电 子 科 技 大 学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

专业学位硕士学位论文

MASTER THESIS FOR PROFESSIONAL DEGREE

论文题目 分布式原位聚类算法研究与实现

学科专业 计算机技术

学　　号 201722060921

作者姓名 陈远帆

指导老师 陈爱国　教授

分类号 _____ 密级 _____

UDC[注1] _____

学　位　论　文

分布式原位聚类算法研究与实现

（题名和副题名）

陈远帆

（作者姓名）

指导老师 陈爱国　教授

（姓名、职称、单位名称）

申请学位级别 专业硕士 学科专业 计算机技术

提交论文日期 _____ 论文答辩日期 _____

学位授予单位和日期 电子科技大学

答辩委员会主席 _____

评阅人 _____

注 1：注明《国际十进分类法 UDC》的类号。

摘　要

聚类分析是数据挖掘领域常用的算法之一，聚类算法能够将数据集分成若干个子集，使得子集内的元素彼此之间有着某种程度的相似性，而不同子集之间的元素则相似性较差；这种算法的输入是全体数据集，而随着大数据时代的到来，集中式数据存储方式已然暴露出越来越多的问题，分布式数据存储方式被广泛采用；在移动互联网时代，用户轨迹数据开始快速积累，如何在分布式环境下对轨迹数据进行分布式聚类计算成为亟待解决的问题。

本文提出的分布式轨迹聚类算法，针对网络带宽消耗、数据隐私性和聚类准确度三方面展开研究，主要研究内容如下：

（1）提出了基于复合采样的的分布式轨迹聚类算法——CSD-Clustering 算法。算法首先分析了当前分布式轨迹聚类算法存在的分布多样性问题；算法针对轨迹数据的特征采用了多项式拟合与最优化理论结合的方式对轨迹模型进行了拟合，这种轨迹模型拟合的思路能够很好保证全局聚类的准确度，也一定程度地保护了轨迹数据的隐私；除此之外，算法还对轨迹数据集进行了由轨迹数量抽样和轨迹点抽样组成的复合抽样方案，复合抽样方案能够有效减少网络传输的消耗。最后，通过仿真实验对算法的有效性和可行性进行了验证，实验结果表明，CSD-Clustering 算法能够准确完成分布式轨迹聚类计算任务，在隐私性和带宽消耗方面也得到了改善。

（2）提出了基于马尔科夫链的分布式轨迹聚类算法——MCD-Clustering 算法。算法首先分析了许多分布式聚类算法在处理高维数据时存在的问题；算法针对高维的轨迹数据中各个维度之间的联系，提出了利用马尔科夫链模型估计轨迹子簇的方案，这种思路对于拥有相似走势的轨迹簇能够很好的描述其整体数据分布，故算法为了得到拥有相似轨迹走势的轨迹子簇，在训练马尔科夫链模型之前进行了局部聚类操作；算法在网络中传输转移矩阵以及其他相关信息，并通过稀疏矩阵存储方式来存储转移矩阵，这种方式极大地缓解了网络传输的压力；最后通过实验对算法的有效性和可行性进行了验证，实验结果表明该算法在计算性能和数据隐私方面相对 CSD-Clustering 算法都有着一定程度的提升，但在聚类准确度方面略差于 CSD-Clustering 算法。

关键词： 分布式，聚类，数据隐私，轨迹数据，网络带宽消耗

ABSTRACT

Cluster analysis is one of the commonly used algorithms in the field of data mining. The clustering algorithm can divide the data set into several subsets, so that the elements within the subsets have some degree of similarity with each other, and between different subsets the similarity of elements is poor; the input of this algorithm is the entire data set, and with the advent of the era of big data, centralized data storage methods have exposed more and more problems, and distributed data storage methods are widely used; In the era of the mobile Internet, user trajectory data has begun to accumulate rapidly. How to perform distributed clustering calculation of trajectory data in a distributed environment has become an urgent problem.

The distributed trajectory clustering algorithm proposed in this paper conducts research on three aspects: network bandwidth consumption, data privacy, and clustering accuracy. The main contents are as follows:

(1) A distributed trajectory clustering algorithm based on composite sampling–CSD-Clustering algorithm is proposed. The algorithm first analyzes the distribution diversity problem of the current distributed trajectory clustering algorithm; the algorithm uses a combination of polynomial fitting and optimization theory to fit the trajectory model for the characteristics of the trajectory data. The idea of the trajectory model fitting methods can well ensure the accuracy of global clustering, and also protect the privacy of trajectory data to a certain extent. In addition, the algorithm also performs a composite sampling scheme on the trajectory data set consisting of trajectory number sampling and trajectory point sampling. The composite sampling scheme can effectively reduce the consumption of network transmission. Finally, the effectiveness and feasibility of the algorithm are verified by simulation experiments. The experimental results show that the CSD-Clustering algorithm can accurately complete the distributed trajectory clustering calculation task, and it also performs better in terms of privacy and bandwidth consumption.

(2) A distributed trajectory clustering algorithm based on Markov chains, the MCD-Clustering algorithm, is proposed. The algorithm first analyzes the problems existing in many distributed clustering algorithms when processing high-dimensional data. The algorithm proposes a scheme to estimate the trajectory sub-cluster using the Markov chain model for the connections between various dimensions in the high-dimensional trajectory

data. This idea can well describe the overall data distribution of trajectory clusters with similar trend. Therefore, in order to obtain trajectory subclusters with trajectories of similar trend, the algorithm performs a local clustering operation before training the Markov chain model. The transfer matrix and other related information are transmitted in the network, and the transfer matrix is stored by the sparse matrix storage method, which greatly relieves the pressure of network transmission. Finally, the validity and feasibility of the algorithm is verified by experiments. The experimental results shows that the algorithm has good performance in terms of data privacy, network bandwidth consumption. It has a certain degree of improvement over the CSD-Clustering algorithm in terms of computing performance and data privacy, but it is slightly worse than the CSD-Clustering algorithm in terms of clustering accuracy.

Keywords: Distributed, clustering, data privacy, trajectory data, network bandwidth consumption

目 录

第一章 绪 论

1.1 研究工作的背景与意义

1.1.1 研究背景

随着移动互联网时代的到来，数据规模开始成倍地增长，如何从大规模数据中挖掘出有价值的信息成为众多企业和机构需要思考的问题。聚类算法作为一种数据挖掘技术已被专业人员广泛地应用，聚类算法能够将大量无标签的数据划分成若干个簇，簇中的元素共同包含着某种隐性的特征，通过聚类算法，我们可以发现数据中隐藏着的内在规律，或能够将异常的噪音数据筛选出来；因为聚类算法输入的数据是无标签的，在机器学习中属于无监督学习中的一种。

随着数据规模的逐步增大，数据的集中式存储和管理由于其存在存储能力限制、计算能力限制以及安全方面的考虑，已经不能满足人们的需求，分布式数据库的使用变得日益广泛，在分布式环境下如何进行数据挖掘操作则称为了新的受到人们广泛讨论的话题。针对地理分布多中心联合分析场景，2010 到 2011 年，惠普创新研究计划资助了 G-Hadoop 项目[1]，旨在开发跨地理分布的分布式 Hadoop 框架和 MapReduce 软件框架，以支持数据密集型应用，其尽量用“移动计算”取代“移动数据”，通过在多个集群使用数据并行处理范式，实现可以同时在多个计算中心运行，而无需复制数据。G-Hadoop 需要系统下的所有集群都基于 Hadoop 构建，没有严格限制原始数据的共享。在 2012 年 Jeff Dean 在论文 [2] 中，引入 data parallelism(数据并行) 的概念，其早期概念是用于模型训练中，即把训练数据拆分成多份，用每一份独立的训练局部模型，并将多个局部模型之间做综合同步，得到全局模型，通过多轮迭代，逼近最优的全局模型。此外，也有从数据隐私的角度，提出跨域数据联合使用的问题，最典型的是 2016 年，Google 提出了联邦学习架构，至此，基于跨地理中心的数据联合分析，成为研究的热点。

对于分布式环境下的聚类算法，需要将数据在网络中进行传输，如果处理不当将导致用户隐私数据的泄漏，有相关调查也对用户的隐私意识进行了报到，调查显示, 约 17% 的用户拒绝向网站提供个人信息, 而约 56% 的用户在得知网站提供严格的保密手段后愿意网站获取自己的信息[3]。除此之外, 对于需要多个机构联合数据进行操作时，各方往往因为数据隐私的问题而不愿共享数据，这使得数据没能得到充分地利用，很多操作也没办法完成[4] [5]。

由此可见，跨属地多中心数据的联合使用，即是面向处理性能，也是面向数据本身的隐私泄露风险问题，在分布式式计算架构研究中，核心面临着计算结果

可信、参数共享过程安全、多方激励等诸多问题亟待解决。由于不同类型的计算，涉及到不同的数据关联难题，聚类分析，是其中最难解决的问题。

1.1.2 研究意义

分布式聚类旨在从大规模的、分布式存储的数据中挖掘出分类模型。针对分布式聚类算法，目前业内的解决思路是：首先在各个计算子节点上进行局部聚类操作，计算子节点将各自计算的结果进行简答处理后发送给中心节点，中心节点依据各个计算子节点传输的局部聚类结果进行全局聚类操作，然后将全局聚类的结果传输给各个计算子节点，计算子节点依据中心节点传输过来的全局聚类结果则可以进行进一步的操作，比如异常判断。计算子节点传输给中心节点的数据一般由统计数据和一小部分代表向量组成，由于代表向量不可能代替本地所有元素信息，这也使得全局聚类的结果准确性不够高；如果代表向量占本地数据比例越来，理论上全局聚类的结果准确度会更高，但这同时给带宽消耗和隐私保护带来了一定的压力。

如上所述，如何在同时考虑带宽消耗和准确度的情况下有效完成分布式聚类算法是目前的研究难点，而且随着移动互联网的普及，用户对于自身隐私问题也越来越关注，分布式环境下，隐私保护问题已成为研究人员不得不考虑的问题。针对上述分布式聚类面临的技术挑战，本文开展从以下三个方面开展研究内容：

1. 为了解决分布式聚类的准确度和带宽消耗之间的矛盾，需要探索一种分布式环境中在保证聚类精度的同时尽可能的降低带宽消耗，在有限网络带宽资源情况下实现高准确度的分布式聚类算法。
2. 由于数据量的大规模增长和用户隐私数据的敏感性，网络中传输原始用户数据成为一种非常危险的做法，对于分布式环境中，各个数据中心位于不同地理位置，如何在不传输原始用户数据情况下完成分布式聚类操作是亟待解决的问题。本文将针对这个问题展开研究，旨在探索一种考虑用户隐私保护，同时具备高准确度的分布式聚类算法。

1.2 国内外研究现状

1.2.1 分布式聚类算法研究现状

在分布式聚类算法研究方面，一种传统的分布式密度聚类方法 Basic-DDP[6]，将原始数据均匀划分为 w 份不相交的子集发送至各个子节点进行处理，同时在每一个子节点缓存一份完整的数据集，该方法提高了聚类计算过程中的并行性，但执行聚类算法之前需要将全局数据聚集在一起在实现特定的数据分发，在数据聚

集阶段将产生大量的网络流量消耗，而且也缺乏对数据隐私性的考虑。Januzaj 等人基于传统的 DBSCAN 算法，提出了一种基于密度的分布式聚类算法[7]，该算法主要分为三个步骤：局部聚类、全局聚类和局部聚类模型更新，在局部聚类阶段，各个计算节点首先针对各自本地数据，在设定邻域半径和最小密度的基础上进行 DBSCAN 聚类算法，DBSCAN 聚类算法将输出核心点集合，从核心点集合中选出具有代表性的原型向量，将其传输给中心节点；在全局聚类阶段，中心节点依据各个计算节点传输的原型向量进行全局聚类，将全局聚类结果传输给所有计算节点；最后，各计算节点依据中心节点回传的全局聚类结果更新聚类模型。文献 [8] 提出了一种基于 k-means 的分布式聚类算法，在该算法中 K 个节点首先计算本地数据的中心向量，然后各节点将自己的中心向量广播给其余的 K-1 各节点，这样每个节点将受到 K 各中心向量，各个节点将本地所有数据与 K 各中心向量进行比较，并将其发送到离该数据最近的中心向量对应的节点。这种算法其准确性与单节点聚类结果一致，但通信开销很大。文献 [9] 针对分布式 k-means 算法通信开销大的问题提出了改进算法——DK-means 算法，该算法只需在网络中传输簇心向量和簇的数据个数，大大减少了网络带宽的压力。文献 [10] 提出一种新的数据集表达方式——特征向量，主要通过坐标和密度来描述某一密度空间，以较少的数据量反映站点数据的分布特征，并在此基础上提出了一种基于特征向量的分布式聚类算法 DCBFV。该算法对于不同的输入参数导致不同的聚类结果，而且当节点数据量很大时，中心节点的计算能力将成为算法的瓶颈。

在隐私保护层面，很多学者针对分布式聚类做出了尝试。文献 [11] 提出了一种考虑隐私保护的分布式聚类算法，在该方算法中，本地节点通过估计概率密度模型来模拟本地数据的生成模型，然后仅在网络中传输生成模型的参数给中心节点，中心节点可以通过本地生成模型参数人工生成数据，然后利用生成的数据对全局概率密度模型进行估计。文献 [12] 提出了一种适用于数据垂直切分的聚类算法——k-nearest 算法，这种算法基于同态加密和随机扰动技术设计了一个安全协议，算法利用半可信第三方参与下的安全求平均值协议，实现了在分布式数据中进行 k-means 聚类挖掘时隐私保护的要求。杨林等人[13] 针对基于欧几里得距离的聚类分析隐私保护问题, 提出了一种新的隐私保护方法。该方法将安全多方计算协议运用于水平分布和垂直分布两种数据模型上, 使得对该两种数据模型进行聚类分析时既满足了保护隐私的前提, 又保证了数据间欧几里得距离不变 (即挖掘结果的准确性)。文献 [14] 提出的 PPDK-means 算法是针对水平划分的分布式数据库的。PPDK-means 基于 K-means 的基本思想，对数据进行分布式聚类，在聚类的过程中引入半可信第三方，并应用安全多方技术保护本站点真实数据不被传送到其他

站点，以达到隐私保护的目的。从多属地数据隐私保护的角度，Klusch 等人提出一种基于密度的分布式聚类方法 KDEC[15]，在此方法中，各个计算节点利用非参数核密度估计来对本地数据分布进行刻画，将局部概率密度模型传输给中心节点，中心节点依据局部概率模型通过累加操作得到全局概率密度模型，并将其模型广播给各计算节点，计算节点依据全局概率密度模型来实现聚类操作，该算法在网络中传输的是概率密度模型，对隐私保护起到了一定的作用。Patel 等人基于椭圆曲线密码（ECC）提出了一种用于水平切分的分布式 k-means 算法[16]，这种方法避免了在每个站点进行多次加密操作，因此在计算成本方面方面表现不俗，算法采用环形拓扑进行通信，降低了通信成本。

1.2.2 轨迹聚类算法研究现状

同时，由于轨迹数据的复杂性，以上很多分布式聚类方法不能直接扩展到在分布式轨迹聚类应用中。

针对轨迹聚类，其距离度量对于聚类结果至关重要，除了常用的欧氏距离可以自然地扩展到轨迹距离计算上，文献[17]提出了 PCA+ 距离度量方式，这种距离度量方式相比于欧式距离有更好的抗噪性，但两条轨迹长度必须一致。文献[18]提出了 Hausdorff 距离，这种度量适用于长度不相等的序列之间的距离计算，然后对于噪声数据十分敏感。Rick 等人提出来了 LCSS 距离度量方式[19]，这种度量方式允许轨迹之间存在一定程度的位置偏移。

在轨迹聚类研究方面，Gaffney 等人根据统计分析中的期望最大化原则提出了一种混合回归模型[20]。Chudova 等人在此研究基础上提出了一种新的混合模型[21]，模型中的参数采用对象的空间和时间偏移综合计算得出，更适应于曲线聚类。Alon 等人通过动态时间序列对轨迹数据中含有的不同类簇进行分析，通过对象在相邻两个位置转换的马尔可夫模型来表示轨迹簇[22]。Nanni 等人根据轨迹时间属性上的差异提出了一种专注于时间的移动对象轨迹聚类算法[23]，该算法根据轨迹时间段的不同分别对轨迹数据进行轨迹空间相似度度量。Birant 等人针对传统 DBSCAN 算法的不足，提出了一种适用于时空轨迹数据的改进 DBSCAN 算法[24]。Palma 等人通过基于密度的轨迹聚类算法发现热点位置[25], 他通过自定义距离度量利用 DBSCAN 算法找到轨迹中停歇状态和运动状态。Jeung 等人使用基于密度的轨迹聚类算法实现了运动物体群体模式的发现[26]，群体模式是许多运动物体同时运动一段时间的活动模式。Hung 等人提出了一种框架，这种框架用于发现代表用户频繁运动行为的轨迹路线[27]。由于轨迹数据长度一般较长，对于以整个路径为基本单位的聚类算法可能会引发如下两个问题：（1）复杂轨迹中的局部特征可能

会被忽略；（2）找不到轨迹的公共子模式。一些研究者针对上述问题提出了基于分段重组的聚类算法。Lee 等人提出来了轨迹聚类的分段和重组框架[28]，框架中定义了一种基于最小描述长度（MDL）理论的形式化轨迹划分算法，并提出了子轨迹聚类算法。

Buchin 等人[29]提出了一系列的时空标准，比如速度、航向、曲率等，在这些标准下提出了一种算法框架，该框架允许用户将轨迹分段成最小数量的片段。以上相关研究，都是面向集中处理场景的。

1.2.3 分布式轨迹聚类研究现状

针对分布式轨迹聚类的研究十分有限，Fan 提出了一个更为通用的行为模式定义，并且基于聚类和 Apriori 算法，实现了分布式系统环境下的 SPARE 框架[30]，能够将时空轨迹数据在新定义的行为模式上进行类型划分，但是这个框架是以全局数据的聚集为前提，对全局数据进行特定的数据划分后执行并行计算实现分布式行为模式挖掘没有考虑隐私安全的问题。文献 [31] 提出了一种用于时空轨迹数据流的分布式聚类算法 DTC，在文中提出了一种基于多运动特征的轨迹划分方法和包含了位置、中心、形状、方向、速率和油耗等六个特征量相似性的轨迹结构相似性度量方法。Wang 等人提出一种基于轨迹数据密度分区的分布式并行聚类方法[32]。首先将整个轨迹数据集抽象在一个矩形区域内，通过该矩形最长维度的变换将数据合理地划分为若干任务量相当的分区，构建可供分布式并行聚类的局部数据集，然后各工作服务器对局部分区分别执行 DBSCAN 聚类算法，管理服务器对局部聚类结果进行合并与整合。Mao 等人以减少通信开销提高分布式轨迹流聚类效率为目标提出了一个在线处理分布式轨迹数据流的增量聚类算法(OCluDTS)[33]。OCluDTS 方法使用基于滑动窗口模型的两层分布式框架，通过多个远程节点并行聚类局部轨迹流以及协调者节点合并局部聚类结果的方式，确保分布式轨迹流聚类获得与集中式方法相同的精度. 此外，为了进一步降低 oCluDTS 算法的总执行开销，提出了仅限于聚类更新的远程节点传输聚类结果给协调者节点以及基于协调者节点相似性计算的剪枝策略等优化措施。Deng 等人提出了一种轨迹大数据聚类算法 Tra-POPTICS[34],Tra-POPTICS 可以以分布式方式处理轨迹大数据，以满足较好的可扩展性。为了提供一种对轨迹大数据进行聚类的快速解决方案，文章还探索了在图形处理单元（GPGPU）上使用当代通用计算的可行性，使得提出的算法在对轨迹大数据进行聚类时具有很好的可扩展性和计算性能。

1.3 研究目标与研究内容

本文主要研究分布式轨迹聚类算法，为了在不泄露用户隐私的情况下，高效、准确的实现分布式轨迹聚类计算，提出了基于复合采样的分布式轨迹聚类算法和基于马尔科夫链的分布式轨迹聚类算法，以下为主要研究内容：

（1）局部聚类与全局聚类相结合的分布式算法在计算准确度表现不太稳定，造成这种不稳定的原因是这类方法在网络中传输的数据结构与数据的真实分布不是一一映射的关系，一个计算节点利用局部聚类结果和统计信息组成的数据结构可能对应多种数据分布，而这种映射到数据分布的多样性将给后续的全局聚类造成影响；本文针对当前分布式轨迹聚类算法存在的分布多样性问题提出解决方案。

（2）对于高维数据，由于训练数据在高维空间上过于稀疏的分布，很多基于统计学的传统方法将不再适用，文本针对轨迹数据中各个点之间存在关联的特点，提出一种适用于高维轨迹数据的分布式聚类算法。

1.4 本论文的结构安排

本文围绕分布式轨迹聚类的关键技术展开研究，全文结构如下：

第一章，绪论。首先论述了本课题的相关背景和研究意义然后介绍了当前国内外研究现状，最后给出了本文的主要研究内容与研究目标以及论文的后续章节安排。

第二章，相关理论和技术。首先介绍了聚类算法，关于聚类算法的评价指标也进行了介绍。然后就多项式拟合理论基础和最优化理论进行了介绍与分析，最后对马尔可夫链模型进行了相关介绍为后续研究的进一步展开提供理论依据。

第三章，基于复合采样的的分布式轨迹聚类算法。首先对当前分布式轨迹聚类方案在准确度上的不足进行了分析，随后从聚类准确度、隐私保护和带宽消耗的角度出发，建立了基于复合采样的的分布式轨迹聚类算法。最后对该算法进行了相关实验，并验证了该算法的准确性和有效性。

第四章，基于马尔科夫链的分布式聚类算法。首先分析了传统分布式轨迹聚类算法存在的缺点，以及第三章中存在的网络负载过重的问题进行了分析，然后马尔科夫链模型建立局部轨迹数据生成模型。最后利用相关数据集进行实验验证证明了该算法的有效性。

