Clustering Based on Correlation Fractal Dimension Over an Evolving Data Stream

Anuradha Yarlagadda[1], Murthy Jonnalagedda[2], and Krishna Munaga[2]
[1]Department of Computer Science and Engineering, Jawaharlal Nehru Technological University, India
[2]Department of Computer Science and Engineering, University College of Engineering Kakinada, India

Abstract: *Online clustering, in an evolving high dimensional data is an amazing challenge for data mining applications. Although, many clustering strategies have been proposed, it is still an exciting task since the published algorithms fail to do well with high dimensional datasets, finding arbitrary shaped clusters and handling outliers. Knowing fractal characteristics of dataset can help abstract the dataset and provide insightful hints in the clustering process. This paper concentrates on presenting a novel strategy, FractStream for clustering data streams using fractal dimension, basic window technology, and damped window model. Core fractal-clusters, progressive fractal-cluster, outlier fractal clusters are identified, aiming to reduce search complexity and execution time. Pruning strategies are also employed based on the weights associated with each cluster, which reduced the usage of main memory. Experimental study of this paper over a number of data sets demonstrates the effectiveness and efficiency of the proposed technique.*

Keywords: *Cluster, data stream, fractal, self-similarity, sliding window, damped window.*

Received January 24, 2014; accepted October 14, 2014

1. Introduction

Huge amount of streaming data is generated in recent years for example internet traffic flow, sensor data, medical systems, online transactions etc., Stream data flows in and out of a system continuously and with varying update rates. They are fast changing, massive, potentially infinite and unbounded [14]. By the virtue of Data stream characteristics, stream clustering became challenging due to limited memory and real time query response requirements. Massive volumes of data should be handled with limited memory. We cannot scan such huge amount of data more than once [15]. New concepts may keep evolving in data streams over time. Evolving concepts require data stream processing algorithms to continuously update their models to adapt the changes. Fractal dimension is a powerful tool to describe self-similarity, and changes in the correlation dimension imply changes of data distribution, which can be used to indicate changes in data trends. Summaries of the processed data help in computing important statistics of new clusters with the arrival of new points [6]. In data stream mining the proposed algorithm should handle online clustering meritoriously and should maintain the clusters considering the potentiality of clusters. Time plays a major role in data stream clustering as a data point belonging to a cluster in some time horizon can become an outlier in some other time horizon as most recent data plays an important role. Similarly, in a cluster where there are no data points in one time horizon may accommodate more points in another time horizon making it more progressive. So the proposed algorithm should maintain a balance not to reject the outliers at once, but should wait for some time to remove it as outlier aiming to find clusters of arbitrary shape. A large number of clustering algorithms [20] for data streams have been proposed, where the similarity of the objects is defined with use of some distance measure or objective function. The proposed algorithm uses correlation fractal dimension for finding arbitrary shaped cluster and further improving the precision of clustering. Data points are merged into a cluster who’s Relative Change in the Fractal Dimension (RCFD) is less than a minimum threshold and if no points are added to any of the clusters within a stipulated time horizon, then they are considered as real outliers. This paper extends the prior work [8] using the concept of fractal dimension and multi layered grid. Weights are assigned to the clusters using damped window model and pruning is done based on the importance of the clusters which improve the results of the clustering.

The rest of the paper is organized as follows. Next section describes some existing data stream clustering algorithms. Section 3 describes some basic concepts and definitions used for clustering the data streams. Section 4 presents the algorithms for initialization step and incremental step for online clustering. Then the experimental results are given in section 5 and section 6 concludes this paper with the discussion of future work.

