

分类号 TP957

学号 0123456

U D C

密级 公开

## 专业学位硕士学位论文

# 云环境下隐私保护聚类方法关键技术研究

硕士生姓名 邓晏湘

专业学位类别 电子信息

专业学位领域 计算机技术

指导教师 付绍静 教授

协助指导教师 柳林 副教授

国防科技大学研究生院

二〇二三年四月

# Research on Key Technologies of Privacy-preserving Clustering Method in Cloud Environment

Candidate: **Deng Yanxiang**

Supervisor: **Prof. Fu ShaoJing**

Associate Supervisor: **A.P. Liu Lin**

A thesis

Submitted in partial fulfillment of the requirements  
for the professional degree of **Master of Engineering**  
in **Computer Technology**

Graduate School of National University of Defense Technology

Changsha, Hunan, P. R. China

April, 2023

# 独 创 性 声 明

本人声明所呈交的学位论文是我本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表和撰写过的研究成果，也不包含为获得国防科技大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文题目：\_\_\_\_\_ 云环境下隐私保护聚类方法关键技术研究 \_\_\_\_\_

学位论文作者签名：\_\_\_\_\_ 日期： 年 月 日

# 学 位 论 文 版 权 使 用 授 权 书

本人完全了解国防科技大学有关保留、使用学位论文的规定。本人授权国防科技大学可以保留并向国家有关部门或机构送交论文的复印件和电子文档，允许论文被查阅和借阅；可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

(保密学位论文在解密后适用本授权书。)

学位论文题目：\_\_\_\_\_ 云环境下隐私保护聚类方法关键技术研究 \_\_\_\_\_

学位论文作者签名：\_\_\_\_\_ 日期： 年 月 日

作者指导教师签名：\_\_\_\_\_ 日期： 年 月 日

# 目 录

摘要	.....	i
Abstract	.....	iii
第一章 绪论	.....	1
1.1 研究背景与意义	.....	1
1.2 云环境下聚类中隐私保护问题概述	.....	4
1.2.1 云环境下聚类相关研究概述	.....	5
1.2.2 云环境下聚类中的隐私问题	.....	7
1.2.3 隐私保护聚类技术手段	.....	8
1.2.4 隐私保护聚类方案设计目标	.....	12
1.3 研究内容和创新点	.....	12
1.3.1 隐私保护外包 K-means 聚类方法研究	.....	13
1.3.2 隐私保护 DBSCAN 系列方案研究	.....	13
1.4 论文的组织结构	.....	14
第二章 相关研究综述	.....	16
2.1 隐私保护 K-means 聚类方案	.....	16
2.1.1 基于同态加密的 K-means 聚类方案	.....	16
2.1.2 基于多方计算的 K-means 聚类方案	.....	17
2.2 隐私保护 DBSCAN 聚类方案	.....	18
2.3 本章小节	.....	19
第三章 隐私保护外包 K-means 聚类方法研究	.....	20
3.1 引言	.....	20
3.2 预备知识	.....	21
3.2.1 半诚实模型	.....	21
3.2.2 基于 Kd-tree 的 K-means 聚类方法	.....	21
3.2.3 基于秘密共享的安全多方计算	.....	23
3.3 问题描述	.....	24
3.3.1 系统与安全模型	.....	24
3.3.2 设计目标	.....	25
3.4 基于秘密共享的隐私保护计算模块	.....	25
3.4.1 安全欧式距离计算协议	.....	25