第五章分布式轨迹聚类系统的设计与实现。首先对分布式计算系统体系架构进行介绍，给出了系统的总体架构设计并分别对系统中的重点模块进行了详细描述，展示了系统相应的功能。

第六章，总结与展望。对全文工作进行总结，并探讨未来的研究方向。

第二章 相关理论与技术

2.1 概论

本章对后续章节所用到的相关理论和技术进行简介，主要包括常用的聚类算法（k-means 算法和 k-medoids 算法）以及聚类算法的评估标准、稀疏矩阵的存储方式和多项式拟合技术，对最优化理论进行了简单的阐述，最后简单介绍了马尔科夫链模型。这些理论为后续研究提供了理论机基础。

2.2 聚类算法

聚类算法是一种无监督学习算法，及数据集中的每个数据项不带有标签信息，通过统计分析技术将数据集分成不同的子集或簇，使得分到同一子集或簇的成员对象都有某种相似的特征。这种技术被广泛应用与各个领域，包括机器学习、数据挖掘、模式识别等。

2.2.1 k-means 算法

k-means 算法由文献 [35] 正式提出，现已成为数据挖掘中常用的聚类算法，其算法主要求解思路是一种启发性算法，求解过程利用了 EM 算法[36]，通过不断迭代优化直到满足停止迭代的条件。

2.2.1.1 算法流程及复杂度分析

现有样本集 $D = \{x_1, x_2, ..., x_m\}\}$，K-Means 算法将数据集划分成互不相交的 K 个簇 $C = \{C_1, C_2, ..., C_K\}\}$, 且满足 $C_1 \bigcup C_2 \bigcup ... \bigcup C_m = D$，每个簇的簇心 $\mu_i = \frac{1}{|C_i|} \sum_{\boldsymbol{x} \in C_i} \boldsymbol{x}$。其流程如下：

算法 2-1 k-means 算法

Data: 样本集 $D=\{x_1,x_2,...,x_m\}\}$，簇个数 K

Result: 簇划分 $C=\{C_1,C_2,...,C_k\}\}$

从数据集 D 中随机初始化 K 个样本，将其作为 K 个初始簇心;

repeat

  将所有的簇置为空，$C_i=\varphi\ \ (1\leqslant i\leqslant K)$

  for *j=1,2,...,m* **do**

    计算样本 x_i 与所有簇心之间的距离，并将 x_i 归为第 α_i 类，使其满足：$\alpha_i=arg\ \min(d_{i\alpha_i})$

    其中：$d_{i\alpha_i}=||x_i-\mu_{\alpha_i}||_2$ 将样本 x_i 划入簇 C_{α_i}

  end

  for *i=1,2,...,K* **do**

    $\hat{\mu}_i=\frac{1}{|C_i|}\sum_{x\in C_i}x$

    if $\hat{\mu}_i\neq\mu_i$ **then**

      μ_i=$\hat{\mu}_i$

    end

  end

until 所有簇心不再更新或更新值小于一定阈值;

从上述流程可以得到，k-means 算法复杂度为 O(n*k*t)，其中 n 表示样本数目，k 表示簇个数，t 是平均迭代次数。k-meas 算法的收敛性可参考论文 [37]。

2.2.2 k-medoids 算法

k-means 算法要求元素维度相同且距离度量必须基于欧式距离，如果需要对不规整的数据进行聚类，k-means 算法显然无法解决此类聚类问题。针对不规整的数据，可以采用 k-medoids 算法[38] 进行聚类。k-medoids 算法只需输入距离矩阵即可，所以对距离度量方法也更加宽泛。k-medoids 流程如下：

算法 2-2 k-medoids 算法

Data: 样本距离矩阵 M_{nn}，簇个数 K

Result: 簇划分 $C=\{C_1,C_2,...,C_k\}\}$

随机初始化 K 个样本，将其作为 K 个初始簇心;
repeat
  将所有的簇置为空，$C_i=\varphi\ \ (1\leqslant i\leqslant K)$
  for *j=1,2,...,m* **do**
    依据距离矩阵，计算序号为 i 的样本与所有簇心之间的距离，并将序号为 i 的样本归为第 α_i 类，使其满足：$\alpha_i=arg\ \min\left(M[ialpha_i]\right)$
  end
  for *i=1,2,...,K* **do**
    $\hat{\mu}_i=arg\ \min_{x\in C_i}\left(\sum_{j\in C_i}M[x,j]\right)$
    if $\hat{\mu}_i\neq\mu_i$ **then**
      μ_i=$\hat{\mu}_i$
    end
  end
until 所有簇心不再更新或更新值小于一定阈值;

从上述流程可以得到，k-medoids 算法复杂度同 k-means 一样为 O(n*k*t)，其中 n 表示样本数目，k 表示簇个数，t 是平均迭代次数。

2.2.3 聚类评估指标

聚类评估指标可以分为两类[39]，一类是将聚类结果与某个“参考模型”进行比较，度量这种比较结果的评估指标称为“外部指标”；另一类是直接评估聚类效果本身，不参考任何其他模型，这种评估指标称为“内部指标”。

2.2.3.1 外部指标

假设通过聚类算法将数据集 D 划分成簇集 $C=\{C_1,C_2,...,C_K\}\}$，参考模型将数据集 D 划分的簇集 $C^*=\{C_1^*,C_2^*,...,C_K^*\}$，相应地，令 λ 和 λ^* 分别表示与 C 和

C^* 对应的簇标记。我们将样本两两配对，定义：

$$\begin{aligned} a &= |SS|, \quad SS = \left\{(x_i, x_j) \,|\lambda_i = \lambda_j, \lambda_i^* = \lambda_j^*, i < j\right)\} \\ b &= |SD|, \quad SD = \left\{(x_i, x_j) \,|\lambda_i = \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j\right)\} \\ c &= |DS|, \quad DS = \left\{(x_i, x_j) \,|\lambda_i \neq \lambda_j, \lambda_i^* = \lambda_j^*, i < j\right)\} \\ d &= |DD|, \quad DD = \left\{(x_i, x_j) \,|\lambda_i \neq \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j\right)\} \end{aligned} \tag{2-1}$$

其中集合 SS 包含了在 C 中隶属于相同簇且在 C^* 中也隶属于相同簇的样本对，集合 SD 包含了在 C 中隶属于相同簇但在 C^* 中隶属于不同簇的样本对，..... 由于样本对 $(x_i, x_j)\,(i < j)$ 仅能出现在一个集合中，因此有 $a + b + c + d = m(m-1)/2$ 成立。

基于2-1，可引出 Rand 指数的定义：

$$\mathrm{RI} = \frac{2(a+d)}{m(m-1)} \tag{2-2}$$

2.2.3.2 内部指标

若直接评估聚类结果本身，则需要考虑内部指标，同样设定聚类算法得到的簇划分为 $C = \{C_1, C_2, ..., C_K\}$。在定义内部指标之前，先定义几种距离公式：

$$\begin{aligned} avg(C) &= \frac{2}{|C|(|C|-1)} \sum_{1 \leqslant i < j \leqslant |C|} dist\,(x_i, x_j) \\ diam(C) &= \max_{1 \leqslant i < j \leqslant |C|} dist\,(x_i, x_j) \\ d_{\min}\,(C_i, C_j) &= \min_{x_i \in C_i, x_j \in C_j} dist\,(x_i, x_j) \\ d_{cen}\,(C_i, C_j) &= dist\left(\mu_i, \mu_j\right) \end{aligned} \tag{2-3}$$

其中，dist(.) 用于计算两个样本之间的距离；μ 代表簇 C 的中心点。avg(C) 对应于簇 C 内样本间的平均距离，diam(C) 对应于簇 C 内样本间的最远距离，$d_{\min}\,(C_i, C_j)$ 对应于簇 C_i 与簇 C_j 最近样本间的距离，$d_{cen}\,(C_i, C_j)$ 对应于簇 C_i 与簇 C_j 中心点间的距离。

基于2-3可引出以下两种常用内部指标：

$$DBI = \frac{1}{k} \sum_{i=1}^{k} \max_{j \neq i} \left(\frac{\mathrm{avg}\,(C_i) + \mathrm{avg}\,(C_j)}{d_{\mathrm{con}}\left(\mu_i, \mu_j\right)} \right) \tag{2-4}$$

$$DI = \min_{1 \leqslant i \leqslant k} \left\{ \min_{j \neq i} \left(\frac{d_{\min}\,(C_i, C_j)}{\max_{1 \leqslant l \leqslant k} \mathrm{diam}\,(C_l)} \right) \right\} \tag{2-5}$$

DBI 的值越小表示聚类效果越好，DI 相反，其值越大表示聚类效果更好。

2.3 矩阵稀疏存储格式

计算机经常会对矩阵形式的数据进行处理，在实际场景中处理的矩阵往往包含很多零元素，大部分元素为零元素的矩阵称为稀疏矩阵，对于稀疏矩阵，可以通过特定的存储方式来代替矩阵格式的存储方式，可以大大减少存储的压力。

2.3.1 COO 存储格式

COO 存储格式对矩阵中的非零元素的坐标和取值进行存储，利用 row 向量、column 向量和 value 向量来表示稀疏矩阵中的非零元素。假设现在有矩阵 T，其内容如下：

$$T = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 \\ 2 & 0 & 0 & 0 \end{bmatrix}$$

利用 COO 存储格式则将矩阵 A 转换成如下三个向量：

row	*column*	*value*
0	2	1
1	0	3
2	3	4
3	0	2

利用坐标和取值来存储稀疏矩阵还有另一种方式：DOK（Dictionary of keys），只不过是通过键值对来表示非零元素的坐标和取值的，上式的矩阵 T 可以通过如下键值对来表示：

key	*value*
$(0,2)$	1
$(1,0)$	3
$(2,3)$	4
$(3,0)$	2

2.3.2 CSR 存储格式

CRS 存储是基于 COO 存储的改进[40]，CRS 通过三个向量来存储稀疏矩阵：A,IA 和 JA。A 向量是存储矩阵所有非零元素的数组，元素顺序是行优先，即从左到右、从上往下，如图2-1所示，IA 向量中的元素是递推得到的，IA 元素的个数为行数加一，递推关系如下：

$$IA[0] = 0$$
$$IA[i] = IA[i-1] + i\text{行非零元素的个数} 0 < i < m$$

第三个向量 JA 是 A 中每个元素的列索引，同样针对稀疏矩阵 T，我们利用 CRS 存储方式可以表示为：

$$A = [1,3,4,2]$$
$$IA = [0,1,2,3,4]$$
$$JA = [2,0,3,0]$$

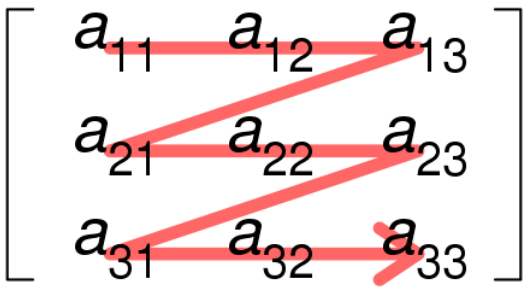


图 2-1 CRS 存储元素的顺序

2.4 多项式拟合

轨迹数据中的每条轨迹一般是由若干个点组成，有时需要通过这若干个点来拟合轨迹曲线，本文使用多项式来实现轨迹的拟合。

2.4.1 理论基础

在二维空间里，一条轨迹由 n 个点组成：$\{(x_1,y_1),(x_2,y_2),\ldots,(x_k,y_k)\}$，若轨迹对应的函数为 $f(x)$，多项式拟合则试图通过多项式来近似 $f(x)$，n-1 阶多项式的形式为 $g(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$。

若通过 n-1 阶多项式来对 n 个坐标点对应的轨迹进行拟合，则将 n 个点的坐标

带入 n-1 阶多项式可得到 n 个等式，将这 n 个等式通过矩阵表示则为：

$$\begin{bmatrix} 1 & x_1 & \dots & x_1^{n-1} \\ 1 & x_2 & \dots & x_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \dots & x_n^{n-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{2-6}$$

记 $A = \begin{bmatrix} 1 & x_1 & \dots & x_1^{n-1} \\ 1 & x_2 & \dots & x_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \dots & x_n^{n-1} \end{bmatrix}$，$|A^T|$ 则为范德蒙德行列式，故有：

$$\left|A^T\right| = \prod_{p>q} (x_p - x_q) \tag{2-7}$$

若 $x_0, x_1, , x_n$ 彼此不相等，故此时存在唯一一个 n-1 次阶多项式能够包含所有数据点，此时用 n-1 阶多项式拟合的函数可以经过每一个点。

2.4.2 麦克劳林级数及其收敛域

若函数 f(x) 在 x_0 的某一个邻域内具有任意阶导数,则幂级数 $\sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!} (x - x_0)^n$ 称为 f(x) 在 x_0 处的泰勒级数；当 $x_0 = 0$ 时，此幂级数则称为麦克劳林级数，即 $\sum_{n=0}^{\infty} \frac{f^{(n)}(0)}{n!} (x)^n$。

假设 f(x) 在 $x_0 = 0$ 处的麦克劳林级数收敛于 f(x)，即当 $n \to \infty$ 时，f(x) 在 $x_0 = 0$ 处的麦克劳林公式的余项 $R_n(x)$ 对邻域中的点趋于 0。此时，则有：

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(0)}{n!} (x)^n \tag{2-8}$$

等式2-8右侧是幂级数，关于幂级数的收敛半径满足阿尔贝定理。

阿贝尔定理：如果幂级数 $\sum_{n=0}^{\infty} a_n x^n$ 当 $x = x_0 (x_0 \neq 0)$ 时收敛，则对于一切满足 $x < |x_0|$ 的 x，幂级数 $\sum_{n=0}^{\infty} a_n x^n$ 均绝对收敛；如果幂级数 $\sum_{n=0}^{\infty} a_n x^n$ 在 $x = x_1$ 处发散，则对于一切 $x > |x_1|$ 的 x，幂级数 $\sum_{n=0}^{\infty} a_n x^n$ 均发散。

因此我们依据2-8和阿贝尔定理可以得到函数 f(x) 其麦克劳林级数的收敛域。

2.5 最优化理论

很多机器学习问题在求解过程中面对的都是在指定约束条件下的最优化问题，其一般形式如下：

$$minimize \quad f_0(x)\, s.t. \quad f_i(x) \le 0, \quad i = 1, ..., m$$

其中 $f_0(x)$ 称为目标函数，$f_i(x) \le 0, \quad i = 1, \ldots, m$ 称为约束函数。若优化问题没有任何约束函数，则称此优化问题为无约束优化问题。

2.5.1 经验误差与正则化

在机器学习中，优化目标一般称为代价函数，代价函数一般与所训练的模型 $\hat{f}(x)$ 在训练集上的误差相关，学得的模型 $\hat{f}(x)$ 在训练集上的误差称为经验误差，经验误差是模型 $\hat{f}(x)$ 关于训练集的平均损失：

$$R_{\text{emg}}\left(\hat{f}\right) = \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, \hat{f}(x_i)\right)$$

其中，$L\left(y_i, \hat{f}(x_i)\right)$ 表示在样本 x_i 上，模型 $\hat{f}(x_i)$ 来判断输出的损失度量。

如果只追求训练得到的模型 $\hat{f}(x)$ 使得经验损失尽可能得小，意味着模型 $\hat{f}(x)$ 在训练集上表现得很好，但是往往会因为模型 $\hat{f}(x)$ 的复杂度过高而导致对未知数据的预测能力很差，即所谓的泛化能力很差，这种现象称为过拟合，为了防止过拟合现象的出现，人们往往给经验损失添加一个正则项，这种优化经验损失和正则项的方法称为正则化，正则化一般有如下形式：

$$\min_{\hat{f} \in \mathcal{F}} \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, \hat{f}(x_i)\right) + \lambda J\left(\hat{f}\right) \tag{2-9}$$

其中，第一项是经验风险，第二项是正则项，$\lambda \geqslant 0$ 为调整两者之间关系的系数。

正则化可以取不同的形式，最常见两种是 L_1 范数和 L_2 范数。L_1 范数表示为：

$$L(w) = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{f}(x_i; w) - y_i\right)^2 + \lambda |w|_1$$

这里，$|w|_1$ 表示参数向量的 L_1 范数。L_2 范数表示为：

$$L(w) = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{f}(x_i; w) - y_i\right)^2 + \frac{\lambda}{2}|w|^2$$

这里，$|w|_2$ 表示参数向量的 L_2 范数。

正则化符合奥卡姆剃刀原理：在所有可能选择的模型中，能够很好的解释已知数据并且十分简单的才是最好的模型。

2.5.2 梯度下降法

梯度下降法是求解无约束优化问题的一种方法，若函数可微，从函数的任意一点，通过梯度下降法可以找到函数的局部极小值。

梯度下降方法基于以下的观察：如果实值函数 F(**x**) 在点 **a** 处可微且有定义，那么函数 F(**x**) 在 **a** 点沿着梯度相反的方向 $-\nabla F(\boldsymbol{a})$ 下降最快。因而，如果

$$\boldsymbol{b} = \boldsymbol{a} - \gamma \nabla F(\boldsymbol{a})$$

对于 γ >0 为一个够小数值时成立，那么 $F(\boldsymbol{a}) \geq F(\boldsymbol{b})$。

考虑到这一点，我们可以从函数 F 的局部极小值的初始估计 $\boldsymbol{x}_0$ 出发，并考虑如下序列 $\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$ 使得

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \gamma_n \nabla F(\boldsymbol{x}_n),\ n \geq 0$$

因此可得到

$$F(\boldsymbol{x}_0) \geq F(\boldsymbol{x})_1) \geq F(\boldsymbol{x})_2) \geq \cdots$$

如果顺利的话序列 $\mathbf{x}_n$ 收敛到期望的局部极小值。注意每次迭代步长 γ 可以改变。

图2-2示例了这一过程，这里假设 F 定义在平面上，并且函数图像是一个碗形。蓝色的曲线是等高线。红色的箭头指向该点梯度的反方向。沿着梯度下降方向，将最终到达碗底，即函数 F 局部极小值的点。

2.6 马尔科夫链理论

2.6.1 模型简介

马尔可夫链，又称离散时间马可夫链[41]，是状态空间中从一个状态到另一个状态的转换的随机过程，该过程要求具备“无记忆”的性质：下一状态的概率分布只能由当前状态决定，在时间序列中它前面的事件均与之无关。这种特定类型的“无记忆性”称作马可夫性质。马尔科夫链作为实际过程的统计模型具有许多应用。

马尔可夫链中的每一步，可以依据概率分别从一个状态转移到另一状态，与状态之间改变相关的概率叫做转移概率。已知满足马尔可夫性质的随机变量序列 X1，

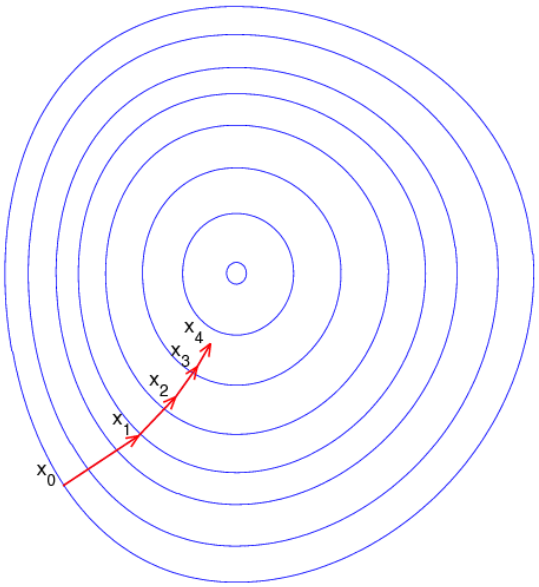


图 2-2 梯度下降示意图

X2，X3，...，依据马尔科夫性质的“无记忆性”有：

$$p\left(X_{n+1}=x|X_1=x_1,X_2=x_2,\ldots,X_n=x_n\right)=p\left(X_{n+1}=x|X_n=x_n\right) \tag{2-10}$$

当转移概率与 n 无关时，则称此马尔科夫链是时齐马尔科夫链，本文后续提出的模型是建立在时齐马尔科夫链模型基础上构建的。时齐马尔科夫链满足：

$$p\left(X_{n+1}=x|X_n=y\right)=p\left(X_n=x|X_{n-1}=y\right) \tag{2-11}$$

2.6.1.1 随机游走

随机游走是一种数学统计模型，它是一连串的轨迹所组成，其中每一次都是随机的。一种较常见的随机游走模型是在规则点阵上进行，称为点阵随机游走。在每一步中，标记的位置根据特定的概率分布从一点跳到另一个点。在简单点阵随机游走中，每一点只能跳到点阵中的相邻位置，形成点阵路径。

设定简单点阵随机游走是时齐的，其转移概率除了满足2-11，还满足：

$$p_{ij}=p\left(X_{n+1}=i|X_n=j\right)=0\left(i\notin Adj\left(j\right)\right)\sum_{i\in Adj(j)}p_{ij}=1 \tag{2-12}$$

其中，Adj(j) 表示点阵中与点 j 相邻的点的集合。

2.7 本章小结

本章对本文涉及到的理论技术进行了简要的概述。首先介绍了两种聚类算法：k-means 算法和 k-medoids 算法，并通过伪代码展示了算法的流程，并分析了聚类算法的两种评估指标，即内部指标和外部指标；然后介绍了多项式拟合技术，引出了幂级数中关于收敛域的讨论；随后简单介绍了最优化理论，并针对无约束优化问题引出了经验误差加正则化的通用方法和常用求解算法；最后介绍了马尔科夫链模型，阐述了马尔科夫链建模的特性。本章介绍的所有理论技术对后文研究起到了举足轻重的作用。

第三章 基于复合采样的的分布式轨迹聚类算法

3.1 引言

目前针对分布式聚类算法的研究已经取得了一些成果，一部分研究方法是以数据聚合为前提的，比如文献 [30] 和文献 [32] 提出的方法，这类方法首先需要将分布式中的数据集合在一起，然后以特定的方式将数据集划分给各个计算节点以提高聚类准确度和计算高效性，这类方法在聚类准确度上固然和数据集中式聚类相当，但是由于需要原始数据在网络中传输，这使得很多需要考虑数据隐私性的场景下变得不适用；鉴于数据隐私层面的考虑，一部分研究基于安全多方计算提出了自定义用于分布式计算加密协议，这类方法虽然在数据隐私层面和聚类准确度上表现良好，但却消耗了大量的带宽资源，特别是对于在数据量爆炸式增长的今天。

