Clustering over Data Streams Based on Grid Density and Index Tree

[1]Jiadong Ren, [2]Binlei Cai , [3]Changzhen Hu
[1,]*College of Information Science and Engineering, Yanshan University, Qinhuangdao, 066004, P.R.China , jdren@ysu.edu.cn*
[*2,]*College of Information Science and Engineering, Yanshan University, Qinhuangdao, 066004, P.R.China , adsert@126.com*
[3,]*School of Computer Science and Technology, Beijing Institute of Technology Beijing City, 100081, P.R.China, cz_hu@sina.com*
doi:10.4156/jcit.vol6.issue1.11

Abstract

Most existing grid-based stream clustering algorithms are low efficient for high-dimensional data streams due to a large number of cells, and can not handle noise points effectively. In this paper, we propose a novel approach PKS-Stream for clustering data streams, which is based on grid density and index tree Pks-tree. The new index structure Pks-tree is introduced to store the non-empty grid cells, which aims to improve the efficiency of storage and indexing. Simultaneously, we define a novel time-based density threshold function to remove the noise points in real time. Based on Pks-tree, the data stream is clustered by grid density in the initial stage. With new data records arriving, the novel pruning strategy is adopted to periodically detect and remove noise points. Also, the generated clusters are dynamically adjusted. The experimental results show that PKS-Stream has better clustering quality and scalability.

Keywords*: Data Streams, Clustering, Grid Density, Index Tree*

1. Introduction

In recent years, a large amount of data streams have been generated. Discovering patterns hidden in data streams is a great challenge for clustering. Clustering based on data streams has been becoming a hot topic [1].

Local-Search [2] was brought up on the idea of divide and conquer. Based on k-means, it uses a continuous iterative process to discover clusters in data streams. Callaghan also proposed Stream [3], which is on the basis of Local-Search. In Stream, the value of *k* in k-means is more flexible, not a fixed value. However, such algorithms can only provide a description for the current data streams, and can not capture the evolution of data streams. Clustering results may be controlled by outdated data records. Aggarwal presented a framework for clustering evolving data streams, named as CluStream [4], the clustering process is divided into an online component and an offline component. The online component periodically stores summary information by the pyramidal time window, while the offline component analyzes the data streams over different horizons by using summary information which is stored in online component. But CluStream needs to predefine the number of micro-clusters. Also, it can't accurately find clusters with arbitrary shape in data streams. HPStream [5] introduces the concept of projected cluster to cluster high dimensional data streams. It is high efficient for high dimensional data streams. But it cannot discover clusters of arbitrary shape in data streams. CluWin [6] is a clustering algorithm based on sliding window. Two types of exponential histogram of cluster features, false positive and false negative, are introduced into it. Based on the actual application, false positive or false negative as summary structure is used to save the data in the current window. E-Stream [7] improves existing stream clustering algorithms by supporting 5 evolutions with a new suitable cluster representation and a distance function. But the algorithm requires the user to enter the maximum number of clusters. HCluWin [8] was proposed by Ren, which introduced a Heterogeneous Temporal Cluster Feature to monitor the distribution statistics of heterogeneous data points.

DenStream [9] and D-Stream [10] can discover clusters of arbitrary shape in data streams. DenStream is an extension of DBSCAN, based on the damped window model, which is capable of discovering clusters with arbitrary shape and emphasizes on detecting noise points. However, the

algorithm is sensitive to the density threshold parameter since it uses the absolute density threshold. D-Stream is based on grid density, and borrows the two-tier framework of Clustream. The online component of D-Stream continuously reads new data records, mapped the data into corresponding grids, and updates the characteristic vector of the density grid. In the offline component, the noise points are periodically removed, and clusters are adjusted. It can discover clusters with arbitrary shape and handle noise points. However, in the grid-based algorithms, the number of cells will increase exponentially with the dimensionality, so it has low efficient for high dimensional data stream.

In order to improve the efficiency of grid-based clustering algorithm, we introduce the index structure Pks-tree to store and index the non-empty grid cells. Moreover, a new noise point disposal strategy will be designed to distinguish and remove noise points, which is on the basis of a density threshold function.

The remainder of the paper is organized as follows. In section 2, we describe the definition of problems. Section 3 introduces the PKS-Stream algorithm. The algorithmic details and theoretical analysis also are discussed in this section. In section 4, we conduct experimental study of PKS-Stream, and compare PKS-Stream with CluStream on synthetic and real-world data sets. Section 5 makes a summary.