---

3.4.2 安全比较协议 .....	26
3.4.3 安全极值计算协议 .....	27
3.4.4 安全过滤协议 .....	29
3.5 基于 Kd-tree 的隐私保护 K-means 方案 .....	30
3.5.1 算法详述 .....	31
3.5.2 讨论 .....	31
3.6 安全性分析 .....	31
3.7 系统评估与性能分析 .....	33
3.7.1 理论分析 .....	33
3.7.2 实验分析 .....	34
3.8 本章小结 .....	37
<b>第四章 隐私保护密度聚类方法研究 .....</b>	<b>39</b>
4.1 引言 .....	39
4.2 预备知识 .....	40
4.2.1 调整兰德系数 .....	40
4.2.2 DBSCAN .....	41
4.2.3 改进的 DBSCAN .....	43
4.2.4 基于 DBSCAN 的层次聚类 .....	44
4.3 问题描述 .....	45
4.3.1 系统模型 .....	46
4.3.2 安全模型 .....	46
4.3.3 设计目标 .....	46
4.3.4 系统输入 .....	47
4.4 基于秘密共享的隐私保护计算模块 .....	47
4.4.1 安全排序协议 .....	47
4.5 隐私保护 DBSCAN 方案 .....	48
4.5.1 预处理元素 .....	51
4.5.2 密度聚类 .....	52
4.5.3 还原聚类结果 .....	53
4.6 改进的隐私保护 DBSCAN .....	55
4.6.1 初始化 .....	56
4.6.2 核心对象聚类 .....	57
4.7 基于 DBSCAN 的隐私保护层次聚类 .....	57
4.7.1 获取 $\epsilon$ 值 .....	58

---

4.7.2 计算初始簇 .....	58
4.7.3 合并初始簇 .....	61
4.8 实验分析 .....	61
4.8.1 理论分析 .....	61
4.8.2 实验设置 .....	62
4.8.3 聚类质量 .....	63
4.8.4 运行开销与分析 .....	65
4.9 本章小结 .....	67
<b>第五章 总结与展望 .....</b>	<b>69</b>
5.1 工作总结 .....	69
5.2 研究展望 .....	69
<b>致谢 .....</b>	<b>71</b>
<b>参考文献 .....</b>	<b>72</b>
<b>作者在学期间取得的学术成果 .....</b>	<b>79</b>

## 表 目 录

表 1.1	常见聚类算法比较	6
表 1.2	同态加密方案对比	10
表 1.3	隐私保护技术对比	12
表 2.1	完全安全的隐私保护 K-means 和 DBSCAN 方案	19
表 3.1	计算开销	33
表 3.2	数据集详细信息	34
表 3.3	通信与计算开销对比 ( $n = 8192, m = 5, k = 3$ )	35
表 4.1	数据集具体信息	40
表 4.2	ARI 指标计算中间值	41
表 4.3	方案复杂度理论分析	62
表 4.4	数据集具体信息	63
表 4.5	聚类质量评估	64
表 4.6	耗时对比	66

## 图 目 录

图 1.1 云计算架构 .....	1
图 1.2 云计算服务交互模式 .....	2
图 1.3 外包计算 .....	3
图 1.4 多方计算 .....	3
图 1.5 DBSCAN 与 K-means 聚类结果对比 .....	7
图 1.6 同态加密发展时间线 .....	9
图 1.7 论文组织结构 .....	15
图 3.1 系统模型 .....	24
图 3.2 小数据集上的安全极值协议 (SMin (S)) .....	28
图 3.3 大数据集上的安全极值协议 (SMin(L)) .....	29
图 3.4 明文 K-means 聚类结果 .....	36
图 3.5 隐私保护 K-means 聚类结果 .....	36
图 3.6 $k = 3, T = 20$ .....	37
图 3.7 $m = 5, T = 20$ .....	37
图 4.1 DBSCAN 中核心定义图示 .....	42
图 4.2 测试数据集 .....	43
图 4.3 DBSCAN 聚类结果 .....	43
图 4.4 数据打乱后, DBSCAN 聚类结果 .....	43
图 4.5 改进的 DBSCAN 算法划分结果 .....	43
图 4.6 隐私保护 DBSCAN 方案系统模型 .....	46
图 4.7 数据可视化 .....	48
图 4.8 核心对象与边界对象 .....	48
图 4.9 聚类处理过程 .....	49
图 4.10 相连关系 .....	49
图 4.11 还原聚类结果 .....	50
图 4.12 初始数据集 .....	55
图 4.13 处理边界对象 .....	55
图 4.14 Lsun、S1 和 Aggregation 数据集正确划分结果 .....	63
图 4.15 聚类结果可视化比较 .....	65
图 4.16 耗时对比 .....	66
图 4.17 通信开销对比 .....	66
图 4.18 聚类与还原结果耗时比较 .....	67