另一部分研究主流思路是基于局部聚类和全局聚类相结合的方式[7] [9] [11]，其主要思想为：在分布式框架中有两种角色，若干个计算节点和一个中心节点，计算节点基于本地的数据先进行局部聚类，然后依据局部聚类结果和一些额外的统计信息组成特定的数据结构，各个计算节点将由局部聚类结果和统计信息组成的数据结构通过网络传输给中心节点，中心节点利用局部聚类结果进行全局聚类，然后将全局聚类结果传输给各个计算节点。这类方法由于其在计算准确度、带宽和隐私性层面三方面的平衡，受到了很多学者的青睐，但这类算法的计算准确度不太稳定，造成这种不稳定的原因是这类方法在网络中传输的数据结构与数据的真实分布不是一一映射的关系，一个计算节点利用局部聚类结果和统计信息组成的数据结构可能对应多种数据分布，而这种映射到数据分布的多样性将给后续的全局聚类造成影响，数据结构到数据分布的多样性如图3-1

图中红色叉表示局部聚类得到的簇心，A 对应的是真实数据分布，而 A1、A2 和 A3 对应的数据分布于 A 对应的数据分布有着同样的数据结构（簇心和统计信息），即一个数据结构（簇心和统计信息）对应着多种数据分布，而不同的数据分布可能使得全局聚类的结果迥然不同。针对这种情况，本文提出一种基于复合采样的分布式轨迹聚类算法（CSD-Clustering），该算法为每一条轨迹建模，在网络中传输的是每一条轨迹的模型参数，使得聚类准确度高且聚类结果稳定性强，且通过复合采样操作有效控制了网络流量的消耗。

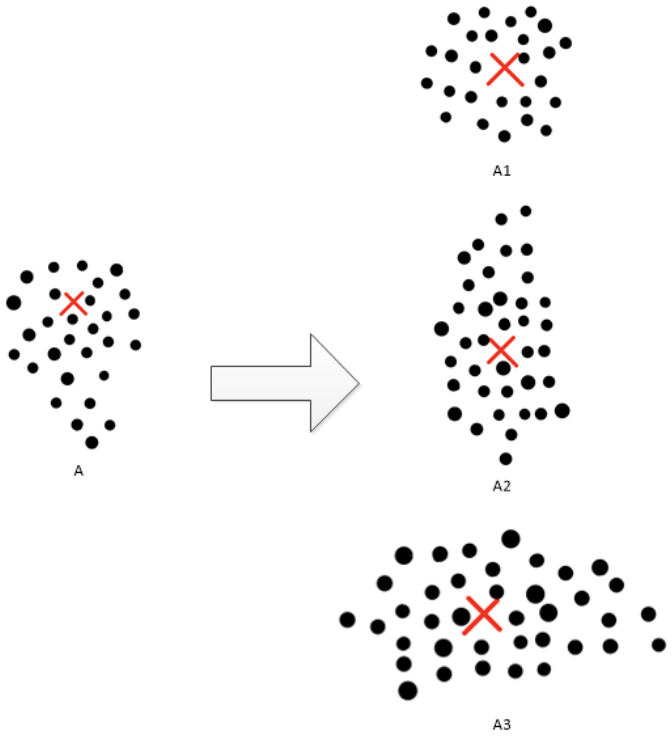


图 3-1 数据结构到数据分布映射的多样性

3.2 基于复合采样的的分布式轨迹聚类算法

3.2.1 问题分析

首先简述基本的网络模型和距离定义，然后是**隐私相关描述**并给出设计目标。

假设多属地分布式系统中，有 k 个跨地理分布的属地中心节点，包含一个中心服务器，中心服务器可以由任意一个提交任务的属地服务器担当。每个属地中心节点拥有 m 条时空轨迹数据。对于每条时空轨迹，其中有 n 个数据点，表示数据对象经过了 n 个空间位置。

对于在指定时间内，以恒定时间间隔组成的、包含 n 个数据点的轨迹，我们通过时空轨迹矩阵 T 来描述：

$$\mathrm{T}=\begin{bmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \\ t_1 & \cdots & t_n \end{bmatrix}$$

其中，轨迹 T 中第 i 个点的坐标为 $(x_i, y_i, t_i)^T$。因为轨迹数据是在指定时间内

以恒定时间间隔组成的，轨迹矩阵 T 可以通过坐标矩阵 P 来表示：

$$P = \begin{bmatrix} x_1 & \dots & x_n \\ y_1 & \dots & y_n \end{bmatrix}$$

若现有轨迹 a 和轨迹 b，其对应的坐标矩阵分别为 P_a 和 P_b，则这两条轨迹空间距离 Dist(a,b) 定义为：

$$Dist(\mathrm{a},\mathrm{b}) = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(a_x^i - b_x^i)^2 + \left(a_y^i - b_y^i\right)^2} \tag{3-1}$$

其中 a_x^i 表示轨迹 a 第 i 个点在 x 维度上的数值，a_y^i 表示轨迹 a 第 i 个点在 y 维度上的数值，b_x^i 、b_y^i 则分别代表轨迹 b 第 i 个点在 x、y 维度上的数值。

我们的基本目标是对分布的轨迹数据进行联合处理，快速获取精度的聚类结果，同时减少数据处理的带宽消耗，并保持每条轨迹数据的隐私安全性。为了提高聚类算法的精度，我们采用了 RI 指数，其计算方式如式2-2所示，RI 用来评价聚类质量，针对网络带宽的压缩率，我们通过自定义的 CR 指标来衡量。

假设每个数据点有 2 个特征维度，每个属地中心节点拥有的时空数据大小可以定义为 IF=2*m*n。同时，在多属地分布式系统中，将必要的信息传输到中心服务器所产生的相应网络流量标记为 TF。最后，将网络通信量与总信息量的比值标记为压缩比 (compression ratio, CR)，即:

$$\mathrm{CR} = \frac{IF}{TF} \tag{3-2}$$

对于算法隐私保护层面的评价指标，我们从两个方面进行刻画：不确定性和覆盖度。

（1）不确定性：假设攻击者能够从网络传输数据中还原算法模型，从算法模型生成的数据与原始数据的相似程度。

（2）覆盖度：从网络中进行传输的数据中获取的信息量占原始数据信息量的百分比。注意这个度量指标与网络中传输数据量的大小和上述提到的不确定性无关，该指标只与训练模型使用到的数据与原始数据的占比有关。

3.2.2 总体流程

本文提出了基于复合采样的分布式聚类算法（CSD-Clustering），算法的总体框架如图**??**所示：

该算法主要包括四个步骤：首先，属地轨迹建模，在各个属地节点对属地的每条轨迹数据进行预处理和抽样（包含轨迹数量抽样和每条轨迹坐标点抽样），并

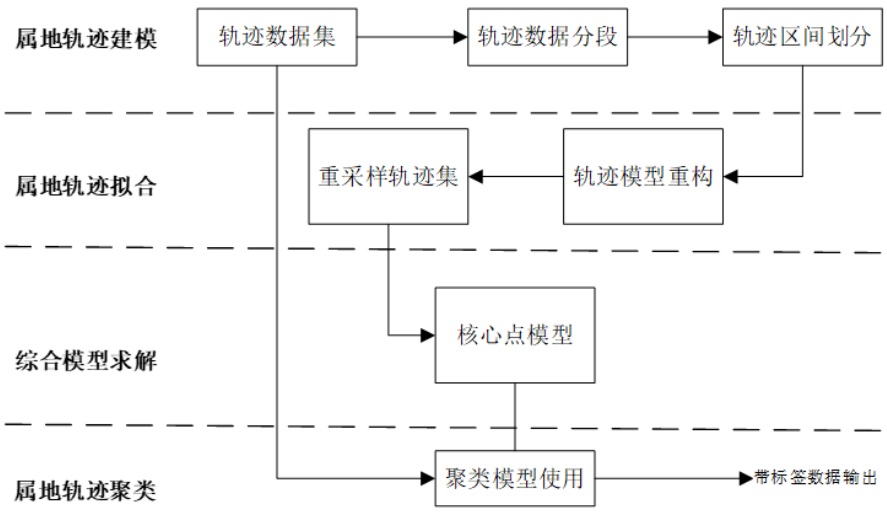


图 3-2 CSD-Clustering 算法框架

采用多项式函数拟合各采样轨迹曲线；第二步，局部模型传输，每个属地中心节点将拟合的轨迹函数参数信息发送到中心服务器上。第三步，聚类与反馈，中心服务器根据从接收参数中恢复的轨迹计算聚类结果。最后，属地轨迹聚类，中央服务器将聚类结果返回给每个属地中心节点，最终结果在本地产生。

3.2.3 属轨迹地数据预处理

属地数据预处理可以分为四个步骤，轨迹数量抽样，轨迹点数抽样，分段和分区。如图3-3所示。

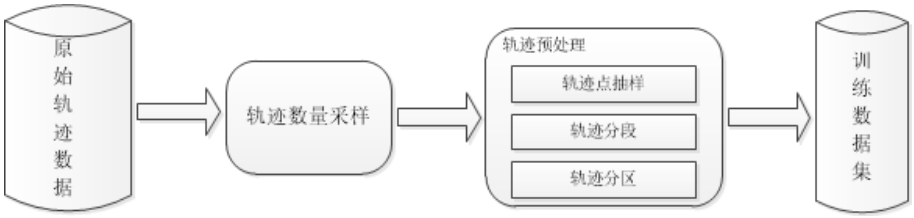


图 3-3 属地轨迹数据预处理

考虑到更大程度上减少网络中的传输流量和各属地轨迹数据集中存在的数据冗余，属地在对轨迹进行建模前进行轨迹数量上的抽样，但应注意，为了保证全局层面聚类的准确度，分布式平台中的所有属地中心在轨迹数量采样上的采样率应保持一致，且采样的轨迹应具有一定的代表性。可以采用一种 VQ 算法，最典型的 VQ 算法即是聚类算法。本文采用了 k-means 聚类算法，其算法流程如算法1所示，将属地轨迹数据集输入到 k-means 算法，得到输出结果，即簇划分 $C = \{C_1, C_2, ..., C_k\}$，在每个簇中，采用指定的抽样频率对簇中的轨迹进行随机抽

样，对抽样得到的所有轨迹集合中的每条轨迹将执行后续预处理操作。

假设每条轨迹由 n 个坐标点组成，由于传感器的采样频率一般很高，所以轨迹中相邻点之间的距离相隔很近，针对轨迹拟合问题，适当降低采样频率对于轨迹的大致走势几乎没有影响，为了降低轨迹拟合的计算复杂度，在模型拟合前执行轨迹点数采样，其采样频率对于后续轨迹模型的拟合效果将会有很大的影响。

在轨迹建模阶段前，CSD-Clustering 需要将每个轨迹分割成连续的段。其根本原因是，通过函数拟合的曲线必须满足自变量到函数值的唯一映射，当一个自变量对应多个函数值时，函数是无法拟合的。因此，我们采用了轨迹分段的方法，目的是使得轨迹分段曲线任意一个变量中值只对应一个因变量值。

具体地对于每一个时空轨迹矩阵，均有与其对应的速度轨迹 V，其刻画了一段时间段内，物体运动的速度变化轮廓。对于上述时空轨迹矩阵 P，V 定义如下：

$$\mathrm{V} = (\mathrm{v}_1, \mathrm{v}_2, ..., \mathrm{v}_{n-1})$$
$$v_i = \frac{(x_{i+1} - x_i, y_{i+1} - y_i)}{\Delta t}$$

其中 v_i 是一个向量,V 是一个 2×(n-1) 矩阵。设有一条由 n 个轨迹点责成的轨迹 P，$T = t_1, t_2, ..., t_n$ 代表轨迹中坐标点对应的时间，对应速度轨迹 $V = (v_1, v_2, ..., v_{n-1})$，如果轨迹 P 划分成两段轨迹，分割时间点为 t_i，则下列式子需要满足：

$$\begin{cases} v_1 \cdot v_k > 0 \quad k = 1, 2, \ldots, i-1 \\ v_1 \cdot v_i \leq 0 \\ v_i \cdot v_{n-1} > 0 \end{cases}$$

这种约束我们称之为速度角约束。整个轨迹将被分割成多个子轨迹段。

假设子轨迹段通过水平平移到其定义域关于 y 轴对称时（不考虑两段的开闭区间）所对应的真实函数模型为 f(x)，我们试图拟合这个子轨迹段的函数记为 $\hat{f}(x)$，毫无疑问，我们希望两者在其定义域内应该尽可能的相似。f(x) 在 x=0 处展开的麦克劳伦级数如式2-8所示，依据其表达式形式可以推出，麦克劳林级数的前 n 项，即：

$$f(x) = \sum_{i=0}^{n} \frac{f^{(i)}(0)}{i!} (x)^i \tag{3-3}$$

在 x=0 处的第 i 阶导数与 f(x) 相等，所以式3-3在 x=0 邻域内与 f(x) 相似度很高，这个邻域范围可以利用幂级数的收敛半径求解方法，若 f(x) 定义域在其收敛半径内，则一定存在一个幂级数能够很好的拟合 f(x)，反之，如果 f(x) 的定义域超过了其收敛半径所囊括的范围，则式 $2-8-f(x)$ 一定无法收敛，故式 $3-3-f(x)$

也一定无法收敛，所以任意幂级数前 n 项拟合 f(x) 时一定存在其第 i 阶导数与 f(x) 的第 i 阶导数不相等，因此这个子轨迹段很难通过幂级数前 n 项式进行拟合。针对这一现象，我们对子轨迹段进行启发式的分区操作，使得通过存在一个幂级数的前 n 项式能够较好拟合特定区间的轨迹，即限制子轨迹段的定义域长度。

现假设有子轨迹段 $\{(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)\}$，如果将分段轨迹分成三个轨迹区间：$\{(x_1,y_1),(x_2,y_2),\ldots,(x_i,y_i)\}$，$\{(x_{i+1},y_{i+1}),(x_{i+2},y_{i+2}),\ldots,(x_j,y_j)\}$ 和 $\{(x_{j+1},y_{j+1}),(x_{j+2},y_{j+2}),\ldots,(x_N,y_N)\}$，则分割坐标 i 和 j 应满足：

$$|x_{i+1}-x_1|>S|x_{j+1}-x_{i+1}|>S$$

其中 S 是人工设置的阈值。

3.2.4 属地轨迹拟合

对第 i 区间含有 k 个数据点，$\{(x_1,y_1),(x_2,y_2),\ldots,(x_k,y_k)\}$，定义损失函数：

$$\mathrm{L}\left(y_i,\hat{\mathrm{f}}(x_i)\right)=\left(\mathrm{y_i}-\hat{\mathrm{f}}(x_i)\right)^2$$

其中 $\hat{f}(x)$ 表示轨迹拟合函数模型。

将代价函数定义成经验风险函数：

$$\mathrm{cost(f)}=\frac{1}{k}\sum_{j=1}^{k}\mathrm{L}\left(y_j,\hat{\mathrm{f}}(x_j)\right) \tag{3-4}$$

若 $\hat{(f)}(x)$ 为 n-1 次多项式，系数分别为 $a_0,a_1,,a_{n-1}$。

若在此区间使用 k-1 阶多项式作为轨迹拟合函数模型，即：

$$\hat{f}(\mathrm{x})=a_0+a_1x+\cdots+a_{k-1}x^{k-1}$$

以此拟合该区间的曲线，将区间的 k 个坐标带入 $\hat{(f)}(x)$ 可以得到 k 个等式，将这 k 个等式写成矩阵形式即：

$$\begin{bmatrix}1 & x_1 & \ldots & x_1^{k-1}\\ 1 & x_2 & \ldots & x_2^{k-1}\\ \vdots & \vdots & \ddots & \vdots\\ 1 & x_k & \ldots & x_k^{k-1}\end{bmatrix}\begin{bmatrix}a_0\\ a_1\\ \vdots\\ a_{k-1}\end{bmatrix}=\begin{bmatrix}y_1\\ y_2\\ \vdots\\ y_k\end{bmatrix}$$

记上式中的系数矩阵记为 A，易得，$|A^T|$ 为范德蒙德矩阵，由范德蒙德性质可

得：

$$\left|A^T\right| = \prod_{p>q}(x_p - x_q)$$

由于 $x_0, x_1, \ldots, x_k$ 彼此不相等，所以行列式 $|A^T| \neq 0$，此时存在唯一一个 k-1 次多项式能够包含所有数据点，此时用 k-1 阶多项式 $\hat{f}(x)$ 拟合曲线的经验风险为 0，拟合的函数曲线穿过每一个坐标点，而且此时求解模型的所有计算量只需计算出系数矩阵 A 即可。但由于数据不可避免受到噪音污染，我们并不要求拟合的函数曲线严格穿过每一个坐标点，通过求解线性方程组解得的模型往往复杂度很高，可以通过简单实验验证，我们假设在 $y = x^2$ 附近取 9 个点，通过以上求解线性方程组的方法可以得到一个 8 阶的多项式，拟合的 8 阶多项式可以通过所有坐标点，但函数形式却过于复杂，如图3-4：

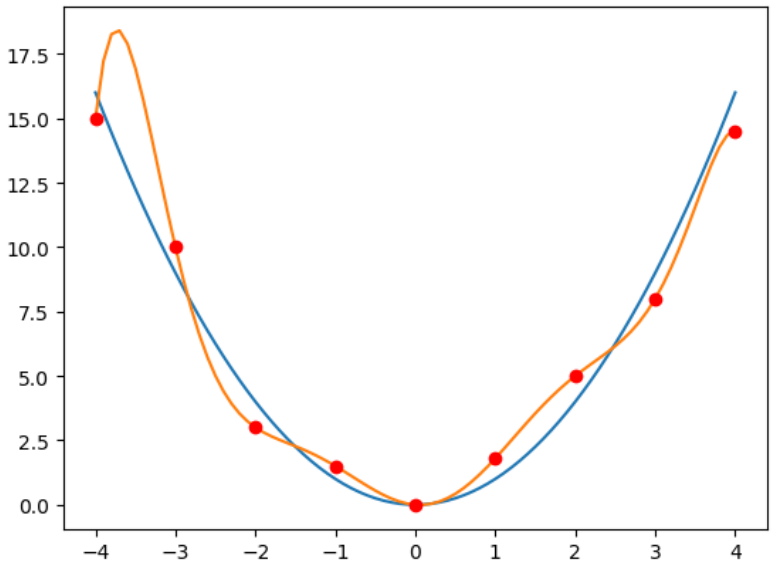


图 3-4 无正则化拟合

而如果我们使用 0 阶多项式，即常数函数去拟合这条曲线是，此时经验风险非常大，而拟合函数复杂度则很小。为了在满足模型拟合精度的同时避免模型复杂度过高，我们借助最优化理论思想采用经验风险加正则化项的方式，于是代价函数定义为：

$$\mathrm{cost}(\hat{f}) = \frac{1}{k}\sum_{j=1}^{k} \mathrm{L}\left(y_j, \hat{f}(x_j)\right) + \mathrm{C} * J(\hat{f}) \tag{3-5}$$

式3-5前一项是经验风险，后一项 $J(\hat{f})$ 则是正则化项，也称作结构风险, 用来表

示轨迹拟合函数 $\hat{f}(x)$ 的复杂度，C 为常量，用来调整两者之间关系，若 C=0，则代价函数退化成经验损失函数，若 C 取值很大，则函数模型倾向于选择更简单的模型而不顾拟合准确度。

我们使用参数向量的第一范数来定义 $f(\hat{f})$，即：

$$J(f)=\sum_{i}^{k}|a_i|_2^2$$

综上，优化目标定义为：

$$\min\frac{1}{N(i)}\sum_{j=1}^{N(i)}\mathrm{L}\left(y_{i,i}(x_j)\right)+\mathrm{C}*J\left(\hat{f}_i\right) \tag{3-6}$$

其中 N(i) 表示第 i 区间数据点的个数，$\hat{f}_i(x)$ 则表示第 i 区间多项式拟合函数，其多项式次数为 N(i-1)。此类问题属于无约束优化问题，可以采用最速下降来求解局部极小值。

以上方法可以通过另一个角度解释，我们可以将上式看成一般岭回归的形式：

$$\min_{w}\|Xw-y\|_2^2+\alpha\|w\|_2^2$$

我们只是指定了一个明确映射形式的高维映射 $\Phi(x)=\left(1,x,x^2,...,x^{k-1}\right)$，故我们求解问题等同于一个指定高维映射的岭回归问题，我们可以通过求解核岭回归的方式求解文中的优化问题，而且核岭回归的求解问题是存在闭式解的，因此可以加快拟合的速度。

将通过以上方法可以得到指定区间的函数参数。为了在中心节点能够还原出轨迹数据，除了传递函数参数信息，还需要传输必要的元信息，将元信息和参数信息打包形成的数据结构成为基于复合采样的轨迹数据单元（CSTD-Unit），其格式如下：

ID	StartX	EndX	Parament

数据单元格式字段含义如下：

- **ID**：此字段包含了属节点编号、轨迹编号、所处区间
- **StartX**：此区间 x 轴上最小值
- **EndX**：此区间 x 轴上最大值
- **Parament**：此区间轨迹拟合函数参数

各个计算节点把轨迹所有分区坐标信息转化成许多 CSTD-Unit 传递给中心节点，中心节点通过这种特殊的数据结构能够将轨迹数据进行还原。

3.2.5 综合模型求解和模型回传

在接收到信息后，中心服务器根据各个计算节点传输的 CSTD-Unit 数据单元还原轨迹拟合函数，然后在还原出来的轨迹函数上依照原始数据的采样频率进行等间隔抽样，得到还原出的数据点，以此组成还原数据，从 CSTD-Unit 数据单元中还原轨迹数据的过程称为轨迹生成过程，轨迹生成过程操作流程如算法3所示。

