数据流聚类算法比较

摘要：数据流技术随着科技的发展应运而生，也给传统数据挖掘方法带来了挑战。本文对数据流聚类算法中的常见算法作概述和聚类评价，并以经典的 K-Means 算法作比较，以真实的数据进行实证分析，对 Leader 算法、One Pass K-Means 算法、BIRCH 算法进行比较。为使比较更加客观并避免偶然性，本文分别从聚类的时间和聚类的效果上算法进行比较，并查看数据的读入顺序对每种算法的聚类效果的影响，然后针对每一种聚类算法分别对该算法的参数进行调整，查看不同算法参数对该算法聚类效果的影响。最终分别从内存的占用、聚类时间消耗、聚类效果的稳定、参数的设定等方面对每种算法的优缺点进行分析和评价。根据本文对算法的评价，可以在不同的实际情况中选择合适的算法来进行数据流挖掘。

关键词：数据流；Leader；One Pass K-Means；BIRCH

1. 研究背景及意义

随着科技和互联网技术的快速发展，数据的获取日趋简单。通过实时产生和获取的数据便形成数据流。数据流在日常生活中随处可见，从用户上网查询的网络记录日志，到金融行业的证券交易数据，再到商业销售进出帐流水记录，都可以看到数据流的无处不在。对于数据流的研究成为数据挖掘领域的新的研究热点，其中聚类技术作为最常用的数据挖掘算法之一，也得到了深入的研究。

目前学者对于数据流的研究可以大体分为两个方向，分别是数据流管理系统和数据流挖掘。数据流管理系统主要研究在流数据的查询，目前已经有多个研究机构对数据流管理系统进行研究。数据流挖掘方面的研究包括多数据流挖掘和单数据流挖掘。传统数据挖掘算法中，聚类是最为常用且有效的算法之一。在数据流挖掘算法中，聚类也被学者深入研究。目前学者们以及研究出多种数据流聚类算法，但是各种算法在时间上、效率上、内存的使用上各不相同。对数据流聚类算法作出分析和比较，了解各种算法优缺点，有利于在实际应用中合适地选择算法来进行数据分析和评价。

2. 数据流及其特性

2.1 数据流概念

数据流是指动态产生的，随着时间不断变化的数据。数据流不来源于一个稳定的分布，且数据量庞大，需要合适的策略进行存储和分析。通过以一定速度进入并且连续，在序列中只能够按照序列进入的先后顺序读取一次。由于数据流的速度以及进入的顺序不能够被控制，数据流具有无限的体量，而数据的保存需要占用磁盘内存或缓存空间，这使得用于对数据流进行处理的系统不能够保存整个数据流，将数据流读取一次之后即抛弃。相比传统的统计学，关注的都是小样本的数据集，并且在对数据进行处理之前，认为数据都是从某种固定的分布而产生。传统统计学通过观测到的数据来对数据的分布进行推断，利用分布的特性取解释观测现象或做预测。数据流给传统的统计学和数据挖掘方法带来了挑战。

2.2 数据流基本假设

数据流对于数据和使用场景有两个基本假设，分别为：（1）数据不可以被随机读取；（2）计算机的内存相对于数据量非常有限。所以对于数据流的计算方法需要特殊的抽样、随机化技术还有新的估计和近似算法。

2.3 数据流处理难点

数据流处理的难点在于：（1）数据是无限的；（2）每个变量的可能取值范围也是无限的；（3）在存储和使用数据时，不可能存储历史所有的数据，然后再对数据进行分析和计算。通常存储一些概括性的统计量，再存储大小和精度方面，不可能两全其美。收到存储和计算时间的约束，对于数据流的数据挖掘，通常有两种解决方案：

（1）ε近似，即近似得到的结果误差不能大于ε

（2）(ε, δ)近似，即近似得到的结果不小于一定的概率$(1-\delta)$使得近似误差不大于ε

3. 数据流的聚类算法

3.1 聚类简介

聚类是一种无监督学习方法,根据类间相似度最下而类内相似度最大原则,将数据集分为若干类。通过将对象分成不同的组，使得同一组内的对象之间具有更多的相似性，不同组之间的对象有更大的不相似性。聚类方法在数据挖掘中广泛使用，可以用来显示数据的分布或者对数据进行预处理。最常用的聚类方法通过样本之间的距离作为相似性的度量。但是这种方法存在缺点：计算空间复杂度较高，需要占用很大的内存空间。在数据流中，聚类要使用尽可能少的空间和时间来维护一个不断变化的聚类分组结果。

传统聚类方法可以划分为以下几种方法：

（1）分割算法：通过最小化某个目标函数，将所有的物体分割成 k 个聚集类，其中 K-Means 为最常见的算法之一。

（2）Micro-Clustering 算法：将聚类过程分为两阶段。首先使用在线方法局部总结数据流，然后通过局部的聚集类得到一个全局的聚类。常见的算法包括 BIRCH 还有 CluStream。

（3）基于密度函数的聚类：主要考虑区域之间的连通性和密度函数之间的联通性。常见的算法包括 DBSCAN 以及 OPTICS。

（4）网格算法，常见算法包括 Fractal Clustering 以及 STING。

（5）基于模型的算法，常见算法包括 COWEB 以及 SOM。

数据流算法的难点和关键点是要实时计算，因此要使用微小更新的算法。数据流的聚类方法需要具有如下特征：（1）表达的紧致性；（2）对新的数据点能进行快速、微小的处理；（3）对聚类变化进行跟踪；（4）迅速清晰地发现离群点。

