A Clustering Algorithm for Data Stream based on Grid-Tree and Similarity

Guoyan Huang, Wenyan Guo, Jiadong Ren, Lijuan Chen

College of Information Science and Engineering
Qinhuangdao, P.R.China, 066004
hgy@ysu.edu.cn, gwylscx@163.com, jdren@ysu.edu.cn, ljchenysu@qq.com

Abstract

Algorithms based on k-means are incompetent to find clusters of arbitrary shapes, and the number of clusters needs to be pre-specified. Moreover, most grid-based clustering algorithms can not deal with boundary points accurately. To address these issues, a novel approach based on density grid-tree and similarity, DGTSstream, is proposed. In DGTSstream, each new data record will be mapped into the gird-tree, and sporadic grids will be removed through setting update cycle and noise density threshold. The average density is exploited to design density threshold. This algorithm repeatedly seeks a maximum density grid without cluster flag, which will be used as a starting point for finding clusters according to depth-first strategy. Finally, the similarity is adopted to deal with the boundary points. Experimental results show that our algorithm can find clusters of arbitrary shapes, and has better clustering accuracy and efficiency.

Keywords: grid, similarity, data stream, clustering

1. Introduction

In recent years, mining data stream has become a very hot topic in data mining become of the large number of relevant application, such as weather monitoring, network monitoring, ecommerce and so on [1]. Stream data clustering is a challenging area of research that attempts to extract useful information from continuously arriving data [2].

LOCALSEARCH [3] is a clustering algorithm based on the divide and conquer scheme. A continuously iterative process based on a k-means algorithm is used to discover clusters. Stream algorithm [4] was proposed by O'Callaghan on the basis of LOCALSEARCH, in which n data points are clustered into k clusters with the minimum sum of square distance (SSQ) between data points and clusters. However, Stream algorithm can only provide a description of the current data stream, and the evolution of data stream can not be well reflected. Clustream [5], a clustering evolving data streams framework, consists of an online component and an offline component. In the online component, summary statistical information is produced and micro clustering is executed. The offline component uses pyramidal time frame to provide the clustering results of different time granularity. But it does not consider the decay of historical data and shows poor performance for clustering high dimensional data stream. To deal with high dimensional data stream, HPStream [6] is proposed by Aggarwal. Projection technology is utilized to cluster high dimensional stream data, and fading cluster structure is used to store historical data. Therefore, the influence on the current clustering results by the historical data will be declined gradually as time goes. Since those three algorithms (Stream, CluStream and HPStream) are the improved algorithms based on k-means, only spherical clusters can be produced. Denstream [7] and D-stream [8] can produce arbitrary shape clusters. The former introduces the potential micro-cluster structure and outlier micro-cluster structure. When the clustering is requested, an improved algorithm of DBSCAN is applied to get the final clustering result. Denstream uses the consistent global density, making the results of clustering sensitive to the parameter values. The latter partitions the data space into grids, and then data stream is clustered by calculating the density of grids. However, when each data is mapped to the grid, the position information of the data will be lost. Moreover, D-stream can not deal with the boundary points effectively. And some data locating in the edge of cluster may be removed, which will affect the clustering accuracy seriously. Meng proposed DG-Tree, which maps each new data record into a tree and eliminates the impact on the clustering results by the empty grids. The relative positions of the non-empty grid cells are preserved. Noise
nodes could be found by setting noise density threshold and update cycle. But selecting a unit arbitrarily as a starting point for clustering is blindness, which may increase the complexity of computation. Moreover, it will reduce the clustering accuracy because boundary points are not handled effectively.

To reduce the impact on clustering accuracy by the choice of parameter, the density threshold based on the average density is adopted. Because the data with similar attributes will be concentrated together, a maximum density grid without cluster flag will be sought repeatedly. And it will be used as a starting point for finding clusters using the depth-first strategy. To reduce the loss of the edge information of the cluster, the similarity is used to process the boundary points.

The rest of the paper is organized as follows. In Section 2, we describe the definition of problems. In Section 3, the clustering algorithm is introduced in detail. We conduct experimental study of DGTSstream in Section 4 and conclude the paper in Section 5.

2. Problem definitions

Each dimension is divided equally into \( \xi \) intervals. Each interval is a right open interval \([l_i, h_i)\), which is denoted by \(1, 2, \ldots, \xi\) successively, and the length of the interval is denoted by \(\text{len}\). Then the space \(S\) is divided into \(\xi^d\) disjoint grids, and each grid \(g\) is a \(d\)-dimensional hypercube, which is expressed as \((v_1, v_2, \ldots, v_d)\), where \(v_i\) is the interval label of the grid \(g\) in the \(i\)-th dimension. For each data record \(x\), we assign it a density coefficient. If \(x\) arrives at time \(t_c\), its density coefficient \(D(x, t)\) at time \(t\) is \(\lambda^{t-t_c}\). The density of grid \(g\) is \(D(g, t) = \sum_{x \in E(g, t)} D(x, t)\), \(E(g, t)\) is the set of data records that are mapped to \(g\) at or before time \(t\). The definition of neighboring grids is similar in [8].