2. Problem Definitions

In PKS-Stream, we partition the *k*-dimensional space *S* into grids. For each data record *x*, we assign it a density coefficient which decreases with *x* ages. If *x* arrives at time t_0, its density coefficient $d(x,t)$ at time t is $d(x,t) = 2^{-\lambda(t-t_0)}$. In this paper, we partition the space into grid cells at various granularities. For example, if we divide each dimension into 2 parts, any grid cell g can be further partitioned into 2^k sub-cells. We introduce the index structure Pks-tree to mirror the space partitioning so that each tree node corresponds to a grid cell. Parameter H is used to control the levels of partitioning. The grid cell at level i is denoted as $(g_{ij_1}, g_{ij_2}, .. g_{ij_m} .. g_{ij_k})$, where $0 < i \le H$, g_{ij_m} is the number of the interval in the m^{th} dimension, $g_{ij_m} = 1,2,..., 2^{i-1}$. The grid cell at H level of partitioning, we call it the minimum cell, and denote it as g_H.

Definition2.1.(Minimum cell Density) The density of the minimum cell g_H is $d(g_H,t) = \sum_{x \in E(g_H,t)} d(x,t)$, where $E(g_H,t)$ is the set of data that are placed to g_H at or before time t.

If $d(g_H,t) \ge \mu$, μ is a density threshold, we call g_H a dense grid, if $d(g_H,t) < \mu$, we call g_H a sparse grid.

Definition2.2. (Neighboring Grids) Two grids $g_1 = (g^1_{ij_1}, g^1_{ij_2}, ..., g^1_{ij_k})$, $g_2 = (g^2_{ij_1}, g^2_{ij_2}, ..., g^2_{ij_k})$, if there exists m, $1 \le m \le k$, such that: (1) $g^1_{ij_l} = g^2_{ij_l}, l = 1,2,..., m-1, m+1, ...k$; (2)$| g^1_{ij_m} - g^2_{ij_m} | = 1$, then g_1 and g_2 are Neighboring grids in the m^{th} dimension.

Definition2.3. (Minimum cell Characteristic Vector) The characteristic vector of minimum cell g_H is a tuple (t_g , t_m, d, $Label$), where t_g is the last time when g is updated, t_m is the last time when g is deleted, d is the grid density after the last update, Label is the cluster label of the grid cell g_H.

Definition2.4. (*K-cover* Grid Cell) A grid cell g is *K-cover* (*K*>1), if
(1) g is a minimum cell, or
(2) There does not exist *K*-1 (or less *K*-1) *K*-cover grid cells $g_1, g_2, ..., g_{K-1} \subset g$, which make $\forall d \in g \Rightarrow d \in g_1 \cup g_2 \cup ..... g_{K-1}$.

3. Clustering Algorithm Based on Grid Density and Pks-tree

3.1. Index Structure Pks-tree

In the grid-based clustering algorithm, there are a lot of empty grid cells, which are meaningless for mining data streams. Especially for high-dimensional data streams, a large amount of empty grid cells are generated. If all of grid cells are stored, clustering process is easy to be executed since the relative position relationships among cells are kept. However, it will lead to large memory cost. If only storing the non-empty grid cells, the utilization rate of memory is improved. But the relative position relationships among grid cells is eliminated, it results in high computational complexity. For example, if the non-empty grid cells are stored in a linked list, a neighbor search will traverse the whole linked list. In order to solve the problem, the index structure Pks-tree is proposed. Pks-tree not only stores non-empty grid cells, but also keeps their relative position relationships, which improves the efficiency of clustering.

Definition3.1. (Pks-tree) Given the parameter H , and a Pks-tree of rank K (K>1) is defined as follows,

(1) The root of Pks-tree at level 1 contains the over synopsis information of space S .

(2) Expect the root, every node in the Pks-tree is corresponding to a K-cover Grid Cell and stores synopsis information of S at i level(or granularity) , which is one-to-one.

(3) For any two nodes g_1 and g_2 in the Pks-tree, g_1 is a child of g_2 , if

(a) g_1 is a proper sub-cell of g_2 that $g_1 \subset g_2$,and

(b) there doesn't exist a node g_3 in the Pks-tree which makes $g_1 \subset g_3$ and $g_3 \subset g_2$.