3.2 数据流聚类算法

3.2.1 Leader 算法

Leader 算法是最简单的一次传递的分割算法。该算法由斯帕思（Spath）于 1980 年提出，通过用户自定义的阈值来规定样本和聚类中心的最大允许距离。

Leader 算法的核心思想是：每一步聚类都会把当前的样本划分到最相似的类，但是如果该样本与它最相似的类之间的距离超过了之前已经定义的阈值，则将该样本自己划分为一类。

Leader 算法伪代码

输入：样本序列X，每一个样本记为x_i

样本与最相似类之间的最大阈值δ

输出：k 个聚类中心

步骤 1：初始化类中心，将第一个样本看作一类，即$C = x_1$

步骤 2：传递数据流中样本$x_i \epsilon X$：

找到离该样本x_i最近的类C_r

步骤 3：如果样本x_i与类C_r的距离 d$(x_i, C_r) < \delta$)：

则产生新的类$C = C \cup x_i$

步骤 4：循环步骤 2 和步骤 3，直到数据流的所有样本传递完毕

3.2.2 One Pass K-Means

在数据挖掘的聚类算法中， K-Means 算法是广泛应用的聚类算法之一。对于样本数据，通过优化类内距离确定聚类情况：

$$\min(\sum_{c=1}^{k}\sum_{x_i \in C}(x_i - x_C)^2)$$

一次传递的 K-Means 算法由费恩斯德姆（Farnstorm）提出，也是数据流挖掘中最为广泛应用的聚类算法之一。

一次传递的 K-Means 算法的核心思想是，首先存储一部分数据作为缓存，然后运用 K-Means 算法，将 k 个类的中心点的聚类特征（CF）进行记录，然后使用加权的聚类特征对新进入的数据作 K-Means 聚类。

One Pass K-Means 算法伪代码

输入：样本序列X，每一个样本记为x_i

样本聚类个数k

输出：k 个聚类中心

步骤 1：随机初始化类中心

步骤 2：将缓存区读满数据，利用 K-Means 得到 k 个类中心 Centroid

步骤 3：将这 k 个类中心分别看作 k 个点，每个点分别赋予权重n_k，即将每个点都看作n_k个同样的点，n_k表示第 k 个中心的点的个数。将 k 个点与缓冲区中的数据作 K-Means 聚类，更新类中心。

步骤 4：清空缓存区的点，并读入新的数据。

步骤 4：循环步骤 3 和步骤 4，直到数据流的所有样本传递完毕。

3.2.3 层次聚类（BIRCH 算法）

（1）聚类特征树

层词聚类与聚类特征树有着密不可分的关系。聚类特征树（CF Tree）的每一个节点是由若干聚类特征（CF）组成。聚类特征树的每个节点包括叶子节点都有若干个 CF，内部节点的 CF 有指向叶子节点的指针，所有的叶子节点通过双向链表连接起来。在聚类特征树中，聚类特征用三元组（N，LS，SS）表示。聚类特征（Cluster Features，CF）又称为 Micro-Cluster，是数据流聚类中的重要的概念，可以用三元组（N，LS，SS）来对其进行表达，用于存放统计量。其中：

N ← 数据的个数

LS ← 数据的和

SS ← 数据的平方和

并且，聚类特征 CF 具有如下两个性质：

（1）微增性，即对点集增加一个点之后，CF 只做出很小的更新。

$$N_A \leftarrow N_A + 1$$
$$LS_A \leftarrow LS_A + x$$
$$SS_A \leftarrow SS_A + \mathrm{x}^2$$

（2）可加性，即将两个不相交的点集合并在一起之后，新的 CF 可以计算得到。

$$N_C \leftarrow N_A + N_B$$
$$LS_C \leftarrow LS_A + LS_B$$
$$SS_C \leftarrow SS_A + LS_B$$

由聚类特征 CF 可以很容易计算得到类中心 Centroid

$$\overrightarrow{X_0} = LS/N$$

将 CF 的可加性放到 CF Tree 中，对于每个父节点的 CF 节点，其 CF 三元组的值等于该节点所指向的子节点的三元组之和。

聚类特征树首先需要定义几个参数，分别为：（1）每个内部节点的最大的 CF 数 B，用于控制内部节点的最大数量；（2）每个叶子节点的最大的 CF 数 L，用于叶子节点的最大数量；（3）针对叶子节点中某个 CF 中的样本点来说的，它是叶子节点每个 CF 的最大样本半径阈值 T，用于控制所有样本点要在半径小于 T 的一个超球体之内。

对于聚类特征树，初始时 CF Tree 为空集，没有任何样本。从训练集中读入第一个样本点并将它放入一个新的 CF 三元组 A。此时该三元组的 CF 特征中 N=1，并将该 CF 放入根节点。然后继续放入下一个样本点，若该样本点与样本点 A 都在半径为 T 的超球体之内的话，则将这个样本点也放入 CF A 中，并更新 A 的三元组的值。此时该三元组的 CF 特征中的 N=2。接着继续放入样本点，若样本点位于半径为 T 的超球体之外，则需要一个新的三元组 B 来放入该样本点，根节点有两个 CF 三元组 A 和 B。同样，根据后进入的样本点，判断是和 A 还是 B 在半径小于 T 的超球体之内，若是则放入对应的三元组中。当继续放入样本点，但是当 CF 数超过 L 的时候，就需要对节点进行分裂，将上一层节点一分为二。对于该 LN 中的所有 CF 元组中最远的两个叶子节点，并将新样本点划分到这两个新叶子节点上。当内部节点的最大 CF 数超过 B 时，需要将根节点进行分

