Overview of different methods for Clustering Evolving Stream Data

Jagan Shanmugam
University of Stuttgart
Stuttgart, Germany
st159117@stud.uni-stuttgart.de

ABSTRACT

Clustering data which is streaming continuously is treated differently from clustering data which is available as a batch. In stream data clustering, we should not only update the clusters in real-time before the next stream of data is available but also track the changes in the clusters to gain insights about the evolution of the clusters in real-time. Density based clustering techniques allows us to work without configuring number of clusters before the arrival of data points and can also identify clusters of arbitrary shapes. Density based clustering algorithm assumes that the cluster centers are surrounded by points with low density and also dense areas are separated from sparse areas. Most of the algorithms work in two phases. In the online phase, since the stream data are unlimited, summary structures are used to summarize the data to reduce the processing overhead. These summary structures are used later in the offline phase to cluster the dataset. Also, there exist methods which take in clustered results as input over a period of time and produce the evolution of the clusters using the temporal relationship between the clusters. The contribution of this work is that it provides a overview of different stream clustering algorithms and also discusses about two methods to track the evolution of clusters.

KEYWORDS

Stream data clustering; Cluster evolution tracking; Density-based clustering

1 INTRODUCTION

In the modern age where data is generated continuously from different sources, it has become increasingly relevant to understand the evolving nature of the data. Primarily, clustering categorizes the data into similar and dissimilar groups based on a similarity measure. Clustering stream data helps in gaining appropriate insights from the data which lead to efficient and timely business decisions. In contrast to the batch mode clustering where the complete data is stored and analyzed, stream data clustering possesses different set of challenges. The two main challenges [4] are: 1. Stream data arrives in high speed and the methods should incrementally update the clustering results. 2. As clusters are evolving over time, cluster evolution should be captured to gain insights about the clusters. The evolution of data as it is streaming is essential in understanding the behavior of trends in data at a particular instant in time. Although traditional clustering methods like K-means or DBSCAN focuses on batch clustering, there are many works on adapting the batch mode clustering algorithms to stream data clustering. These adapted algorithms are single pass that looks at the data only once or in batches and produce clustering results. Nowadays, as data streams are generated from different sources, the dimension of the input to the algorithms is very high which necessitates the algorithm to handle high dimensional data streams. As stream data is considered as unbounded in time and memory, nature of the problem demands an algorithm which can cluster high dimensional input data efficiently using limited resources in real-time.

Methods for stream data clustering can be broadly classified into Partitioning, Hierarchical, Density-based, Grid-based, Model-based [5]. In this study, we focus on three stream data clustering methods namely EDMStream [4], CluStream [1], D-Stream [3] and their ability to track clusters and two cluster evolution detection methods namely E-Stream [9], MEC algorithm [7]. CluStream is a hierarchical method which extends from STREAM [6] whereas EDMStream and D-Stream are density based methods. Density based methods do not expect us to configure the number of clusters to be found beforehand and usually rely on the assumption that the high density data point is surrounded by lower density points. Density Peaks clustering [8] serves as a base for EDMStream in modeling the dependencies of data points in a tree like data structure.

One of the popular approaches to stream data clustering is the two phase online-offline method where streaming data is summarized in the online phase and used for clustering in the offline phase whenever clustering is initiated. This might not provide results in real-time as the user has to sometimes start the clustering process or wait until the clustering is complete if it takes time.

Although there are endless applications for stream data clustering, we would like to mention the classic recommendation systems in which the results of clustering methods can directly influence the end user. A typical real-time product like news which is consumed real-time is recommended based on the cluster in which the user has already visited or liked earlier. News is generated continuously and fades out as time progresses which calls for a real-time stream data clustering method to effectively cluster and deliver it to the end user. Density-based methods are appropriate for such data streams as the clusters are extended or merged iteratively using density information.

The following is the structure of this paper. Section 2 introduces the basic concepts related to stream data clustering and the overview of summary and data structures used in the literature studied. Section 3 introduces the algorithms EDMStream, CluStream, E-Stream, D-Stream and MEC briefly and lists out the most important characteristic features. In section 4, experimental evaluations performed in the datasets in the literature is discussed. Section 5 concludes the paper with few open ended questions in the paradigm of clustering evolving stream data.

Advanced Data Engineering, IPVS
2019.

2 RELEVANT ASPECTS

In this section, basic concepts related to stream data clustering and the structures used to summarize the stream data are introduced. Also a brief overview Density Peaks clustering is given. Building upon these, data structures used to represent the summary structures are shortly discussed.