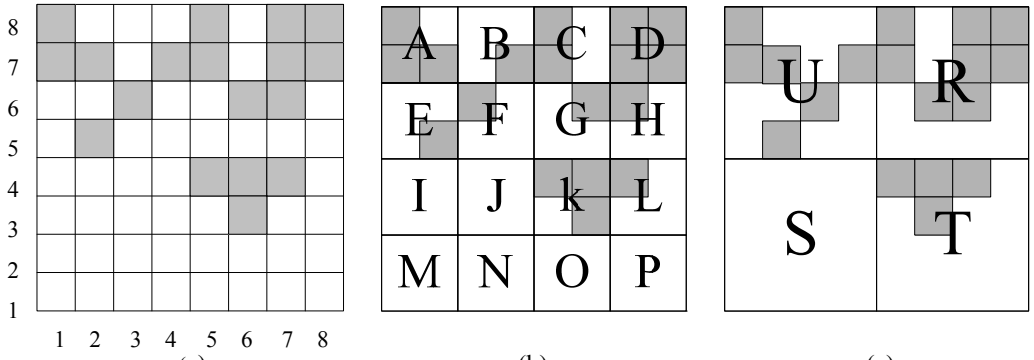

Figure 1. The partition of the space and the possible intermediate nodes

Figure 1 illustrates how the two-dimensional space is partitioned into three levels. If we set K=3, each non-empty grid cells in Figure 1(a) is a *K-cover* Grid Cell since they are minimum cells. In Figure1(b), the grid cell A is a *K-cover* Grid Cell. However, the grid cell *C* can be covered by 2 (2<K) *K-cover* Grid Cells (5,7) and (5,8), so it is not a *K-cover* Grid Cell. Figure 2 is a Pks-tree of rank 3, which is built on the two-dimensional data set of Figure 1.

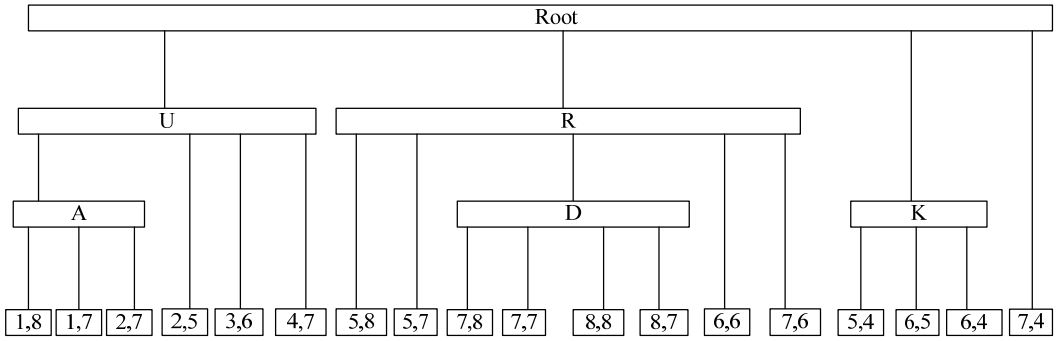

Figure 2. A Pks-tree of rank 3 built on Figure1

Pks-tree is not a balanced tree. The value of *K* is more larger, the height of the tree is more lower. If there exist *N* non-empty minimum cells in the Pks-tree, the average height of the tree is $O(\log^{N})$. Since the computational complexity of insertion, deletion, and search are linearly proportional to the height of the Pks-tree, the average computational complexity of Pks-tree is $O(\log^{N})$, and the worse case complexity is $O(N)$. Moreover, the concept of *K-cover* Grid Cell is introduced to remove the empty grid cells. It also avoids the case that the grid cell containing few points occupys a long path. All of these reduce the computational complexity.

3.2. Analyzing and Determining Detecting Period *per*

Data streams are dynamically changing. Thus, after a period of time, the density of each grid cell should be detected and the clusters should be adjusted. However, how to determine the period is the key problem. If the period is too short, it will result in frequent computation. If the period is too long, it will lead to miss some changes of data streams. We regard the minimum time needed for a dense grid to degenerate to a sparse grid as time period *per*, which ensures that the detection is frequent enough to capture the density changes of any grid cell.

Theorem3.1. The minimum time needed for a dense grid to degenerate to a sparse grid is

$$per = \frac{1}{\lambda}\log^{\frac{\mu}{\mu-1}} \tag{1}$$

Where μ (μ >0) is a density threshold, and λ >0.