裂，且分裂的方法与叶子节点的分裂一样。

聚类特征树的生成过程总结为：

（1）从根节点向下寻找和新样本距离最近的叶子节点和叶子节点里最近的CF 节点

（2）如果新样本加入后，这个 CF 节点对应的超球体半径仍然满足小于阈值T，则更新路径上所有的 CF 三元组，插入结束。否则转入3.

（3）如果当前叶子节点的 CF 节点个数小于阈值 L，则创建一个新的 CF 节点，放入新样本，将新的 CF 节点放入这个叶子节点，更新路径上所有的 CF 三元组，插入结束。否则转入4。

（4）将当前叶子节点划分为两个新叶子节点，选择旧叶子节点中所有 CF 元组里超球体距离最远的两个 CF 元组，分布作为两个新叶子节点的第一个 CF 节点。将其他元组和新样本元组按照距离远近原则放入对应的叶子节点。依次向上检查父节点是否也要分裂，如果需要按和叶子节点分裂方式相同。

（2）层次聚类算法

层次聚类（Balanced Iterative Reducing and Clustering using Hierachies，BIRCH），全称为利用层次方法的平衡迭代规约和聚类，是数据流中一个重要的层次聚类算法。该算法只需要一次扫描数据集就可以进行聚类。

BIRCH 算法利用聚类特征树（Clustering Feature Tree，CF Tree）进行快速聚类。将所有的训练集样本建立了 CF Tree，一个基本的 BIRCH 算法完成，对应的输出就是若干个 CF 节点，每个节点里的样本点就是一个聚类的簇。也就是说 BIRCH 算法的主要过程，就是建立 CF Tree 的过程。但是，BIRCH 算法除了建立 CF Tree 来聚类，还有一些可选的算法步骤的， BIRCH 算法的流程如下：

BIRCH 算法伪代码

输入：样本序列X，每一个样本记为x_i

输出：k 个聚类中心

步骤 1：将最先传入的样本x_1作为根节点

步骤 2：对于新传入的样本x_i，寻找和x_i距离最近的叶子节点和叶子节点里面最近的 CF 节点。

步骤 3：判断x_i加入后是否该 CF 节点对应的超球体半径小于阈值 T。

若满足，则更新路径上所有的 CF 三元组。

否则：判断当前叶子节点的 CF 节点个数小于阈值 L。

若满足，则创建一个新的 CF 节点，放入新样本，将新的 CF 节点放入这个叶子节点，更新路径上所有的 CF 三元组

否则：将当前叶子节点划分为两个新叶子节点，选择旧叶子节点中所有 CF 元组里超球体距离最远的两个 CF 元组，分布作为两个新叶子节点的第一个 CF 节点。将其他元组和新样本元组按照距离远近原则放入对应的叶子节点。依次向上检查父节点是否也要分裂，如果需要按和叶子节点分裂方式相同。

步骤 4（可选）：将步骤 3 建立的 CF Tree 进行筛选，去除一些异常 CF 节点，这些节点一般里面的样本点很少。对于一些超球体距离非常近的元组进行合并

步骤 5（可选）：利用其它的一些聚类算法比如 K-Means 对所有的 CF 元组进行聚类，得到一颗比较好的 CF Tree.这一步的主要目的是消除由于样本读入顺序导致的不合理的树结构，以及一些由于节点 CF 个数限制导致的树结构分裂。

步骤 6（可选）：利用步骤 4 生成的的 CF Tree 的所有 CF 节点的质心，作为初始质心点，对所有的样本点按距离远近进行聚类。这样进一步减少了由于 CF Tree 的一些限制导致的聚类不合理的情况。

BIRCH 算法不需要提前确定类别数 K。如果不提前确定 K 值，则最后的 CF 元组的组数即为最终的 K，否则会按照输入的 K 值对 CF 元组按距离大小进行合并。一般 BIRCH 算法适用于样本量较大的情况并且 BIRCH 除了聚类还可以额外做一些异常点检测和数据初步按类别规约的预处理。但是如果数据特征的维度非常大，比如大于 20，则 BIRCH 不太适合。

3.3 数据挖掘算法评估

数据挖掘需要从多角度对算法进行评价，数据流的挖掘算法也需要从多角度进行评价。但是区别于传统统计学方法，数据流中的数据和算法都是随着时间在变化的。数据流处理中，由于得到的是一个连续时间的流式数据，而不是具有固定样本大小的来自平稳分布的独立同分布样本，且模型是动态的，而不是静态的，使得数据是从一个动态环境、非平稳分布得到的，而不是来自一个平稳分布。这使得数据流的评价存在难点。