2.1 Basic Concepts

Stream data clustering: Given data stream S_N where each point is a d-dimensional vector, stream clustering returns a disjoint set of clusters at any time $t_1, t_2, ..., t_N$. Here time stamp is added to the data point to decay the freshness of the data as time progresses.

Cluster Evolution: As data is evolving, clusters may be added, deleted or modified which is tracked over a period of time. Clustered results over different intervals in time are needed to perform this evolution tracking. Types of evolution are pre-defined according to the nature of the algorithm.

Decay model: As importance of data is decayed over time, it is usually modeled using exponential time dependent weighting function. If no nearby points arrive, point density of the data point decreases over time. All points decay at same pace using freshness of point i at time t.

2.2 Summarization of Data

Synopsis of unbounded stream data is required to reduce the memory overhead and to produce clustering results quickly. Different summarization techniques such as Cluster-cell, Micro-clusters and Grids have been employed and later, clustering algorithm is performed on the summarized data.

Cluster-cell: Set of close data points are summarized as a Cluster cell. Cluster cell acts as the basic unit instead of a data point. It is characterized by a three-tuple: {seed point, density, dependent distance} at any time t. Seed point summarizes a set of points whose distance to seed point is less than or equal to a predefined radius r i.e., $P_c = \{p_i : s_c = argmin(|p_i, s_k|), |p_i, s_c \leq r|\}$, where s_k belongs to set of all cluster seeds. Intuitively, a cluster-cell is parametrized by radius r which has an impact on the size of the clusters we obtain after clustering. Smaller value for the radius r results in fine-grained clusters whereas larger value for the radius r produces coarse-grained clusters. Summarization of all cluster-cell points timely density is stored as density.

$$\rho_c = \sum_{p_i \in P_c} f_i^t \tag{1}$$

where $f_i^t = a^{\lambda(t-t_i)}$ is the freshness of point i at time t and parameters a and λ control the decay function. From its seed point to its nearest cluster-cell seed point with higher cluster-cell density is stored as dependent distance, which is defined similar to that of Density peaks clustering in Section 2.3.

Micro-clusters and Pyramidal time frame: Stats of data locality are stored as Micro-clusters which are considered as a temporal extension of Cluster Feature (CF) vector, which was used in one of the earliest stream clustering method BIRCH [10]. As Micro-clusters are additive, they are stored as snapshots at pyramidal time frame structure to manage the storage complexity. Using the subtractive property later, snapshots from different time frames are fetched and compared to produce higher level clusters. Micro-clusters are denoted by $M_1...M_q$ of multi-dimensional records $X_1...X_k$ arriving at time stamps $T_1...T_k$. Each X_i is a d-dimensional record which are denoted by $X_i = (x_i^1...x_i^d)$. Micro-cluster for a set of d-dimensional points $X_{i_1}...X_{i_n}$ with time stamps $T_{i_1}...T_{i_n}$ is a $(2.d + 3)$ tuple which contains sum of squares of data values $\sum_{j=1}^{n}(x_{i_j}^p)^2$, sum of data values $\sum_{j=1}^{n} x_{i_j}^p$ in the p entry [1]. Also, sum of squares of time stamps and sum of time stamps are also maintained along with these features.

Grids: Input stream (d-Dimensional) is mapped into the density grids. A data record x is mapped to the density grid g and the corresponding density coefficient at time t is computed as $D(x, t) = \lambda^{(t-t_c)}$ where t_c is the arrival time of data point x and $\lambda \in (0, 1)$ is a constant decay factor. Density coefficient decreases as x ages and density of grids is updated considering the decay factor into account. All grids decay at the same proportional rate and the update is lazily performed only when the density value in the grid is updated with the addition of a new data point changing over time. A data record $x = (x_1...x_d)$ is mapped to a density grid $g(x)$ as:

$$g(x) = (j_1, j_2..., j_d) \tag{2}$$

where $x_i \in S_{i,j}$ and $j_i = 1, ...p_i$ indicating that the input space $S_{i,j}$ has p_i partitions. Density of the grid g at any time t is computed as the sum of the density coefficients of all data records that are mapped to g. Figure 1 shows how the density grids fit into the stream data clustering process. Further to avoid storing time stamps and densities of all data records, characteristic vector is proposed. *Characteristic Vector*: For a grid g, Characteristic Vector is the tuple $(t_g, t_m, D, label, status)$, where t_g is the last time g is updated, t_m is the last time when g is removed from sporadic grid list, D is the grid density at the last update, $label$ is the class label of the grid and $status = \{Sporadic, Normal\}$ is used to remove sporadic grids.