2. Related Works

Massive volumes of stream data are generated every day in recent years, as there is rapid development of computer, communication and network [11]. Analysing and clustering such type of data has become highly demanding [15, 19]. Guha *et al.* [12] proposed the LOCALSEARCH in which description for current data stream is projected, but the evolution of data stream is not reflected. A primitive algorithm in Data Stream area is STREAM [18] where clustering of objects is done using K-means and medians of each group are considered for clustering. But STREAM neither considers evolution of data nor the time granularity. Aggarwal *et al.* [1] proposed the CluStream, which was an evolution framework of data stream clustering. CluStream is an incremental clustering algorithm, and it can offer clustering results with different time granularity. But the clustering result tends to be spherical and doesn't work well for the clusters of arbitrary shapes. HPStream [3] solved the high-dimensionality problem by the projection technology. But in this algorithm, the framework of CluStream is abandoned. Udommanetanakit *et al.* [24] is proposed by for E-Stream clustering which supports evolution in terms of five properties and identifies only spherical clusters where the similarity between the two objects is estimated by using the distance function. Chen and Tu [9] presented D-Stream in which two-phased clustering framework is adopted using the concept of density grids. It can find clusters with arbitrary shapes but it sequentially examines neighbouring grids on all the dimensions, which is very time-consuming. Cao *et al.* [8] proposed DenStream algorithm, a density-based clustering algorithm. DenStream also involves two phase clustering. The concept of core point introduced in DBSCAN [10] is extended in this method and the notion of micro-cluster is employed to store an approximate representation of the data points. DenStream identifies clusters with arbitrary shapes and inspects outliers. But the clustering result is very sensitive to parameters. rDenStream [17] is based on DenStream and presents the concept of outlier retrospect which is a three step method. HDenStream [16] is also a density based algorithm capable of clustering data streams with heterogeneous features where both continuous attributes and categorical attributes are used for the clustering. But the storage of the heterogeneous attributes is very inefficient and the calculation of distances is time consuming. Ren *et al.* [21, 22] proposed clustering algorithms for high dimensional data streams, inspired by HPStream [2] where each dimension of the corresponding cluster in the matrix is associated with a weight. The weight of a dimension shows the importance of that dimension to the corresponding cluster. Fractal has been widely used in data mining field [13]. Distributed Grid based clustering protocol is proposed by Ali and Madani [4]. Feature attributes selection method based on the fractal dimension is proposed by Traina *et al.* [23], in which the change degree of fractal dimension is regarded as the standard of selecting feature attributes. Barbara and Chen [5] applied fractal algorithm to cluster data sets, whose basic idea is that the self-similarity within cluster is higher than in different clusters. Arbitrary shaped clusters are formed but multiple scans are needed in the two methods above, which is not suitable for data stream clustering. In order to find arbitrary shape cluster effectively in data stream and improve the precision of clustering with single data scan, a fractal clustering algorithm FractStream is designed and proposed.

3. Basic Concepts

A multi layered nested grid structure [25] which stores the statistics of data is constructed for finding the Fractal dimension. Fractal dimension of data set is calculated easily based on number of data points included by each of the lowest grids of a multi-layered Grid, by scanning the data set only once. Cluster partitions on evolving data streams are often computed based on certain time intervals (or windows). There are three well-known window models: landmark window, sliding window and damped window.

In order to meet the requirements of evolving data stream, we adopt basic window technology based on sliding window model [26]. The sliding window is equally divided into shorter windows, called as basic windows. The relationship between two kinds of window is shown as Figure 1.

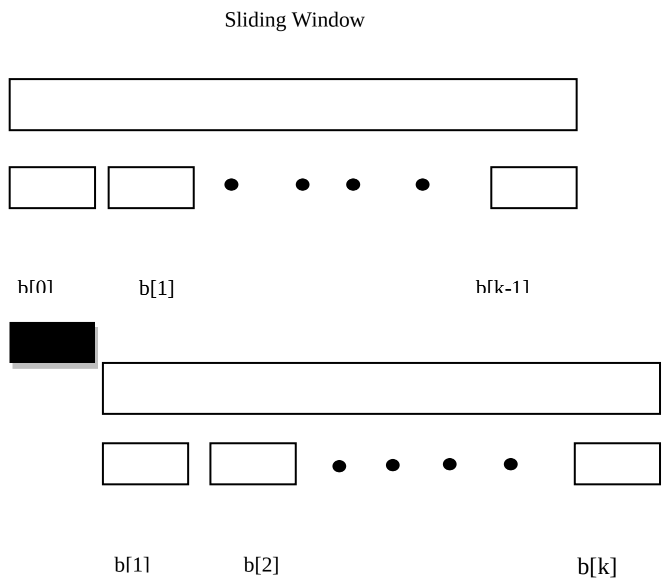

Figure 1. Sliding window and basic windows.

Let a be the length of the basic windows, suppose w=k*a, k denotes the number of basic windows in a sliding window. Let b[0], b[1], ..., b[k-1] denote the current basic windows, b[k] will be the coming new basic window and b[0] will be the expiring basic window. Both the sliding window and basic windows are maintained.

Each data point is associated with time stamp in an evolving data stream. A time window of data point is added to the low level cell thus obtaining the number of occupancies in the cell. A damped window model is considered to manage huge volumes of data and subsequently the memory. In damped window model, it is assumed that each data point has a time-dependent weight defined by the function $f(t)$ given by Equation 1. The function $f(t)$ is also referred to as the fading function and is a non-monotonic decreasing function which decays uniformly with time t and defined in Equation 1 [1]. Suppose the initial time of the current window is t_o and the numbers of points are $n_o, n_1, \ldots, n_e$ at $t_o, t_l, \ldots, t_e$ time respectively, the weight of the current window is calculated as Equation 2.

$$f(t)=2^{-\lambda t}, \lambda>0 \tag{1}$$

$$W(t_c)=\sum_{i=1}^{e} f(t_i - t_0) \tag{2}$$

- *Definition 1*: correlation fractal dimension given a data set containing N data points which show self-similarity in the range of scales (r_{min}, r_{max}), the correlation fractal dimension D_2 [7] is measured as follows:

$$D2=\frac{\partial \log \Sigma C_{r,i}^{2}}{\partial \log r}, \quad r \in [r_{min}, r_{max}] \tag{3}$$

Where $C_{r,\,i}$ is the occupancy with which the data points fall in the i^{th} cell when the original space is divided into grid cells with sides of length r.

