorithm 1;
If $(\mathbb{C}^{(m)} \neq \mathbb{C}^{(current)})$ {
$T = T \cup \{\mathbb{C}^{(m)}\}$;
$m = m + 1$;
$\mathbb{C}^{(current)} = \mathbb{C}^{(m)}$;
}
}
Return $T$;

# 5. Theoretical Analysis

## 5.1. Correctness and Convergence

We consider the objective function (9), whose constrained solution satisfies KKT (Karush Kuhn- Tucker) complementary conditions under the updating rules in (12)-(16). The proof can be shown by using the auxiliary functions.

**Definition 1.** $Z(R^{(m)}, R^{(m-1)})$ is an auxiliary function for $J(R^{(m)})$ if the conditions $Z(R^{(m)}, R^{(m-1)}) \geq J(R^{(m)})$, $Z(R^{(m-1)}, R^{(m-1)}) = J(R^{(m-1)})$ are satisfied.

**Lemma 1:** For any $R^{(m)}$ and $R^{(m-1)}$, if $Z_1(R^{(m)}, R^{(m-1)})$ and $Z_2(R^{(m)}, R^{(m-1)})$ are auxiliary functions for $J_1(R^{(m)})$ and $J_2(R^{(m)})$, respectively, then $Z = Z_1 + Z_2$ is the auxiliary function for $J = J_1 + J_2$. Furthermore, the objective function $J$ decreases monotonically under the condition $R^{(m)} = \arg\min_R Z(R, R^{(m-1)})$.

**Proof:** The proof that $Z$ is an auxiliary function of $J$ is as follows:

$$\begin{aligned} J(R^{(m)}) &= J_1(R^{(m)}) + J_2(R^{(m)}) \\ &\leq Z_1(R^{(m)}, R^{(m-1)}) + Z_2(R^{(m)}, R^{(m-1)}) \\ &\leq Z_1(R^{(m-1)}, R^{(m-1)}) + Z_2(R^{(m-1)}, R^{(m-1)}) \\ &= J(R^{(m-1)}) \end{aligned} \quad (17)$$

Thus, $Z(R^{(m)}, R^{(m-1)}) \leq Z(R^{(m-1)}, R^{(m-1)})$, $J(R^{(m)}) \leq J(R^{(m-1)})$. The proof is complete.

**Theorem 1:** For fixed $R^{(m-1)}$, $U^{(m)}$, $U^{(m-1)}$ and $U^{(m-1,m)}$, the update formula for $R^{(m)}$ given in equation (12), the objective function in (9) decreases monotonically.

**Proof:** Lemma 1 suggests that it is sufficient to show that minimizing the auxiliary function of the summation decreases the overall objective function[4]. To prove the correctness and convergence of our proposed algorithm, here we give the auxiliary function $Z_1$ for the term $D(\tilde{X}^{(m)} \,||\, C_X^{(m)} U^{(m)} R^{(m)^T})$ which is quadratic in $R^{(m)}$. The auxiliary function for other terms can be

similarly derived. First, the objective function can be written as follows:

$$J_1 = \sum_{ij} \tilde{X}_{ij}^{(m)} \log \frac{\tilde{X}_{ij}^{(m)}}{\left[C_X^{(m)} U^{(m)} R^{(m)^{\mathrm{T}}}\right]_{ij}} - \tilde{X}_{ij}^{(m)} + \left[C_X^{(m)} U^{(m)} R^{(m)^{\mathrm{T}}}\right]_{ij}. \quad (18)$$

By minimizing $D(\tilde{X}^{(m)} \,||\, C_X^{(m)} U^{(m)} R^{(m)^{\mathrm{T}}})$, we can obtain the updating rule of $R^{(m)}$ as follows:

$$R_{ij}^{(m)} \leftarrow R_{ij}^{(m)} \frac{\sum_{a=1}^{N} \frac{\tilde{X}_{ja}^{(m)} [C_X^{(m)} U^{(m)}]_{ia}}{[C_X^{(m)} U^{(m)} R^{(m)^{\mathrm{T}}}]_{aj}}}{\sum_{a=1}^{N} [C_X^{(m)} U^{(m)}]_{aj}}. \quad (19)$$

To ensure $Z_1(R, R^{(m)}) \geq J_1(R^{(m)})$, the parameter $\alpha_{k,l}$ is defined as follows:

$$\alpha_{k,l} = \frac{\left(C_X^{(m)}\right)_{ik} U_{kl}^{(m)} R_{lj}^{(m)^{\mathrm{T}}}}{\sum_{rs} \left(C_X^{(m)}\right)_{ir} U_{ks}^{(m)} R_{sj}^{(m)^{\mathrm{T}}}}. \quad (20)$$

