Constraint-based discriminative dimension selection for high-dimensional stream clustering

Kitsana Waiyamai [a,1,*], Thanapat Kangkachit [b,2]

[a] Department of Computer Engineering, Kasetsart University, Bangkok, Thailand
[b] College of Innovative Technology and Engineering, Dhurakij Pundit University, Bangkok, Thailand
[1] fengknw@ku.ac.th; [2] thanapat.kan@dpu.ac.th
* corresponding author

ARTICLE INFO

Article history
Selected paper from The 2018 4th International Conference on Science in Information Technology (ICSITech) (Melaka-Malaysia, 30-31 October 2018) (http://icsitech.org/). Peer-reviewed by ICSITech Scientific Committee and Editorial Team of IJAIN journal.

Received July 1, 2018
Revised August 30, 2018
Accepted September 9, 2018
Available Online November 30, 2018

Keywords
Incremental stream clustering
High-dimensional data streams
Dimension selection
Projected clustering
Constraint-based clustering

ABSTRACT

Clustering data streams is one of active research topic in data mining. However, runtime of the existing stream clustering algorithms increases and their performance drop in the face of large number of dimensions. Complexity of the stream clustering methods is increased when perform on data with large number of dimensions. In order to reduce the clustering complexity, one possible solution consists in determining the appropriate subset of cluster dimensions via dimension projection. SED-Stream is an efficient clustering algorithm that supports high dimension data streams. The aim of this paper is to increase performance of SED-Stream in terms of both clustering quality and execution time. In order to improve the clustering process, background or domain expert knowledge are integrated as "constraints" in SEDC-Stream. The new algorithm, SEDC-Stream, supports the evolving characteristics of the dynamic constraints which are activation, fading, outdating and prioritization. SEDC-Stream algorithm is able to reduce cluster splitting time, and place new incoming points to their suitable clusters. Compared to SED-Stream on the three real-world streams datasets, SEDC-Stream is able to generate a better clustering performance in terms of both purity and f-measure.

1. Introduction

Clustering data streams is one of active research topic in data mining. The streams clustering processes data in a single pass and summarizes them in real-time, while using limited resources. Many techniques have been proposed for clustering data streams [1]–[7]. Nevertheless, small number of them have been developed for monitoring and detecting change of the clustering structures [8]–[10]. E-Stream [9] is an evolution-based stream clustering technique that has been developed to detect change of the evolving clustering structures. However, its runtime increases and its performance drops when perform on streams with large number of dimensions.

Complexity of the stream clustering methods is increased when perform on data with large number of dimensions. The use of dimension projection technique [2], [11]–[17] is one possible solution to reduce complexity in dealing high dimensional streams. The concept of "projected clustering" has been introduced in HP-Stream [2]. With its dimension projection mechanism, HPStream is able to determine the specific set of cluster dimensions. Similar to HPStream, SED-Stream [18] is developed to deal with high dimensional data streams. By using the standard-deviation of the attributes, SED-Stream is able to select relevant subset of cluster dimensions. Experimental results over several stream datasets demonstrate that SED-Stream is able to generate higher clustering quality.

One way to improve quality of the clustering result is via the use of domain expert knowledge [19], [20]. Semi-supervised stream clustering is a technique that performs cluster analysis over data streams by using domain expert knowledge as "constraint". Although large number of stream clustering techniques have been proposed, a small number of them have been involved in semi-supervised manner [21]–[23]. A conceptual model for analyzing data streams using constraints has been proposed in Ruiz et al. [24]. The traditional K-means is extended to obtain the Constraint-based K-means algorithm [25]. C-Denstream [26] is the extended version of Denstream, by utilizing constraints during the clustering process. An instance-level Must-link constraint is defined as a pair of instances (x,y) that must be members to the same cluster [1]. Must-Link constraints are integrated as background knowledge in E-stream [9] to obtain a semi-supervised stream clustering technique named CE-Steam [27]. By adapting its clustering structure over the continuous flow of data points, CE-Stream continuously updates its constraints based on their hit-rates.

The main goal of this paper is to improve performance of the existing evolution-based stream clustering algorithms such as E-Stream [9], CE-Stream [27] and SED-Stream [18]. To deal with high dimensional data streams, the idea is to combine dimension selection technique with the use of domain expert knowledge as constraints. The new algorithm is named SEDC-Stream (Constraint-based discriminative dimension selection for clustering data streams with large number of dimensions). For dimension selection, attributes of each cluster are projected to its discriminative dimension attributes. During the progression of data streams, SEDC-Stream locates all the active clusters while determining their discriminative attributes. Two types of instance-level constraints, Must-link and Cannot-link, are introduced and integrated in a semi-supervised manner. SEDC-Stream does support the evolving characteristics of the dynamic constraints which are constraint activation, fading, outdating and prioritization. SEDC-Stream is able to reduce an excessive splitting during the clustering process, and place new incoming points to their suitable clusters, during the data streams progression. Compared to SED-Stream on three real-world streams datasets [28], [29], the results reveal that SEDC-Stream has improved both the clustering output quality and the execution time.

To summarize, the new SEDC-Stream algorithm provides the following mechanisms to support high dimensional data streams are discriminative dimension selection to support the evolving clustering structure; and Integration of activated, faded and obsolete instance-level constraints during the progression of data streams. SEDC-Stream outperforms the existing evolution-based stream clustering algorithms such as E-Stream, CE-Stream and SED-Stream in terms of clustering quality.

The remaining of this paper is structured as follows. Section 2 divided into two sections. In section 2.1, definitions and concepts related to the stream clustering, discriminative dimension selection, and stream constraints are given, while in section 2.2, the SEDC-Stream is presented in detail. In section 3, the performance of SEDC-Stream and SED-Stream are compared over three real-world stream datasets. In section 4, the conclusion and the future works are explained.

2. Method

2.1. Basic Concepts

In the following, basic concepts and techniques related to the stream clustering, discriminative dimension selection, and stream constraints are given.