The relative fractal dimension change in more accurate in describing the change in pattern of the data set to form a new cluster. In order to measure the change in fractal dimension which corresponds to the change in patterns, we formally define relative fractal dimension change, *RCFD*. The smaller the *RCFD* the greater is the self-similarity with in a cluster otherwise it implies low self-similarity [13].

- *Definition 2*: *RCFD* Given data stream S, cluster C, and a new cluster C' formed by joining a new set of data points. $Fd(C)$ is the fractal dimension of the old cluster, C and $Fd(C')$ is the fractal dimension of the new cluster, C'. Relative change in the fractal dimension is defined as

$$RCFD=\frac{|Fd(c')-Fd(c)|}{|Fd(c)|} \tag{4}$$

Here Fd refers to correlation fractal dimension D_2, which can reflect the data distribution and indicate the change of data trend.

- *Definition 3*: Core Fractal Cluster (CFC). Let the cluster has $P_{i1}, P_{i2}, P_{i3}, \ldots, P_{in}$ data points with time stamps $t_{i1}, t_{i2}, t_{i3}, \ldots, t_{in}$ respectively, the *CFC* is defined as tuple $CFC(Fd, w, n, t_c)$ in which Fd is the fractal dimensions of the cluster, $w=\sum_{j=1}^{n} f(t_c - t_{ij}) > \mu$ is the weight of the cluster at time t_c, n is the total number of points in the cluster and t_c is the initial time of the cluster and μ is the ceiling value.

The weight of the cluster varies with time in the real time data stream environment and it decreases with time decay. Definition of Progressive Fractal Clusters (PFC) and Outlier Fractal Clusters (OFC) are introduced to track the evolution of the clusters according to their importance.

- *Definition 4*: A PFC at time t_p is defined as $PFC(Fd, w, n, t_p)$ for a group of points $P_{i1}, P_{i2}, P_{i3}, \ldots, P_{in}$ with time stamps $t_{i1}, t_{i2}, t_{i3}, \ldots, t_{in}$. w is the weight and $\beta.\mu < w = \sum_{j=1}^{n} f(t_p - t_{ij}) < \mu, 0<\beta<1$ is the parameter to determine the threshold of outlier relative to progressive fractal cluster.
- *Definition 5*: *OFC*. An at time to for a group of points $P_{i1}, P_{i2}, P_{i3}, \ldots, P_{in}$ with time stamps $t_{i1}, t_{i2}, t_{i3}, \ldots, t_{in}$ is defined as $OFC(Fd, w, n, t_o)$. Fd is the fractal dimension of the Outlier fractal cluster and w is the weight, $w=\sum_{j=1}^{n} f(t_o - t_{ij}) < \beta\mu$, n is the number of points in the cluster and $t_0=t_{i1}$ is the creation time of the Outlier Fractal cluster.

4. The FractStream Clustering Algorithm

The main idea behind FractStream, an incremental grid-based clustering technique, is to group points in a cluster in such a way that none of the points in the cluster changes the cluster's fractal dimension radically making it more self-similar. After initializing a set of clusters, our algorithm incrementally adds points to the existing clusters, if *RCFD* is within a threshold else create a new cluster. To reduce the search for clusters in which new point fits, a concept of prioritizing the clusters is developed by introducing a concept of CFC, PFC, OFC. The algorithm for clustering the data stream is divided into two phases- the online phase and the offline phase. In the online phase authors present two steps, Initialization step and incremental step. Initialization step finds core clusters and in the Incremental step online clustering is performed based on *RCFD*. Weights of the clusters are calculated and updated, and in the offline phase the clustering is performed as demanded by the user. The basic algorithm steps are based on DenStream [8] .The time interval T_p for checking whether any OFC has become PFC or PFC has been decayed to an OFC, or removal of an OFC that cannot grow to PFC anymore, is also derived from [8] and is given by:

$$T_p=\frac{1}{\lambda}log(\frac{\beta\mu}{\beta\mu-1}) \tag{5}$$

CFC_i, PFC_i and OFC_i are periodically checked for their weights at time interval T_p. OFC_i is considered as real outlier and removed from the outlier buffer if its weight is less than a lower limit of ξ [8] given by

$$\xi(t_c, t_0) = \frac{2^{-\lambda(t_c - t_0 + T_p)} - 1}{2^{-\lambda T_p} - 1} \quad (6)$$

Where t_c is the current time and t_o is the creation time of the OFC_i. All the three categories of clusters are maintained in a separate memory space. This is based on the observation that most of the new points merge to existing clusters, and therefore can be absorbed by existing *PFC* or get created as new outlier clusters. Clusters with less weight are treated as outliers. Algorithm for initialization step is described below.