Obviously, $\sum_{rs} \alpha_{rs} = 1$. We use convexity of the logarithmic function to derive the following inequality:

$$-\log \sum_{kl} \left(C_X^{(m)}\right)_{ik} U_{kl}^{(m)} R_{lj}^{(m)^{\mathrm{T}}} \leq -\sum_{kl} \alpha_{kl} \frac{\left(C_X^{(m)}\right)_{ik} U_{kl}^{(m)} R_{lj}^{(m)^{\mathrm{T}}}}{\alpha_{kl}}. \quad (21)$$

Substituting (21) in (18), the following expression is obtained

$$D(\tilde{X}^{(m)} \,||\, C_X^{(m)} U^{(m)} R^{(m)^{\mathrm{T}}}) \leq \sum_{ij} \left[ \tilde{X}_{ij}^{(m)} \log \tilde{X}_{ij}^{(m)} - \tilde{X}_{ij}^{(m)} - \tilde{X}_{ij}^{(m)} \sum_{kl} \alpha_{kl} \log \frac{\left(C_X^{(m)}\right)_{ik} U_{kl}^{(m)} R_{lj}^{(m)^{\mathrm{T}}}}{\alpha_{kl}} \times \right. \left. \tilde{X}_{ij}^{(m)} \sum_{kl} \alpha_{kl} \log \frac{\left(C_X^{(m)}\right)_{ik} U_{kl}^{(m)} R_{lj}^{(m)^{\mathrm{T}}}}{\alpha_{kl}} + \sum_{kl} \left(C_X^{(m)}\right)_{ik} U_{kl}^{(m)} R_{lj}^{(m)^{\mathrm{T}}} \right]. \quad (22)$$

Considering $\alpha_{k,l}$ in equation (22), the following objective function is obtained

$$\sum_{ij} \left[ \tilde{X}_{ij}^{(m)} \log \tilde{X}_{ij}^{(m)} - \tilde{X}_{ij}^{(m)} - \tilde{X}_{ij}^{(m)} \sum_{kl} \frac{\left(C_X^{(m)}\right)_{ik} U_{kl}^{(m)} R_{lj}^{(m)^{\mathrm{T}}}}{\sum_{rs} \left(C_X^{(m)}\right)_{ir} U_{ks}^{(m)} R_{sj}^{(m)^{\mathrm{T}}}} \times \left( \log \left( \left(C_X^{(m)}\right)_{ik} U_{kl}^{(m)} R_{lj}^{(m)^{\mathrm{T}}} \right) - \log \frac{\left(C_X^{(m)}\right)_{ik} U_{kl}^{(m)} R_{lj}^{(m)^{\mathrm{T}}}}{\sum_{rs} \left(C_X^{(m)}\right)_{ir} U_{ks}^{(m)} R_{sj}^{(m)^{\mathrm{T}}}} \right) + \sum_{kl} \left(C_X^{(m)}\right)_{ik} U_{kl}^{(m)} R_{lj}^{(m)^{\mathrm{T}}} \right]. \quad (23)$$

Equation (23) is the auxiliary function of $J_1$. We denote it as $Z_1(R, R^{(m)})$. Taking derivative of $Z_1$ with respect to $R_{pq}^{(m)}$, we have

$$\frac{\partial Z_1}{\partial R_{pq}^{(m)}} = -\sum_{ik} \tilde{X}_{ip}^{(m)} \frac{\left(C_X^{(m)}\right)_{ik} U_{kq}^{(m)} R_{qp}^{(m)^{\mathrm{T}}}}{\sum_{rs} \left(C_X^{(m)}\right)_{ir} U_{rs}^{(m)} R_{sp}^{(m)^{\mathrm{T}}} R_{pq}^{(m)}} + \sum_{ik} \left(C_X^{(m)}\right)_{ik} U_{kq}^{(m)}. \quad (24)$$

Letting $\partial Z_1 / \partial R_{pq}^{(m)} = 0$, we obtain the updating rule as follows

$$R_{pq}^{(m)} \leftarrow R_{pq}^{(m)} \frac{\sum_i \frac{\tilde{X}_{ip}^{(m)} [C_X^{(m)} U^{(m)}]_{iq}}{[C_X^{(m)} U^{(m)} R^{(m)^T}]_{ip}}}{\sum_{i=1}^{N} [C_X^{(m)} U^{(m)}]_{iq}}. \quad (25)$$

Equation (25) is the same as (19). Similarly, the term $D(\tilde{X}^{(m-1,m)} \,||\, R^{(m)} U^{(m-1,m)} R^{(m-1)^{\mathrm{T}}})$ can be written as

$$J_2 = \sum_{ij} \tilde{X}_{ij}^{(m-1,m)} \log \frac{\tilde{X}_{ij}^{(m-1,m)}}{\sum_{kl} R_{ik}^{(m)} U_{kl}^{(m-1,m)} R_{lj}^{(m-1)^T}} - \tilde{X}_{ik}^{(m-1,m)} + \sum_{kl} R_{ik}^{(m)} U_{kl}^{(m-1,m)} R_{lj}^{(m-1)^{\mathrm{T}}}. \quad (26)$$