Proof: Let g be a dense grid, Δt is the minimum time needed for g to degenerate to a sparse grid.

$$w(g,t+\Delta t)+1 = 2^{-\lambda\Delta t}w(g,t)+1 = \mu, 2^{-\lambda\Delta t} = \frac{\mu-1}{w(g,t)} \text{ , } 2^{\lambda\Delta t} = \frac{w(g,t)}{\mu-1},$$

$$\Delta t = \frac{1}{\lambda}\log^{\frac{w(g,t)}{\mu-1}} \text{ , } per = \min \Delta t, \text{ due to } w(g,t) \geq \mu, per = \frac{1}{\lambda}\log^{\frac{\mu}{\mu-1}}.$$

□

3.3. Analyzing and Determining Density Threshold Function

There will inevitably be noise points in data streams. Some non-empty grids are made of noise points, so we call them scattered grids. Scattered grid makes no sense to the user, thus removing it from memory to release resources is an immediate work. But not all of the sparse grids are scattered grids. There are two reasons for a grid to be a sparse grid. The first cause is that very few data records are mapped to it, while the second is that many data records have been mapped to the grid but its density is reduced by the decay factor λ . The sparse grids in the former case are scattered grids and should be removed. But the sparse grids in the latter case should not be removed since they contain some data records and are possible upgraded to dense grids. So we define a density threshold function to recognize the scattered grids, as follows,

$$\rho(t_c,t_0) = \frac{2^{-\lambda(t_c-t_0+per)}-1}{2^{-\lambda per}-1} \tag{2}$$

Where t_c is the current time, t_0 is the last time when the grid inserted into Pks-tree.

The function has the following properties.

Property3.1. $\lim_{t_c\to\infty}\rho(t_c,t_0) = \mu$, and it is an increasing function.

Proof: (1) $\lim_{t_c\to\infty}\rho(t_c,t_0) = \frac{1}{1-2^{-\lambda per}} = \mu$

$$(2)\ \rho'(t_c,t_0)=\frac{1}{2^{-\lambda per}-1}((2^{-\lambda t_c})'\cdot 2^{-\lambda(per-t_0)})$$
$$=\frac{1}{2^{-\lambda per}-1}(2^{-\lambda t_c}\cdot\ln 2^{-\lambda}\cdot 2^{-\lambda(per-t_0)})$$
$$>0$$

□

$\rho(t_c,t_0)$ is used to recognize scattered grids from all sparse grids. The limit of $\rho(t_c,t_0)$ is μ, which ensures the removed grids to be sparse. The value of $\rho(t_c,t_0)$ is increasing with time, which makes the density of grids that received few new data for a long time less than $\rho(t_c,t_0)$ as soon as possible, so as to remove scattered grids in time.

Property3.2. If $d(g,t_c)<\rho(t_c,t_0)$, $d(g,t_c)$ is the density of g, then we have

$$\mu f(n\cdot per+t_c-t_0)+\rho(t_c,t_0)\le\mu(n=1,2,3.....). \quad (3)$$

Proof:

$$\mu f(n\cdot per+t_c-t_0)+\rho(t_c,t_0)$$
$$=\mu\cdot 2^{-\lambda(n\cdot per+t_c-t_0)}+\frac{2^{-(t_c-t_0+per)}-1}{2^{-\lambda\cdot per}-1}$$
$$=\frac{(2^{-\lambda\cdot per}-1)\cdot\mu\cdot 2^{-\lambda(n\cdot per+t_c-t_0)}+2^{-\lambda(t_c-t_0+per)}-1}{2^{-\lambda\cdot per}-1}$$
$$=\frac{-2^{-\lambda(n\cdot per+t_c-t_0)}+2^{-\lambda(t_c-t_0+per)}-1}{2^{-\lambda\cdot per}-1}$$
$$=\frac{2^{-\lambda(t_c-t_0+per)}-2^{-\lambda(n\cdot per+t_c-t_0)}-1}{2^{-\lambda\cdot per}-1}$$
$$\le\frac{-1}{2^{-\lambda\cdot per}-1}=\mu$$

Property3.2 illustrates that, if g is deleted as a scattered grid at time t_c since $d(g,t_c)<\rho(t_c,t_0)$, then even if all the previous data records in g have not been removed, it is still scattered. It ensures that the potential dense grids will not be deleted accidentally.

3.4. Clustering Algorithm PKS-Stream

In the PKS-Stream algorithm, the index structure Pks-tree is adopted to improve its efficiency. The new data records in the data stream are continuously read and mapped to the corresponding grid cells at all levels. If the grid cell corresponding to this data record exists, insert the data record into it. Otherwise, create a new grid cell. The density of each minimum cell is detected periodically to remove the noise points in real time. Meanwhile, based on the concept of *K-cover* Grid Cell, the construction of the Pks-tree is adjusted every *per* time steps. The Pks-tree building is introduced in Algorithm1.