4.1. Algorithm Initialization Step

In clustering algorithms the quality of initial clusters is extremely important, and has direct effect on the final clustering quality. Some initial clusters are necessary to apply the main concepts of the proposed technique i.e., incremental addition of points to the clusters, based on how they affect the clusters fractal dimension. In other words, we need to "bootstrap" our algorithm via an initialization procedure that finds a set of clusters each with sufficient points, so as to find their fractal dimensions. Out of many choices for the initialization step, we present the result of a traditional distance based procedure for finding the initial clusters.The algorithm builds clusters by considering a random point from unclustered initial set of points and recursively finds the nearest neighbour within a given distance threshold κ. The neighbour is added into the cluster if the distance is within the threshold else a new cluster is formed and the search continues in depth-first fashion, until no more points can be added to clusters.

Algorithm 1: To find initial core clusters.

Given an initial set of points $P_{i1}, P_{i2}, P_{i3}, \ldots, P_{in}$ that fit in main memory and a distance threshold κ (Initially $\kappa = \kappa_0$)
Mark all points as unclustered and make k=0;
Choose a point P (At random) out of set of unclustered points
Mark P as belonging to cluster C_k
Starting at P and in a recursive depth-first fashion. Choose next point P' close to P based on Euclidian distance
If distance between the |P-P'|< κ merge P' in C_k
Else k=k+1
create a new cluster C_k with point p in C_k
Repeat the process until all points are clustered

4.2. Incremental Merging Using a Relative Change in Fractal Dimension

Initialization step generates k number of Core Fractal Cluster $\{C_1, C_2, \ldots, C_k\}$ and are represented as multi-layered grid. Let $Fd(C_i)$ be the fractal dimension of the i^{th} cluster C_i. The incremental step brings a new set of points to main memory and proceeds to take a basic window of data points and add them to each cluster, computing its new fractal dimension. Suitable cluster is found to merge the points by computing the minimal fractal impact, i.e., the minimal *RCFD*. When a new basic window of points arrives, the procedure of online clustering is described in Algorithm 2.

Algorithm 2: onlineMerging($p_1, p_2, \ldots, p_k$).

Input: Basic Window of data points ($p_1,p_2,\ldots p_k$)
Output : Merge status of data points ($p_1,p_2,\ldots p_k$)
Begin
Normalize Data points
While stream not end.
For all Core Fractal Clusters, CFC_i
Let $CFC_i' = CFC_i \cup \{p_1,p_2,\ldots p_k\}$
Compute Fd(CFCi')
If $|Fd(CFC_i')-Fd(CFC_i)|/Fd(CFC_i) < €$
Then merge $\{p_1,p_2,\ldots p_k\}$ to CFC_i.
Update weight w.
else
For all Progressive Fractal Clusters PFC_i
Let $PFC_i' = PFC_i \cup \{p_1,p_2,\ldots p_k\}$
Compute $Fd(PFC_i')$
If $|Fd(PFC_i')-Fd(PFCi)|/Fd(PFC_i) < €$
Then merge P to PFC_i;
Update weight w.
else
For all Outlier Fractal Clusters OFC_i
Let $OFC_i' = OFC_i \cup \{p_1,p_2,\ldots p_k\}$
Compute $Fd(OFC_i')$
If $|Fd(OFC_i')-Fd(OFC_i)|/Fd(OFC_i) < €$
Then merge P to OFC_i;
Update weight w.
If w_o (new weight of OFC_i) > $\beta\mu$
Then remove OFC_i from outlier hash table and create a new progressive Fractal Cluster by PFC_i.
end if
else
Create a new Outlier Fractal Cluster with $\{p_1,p_2,\ldots p_k\}$ and insert it into outlier fractal clusters hash table.
end if.
end if

Select the initial core clusters found from the initialization step and use the algorithm in [4] to find the initial fractal dimensions of *CFC*.

1. At first, we try to insert the points into all *CFC*. We compute the relative change in the fractal dimension, *RCFD*. Insert the points into that cluster whose *RCFD* change is within a minimal threshold and remove the points from rest of the *CFC*. If w is below μ and above $\beta\mu$ it means that CFC_i has become a progressive fractal cluster. Therefore, we remove CFC_i from the core fractal buffer and create a new progressive fractal cluster by PFC_i.
2. Else, we try to insert the points into all *PFC*. We compute *RCFD*. Insert the points into that cluster whose *RCFD* change is within a minimal threshold and remove the points from rest of the *PFC* updating the weight of the cluster.
3. Else, we try to merge points into all *OFC* and compute *RCFD*. Insert the points into *OFC* whose *RCFD* is within the minimum threshold and remove the points from rest of *OFC*. And then, we check w the new weight of *OFC*. If w is above $\beta\mu$ it means that OFC_i has grown into a progressive Fractal cluster. Therefore, we remove OFC_i from the outlier-buffer and create a new Progressive Fractal Cluster by PFC_i.