Suppose a grid \(g\) receives a new data record at time \(t_n\), the last update time is \(t_l\), then the density of \(g\) can be updated as follows [9]:

\[
D(g, t_n) = \lambda^{t_n-t_l} D(g, t_l) + 1
\]

(1)

Suppose the last update time of a grid \(g\) is \(t_l\), it does not receive a new data record from \(t_l\) to current time \(t_c\), then the density of \(g\) can be updated at time \(t_c\) as follows:

\[
D(g, t_c) = \lambda^{t_c-t_l} D(g, t_l)
\]

(2)

**Definition 1.** Grid Characteristic Vector

\((p, \text{Count}, \text{Density}, \text{Centroid}, t_l, \text{Coordinate}, \text{Flag})\) is called grid characteristic vector, which is used to denote each grid, where \(p\) is the grid coordinate, \(\text{Count}\) is the number of data points in the grid, \(\text{Density}\) is the grid density at time \(t_l\), \(\text{Centroid}\) is centroid of the grid, \(t_l\) is the last time when \(g\) received a data record, \(\text{Coordinate}\) storages the position of data points, which is released when the grid is a dense grid, \(\text{Flag}\) is the cluster label of the grid.

The density threshold adopted by DGTSstream is as follows:

\[
M = \frac{\text{Max}(D) + \sum_{i=1}^{\xi^d} \frac{D_i}{k}}{2}
\]

(3)

Where \(\text{Max}(D)\) is the maximum density of all grids, \(D_i\) is the \(i\)-th non-empty grid density, \(k\) is the number of non-empty grid. At time \(t\), if \(D(g, t) \geq M\), then \(g\) is a dense grid; otherwise, \(g\) is a sparse grid.

3. Clustering algorithm

3.1. The grid-tree structure

The grid-tree is adopted which is a \(d+1\) dimensional multi-layer tree and has one tree root. The first \(d\) layers of the tree are corresponding to every dimension of the data respectively, and the \(d+1\) layer corresponds to the non-empty grids. A grid cell can be fixed uniquely by the path from the root to a
leaf node. Each non-leaf node has a storage space, which is used to store the maxDensity of the child nodes [10].

Figure 1 is the partition of two-dimensional space. Each dimension is divided into 5 equal sections and black grid denotes the non-empty grid. Figure 2 gives the grid-tree structure with 3 layers corresponding to the grids in Figure 1. The first two layers correspond to X and Y dimension respectively, the last layer corresponds to all the non-empty grids. The X dimension is divided into 5 intervals, but only the intervals 1, 3, 4, 5 contain data records, so the root has four internal nodes. The first internal node 1 of the root points to 5 on the second layer, and there is only a non-empty grid corresponding to 1 in the X dimension. From Figure 2, the maxDensity 5 can be got.

3.2. Time interval $t_m$

As data streams flow, the density of leaf nodes changes constantly. A dense grid can degenerate to a spare grid if it does not receive a new data for a long time, on the other hand, a spare grid may become a dense grid after it receives some new data records. Therefore, after a period of time, the density of each leaf node should be inspected and noise nodes should be removed. The value of the time $t_m$ cannot be too big or too small. If $t_m$ is too big, dynamical changes of data streams will not be adequately recognized. If $t_m$ is too small, it will result in frequent computation and increase the workload. We consider the minimum time needed for a dense grid to degenerate to a sparse grid as the time interval, the minimum time interval $t_m$ is

$$t_m = \log_\frac{M}{M-1}$$

3.3. Detecting and removing sporadic grids

Such grids that contain very few data are called sporadic grids. If these sporadic grids are not controlled, the total number of grids will become very large, thus we need to remove them. But not all of the spare grids are sporadic grids. Sparse grid is divided into two cases: (1) the grid received only
few data points; (2) the grid received few data points recently, but it may become dense grids with the data stream flowing. For (1), the spare grids are sporadic grids and should be removed. Therefore, we define a noise density threshold function to recognize the sporadic grids, as follows [11].

\[
\rho(t_c, t_0) = \frac{(t_c - t_0)^{\frac{1}{4}} - 1}{\frac{1}{4} t_c - 1}
\]  

(4)

Where \( t_c \) is the current time, \( t_0 \) is the last update time of the grid. If \( D(g, t_c) < \rho(t_c, t_0) \), then \( g \) is a sporadic grid.