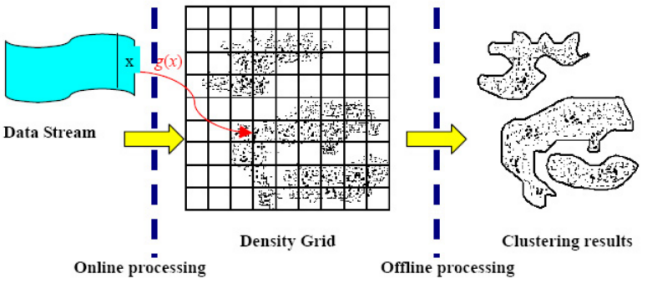

Figure 1: Density grids in the process of stream clustering [3]

2.3 Density Peaks Clustering

Density Peaks (DP) Clustering [8] is based on two metrics, the local density and dependent distance of the data points. It is based on the observation that the cluster centers are usually surrounded by points with lower local densities and they are at a relatively large distance from any points with higher local densities [4].

Local density: Number of points whose distance to a data point p_i is smaller than the cutoff distance d_c is computed as local density of point p_i.

$$\rho_i = |\{p_j \mid |p_i, p_j| < d_c\}| \tag{3}$$

where $|p_i,p_j|$ is the euclidean distance from point p_i to p_j.

Dependent distance: The minimum distance from point p_i to any other point whose local density is higher than that of point p_i is computed as dependent distance δ_i of point p_i.

$$\delta_i = \min_{j:\rho_j > \rho_i} (|p_i, p_j|) \tag{4}$$

This method uses these two metrics to find the cluster centers which are the density peaks. A point p_i is declared as a *dependent* on point p_j if p_j is the highest density nearest neighbor of p_i.

Points with lower local densities are termed as *outliers* regardless of the value of dependent distance. For example, if the local density of the point ρ_i is lower than the predefined value ς then the point is termed as an outlier ($\rho_i <= \varsigma$) irrespective of the dependent distance. Points with high local density and large dependent distance are recognized as the cluster centers. Based on the threshold τ, a strong dependency or a weak dependency is established between the points which assists in forming a dependency chain. These dependency chains are the clusters with the root of the chain being the cluster center. After finding density peaks and fixing cluster centers, remaining points are alloted to same cluster as its dependent point. Figure 2 shows the distribution of 2D data points on plane view and a decision graph by using local density ρ and δ. Density peaks are the top right area in the decision graph ($\rho_i > \varsigma$ and $\delta_i > \tau$) whereas outliers are in the left region ($\rho_i < \varsigma$).

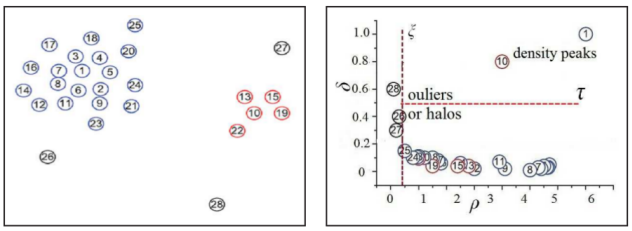

Figure 2: Density Peaks Clustering [8]

2.4 Data Structures

In the methods studied, tree like structures are used to abstract density information and graph like structures are used to model the relationship between clusters over time. In this section, the data structures used in the stream clustering algorithms are discussed.

Dependency Tree: In Density Peaks clustering, local density and dependent distance of each data point are computed and are used to find the density peaks which are termed as the cluster centers. Points with low local density are termed as outliers and excluded from the clustering process until the density around the outliers increases. In Dependency tree, local density and dependent distance are modeled as a tree-like structure as shown in the figure 3. Point to point dependencies are abstracted since each point is only dependent on a single point except for the root. If all links in a subtree are strongly dependent, then the subtree is declared as a strongly dependent subtree. If there is no other strongly dependent subtree T_j such that T_i is a subtree of T_j, then T_i is Maximal Strongly Dependent SubTree (MSDSubTree).

Following points describe the clustering process based on the DP-Tree: 1. Each MSDSubTree corresponds to a cluster - Root of a MSDSubTree is the cluster center 2. Stream Clustering using DP-Tree: Find all MSDSubTrees from a dynamic DP-Tree 3. Evolution Tracking using DP-Tree: Monitoring DP-Tree changes. Goal is to monitor and maintain the dynamic DP-Tree and return the MSDSubTrees.