2.1.1. Cluster Representation

Assume that data streams consist of a set of data points $X_1 \ldots X_k$ arriving at time stamps $T_1 \ldots T_k$. Each data point X_i contains d dimensions, denoted as $X_i = (x^1{}_i \ldots x^d{}_i)$. During the progression of data streams, there exist a large number of incoming data points that cannot be stored into the limited-size memory. Instead, cluster representation is used. In Aggarwal et. al. [2], *a fading cluster structure (FCS)* was introduced. Later, Udommanetanakit et. al. [9] proposed *FCH* that extended *FCS* by adding α-bin histogram to detect change of the clustering structure. In Chairukwattana et. al. [30], the notion of dimension projection was added into FCH. Finally, FCH is defined as $FCH = (FC1(t), FC2(t), W(t), BS(t), H(t))$. The description of FCH can be described as follows.

Let N be the total number of data points of such cluster, T_i be the time when data point x_i is retrieved, and t be the current time. The fading weight of data point x_i is defined as $f(t - T_i)$ where $f(t) = 2 - \lambda t$ and λ is the user-defined decay rate.

$\boldsymbol{FC1(t)}$ is a vector of weighted sum of each dimension at time t. The j^{th} dimension is

$$FC1^j(t) = \sum_{i=1}^{N} f(t - T_i) \cdot \left(X_j^i\right) \quad (1)$$

$\boldsymbol{FC2(t)}$ is a vector of weighted sum of square of each dimension at time t. The j^{th} dimension is

$$FC2^j(t) = \sum_{i=1}^{N} f(t - T_i) \cdot \left(X_j^i\right)^2 \quad (2)$$

$\boldsymbol{W(t)}$ is a sum of all weights of data points in the cluster at time t, i.e.,

$$W(t) = \sum_{i=1}^{N} f(t - T_i) \quad (3)$$

$\boldsymbol{BS(t)}$ is a bit vector of projected dimensions at time t. For the j^{th} dimension is

$$BS(t) = \begin{cases} 1, & if\ dimension\ j\ is\ within\ a\ set\ of\ relevant\ cluster\ dimensions \\ 0, & otherwise. \end{cases} \quad (4)$$

Note that the number of projected dimensions in each cluster may be different.

$\boldsymbol{H(t)}$ is a $\alpha - bin$ histogram with equal intervals of data values. For the j^{th} dimension at time t, the m^{th} bin histogram is

$$H_m^j(t) = \sum_{i=1}^{N} f(t - T_i) \cdot \left(X_j^i\right) \cdot \left(y_{1,m}^i\right) \quad (5)$$

where

$$y_m^i = \begin{cases} 1, & \text{if } m \cdot r + \min(x^j) \le x_i^j \le (m+1) \cdot r + \min(x^j);\ r = \frac{\max(x^j) - \min(x^j)}{\alpha} \\ 0, & \text{otherwise.} \end{cases} \quad (6)$$

Note that the value of bin width (r) may be different in each dimension.

Each cluster can be categorized as active or inactive cluster when its weight is greater than or lower than the user-specified threshold *active_cluster_weigth* respectively. The active clusters are capable of merging with their nearest incoming data points resulted in their self-evolution clustering structure. Contrastingly, to become an active cluster, such inactive clusters must be merged together with other inactive or active clusters.

2.1.2. Distance Functions

Here, the distance functions are modified in order to deal with the different subset of projected dimensions of each cluster. We recall the notion of $BS(t)$ (in section 2.1.1) which is a bit vector represented the projected dimensions of such cluster at timestamp t. The distances functions can be defined as follows.

Cluster-Point distance $dist(C, X_i)$ is measured from a center of the active cluster C to a data point X_i. For each dimension j, the distance is normalized by the radius (standard deviation) of the cluster radius j_c. The $dist(C, X_i)$ function at timestamp t is

$$dist(C, X_i) = \frac{1}{n} \cdot \sum_{j \in BS(t)} \left| \frac{center_C^j - x_i^j}{radius_C^j} \right| \quad (7)$$

where n is the number of projected dimensions of cluster C represented by bit vector $BS(t)$. Thus, an incoming data point is then merged into its closet active cluster i.e. having minimum Cluster-Point distance.

Cluster-Cluster distance $dist(C_a, C_b)$ is measured between two cluster centers (C_a and C_b). The $dist(C_a, C_b)$ function at timestamp t is

$$dist(C_a, C_b) = \frac{1}{n} \cdot \sum_{j \in BS(t)} \left| center_{C_a}^{j} - center_{C_b}^{j} \right| \quad (8)$$

where n is the total number of projected dimensions represented by bit vector $BS(t)$. Note that $BS(t)$ corresponds to union set of the projected dimensions of these two clusters. Therefore, a pair of clusters can be merged by determining its cluster-cluster distance.

2.1.3. Instance-Level Stream Constraint

In the following, definitions related to instance-level stream constraint are given. More detailed explanation of these concepts can be found in [27]. We also refer to the notation of data streams, data point and fading weight as introduced in section 2.1.1.

Must-link constraint denoted as ML is a pair of data points (ML_x, ML_y) that must be assigned to the same cluster.

Cannot-link constraint denoted as CL is a pair of data points (CL_x, CL_y) that must not be assigned to the same cluster.

Active constraint denoted as AL is a must-link or cannot-link constraint where all weights of its data points (AL_x, AL_y) are greater than user-specified threshold W.

Constraint weight is the minimum weight of the active constraint data points AL_x and AL_y, and can be defined as

$$CT_{wight} = \min(weight(AL_x), weight(AL_y)) \quad (9)$$

Constraint fading function is used to gradually reduce weight of active constraints over time. Obviously, lifetime of constraints is much longer than lifetime of data points. Thus, the constraint fading function can be defined as

$$fad(t) = 2^{-\gamma_r^t} \quad (10)$$

where r is the time interval.

Constraint Hit-Rate specifies the number of times when a constraint is utilized within the stream clustering process. For effective data processing, constraints are sorted by their hit-rate in descending order.

Check Active and Update Constraints is a function to activate constraints before being used. This can be done by matching such constraints with the satisfied incoming data points. However, constraints are faded and obsolete over time.

Prioritize Constraints is a function to sort constraints based on their hit-rate resulted in effective computation.