When we prune the grid-tree, the maximum density value in non-leaf nodes will be searched first. If the maximum density value is smaller than the noise density threshold, this node and its branches will be deleted. Suppose the current noise density threshold is 3, the rightmost branch of the grid-tree will be cut off in Figure 2. Otherwise, each child node will be compared with the noise density threshold.

**PruningTree Algorithm**

**Input:** the grid-tree

**Output:** pruning the grid-tree

(1) pruningTree(q, level) //q points to the current node, level is the layer of the tree

(2) {

(3)   If (q.maxDensity<\(\rho(t_c, t_0)\))

(4)   {  Delete the current node corresponding to branches ;

(5)     Release ;

(6)   }

(7)   Else

(8)   {   For each child of q

(9)     pruningTree (q.child, level+1)

(10)   End for

(11) }

(12) }

Such grid that contains very few data should be removed in the pruning processing. The maximum density value storing in the current non-leaf node is \( q.\text{maxDensity} \), the noise density threshold is \( \rho(t_c, t_0) \). If \( q.\text{maxDensity} \) is smaller than \( \rho(t_c, t_0) \), the branch where this node locates can be removed directly, which avoids the density of each leaf node is compared with the noise density threshold.

### 3.4. Clustering

To address the blindness caused by choosing clustering starting point arbitrarily, the algorithm seeks a maximum density grid without cluster flag repeatedly, and this grid is used as a starting point for searching clusters based on density according to depth-first strategy. In the pseudo codes as follows, grid is the maximum density grid without cluster label, flag denotes the current cluster label to be formed; \( d \) is the dimension of data space.

**Clustering Algorithm**

**Input:** grids, flag

**Output:** clusters

(1) DepthFirstSearch (grid, flag)

(2) {

(3)    g.flag = f;

(4)    for(i=1; i≤d; i++)

(5)    {

(6)      //examine the left neighbor of grid in dimension i

(7)      gl=g.left ;

(8)      if((gl.density)>M & &(gl.flag is undefined))

(9)      DepthFirstSearch (gl, flag) ;
(10) //examine the right neighbor of grid in dimension i
(11) gr=g.right;
(12) if((gl.denstiy)>M & (gr.flag is undefined))
(13) DepthFirstSearch (gr, flag);
(14) }
(15)}

The maximum density grid is used as a starting point for clustering ((1)-(3)), the depth-first strategy is adopted to find clusters based on density ((4)-(13)). The blindness of choosing clustering starting point arbitrarily can be solved, which can avoid generating a lot of meaningless clusters and increasing the computation complexity.

3.5. The treatment of boundary points

If g is a spare grid and there exists dense grids in its neighboring grids, we compute the sim between each data point x in g and the centroid of the neighboring dense grid having the nearest distance to x. If sim>ε, x is a boundary point; Otherwise, x is an outlier.

\[
sim = 1 - \frac{\sqrt{\sum_{i=1}^{d} (X_i - C_i)^2}}{\text{len}}
\]  

Where \( X_i \) is the value of the data points for the i-th dimension, \( C_i \) is the value of centroid for the i-th dimension, and len is the side of the grid.

**Boundary Algorithm**

**Input:** Centroid, data record x

**Output:** \( x \in \text{flag C} \) or x is an outlier

(1) for each g
(2) if (g is a spare grid and there exists dense grids in its neighboring grids )
(3) {
(4) for each data record x\( \in g \)
(5) find out the dense grid h which has the nearest distance to x ;
(6) compute \( \text{sim}(x, \text{Centroid}); \) //Centroid is the centroid of h
(7) if  \( \text{sim} > \varepsilon \)
(8) \( x \in \text{flag C}; \) //C is the cluster flag of
(9) else
(10) \( x \) is an outlier;
(11) End for
(12) }
(13)End for

3.6. DGTSstream algorithm framework

DGTSstream consists of an online component and an offline component. In the online component, new data records are mapped into grid-tree and the characteristic vector of the grid is updated. In the offline component, the density of grids is updated to the time \( t_c \), maxDensity of the non-leaf nodes is registered and the grid-tree is pruned. When a clustering request arrives, the data stream is clustered.