Algorithm1.Create Pks-tree

Input: Data Stream $X =< x_1, x_2, ..., x_d >$
Output: Pks-tree
(1)Create the root node of Pks-tree, t_c =0;
(2) For the current object x^t in stream X
(3) For *i* from to *H*
(4) compute the coordinate $(g_{ij_1}, g_{ij_2}, ..g_{ij_m} ..g_{ij_k})$ of the cell g corresponding to x^t;
(5) if g exists, then insert x^t into it, go(3);
(6) else insert g to the corresponding level of Pks-tree, its par.childcount++, and insert x^t into it.
(7) End;
(8) $t_c = t_c + 1$;
(9) End;
(10) if (t_c mod *per* = =0)
(11) detect each leaf node of Pks-tree;
(12) if(g_H .dense< $\rho(t_c, t_0)$)
(13) delete g_H ;
(14) End if;
(15) adjust tree;
(16) End if ;

The following procedure is introduced to adjust Pks-tree:

g_m is a middle node of Pks-tree, and all children of g_m are in ChildrenSet. P is the parent of g_m .

(1) If |ChildrenSet|<*K*, we will make all nodes in ChildrenSet be children of P.
(2) If exist ($g' \subset g_m$ and g'_1,....., $g'_K \in$ ChildRenSet, such that g'_1,....., $g'_K \in g_m$), and not exist ($g'' \subset g'$ and g''_1,....., $g''_K \in$ ChildRenSet, such that g''_1,....., $g''_K \in g'$), then make g' be a child of g_m . For each node g in ChildrenSet, if $g \subset g'$, we will make g be a child of g' .

According to Algorithm 1, the Pks-tree will store and index non-empty grid cells. It not only improves the efficiency of memory, but also makes clustering very convenient. The average height of the tree is $O(\log^N)$ since the construction of the Pks-tree is adjusted periodically. Moreover, the density of each minimum cell is detected by the density threshold function every *per* time steps, which removes the noise points in real time and improves the clustering quality.

The clustering procedure on data stream is given in Algorithm 2. PKS-Stream clusters the minimum cells which located in the leaf-node level of the Pks-tree. It makes the neighboring dense grids with the same cluster label. The result of clustering is all of the dense minimum cells are marked by the label of clusters.

Algorithm 2.Clustering

Input: Pks-tree
Output: Clusters
(1) If (t_c mod *per* = =0)
(2) For each leaf node grid g_H of Pks-tree
(3) If(g_H is unmarked and g.dense $\geq \mu$)
(4) mark g_H with a new cluster label;
(5) For each neighbor $g_H^{'}$ of g_H
(6) If g' meets 1)neighboring with g 2) is unmarked
3) $g_H^{'}$ .dense $\geq \mu$
(7) mark $g_H^{'}$ with the cluster label of $g_H^{'}$;
(8) End;
(9) End ;
(10) End if ;
(11) If a clustering request arrives, then
(12) generating cluster;
(13)End if ;

In general, the finer the partition is, the higher clustering accuracy is, so we cluster the data stream based on the minimum cells of the leaf-node level. Moreover, in order to capture the dynamic evolution of data streams, the generated clusters are adjusted every *per* time steps.

4. Experimental Evaluation

In this section, we present an experimental evaluation of PKS-Stream, and compare it with CluStream. All the experiments are conducted on a PC with Intel(R) Core(TM)2 1.86GH CPU and 1GB RAM. We have implemented PKS-Stream in Visual C++ 6.0 and MATLAB 7.0, the parameters are as follows, $\mu = 2, \lambda = 0.25$, *H*=5. Two data sets, DataSet1, KDD-CUP-99, are used in the experiments. DataSet1 is a two-dimensional synthetic data set, and is composed of 100,000 data records. The KDD-CUP-99 Network Intrusion Detection data set is a real data sets. It contains five clusters and each connection record contains 42 attributes. All 34 continuous attributes out of the total 42 available attributes are used in the experiments.

4.1. Analysis of Evolving Date Stream

In order to validate the effectiveness of PKS-Stream on mining the dynamic characteristics of data streams, a synthetic data set DataSet1 is adopted.

The original distribution of the data in DataSet1 is shown in Figure 3, there are 4 clusters in the set, and some clusters have non-convex shape. Many noise points are scattered in DataSet1. In the first test, any data point that has not been generated is equally to be picked at each time. Noise points and data points from different clusters may alternately appear in the data stream. In this test, the final clustering result of PKS-Stream is shown in Figure 4. The clustering process of PKS-Stream is based on grid density, so it can correctly find four clusters. The noise points are removed effectively since the noise point disposal strategy is adopted, which is based on the density threshold function.