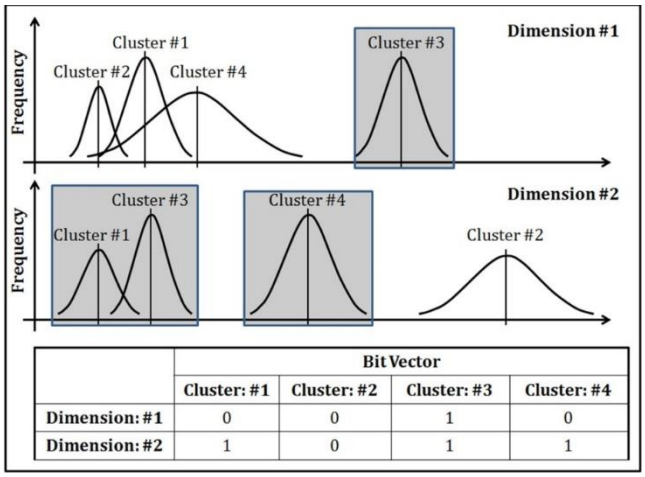
2.2. SEDC-Stream Algorithm

2.2.1. Discriminative Dimension Selection

When clustering over high-dimensional data streams, the problem of data sparsity frequently happens. One possible solution to deal with this problem is to apply the projected clustering technique which has been proposed in [8]. For each cluster, projected clustering technique determines specific subset of relevant dimensions. As result, data points are less spread since their intra-dissimilarity within

a cluster is minimized. Unfortunately, the distance between clusters (inter-dissimilarity) may become closer. The reason is that, in some specific dimensions, there exist the overlapped ranges which can be covered by radii of several other clusters, as shown in Fig. 1.

Another approach named "discriminative dimension selection" for projected clustering is proposed [18]. Unlike the traditional projected clustering, the discriminative dimension selection consists in identifying all the discriminative dimensions. A discriminative dimension is highly relevant to its cluster and very distinguished from the other clusters. To determine all the discriminative dimensions, two steps performed which are dimension uniqueness filtered-out and dimension radii ranking. Dimension uniqueness is the number of clusters for which their radii are overlapped. For a given dimension, if its dimension uniqueness is more than the overlapped ratio, then the dimensions (and its overlapped dimensions) are filtered out, and all the remaining dimensions are ranked according to their radii. Discriminative dimensions are selected by choosing the remaining dimensions at the top $|FCH| \cdot l$ ranks where $|FCH|$ is the number of clusters and l is the average number of selected dimensions for each cluster.



	Bit Vector			
	Cluster: #1	Cluster: #2	Cluster: #3	Cluster: #4
Dimension: #1	0	0	1	0
Dimension: #2	1	0	1	1

Fig. 1. Example of discriminative dimension selection (overlapped threshold= 0.7) and its associated bit vector [18]

Let l be the average number of selected dimensions for each cluster, and v be the overlapped threshold. At first step, l and v are set to 1 and 0.7, respectively. The discriminative dimension selection mechanism and its associated bit vector are explained in Fig. 1. Suppose that, at time stamps t, the clustering output is composed of 4 clusters using 2 dimensions. First, for each cluster, dimension radius is computed based on its FCH. The radius of one cluster may be overlapped by the other clusters' radius i.e. dimension #1 of cluster #1, #2 and #3. If the number of overlapped clusters ratio is more than v (0.7), then all the overlapped dimensions are filtered out. After that, the remaining dimensions are ranked (in ascending order) based on their radii. Finally, the dimensions that are at the top $|FCH| \cdot l$ ranks will be selected as discriminative dimensions i.e. dimension #1 and #2 of cluster #3 and dimensions #2 of cluster #4 (in grey color). Notice that, the number of discriminative selected dimensions may be different in each cluster.

2.2.2. Use of Must-Link constraints to guide the clustering process

Must-link constraints are utilized to guide the cluster splitting process. An active cluster can be split into two clusters in case of existing of two separate density areas in such cluster dimension. However, this might mislead cluster splitting as shown in Fig. 2. The first and second dense area (represented by blue circle) might be split since there is a largest gap between those two areas. Unfortunately, these two dense areas belong to the same class. To prevent the improper cluster splitting, the must-link constraints must be applied.

Regarding to cluster splitting process, FCH (Fading Cluster Histogram) is used to determine the best split point (i.e. i^{th} bin of FCH) as shown in Fig. 2. Here, must-link constraints are taken into

account. The cluster splitting is ignored if there exists at least one active constraint $ML(ML_x, ML_y)$ such that both ML_x and ML_y separately appear in the 1^{st} to $i-1$ bin of FCH and the i^{th} to the last bin of FCH.

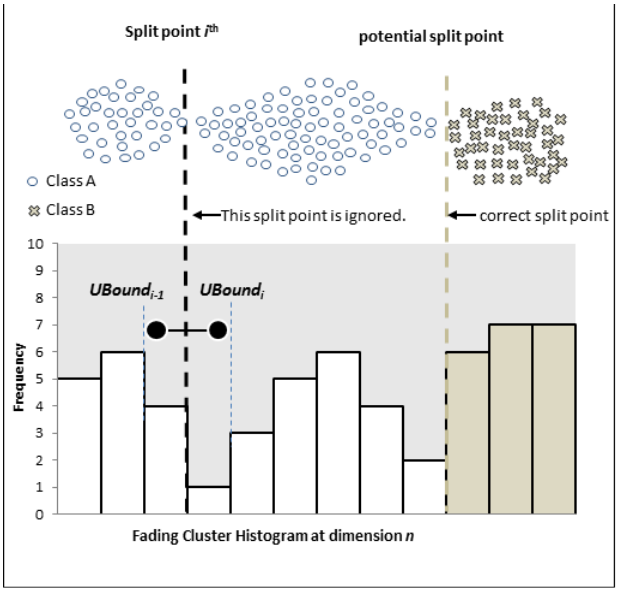


Fig. 2. FCH on Cluster Splitting

2.2.3. Use of Cannot-Link constraints to guide the clustering process

Cannot-link constraints are utilized to guide cluster assignment of a new data point. Intuitively, a new data point is to be assigned to its closest cluster. However, in some circumstances, the new data point might be in conflict with the members of the closest cluster. With only few numbers of inaccurate-assigned new data points, the resulting clustering structure might be changed. To overcome this problem, cannot- link constraints $CL(CL_x, CL_y)$ can be used to guide the clustering process to find suitable cluster for the new incoming data point by considering both smallest distance and without violation of the cannot-link constraint(s).