算法 3-1 轨迹生成过程

```
Data: CSTD-Unit 数据单元集合 U
Result: 生成轨迹集合 G
for unit in U do
    初始化序列 ys;
    id = unit['ID'];
    para = unit['Prament'];
    xmin = unit['StartX'];
    xmax = unit['EndX'];
    以一定采样率在区间 [xmin,xmax] 上进行均匀采样，得到序列 xs;
    for x in xs do
        lx = np.array([x**i for i in range(len(params))]);
        y = np.dot(x,params);
        将 y 纳入 ys;
    end
    利用序列 xs 和序列 ys 组成轨迹序列 L;
    将轨迹序列 L 纳入生成轨迹集合 G;
end
```

将恢复的轨迹数据集输入到聚类模型中，本文选择的是 K-means++ 算法，在第二章中描述了 k-means 算法的具体流程，k-means++ 只是在 k-means 基础上针对开始 k 个均值向量的初始化提供了具体的方法。k-means++ 方法，尽可能地使初始聚类中心分散开。这种方法首先随机产生一个初始聚类中心点，然后按照概率 $P = \omega \sum_i (x - c_i)^2$ 选取其他的聚类中心点。其中 ω 是归一化系数。

k-means++ 初始化操作如下：

算法 3-2 kmeanspp$_i$*nitialization*

Data: 样本集 $D=\{t_1,t_2,...,t_n\}\}$，簇个数 K

Result: 簇心向量 $\mu=\{\mu_1,\mu_2,...,\mu_K\}\}$

从数据集 D 中随机初始化 1 个样本，将其作为第一个簇的簇心 $\mu=\mu_1$;

repeat

  R=D-μ

  for t_i *in* R **do**

    计算 $P(x_i)$，$P(t_i)=\omega\sum_{\mu_i\in\mu}(t_i-\mu_i)^2$，其中 ω 是归一化系数;

  end

  按照 R 集合中所有点的概率分布选出 μ_k，将其加入集合 μ;

until *k=2,3,...,K*;

全局聚类的具体流程如下：

算法 3-3 全局聚类流程

Data: 样本集 $D=\{t_1,t_2,...,t_n\}\}$，簇个数 K，阈值 S

Result: 簇划分 $C=\{C_1,C_2,...,C_K\}\}$，簇心向量 $\mu=\{\mu_1,\mu_2,...,\mu_K\}\}$

K=2;

kmeanspp$_i$*nitialization*D,K;

C,μ=kmeansD,K;

DI_K=DI(C);

flag=true;

repeat

  K=K+1;

  kmeans$_i$*nitialization*D,K;

  C,μ=kmeansD,K;

  $\mathrm{DI}_K=DI(C)$

  if $DI_K-DI_{K-1}\leqslant S$ **then**

    flag=false;

  end

until *flag*;

对于 k 值的选择，在上面的讨论中，我们一直把 k 当做已经给定的参数。但是在实际操作中，聚类类别数通常都是未知的。如何确定超参数 k 我们采用如下

方法。

我们可以使用不同的 k 值进行试验，并作出不同试验收敛后目标函数随 k 的变化，将 k 值趋向稳定时的值作为我们的超参数。如下图3-5所示，分别使用 1 到 7 之间的几个数字作为 k 值，曲线在 k=3 处有一个较为明显的弯曲点。故确定 k 取值为 3。

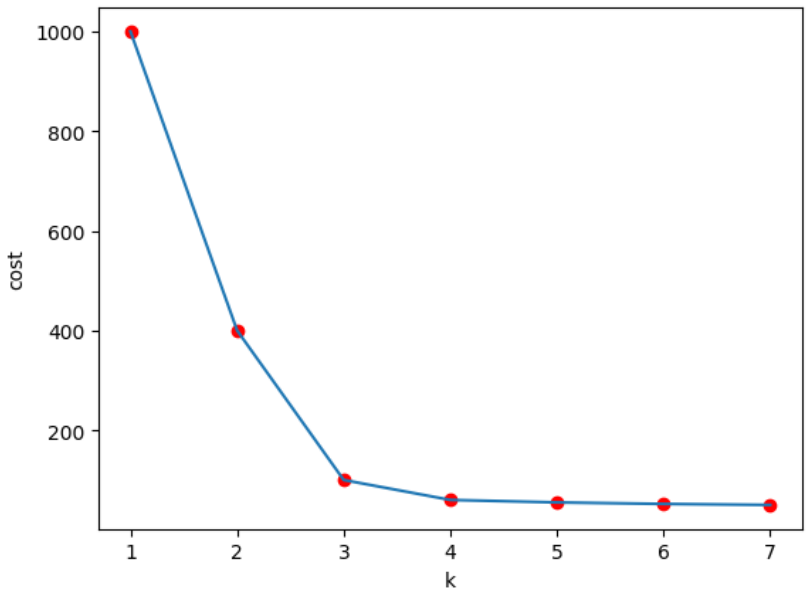


图 3-5 k 值选择

中心服务器计算出 k 个均值向量后，将 k 个均值向量回传给各个计算节点，当计算节点侦查到新轨迹时，判断其与回传的 k 个均值向量的距离来判定是否为异常轨迹：若新轨迹与所有均值向量距离都超过了设定的阈值，则可以判定为轨迹为异常轨迹。

3.3 实验与分析

3.3.1 理论分析

（1）时间复杂度

计算节点时间复杂度分为两个阶段: 轨迹预处理和优化函数求解。

轨迹预处理分为三个步骤，，轨迹数量抽样，轨迹点数抽样，分段和分区。假设轨迹数目为 m 条，每条轨迹含有点数为 n，假设轨迹数量抽样阶段使用 k-means 算法进行聚类，其复杂度为 O(m*n*k*t)，k 表示簇个数，t 是平均迭代次数。轨迹点数抽样复杂度为 O(m)，轨迹分段阶段，对于每一条轨迹计算复杂度为 O(2*(n-1))，

对于 m 条轨迹则为 O(m*2*(n-1))，化简即为 O(m*n)；区间划分阶段，假设区间中数据点数为 i，复杂度为 O(2*(i-1))，若对于所有轨迹的所有区间进行分析，则复杂度为 O(2*(n-1)*m)，化简为 O(m*n)；优化函数求解阶段，若以最速下降法为例，每次迭代复杂度为 O(n*m)，若平均迭代次数为 t，则在此阶段时间复杂度为 O(t*m*n)。综上，函数拟合时间复杂度为 O(k*m*n*t)。

中心节点的时间复杂度分为两个阶段：轨迹生成和全局聚类。假设轨迹数目为 m，每条轨迹包含 n 个点, 轨迹生成时间复杂度为 O(m*n)；全局聚类，其时间复杂度为 O(m*n*k*t)。

（2）带宽消耗

对于带宽分析，若对于每个节点，若轨迹数量抽样率为 γ，轨迹点数抽样率为 α；对于 n 个数据点进行轨迹拟合后的参数向量通过稀疏表示后的压缩率为 β（即需要 n*α 参数拟合），则对于一个含有 m 条轨迹、每条轨迹拥有 n 个数据点、每个数据点维度为 2 的节点来说，将其传递给中心节点的网络通信量为 $(m*n)/(\gamma*\alpha*\beta)$；若是传输数据，则网络通信量为 m*n*2，故网络通信量的压缩比为 $2*\gamma*\alpha*\beta$，在此压缩比下，聚类准确度分析将在第五章详细介绍。

（3）隐私性

算法隐私性层面主要从两个角度进行考量：不确定性和覆盖率。

不确定性通过生成轨迹数据与原始属地轨迹数据之间的相似度进行度量，首先，拟合轨迹的多项式模型是通过最小化代价函数得到，而代价函数是由经验损失和正则化构成，所以拟合出的的多项式模型是不经过所有训练数据的，故原始属地轨迹数据不可能由多项式轨迹准确还原；再者，由于生成轨迹数据是拟合多项式模型在指定区间均匀采样得到，故与原始属地轨迹数据存在一定的差异；但生成轨迹数据与原始轨迹数据有着相同的轨迹走势，这一点不容否认。算法在隐私性的覆盖率方面，由于属地节点在属地轨迹预处理阶段经过了轨迹数量抽样处理，故生成的轨迹数据是无法覆盖所有原始数据的，覆盖率的大小取决于轨迹数量抽样操作采样的抽样率，抽样率越低则覆盖率越低，反之，覆盖率则越高。

3.3.2 实验与分析

实验是在真实的数据集上进行的，这些数据集包含了在大阪的 ATC 购物中心的游客的轨迹。在我们的实验中，我们应用了时间、位置 x、位置 y、人物 id 等字段[①]。经过预处理后，我们选择了 4939 条轨迹作为原始数据集，每条轨迹包含 500 个点。

① https://irc.atr.jp/crest2010$_{HRI}/ATC_{d}ataset/$

实验系统由三个计算节点和一个中心节点组成。将该算法与两种聚类方法baseline 进行了比较。第一种算法对整个数据集执行标准的 k-means 算法 (baseline I)，而第二种算法直接将抽样的轨迹传输到中心服务器进行聚类 (baseline II)。baseline I 的结果视为最优结果，后续实验中使用的 RI 指数则是以 baselien I 的聚类结果作为标签的。第三种算法则是本章提出的 CSD-Clustering 算法。

评估实验结果采用两个指标：CR，表示算法的数据压缩能力；RI，表示聚类结果的精度。

在拟合轨迹曲线前，需要为轨迹区间划分阶段的参数 S 赋值及拟合轨迹曲线上的结构风险惩罚系数 C 赋值。

首先对惩罚系数 C 赋值不同，观察拟合曲线的变化。我们保持轨迹间隔相同，并将 C 的值设置为 0.001、1100、10000。结果如图3-6所示。

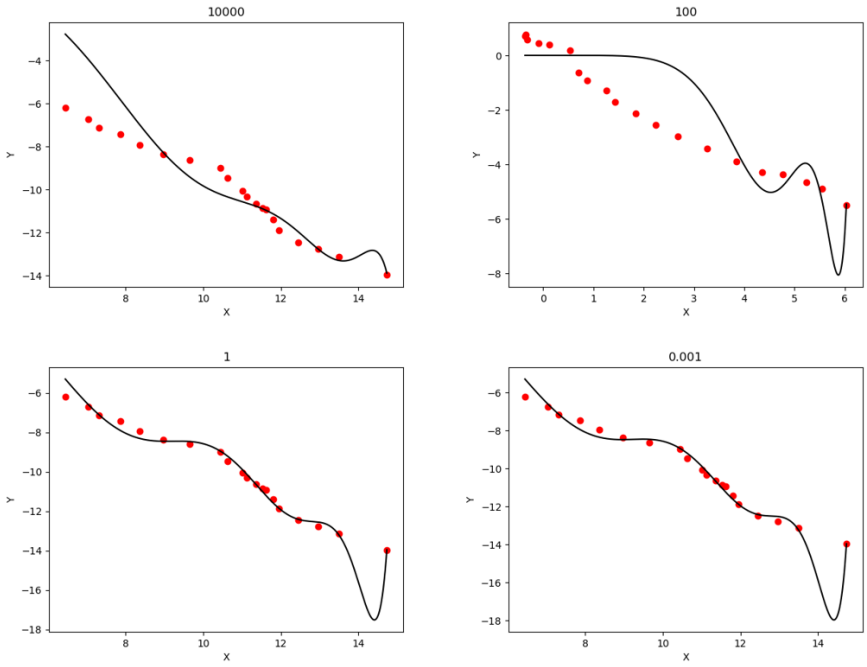


图 3-6 变量 C 不同取值的结果

从图3-6可以看出，随着 C 值的减小，拟合性能得到改善。然而，当 C 越来越小时，模型的结构也变得更加复杂。

然后观察不同参数 S 对轨迹曲线拟合的影响。我们选择相同的轨迹区间，令 S=12 和 5。结果如图3-7所示。

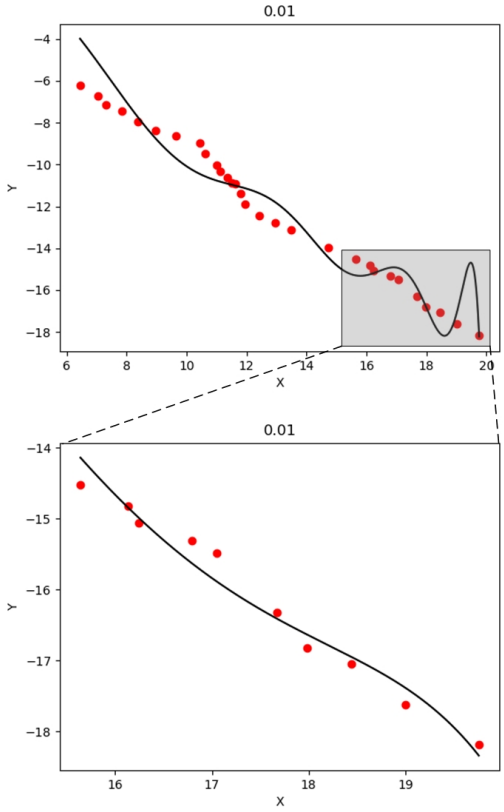
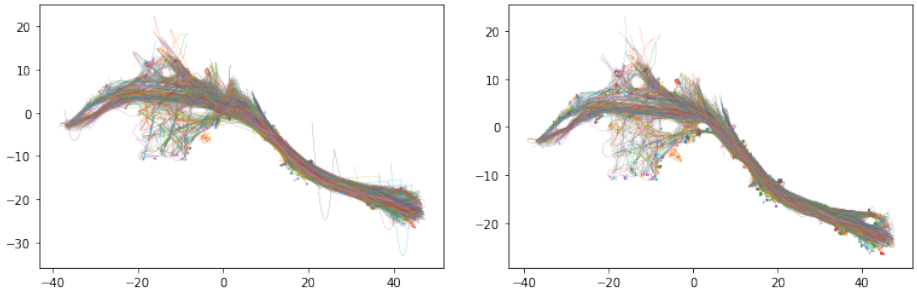


图 3-7 变量 S 不同取值的结果

从图3-7可以看出，上面的那幅图显示当 S=12 时，拟合曲线的 $X(6,20)$, 下面那幅图显示当 S=5 时，拟合曲线的 $X(15,20)$, 根据这个结果可以看出，当 S 越小，曲线拟合得越好。

各个属地节点完成模型训练完成后，将 CSTD-Unit 传递给中心节点，中心节点依据算法3生成轨迹数据，生成后的轨迹数据与原始轨迹数据如图3-8所示，其中图3-8(a)展示了生成轨迹数据，图4-6(b)展示了原始轨迹数据。



(a) (b)

图 3-8 轨迹生成效果对比图

实验中选取不同的 K 值其损失函数值变化情况如图3-9。

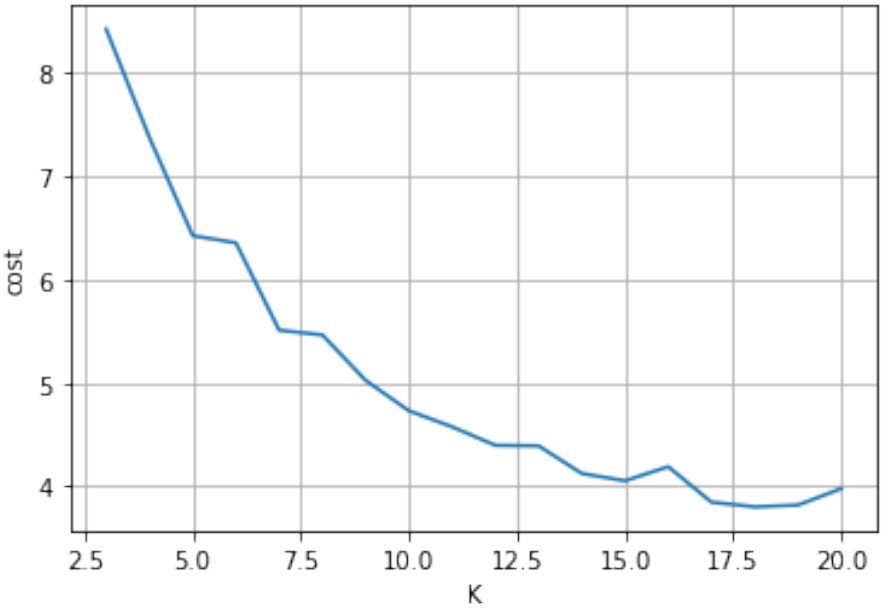


图 3-9 不同 K 值损失函数变化

关于聚类效果，我们设定参数 S 的取值为 5，C 取值为 1。在聚类过程中，我们将 K 的值设为 7。

当我们固定轨迹数量抽样率 γ=1，为轨迹点数抽样率 α 设置不同值。根据表3-1，随着 β 的增加，聚类算法 baseline II 的结果逐渐接近 baseline I。然而，更高的 RI 值会造成 CR 值变低，导致消耗更多的网络带宽。

表 3-1 不同轨迹点数采样率对比

”α	0.1		0.3		0.5	
	baseline II	CSD-Clustering	baseline II	CSD-Clustering	baseline II	CSD-Clustering
评价指标RI	0.79	0.81	0.80	0.93	0.81	0.91
CR	10	20	3.33	6.67	2	4

随着轨迹点数抽样率 β 的增加，CSD-Clustering 聚类的 RI 值起初上升然后稍微降低。虽然我们也采取了相应的措施，只是减轻了朗格现象的影响，而不是消除它。当轨迹数量抽样率 α 相同时，我们的算法在 RI 值和 CR 两个指标上表现比 baselineII 要好。

从表3-2可以看出，baseline II 和 CSD-Clustering 算法在有相同 CR 值时，我们的算法精度更高。此外，我们的算法避免了原始数据的网络传输，保护了数据的隐私。

表 3-2 相同带宽下的对比实验

评价指标	实验组	
	baseline II	CSD-Clustering
RI	0.795	0.926
CR	10	10

然后我们测试轨迹数量采样率对实验的影响。

此处，设定轨迹点数抽样率 α=0.2，为轨迹数量抽样率 γ 设置不同值来验证不同采样大小的影响。从图3-10可以看出，随着 γ 的增加，聚类的精度也不断增加。同时，随着 γ 的增加，模型会逐渐收敛，带宽消耗的效率较低。

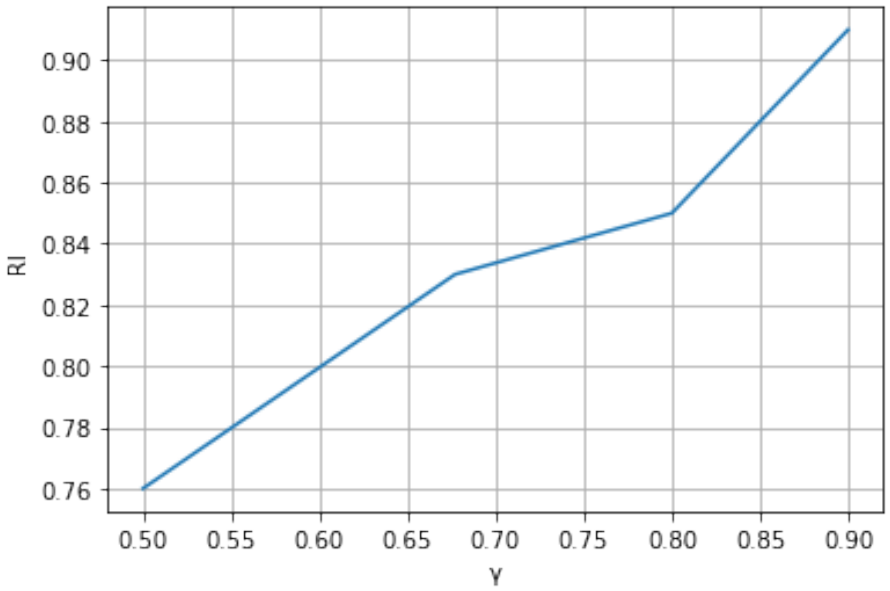


图 3-10 不同轨迹数量采样率对比

3.4 本章小结

本文提出了一种新型的分布式聚类方案 CSD-Clustering，该方案具有节约带宽和保护隐私两方面的优点。网络带宽的降低主要来自三个方面: 轨迹集合上的采样，每个轨迹内的部分采样，轨迹模型的拟合函数。此外，该方案还有效地避免了原始数据和直接相关结构信息的传输，加强了隐私保护。评价结果验证了该方法的有效性。

第四章 基于马尔科夫链的分布式轨迹聚类算法

4.1 引言

针对由局部聚类结果和统计信息组成的数据结构到数据分布的多样性问题，有部分研究采用概率密度分布模型来描述本地数据分布[15] [11]，这类方法可以有效估计低维数据的概率密度分布，但是针对高维数据的概率密度分布则由于训练数据规模大小的限制，在估计数据的概率密度分布上表示很差，这是因为估计数据的概率密度分布需要训练数据尽量多地填充数据所处的空间，而随着数据维度的增大，所需训练数据的数据量需求将会呈指数级增长，现实状况下，训练数据的数据量是很有限的，所以训练高维数据的概率密度分布几乎是不可能完成的任务。而轨迹数据是一种高维数据，故通过概率密度分布模型来估计本地轨迹数据分布是行不通的，直接估计数据的概率密度分布有着默认的先验假设：维度之间是相互独立的。轨迹数据维度之间显然是存在相互关联的，故通过概率密度取估计轨迹数据的分布式不合适的。对于第三章提出的 CSD-Clustering 算法，计算节点需要拟合所有轨迹的多项式模型，在本地数据规模大的情形下，计算节点的计算能力将成为算法的瓶颈，而且通过拟合的多项式模型还原出的轨迹数据，虽然与原始轨迹数据不相同，但却与原始数据有着一定程度的相似性，这对中心节点的权威性也提出了一定的挑战。

针对以上问题，本章提出一种基于马尔科夫链的分布式轨迹聚类算法：MCD-Clustering 算法，该算法通过马尔科夫链模型来估计轨迹数据分布，通过转移矩阵来还原轨迹数据，从而有效解决了对轨迹数据分布估计问题，缓解了 CSD-Clustering 算法中存在的计算性能瓶颈，而且也增强了数据隐私性的保护。