在数据流评价准则中可以使用“序贯预测误差”。Hulten and Domingos(2001)认为从数据流学习决策模型的高效算法应该具有以下特征：

（1）每个数据点的计算时间小；

（2）处理数据流所占用的内存不会随着时间增加而增加；

（3）一次传递数据即可得到决策模型；

（4）模型不随着样本进入的顺序而改变；

（5）可以处理“概念漂移”；

（6）对于平稳分布的数据，数据流方法的得到的决策模型应该近似于使用传统方法得到的模型。

可以看到，对于数据流产生影响的三个主要方面分别是空间、时间和推广力：

（1）空间上要求数据流处理所占用的内存大小是固定的；

（2）时间上要求处理新进来的样本需要的时间是固定的；

（3）推广力上要求数据流学习模型不会依赖于观测数据的方法，可以学习到数据的本质。

对数据流的评估，通常有两种方法：

1. 保留一个独立的测试集。每次预留出一段时间窗口的测试集。用当前的model去做预测，用预测集得到的预测误差作为评估度量。

2. 序贯预测。 对于每一个观测到的样本，首先只用它的属性值代入到学习的 model 进行预测，然后和观测到的值对比得到这个观测样本的误差。序贯预测误差等于每个观测样本的误差累计和：

$$S_n = \sum_{i=1}^{n} L(y_i - \hat{y_i})$$

还可以计算平均误差$M_n = \frac{S_n}{n}$，以及M_n的置信区间$M_n \pm \epsilon$

对于独立有界随机变量的独立和，可以使用 Hoeffding bound，确定置信区间：

$$\epsilon = \sqrt{\frac{Rlog(\frac{2}{\delta})}{2n}}$$

其中，R 是M_n的取值范围的宽度，δ是置信水平。

为了消除历史数据对M_n的影响，可以考虑使用加权的累积和：

$$E_i = \frac{S_i}{B_i} = \frac{L_i + \alpha \times S_{i-1}}{n_i + \alpha \times B_{i-1}}$$

上式中,$0 < \alpha < 1$。B_i代表加权的样本量， S_i代表加权的误差累计和。

本文用到的是聚类算法，对于上述数值型指标不适用。对于实际类别信息未知的聚类算法，可以采用 Calinski-Harabaz（CH）指标和轮廓系数（Silhouette Coefficient）进行评价。CH 指标通过计算类中各点与类中心的距离平方和来度量类内的紧密度，通过计算各类中心点与数据集中心点距离平方和来度量数据集的分离度，CH 指标由分离度与紧密度的比值得到。从而，CH 越大代表着类自身越紧密，类与类之间越分散，即更优的聚类结果。

轮廓系数最早由 Peter J. 等人提出，用于评价聚类的性质[21]。

对于数据集中的样本d_i，假设该样本被聚到某一类，则其轮廓系数s_i定义为：

$$s_i = \frac{a_i - b_i}{max(a_i, b_i)} \quad (12)$$

其中，a_i为样本d_i与其归属类的其他样本的平均距离，用于度量类内的凝聚度；b_i为样本d_i到其他所有类内样本的平均距离的最小值，度量了类的分离度。对于样本d_i所在类 k 的每一个样本点，计算该点对应的轮廓系数，取均值得到这一类的轮廓系数，即：

$$s_k = \frac{1}{n}\sum_{i=1}^{n} s_i \quad (13)$$

其中，n 为该数据集的样本数量，k 为聚类个数，s_k即为分为 k 类时对应的轮廓系数。轮廓系数越趋近于 1，表明聚类效果越好。

4. 实证比较数据流聚类算法

4.1 数据说明

为使本文对数据流聚类算法的比较更加具有客观性和说服力，选择“中国天气网”发布的 2019 年 5 月 2 日中国各地区的天气数据。数据共包含 2293 个中国地区，包含各个地区当天的最高温度（℃）、最低温度（℃）、还有该地区的经纬度。本文选择将最高温度和最低温度作为数据特征，分别采用 Leader、一次传递的 K-Means 算法、BIRCH 算法对各地区天气数据进行聚类。为了比较本文还将传统数据挖掘中的 K-Means 算法作为参照共同比较。

4.2 聚类效果对比

4.2.1 聚类算法效果比较

分别对各地区天气数据采用传统数据挖掘的 K-Means 算法，以及 Leader、一次传递的 K-Means 算法、BIRCH 算法对数据流进行聚类。为了具有可比性，本文将 Leader 算法、以及其它聚类算法分别调试参数使得聚类数目保持一致，为 12 类。最终几种聚类算法对应的聚类结果效果如图 1 所示。