$\beta_{kl}$ is defined as follows

$$\beta_{kl} = \frac{R_{ik}^{(m)} U_{kl}^{(m-1,m)} R_{lj}^{(m-1)^{\mathrm{T}}}}{\sum_{rs} R_{ir}^{(m)} U_{rs}^{(m-1,m)} R_{sj}^{(m-1)^{\mathrm{T}}}}. \quad (27)$$

When $\sum_{kl} \beta_{kl} = 1$, the auxiliary function of $D(\tilde{X}^{(m-1,m)} \,||\, R^{(m)} U^{(m-1,m)} R^{(m-1)^{\mathrm{T}}})$ is obtained as

$$Z_2(R, R^{(m)}) = \sum_{ij} \left[ \tilde{X}_{ij}^{(m-1,m)} \log \tilde{X}_{ij}^{(m-1,m)} - \tilde{X}_{ij}^{(m-1,m)} - \tilde{X}_{ij}^{(m-1,m)} \sum_{kl} \frac{R_{ik}^{(m)} U_{kl}^{(m-1,m)} R_{lj}^{(m-1)^{\mathrm{T}}}}{\sum_{rs} R_{ir}^{(m)} U_{rs}^{(m-1,m)} R_{sj}^{(m-1)^T}} \log R_{ik}^{(m)} U_{kl}^{(m-1,m)} R_{lj}^{(m-1)^{\mathrm{T}}} - \log \frac{R_{ik}^{(m)} U_{kl}^{(m-1,m)} R_{lj}^{(m-1)^{\mathrm{T}}}}{\sum_{rs} R_{ir}^{(m)} U_{rs}^{(m-1,m)} R_{sj}^{(m-1)^{\mathrm{T}}}} + R_{ik}^{(m)} U_{kl}^{(m-1,m)} R_{lj}^{(m-1)^{\mathrm{T}}} \right]. \quad (28)$$

Taking derivative of equation (28) with respect to $R_{pq}^{(m)}$ and equating it to zero, we have

$$R_{pq}^{(m)} \leftarrow R_{pq}^{(m)} \frac{\sum_{i=1}^{M} \frac{\tilde{X}_{pi}^{(m-1,m)} [R^{(m-1)} U^{(m-1,m)^{\mathrm{T}}}]_{iq}}{[R^{(m)} U^{(m-1,m)} R^{(m-1)^{\mathrm{T}}}]_{pi}}}{(\sum_{i=1}^{M} [R^{(m-1)} U^{(m-1,m)^{\mathrm{T}}}]_{iq})}. \quad (29)$$

Minimizing the original objective function (8) with respect to $R^{(m)}$, we have

$$\min_{R^{(m)}} J(R^{(m)}) = \min_{R^{(m)}} \left( J_1(R^{(m)}) + J_2(R^{(m)}) \right) \leq \min_{R^{(m)}} \left( Z_1(R, R^{(m)}) + Z_2(R, R^{(m)}) \right). \quad (30)$$

Taking derivative of $Z_1(R, R^{(m)}) + Z_2(R, R^{(m)})$ with respect to $R^{(m)}$ and equating it to zero we get update formula (12). By **Lemma 1**, we know that the objective function in (30) is monotonically decreasing. The proof is complete.

Similarly, by reversing the roles of $R^{(m)}$, $U^{(m)}$, $U^{(m-1)}$ and $U^{(m-1,m)}$, the update rule for $R^{(m-1)}$ has been given in (13). The objective function is decreasing monotonically and the other three update rules are all similar.

# 6. Experimental Results

## 6.1. Evaluation Methods

The candidates for comparison include normalized cut by Shi and Malik [20]. In this paper, Ncut and Ncut-M refer to normalized cut algorithm applied on the individual block and multi-block, respectively. We also compare it with the other algorithms at the same objective function given in (9). For a comparison, we applied the following evaluation metrics.

To evaluate the clustering quality, all our comparisons are conducted using Normalized mutual information (NMI). Between two random variables CAT (Category Label) and CLS (Cluster Label), Normalized Mutual Information (NMI) is defined as follows [25]

$$\mathrm{NMI}(CAT, CLS) = \frac{I(CAT, CLS)}{\sqrt{H(CAT)H(CLS)}}, \quad (31)$$

where $I(CAT, CLS)$ is the mutual information between $CAT$ and $CLS$.