4. Otherwise, we create a new outlier fractal cluster OFC_i by basic window of points and insert the points into the outlier-buffer. This is because these points do not naturally fit into any existing fractal clusters. These points may be an outlier or the seed of a new cluster.

The weight of all the clusters needs to be checked periodically, because if no points are added to the clusters then the weight reduces as time passes on. If the weight of PFC_i is below $\beta\mu$, then it is no more progressive and should be deleted to release the memory space for new *PFC's*. The time span at which the weight of the clusters checked is T_p [8]. As data streams advances the number of Outlier fractal Clusters also increases. So *OFC*'s are to be restored which are potential to grow into *PFC* deleting the real outlier. The lower limit weight of *OFC* is defined as ξ [8] which is a function of t_c (i.e., current time) and t_o (i.e., the creation time of the *OFC*. t_o is maintained in the t field of the *OFC*. The detailed procedure is described in Algorithm 3.

Algorithm 3: FractStream (D_s, €, μ, β, λ).

$Tp=(1/\lambda)\{\log(\beta\mu/\beta\mu-1)\}$
For all points (p1,p2,…pk) from the data stream D_s,
onlineMerging(p1,p2,…pk)
if($t\bmod T_p$)=0 then
for each Core Fractal Fluster CFC_i
if wc(the weight of CFC_i) <μ then
delete CFC_i from core fractal cluster hash table and
create a new progressive fractal cluster PFC_i
end if
end for
for each Progressive Fractal Cluster PFC_i
if w_p(the weight of PFC_i)<βμ then
delete PFC_i and place it in a file
end if .
end for.
for each Outlier Fractal Cluster OFC_i
$\xi = 2^{-\lambda(t-to+Tp)}-1/ (2^{-\lambda Tp}-1)$
if w_O(the weight of OFC_i)< ξ then
delete OFC_i and place it in a file
end if
end for
end if.
If a clustering request arrives then generating clusters
end if.

5. Experimental Evaluation

This section concentrates on presenting the effectiveness of FractStream and a comparative evaluation with DenStream on real life data sets. FractStream algorithm is been implemented in Java and all the experiments have been performed on an Intel(R) Core(TM)i7-3630QM having 8GB of memory. The parameters used for FractStream and DenStream are the same, that is initial number of points=1000, stream speed v=1000, decay factor λ=0.25, minimum number of points necessary to create a Core Fractal cluster μ=10, outlier threshold β=0.2, the number of grid layers is 10 and the time span of sliding window is one second. We mean horizon (or window) as the number of time steps from the current time considered while running the clustering algorithms. Thus, for example, if the horizon is 10, it means that the last 10 blocks of data are clustered.The two real data sets used for experimental evaluation are the KDD Cup 1999 data set and the forest cover type data set.

The KDD CUP 99 data set consists of 10% of original dataset that is approximately 494,020 single connection vectors each of which contains 41 features and is labelled with exact one specific attack type i.e., either normal or an attack. Each vector is labelled as either normal or an attack, with exactly one specific attack type. Deviations from ‘normal behaviour’, everything that is not ‘normal’, are considered attacks. Attacks labelled as normal are records with normal behaviour. There are four main categories of attacks namely DOS, U2R, R2L and PROBE where each of these categories has subcategories.

The dataset consisted of 494,020 records, among which 97,277 (19.69%) were 'normal', 391,458(79.24%) DOS, 4,107 (0.83%) Probe, 1,126 (0.23%) R2L and 52 (0.01%) U2R attacks.

The forest cover type data set is the prevision forest cover type from cartographic variables only (no remotely sensed data), made available from the Remote Sensing and GIS Program department of Forest Sciences College of Natural Resources, Colorado State University Fort Collins. The area includes four wilderness areas located in the Roosevelt National Forest of northern Colorado. The data set is composed by 581,012 instances (observations) represented by 54 geological and geographical attributes describing the environment in which trees were observed. Both data sets have been transformed into data streams by taking the input order as the streaming order. When using the first initialization algorithm, our observations reveal that initially 5 clusters are formed starting with 1000 initial points. The core clusters are evaluated using Davies-Bouldin index and silhouette index to ensure the validity of the core clusters. The best values observed for the clusters are, the Davies-Bouldin index is 1.4556 and silhouette index=0.92 at a threshold value of 0.23.These values are listed in the following Table 1.

Table 1. Silhouette index and Davies-Bouldin index for different threshold value.

Threshold Value κ	No. of Clusters	Silhouette Index	Davies-Bouldin index
0.05	10	0.45	2.5454
0.1	7	0.56	2.2036
0.2	6	0.86	1.8514
0.22	5	0.92	1.4556
0.24	5	0.93	1.4651
0.3	4	0.77	1.9622

This evolution of data changes the behaviour of clusters over time when online clustering is performed by considering a window of data points with stream size of 1000. We can segment it into intervals as follows, shown in Figure 2.