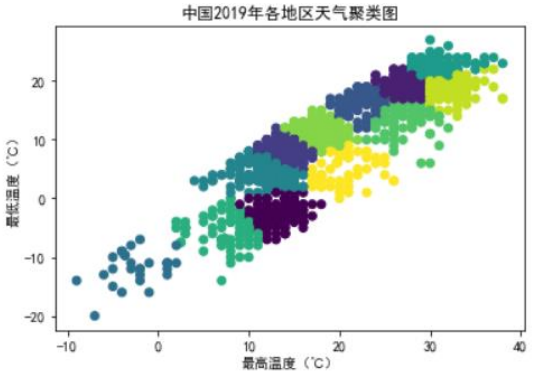

图 1-1 传统 K-Means 聚类效果

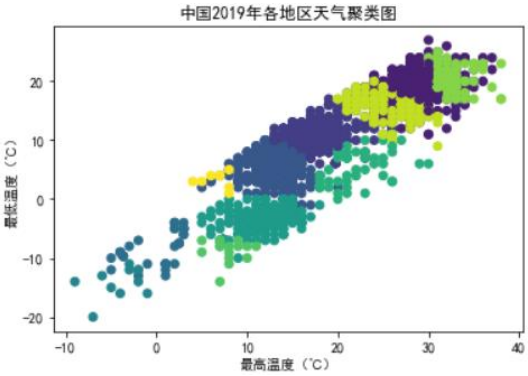

图 1-2 Leader 聚类效果

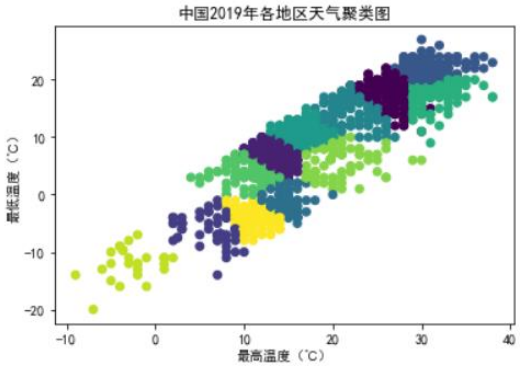

图 1-3 一次传递的 K-Means 聚类效果

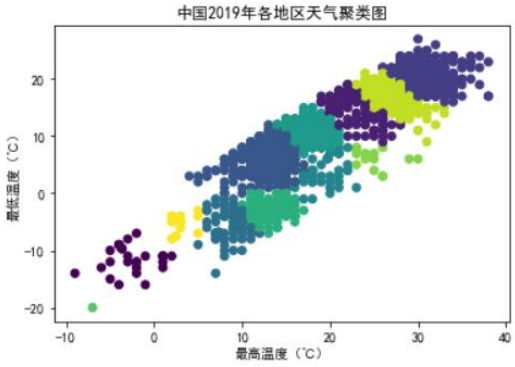

图 1-4 BIRCH 聚类效果

由图 1 可以看到，在三种数据流聚类算法中，一次传递的 K-Means 算法的聚类效果是最好的。表现为按照算法得到的数据之间的类别划分界限较为清晰。由于直观上看可能存在偏差，本文还采用聚类评价准则中的 Calinski-Harabaz 准则还有 Silhouette Coefficient 对几种聚类算法效果进行评价。聚类速度按照算法运用的时间作为衡量，最终得到聚类速度以及聚类效果见表 1。

表 1 聚类算法速度与效果比较

算法	K-Means	Leader（δ=9）	one pass K-Means	BIRCH (B=50, T=3)
$n_{cluster}$	12	12	12	12
time(s)	0.084	4.918×10^{-5}	0.656	0.064
Calinski-Harabaz	5390.975	2150.010	4998.523	3891.324
Silhouette Coefficient	0.366	0.161	0.341	0.377

由表 1 可以看到，在所有算法中，Leader 算法的聚类速度是最快的，所耗时长是其余算法的万分之一级别，但是 CH 指标和轮廓系数都是最低的，表明该算法的聚类的效果并不好。在几种数据流聚类算法中，一次传递的 K-Means 算法的聚类效果是最好的，但是算法的运行时间也是最长的，这是由于多次运行 K-Means 聚类需要占用一定时间。由于数据中包含了经纬度的数据，所以本文将聚类后的样本按照经纬度标注在图中，由颜色可以看到温度的差异。