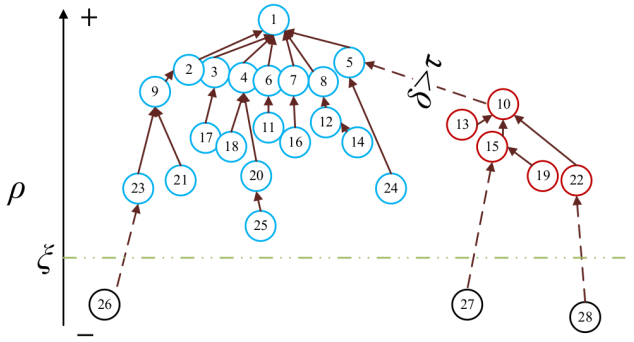

Figure 3: Illustrative figure of DP Tree [4]

Bipartitie graphs: To model the evolution of clusters over time, bipartite graphs are used with weights over the edges. Conditional probabilities are computed for every possible pair of clusters obtained at consecutive time frames and are represented as edges in the bipartite graph. This is used to evaluate the similarity between two clusters in consecutive periods as it is a established mathematical concept [7]. Weighted bipartite graphs are shown in Figure 4 which displays two subsets, U and V in t and $t+1$ time frames.

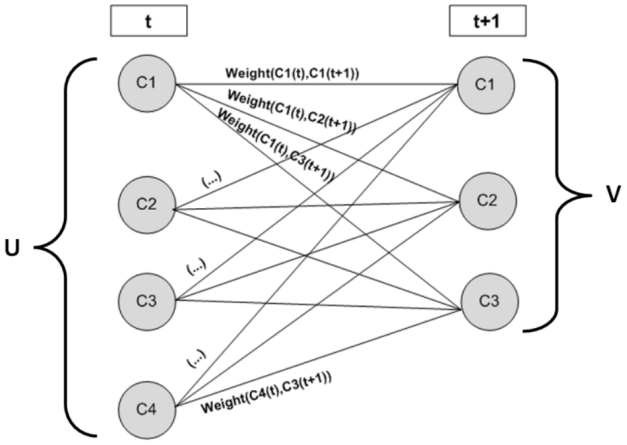

Figure 4: Bipartite graph between two consecutive time frame [7]

Fading Structure with Cluster Histograms: In E-Stream[9], each cluster is represented as a Fading Cluster Structure (FCS) utilizing a α-bin histogram for each feature of the dataset. The cluster representation is called as Fading Cluster Structure with Histogram (FCH). They compute $FC1$, $FC2$ and Weights W from the data points along with the fading weight of the data point. $FC1(t)$ is a vector of weighted sum of data feature values at time t. Similarly, $FC2(t)$ is the weighted sum of each data feature at time t. The Histogram of cluster data values is used to identify cluster splits. A α-bin histogram summarizes the distribution of cluster data for each dimension of each cluster [9]. Cluster split is based on the distribution of feature values as summarized by the cluster histogram.

3 ALGORITHMS FOR STREAM DATA CLUSTERING

In this section, overview of stream clustering algorithms which use the summary structures and data structures in the previous section is presented.

3.1 EDMStream

The core idea of this algorithm is to summarize the set of close points as a cluster-cell introduced in section 2.2 and use the tree structure, Dependency Tree introduced in section 2.3 to perform clustering. Efficient hierarchical tree like structure Dependency Tree is used to abstract the density information of the data points. Dependency Tree (DP-Tree) is updated to reflect the relationship between cluster-cells. It also tracks the cluster evolution activities by tracking updates of DP-Tree. Authors claim that it can track five types of cluster evolution namely *Emerge, Disappear, Split, Merge, Adjust* which are defined according to the conventional evolution ideas in [4]. Some stream clustering algorithms work by taking user input into consideration in the offline clustering stage. In the algorithm EDMStream proposed by [4], the authors claim that one of the characteristic features is the automatic adjustment of clustering results based on the user input for cluster granularity.

Two storage structures, *DP-Tree* and *Outlier reservoir* are used in order to cluster the stream data. DP-Tree is used to model the dependencies between cluster-cells. Outlier Reservoir is used to limit the size of DP-Tree. There are four key operations involved in the algorithm which is described below that sums up the clustering process as shown in the Figure 5.

New point assignment: Newly arrived point is not directly inserted to the DP-Tree but used to generate a new cluster-cell or increase an existing cluster-cell's density. This leads to DP-Tree update which is covered in the next key operation called Dependencies update. Whenever new data points arrive, those are added to the existing cluster-cell or forms a new outlier cluster-cell. It is added to a cluster-cell c if the distance to cluster seed Sc is smaller or equal to r and Sc is the closest seed.