2.2.4. SEDC-Stream algorithm

This section describes SEDC-Stream main algorithm. SEDC-Stream combines dimension selection technique with the use of background knowledge as constraints as introduced in section 2.2.1 and section 2.2.2 respectively. To guide the clustering process, background knowledge is used as constraints in two scenarios based the types of constraints. First, must-link constraints are used to prevent cluster-splitting. In some circumstances, SEDC-Stream detects the cluster-splitting behavior which results in a separation of data points (ML_x, ML_y) of the activated must-link constraint ML. It implies that all the data points within the cluster are still related, and the splitting operation is not necessary. Second, cannot-link constraints are used in placing a new incoming data point into its suitable cluster. A new incoming data point is normally added into its closest cluster. However, if cannot-link constraint is activated, then the data point will be added to the cluster with smallest distance and without conflict to the cannot-link constraint. For dimension selection, any active cluster is projected to its discriminative dimensions that are highly relevant to its cluster and very distinguished from the other clusters.

The main algorithm of SEDC-Stream is given in Fig. 3. In line 1, a new data point is retrieved. In line 2, constraints are activated and are updated their weights if they satisfied with the new data point. In line 3, all clusters are faded. The clusters with weight less than user-specified threshold (χ) are deleted. In line 4, to speed up the execution time, constraints are sorted according to their weight. In line 5, any cluster can be split when behavior inside the cluster is obviously separated except there is a signal from must-link constraint (MLS). In line 6, the overlapping-active clusters are merged. In line 7, when the number of cluster count exceeds the limit, it begins to merge the closet pair of clusters until the number

of cluster count reaches the limit. In line 8, constraints are faded and deleted if their weight is less than user-specified threshold (ƛ). In line 9, it checks all clusters whether their statuses are active. In line 10-12, all active clusters are re-projected to their new discriminative dimensions if members in the set of active clusters are changed. In line 13-18, the incoming data point is included into its closet cluster that does not satisfy any cannot-link constraints, if its distance is within *radius_factor* (as an input parameter). Otherwise, a new isolated data point is created from this incoming data point. Then the algorithm returns to the top and waits for a new data point.

Algorithm *SEDC-Stream*

Input X_t : data point X at time t MLS : activated constraint-set
ƛ : decay rate W : weight Threshold
$IMLS$: input constraint-set

1: Retrieve new data X_t
2: ***CheckActive&UpdateConstraints (IMLS,MLS,W)***
3: FadingAllCluster(ƛ)
4: ***PrioritizeConstraints(MLS)***
5: ***ClusterSplitting(MLS)***
6: MergeOverlapCluster()
7: LimitMaximumCluster()
8: ***FadingConstraints(ƛ)***
9: CheckActiveCluster()
10: **IF** (found new active cluster(s) or active cluster(s) turn into inactive cluster(s))
11: DiscriminativeDimensionSelection()
13: (minDistance, index) ← FindClosetCluster()
14: **IF** minDistance < radius_factor
15: ClusterAssignment(X_t, $FCH_{[index]}$) *// add X_t into its suitable cluster*
16: **ELSE**
17: Create new FCH from X_t
18: **END**
19: Waiting for new data

Fig. 3. Main SEDC-Stream algorithm

The modified and added functions from E-stream are given in Fig. 4 and briefly described as follows.

FadingAll: All clusters are faded. Then, clusters with small weight (i.e. less than ε) are deleted.

CheckSplit: In the discriminative dimensions of any active cluster, if there is a splitting point and no established must-link constraints after splitting, then the cluster splitting process can be done. Otherwise, SEDC-Stream ignores splitting.

MergeOverlapCluster: The pairs of active overlapped clusters can be merged, unless they are the results of cluster splitting process. Notice that, only active clusters that include the new incoming data points are considered due to their self-revolution change.

LimitMaximumCluster: If the total number of clusters reaches the *maximum_cluster* threshold, the closest pair of clusters is merged until the number of remaining clusters does not exceed the *maximum_cluster*. If there exists a new active cluster, **DiscriminateProjectDimension** method is performed after receiving a new data point to extract its discriminate dimensions as described in section 2.2.1.

FindClosestCluster: An incoming data point is assigned to its closet active cluster. This is done by determining the minimum cluster-point distance (*CPDistance*). Note that the distance is calculated only the discriminative dimensions of such active cluster.

Notice that, after merging process in the **MergeOvelapCluster** and **LimitMaximumCluster** procedures, there are 3 cases of discriminative projected dimensions (*BS*). First, if both clusters are active clusters, the result is bitwise OR operation on both *BS*. Second, if it has only one active cluster, the result is the *BS* of active cluster. Otherwise the result is all dimensions (all bits of *BS* is set to 1).

Procedure ***DiscriminativeDimensionSelection***
1: Compute the $|FCH_{active}|*d$ radius along d dimensions
2: Find the number of overlap $|FCH_{active}|*d$
3: If the number of overlap is greater than threshold, neglect it.
4: Pick $|FCH_{active}|*l$ remaining dimensions with the least radius.
5: Set bit vector for each cluster reflecting its projected dimensions (from line 4).