4.2 基于马尔科夫链的分布式聚类算法

4.2.1 总体方案

为了避免分布不确定性问题和高维数据生成模型轨迹问题，我们提出基于马尔科夫链的分布式聚类算法（MCD-Clustering）。为了避免分布不确定性问题，很自然想到直接传输属地数据概率分布模型参数到中心节点，这种方法在数据量大和数据维度低时使用，而一条具有 500 个二维坐标点的轨迹数据可以将其看做维度为 1000 的数据，此时估计数据概率分布模型的方法将不再使用，我们必须通过另外一种模型来估计轨迹数据的概率分布模型。因为每条轨迹各个点之间显然不是独立于彼此的，故当我们将轨迹数据看做是高维数据时，由于每条轨迹数据相

邻点存在相关性，可以看做维度和维度之间存在相关性，我们基于这种认知提出使用马尔科夫链来模拟一簇相似轨迹的生成模型，于是提出了基于马尔科夫链的分布式聚类算法（MCD-Clustering）。方案大致流程如图4-1所示解：

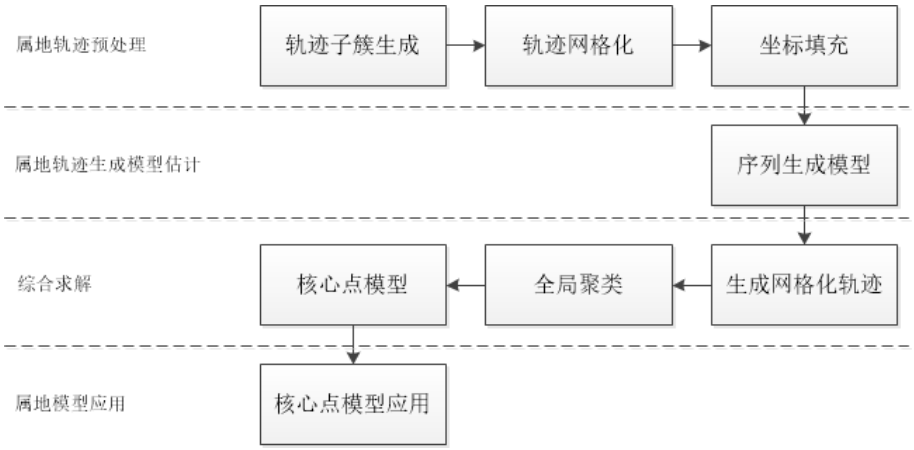


图 4-1 MCD-Clustering 算法框架

整个方案可以分为四个部分：属地轨迹预处理、属地轨迹生成模型估计、综合求解和属地模型应用。属地轨迹预处理，该阶段分为三个步骤：轨迹子簇生成、轨迹数据网格化和轨迹坐标填充。为了通过马尔科夫链能够更好的估计轨迹生成模型，首先对属地轨迹数据集进行聚类操作来得到多个轨迹子簇，我们假设一个子簇里面的轨迹大部分有着相同的轨迹形状。子簇生成后，针对每个子簇中的轨迹进行网格化操作，网格化的作用是通过更少数量的共用坐标点来表示子簇中所有轨迹中的坐标点。由于传感器在采样运动物体轨迹时存在不同物体速度的差异，所以经常会出现一条轨迹时间上相邻的坐标点在网格坐标中并不相邻，此时需要在这两个网格中不相邻的点以指定的策略补充一些坐标点，使得所有轨迹中时间上相邻的点在网格中也相邻。属地轨迹生成模型估计，通过马尔科夫链模型来模拟轨迹生成模型，利用属地轨迹数据集写出含有马尔科夫链参数的最大似然函数，利用最优化理论方法求得使最大似然函数取最大值的马尔科夫链参数。综合求解和属地模型应用，属地将求得的马尔科夫链参数传递给中心节点，中心节点通过这些参数重新生成轨迹坐标，利用重新生成的轨迹坐标作为全局聚类的输入，经过全局聚类输出若干簇心向量，将所有的簇心向量传递给各个计算节点，计算节点利用通过全局聚类计算出的簇心向量来对未知轨迹进行异常检测。

在进入方案具体流程之前，首先对常用符号进行定义。

假设计算节点的轨迹数据集记为 D，轨迹数据集包含有 n 条轨迹，每条轨迹

由 m 个坐标点构成，坐标点的维度为 2，即：

$$D=\{t_1,t_2,...,t_n\}\ t=\{(x_1,y_1),(x_2,y_2),...,(x_m,y_m)\}$$

轨迹数据通过聚类算法产生的簇集合记为 $C=\{C_1,C_2,...,C_k\}$，其对应的簇心向量记为 $\mu=\{\mu_1,\mu_2,...,\mu_k\}$。

4.2.2 属地轨迹预处理

属地轨迹预处理分为三个阶段：轨迹子簇生成、轨迹数据网格化和轨迹坐标填充。

轨迹子簇生成阶段主要通过聚类算法来实现，本节选择 k-means++ 算法进行聚类，通过聚类算法生成的轨迹子簇记为集合 C，其中聚类中使用的距离度量与第三章定义的距离一样，假设两条轨迹 a 和 b，则轨迹 a 和 b 之间的距离为：

$$\mathrm{Dist}(\mathrm{a},\mathrm{b})=\frac{1}{m}\sum_{i=1}^{m}\sqrt{(a_x^i-b_x^i)^2+\left(a_y^i-b_y^i\right)^2}$$

k-means++ 聚类在选择 k 值时需要聚类评价指标来评判不同 k 值下的聚类效果，这里采用内部指标 DI 来衡量，DI 表达式如式2-5所示。轨迹子簇生成的具体流程如5所示，这里不再赘述。

轨迹子簇生成后，针对每个子簇进行网格化。原始轨迹数据中的坐标点在二维连续空间上取值，将轨迹数据网格化后，轨迹所有坐标点在二维离散网格空间取值。最简单的网格化则是将二维连续空间中的点坐标取整，即将二维连续空间映射到由所有整数构成的坐标形成的二维离散网格空间，如图4-3所示。

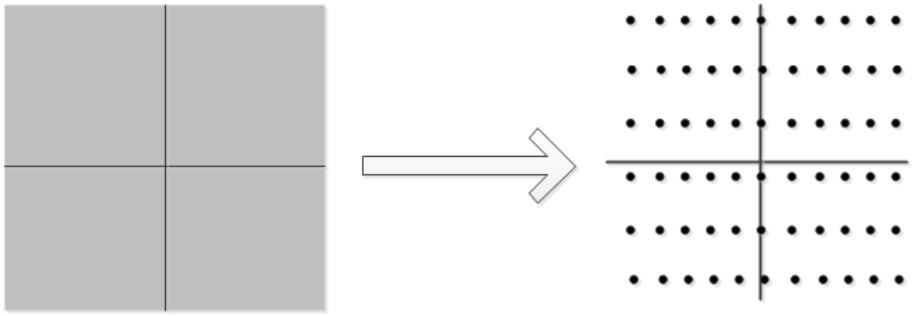


图 4-2 空间网格化

对于轨迹数据可以采用同样的方法对其进行网格化，网格化规则可以通过下面式子表示：

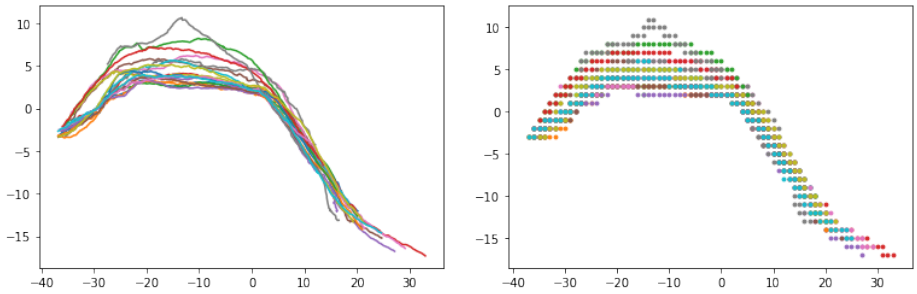
$$f(x)=\begin{cases} sign(x)*(\lfloor x\rfloor+1) & if\ |x-\lfloor x\rfloor|>0.5 \\ sign(x)*(\lfloor x\rfloor-1) & if\ |x-\lfloor x\rfloor|\leqslant 0.5| \end{cases}$$

$$\Phi((x,y))=(f(x),f(y))$$

$$g\{(x_1,y_1),(x_2,y_2),...,(x_l,y_l)\}=(x_1,y_1)$$

$$if\ (x_1,y_1)=(x_2,y_2)=...=(x_l,y_l)$$

其中，sign(x) 表示实数 x 的符号。通过上式规则对轨迹中的每一个坐标点进行网格化，图4-3展现了轨迹经过网格化后的效果，图4-3(a)表示原始轨迹数据，图4-3(b)表示网格化后的轨迹数据。网格化对轨迹数据的精度造成了一定程度的影响，但是完全不影响其对轨迹形状的刻画。



(a) (b)

图 4-3 轨迹网格化前后对比

由于传感器在采样运动物体轨迹时存在不同物体速度的差异，所以经常会出现一条轨迹时间上相邻的坐标点在网格空间中并不相邻，此时需要在这两个网格中不相邻的点之间通过指定的操作填充一些坐标点，使得所有轨迹中时间上相邻的点在网格空间中也相邻，这种保证了网格空间数据相邻性的操作称之为轨迹数据填充操作。假设一条网格化后的轨迹数据记为 $g=\{(x_1,y_1),(x_2,y_2),...,(x_l,y_l)\}$，轨迹数据填充操作流程如下：

算法 4-1 轨迹数据填充操作

Data: 一条网格化的轨迹 $g=\{(x_1,y_1),(x_2,y_2),...,(x_l,y_l)\}$

Result: 轨迹数据填充后的轨迹数据 $\hat{g}=\{(x_1,y_1),(x_2,y_2),...,(x_s,y_s)\}$

repeat

  if $x_i == x_{i+1}$ $|y_i - y_{i+1}| > 1$ **then**

    s = sign($y_{i+1} - y_i$) 在坐标 (x_i,y_i) 后面填充坐标 $(x_i,y_i+s),(x_i,y_i+2*s),...,(x_i,y_{i+1}-s)$

  end

  else if $y_i == y_{i+1}$ $|x_i - x_{i+1}| > 1$ **then**

    s = sign($x_{i+1} - x_i$) 在坐标 (x_i,y_i) 后面填充坐标 $(x_i+s,y_i),(x_i+2*s,y_i),...,(x_{i+1}-s,y_i)$

  end

  else if $|x_i - x_{i+1}| \geqslant 1$ $|y_i - y_{i+1}| \geqslant 1$ **then**

    xs = sign($x_{i+1} - x_i$) ys = sign($y_{i+1} - y_i$) 在坐标 (x_i,y_i) 后面填充坐标 $(x_i+xs,y_i),(x_i+xs,y_i+ys)$ **while** 假设最后填充的坐标 (x_t,y_t) 不满足：$x_t == x_{i+1}$ || $y_t == y_{i+1}$ || $(|x_t - x_{i+1}| == 1$ $|y_t - y_{i+1}| == 1)$ **do**

      在坐标 (x_t,y_t) 后面填充坐标 $(x_t+xs,y_t),(x_t+xs,y_t+ys)$

    end

    if $x_t == x_{i+1}$ $|y_t - y_{i+1}| > 1$ **then**

      s = sign($y_{i+1} - y_t$) 在坐标 (x_t,y_t) 后面填充坐标 $(x_t,y_t+s),(x_t,y_t+2*s),...,(x_t,y_{i+1}-s)$

    end

    else if $y_t == y_{i+1}$ $|x_t - x_{i+1}| > 1$ **then**

      s = sign($x_{i+1} - x_t$) 在坐标 (x_t,y_t) 后面填充坐标 $(x_t+s,y_t),(x_t+2*s,y_t),...,(x_{i+1}-s,y_t)$

    end

    else if $|x_t - x_{i+1}| == 1$ $|y_t - y_{i+1}| == 1$ **then**

      s = sign($x_{i+1} - x_t$) 在坐标 (x_t,y_t) 后面填充坐标 (x_t+s,y_t)

    end

  end

until *i=1,2,...,l-1*;

4.2.3 属地轨迹生成模型估计

针对轨迹数据网格化和轨迹数据填充后的轨迹子簇，通过马尔科夫链模型来估计其轨迹生成模型。假设子簇 C 中含有 l 条轨迹 $C=\{t_1,t_2,...,t_l\}$，因为每条轨迹都需要经过轨迹数据网格化和轨迹数据填充两个操作，故子簇 C 中每条轨迹包含的坐标点个数将不再相同。记子簇中轨迹所有坐标点的集合为 S，集合 S 中元素个数为 |S|, 将集合 S 中的每一个元素看做是马尔科夫链中的一种状态，其转移矩阵记为 P 可表示为：

$$P=\begin{array}{c|cccc} c|cccc & s_1 & s_2 & \cdots & s_{|s|} \\ \hline s_1 & p_{11} & p_{12} & \cdots & p_{1|s|} \\ s_2 & p_{21} & p_{22} & \cdots & p_{2|s|} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_{|s|} & p_{|s|1} & p_{|s|2} & \cdots & p_{|s||s|} \end{array} \tag{4-1}$$

轨迹子簇 C 对应的似然函数可以表示为：

$$L=p_{11}^{n_{11}}*p_{12}^{n_{12}}*p_{1|s|}^{n_{1|s|}}*...*p_{|s||s|}^{n_{|s||s|}}$$

对上式对数似然函数取对数不影响似然函数取最大值时的参数取值，故轨迹子簇 C 对应的对数似然函数可以表示为：

$$LL=n_{11}\ln(p_{11})+n_{12}\ln(p_{12})+...+n_{|s||s|}\ln\left(p_{|s||s|}\right) \tag{4-2}$$

求解式4-2的最大值所得参数矩阵作为马尔科夫链转移矩阵。

为了更清楚上式的求解过程，我们针对其中任意一个状态 A 的转移概率来进行求解，假设状态 A 在轨迹子簇 C 中可以转移到状态 B、状态 C 和状态 D，对应的转移概率记为 p_1,p_2p_3，且在轨迹子簇 C 中发生的次数分别记为 n_1,n_2n_3，设定轨迹子簇 C 包含的轨迹数据是充分的，即状态 A 只能转移到以上三个状态，此时则有：$p_1+p_2+p_3=1$。针对状态 A 的对数似然函数可以表示为：

$$LL(A)=n_1\ln(p_1)+n_2\ln(p_2)+n3\ln(1-p_1-p_2)$$

LL(A) 分别对 p_1 和 p_2 求导：

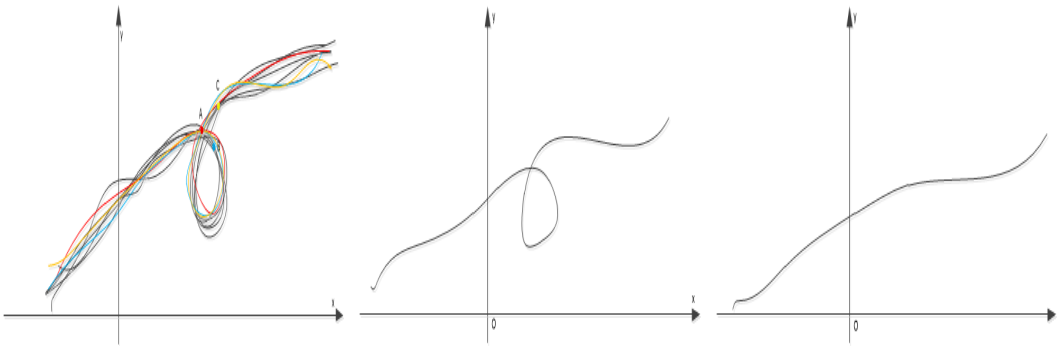
$$\begin{cases} \frac{\partial LL(A)}{\partial p_1} = \frac{n_1}{p_1} + \frac{-n_3}{1-p_1-p_2} \\ \frac{\partial LL(A)}{\partial p_2} = \frac{n_2}{p_2} + \frac{-n_3}{1-p_1-p_2} \\ p_1 + p_2 + p_3 = 1 \end{cases}$$

通过求解上式方程组可得到方程组的解为：

$$\begin{cases} p_1 = \frac{n_1}{n_1+n_2+n_3} \\ p_2 = \frac{n_2}{n_1+n_2+n_3} \\ p_3 = \frac{n_3}{n_1+n_2+n_3} \end{cases}$$

依照上式解的形式，转移矩阵可以在 O(n) 时间复杂度内可以得到。

以上构建的模型在轨迹无交叉点的时候能够使用，当轨迹子簇中大部分轨迹形状走势存在交叉点时，我们可以简单分析这种情形下上式模型的不足之处。假设现有轨迹子簇 C 如图4-4(a)所示，轨迹依据时间顺序从左下角依次经过 A 点和 B 点，最后经过 C 点到达整个轨迹的右上角，此时轨迹子簇中许多轨迹存在交叉点 A，轨迹在 A 点既可以转移到 B 点，同样可以转移到 C 点，而这两种转移将对应两种完全不同的轨迹走势。当轨迹在 A 点转移到 B 点时，依据转移矩阵生成轨迹坐标点时会再次经过 A 点，然后经过 C 点，此时轨迹形状走势满足整个轨迹子簇形状走势，如图4-4(b)；而当轨迹在 A 点转移到 B 时，依据转移概率矩阵，轨迹将直接经过 C 点顺势而上到达轨迹右上角，最终将生成许多类似如图4-4(c)的轨迹，而这些轨迹与整个轨迹子簇的轨迹形状走势存在较大的差异。



(a) (b) (c)

图 4-4 轨迹子簇中存在交叉点的情形

为了解决轨迹子簇中轨迹存在交叉点情况，我们可以将二维的轨迹想象成三

维轨迹在平面上的投影，在二维平面上 A 点是一个点，但对应到三维空间则是两不同个点，只是它们在平面上的投影相同而已。故我们可以将 A 点对应的状态分成两个状态，记为 A1 和 A2，若轨迹子簇中存在如图4-4(c)的轨迹，则这些轨迹经过 A 点时状态记为 A1，过若轨迹子簇中存在如图4-4(b)的轨迹，则这些轨迹在经过 A 点时状态记为 A2，将交叉点对应的状态一分为二，后续求解问题则如同轨迹无交叉点的情形一致。通过把交叉点对应状态一分为二的策略，即可在轨迹子簇中轨迹存在交叉点情况下也能准确描述轨迹生成模型。

针对指定的轨迹子簇，其马尔科夫链模型的转移矩阵可以表示为式4-1，其状态与轨迹子簇中轨迹坐标点对应，若将马尔科夫链通过图模型表示：所有状态构成图中的点集，状态与状态之间的所有转移构成了图的边集。显然，用于估计轨迹生成模型的马尔科夫链模型对应的图模型不是一个完全图，而是一个稀疏图，因为任意一个状态只能向其在网格空间中相邻的网格点对应的状态进行转移，即任意一个状态只能向极其有限的状态进行转移，这使得马尔科夫链的转移矩阵中存在大量的零元素。于是我们可以通过矩阵稀疏表示方法来代替转移概率矩阵，从而可以减少一部分带宽传输的压力。稀疏矩阵表示方法有很多种，本文采用 CSR 稀疏矩阵表示转移矩阵。

为了中心节点能够对轨迹数据进行有效的还原，除了传输转移矩阵，还需传输一些额外的信息，包括状态集合、初始状态分布和长度限制。将这些额外信息和转移矩阵打包成的指定的数据包称为基于马尔科夫链的轨迹数据单元（MTD-Unit），其格式如下：

States	InitSD	LenMax	TransMat

数据单元格式字段含义如下：

- **States**：此字段包含了转移矩阵中的所有状态，每一个状态代表了网格空间中的一个坐标点，以二元组的形式表示
- **InitSD**：此字段表示轨迹初始状态分布情况
- **LenMax**：此字段限制了生成轨迹序列时的最大序列长度
- **TransMat**：此字段表示马尔科夫链对应的转移矩阵

各个计算节点将所有子簇对应的马尔科夫链模型转化成许多 MTD-Unit 传递给中心节点，中心节点通过这种特殊的数据包能够将轨迹数据进行还原。

4.2.4 综合求解和属地模型应用

各个计算节点将所有轨迹子簇对应的转移概率矩阵、状态初始分布和轨迹输传输给中心节点，中心节点接收到计算节点传递来的各种参数后，首先将转移概率矩阵稀疏表示还原出原始的轨迹转移矩阵，中心节点通过转移概率矩阵来生成轨迹。假设我们将还原的转移概率矩阵记为 A，状态初始概率分布记为 π，轨迹子簇包含的轨迹数量记为 N，生成轨迹过程主要流程如下：

算法 4-2 马尔科夫链模型轨迹生成过程

Data: 转移概率矩阵 A，状态初始概率分布 π，轨迹数量 N

Result: 生成轨迹集合 G

```
for i=1,2,,...,N do
    按照状态初始状态分布 π 产生状态 s1;
    状态 1 在转移概率矩阵中对应的概率分布记为 S;
    轨迹序列 L;
    repeat
        依据概率分布产生状态 s;
        将状态 s 加入轨迹序列 L;
        S= 状态 s 在转移概率矩阵中对应的概率分布;
    until 若概率分布 S 中存在非零元素;
    将轨迹序列 L 纳入生成轨迹集合 G;
end
```

依据上述流程即还原出轨迹数据，而此时的轨迹数据长度是不一致的，所以式3-1定义的距离度量将不再适用，此时我们将采用霍夫曼距离来度量两条轨迹之间的距离，现有轨迹 A 和轨迹 B，其间的距离 $d_H(A,B)$ 可表示为：

$$d_{\mathrm{H}}(A,B)=\max\left\{\sup_{a}\in A\inf_{b}\in Bd(a,b),\sup_{b}\in B\inf_{a}\in Ad(a,b)\right\} \tag{4-3}$$