Dependencies update: As new data points are merged into the existing cluster-cell, densities of the cells need to be updated according to,

$$\rho_c^{t_j+1} = a^{\lambda(t_j+1-t_j)}\rho_c^{t_j} + 1 \tag{5}$$

where parameters a and λ describe the decay model. Density changes result in dependency changes and this in turn results in increase

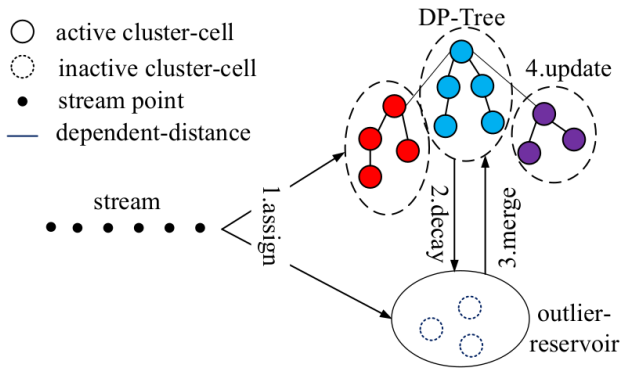

Figure 5: Overview of EDMStream [4]

of computational cost [4]. Set of cluster-cells whose density higher than c at time t_j are collected as,

$$F_c{}^{t_j} = \{c' | \rho_c^{t_j} < \rho_{c'}^{t_j}\} \tag{6}$$

and the dependent cluster cells of c at time t_j are updated. If c' absorbs a new point, for each c, the local density previously higher than or equal to but currently lower than should be updated. From the DP-Tree point of view, nodes previously at higher levels than c' but now at lower levels need to be updated. To reduce update cost, EDMStream filters out the cluster-cells which does not require update by using two filtering techniques which are density filter and triangle inequality filter.

Cluster-cell emergence: Active cluster-cells reside in DP-Tree and inactive cluster-cells reside in outlier reservoir. Over a period of time, low density cluster-cells might absorb new points and become dense. A cluster-cell is declared as active based on the condition,

$$\rho_c^t \geq \beta.v/(1-a^{\lambda}) \tag{7}$$

where β is a tunable parameter and v is the point arrival rate.

Cluster-cell decay: Active cluster-cell becomes inactive if it doesn't absorb points. Once it has become inactive, it is moved from DP-Tree to outlier reservoir. Successors of the corresponding cluster cell should also be moved to the outlier reservoir. Also, memory recycling is done to prevent overflow issues in outlier reservoirs.

3.2 CluStream

CluStream [1] separates the process of stream clustering into online and offline components. Summary statistics of the data are stored in the online phase as Micro-clusters defined in Sec. 2.2.1 and in the offline phase, using the summary data clustering is performed along with the user input.

In the online phase, summary statistics of data is stored as micro-clusters as snapshots in time. Snapshots are taken based on the pyramidal time frame. These are needed to save the memory space and to fetch the micro-clusters in history whenever the request is made. When a new data point X_i comes in, it should be either added to the existing micro-cluster or should form a new outlier micro-cluster. The decision to add it to the existing micro-cluster M_p is made by using Maximum boundary of micro-cluster which

is defined as a factor of t of the RMS deviation of the data points in M_p from the centroid. If the new data point X_i lies within the maximum boundary of the micro-cluster M_p it is added to the micro-cluster M_p. Otherwise a new micro-cluster with data point X_i is created. In order to create a new cluster, memory space is freed by either deleting an existing outlier cluster or merging two by the maintenance algorithm.

In the offline phase, when the time horizon of length h is inputted from the user at time t_c, the stored snapshot is fetched by approximating in the pyramidal time frame before $t_c - h$. Using a modified K-means algorithm and these micro-clusters from the snapshots as input, macro-clustering and evolution analysis is performed.

3.3 D-Stream

[3] provides an algorithm called D-Stream which is similar to CluStream [1] in terms of online and offline components. D-Stream maps the streaming high dimensional input data into a grid in the online phase and computes the density of grids defined in Section 2.2. New data records are mapped into the corresponding grid and it also adopts a density decaying technique to track the dynamic changes of the data stream. Based on density grids defined in Sec. 2 , for the new data point x, the corresponding grid is mapped and characteristic vector is updated. After gap (integer parameter) time steps, D-Stream removes the sparse grids and updates the clusters. Here, a cluster is a set of adjacent dense grid cells.

```
1.  procedure D-Stream
2.      t_c = 0;
3.      initialize an empty hash table grid_list;
4.      while data stream is active do
5.          read record x = (x_1, x_2, ..., x_d);
6.          determine the density grid g that contains x;
7.          if(g not in grid_list) insert g to grid_list;
8.          update the characteristic vector of g;
9.          if t_c == gap then
10.             call initial_clustering(grid_list);
11.         end if
12.         if t_c mod gap == 0 then
13.             detect and remove sporadic grids from grid_list;
14.             call adjust_clustering(grid_list);
15.         end if
16.         t_c = t_c + 1;
17.     end while
18. end_procedure
```