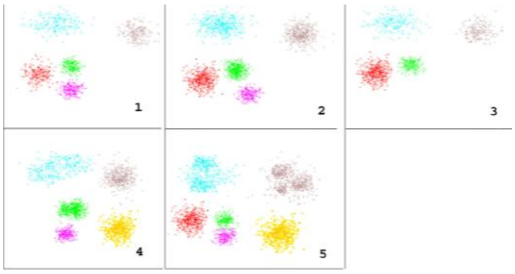

Figure 2. The cluster evolution.

1. Initially, there are 5 clusters in a steady state called CFC. Data point from 1 to 1000.
2. After starting the stream with sliding window of 1000 data points 257 data points are added to cluster 2 , 383 are added to cluster 0 and 360 to cluster 4. Cluster 0, cluster 2, and cluster 4 becomes progressive clusters.
3. Cluster 1 becomes OFC during the stream process from 4000 to 5000.
4. The rest of the clusters still continue to be progressive fractal clusters and a new cluster is formed. Data point from 5000 to 6000.
5. 324 data points are added to cluster 4,145 are added to cluster 2,185 to cluster 1 and 346 to cluster 0.

The clustering quality is evaluated by the average purity [8] of clusters which is defined as follows:

$$purity = \frac{1}{k}\left(\sum_{i=1}^{k}\frac{c_i^d}{c_i}\times 100\%\right) \tag{7}$$

Where K denotes the number of clusters. C_i^d denotes the number of points with the dominant class label in cluster i. C_i denotes the number of points in cluster i. Intuitively, the purity measures the purity of the clusters with respect to the true cluster (class) labels that are known for our data sets. Since the weight of points fades out gradually, we compute the purity by only the points arriving in a pre-defined horizon H (or window) from current time. Our experimental results show that the purity results are insensitive to the horizon.

On the network intrusion data set, the stream speed was set to 1000 points per time unit. Since the network intrusion data set was already used in [8], the same parameter settings are chosen for FractStream. For the evaluation, measurements at timestamps where some attacks exist were selected. The data recordings at timestamp 100 and all the recordings within the horizon 5 were only attacks of the type smurf. At this time unit the algorithm could achieve 100% purity. The attacks that appeared within horizon H=5 in different timestamps are listed in the Table 2 below.

Table 2. Labels of KDD cup data stream within the horizon H=5, stream speed=1000.

Normal or attack Type	Objects Within Horizon H = 5 at Time Unit			
	150	250	300	350
Normal	4014	4117	892	406
Satan	370	0	0	0
Teardrop	91	0	365	0
Smurf	144	99	289	2988
Ipsweep	48	202	0	0
Loadmodule	10	0	0	1
Warezclient	307	0	0	0
Neptune	6	528	3280	1603
Pod	10	50	49	0
Portsweep	0	2	125	1
Land	0	2	0	1
Sum	5000	5000	5000	5000

The comparison between FractStream and DenStream on the network intrusion data set is shown in Figures 3 and 4. The results have been computed by setting the horizon to 1 and 5, whereas the stream speed is 1000. We can clearly see the very high clustering quality achieved by FractStream on this data set. As time passes, cluster results became better, because, in the calculation of fractal dimension, all data points are contained in the lowest level grid which is the optimal situation. In other way, the accumulation of data points is beneficial to the accuracy of the calculating results. The larger the amount of data, the more accurate is the self-similarity. Therefore, FractStream algorithm can achieve better clustering quality and this is the best quality obtained in the clustering process.

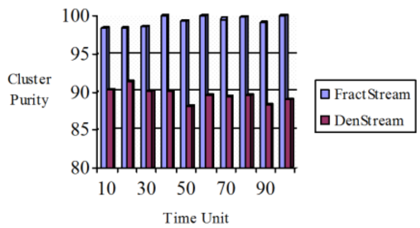

Figure 3. Clustering quality (network intrusion data set, horizon=1, stream speed=1000).

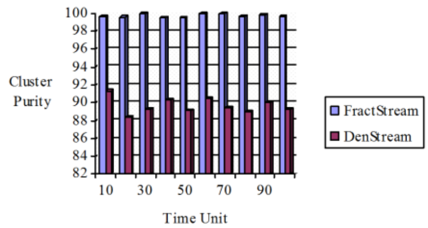

Figure 4. Clustering quality (network intrusion data set, horizon=5, stream speed=1000).

Cluster purity is above 98%, for all the time units, and reaches 100% at time units 40, 60, 100. Purity is above 99% when the horizon is set to 5. On this data set FractStream always outperforms DenStream, which obtains a maximum value of purity of 91% for both the horizons. Figures 5 and 6 reports the same experiments executed on the Covertype data set. Also for this data set FractStream outperforms DenStream with respect to the cluster purity.