## 摘要

聚类作为一种无监督机器学习算法，广泛用于数据挖掘与特征提取等研究领域，影响着人们生活的方方面面。随着数据量的迅速增加，面向大数据的聚类算法对金融行业中的股票投资分析，商业环境中的市场营销分析以及医疗领域的影像分析都具有重要应用价值。然而，一方面，随着移动手机，穿戴设备以及各种智能设备的广泛使用，用于聚类分析的数据量呈现爆炸性增长的趋势；另一方面，当下许多应用场景要求联合多方数据进行聚类以提升数据分析的质量，传统聚类形式难以满足新场景的需求。

云计算以开放的标准和服务为基础，以互联网为中心，提供安全、快捷、便利的数据存储和网络计算服务，具有成本低、效率高、灵活和可扩展等优点。越来越多公司、机构和独立用户选择将数据存储到云平台上，运行聚类算法获取数据分析的结果。在数据挖掘分析技术的发展过程中，基于海量丰富的数据样本和可靠多样的聚类算法，云计算以其强大的计算和存储资源，灵活的服务方式，为大数据上的聚类应用发展提供了坚实的基础。然而，数据安全问题始终是阻碍云计算进一步推广和应用的主要因素之一，将未经处理的数据上传到云平台，即失去了对数据的有效控制，数据可能在存储和处理的过程中被拦截、篡改或传播，带来巨大的数据泄露风险。为了解决上述问题，通常采取数据加密后上传的方式来保护数据的安全性，但是加密后的数据在聚类过程中的可用性大大降低。因此，如何在云计算环境下大数据上进行安全高效的聚类，成为我们面临的重大挑战，也受到学术界以及工业界的广泛关注和研究。

本文深入分析云计算中聚类应用所面临的安全问题，基于数据挖掘中广泛应用的 K-means 聚类和密度聚类算法开展隐私保护问题的关键技术研究。

本文的主要研究内容和贡献包含以下几个方面：

1. 针对外包计算中隐私保护 K-means 聚类方案中存在的数据安全和效率问题，本文提出了一种基于 Kd-tree 的隐私保护外包 K-means 聚类方案。首先，用户在本地数据的基础上构造 Kd-tree，然后通过秘密共享的方式上传至云平台，两个云服务器通过执行系列高效的安全计算协议获取最终结果。一方面，方案借助 Kd-tree 数据结构加速聚类划分的过程，减少冗余计算，提升了算法运行效率；另一方面，基础安全计算协议可以迁移到基于秘密共享的隐私计算场景中，具备良好的扩展性。理论分析以及实验验证均表明，本方案不仅提供了完备的数据安全保障，同时也实现了高效精准的外包 K-means

聚类。

2. 针对密度聚类（Density-Based Spatial Clustering Application with Noise, DBSCAN）设计了三种隐私保护方案。首先，基于 DBSCAN 设计了全新的隐私保护计算协议，引入临时簇的概念，通过记录相连信息来还原聚类结果，在提升安全性的前提下，运行效率相较于前沿方案提升近 100 倍。其次，为解决 DBSCAN 聚类结果不稳定的问题，设计了一个能够获取稳定划分结果的改进隐私保护协议，在牺牲一定性能的情况下，显著提升了聚类结果的质量。最后，针对重要参数依赖数据分布与人工设定的问题，给出了一种基于 DBSCAN 的层次聚类方法，借助  $k$  线图和 knn 算法，获取关键参数，能够在包含不同密度数据的数据集上较好的进行聚类。经过全面的实验与理论分析，证明方案能够在保护原始数据、中间结果以及聚类结果安全的同时，高效的完成聚类。