Figure 6: Overall procedure of D-Stream [3]

Figure 6 shows the procedure of D-Stream from [3]. At every time step, the online component of D-Stream reads the new data record and maps the data record to the corresponding grid (lines 5-6). A list of grids considered for clustering at any point in time is maintained and the characteristic vector is updated for the grid (lines 7-8). The parameter *gap* determines the time interval between the consecutive offline clustering. After the first gap, initial clustering is performed (lines 9-10). In the offline phase, adjustment of clustering and removing sporadic grids from the list of grids is performed periodically according to the *gap* parameter (lines 12-15).

3.4 E-Stream

As properties of data evolve over time, [9] proposes a new technique called E-Stream for stream data clustering and improves upon the existing stream clustering algorithm CluStream by supporting five evolutions that are appearance, disappearance, self-evolution, merge and split. Each cluster is represented as a Fading Cluster Structure with Histogram (FCH) and tracked by E-Stream with the help of cluster histograms when it evolves over time.

```
Algorithm E-Stream
1   retrieve new data X_i
2     FadingAll
3     CheckSplit
4     MergeOverlapCluster
5     LimitMaximumCluster
6     FlagActiveCluster
7     (minDistance, index) ← FindClosestCluster
8     if minDistance < radius_factor
9       add x_i to FCH_index
10    else
11      create new FCH from X_i
12  waiting for new data
```

Figure 7: E-Stream algorithm [9]

Algorithm from the original paper [9] is given below in figure 7. Once a new data point is received (line 1), it fades all the clusters and performs histogram analysis to check for the split in clusters (lines 2-3). It merges the overlapping clusters based on the merge threshold and limits the cluster by merging closest pairs if the cluster count exceeds the maximum cluster limit (lines 4-5). All active clusters are flagged (line 6) and the distance between the closest point in the matching cluster and the new point is found (line 7). If the distance is less than the $radius\_factor$, the point assigned to the cluster otherwise it is classified as isolated (lines 8-11). The flow awaits the new point and continues after it receives a new point as discussed.

3.5 MEC Algorithm

To monitor the evolution of clusters over time, a method called MEC is proposed by [7]. Based on computation of conditional probabilities as weights on the edges of the bipartite graphs described in Sec. 2.2, MEC detects and categorizes cluster transitions. Previously discovered clusters are assumed to be passed as input to the MEC algorithm and it focuses on tracking the evolution of clusters. Since it uses conditional probabilities as weights of edges, it requires structural consistency in datasets.

Cluster is represented by enumeration which is the straight forward approach of assigning data points to the cluster. Cluster's birth is characterized either by outlier points absorbing new points or new set of dense points from the data stream. Cluster's death is the disappearance of the existing cluster as it decays over time. Cluster is split or merged whenever a distribution of the data point change based on a split threshold defined.. Survival of a cluster at $t_i + \delta t$ is the either adjustment of data points within the set of clusters at

Transitions' Taxonomy	Notation
Cluster's Birth	$\emptyset \rightarrow C_u(t_{i+\Delta t})$
Cluster's Death	$C_m(t_i) \rightarrow \emptyset$
Cluster's Split	$C_m(t_i) \overset{S}{\rightarrow} \{C_1(t_{i+\Delta t}), ..., C_r(t_{i+\Delta t})\}$
Cluster's Merge	$\{C_1(t_i), ..., C_p(t_i)\} \overset{S}{\rightarrow} C_u(t_{i+\Delta t})$
Cluster's Survival	$C_m(t_i) \rightarrow C_u(t_{i+\Delta t})$

Figure 8: Cluster transitions defined in MEC [7]

previous time t or the fading of existing cluster without absorption of any points. First four transformations result in change of number of clusters whereas last one results in change of point to cluster assignment. Based on these, a taxonomy of cluster transitions are defined which is shown in Figure 8.

4 EVALUATION AND DISCUSSION

Metrics for evaluating clustering techniques is based on quality and purity of clustering results on synthetic and real world data sets.