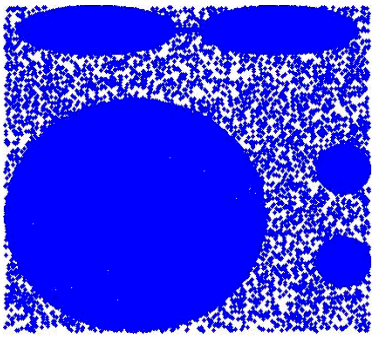

Figure 3. Original data distribution of DataSet1

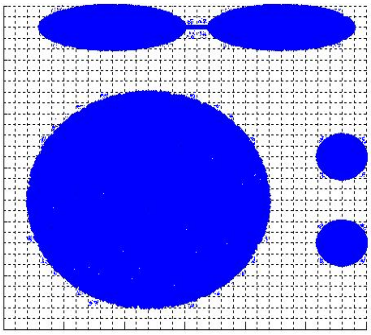

Figure 4. Final clustering results

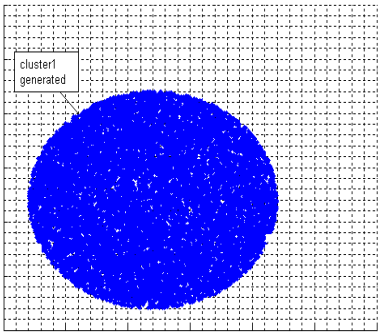

Figure 5. Clustering result at $t_1 = 15$

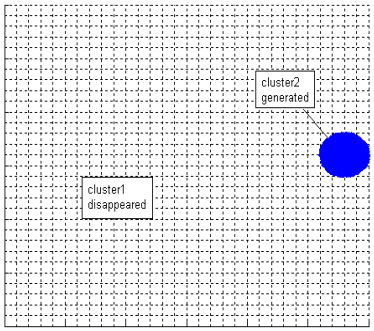

Figure6. Clustering result at $t_2 = 30$

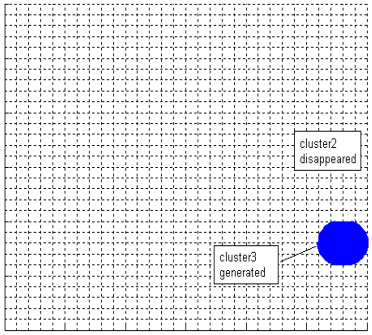

Figure7. Clustering result at $t_3 = 45$

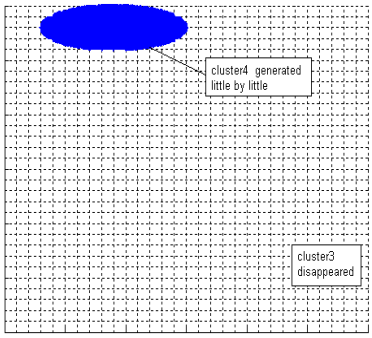

Figure 8. Clustering result at $t_4 = 60$

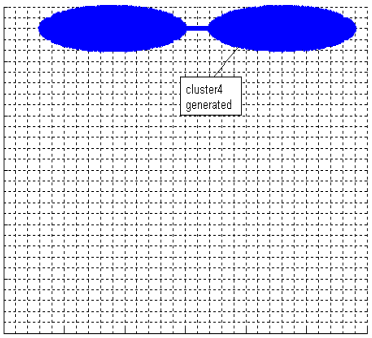

Figure 9. Clustering result at $t_5 = 75$

In the second test, we aim to show that PKS-Stream can mine the dynamic characteristics of data streams and remove noise points in time. According to the data distribution in DataSet1, when the data stream is generated, we order the four clusters, and they are generated sequentially. A number of noise points are mixed in the data stream, and the speed of the stream is 1000 points/s. The clustering results at time t_1=15(s), t_2=30(s), t_3=45(s), t_4=60(s), t_5=75(s), which are shown in Figure 5-9. According to the change of the density of the grid, the algorithm adjusts the generated clusters periodically, so it is able to capture the evolution of the data stream. Since the noise points disposal strategy based on density threshold unction is used, the noise points can be removed in real time.

Figure 5-9 also show that cluster boundary accuracy of PKS-Stream is lower than GDC-Stream [11]. But PKS-Stream adopts the index structure Pks-tree, which makes it has a higher efficiency.