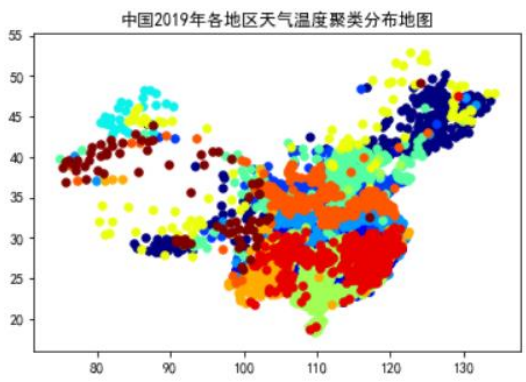

图 2-1 传统 K-Means 聚类地图

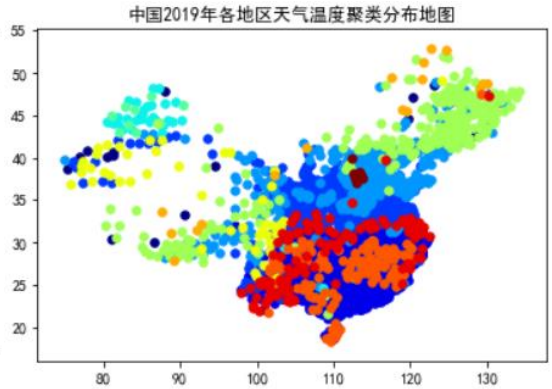

图 2-2 Leader 聚类地图

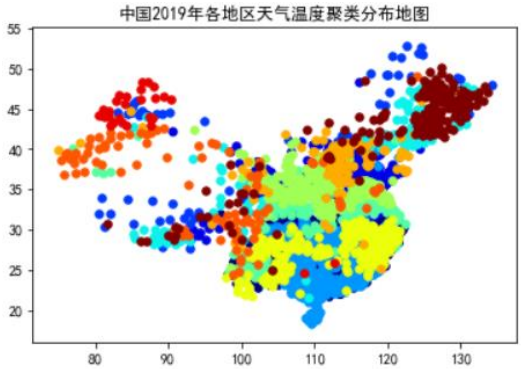

图 2-3 一次传递的 K-Means 聚类地图

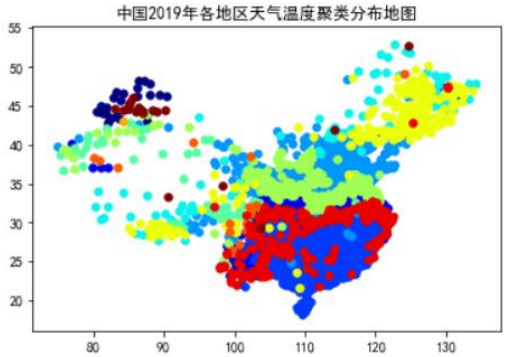

图 2-4 BIRCH 聚类地图

为了查看数据读入顺序的改变是否会对结果产生影响，本文将数据的读入顺序进行改变，并查看各算法的聚类效果。结果汇总如表 2。

表 2 改变数据读入顺序对聚类算法效果的影响（改变量）

算法	K-Means	Leader（δ=9）	one pass K-Means	BIRCH (B=50, T=3)
$n_{cluster}$	0	0	0	+5
time(s)	0	-0.158	0.033	-0.033
Calinski-Harabaz	-139.122	165.375	130.524	-122.295
Silhouette Coefficient	0	0.081	0.015	-0.015

可以看到，当数据的读入顺序发生改变时，数据流的聚类算法发生了明显的变化，这也符合数据流的特点。在 Leader、一次传递的 K-Means 算法、BIRCH 算法中，Leader 算法受数据读入顺序的影响最大，表现在 CH 指标和轮廓系数都发生了最大的变化。并且由表 1 可知，Leader 在这两个指标上的值都是最小的，但是当数据读入顺序发生了改变时却产生了最大的变化值。可以看出 Leader 算法的聚类效果很不稳定。

4.2.2 Leader 聚类算法特点

为了查看 Leader 聚类算法的特点，本文分别对阈值δ作不同的调整，分别查看聚类效果。

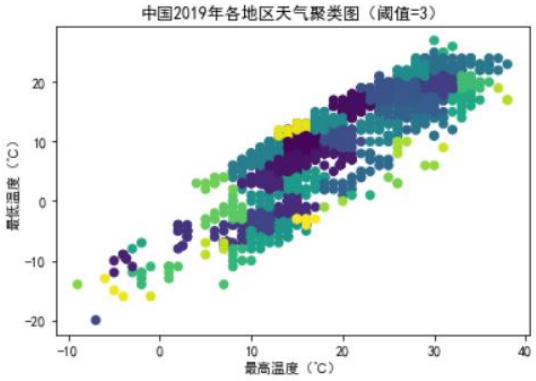

图 3-1 阈值为 3 时 Leader 聚类效果

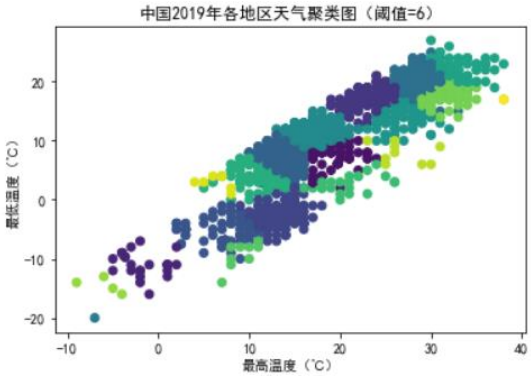

图 3-2 阈值为 6 时 Leader 聚类效果

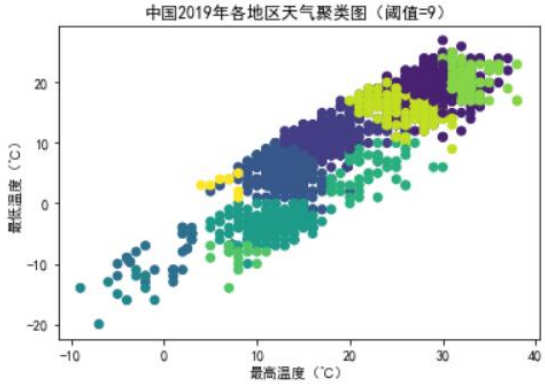

图 3-3 阈值为 9 时 Leader 聚类效果

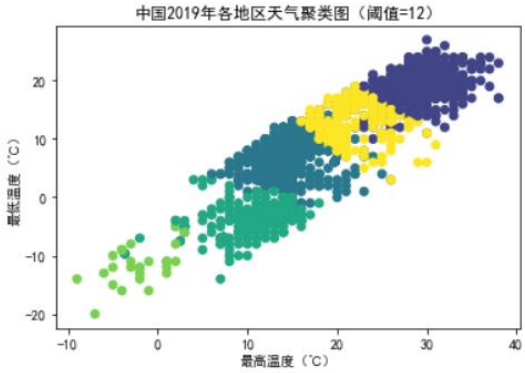

图 3-4 阈值为 12 时 Leader 聚类效果

表 3 对 Leader 算法调整阈值δ的聚类结果

阈值δ	3	6	9	12
$n_{cluster}$	56	20	12	6
time(s)	1.801×10^{-4}	7.329×10^{-5}	4.918×10^{-5}	3.409×10^{-5}
Calinski-Harabaz	2859.014	2833.330	2150.010	2835.537
Silhouette Coefficient	0.298	0.287	0.161	0.400