Measuring cluster quality: Measures for evaluating clustering quality are broadly classified as internal and external measures [5]. Sum of Squared Distance: The internal measure to evaluate the similarity within a cluster in partitioning based algorithms is the sum of squared:

$$SSQ = \sum_{k=1}^{k} \sum_{x_i \in D_k} \| x_i - \mu_k \|^2 \tag{8}$$

where D_k is the set of objects in cluster k and μ_k is its centroid. It measures how far the objects of each cluster are from the center of its cluster, small SSQ means better compactness of each cluster. Also, in real time scenario the ability to respond faster is crucial for a stream clustering method.

Purity: Purity is related to entropy in Information theory. Each cluster is assigned to its majority class as in,

$$v_i = \frac{1}{N_j} argmax_i(N_j^i) \tag{9}$$

where v_i is the number of objects in cluster C_j from class i. Purity is the sum of v_j over all clusters as given by,

$$Purity = \sum_{j=1}^{k} \frac{N_j}{N} v_j \tag{10}$$

Evaluation on real world datasets: In the case study presented in [4], cluster evolution in news recommendation is monitored. Using NADS news stream dataset, cluster evolution is tracked. Based on the results, EDMStream can identify different types of cluster evolution activities [4]. It reflects the trends in real world.

On the network intrusion dataset KDDCUP99 (which has 494,021 instances, 34 dimension and 23 clusters), CluStream and D-Stream is compared using the mean SSQ value. D-Stream outperforms CluStream having clusters similar to the ground truth than the latter which is shown in the figure 9. For KDDCUP99 dataset, response time of the methods is compared. EDMStream has lower response time than D-Stream as the length of the stream increases

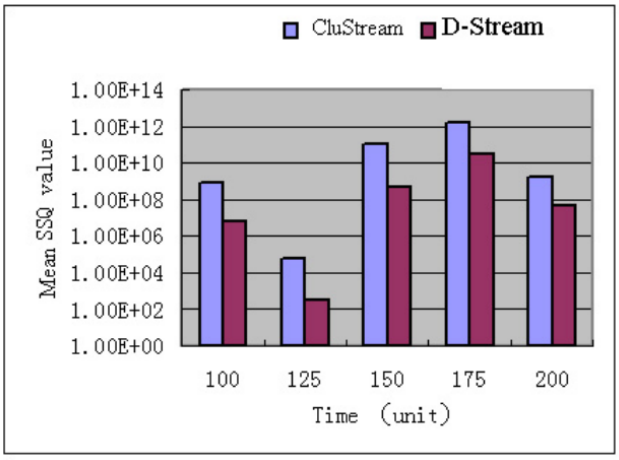

Figure 9: Comparison of CluStream and D-Stream on KDD CUP-99 dataset [3]

and clearly outperforms its counterparts as shown in figure 10. The counterparts DenStream, DBStream are density-based stream data clustering methods based on the traditional DBSCAN algorithm.

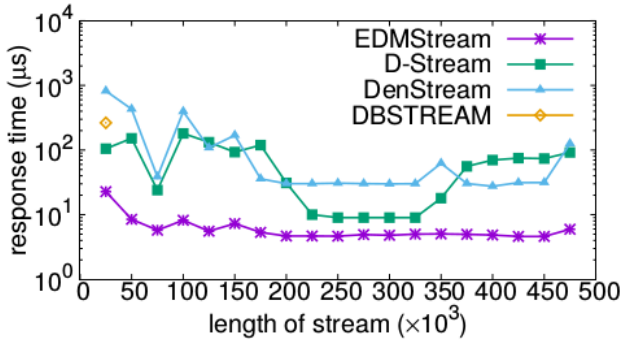

Figure 10: Comparison of response time on KDDCUP99 dataset [4]

As CluStream acts as a baseline for E-Stream, performance of E-Stream is similar to that of CluStream with additional support cluster evolution. In figure 11, the purity of E-Stream algorithm is measured by varying the number of clusters k as the input parameter on a synthetic dataset. E-Stream achieves a purity of 0.9 and is not sensitive to the input parameter. MEC algorithm is applied to real world economic and financial datasets and case studies are presented in [7].

Summary of EDMStream [4], CluStream [1] and D-Stream [3] is provided in the table 1.