4.2. Analyzing the Clustering Quality and Scalability

The clustering quality of the algorithm is measured by clustering purity. The comparison of clustering purity between PKS-Stream and CluStream on dataset1 is showed in Figure 10. At the beginning, the clustering purity of PKS-Stream and CluStream are high. However, the noise point disposal strategy based on density threshold function is adopted by PKS-Stream, so the grids generated by noise are distinguished from the non-empty grids, and deleted in real time. But CluStream lacks such method to distinguish the micro-cluster generated by noise points. So the clustering purity of PKS-Stream is higher than CluStream. As more and more data points arriving, the non-convex cluster appears. PKS-Stream is based on grid density, so it is capable to discover cluster with arbitrary shape. However, CluStream cluster the data basing on distance, it is inefficient for discovering the non-convex cluster. So PKS-Stream gets higher clustering purity than CluStream. For instance, at time 60, the non-convex cluster is generated little by little, and PKS-Stream can discover the cluster correctly, while CluStream is incapable.

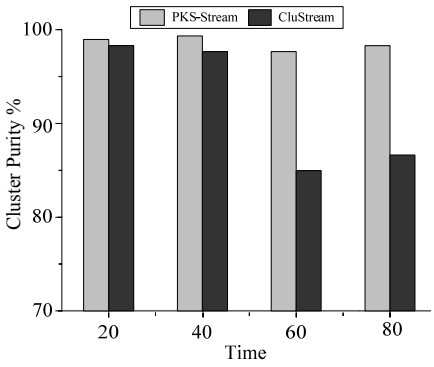

Figure 10. Comparison of clustering quality (DataSet1stream,speed=1000).

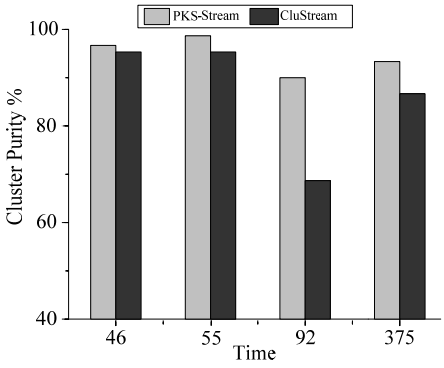

Figure 11. Comparison of clustering quality (KDD CUP'99,speed=1000)

In Figure11, the comparison of cluster purity between PKS-Stream and CluStream on Network Intrusion Detection data set KDD-CUP-99 is also given. The non-convex cluster exists in this real data set, and the number of clusters in the dataset is unknown. PKS-Stream needn't to predefine the number of clusters, and cluster the data stream based on grid density. So it can assign the data records into the corresponding cluster correctly. However, CluStream need to predefine the number of the clusters, and is not good on handling non-convex clusters. For instance, at time 92, the cluster purity of PKS-Stream is higher than 90%, and is 20% higher than CluStream. PKS-Stream can mapped different intrusions into their corresponding clusters, while CluStream relies on a predefined number of clusters, and may mapped different intrusions into one cluster incorrectly.

The scalability of PKS-Stream and CluStream are tested on KDD-CUP-99. As is shown in Figure12, the runtime of PKS-Stream and CluStream are increasing linearly when the data stream scale is increasing. However, PKS-Stream is more efficient. It assigns data records into corresponding grid cells, and adopts the index structure Pks-tree to store and index the grid cells. CluStream needs to

compute the distance between data records and clusters, and store the snapshots. All of these reduce the efficiency of CluStream.