通过多次参数的调整，可以看到 Leader 算法在聚类的时间上具有很好地优势，但是在聚类的效果上并不好，并且由上文比较得到，该算法会随着数据的变化而变化，结果很不稳定。Leader 算法总结优点为：（1）只需要一次传递数据；（2）计算速度快；（3）不需要实现了解数据分为多少类。该算法的缺点为：（1）聚类结果不稳定；（2）且聚类的结果依赖于数据提供的顺序；（3）聚类结果依赖

于事先指定的阈值。

4.2.3 一次传递的 K-Means 聚类算法特点

为了查看一次传递的 K-Means 聚类算法的特点，本文分别对聚类数目 K 作不同的调整，分别查看聚类效果。结果如图 4

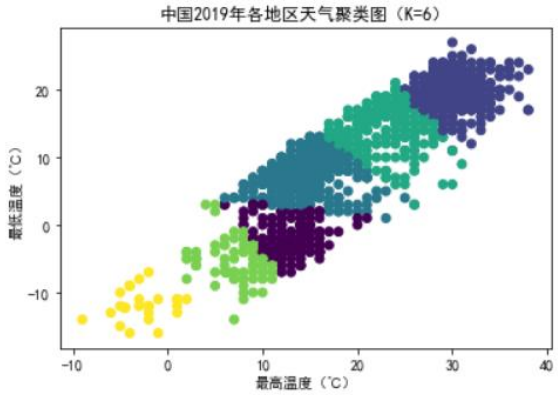

图 4-1 K=6 时 One Pass K-Means 聚类效果

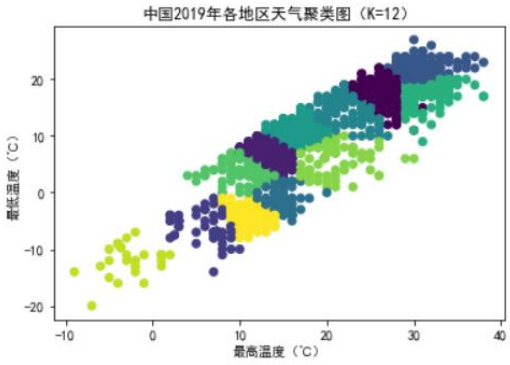

图 4-2 K=12 时 One Pass K-Means 聚类效果

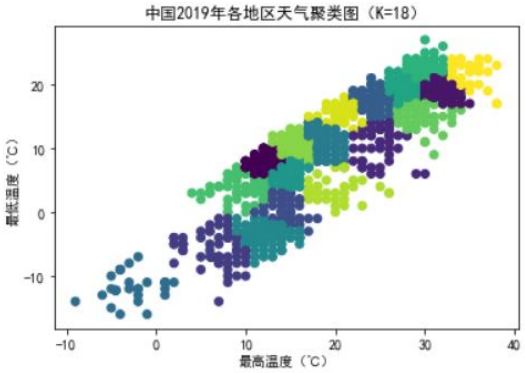

图 4-3 K=18 时 One Pass K-Means 聚类效果

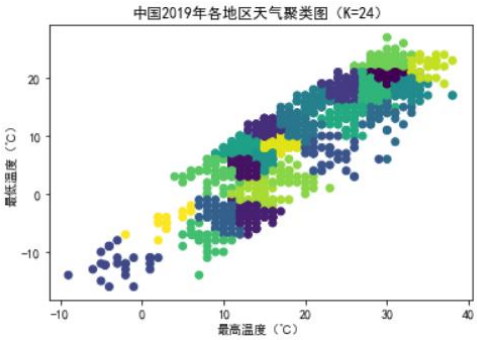

图 4-4 K=24 时 One Pass K-Means 聚类效果

表 4 对 One Pass K-Means 算法调整聚类数 K 的聚类结果

聚类数 K	6	12	18	24
time(s)	0.382	0.656	1.115	1.518
Calinski-Harabaz	5153.391	4998.523	5168.889	5009.451
Silhouette Coefficient	0.484	0.341	0.365	0.342

通过多次参数的调整，可以看到一次传递 K-Means 算法在聚类的时间并不占优势，这是由于该算法需要对每次读入的缓存区的数据进行 K-Means 聚类，占用了一定的时间。但是在聚类的效果上很好，CH 指标和轮廓系数的值都很高。总结一次传递 K-Means 算法的优点为：（1）只需要一次传递数据；（2）聚类效果

较好；（3）数据保存在指定内存的缓存区，占用的内存大小固定。该算法的缺点为：（1）需要提前确定聚类数目 K（2）相比其他聚类算法需要消耗更多的时长；（3）聚类的结果依赖于数据提供的顺序.