其中 a,b 分别表示轨迹 A 和 B 上的点，d(a,b) 表示 a 点和 b 点之间的欧氏距离。将式4-3定义的距离替换 kmedoids 算法中的欧氏距离，则可以对生成的轨迹数据集进行 kmedoids 算法聚类操作了。具体聚类流程如算法9所示。

算法 4-3 kmedoidspp*initialization*

Data: 样本距离矩阵 M_{nn}，簇个数 K

Result: 簇心向量序号 $\mu = \{\mu_1, \mu_2, ..., \mu_K\}\}$

从 0 到 n-1 中随机选取一个整数 μ_1，将其作为第一个簇心对应的元素序号，此时 $\mu = \mu_1$;

D=1,2,...,n-1;

repeat

  R=D-μ **for** t_i *in* R **do**

    计算 $P(t_i)$，$P(t_i) = \omega \sum_{\mu_i \in \mu} M^2_{t_i,\mu_i}$，其中 ω 是归一化系数;

  end

  按照 R 集合中所有点的概率分布选出 μ_k，将其加入集合 μ;

until *k=2,3,...,K*;

算法 4-4 k-medoids++ 算法

Data: 样本距离矩阵 M_{nn}，簇个数 K

Result: 簇划分 $C = \{C_1, C_2, ..., C_k\}\}$

kmdoidsspp$_i$*nitialization*(M$_{nn}$,K)

repeat

  将所有的簇置为空，$C_i = \varphi \ \ (1 \leqslant i \leqslant K)$

  for *j=1,2,...,m* **do**

    依据距离矩阵，计算序号为 i 的样本与所有簇心之间的距离，并将序号为 i 的样本归为第 α_i 类，使其满足：$\alpha_i = arg \ \min(M[ialpha_i])$

  end

  for *i=1,2,...,K* **do**

    $\hat{\mu}_i = arg \ \min_{x \in C_i} \left(\sum_{j \in C_i} M[x,j] \right)$

    if $\hat{\mu}_i \neq \mu_i$ **then**

      μ_i=$\hat{\mu}_i$

    end

  end

until 所有簇心不再更新或更新值小于一定阈值;

中心服务器计算出 k 个均簇心向量后，将 k 个簇心向量回传给各个计算节点，当计算节点侦查到新轨迹时，判断其与回传的 k 个簇心向量的距离来判定是否为异常轨迹：若新轨迹与所有均值向量距离都超过了设定的阈值，则可以判定为轨

迹为异常轨迹。

4.3 实验与分析

4.3.1 理论分析

（1）时间复杂度

计算节点时间复杂度分为两个阶段: 轨迹预处理和生成模型参数求解。

假设计算节点含有 m 条轨迹，每条轨迹含有 n 个坐标点。轨迹预处理阶段分为轨迹子簇生成个轨迹数据网格化两个步骤。假设轨迹子簇生成中使用的聚类模型为 k-means 算法，其时间复杂度为 O(n*m*k*t)，其中 k 是簇个数，t 表示平均迭代次数。轨迹数据网格化，其时间复杂度为 O(n*m)。生成模型参数求解，其时间复杂度为 O(n*m)。

中心节点时间复杂度分为两个阶段：轨迹生成和全局聚类。假设轨迹数目为 m，每条轨迹包含 n 个点, 轨迹生成时间复杂度为 O(m*n)；全局聚类，其时间复杂度为 O(m*n*k*t)。

（2）带宽消耗

带宽分析可以通过每个轨迹子簇需要传输的带宽层面进行分析，假设一个轨迹子簇含有 m 条轨迹，每条轨迹含有 n 个坐标点，则将子簇中的轨迹坐标在网络中传输的数据量为 m*n*2。通过 MTD-Unit 数据包传输所需要在网络中传输的数据量记为 Q，则网络通信量压缩比为 m*n*2/Q。其中，由 MTD-Unit 数据包字段含义可得，当子簇中的轨迹数量增加，初始分布、转移矩阵和状态集合其大小均不会发生很大的变化，故随着轨迹数量规模的增大，算法将会带来更有效的网络数据压缩能力。

（3）隐私性

算法隐私性层面主要从两个角度进行考量：不确定性和覆盖率。

不确定性通过生成轨迹数据与原始属地轨迹数据之间的相似度进行度量，由马尔科夫链生成轨迹数据过程可知，生成的轨迹数据簇与对应的原始轨迹子簇整体上有着相似的分布，但针对单个轨迹，生成的轨迹可能与原始轨迹子簇中的任意一条轨迹都不相同，故该算法在不确定层面将有着很好的表现。算法在隐私性的覆盖率方面，由于属地节点是针对本地所有轨迹数据来训练马尔科夫链模型，故算法的覆盖率为 1。

4.3.2 实验与分析

实验是在真实的数据集上进行的，这些数据集包含了在大阪的 ATC 购物中心的游客的轨迹。在我们的实验中，我们应用了时间、位置 x、位置 y、人物 id 等字段[①]。经过预处理后，我们选择了 4939 条轨迹作为原始数据集，每条轨迹包含 500 个点。

实验系统由三个计算节点和一个中心节点组成。将该算法与两种聚类方法进行了比较。第一种算法对整个网格化数据集执行标准的 kmedoids 算法 (baseline)，baseline 的结果视为最优结果，后续试验中使用的 ARI 指数则是以 baseline 聚类结果作为标签的。第三种算法则是本章提出的 MCD-Clustering 算法。

对于三种聚类方案中，都需要对簇个数进行选择，我们以 baseline I 实验中选取的 k 视为标准，在 baseline I 实验中选取不同的 K 值其损失函数值变化情况如图4-5。

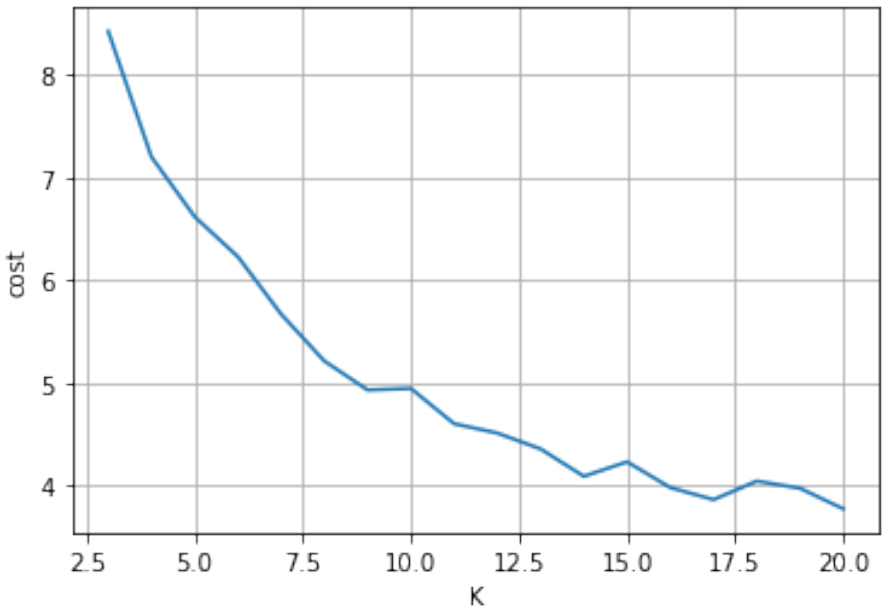
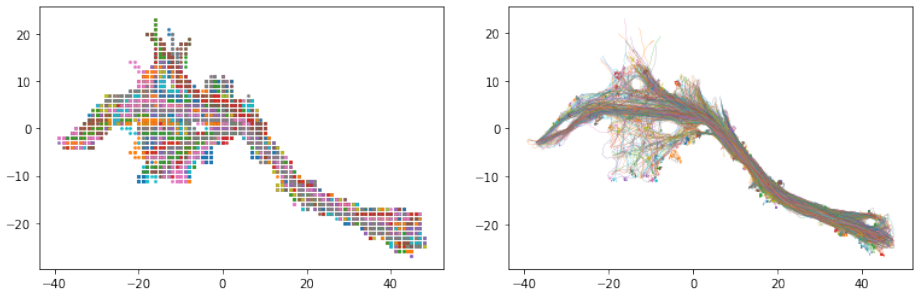


图 4-5 不同 K 值损失函数变化

通过上图可以确定全局聚类簇个数为 8。

各个属地节点完成模型训练完成后，将 MTD-Unit 传递给中心节点，中心节点依据算法7生成轨迹数据，生成后的轨迹数据与原始轨迹数据如图4-6所示，其中图4-6(a)表示生成的网格化轨迹数据点，图4-6(b)表示原始轨迹数据。

① https://irc.atr.jp/crest2010$_{HRI}$/$ATC_{d}ataset$/



(a) (b)

图 4-6 轨迹生成效果对比图

在 MCD-Clustering 算法中的局部聚类过程，K 值选取将会影响马尔科夫链对轨迹生成模型的估计，图4-7展示了局部聚类不同 K 值下的 RI 指数变化。（保持全局聚类的 K 值不变）

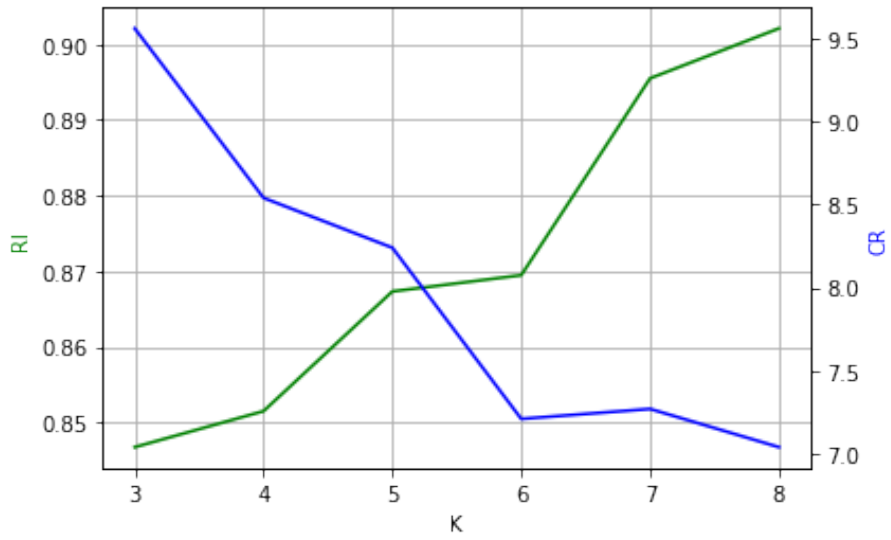
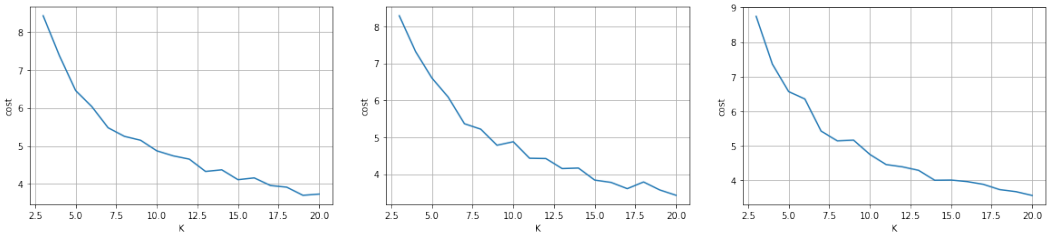


图 4-7 不同 K 值对实验结果的影响

依据上图，可以看到，随着 K 值增大，ARI 指数也会随之增大，即聚类效果更好。这是因为当 K 值很小时，轨迹子簇中的轨迹相似性不高，可能会有多种走势形状的轨迹存在，通过这样的轨迹子簇数据训练出来的生成模型会生成多种走势形状混合的轨迹数据，而走势性形状混合的轨迹数据在原始轨迹数据中是不存在的，这样就会给后续全局聚类操作造成影响。但随着 K 值增大，需要在网络中传输更多的转移矩阵，也因此会使网络信息传输压缩率有所降低, 但 K 值增大的同时，子簇更具有相似性，故而转移矩阵也会相应减小，可以看到当 K 大于等于 6 时，CR 值变化放缓。

为了网络带宽消耗和聚类准确度的平衡，对于局部聚类的 K 值选择，我们依据 K 值与损失函数的变化曲线来决定 K 的取值，实验结果如图4-8所示, 图中分别展示了三个属地节点局部聚类 K 值选择过程:



(a) (b) (c)

图 4-8 各节点局部聚类 K 值与损失函数曲线

依据上图，确定三个属地节点 K 值取值分别为 7,7,9。在此基础上，综合中心进行全局聚类的 K 值选择也采用同样的方法，实验结果如图4-9所示：

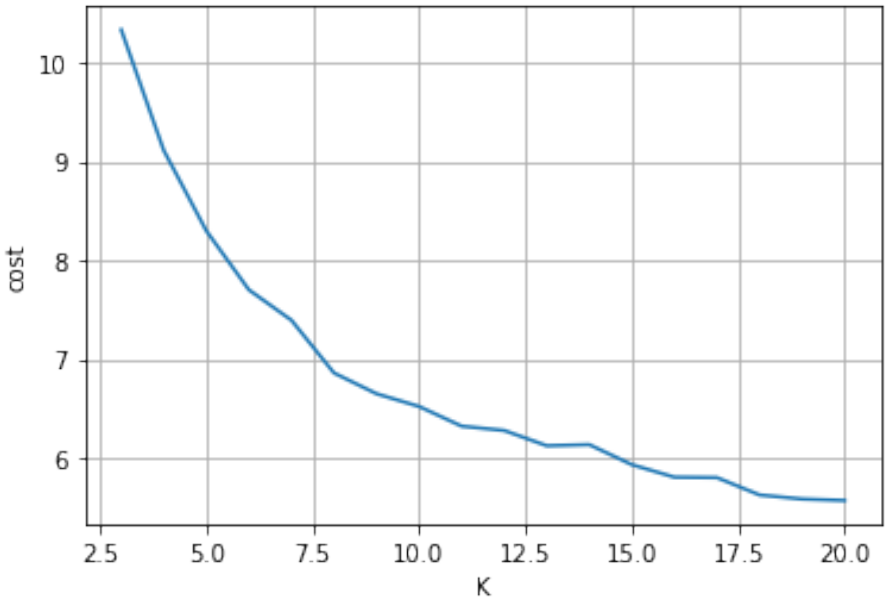


图 4-9 全局部聚类 K 值与损失函数曲线

在 MCD-Clustering 算法中对轨迹数据进行了网格化处理，假设网格化粒度记为 δ，若 δ 越大则表示网格化粒度越粗，反之，则网格化粒度越精细。我们对不同网格化粒度情况下 MCD-Clustering 算法的聚类效果进行了实验，实验结果如图4-10。

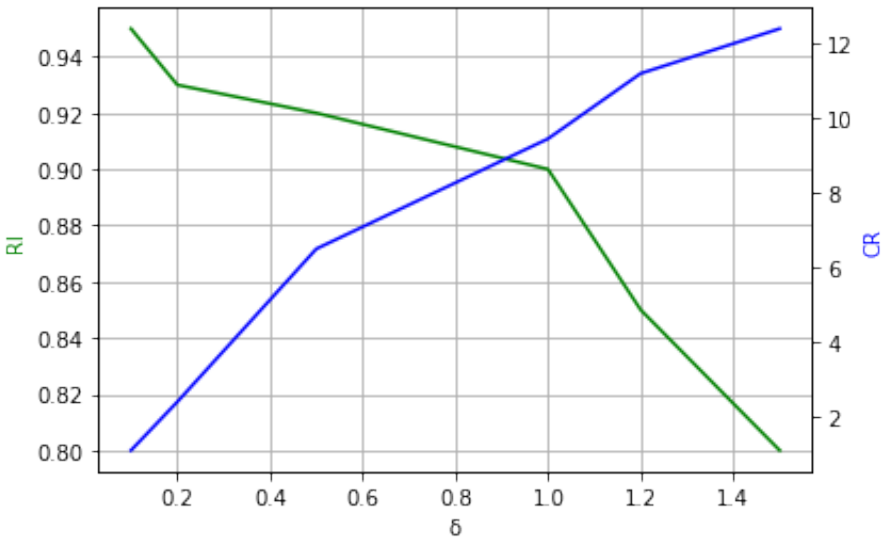


图 4-10 不同网格粒度对实验结果的影响

从上图可以看出，当网格粒度越来越精细，ARI 指数也会随之增大，即聚类效果越来越好。但随着网格粒度越来越精细，需要在网络中传输的转移矩阵会越来越大，也因此会使网络信息传输压缩率有所降低，相反，网格粒度越粗，需要在网络中传输的转移矩阵会越来越小，网络信息传输压缩率将会有所增长。

4.4 本章小结

本章提出了一种基于马尔科夫链的分布式聚类算法 MCD-Clustering，该算法利用马尔科夫链模型来估计轨迹子簇的数据分布，在实现了分布式轨迹聚类的同时也考虑了数据隐私问题和网络带宽消耗问题，相比于第三章的 CSD-Clustering 算法，在隐私性方面（不确定性和覆盖率）本章提出的算法增强了不确定性，但 CSD-Clustering 算法在覆盖率层面要优于本章算法；在计算性能层面，由于本章算法在求解参数方面无需迭代求解，故计算性能要略优于第 CSD-Clustering 算法；在带宽消耗层面，当数据规模不大时，CSD-Clustering 算法由于进行了轨迹数量抽样，其带宽消耗可能略优于本章算法，当数据规模逐渐增大时，本章算法在带宽消耗方面将会优于 CSD-Clustering 算法。本章通过实验验证了该算法的有效性。

第五章 分布式原位聚类算法系统实现

5.1 概述

随着数据规模的快速增长，数据集中式存储的弊端越来越明显，分布式数据存储称为了常用的数据存储方式。在分布式场景下，涉及全局数据的计算则需要把各个节点数据通过网络传输到单个节点，然后再进行计算；然而很多隐私数据或涉密数据直接在网络中传输是非常危险的，可能导致机密数据的泄露或用户隐私的泄露问题，这无疑会对很多大数据公司、国家和广大用户造成巨大的损失。聚类算法，是非常依赖全局数据进行计算的，考虑数据隐私性的分布式聚类算法是目前亟待解决的问题。本系统设计背景是实验室原位计算平台，该平台则是围绕上述涉及到的问题提出一种系统解决方案，平台上聚类任务的实现则是采用了本文第三章和第四章所提出的算法。本系统开发主要利用 Java+ZooKeeper+Apache 以及 Mysql 完成，其中，前端可视化主要利用 JavaScript+Bootstrap 进行开发，系统运行环境主要为 Ubuntu14.04 ，系统设计框架如图5-1所示。

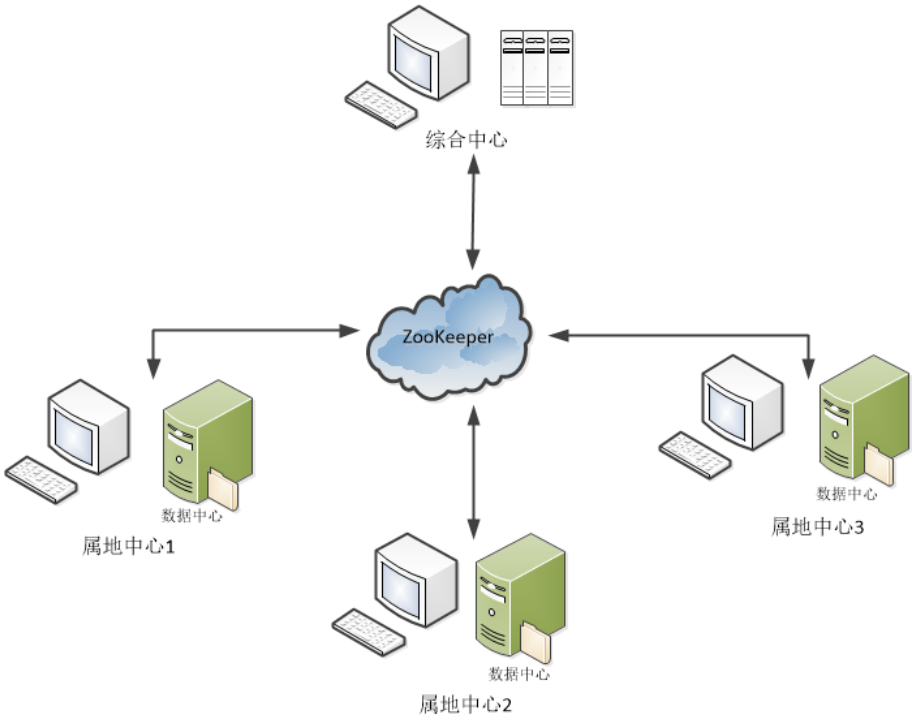


图 5-1 系统架构设计图

5.2 总体设计

原位计算平台旨在为分布式各类计算任务提供解决方案，轨迹聚类计算是此平台中重点攻克的计算任务之一。本节提出了原位计算平台中轨迹聚类计算部分所涉及到的模块，主要由五个模块组成：模型训练模块、网络通信模块、综合计算模块、聚类评估模块和可视化展示模块。各模块层级结构如图5-2所示。