Procedure ***CheckSplit***
1: for i ←1 to $|FCHActive|$
2: for i ←1 to d
3: if($BS(FCHactive_i|^j) == 1$) //discriminative dimension
4: if($FCHactive_i^j$ has split point(s))
5: if there is no separation of activated must-link constraint(s) after splitting
6: split FCH_{active} using j^{th} dimension
7: $S \leftarrow S \cup \{ (i, |FCH|) \}$

Procedure ***MergeOverlapCluster***
1: for $i \leftarrow 1$ to $|FCHActive|$
2: for $j \leftarrow i+1$ to $|FCHActive|$
3: $overlap[i,j] \leftarrow CCDistance(FCHactive_i, FCHactive_j)$
4: $m \leftarrow merge\_threshold$
5: if($overlap[i,j]$) > m*($FCHactive_i.sd + FCHactive_i.sd$)
6: if(i,j) not in S
7: merge($FCHactive_i, FCHactive_j$)

Procedure ***FindClosetCluster***
1: for $i \leftarrow 1$ to $|FCHActive|$
2: $dist[i] \leftarrow$ CPDistance($FCHactive_i, X_i$)
3: ($mindistance, i$) ← min($dist[i]$)
4: return ($mindistance, i$)

Procedure ***FadingAll***
1: for i ←1 to $|FCH|$
2: Fading FCH_i
3: If ($FCH_i.w < \varepsilon$) // weight of cluster less than ε
4: delete FCH_i

Procedure ***LimitMaximumCluster***
1: while $|FCHactive| > maximum\_cluster$
2: for $i \leftarrow 1$ to $|FCH|$
3: for $j \leftarrow i+1$ to $|FCH|$
4: $dist[i,j] \leftarrow CCDistance(FCH_i, FCH_j)$
5: ($first, second$) ← argmin$_{(i,j)}$($dist[i,j]$)
6: merge($FCHfirst, FCHsecond$)

Procedure ***CCDistance (FCH, FCH)***
1: Create BS_{temp} and set all bit to 1
2: if (both FCH_i and FCH_j are active clusters)
3: $BS_{temp} = BS(FCH_i) \mid BS(FCH_j)$
4: else if (FCH_i or FCH_j is active cluster)
5: Set BS_{temp} as the same as BS of the active cluster
6: for k ←1 to d
7: if ($BS_{temp}^k == 1$)
8: $distance$ = abs(center(FCH_i)^k - center(FCH_j)^k)
9: d' = the number of bits in BS_{temp} with value of 1
10: $distance = distance / d'$
11: return $distance$

Procedure ***CPDistance(FCH, X)***
1: for k ←1 to d
2: if($BS(FCH_i|^k) == 1$) //discriminative dimension
3: $distance$ = abs(center(FCH_i)^k - center(FCH_j)^k)
4: d' = the number of bits in BS_{temp} with value of 1
5: $distance = distance / d'$
6: return $distance$

Fig. 4. Details of each procedure in SEDC-Stream.

3. Results and Discussion

In this section, clustering quality of SEDC-Stream will be demonstrated. The experiments are performed on three well-known streams datasets: Forest Cover Type dataset [28], KDD-99 dataset [28], and Electricity dataset [29]. For comparison, SED-Stream [18] and SEDC-Stream have been implemented using C++. All the experiments are conducted on a 2.6 GHz Intel® Core i5 with 8GB memory. All the parameter settings of SEDC-Stream are as the following: stream speed, horizon and decay rate are set to 500, 2 and 0.1 which are similar to SED-Stream.

3.1. Clustering Performance Comparision

The first dataset is Forest Cover Type dataset which contains 581,012 instances with 10 numerical attributes of 7 classes. First 300,000 instances are used to evaluate the performance of both clustering algorithms and the data is split into smaller chunk of size 25,000 instances for evaluation. Fig. 5 shows that SEDC-Stream achieve better f-measure almost all the time of clustering, and gains comparable purity as SED-Stream.

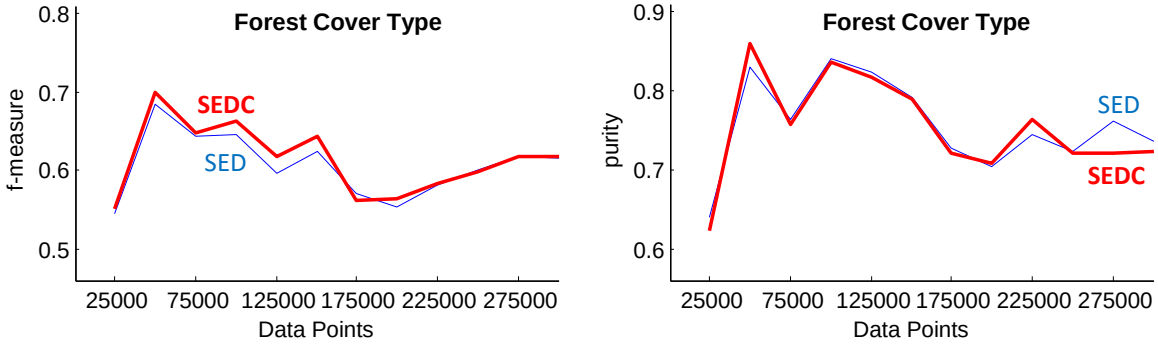


Fig. 5. F-measure and purity of SEDC-Stream on Forest Cover Type dataset

The second dataset is KDD-99 dataset, which contains 250,000 instances with 43 attributes of 23 classes. The results in Fig. 6 show that SEDC-Stream outperforms SED-stream in term of f-measure with equivalent purity.

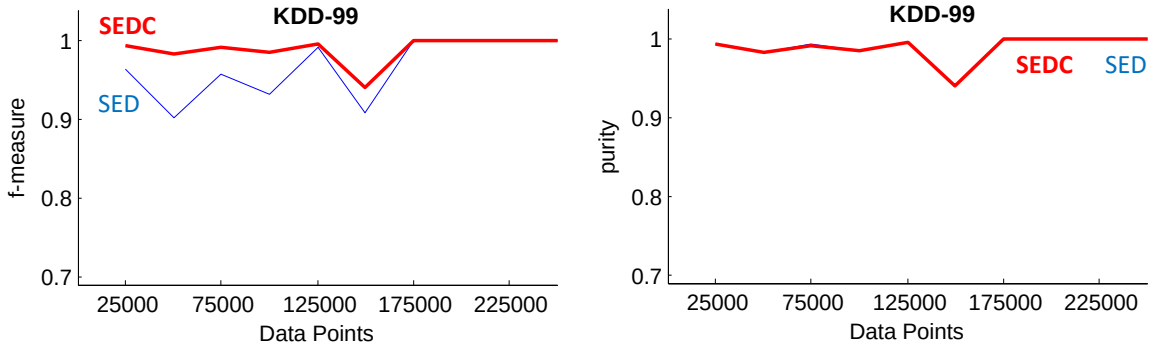


Fig. 6. F-measure and purity of SEDC-Stream on KDD-99 dataset

The last dataset is Electricity dataset, which contains 45,312 instances with 8 attributes of 2 classes. As shown in Fig. 7, SEDC-Stream can achieve much higher f-measure, with lower purity compared to SED-Stream.