4.2.4 BIRCH 聚类算法特点

为了查看不同参数对聚类结果的影响，分别对参数进行调整并查看聚类时间和结果。其中每个叶子节点的最大的 CF 数 L 保持默认值 3 不变。改变参数 B，即每个内部节点的最大的 CF 数 B，用于控制内部节点的最大数量；还有参数 **Threshold** 是叶子节点每个 CF 的最大样本半径阈值，用于控制所有样本点要在半径小于 T 的一个超球体之内。

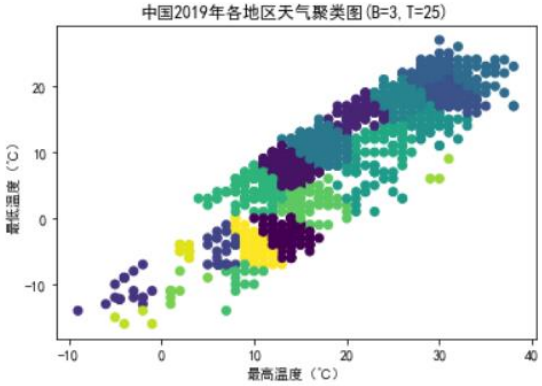

图 5-1 B=3, T=25 时 BIRCH 聚类效果

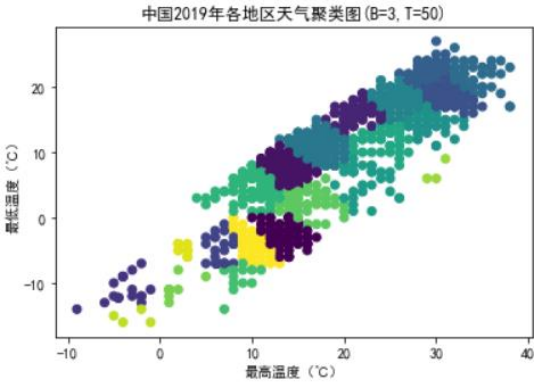

图 5-2 B=3, T=25 时 BIRCH 聚类效果

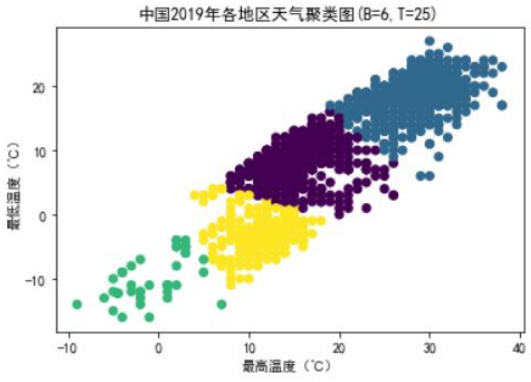

图 5-3 B=6, T=25 时 BIRCH 聚类效果

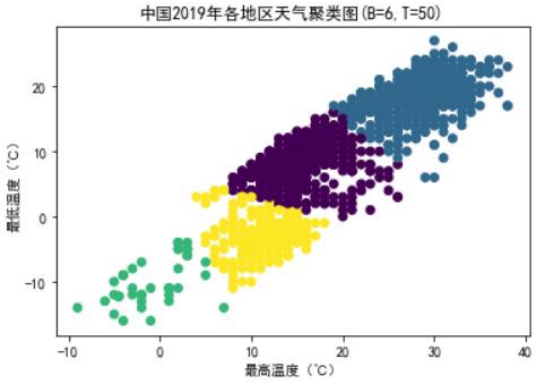

图 5-4 B=6,T=50 时 BIRCH 聚类效果

表 5 对 BIRCH 算法调整参数聚类结果

B	**3**		**6**	
Threshold	**25**	**50**	**25**	**50**
$n_{cluster}$	**21**	**12**	**4**	**4**
time(s)	**0.030**	**0.064**	**0.029**	**0.030**
Calinski-Harabaz	**4102.404**	**3891.324**	**5401.403**	**5401.402**
Silhouette Coefficient	**0.361**	**0.3771**	**0.587**	**0.587**

通过对参数进行调整并查看表中的聚类时间和结果。并根据上文 BIRCH 算法的表现，可以总结出 BIRCH 算法的优点为：（1）节省内存，所有的样本都在磁盘上， CF Tree 仅仅存了 CF 节点和对应的指针。（2）聚类速度快，只需要一遍扫描训练集就可以建立 CF Tree 并且 CF Tree 的增删改的速度快。（3）该算法可以识别噪音点，还可以对数据集进行初步分类的预处理。此外，结合本文实证总结 BIRCH 算法的缺点为：（1）由于 CF Tree 对每个节点的 CF 个数有限制，导致聚类的结果可能和真实的类别分布不同。(2)对高维特征的数据聚类效果不好。（3）如果数据集的分布簇不是类似于超球体，或者说不是凸的，则聚类效果不好。

5. 总结

本文对数据流聚类算法进行算法概述和聚类评价，并选择真实的数据进行实证分析，对 Leader 算法、One Pass K-Means 算法、BIRCH 算法进行比较。为了比较更加客观并避免偶然性，本文分别从聚类的时间和聚类的效果上算法进行比较，并查看数据的读入顺序对每种算法的聚类效果的影响，然后针对每一种聚类算法分别对该算法的参数进行调整，查看不同算法参数对该算法聚类效果的影响。最终分别从内存的占用、聚类时间消耗、聚类效果的稳定、参数的设定等方面对每种算法的优缺点进行分析和评价。根据本文对算法的评价，可以在不同的实际情况中选择合适的算法来进行数据流挖掘。