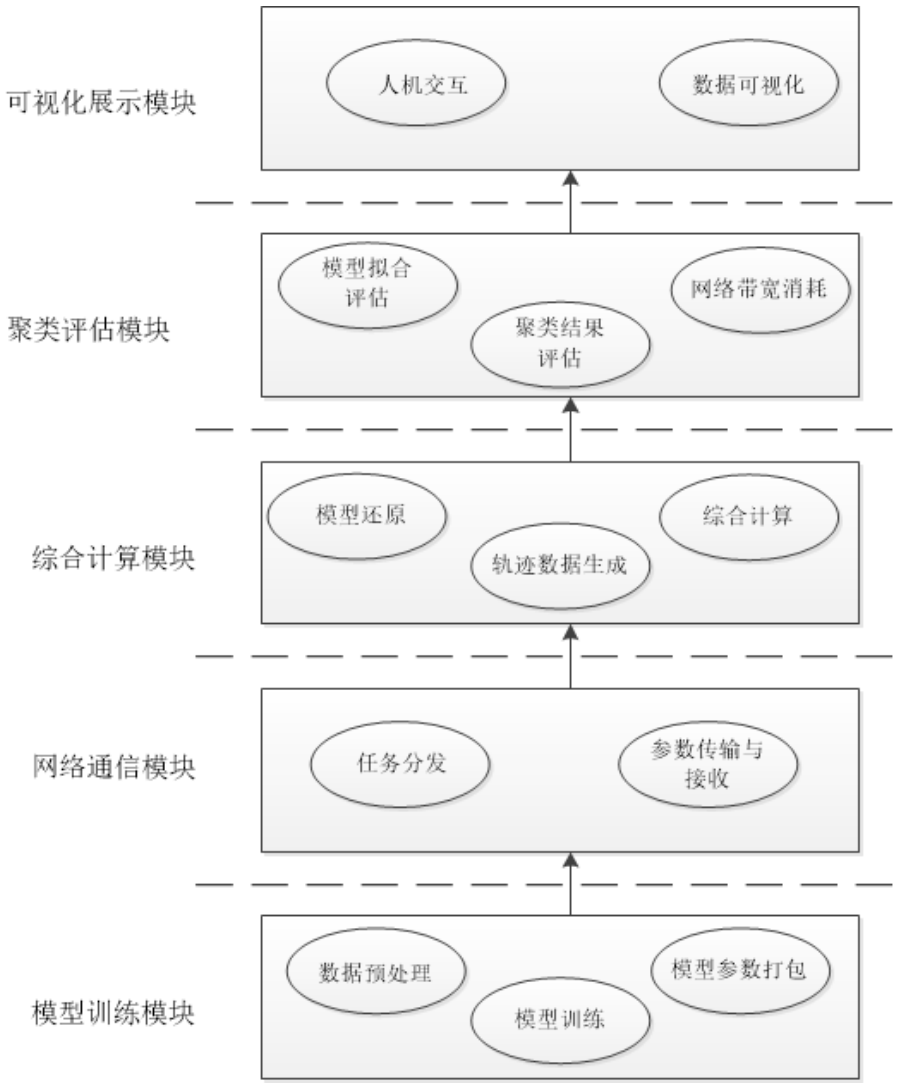


图 5-2 总体模块结构图

现对各个模块的功能作简要阐述：

1. 模型训练模块

该模块主要负责各属地中心的局部模型训练，模块由三个部分组成：数据预处理、模型训练和模型参数打包。数据预处理部分负责将属地中心原始数据进行加工处理，使得加工后的数据能够直接用于模型训练；模型训练部分则利用数据预处理产生的数据进行轨迹生成模型估计；模型参数打包部分则是

将模型训练得到的模型参数结构化，便于综合中心对于模型参数的处理。

2. 网络通信模块

该模块主要负责综合中心与各个属地中心的网络交互，模块由两个部分构成：任务分发和参数的传递与接收。任务分发部分负责将综合中心需要执行的计算任务分发至各个属地中心；参数的传递与接收部分主要负责综合中心和属地中心之间的信息传递以及信息同步操作。

3. 综合计算模块

该模块主要负责综合中心对全局数据的计算处理，模块由三个部分组成：模型还原、数据生成和综合计算。模型拟合评估是属地中心对局部生成模型拟合准确度的评估；模型还原部分是将打包好的局部模型参数还原出属地中心训练得到的轨迹生成模型；数据生成部分则通过还原的轨迹生成模型产生轨迹数据，以此来近似属地中心的轨迹数据；综合计算部分将生成的全局轨迹数据输入到聚类模型完成此次聚类计算任务。

4. 系统评估模块

该模块主要负责对聚类计算任务结果进行评估，模块由三部分组成：模型拟合评估、聚类结果评估和带宽消耗。模型拟合部分聚类结果评估度部分对综合计算得到的聚类结果进行评估；带宽消耗部分对此次聚类计算任务中造成的网络带宽消耗做出评价。

5. 可视化展示模块

该模块主要是通过可视化的界面对综合中心和属地中心的状态进行呈现，对整个系统的运行情况起到监察和管理作用。

5.3 详细设计

5.3.1 模型训练模块

模型训练模块是属地中心利用本地轨迹数据训练轨迹生成模型的模块，其中模型的选择十分重要，对算法的准确度、带宽消耗和数据隐私性都有着至关重要的影响。第三章基于复合采样的分布式轨迹聚类算法选择了多项式拟合技术来生成轨迹数据，第四章基于马尔科夫链的分布式轨迹聚类算法选择了随机游走模型来生成轨迹数据。前者在轨迹数据还原程度上更加准确，但由于需要对每一条轨迹建模，所以带宽消耗比较大；后者在轨迹数据还原程度上没前者精确，但可以很好还原轨迹数据分布情况，而且在带宽消耗上也得到了缓解。

第三章模型训练模块流程如图5-3所示：

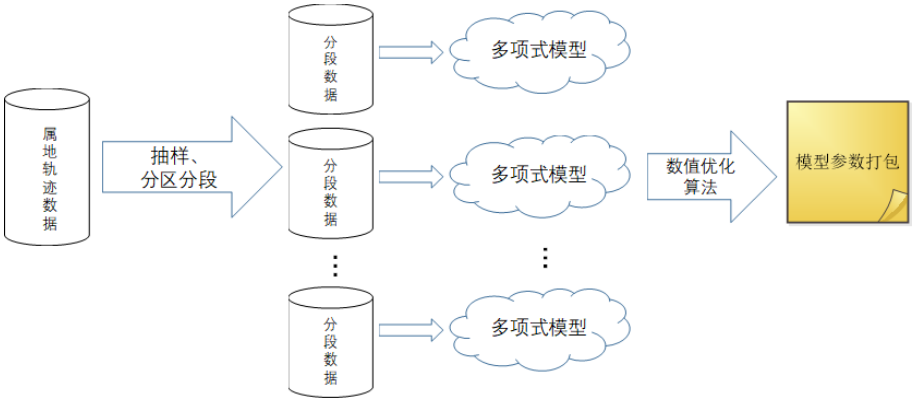


图 5-3 CSD-Clustering 模型训练模块流程

在第三章算法中，属地中心首先对属地轨迹数据集进行了数据预处理，预处理包括轨迹数据抽样、轨迹点数抽样、分区和分段操作，然后针对每个分段轨迹构造多项式模型，利用最优化理论求解出模型参数，将模型参数格式化后形成模型参数包，模型参数包将接着由网络通信模块进行处理。

第四章模型训练模块流程如图5-4所示：

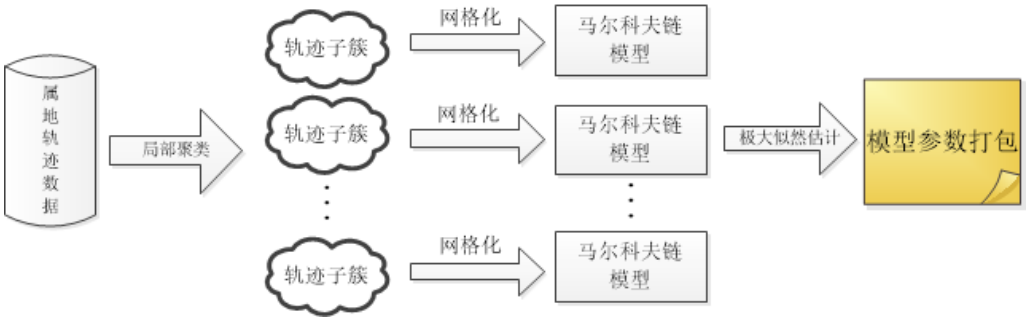


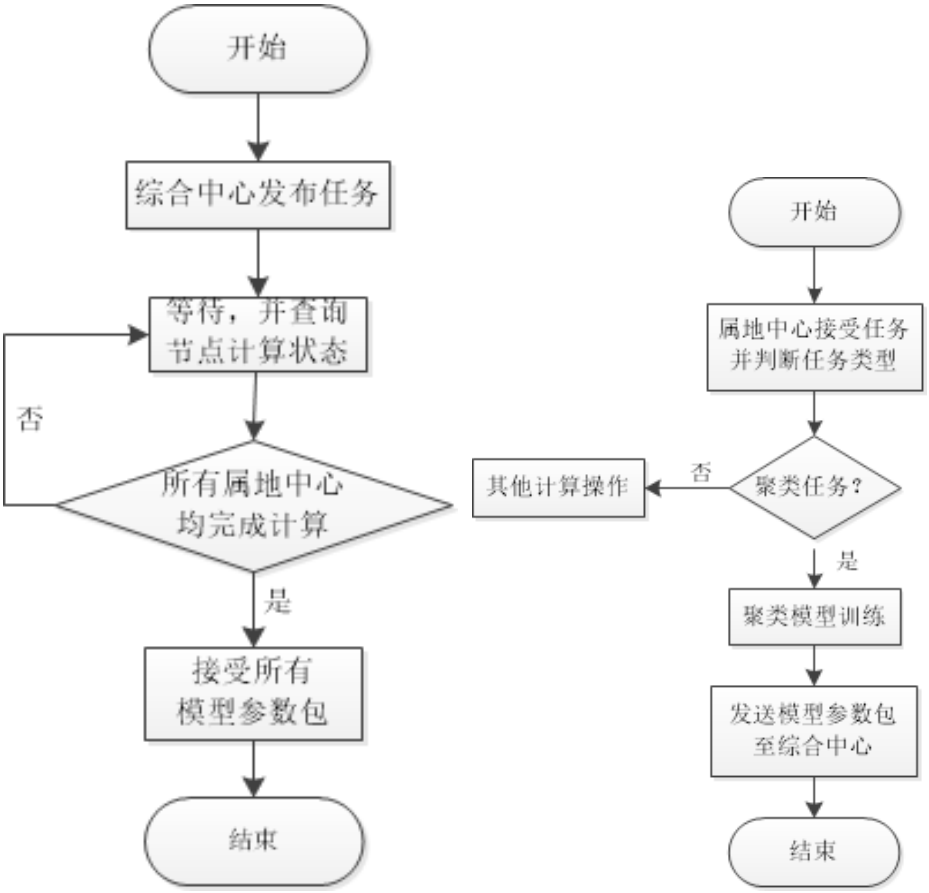
图 5-4 MCD-Clustering 模型训练模块流程

在第四章算法中，属地中心首先对属地轨迹数据进行局部聚类处理，经过局部聚类处理后将得到若干个轨迹子簇，依据轨迹间距离度量，在各个轨迹子簇中的轨迹在轨迹走势上具有很大程度上的相似性，然后对所有轨迹进行网格化处理，使轨迹中的所有点都由网格中的点构成，对于每个网格化活动轨迹子簇数据，构造一个马尔科夫链模型，并通过极大似然估计出概率转移矩阵中的所有转移概率，将求解出来的转移概率参数和网格化轨迹信息格式化后打包成模型参数包，模型参数包将接着由网络通信模块进行处理。

5.3.2 网络通信模块

网络通信模块提供了综合中心和属地中心在分布式环境中的相互通信功能。网络通信模块功能主要是通过 ZooKeeper 框架实现的，综合中心通过 ZooKeeper

的配置文件配置所有其他属地节点的 ip 地址，所有属地节点则在 ZooKeeper 中配置综合中心的 ip 地址。基于 ZooKeeper 实现的网络通信模块主要实现综合中心的任务分发以及所有节点（包括综合节点）的参数发送和接受，其流程如图5-5所示，其中图5-5(a)展示了综合中心网络通信流程，图5-5(b)展示了属地中心网络通信流程。



(a) (b)

图 5-5 网络通信流程图

当综合中心发放任务时，会通过 ZooKeeper 向分布式环境中的所有属地中心发送计算任务信息，对于此任务，综合中心会进入等待状态，并周期性地查看计算状态，直至发现所有属地中心的计算结果均已返回，则会一次性读取所有返回的模型参数包，并进行接下来的综合计算。属地中心接收到综合中心发放的任务，首先解析任务类型，然后依据任务类型执行不同的计算操作，计算完成后将计算得到的模型参数包发送至综合中心。

5.3.3 综合计算模块

综合计算模块输入为网络通信模块传递过来的模型参数包，输出为全局聚类结果。此模块主要包含模型还原、轨迹数据生成和综合计算三个部分，若局部模型使用的模型不同，这三个部分的操作都将会不同。图5-6展示了分别选择CSD-Clustering 算法和 MCD-Clustering 算法所对应的综合计算流程。

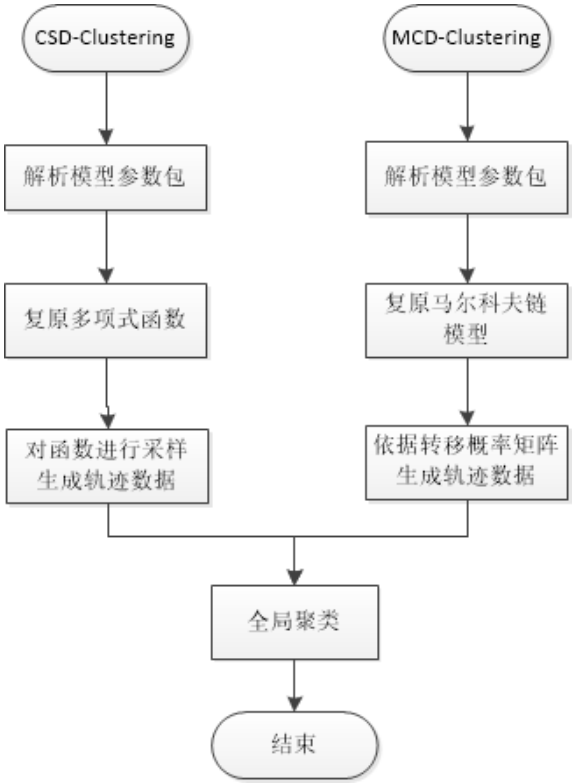


图 5-6 综合计算模块流程图

对于 CSD-Clustering 算法对应的综合计算流程，首先从模型参数包中解析出多项式模型的系数参数，从而恢复多项式函数模型，然后从该函数模型上进行等间距采样，将采样得到的坐标点作为轨迹数据；对于 MCD-Clustering 算法对应的综合计算流程，首先从模型参数中解析出概率转移矩阵和初始状态，然后依据马尔科夫链产生数据的方式生成状态序列，状态序列可以转换为估计及坐标序列；得到生成的全局轨迹数据后，把轨迹数据作为输入数据输入到聚类模型中，在 CSD-Clustering 算法中生成的轨迹数据长度是相同的，故可以采用欧氏距离定义的 k-means 算法进行全局聚类操作，在 MCD-Clustering 算法中生成的轨迹数据长度不等，采用基于霍夫曼距离的 k-medoids 算法进行全局聚类操作，产生的聚类结果会输入到系统评估模块进行性能评估。

5.3.4 系统评估模块

系统评估模块故能是对此次聚类任务完成质量评估，该模块从三个方面对任务完成质量进行评估，如图5-7所示，分别为：局部模型拟合评估、全局聚类评估和带宽消耗。

局部模型拟合评估对于不用的局部模型评价方式也不同，对于 CSD-Clustering 算法，可以直接通过损失函数来刻画；对于 MCD-Clustering 算法，可以通过原始子簇轨迹集合和生成子簇轨迹集合之间的霍夫曼距离来刻画。全局聚类评估是对此次聚类任务结果进行评价，由于综合中心也不知道真实轨迹数据，所以只能采用内部指标对聚类本身进行评价，该系统采用 DBI 指数对聚类结果进行评价。带宽消耗是对本次聚类任务产生的网络通信流量情况进行展示，并原始轨迹数据统计信息可以计算出信息压缩比，体现本文提出的算法在带宽消耗上的改进。

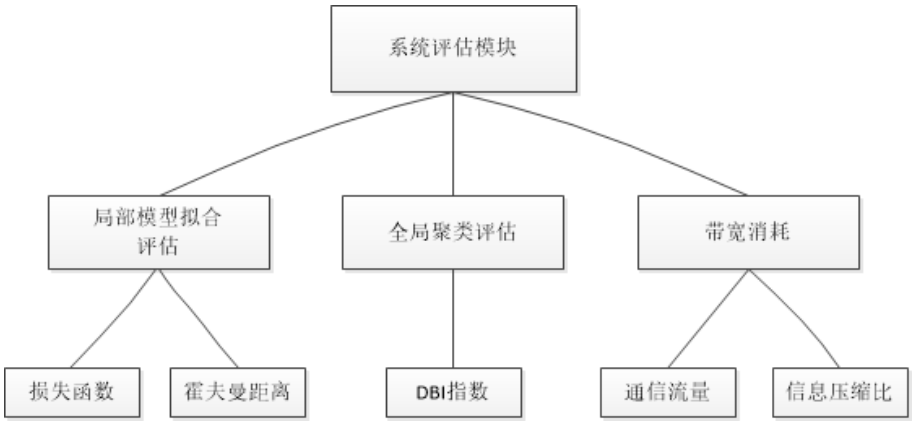


图 5-7 系统评估模块

5.4 平台展示

本系统基于 B/S 架构设计，系统登录界面如图5-8所示，各个属地中心都通过分配的账号访问系统。

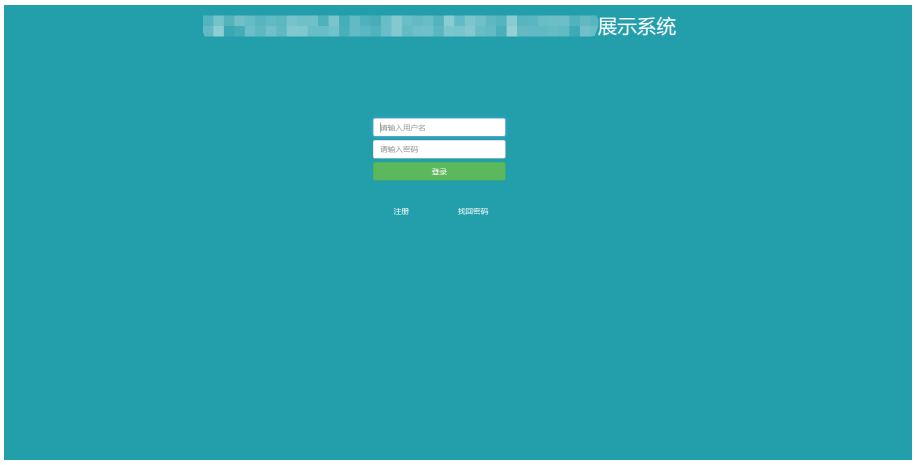


图 5-8 系统登录界面

登入系统，系统首页如图5-9所示，系统首页展示了数据中心的地理分布。除此之外，系统总共分为六个部分：系统展示屏、业务操作平台、任务管理、系统管理、数据管理和内部威胁检测。本章节将针对本文涉及算法介绍其中的业务操作平台和任务管理两部分。

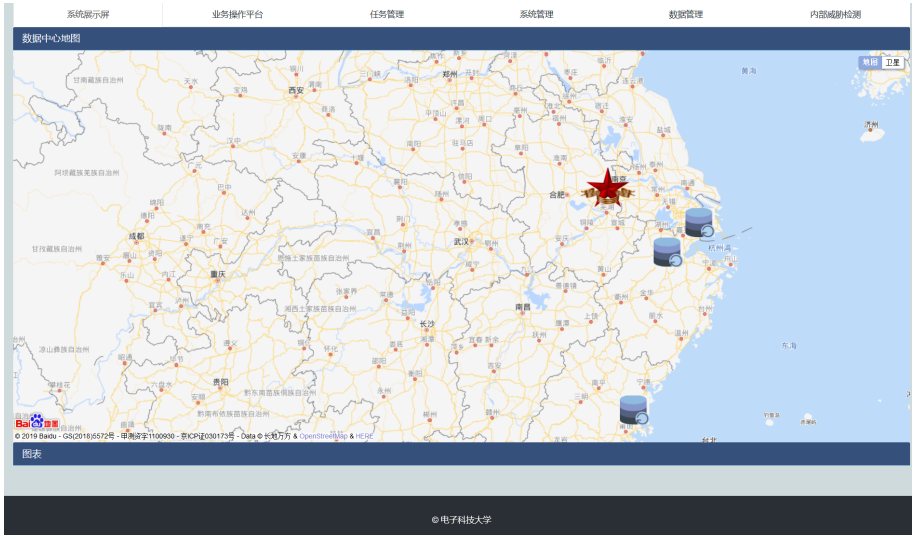


图 5-9 系统主界面

业务操作模块页面如图5-10所示，该模块通过可视化移拖拽、连接算子来完成自定义任务的发放，页面分为三个部分，左边部分用来展示可使用的算子，其中异常轨迹查询需要使用分布式轨迹聚类算法；页面的中间部分则为算子操作空间，专业人员可以通过组合算子来形成自定义任务；页面的右侧为算子属性参数，在这一栏可以定义算法所需参数。

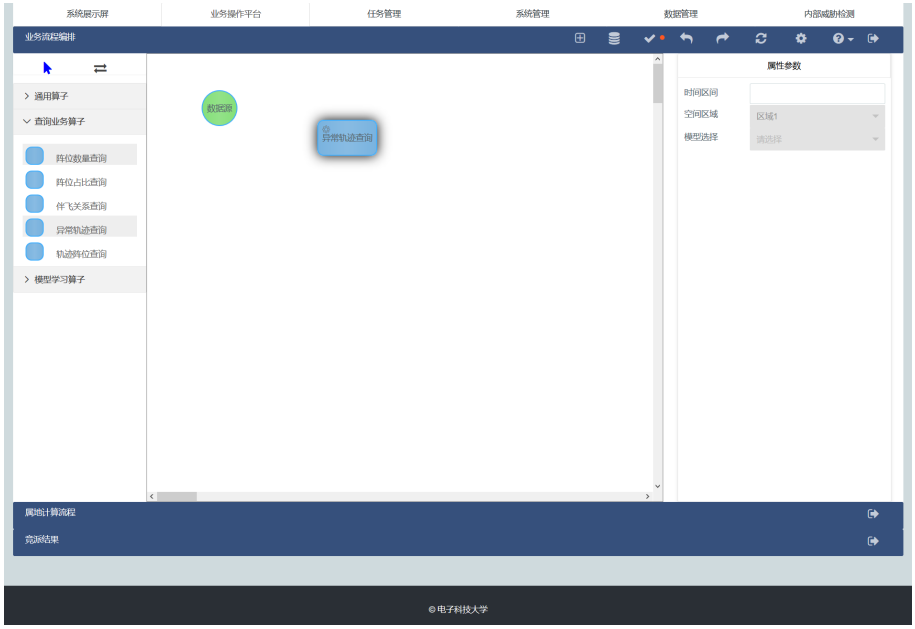


图 5-10 业务操作模块界面

任务管理模块如图5-11所示，该模块功能由两部分组成：查看历史任务和编辑任务。查看历史任务可以查看平台上所有已发布任务以及任务相关信息，包括任务编号、任务发起单位、任务当前状态、任务完成时间和数据源数目。