**Online component Algorithm**

**Input:** Data Stream X

**Output:** the characteristic vector of the grid(\( p, \text{Count, Density, Centroid, } t_l, \text{Coordinate, Flag} \))

(1) \( t_c = 0; \)
(2) while data stream is active do
(3) read data record \( x=(x_1,x_2\ldots x_d) \)

(4) map each input data record into corresponding grid \( g \)

(5) update \((p, \text{Count}, \text{Density}, \text{Centroid}, t_l, \text{Coordinate}, \text{Flag})\)

(6) \( t_c = t_c + 1 \)

(7) end while

**Offline component Algorithm**

**Input:** the characteristic vector of the grid \((p, \text{Count}, \text{Density}, \text{Centroid}, t_l, \text{Coordinate}, \text{Flag})\)

**Output:** clusters

(8) Construction grid-tree

(9) if \((t_c \mod t_m == 0)\) 

(10) 

(11) update the density of grids to the time \( t_c \); 

(12) register maxDensity of the non-leaf nodes; 

(13) call PruningTree Algorithm; 

(14) 

(15) if clustering request arrives 

(16) 

(17) call Clustering Algorithm; 

(18) call Boundary Algorithm; 

(19) 

In the online component, data points are mapped to the corresponding grid, and the grid characteristic vector is updated ((1)-(6)). In the offline component, grid-tree is constructed, and then two tests are conducted. First, if the tree needs to be pruned at the current time, pruning module is called ((9)-(14)). Second, if clustering request is made, clustering and boundary modules are called ((15)-(19)).

4. Experimental results and analysis

All of our experiments are conducted on a PC with Intel Core 2.2 GHz CPU and 2GB memory running windows XP. DGTSstream has been implemented in Visual C++ 6.0. In all experiments, we use \( \lambda = 0.3, \quad \varepsilon = 0.7 \). We adopt two testing sets, the synthetic data set and the real data set. The real data set uses KDD CUP99 network intrusion detection data sets, which contains a total of five clusters and each connection record contains 42 attributes. As in [3], all the 34 continuous attributes are used for clustering.

4.1. Clustering quality comparison

Clustering purity is used to test clustering quality in this paper. The comparison of clustering purity between DGTSstream and Clustream on the synthetic data sets is showed in Figure 3. We can see that the clustering purity of DGTSstream is higher than Clustream. DGTSstream distinguishes sporadic grids from the spare grids by the noise density threshold and removes them. But Clustream does not distinguish the micro-cluster generated by the noise points.

Figure 4 shows the comparison of clustering purity between DGTSstream and Clustream on KDD CUP 99 data set. The clustering purity of DGTSstream is higher than Clustream at most of the times except time 310. At that time, the two algorithms achieve 100% purity. We checked the data set and found that all the points belonged to one attack. At time 410, the clustering purity of DGTSstream is 15% higher than Clustream, because Clustream mapped different intrusions into one cluster, which stresses the weakness that Clustream needs to predefine number of clusters. However, DGTSstream algorithm is based on the grid, which does not need to specify the number of clusters in advance, and can map different intrusions to the corresponding clusters.
4.2. Scalability

The efficiency of algorithms is measured by the execution time. Figure 5 shows the comparison of efficiency between DGTSstream and Clustream on KDD CUP 99 data set. The runtime of DGTSstream and Clustream grow linearly with the data stream scale increasing, but DGTSstream is more efficient than Clustream. DGTSstream maps data records into grids, and uses the tree structure to store grids. However, Clustream needs to store the current snapshots of micro-clusters and compute the distance between clusters and data records, which has an effect on the efficiency.

The comparison of the execution time between DGTSstream and Clustream with various dimensionalities on KDD CUP 99 data set is showed in Figure 6. It can be seen that as the dimensionality increases, the execution time increases linearly. But the execution time of DGTSstream is smaller than Clustream. The number of grids increases sharply with the increase of the dimensionality, but many grids are empty. DGTSstream adopts grid-tree structure to store the non-empty grids, and removes sporadic grids. However, Clustream computes the micro-cluster in all dimensions, so it costs much time.
5. Conclusion

In this paper, DGTSstream based on gird-tree and similarity is proposed. The algorithm is composed of an online component and an offline component. In the online component, each new data record is mapped into gird-tree, which stores the summary information of data stream, as well as the relative position of the non-empty grid cells. In the offline component, by using the maximum density grid as a starting point, clusters will be found in the depth-first strategy. In contrast to previous algorithms, a method based on similarity is adopted to deal with the boundary points, so as to improve the accuracy of clustering. According to the characteristics of the data with similar attributes concentrating together relatively, a maximum density grid is used as the starting point of clustering. This solves the blindness caused by choosing clustering starting point arbitrarily. Experimental results show that DGTSstream has a linear scalability, and the clustering quality is better than Clustream.

6. Acknowledgements

This work is supported by the Natural Science Foundation of Hebei Province P.R. China No. F2009000477. We also feel grateful for the helpful comments and suggestions of the experts.

7. References