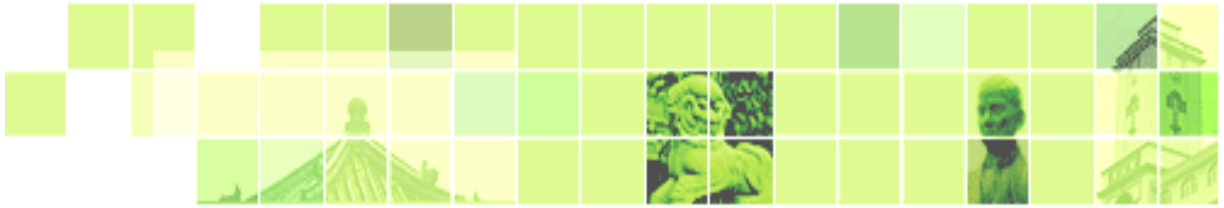


数据挖掘第四次实验课

2017.03.22

目录

- CluStream
- Others

CluStream

- CluStream：数据流聚类模型
  - Online: micro-clustering 微聚类
  - Offline: macro-Cluster 宏聚类

CluStream - Description

- 数据流表达：
  - 在$T_1 \ldots T_k \ldots$时刻到达的数据流可以看作是一系列的多维记录：$\overline{X_1} \ldots \overline{X_k} \ldots$
  - 其中$\overline{X_i} = (x_i^1 \ldots x_i^d)$是一个d维的记录(record)
- 两个关键的概念
  - Micro-cluster
  - Pyramidal Time Frame

CluStream – Micro-cluster

- 用微簇来描述关于数据位置上的统计信息。
- 是簇特征向量的一个扩展（维度为2*d+3的向量）
  - e.g：有n个点$\overrightarrow{X_1}$ … $\overrightarrow{X_k}$… $\overrightarrow{X_n}$分别对应的时间戳是T_1 … T_k…. T_n
  - 特征向量可以写作一个2*d+3维的向量

$$(\overrightarrow{CF2^x}, \overrightarrow{CF1^x}, CF2^t, CF1^t, n)$$

第p个元素：$\sum_{j=1}^{n} x_{i_j}^p$ $\sum_{j=1}^{n} (x_{i_j}^p)^2$.

CluStream – Pyramidal Time Frame

- 上述微簇需要在某些时刻维护和存储到磁盘以供离线阶段查询。
- 由于数据量巨大，不可能将所有时刻的微簇信息都存储到磁盘（这部分信息叫做快照），
- 因此引入时间帧结构。它将时间轴划分成不同粒度的时刻，结果是离现在的越近粒度越细，反之越粗。

CluStream – Pyramidal Time Frame

- 如果确定order？
- 每个order的元素数量
- 如何做到无冗余？

siderable redundancy in storage of snapshots. For example, the clock time of 8 is divisible by 2^0, 2^1, 2^2, and 2^3 (where $\alpha = 2$). Therefore, the state of the micro-clusters at a clock time of 8 simultaneously corresponds to orde 0, order 1, order 2 and order 3 snapshots. From an implementation point of view, a snap-

CluStream – Pyramidal Time Frame

Order of Snapshots	Clock Times (Last 5 Snapshots)
0	55 ~~54~~ 53 ~~52~~ 51
1	54 ~~52~~ 50 ~~48~~ 46
2	52 ~~48~~ 44 ~~40~~ 36
3	~~48~~ 40 ~~32~~ 24 ~~16~~
4	48 ~~32~~ 16
5	32

Table 1: An example of snapshots stored for $\alpha = 2$ and $l = 2$

总的快照数：$(\alpha^l + 1) \cdot \log_\alpha(T)$

CluStream – Pyramidal Time Frame

- 这种时间帧结构的一些好处。
  - 能满足用户对最近数据感兴趣的需求；
  - 运行100年的数据流仅仅需要存储大概95个快照（$(2+1)\cdot\log2(100*365*24*60*60) \approx 95$），这能满足有限内存的需求。

CluStream – Online Component

- 需要完成几个功能
  - 实时处理新到达的数据（这些数据应该分到哪个簇或者形成新的簇）
  - 周期存储统计结果——Pyramidal Time Frame

CluStream – Online Component

- 问题一：新的点到来时，怎么判断是否属于已有的簇？
- 问题二：如果不属于，是一个outliner，如何确定以它为中心新建一个簇而不增加内存的压力？

CluStream – Online Component

- 问题一解答：
  - 初始化簇：首先在磁盘上存储最初始的initNumber个数据点，然后采用标准的k-means算法形成q个微簇：M_1、M_2…M_q。
  - 在线处理：对于以后达到的每一个数据点X_{ik}。首先计算X_{ik}与q个微簇中的每一个中心的距离 $dist(M_j, \overline{X_{i_k}})$ 将其放到离它最近的那个簇M_p中。

CluStream - Online Component

- 问题一解答：
  - 特殊情况：
    - 1.X_{ik}虽然离M_p最近，但是X_{ik}却在M_p的边界；
    - 2.由于数据流的演化，X_{ik}可能是一个新簇的开端。

CluStream - Online Component

- 问题二解答——处理方法
  - 为落在边界外的数据点创建一个带独有标志id的新簇，这需要减少一个其他已经存在的簇。
  - 这可以通过删除一个最早的簇或者合并两个最早的簇来实现。

CluStream - Online Component

- 问题二解答：——怎么安全删除一个簇
  - 估计每一个簇中最后m个达到的数据点的平均时间戳，然后删除带有最小时间戳的值（时间越早值越小且小于用户定义的阈值）的那个簇。
  - 这种方法只增加了存储每个簇中最后m个点的数据的信息（时间戳）。

CluStream - Online Component

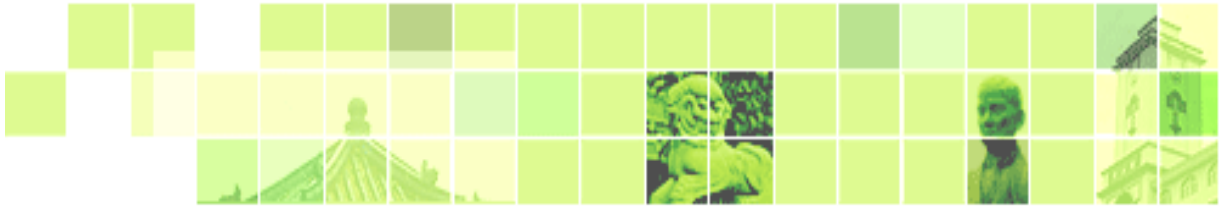
- 问题二解答：——怎么合并两个簇的id
  - 此时需要合并某两个靠的最近的微簇。此时用它们原来的id一起标志这个新的微簇。
  - 同时，需要存储金字塔时间结构对应时刻的微簇（实际上指的是微簇的特征向量值）到磁盘。

CluStream – Offline Component

- 离线部分使用k-Means来完成宏聚类
- 用户提供两个参数h和k，h是时间幅度，k是预定义的需要形成的簇的数目。

CluStream - Offline Component

- 该部分采用改进的k-means算法
  - 初始阶段： 不在随机的选取种子，而是选择可能被划分到给定簇的种子，这些种子其实是对应微簇的中心。
  - 划分阶段：一个种子到一个“伪数据点”（也就是微簇）的距离就等于它到“伪数据点”中心的距离。
  - 调整阶段：一个给定划分的新种子被定义成那个划分中带权重的微簇中心。



Thank you !

Q&A