关键词：云计算；隐私保护聚类；秘密共享；安全多方计算

## Abstract

As an unsupervised machine learning algorithm, clustering is widely used in research fields such as data mining and feature extraction, affecting all aspects of people's lives. With the rapid increase of data volume, the clustering algorithm for big data has significant application value for stock investment analysis in the financial industry, marketing analysis in the business environment, and image analysis in the medical field. However, on the one hand, with the widespread use of mobile phones, wearable devices, and various smart facilities, the amount of data used for cluster analysis shows an explosive growth trend; On the other hand, many application scenarios require clustering of multi-party data to improve the quality of data analysis, and traditional clustering forms are difficult to meet the needs of new scenarios.

Cloud computing is based on open standards and services, centered on the Internet, and provides secure, fast, and convenient data storage and network computing services, with the advantages of low cost, high efficiency, flexibility, and scalability. More and more companies, institutions, and independent users choose to store data on cloud platforms and run clustering algorithms to obtain data analysis results. In the development process of data mining and analysis technology, based on massive and rich data samples and reliable and diverse clustering algorithms, cloud computing provides a solid foundation for the development of clustering applications on big data with its powerful computing and storage resources and flexible service methods. However, data security issues have always been one of the main factors hindering the further promotion and application of cloud computing, uploading unprocessed data to the cloud platform, that is, losing effective control over the data, data may be intercepted, tampered with or spread in the process of storage and processing, bringing huge data leakage risks. In order to solve the above problems, the security of data is usually protected by uploading data after encryption, but the availability of encrypted data in the clustering process is greatly reduced. Therefore, how to carry out safe and efficient clustering of big data in the cloud computing environment has become a major challenge for us, and has also received extensive attention and research from academia and industry.

This paper deeply analyzes the security problems faced by clustering applications in cloud computing and conducts research on key technologies of privacy protection based

on K-means clustering and density clustering algorithms widely used in data mining. The main research content and contributions of this paper include the following aspects:

1. Aiming at the data security and efficiency problems in the privacy protection K-means clustering scheme in outsourced computing, this paper proposes a Kd-tree-based privacy protection outsourcing K-means clustering scheme. First, users construct Kd-trees based on their local data, and then upload them to the cloud platform through secret sharing, and the two cloud servers obtain the final result by executing a series of efficient secure computing protocols. On the one hand, the scheme accelerates the process of clustering and division with the help of the Kd-tree data structure, reduces redundant calculation, and improves the operational efficiency of the algorithm. On the other hand, the basic secure computing protocol can be migrated to the privacy computing scenario based on secret sharing, which has good scalability. Theoretical analysis and experimental verification show that this scheme not only provides complete data security but also realizes efficient and accurate outsourcing K-means clustering.

2. Three privacy protection schemes are designed for Density-Based Spatial Clustering Applications with Noise (DBSCAN). Firstly, a new privacy-preserving computing protocol is designed based on DBSCAN, the concept of temporary cluster is introduced, and the clustering results are restored by recording the connected information, and the operation efficiency is nearly 100 times higher than that of the cutting-edge scheme under the premise of improving security. Secondly, to solve the problem of unstable DBSCAN clustering results, an improved privacy protection protocol is designed that can obtain stable partitioning results, which significantly improves the quality of clustering results at the expense of certain performance. Finally, aiming at the problem of important parameters dependent on data distribution and manual setting, a hierarchical clustering method based on DBSCAN is proposed, and key parameters are obtained with the help of  $k$  line plot and knn algorithm, which can be better clustered on datasets containing data of different densities. After comprehensive experimental and theoretical analysis, it is proved that the scheme can efficiently complete clustering while protecting the security of original data, intermediate results, and clustering results.