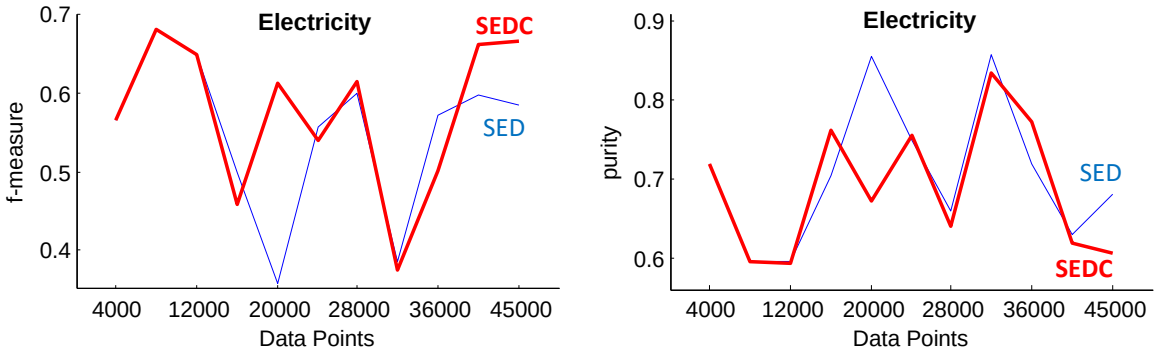


Fig. 7. F-measure and purity of SEDC-Stream on Electricity dataset

3.2. Performance on Integrating Instance-level Constraints

To evaluate performance of SEDC-Stream on constraint selection, two types of constraints are used which are must-link constraints and cannot-link constraints. As explained in the previous section, both types of constraints are the constraints between a pair of instances. Must-link constraints (ML) indicate

that the two instances should be in the same cluster, and cannot-link constraints (CL) indicates that the two instances should not be grouped into the same cluster.

Constraints are very expensive; therefore, SEDC-Stream is designed to use only a small number of constraints (less than 0.01%). For example, with Forest Cover Type dataset contains 300,000 instances; therefore 300,000 x 299,999/2 is about 45 million pairs of instances. For example, with Forest Cover Type dataset containing 300,000 instances, about 45 million pairs of instances (300,000 x 299,999/2) are generated. Only 0.0004% (200 constraints) of them is used in SEDC-Stream algorithm. Notice also that, even with very small number of constraints, those constraints are able to help improving the quality of the output clustering, as shown in the previous section.

Because SEDC-Stream used both must-link and cannot-link constraints, in this section, both types of constraints are evaluated separately. The results in Fig. 8 show that using both types of constraints can achieve higher f-measure. Note that the original SED-stream with or without constraints obtains very similar level of purity, so those results are omitted. Both types of constraints achieve indifferent purity.

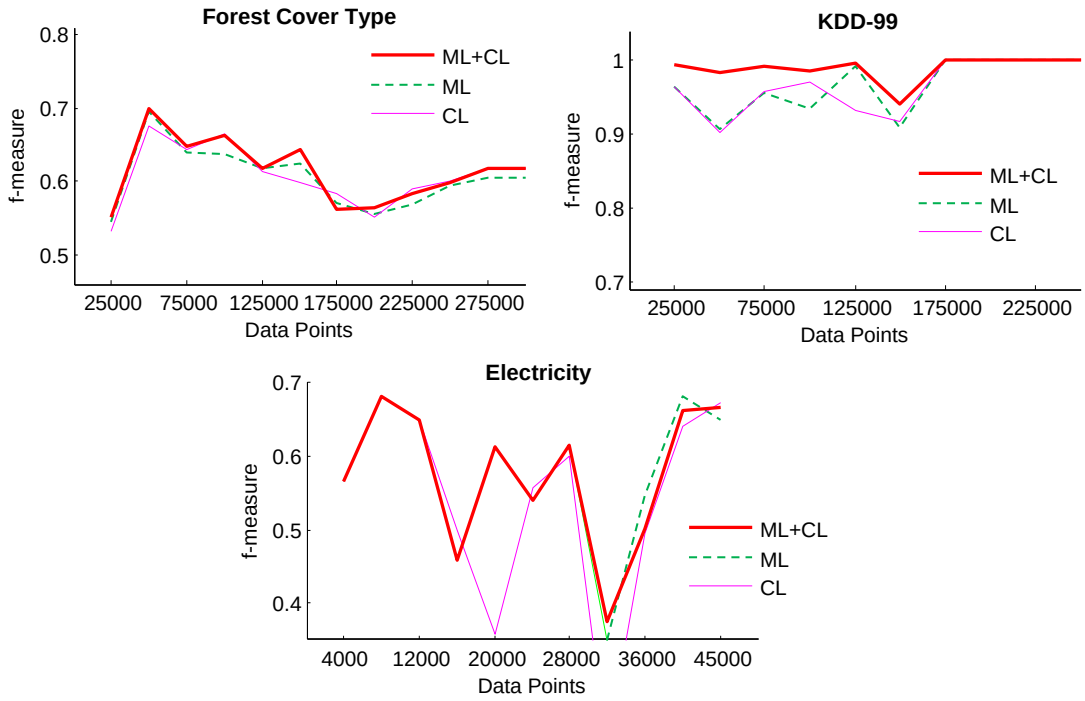


Fig. 8. Cluster quality when only must-link or cannot-link or both constraints are allowed

3.3. Time Complexity

Compared to SED-Stream, SEDC-Stream require additional time to evaluate constraint conditions and to use those constraints to maintain the output clustering. Thus, SEDC-Stream uses more time to perform the clustering than SED-Stream in almost datasets as shown in Fig. 9. With the high-dimensional datasets, it is clearly seen that the execution time of SEDC-Stream is significantly higher than SED-stream (i.e. Forest Cover Type and Electricity datasets). However, SEDC-Stream is faster than SED-Stream on KDD-99 dataset. This can be explained by the following observations. First, only a few attributes are used in the clustering process, and only 13 from 30 pairs of constraints meet the constraint conditions. Thus, a very few additional time is used in order to manage the constraints. Second, unnecessary clustering operations are dramatically decreased. This is due to less numbers of output clusters which are resulted from clustering-splitting prevention of must-link constraints.