The entropies $H(CAT)$ and $H(CLS)$ are used for normalizing the mutual information to be in the range of [0, 1]. Before presenting the results of quality evaluation, we discuss the definition of NMI. The NMI can be depicted as:

$$\mathrm{NMI} = \frac{\sum_{i=1}^{k^{(a)}} \sum_{j=1}^{k^{(b)}} n_{i,j} \log \left( \frac{n \cdot n_{i,j}}{n_i \cdot n_j} \right)}{\sqrt{\left( \sum_{i=1}^{k^{(a)}} n_i \log \frac{n_i}{n} \right) \left( \sum_{j=1}^{k^{(b)}} n_j \log \frac{n_j}{n} \right)}}, \quad (32)$$

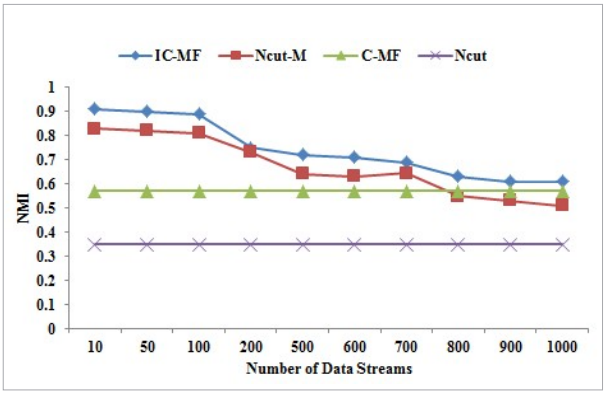
where $n$ is the number of documents, $n_i$ and $n_j$ denote the number of documents in category $i$ and cluster $j$, respectively, and $n_{i,j}$ denotes the number of documents in category $i$ as well as in cluster $j$, and $k^{(a)}$ and $k^{(b)}$ are cluster numbers of the true and predicted clusters. The NMI score is **1** if the clustering results perfectly match the category labels, and the score is **0** if data are randomly partitioned. The higher the NMI score, the better the clustering quality.

The CPU spends time in computing the output matrices as the metric to quantify the computational expense. Note that all the experiments are implemented on the same machine. At each time step, for each given clustering number, 10 times run will be implemented in the experiment and the average value of these 10 times run results are obtained.

## 6.2. Synthetic Datasets

In this section, we apply synthetic datasets to demonstrate that IC-MF can efficiently cluster data for multiple evolving data streams over time. The number of nodes and the number of clusters at each time step are given. The synthetic datasets are generated by applying prototype systems used in [2], which is defined as follows: $f(t + \Delta t) = f(t) + f'(t + \Delta t)$, $f'(t + \Delta t) = f'(t) + u(t)$, in which $f(\cdot)$ is a stochastic process, $t = 0, \Delta t, 2\Delta t, \ldots$, $u(t)$ are independent random variables uniformly distributed in an interval $[-a, a]$. So, the data stream $S(\cdot)$ can be defined as $S(t) = f(t + h(t)) + g(t)$, where $h(\cdot)$ and $g(\cdot)$ are stochastic processes that are generated in the same way as the prototype $f(\cdot)$. The constant $a$ determines the smoothness of a data stream, and can be different for $p(\cdot)$, $h(\cdot)$ and $g(\cdot)$, for example, 0.05, 0.1

**Figure 3**
The average NMI value with the number of data stream variation



and 0.2, respectively. In this paper, we apply a window size of $w(t)$ seconds to construct an adjacency matrix every $w(t)$ seconds.

To examine the quality of IC-MF for clustering multiple data streams, we generate random datasets with the number of streams ranging from 10 to 1000. Each stream has a length of 2000 points. For each method, the length of the block is 100 points. The average NMI values for three methods of synthetic datasets are plotted in Figure 3. As expected, IC-MF is manifestly more efficient than other three methods. The reason is that IC-MF incorporates the previous clustering results and considers the geometric structure of the data in this experiment.

The performances of four methods are given while the time window evolving over time and the number of clusters in the data varies from 2 to 10. The number of streams is fixed at 400, and each of them contains 1000 points. The data streams are split into 10 windows, and each window consists of 100 data records. In Table 1, the averages NMI among four methods with different time window on synthetic dataset are shown. The IC-MF algorithm has highest NMI values at evolving time steps, so this algorithm greatly outperforms the others. The reason is that it incorporates the last approximation results, clustering results, and the connection between two time steps for clustering multiple evolving data streams.

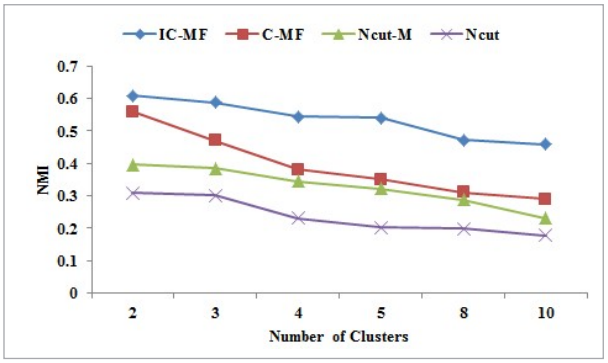
The average NMI values for each method with different clusters are shown in Figure 4. We can find that the NMI value decreases by applying these methods as the numbers of clusters increase. This is due to a greater number of clusters in the dataset, and the clusters become closer to each other. The algorithms merge some of the clusters and lead to the NMI values of the clustering algorithms decrease.