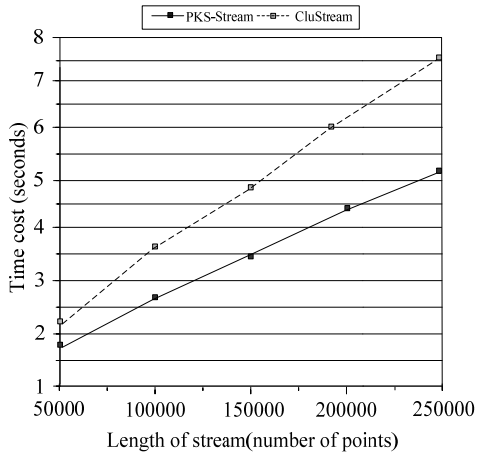

Figure 12. Execution time with varying length of stream

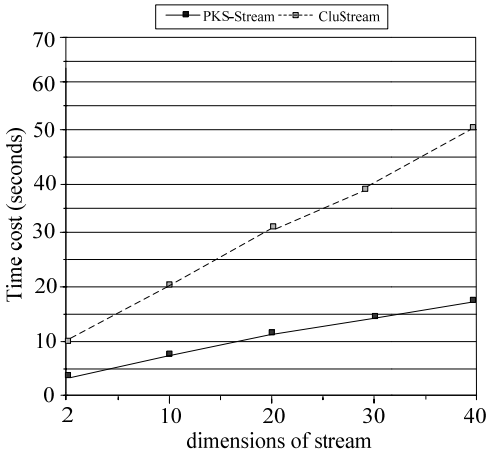

Figure 13. Execution time with varying dimensionality

Also, we test them on KDD-CUP-99 with different dimensionality. The result is shown in FIGURE 13, the runtime of PKS-Stream and CluStream are increasing almost linearly. PKS-Stream is faster than CluStream. Although the number of grid cells increases sharply with dimensionality of growth, many of the grid cells are empty. PKS-Stream introduces the Pks-tree to only store and index the non-empty grid cells, which reduces the computational complexity and makes it convenient to cluster the data stream. So its runtime does not soar with the increasing of dimensionality. However, CluStream needs to compute the micro-cluster in all dimensions, which costs a lot of time.

5. Conclusion

In this paper, we proposed PKS-Stream, a grid-density and Pks-tree based clustering algorithm. It is composed of two parts: online component and offline component. Online component mapped multi-dimensional data into corresponding grid cells at all levels of Pks-tree, and the characteristic vector of each minimum cell is updated. Offline component initially clusters the non-empty minimum cells by grid density. Then, based on grid density, it adjusts the generated clusters periodically to cluster evolving data stream more effectively. In contrast to previous algorithms, a new index structure Pks-tree is proposed to store and index the non-empty grid cells, which improves the efficiency of the algorithm. Moreover, a new pruning strategy is adopted to identify and remove noise points in real time, which is based on the density threshold function. Our experiments show that PKS-Stream can capture the evolving behaviors of the data stream and has a linear scalability. Also, the clustering quality of PKS-Stream is better than CluStream.

6. Acknowledgment

This work is supported by the National High Technology Research and Development Program ("863"Program) of China under Grant No. 2009AA01Z433, and the Natural Science Foundation of Hebei Province P. R. China under Grant No.F2010001298. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

7. References

[1] J.Gama, P.Rodrigues, An Overview on Mining Data Streams, Springer-Verlag Berlin Heid- elberg,

pp.29-45, 2009.
[2] S.Guha, N.Mishra, R.Motwani, Clustering data streams, Proceeding(s) of 41st Annual Symposium on Foundations of Computer Science, Los Alamitos, CA:IEEE Computer Society Press, pp.359-366, 2000.
[3] L.O'Callaghan, N.Mishra, A.Meyerson, Streaming data algorithms for high-quality clustering, Proceeding(s) of 18th Int'l Conf. Data Engineering, Los Alamitos, CA:IEEE Computer Society Press, pp.685-704, 2002.
[4] C.Aggarwal, J.Han, J.Wang and P.Yu, A Framework for Clustering Evolving Data Streams, Proceeding(s) of the 29th VLDB Conference, Berlin, Germany, pp.81-92, 2003.
[5] C.Aggarwal, J.Han, J.Wang, P.Yu, A Framework for Projected Clustering of High Dimensional Data Streams, Proceeding(s) of the 30th VLDB Conference, Toronto, Canada,pp.852-863, 2004.
[6] J.Chang, F.Cao, A.Zhou, Clustering Evolving Data Streams over Sliding Windows, Journal of Software, vol.18, no.4, pp.905-918, 2007.
[7] K.Udommanetanakit, T.Rakthanmanon, K.Waiyamai, E-Stream: Evolution-Based Technique for Stream Clustering, Springer-Verlag Berlin Heidelberg, pp.605-615, 2007.
[8] J.Ren, C.Hu, R.Ma, HCluWin: An Algorithm for Clustering Heterogene- ous Data Streams over Sliding Windows, Proc.International Journal of Innovative Computing, Information and Control, vol.6, no.5, pp.2171-2179, 2010.
[9] F.Cao, M.Ester, W.Qian, A.Zhou, Density-Based Clustering over an Evolving Data Stream with Noise, Proceeding(s) of the SIAM Conference on Data Ming, Bethesda, MD, pp.326-337, 2006.
[10] Y.Chen, L.Tu, Density-Based Clustering for Real-Time Stream Data, Proceeding(s) of the ACM SIGKDD 2007 Conference, California, USA, pp.133-142, 2007.
[11] J.Ren, B.Cai, C.Hu, Clustering Over an Evolving Data Stream Based on Grid Density and Correlation, ICIC Express Letters, vol.4, no 5(A), pp.1603-1609, 2010.