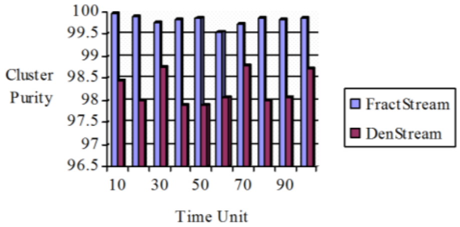

Figure 5. Clustering quality(forest cover type data set for stream speed=1000 and horizon=1).

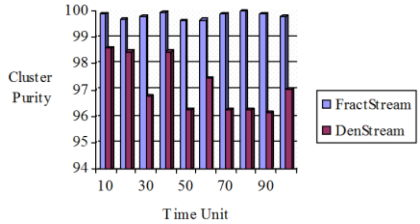

Figure 6. Clustering quality(the forest cover type data set for stream speed=1000 and horizon=5).

5.1. Time Complexity Analysis

Calculating fractal dimension is the major task in FractStream. The grids are scanned only once to get the statistics of bottom layer. Next layer statistics is obtained by just summing up the bottom layers statistics reducing the search space by $1/2^E$. FractStream runtime complexity is upper-bounded by $EN+EN\Sigma_j 1/2E^{(j-1)}$ for $j=1$ *to* $|R|$, thus it is $O(eEN)$, with $e=1+2^E/2^{(E-1)}$. The resulting algorithm scales independently of $|R|$. For example with 2*D* data, *e* equals to 2.333 while $|R|$, the number of levels in the multi layered grid which is 10. For higher dimensionalities, the runtime difference between FractStream and DenStream increases. Figure 7 shows the execution time for Network Intrusion data.

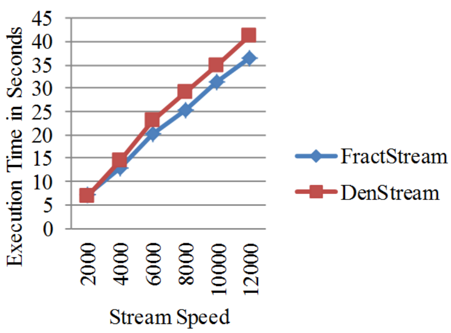

Figure 7. Execution time for increasing stream length on the Network Intrusion data set.

5.2. Space Complexity Analysis

Space complexity is mainly due to storage of grids. The storage space is independent of the dimensions because, in nested grid structure, only statistics of the data points are stored. The space complexity of nested grid structure is *O(N)* since the bottom layer grid consist of points whose number is same as the data set number, where *N* is the number of data points. The space complexity of the *FractStream* is *O(N·n)*, the *n* is the number of clusters since one grid is constructed for each cluster.

6. Conclusions and Future Work

In this paper, authors proposed a Grid based clustering algorithm which can find the clusters with arbitrary shapes. The weighting scheme used in this algorithm enables it to assign weights to the clusters, based on their priority and time in the clustering process. Experimental results show that this algorithm is more effective than DenStream and achieves higher cluster quality. FractStream, proposed in this paper is an effective and efficient method for clustering in an evolving data stream using correlation fractal dimension. The structures of core fractal clusters, progressive fractal clusters and outlier fractal clusters maintain sufficient information for clustering and reduce the search complexity thereby reducing the execution time. The pruning strategy designed to limit the memory consumption guarantee precision. Our experimental performance evaluation over a number of real data sets demonstrates the effectiveness and efficiency of FractStream in discovering clusters of arbitrary shape in data streams. Future work aims at extending the method to a distributed framework, more perfect for real life applications.

References

[1] Aggarwal C., Han J., Yu P., and Wang J., "A Framework for Clustering Evolving Data Streams," *in Proceedings of the 29th Very Large Databases Conference*, Berlin, pp. 81-92, 2003.

[2] Aggarwal C., Han J., Yu P., and Wang J., "A Framework for Projected Clustering of High Dimensional Data Stream," *in Proceedings of the 13th International conference on Very Large Data Bases*, Toronto, pp. 852-863, 2004.

[3] Aggarwal C., Han J., Yu P., and Wang J., "On High Dimensional Projected Clustering of Data Streams," *Data Mining and Knowledge Discovery*, vol. 10, no. 3, pp. 251-273, 2005.

[4] Ali S. and Madani S., "Distributed Grid Based Robust Clustering Protocol for Mobile Sensor Networks," *The International Arab Journal of Information Technology*, vol. 8, no. 4, pp. 414-421, 2011.

[5] Barbara D. and Chen P., *Fractal Mining Self Similarity based Clustering and its Applications*, Springer, 2010.

[6] Barbara D., "Requirements for Clustering Data Streams," *ACM SIGKDD Explorations*, vol. 3, no. 2, pp. 23-27, 2002.

[7] Belussi A. and Faloutsos C., "Estimating the

Selectivity of Spatial Queries Using the Correlation Fractal Dimension," *in Proceedings of the 21th International Conference on Very Large*, San Francisc, pp. 299-310, 1995.

[8] Cao F., Ester M., Qian W., and Zhou A., "Density-based Clustering Over Evolving Data Stream with Noise," *in Proceedings of the 6th SIAM International Conference on Data Mining*, Bethesda, pp. 326-337, 2006.