图 5-11 任务管理模块

若任务正在执行，点击单个任务则可以查看任务相关节点信息，展示信息包括：分中心任务编号、所属集群 IP、所属任务 ID、当前运行状态和任务类型，如图5-12所示。

图 5-12 查看任务相关信息

若任务已经执行完成，则点击任务则可查看任务结果，如图5-13所示，图中展示了轨迹聚类完成的效果图。

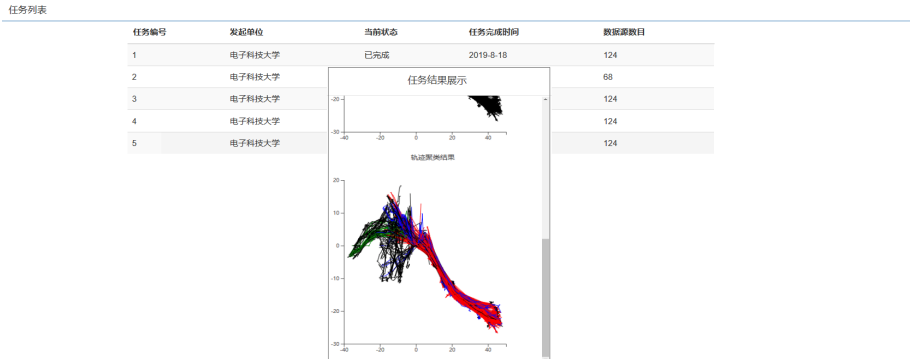


图 5-13 任务执行结果展示

5.5 本章小结

本章利用前面部分提出的算法模型，设计并实现了分布式计算平台。首先介绍了系统的总体设计以及对应的体系架构，然后详细概述系统中的主要子模块的功能和作用，具体包括模型训练模块、网络通信模块、综合计算模块和聚类评估模块。最后通过可视化模块展示了系统完成计算任务的过程，使得管理人员能够实时了解任务执行情况。

第六章 总结与展望

6.1 总结

随着移动互联网的到来，数据规模呈现指数级的增长，传统集中式存储方式暴露出越来越多的弊端，分布式存储方式已经被业界广泛使用，而分布式存储涉及到的计算性能、数据隐私性和网络带宽消耗等问题则开始凸显出来。聚类算法，是数据挖掘算法中的一种，这种算法是典型的依赖全局数据的数据挖掘算法，如何在分布式环境中准确地完成聚类计算引起了研究人员广泛的关注；轨迹聚类算法是聚类算法在轨迹数据上的应用，由于轨迹数据的特殊性，很多研究人员针对轨迹数据提出了自己的轨迹聚类算法。本文则是针对分布式轨迹聚类问题提出了两种算法，旨在解决分布式环境下轨迹聚类的数据隐私问题、计算准确问题和网络带宽消耗过高问题。文本的主要内容如下：

（1）首先阐述了分布式轨迹聚类的研究背景，并探讨了研究分布式轨迹聚类算法的意义所在，通过对研究现状进行归纳总结，分析了分布式轨迹聚类当前的发展状况及技术路线，并针对目前存在的不足给出合理可靠的解决方法，然后对方法利用的相关理论与技术进行了探讨，最后对解决方案进行了实验论证与测试。

（2）提出了基于复合采样的的分布式轨迹聚类算法——CSD-Clustering 算法。首先分析了当前分布式轨迹聚类算法存在的分布多样性问题；算法针对轨迹数据的特征采用了多项式拟合与最优化理论结合的方式对轨迹模型进行了拟合，这种轨迹模型拟合的思路能够很好保证全局聚类的准确度，也一定程度地保护了轨迹数据的隐私；除此之外，算法还对轨迹数据集进行了由轨迹数量抽样和轨迹点抽样组成的复合抽样方案，复合抽样方案能够有效减少网络传输的消耗。最后，通过仿真实验对算法的有效性和可行性进行了验证，实验结果表明，CSD-Clustering 算法能够准确完成分布式轨迹聚类计算任务，在隐私性和带宽消耗方面也得到了改善。

（3）提出了基于马尔科夫链的分布式轨迹聚类算法——MCD-Clustering 算法。首先分析了许多分布式聚类算法在处理高维数据时存在的问题；算法针对高维的轨迹数据中各个维度之间的联系，提出了利用马尔科夫链模型估计轨迹子簇的方案，这种思路对于拥有相似走势的轨迹簇能够很好的描述其整体数据分布，故算法为了得到拥有相似轨迹走势的轨迹子簇，在训练马尔科夫链模型之前进行了局部聚类操作；算法在网络中传输转移矩阵以及其他相关信息，并通过稀疏矩阵存储方式来存储转移矩阵，这种方式极大地缓解了网络传输的压力；最后通过实验

对算法的有效性和可行性进行了验证，实验结果表明该算法在计算性能和数据隐私方面相对 CSD-Clustering 算法都有着一定程度的提升，但在聚类准确度方面略差于 CSD-Clustering 算法。

（4）文本为两种分布式轨迹聚类算法完成了系统实现。开发主要利用 Java+ZooKeeper+Apache 以及 Mysql 完成，给出系统的总体架构设计，并对系统的各模块架构设计进行详细说明，通过对数据进行可视化展示表明系统具有良好的应用价值。

6.2 展望

针对分布式轨迹聚类算法研究，本文虽然取得了一定的研究成果，但仍有许多问题有待改善和解决。本文提出的 CSD-Clustering 算法在聚类准确度上有很好的表现，但在数据隐私方面还有待提升；本文提出的 MCD-Clustering 算法对数据隐私有着很好的保护，但在聚类准确度上还有提升的空间。结合分布式轨迹聚类的发展趋势和论文研究过程的思考，对未来的分布式轨迹聚类算法研究工作给出以下几点建议：

（1）关于隐私性层面的度量方法，本文提出了不确定性和覆盖率的度量方法，这种度量方法对算法在隐私性方面是一种定性的评价方式，未来需要提出一种定量的评价指标来使得算法隐私性评价标准更加具体和准确。

（2）本文提出的两种算法在分布式环境中均存在综合中心的角色，该角色在算法中需要对还原的全局数据进行聚类，这对综合中心的计算性能提出了挑战，未来应考虑一种去中心化的算法思路来改善这一现状。

（3）在真实的分布式环境中，往往很多企业和机构不愿意共享自己的数据，这需要分布式平台提供一种有效的激励机制来刺激各企业和机构来共享自己的数据，企业和机构能够从中获得奖励，同时因为各方数据的共享使得更多社会问题得到解决，从而实现双赢的局面。

致 谢

三年的硕士研究生生活转眼就要接近尾声了，而为了组合数学的一次低级失误而懊悔的日子仿佛才刚刚从身边溜走。在校园的时光总是以人们不愿承认的速度在我们面前呼啸而过，回首这三年，除了学到了很多专业知识，但更重要的是，在这段时间中，我开始认真思考自身与世界的关系，思考社会的运作规律并赞叹人类的智慧，思考文化和艺术给个人带来的巨大冲击并对那些伟大的人报以崇高的敬意，“他们在这世界上面不停的奔跑，一不小心就改变了我们；我们在这世界上面不停的奔跑，一不小心就改变了生活”，这些思考让我明白，我们参与社会的方式除了专业，还有常识和勇气，还有“对人类苦难的同情心”，还有对理性的尊重。在我即将离开校园之际，对那些深刻改变了我的思想家，作家，艺术家和企业家，表达我发至内心的感激之情和敬畏之心。

除了那些素未谋面的人，身边的人和事对我生活中的点点滴滴也有着重要的影响。

首先，我要感谢我的导师陈爱国教授，感谢您在学习上对我无私的教导和帮助，让我在学术的海洋里找到自己的方向，也感谢您在生活中对我无微不至的关心和理解，让我倍感温暖。同时，您平时对科学的严谨态度和勤奋努力的精神，我相信实验室的很多兄弟姐妹都有所感触，每当我们开始埋怨睡眠不够或任务量太大时，想起您凌晨三点还在给我们回复微信消息，也立马振奋起了精神；每当我们为学术上的困难准备做出妥协时，您对学术细节的严格把控常常激励着我们迎难而上。您的优秀品质我会永记于心，在往后道路定加倍努力，不辜负您的教导和期望。

同时，我要感谢实验室的郑旭老师，赵太银老师，罗光春老师和其他所有老师，感谢你们的平易近人，让内向的我逐渐融入到实验室的大家庭；感谢我的朋友和实验室的同学，你们在生活中给与我不求回报的关心和帮助，是你们让我研究生的三年时光变得如此开心和愉快，和你们的相处过程中，也学到你们乐观的品质和思考问题的方式，感谢我们的遇见，愿我们的友谊地久天长！

在这个特殊的时期，我还要感谢所有为新型冠状病毒奋斗在一线的医务人员，特别感谢从广东来到我家乡洪湖的医疗支援队，是你们大无畏的精神激励着我们，让我们相信春天总会来临；还要感谢在此期间关心我的陈爱国导师、辅导员和我的朋友，是你们的关心消散了我的焦虑情绪。

最后要感谢我的家人，是你们一直养育我、鼓励我和无条件的支持我，在这

二十多载的岁月里，你们包容着我的任性和脾气的同时还给与了我无限的爱，我会继续努力，在今后的工作中认真努力，让我们的生活更加幸福！

参考文献

[1] L. Wang, J. Tao, R. Ranjan, et al. G-hadoop: Mapreduce across distributed data centers for data-intensive computing[J]. Future Generation Computer Systems, 2013, 29(3): 739-750

[2] J. Dean, G. Corrado, R. Monga, et al. Large scale distributed deep networks[C]. Advances in neural information processing systems, 2012, 1223-1231

[3] L. F. Cranor, J. Reagle, M. S. Ackerman. Beyond concern: Understanding net users' attitudes about online privacy[J]. The Internet upheaval: raising questions, seeking answers in communications policy, 2000, 47-70

[4] J. Vaidya, C. Clifton. Privacy preserving association rule mining in vertically partitioned data[C]. Eighth Acm Sigkdd International Conference on Knowledge Discovery Data Mining, 2002,

[5] J. H. M. KAMBER. 数据挖掘概念与技术 (原书第 2 版)(计算机科学丛书)[M]. , 2008

[6] Y. He, H. Tan, W. Luo, et al. Mr-dbscan: an efficient parallel density-based clustering algorithm using mapreduce[C]. 2011 IEEE 17th International Conference on Parallel and Distributed Systems, 2011, 473-480

[7] E. Januzaj, H. P. Kriegel, M. Pfeifle. Scalable density-based distributed clustering[C]. European Conference on Principles of Data Mining and Knowledge Discovery, 2004,

[8] S. Kantabutra, A. L. Couch. Parallel k-means clustering algorithm on nows[J]. NECTEC Technical journal, 2000, 1(6): 243-247

[9] 郑苗苗, 吉根林, ZhengMiaomiao, et al. Dk-means——分布式聚类算法 k-dmeans 的改进 [J]. 计算机研究与发展, 2007, 44(z2): 84-88

[10] 李锁花, 孙志挥, 周晓云. 基于特征向量的分布式聚类算法 [J]. 计算机应用, , 2): 125-128

[11] S. Merugu, J. Ghosh. Privacy-preserving distributed clustering using generative models[C]. Third IEEE International Conference on Data Mining, 2003, 211-218

[12] 张国荣, 印鉴. 分布式环境下保持隐私的聚类挖掘算法 [J]. 计算机工程与应用, 2007, 18): 165-167

[13] 杨林, 张霞萍, 白治国, et al. 基于 smc 协议的分布式聚类分析隐私保护的研究 [J]. 计算机工程与设计, , 21): 24-26

[14] 姚瑶, 吉根林. 一种基于隐私保护的分布式聚类算法 [J]. 计算机科学, , 03): 106-108+111

[15] M. Klusch, S. Lodi, G. Moro. Distributed clustering based on sampling local density estimates.[C]. , 2003,

[16] S. J. Patel, D. Punjani, D. C. Jinwala. An efficient approach for privacy preserving distributed clustering in semi-honest model using elliptic curve cryptography[J]. International Journal of Network Security, 2015, 17(3): 328-339

[17] Z. Zhang, K. Huang, T. Tan. Comparison of similarity measures for trajectory clustering in outdoor surveillance scenes[C]. 18th International Conference on Pattern Recognition (ICPR'06), 2006, 1135-1138

[18] J. Chen, R. Wang, L. Liu, et al. Clustering of trajectories based on hausdorff distance[C]. 2011 International Conference on Electronics, Communications and Control (ICECC), 2011, 1940-1944

[19] C. Rick. Efficient computation of all longest common subsequences[C]. Scandinavian Workshop on Algorithm Theory, 2000, 407-418

[20] S. Gaffney, P. Smyth. Trajectory clustering with mixtures of regression models[C]. Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, 1999, 63-72

[21] D. Chudova, S. Gaffney, E. Mjolsness, et al. Translation-invariant mixture models for curve clustering[C]. Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, 2003, 79-88

[22] J. Alon, S. Sclaroff, G. Kollios, et al. Discovering clusters in motion time-series data[C]. 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings., 2003, I-I

[23] M. Nanni, D. Pedreschi. Time-focused clustering of trajectories of moving objects[J]. Journal of Intelligent Information Systems, 2006, 27(3): 267-289

[24] M. Veloso, S. Phithakkitnukoon, C. Bento. Urban mobility study using taxi traces[C]. Proceedings of the 2011 international workshop on Trajectory data mining and analysis, 2011, 23-30

[25] A. T. Palma, V. Bogorny, B. Kuijpers, et al. A clustering-based approach for discovering interesting places in trajectories.[C]. , 2008,

[26] H. Jeung, M. L. Yiu, X. Zhou, et al. Discovery of convoys in trajectory databases[J]. Proceedings of the VLDB Endowment, 2008, 1(1): 1068-1080

[27] Chih-Chieh, Hung, Wen-Chih, et al. Clustering and aggregating clues of trajectories for mining trajectory patterns and routes[J]. , ,

[28] J.-G. Lee, J. Han, K.-Y. Whang. Trajectory clustering: a partition-and-group framework[C]. Proceedings of the 2007 ACM SIGMOD international conference on Management of data, 2007, 593-604

[29] M. Buchin, A. Driemel, M. Van Kreveld, et al. Segmenting trajectories: A framework and algorithms using spatiotemporal criteria[J]. Journal of Spatial Information Science, 2011, 2011(3): 33-63

[30] Q. Fan, D. Zhang, H. Wu, et al. A general and parallel platform for mining co-movement patterns over large-scale trajectories[J]. Proceedings of the Vldb Endowment, , 10(4): 313-324

[31] 肖源. 分布式车辆时空轨迹异常检测算法研究 [D]. , ,

[32] J. Wang, Z. Zhang, Z. Chu, et al. A trajectory data density partition based distributed parallel clustering method[J]. Journal of University of Science Technology of China, 2018,

[33] J. Mao, Q. Song, C. Jin, et al. Tscluwin: Trajectory stream clustering over sliding window[C]. , 2016,

[34] Z. Deng, Y. Hu, M. Zhu, et al. A scalable and fast optics for clustering trajectory big data[J]. Cluster Computing, 2015, 18(2): 549-562

[35] J. A. Hartigan, M. A. Wong. Algorithm as 136: A k-means clustering algorithm[J]. Journal of the Royal Statistical Society. Series C (Applied Statistics), 1979, 28(1): 100-108

[36] A., P., Dempster, et al. Maximum likelihood from incomplete data via the em algorithm[J]. , ,

[37] R. M. Neal, G. E. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants[M]. Springer, 1998, 355-368

[38] L. K. P. J. RDUSSEEUN. Clustering by means of medoids[J]. , 1987,

[39] 周志华. 机器学习 [M]. Qing hua da xue chu ban she, 2016

[40] A. Buluç, J. T. Fineman, M. Frigo, et al. Parallel sparse matrix-vector and matrix-transpose-vector multiplication using compressed sparse blocks[C]. Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures, 2009, 233-244

[41] J. R. Norris, J. R. Norris. Markov chains[M]. Cambridge university press, 1998

[42] W. M. Rand. Objective criteria for the evaluation of clustering methods[J]. Journal of the American Statistical association, 1971, 66(336): 846-850

[43] 李航. 统计学习方法 [M]. Qing hua da xue chu ban she, 2012

附录 A k-means 算法收敛性证明

k-means 算法流程第二章已提到，见1，但 k-means 算法是否收敛从算法力流程中无从得知，这里给出 k-means 算法收敛的数学证明供参考。

由于 k-means 算法是由 EM 算法推导的，故这里首先介绍 EM 算法。EM 算法是针对包含隐变量的问题，令 X 表示观测变量集，Y 表示隐变量集，Θ 表示模型参数变量集，联合分布记为 $P(X,Y|\theta)$，条件分布记为 $P(Y|X,\theta)$，则 EM 算法流程可以通过以下步骤表示：

1. 选择参数的初始值 $\theta^{(0)}$；
2. E 步：记 $\theta^{(i)}$ 为第 i 次迭代参数 θ 参数的估计值，在第 i+1 次迭代的 E 步，计算

$$\begin{aligned} Q\left(\theta,\theta^{(i)}\right) &= E_Y\left[\log P\left(X,Y|\theta\right)|X,\theta^{(i)}\right] \\ &= \sum_Y \log P\left(X,Y|\theta\right) P\left(Y|X,\theta^{(i)}\right) \end{aligned}$$

   其中，$P\left(Y|X,\theta^{(i)}\right)$ 是在给定观测数据 X 和当前的参数估计 $\theta^{(i)}$ 下隐变量数据 Y 的条件概率分布；
3. M 步：求使 $Q\left(\theta,\theta^{(i)}\right)$ 极大化的 θ，确定第 i+1 次迭代的参数的估计值 $\theta^{(i+1)}$

$$\theta^{(i+1)} = \arg\max_\theta Q\left(\theta,\theta^{(i)}\right)$$

4. 重复第（2）步和第（3）步直到收敛。

在 k-means 算法中，观测变量 X 则为数据属性，隐变量 Y 则为数据所属的簇序号，在此基础上定义：

$$P\left(X|y,\theta\right) \propto \exp\left\{-\left(x-u_y\right)^2\right\}$$
$$P\left(Y|x,\theta\right) = \{p_1,p_2,...,p_y,...,p_k\}$$

其中 u_y 表示序号为 y 的簇心，$p_y = 1$ 若 x 距离 u_y 最近，则 $p_i = 0, i = 1,2,...,y-1,y+1,...,k$。

通过以上定义，经过 EM 算法的 E 步，数据集中的所有元素将被归入离自己最近的簇心向量对应的簇，M 步针对单个簇中的所有元素，可转换成下式：

$$\begin{aligned}\theta^{(i+1)} &= arg\ \max_{\theta} Q\left(\theta, \theta^{(i)}\right) \\ &\Leftrightarrow arg\ \max_{\theta} LL\left(X|y, \theta\right) \\ &\propto arg\ \max_{\theta}\ \ln\left(\prod_{i=1}^{n} \exp\left(-|x_i - u_y|^2\right)\right) \\ &\propto arg\ \max_{\theta}\ \sum_{i=1}^{n} -|x_i - u_y|^2\end{aligned}$$

对上式等号右边求导可得：

$$u_y = \frac{\sum_{i=1}^{n} x_i}{n}$$

即，由 M 步得到的结果是，各个簇心更新为簇中所有元素向量的均值，此结论与 k-means 算法流程等同。

以上介绍了 k-means 算法在 EM 算法层面的解释，要想知道 k-means 算法的收敛性，只要证明 EM 算法的收敛性即可，下面则对 EM 算法收敛性进行说明。关于 EM 算法收敛性可以由两个定理得到。

定理一：设 $P(X|\theta)$ 为观测数据的似然函数，$\theta^{(i)}$ 为 EM 算法得到的参数估计序列，$P(X|\theta^{(i)})$ 为桂英的似然函数序列，则 $P(X|\theta^{(i)})$ 是单调递增的，即 $P\left(X|\theta^{(i+1)}\right) \geqslant P\left(X|\theta^{(i)}\right)$。

定理二：设 $L(\theta) = lnP(X|\theta)$ 为观测数据的对数似然函数，$\theta^{(i)}$ 为 EM 算法得到的参数估计序列，$L(\theta^{(i)})$ 为对应的对数似然函数序列。则（1）如果 $P(X|\theta)$ 由上界，则 $L(\theta^{(i)}) = lnP(X|\theta^{(i)}$ 收敛到某一值 L；（2）在函数 $Q(\theta, \theta^{(i)})$ 与 $L(\theta)$ 满足一定条件下，由 EM 算法得到的参数估计序列 $\theta^{(i)}$ 的收敛值 θ^* 是 $L(\theta)$ 的稳定点。

其中定理二关于函数 $Q(\theta, \theta^{(i)})$ 与 $L(\theta)$ 的条件在大多情况下都是满足的。关于定理的证明可以参考书籍 [43] 和论文 [37]，这里不再给出证明过程。

有上述两个定理可得，EM 算法在大多数情况都能证明过程是收敛的，故 k-means 算法也有着同样的收敛性。

攻读专业硕士学位期间取得的成果

[1] [1] 2018 年 10 月，学业三等奖学金

[2] [2] 2019 年 10 月，学业二等奖学金