Finally, the average processing time is evaluated for a round of clustering multiple data streams. There are different factors which can affect the execution time. In the experiment, the effects of the window size and the number of data streams on the response time are evaluated by the compared algorithms.

**Table 1**
The NMI among four methods with different time windows

| Time step | Ncut | Ncut-M | C-MF | IC-MF |
|---|---|---|---|---|
| 1 | 0.35 | 0.42 | 0.54 | 0.73 |
| 2 | 0.38 | 0.41 | 0.53 | 0.69 |
| 3 | 0.39 | 0.39 | 0.66 | 0.78 |
| 4 | 0.31 | 0.43 | 0.54 | 0.71 |
| 5 | 0.34 | 0.41 | 0.69 | 0.74 |
| 6 | 0.32 | 0.45 | 0.51 | 0.68 |
| 7 | 0.33 | 0.39 | 0.67 | 0.72 |
| 8 | 0.3 | 0.41 | 0.57 | 0.75 |
| 9 | 0.29 | 0.36 | 0.51 | 0.68 |
| 10 | 0.36 | 0.37 | 0.53 | 0.79 |

**Figure 4**
The average NMI value with the number of clusters variation




The following experiment, the effect of window size on the execution time is evaluated by four algorithms: Ncut, Ncut-M, C-MF and IC-MF. In this experiment, the window size $w(t)$ from 300 to 3600, the number of data streams is fixed at 400, each of data streams contains 1000 points, and the dataset contains 4 clusters. In Figure 5, as the window size increases, the average processing time of these algorithms increases. The $y$-axis shows the execution time, and the $x$-axis represents the window size $w(t)$. Although the time increases for all these algorithms, the IC-MF algorithm is superior to the others.

In Figure 6, the $y$-axis shows the execution time, while the $x$-axis represents the number of data streams, and the number of data streams varies from 10 to 1000. The other parameters are fixed as follows: $w(t) = 300$s, $k = 4$. The processing time of C-MF for the clustering data streams is presented on the present time step. Note that, as the number of streams increases, the execution time of these methods increases. However,

**Figure 5**
The average processing time when the window size $w(t)$ varies from 300 to 3600

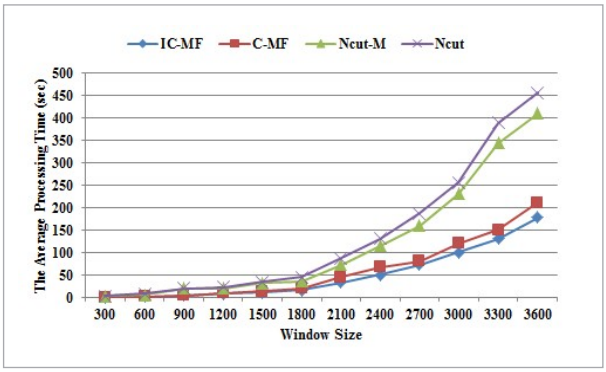
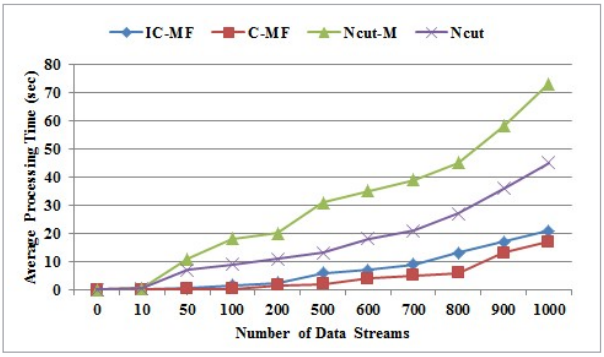



**Figure 6**
The average processing time when the number of streams varies




the time of the IC-MF increases much more slowly than others. Since IC-MF and C-MF are all in the approximation subspace, the execution efficiencies of these two methods are higher than Ncut-M and Ncut.

### 6.3. Real World Datasets

The highway traffic monitoring system is a typical application of the CPS. In real experiments, we employ multiple sensor data streams of transportation system to evaluate the performance of IC-MF. The datasets are collected from the PeMS traffic monitoring system. With the sensor devices installed on road networks, the monitoring system watches the traffic flow of major U.S highways in 24 hours × 7 days. The datasets are collected from over 4000 sensors on 38 highways.