5 CONCLUSION AND OUTLOOK

In this work, an overview of different methods for clustering stream data is presented. In evolving data streams, it is crucial to identify trends in real-time. EDMStream provides real-time tracking method better than other clustering methods. However, since EDMStream

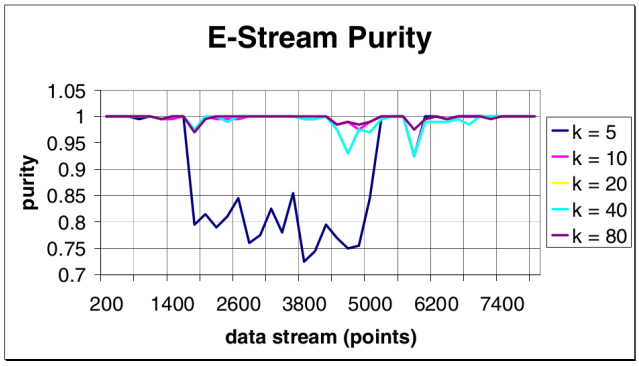

Figure 11: EStream - Sensitivity with number of cluster [9]

	EDMStream	CluStream	D-Stream
Summary structure	Cluster-cell	Micro-clusters	Grids
Based on	Density	Partitioning	Density
Real-time response	Yes (in 7-23 μs)	No	No
Two phase (online-offline)	No	Yes	Yes
Remarks	Tracks cluster evolution and provides incremental update	It is a hierarchical clustering approach and based on modified version of K-means	Efficient in identifying outliers and cannot handle high dimensional data

Table 1: Summary of EDMStream, CluStream and D-Stream

is a density based method, expected density (size of cluster-cell) should be configured as a parameter and the performance can vary based on the value for bigger datasets. In CluStream, as it is a modified k-means algorithm in offline phase, expected number of clusters should be configured. Due to the dynamic nature of the data stream, the stream clustering algorithms should be robust enough to deal with outliers and noise.

Based on the different works studied, it seems that there is still room for improvements in terms of improving the clustering quality by effectively learning a forgetful model [2] and evolve according to the data stream. Also, context based clustering algorithms which provide an intuitive meaning to the evolution of clusters based on the context is required. Stream clustering methods should handle heterogeneous data sources as there is an increasing need to handle and cluster data from multiple types of data sources in the future.

REFERENCES

[1] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. 2003. A Framework for Clustering Evolving Data Streams. In *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29 (VLDB '03)*. VLDB Endowment, 81–92. http://dl.acm.org/citation.cfm?id=1315451.1315460

[2] Douglas O. Cardoso, Felipe França, and João Gama. 2016. Clustering Data Streams Using a Forgetful Neural Model. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing (SAC '16)*. ACM, New York, NY, USA, 949–951. https://doi.org/10.1145/2851613.2851889

[3] Yixin Chen and Li Tu. 2007. Density-based Clustering for Real-time Stream Data. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '07)*. ACM, New York, NY, USA, 133–142. https://doi.org/10.1145/1281192.1281210

[4] Shufeng Gong, Yanfeng Zhang, and Ge Yu. 2017. Clustering Stream Data by Exploring the Evolution of Density Mountain. *Proc. VLDB Endow.* 11, 4 (Dec. 2017), 393–405. https://doi.org/10.1145/3186728.3164136

[5] Umesh Kokate, Arvind Deshpande, Parikshit Mahalle, and Pramod Patil. 2018. Data Stream Clustering Techniques, Applications, and Models: Comparative Analysis and Discussion. *Big Data and Cognitive Computing* 2 (10 2018), 32. https://doi.org/10.3390/bdcc2040032

[6] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani. 2002. Streaming-data algorithms for high-quality clustering. In *Proceedings 18th International Conference on Data Engineering*. 685–694. https://doi.org/10.1109/ICDE.2002.994785

[7] Márcia Oliveira and João Gama. 2012. A Framework to Monitor Clusters Evolution Applied to Economy and Finance Problems. *Intell. Data Anal.* 16, 1 (Jan. 2012), 93–111. https://doi.org/10.3233/IDA-2011-0512

[8] Alex Rodriguez and Alessandro Laio. 2014. Clustering by fast search and find of density peaks. *Science* 344, 6191 (2014), 1492–1496. https://doi.org/10.1126/science.1242072 arXiv:http://science.sciencemag.org/content/344/6191/1492.full.pdf

[9] Komkrit Udommanetanakit, Thanawin Rakthanmanon, and Kitsana Waiyamai. 2007. E-Stream: Evolution-Based Technique for Stream Clustering. In *Proceedings of the 3rd International Conference on Advanced Data Mining and Applications (ADMA '07)*. Springer-Verlag, Berlin, Heidelberg, 605–615. https://doi.org/10.1007/978-3-540-73871-8_58

[10] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. 1996. BIRCH: An Efficient Data Clustering Method for Very Large Databases. *SIGMOD Rec.* 25, 2 (June 1996), 103–114. https://doi.org/10.1145/235968.233324