Density-based clustering of data streams at multiple resolutions

Wan, Li

2008

Wan, L. (2008, March). Density-based clustering of data streams at multiple resolutions. Presented at Discover URECA @ NTU poster exhibition and competition, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/84871>

© 2008 The Author(s).

Downloaded on 26 Apr 2023 22:28:01 SGT

Density-Based Clustering of Data Streams at Multiple Resolutions

Introduction

In data stream clustering, it is desirable to have algorithms that are able to detect clusters of arbitrary shapes, changing clusters that evolve over time, and clusters with noise. In recent years, stream data clustering algorithms are based on an online-offline approach: The online component captures synopsis information from the data stream (thus, overcoming the real-time and memory constraint issues) and the offline component generates clusters using the stored synopsis.

The online-offline approach affects the overall performance of stream data clustering in various ways: (1) How easily is the synopsis information derived from stream data? (2) The complexity of data structure used to store and manage the synopsis information. (3) The frequency with which the offline component is used to generate clusters.

In this project we propose an algorithm that (1) computes and updates synopsis information in constant time; (2) allows users to discover clusters at multiple resolutions; (3) determines the right time for users to generate clusters from the synopsis information; (4) generates clusters of higher purity than existing algorithms; and (5) determines the right threshold function for density-based clustering based on the fading model of stream data. To the best of our knowledge, no existing data stream algorithm has all of these features. Experimental results show that our algorithm is able to detect arbitrarily shaped evolving clusters of high quality.

MR-Stream Algorithm

Hash Tree Data Structure

We assume that the input stream data has n dimensions and forms an n -dimensional space S . Any record of the data stream is a n -dimensional vector $x = [x_1, x_2, \dots, x_n]$ in space S .

We conceptually partition the space S into well-defined partitions or cells. Any cell C can be further partitioned into 2^n sub-cells if we divide each dimension into 2 parts. The end result of this partitioning is a recursive partitioning of space S into cells at various granularities.

We use a tree-like data structure to mirror the space partitioning so that each tree node corresponds to a cell. A tree node that has children nodes corresponds to the further partitioning of the parent cell. The space partitioning and its corresponding tree structure are shown in Figure 1.

We may use a user-defined parameter H to control the levels of partitioning so that the height of the tree is no more than H (the root node is at height 0). Thus, each dimension is divided into 2^H intervals.

We assign a weight value to each record of the data stream. This weight value decreases over time if the record does not appear again frequent enough in the stream.

Figure 1: 2D space case with $H = 3$

$$w(x, t_j) = w(x, t_i) f(t_j - t_i)$$

$$f(t) = \lambda^{-at}$$

Fading function

MR-Stream Algorithm

Algorithm 1 Online Component of MR-Stream

```

1:  $t \leftarrow 0$ 
2: initialize tree  $T$ 
3: while data stream is active do
4:   read record  $e$ 
5:    $g \leftarrow \text{updateTree}(e, T)$ 
6:    $\text{pruneTreeUp}(g, T)$ 
7:   if  $t \bmod t_p = 0$  and  $t \neq 0$  then
8:     sample number of nodes in  $T$ 
9:      $\text{updateTreeWeight}(T)$ 
10:     $\text{pruneTreeDown}(T, \text{root})$ 
11:     $\text{mergeTreeDown}(T, \text{root}, \text{DENSE})$ 
12:     $\text{mergeTreeDown}(T, \text{root}, \text{SPARSE})$ 
13:   end if
14:    $t \leftarrow t + 1$ 
15: end while

```

Algorithm 2 Offline Component of MR-Stream

```

1: initialize cell set  $\text{cellSet}$ 
2:  $\text{generateSetForClustering}(T, \text{root}, \text{height})$ 
3:  $\text{cluster}(\text{cellSet})$ 

```

Detecting Sporadic Cells

THEOREM 3. In order that an algorithm prune sufficient cells without removing any potential cluster cell, the following two conditions must hold:

1. density threshold function p should be defined as

$$p(\Delta t) = \frac{C_t}{V(1-\lambda^{-a})} (1 - \lambda^{-a\Delta t - c})$$

where $0 < c \leq at_p$

2. The minimum bound at maximum tree high

$$H \geq \log_2 \left(\frac{C_t(1 - \lambda^{-at_p})}{(1 - \lambda^{-a})} \right)$$

Memory Sampling Method

In the data streaming process, the number of nodes in the tree is not stable. We establish the relationship between the number of nodes in the tree (which also determines the memory cost) and the evolution of clusters in this section. This relationship is important as it allows one to obtain evolving cluster information by sampling the memory cost rather than running the relatively computational expensive offline component.

Experiment Results

Detecting evolving data streams (2 dimensional data stream)

Figure 2: 2 Dimensional Synthetic Dataset

Figure 3: $t=10$

Figure 4: $t=28$

Figure 5: $t=34$

Figure 6: $t=49$

Figure 7: $t=56$

Memory Sampling Method

The changing of memory cost reflects the evolving of clusters

Performance analysis (34 dimensional data stream)

1. High Accuracy

2. Low Memory Cost

3. Short Process Time

Our Contribution

- The MR-Stream algorithm provides a hierarchical, multi-resolution view of clusters at any time; that is, MR-Stream can perform clustering algorithm at different resolution by the offline component while not affecting the online component.
- The online component of MR-Stream has constant time complexity to accept a new record from data stream. The offline component can provide a better cluster result by extending the neighbor cell concept.
- As all density-based clustering algorithms require a threshold function to control the discovery of clusters with respect to memory limitations, we provide a general method to determine the threshold function for a given memory bound. The function enables any data stream clustering algorithm that is based on the fading model to work within a fixed memory bound without missing any potential clusters.
- We propose a memory sampling method that enable users to detect the evolving nature of clusters without executing the expensive offline component. It is important as it helps users to determine when to run the offline component on-demand rather than running it periodically.