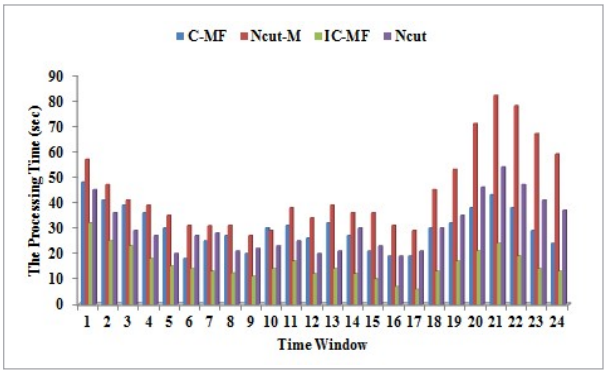
In the following experiment, the quality of the NMI on real world datasets is evaluated by these algorithms. In this experiment, the window size $w(t)$ = 3600s, and the number of data streams is fixed at 1000. We extract records generated in some day. In Table 2, the average values of NMI among four methods during 00:00:00 to 23:59:59 on real world dataset are shown. The IC-MF has more outperformance than all the other approaches, and it also has highest NMIs at evolving time steps. The reason is that the IC-MF incorporates the last approximation results, clustering results, and the connection between two time steps for clustering multiple evolving data streams over time.

Applying the real world datasets, we investigate the relationship among multiple data streams evolve

**Table 2**
Average NMI on Real World Dataset

| Time/hours | Ncut | Ncut-M | C-MF | IC-MF |
|---|---|---|---|---|
| 1 | 0.28 | 0.3 | 0.35 | 0.42 |
| 2 | 0.29 | 0.31 | 0.38 | 0.41 |
| 3 | 0.31 | 0.32 | 0.39 | 0.39 |
| 4 | 0.21 | 0.27 | 0.31 | 0.43 |
| 5 | 0.24 | 0.3 | 0.34 | 0.41 |
| 6 | 0.33 | 0.35 | 0.32 | 0.45 |
| 7 | 0.22 | 0.24 | 0.33 | 0.39 |
| 8 | 0.2 | 0.28 | 0.3 | 0.41 |
| 9 | 0.19 | 0.23 | 0.29 | 0.36 |
| 10 | 0.23 | 0.25 | 0.36 | 0.37 |
| 11 | 0.25 | 0.27 | 0.35 | 0.42 |
| 12 | 0.28 | 0.31 | 0.38 | 0.41 |
| 13 | 0.29 | 0.3 | 0.39 | 0.39 |
| 14 | 0.22 | 0.3 | 0.31 | 0.43 |
| 15 | 0.19 | 0.29 | 0.34 | 0.41 |
| 16 | 0.21 | 0.25 | 0.32 | 0.45 |
| 17 | 0.2 | 0.27 | 0.33 | 0.39 |
| 18 | 0.19 | 0.29 | 0.3 | 0.41 |
| 19 | 0.17 | 0.29 | 0.29 | 0.36 |
| 20 | 0.23 | 0.3 | 0.36 | 0.37 |
| 21 | 0.21 | 0.31 | 0.33 | 0.39 |
| 22 | 0.22 | 0.3 | 0.3 | 0.41 |
| 23 | 0.19 | 0.25 | 0.29 | 0.36 |
| 24 | 0.2 | 0.29 | 0.36 | 0.37 |

**Figure 7**
The average processing time on Real World Datasets




in a period of time. By the records from the highway traffic monitoring system, we apply a window size of $w(t)$ = 300s to construct an adjacency matrix. As shown in Figure 7, C-MF is faster than IC-MF method. The reason is that the objective function of IC-MF method has taken the previous results into the current time block. IC-MF is applied to the clustering multiple data streams by matrix factorization in the low rank approximation subspace, so IC-MF has more outperformance than Ncut-M. In addition, IC-MF takes less processing time than the other algorithms.

## 6.4. NHST (Null Hypothesis Significance Testing) of Algorithm Out-Performance

To verify the superiority and effectiveness of the proposed algorithm, the NHST (Null Hypothesis Significance Testing) of algorithm out-performance is verified by the data sets collected from the PeMS traffic monitoring system. The Wilcoxon signed-ranks test is a nonparametric alternative to the paired $t$-test, which ranks the differences in performances of two classifiers for each data set, ignoring the signs, and compares the ranks for the positive and the negative differences [12].

Let $d_i$ again be the difference between the performance scores of the two classifiers on the $i$-th out of $N$ data sets. The differences are ranked according to their absolute values; average ranks are assigned in case of ties. Let $R^+$ be the sum of ranks for the data sets on which the second algorithm outperformed the first, and $R^-$ the sum of ranks for the opposite. Ranks of $d_i$=0 are split evenly among the sums; if there is an odd number of them, one is ignored:

$$\begin{cases} R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \\ R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \end{cases}. \quad (33)$$

Let $T$ be the smaller of the sums, $T = \min(R^+, R^-)$.