[9] Chen Y. and Tu L., "Density-Based Clustering for Real-Time Stream Data," *in Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Jose pp. 133-142, 2007.

[10] Ester M., Kriegel H., Sander J., and Xu X., "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," *in Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining*, Portland, pp. 226-231, 1996.

[11] Gama J. and Rodrigues P., *An Overview on Mining Data Streams*, Springer-Verlag Berlin Heidelberg, 2009.

[12] Guha S., Meyerson A., Mishra N., and O'Callaghan R., "Clustering Data Streams: Theory and Practice," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 3 pp. 515-528, 2003.

[13] Guiling Li., Wang Y., Gu S., and Zhu X., "Fractal-Based Algorithm for Anomaly Pattern Discovery on Time Series Stream," *Journal of Convergence Information Technology*, vol. 6, no. 3, pp. 181-187, 2011.

[14] Han J. and Kamber M., *Data Mining: Concepts and Techniques (Second Edition)*, Elsevier, 2006.

[15] Khalilian M. and Mustapha N., "Data stream Clustering: Challenges and Issues," *in Proceedings of International Multi Conference of Engineers and Computer Scientists*, Hong Kong, pp. 17-19, 2010.

[16] Lin J. and Lin H., "A Density-Based Clustering Over Evolving Heterogeneous Data stream," *in Proceedings of International Colloquium on Computing Communication Control and Management*, Sanya, pp. 275-277, 2009.

[17] Lui L., Huang H., Guo Y., and Chen F., "rDenStream, A Clustering Algorithm over an Evolving Data Stream," *in Proceedings of International Conference on Information Engineering and Computer Science*, Wuhan, pp. 1-4, 2009.

[18] O'Callaghan L., Motwani R., Mishra N., Meyerson A., and Guha S., "Streaming Data Algorithms for High-Guality Clustering," *in Proceedings of 18th International Conference on Data Engineering*, San Jose, pp. 685-694, *2002*.

[19] Osama A., "Comparisons Between Data Clustering Algorithms," *The International Arab Journal of Information Technology*, vol. 5, no. 3, pp. 320-325, 2008.

[20] Qian Q., Chao-Jie X., and Rui Z., "Grid-Based Data Stream Clustering for Intrusion Detection," *International Journal of Network Security*, vol. 15, no. 1, pp. 1-8, 2013.

[21] Ren J., Cai B., and Hu C., "Clustering Over Data Streams Based on Grid Density and Index Tree," *Journal of Convergence Information Technology*, vol. 6, no. 1, pp. 83-93, 2011.

[22] Ren J., Li L., and Hu C., "A Weighted Subspace Clustering Algorithm in High-Dimensional Data Streams," *in Proceedings of 4th International Conference on Innovative Computing, Information and Control*, Kaohsiung, pp. 631-634, 2009.

[23] Traina C., Traina A., and Faloutsos C., "Fast Feature Selection Using Fractal Dimension-Ten Years Later," *Journal of Information and Data Management*, vol. 1, no. 1, pp. 17-20, 2010.

[24] Udommanetanakit K., Rakthanmanon T., and Waiyamai k., "E-Stream: Evolution based Technique for Stream Clustering," *in Proceedings of International Conference on Advanced Data Mining and Applications*, Harbin, pp. 605-6015, 2007.

[25] Yarlagadda A., Murthy J., and KrishnaPrasad M., "Estimating Correlation Dimension using Multi Layered Grid and Damped Window Model Over Data Streams," *Procedia Technology*, vol. 10, pp. 797-804, 2013.

[26] Zhu Y. and Shasha D., "StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time," *in Proceedings of the 28th International Conference on Very Large Data Bases*, Hong Kong, pp. 358-369, 2002.

Anuradha Yarlagadda received her Master's in Computer Science and Engineering from Visvesvaraya Technological University, India, and is pursuing her Doctoral degree at Jawaharlal Nehru Technological University Hyderabad, India. Her research interest is data warehousing and mining.

Murthy Jonnalagedda is currently, a Professor of the Department of Computer Science and Engineering, University College of Engineering Kakinada, JNTUK, Andhra Pradesh. He received his B.Tech degree from JNTU College of Engineering, Kakinada, M.Tech degree from IIT Kharagpur and Ph.D. degree from JNTU, Kakinada. His research interests include data warehousing and mining, data bases, big data analytics and high performance computing.

Krishna Munaga is currently, an Associate Professor of the Department of Computer Science and Engineering, University College of Engineering Kakinada, JNTUK, Andhra Pradesh. He received his BE degree from Osmania University, Hyderabad, M.Tech degree and Ph.D. in Computer Science and Engineering and from JNTU, Hyderabad. He successfully completed a two-year MIUR fellowship at the University of Udine, Udine, Italy. His research interests include data mining, big data analytics and high performance computing.