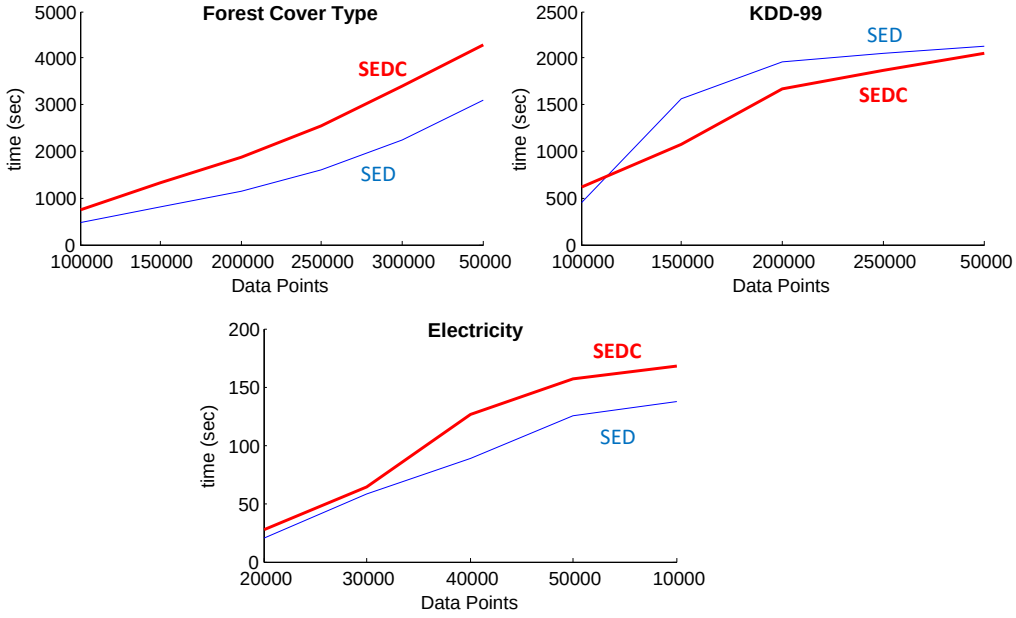


Fig. 9. F-measure and purity of SEDC-Stream on electricity dataset

4. Conclusion

This paper proposed SEDC-Stream, an evolution-based clustering algorithm for high dimensional data streams. Two solutions have proposed to alleviate the complexity of high dimensional data streams processing: dimension selection & use of constraints. Dimension selection is able to find evolving clusters with subgroups of discriminative dimensions during the progression of data streams. During the stream progression, instance-level constraints which are must-link and cannot-link constraints are integrated by means of constraints prioritization, activation, fading and outdating in order to improve the clustering output.

Several research directions are possible to improve the proposed algorithm. First, the use of background or domain expert knowledge in a semi-supervised clustering manner. Indeed, the use of constraints may not be appropriate with respect to the dynamic nature of data streams. Exploiting background knowledge as single labeled data points (not pair of points) is more appropriate for data streams. Labeled data points can be immediately utilized for determining the class of clusters, and effectively identifying the most appropriate clustering structure evolution operations. Second, an alternate and efficient version of SEDC-Stream for generating arbitrary shape clusters is needed. Indeed, trying to minimize the squared error, SEDC-Stream is only able to generate spherical shape clusters. Clusters with arbitrary shapes are observed in many application areas of science. Online density-based and hierarchical clustering can be combined to obtain an efficient evolution-based and shape-based clustering algorithm for high dimensional data streams.

References

[1] C. C. Aggarwal, P. S. Yu, J. Han, and J. Wang, "A Framework for Clustering Evolving Data Streams," 2003, pp. 81–92, doi: https://doi.org/10.1016/B978-012722442-8/50016-1.

[2] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A Framework for Projected Clustering of High Dimensional Data Streams," 2004, pp. 852–863, doi: https://doi.org/10.1016/B978-012088469-8.50075-9.

[3] F. Cao, M. Estert, W. Qian, and A. Zhou, "Density-Based Clustering over an Evolving Data Stream with Noise," 2006, pp. 328–339, doi: https://doi.org/10.1137/1.9781611972764.29.

[4] J. Gao, J. Li, Z. Zhang, and P.-N. Tan, "An Incremental Data Stream Clustering Algorithm Based on Dense Units Detection," 2005, pp. 420–425, doi: https://doi.org/10.1007/11430919_49.

[5] S. Mansalis, E. Ntoutsi, N. Pelekis, and Y. Theodoridis, "An evaluation of data stream clustering algorithms," *Stat. Anal. Data Min. ASA Data Sci. J.*, vol. 11, no. 4, pp. 167–187, Aug. 2018, doi: https://doi.org/10.1002/sam.11380.

[6] M. Ghesmoune, M. Lebbah, and H. Azzag, "State-of-the-art on clustering data streams," *Big Data Anal.*, vol. 1, no. 1, p. 13, 2016, available at : https://bdataanalytics.biomedcentral.com/articles/10.1186/s41044-016-0011-3.

[7] Y. Chen and L. Tu, "Density-based clustering for real-time stream data," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2007, pp. 133–142, doi: https://doi.org/10.1145/1281192.1281210.

[8] K. Chen and L. Liu, "HE-Tree: a framework for detecting changes in clustering structure for categorical data streams," *VLDB J.*, vol. 18, no. 6, pp. 1241–1260, Dec. 2009, doi: https://doi.org/10.1007/s00778-009-0134-5.