For a larger number of data sets, the statistics are defined as follows

$$Z = \frac{T - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}}, \quad (34)$$

where $N$ is the number of data sets. $Z$ is distributed approximately normally. With a confidence level of $\alpha$ = 0.05, the null-hypothesis can be rejected if Z is smaller than -1.96. In Table 3, the average NMI on Real World Dataset is a part of Table 2.

**Table 3**
Average NMI values of the C-MF and IC-MF on Real World Dataset

| Time/hours | C-MF | IC-MF | Difference | Rank |
|---|---|---|---|---|
| 1 | 0.35 | 0.42 | 0.07 | 6 |
| 2 | 0.38 | 0.41 | 0.03 | 3 |
| 3 | 0.39 | 0.39 | 0 | 1 |
| 4 | 0.31 | 0.43 | 0.08 | 8 |
| 5 | 0.34 | 0.41 | 0.07 | 6 |
| 6 | 0.32 | 0.45 | 0.13 | 10 |
| 7 | 0.33 | 0.39 | 0.06 | 4 |
| 8 | 0.3 | 0.41 | 0.11 | 9 |
| 9 | 0.29 | 0.36 | 0.07 | 6 |
| 10 | 0.36 | 0.37 | 0.01 | 2 |

From Table 3, the sum of ranks for the positive differences is computed as $R^+$ = 6+3+0.5+8+6+10 +4+9+6+2=54.5 and the sum of ranks for the negative differences equals $R^-$ = 0. From (34), we obtain $Z$=-2.8. According to the table of exact critical values for the Wilcoxon's test, for a confidence level of $\alpha$=0.05 and $N$=10 data sets, the difference between the classifiers is significant if $Z$ is smaller than -1.96. We therefore reject the null-hypothesis. This is to say the difference between the two classifiers (C-MF and IC-MF) is significant.

## 7. Conclusions

In this paper, the IC-MF method is proposed to extract and retain meaningful information on large sensor data streams from the traffic CPS. The proposed method is able to monitor the nonstationary data streams, and also can dynamically construct and analyze time- evolving data streams from sensor networks. Compared with previous methods, the proposed IC-MF algorithm can keep updating an existing clustering model to mine the correlation between multiple sensor data streams. IC-MF is an incremental approach of matrix factorization to handle evolving data streams over time. Based on matrix factorization, a framework for clustering multiple data streams evolving over time is proposed. Comparing with summary statistics, the graph theory allows us to model the relationship in multiple sensor data streams more accurately. The effectiveness of the proposed algorithm has been demonstrated by experiment on the real synthetics datasets. The experimental results show the effectiveness and superiority of the proposed IC-MF algorithm in incremental clustering for multiple sensor data streams.

**Acknowledgement**

This work was partly supported by the National Natural Science Foundation of China (No. U1404614), the Henan Province Education Department Foundation (No. 17A413002), the Henan Province Scientific and Technological Foundation (182102310926).

## References

1. Aghbari, Z. A., Kamel, I., Awad, T. On Clustering Large Number of Data Streams. Intelligent Data Analysis, 2012, 16(1), 69-91. https://doi.org/10. 3233/IDA-2011-0511
2. Beringer, J., Hullermeier, E. Online Clustering of Parallel Data Streams. Data and Knowledge Engineering, 2006, 58(2), 180-204. https://doi.org/10.1016/j.datak.2005.05.009
3. Chakrabarti, D., Kumar, R., Tomkins, A. Evolutionary Clustering. Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2006, 554-560. https://doi.org/10.1007/978-0-387-30164- 8_27
4. 4. Chi, Y., Song, X., Zhou, D., Hino, K., Tseng, B. L. Evolutionary Spectral Clustering by Incorporating Temporal Smoothness. Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (ACM 2007), 2007, 153-162. https://doi.org/10.1145/1281 192.1281212
5. 5. Comar, P. M., Tan, P. N., Jain, A. K. A Framework for Joint Community Detection Across Multiple Related Networks. Neurocomputing, 2012, 76(1), 93-104. https://doi.org/10.1016/j.neucom. 2011.04. 035
6. 6. Dai, B. R., Huang, J. W., Yeh, M. Y., Chen, M. S. Adaptive Clustering for Multiple Evolving Streams. IEEE Transactions on Knowledge and Data Engineering, 2006, 18(9), 1166-1180. https://doi.org/10.1109/TKDE.2006.137 https://doi.org/10.1109/TKDE.2006.137
7. Demšar, J. Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Machine Learning Research, 2006, 7(1), 1-30.
8. Desikan, P., Pathak, N., Srivastava, J., Kumar, V. Incremental Page Rank Computation on Evolving Graphs. Special Interest Tracks and Posters of the 14th International Conference on World Wide Web, 2005, 1094-1095. https://doi. org/ 10.1145/1062745.106288
9. 9. Ding, C., He, X. F., Simon, H. D. On the Equivalence of Nonnegative Matrix Factorization and Spectral Clustering. Proceedings of the 2005 SIAM International Conference on Data Mining, 2005, 606-610. https://doi.org/10.1137/1.97816119 72757.70
10. 10. Ding, C., Li, T., Jordan, M. I. Convex and Semi- Nonnegative Matrix Factorizations. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2010, 32(1), 45-55. https://doi.org/10.1109/TPAMI.2008.277. https://doi.org/10.1109/TPAMI.2008.277
11. 11. Domingos, P., Hulten, G. A General Framework for Mining Massive Data Streams. Journal of Computational and Graphical Statistics, 2003, 12(4), 945-949. https://doi.org/10.1198/10618600325 44
12. Drineas, P., Kannan, R., Mahoney, M. W. Fast Monte Carlo Algorithms for Matrices III: Computing a Compressed Approximate Matrix Decomposition. Siam Journal on Computing, 2006, 36(1), 184-206. https://doi.org/10.1137/S00 97539704442702

13. Golub, G. H., Van Loan, C. F. Matrix Computations, 4th Edition. The Johns Hopkins University Press, Baltimore, 2012.

14. Kang, K. D., Basaran, C. Adaptive Data Replication for Load Sharing in a Sensor Data Center. Distributed Computing Systems Work- shops. ICDCS Workshops'09. 29th IEEE International Conference, 2009, 20-25. https:// doi.org/10.1109/ICDCSW.2009.12

15. Lee, D. D., Seung, H. S. Learning the Parts of Objects by Non-Negative Matrix Factorization. Nature, 1999, 401(6755), 788-791. https://doi.org/10.1038/44565

16. Ma, Y. J., Guo, Y. K., Tian, X. C., Ghanem, M. Distributed Clustering-Based Aggregation Algorithm for Spatial Correlated Sensor Networks. IEEE Sensors Journal, 2011, 11(3), 641-648. https://doi.org/10.1109/JSEN.2010.2056916

17. Ning, H., Xu, W., Chi, Y., Gong, Y., Huang, T. S. Incremental Spectral Clustering by Efficiently Updating the Eigen-System. Pattern Recognition, 2010, 43(1), 113-127. https://doi.org/10.1016/j. patcog.2009.06.001

18. Park, K. J., Zheng, R., Liu, X. Cyber-Physical Systems: Milestones and Research Challenges. Computer Communications, 2012, 36(1), 1-7. https://doi.org/10.1016/j. comcom.2012. 09.006

19. Rodrigues, P. P., Gama, J., Lopes, L. Clustering Distributed Sensor Data Streams. Joint European Conference on Machine Learning and Knowledge Discovery in Databases, ECML/ PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part II, 2008, 282-297. https://doi.org/10.1007/978-3-540-87481- 2_19

20. Shi, J. B., Malik, J. Normalized Cuts and Image Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000, 22(8), 888-905. https://doi.org/10.1109/34.868688

21. Strehl, A., Ghosh, J. Cluster Ensembles-A Knowledge Reuse Framework for Combining Multiple Partitions. Journal of Machine Learning Research, 2002, 3(3), 583-617. https://doi.org/10.1162/153244303321897735

22. Sun, J., Xie, Y., Zhang, H., Faloutsos, C. Less is More: Compact Matrix Decomposition for Large Sparse Graphs. Siam International Conference on Data Mining, April 26-28, Minneapolis, Minnesota, USA, 2007, 366-377. https://doi.org/ 10. 1137/ 1.9781611972771.33

23. Tong, H., Papadimitriou, S., Sun, J., Yu, P. S., Faloutsos, C. Colibri: Fast Mining of Large Static and Dynamic Graphs. Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (ACM2008), 2008, 686-694. https://doi.org/10.1145/1401890.1401973

24. Wang, L. J., Rege, M., Dong, M., Ding, Y. S. Low-Rank Kernel Matrix Factorization for Large-Scale Evolutionary Clustering. IEEE Transactions on Knowledge and Data Engineering, 2012, 24(6), 1036-1050. https://doi.org/10.1109/TKDE.2010.258 https://doi.org/10.1109/TKDE.2010.258

25. Yeh, M. Y., Dai, B. R., Chen, M.S. Clustering Over Multiple Evolving Streams by Events and Correlations. IEEE Transactions on Knowledge and Data Engineering, 2007, 19(10), 1349-1362. https://doi.org/10.1109/TKDE.2007.1071

26. Zhu, Y., Shasha, D. Statstream: Statistical Monitoring of Thousands of Data Streams in Real Time. Proceedings of the 28th International Conference on Very Large Data Bases, (VLDB Endowment 2002), Hong Kong SAR, China 20-23 August 2002, 358-369. https://doi.org/ 10.1016/B978-155860869-6/50039-1