[9] K. Udommanetanakit, T. Rakthanmanon, and K. Waiyamai, "E-Stream: Evolution-Based Technique for Stream Clustering," 2007, pp. 605–615, doi: https://doi.org/10.1007/978-3-540-73871-8_58.

[10] S. Gong, Y. Zhang, and G. Yu, "Clustering stream data by exploring the evolution of density mountain," *Proc. VLDB Endow.*, vol. 11, no. 4, pp. 393–405, 2017, available at : https://dl.acm.org/citation.cfm?id=3164136.

[11] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park, "Fast algorithms for projected clustering," *ACM SIGMOD Rec.*, vol. 28, no. 2, pp. 61–72, Jun. 1999, doi: https://doi.org/10.1145/304181.304188.

[12] I. Ntoutsi, A. Zimek, T. Palpanas, P. Kröger, and H.-P. Kriegel, "Density-based Projected Clustering over High Dimensional Data Streams," 2012, pp. 987–998, doi: https://doi.org/10.1137/1.9781611972825.85.

[13] S. Laohakiat, S. Phimoltares, and C. Lursinsap, "A clustering algorithm for stream data with LDA-based unsupervised localized dimension reduction," *Inf. Sci. (Ny).*, vol. 381, pp. 104–123, Mar. 2017, doi: https://doi.org/10.1016/j.ins.2016.11.018.

[14] O. Makul and M. Ekinci, "A graph form data stream clustering approach based on dimension reduction," in *2017 25th Signal Processing and Communications Applications Conference (SIU)*, 2017, pp. 1–4, doi: https://doi.org/10.1109/SIU.2017.7960504.

[15] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," *ACM SIGMOD Rec.*, vol. 27, no. 2, pp. 94–105, Jun. 1998, doi: https://doi.org/10.1145/276305.276314.

[16] W. Meesuksabai, T. Kangkachit, and K. Waiyamai, "Evolution-Based Clustering Technique for Data Streams with Uncertainty," *Kasetsart J. (Nat. Sci.)*, vol. 46, pp. 638–652, 2012, available at : https://pdfs.semanticscholar.org/664b/c9c63f8d88590da15ac33d3f791e1ad9626c.pdf.

[17] I. Ahmed, I. Ahmed, and W. Shahzad, "A Novel High Dimensional and High Speed Data Streams Algorithm: HSDStream," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 9, 2016, doi: https://doi.org/10.14569/IJACSA.2016.070952.

[18] K. Waiyamai, T. Kangkachit, T. Rakthanmanon, and R. Chairukwattana, "SED-Stream: discriminative dimension selection for evolution-based clustering of high dimensional data streams," *Int. J. Intell. Syst. Technol. Appl.*, vol. 13, no. 3, p. 187, 2014, doi: https://doi.org/10.1504/IJISTA.2014.065174.

[19] S. Basu, A. Banerjee, and R. J. Mooney, "Active semi-supervision for pairwise constrained clustering," in *Proceedings of the 2004 SIAM international conference on data mining*, 2004, pp. 333–344, available at : https://epubs.siam.org/doi/abs/10.1137/1.9781611972740.31.

[20] P. S. Bradley, K. P. Bennett, and A. Demiriz, "Constrained k-means clustering," *Microsoft Res. Redmond*, pp. 1–8, 2000, available at :http://machinelearning102.pbworks.com/f/ConstrainedKMeanstr-2000-65.pdf.

[21] K. Treechalong, T. Rakthanmanon, and K. Waiyamai, "Semi-Supervised Stream Clustering Using Labeled Data Points," 2015, pp. 281–295, doi: https://doi.org/10.1007/978-3-319-21024-7_19.

[22] V. Antoine, N. Labroche, and V.-V. Vu, "Evidential seed-based semi-supervised clustering," in *2014 Joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS) and 15th International*

Symposium on Advanced Intelligent Systems (ISIS), 2014, pp. 706–711, doi: https://doi.org/10.1109/SCIS-ISIS.2014.7044676.

[23] C. Ruiz, M. Spiliopoulou, and E. Menasalvas, "Density-based semi-supervised clustering," *Data Min. Knowl. Discov.*, vol. 21, no. 3, pp. 345–370, 2010, doi: https://doi.org/10.1007/s10618-009-0157-y.

[24] C. R. Moreno, M. Spiliopoulou, and E. Menasalvas, "User constraints over data streams," *Knowl. Discov. from Data Streams*, p. 117, 2006, available at : http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.61.7653&rep=rep1&type=pdf#page=121.

[25] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl, and others, "Constrained k-means clustering with background knowledge," in *ICML*, 2001, vol. 1, pp. 577–584, available at : https://web.cse.msu.edu/~cse802/notes/ConstrainedKmeans.pdf.

[26] C. Ruiz, E. Menasalvas, and M. Spiliopoulou, "C-DenStream: Using Domain Knowledge on a Data Stream," 2009, pp. 287–301, doi: https://doi.org/10.1007/978-3-642-04747-3_23.

[27] T. Sirampuj, T. Kangkachit, and K. Waiyamai, "CE-Stream : Evaluation-based technique for stream clustering with constraints," in *The 2013 10th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 2013, pp. 217–222, doi: https://doi.org/10.1109/JCSSE.2013.6567348.

[28] K. Bache and M. Lichman, "UCI Machine Learning Repository, University of California, School of Information and Computer Science," *Irvine, CA*, 2013, available at : http://archive.ics.uci.edu/ml.

[29] M. Harries and N. S. Wales, "Splice-2 comparative evaluation: Electricity pricing," Citeseer, Sydnesy, 1999, available at : http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.43.9013.

[30] R. Chairukwattana, T. Kangkachit, T. Rakthanmanon, and K. Waiyamai, "Efficient evolution-based clustering of high dimensional data streams with dimension projection," in *2013 International Computer Science and Engineering Conference (ICSEC)*, 2013, pp. 185–190, doi: https://doi.org/10.1109/ICSEC.2013.6694776.