**Key Words:** Cloud Computing; Privacy-Preserving Clustering; Secret Sharing; Secure Multiparty Computation

## 符号使用说明

$x_i = \{x_{i1}, \dots, x_{im}\}$	维度为 $m$ 的数据点 $x_i$
$C = \{c_1, \dots, c_m\}$	Kd-tree 中每个点的中心
$Z = \{z_1, \dots, z_k\}$	$k$ 个候选簇
$\langle x \rangle / \langle x \rangle^A$	$x$ 在 $\mathbb{Z}_n$ 上的加性秘密共享值
$\langle x \rangle^B$	$x$ 在 $\mathbb{Z}_2$ 上的布尔共享值
MUL	安全乘法
SED	带缩放因子的安全欧式距离计算协议
SC	安全比较协议
SMin(S/L)	安全极值计算协议（大数据集 / 小数据集）
SF	安全过滤协议
SSORT	安全排序协议
DIST	安全欧式距离计算

# 第一章 绪论

云计算在赋能海量数据上的聚类应用的同时，也带来了数据泄露的风险，引起人们的广泛关注。如何在提供便捷的云计算服务的同时，保护用户的数据安全成为了当今学术研究的热门问题，极具现实意义。本章首先介绍云环境下的聚类应用的相关背景，然后阐述设计隐私保护聚类方案的重要意义，最后总结文章主要工作内容及创新点，并给出了整体组织结构。

## 1.1 研究背景与意义

随着现代社会数字化的不断演进，数据使用量呈指数级增加，个人和小型组织越来越难以在内部计算机服务器上维护所有重要信息、运行大型程序和系统。为解决这些问题，云计算应运而生。自云计算在 2006 年被提出后，它就被认为是能够推动下一代互联网革命的技术，并迅速成为了研究领域的热门探索方向<sup>[1]</sup>。云计算既可以指在网络上提供的应用服务，也可以指数据中心中提供这些服务的硬件和系统软件，它的核心思想是，将众多用网络连接的计算资源统一管理和调度，构成一个计算资源中心向用户提供按需服务，它的运行原理和基于 Web 的电子邮件客户端类似，允许用户访问系统的所有功能和文件，但是无需将系统的大部分内容保存在自己的计算机上。此外，云计算在各个领域都有广泛且深刻的应用（如图1.1），例如在智能教育领域，为学生提供在线课程和为老师提供授课工具；在智慧医疗领域，为医生和患者提供远程诊疗、电子病例和医疗数据分析等功能；在商业领域，为企业组织提供电子商务、市场分析以及资源规划等功能。

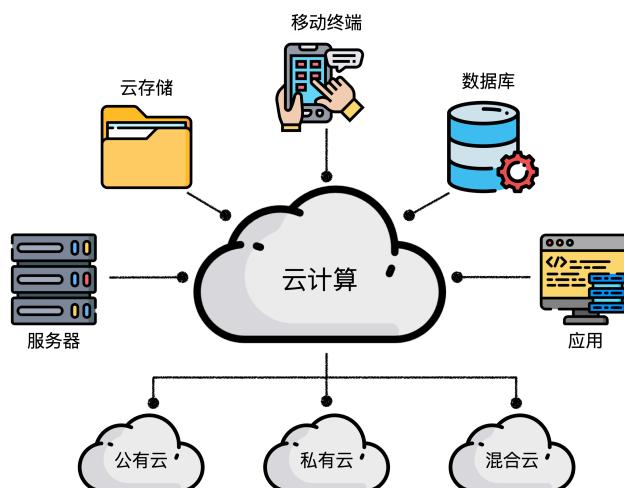


图 1.1 云计算架构

云计算具有经济实惠、按需服务、方便、通用、多租户、灵活、稳定等特点，它主要提供了三种服务交付模式，如图1.2所示，分别是基础设施即服务（IaaS）、平台即服务（PaaS）、以及软件即服务（SaaS）。IaaS将计算机硬件（例如网络存储，虚拟机，数据中心，处理器和内存）视为一种服务，无需大量资金和时间即可提供可扩展基础架构。同时，IaaS还可以用于构建防火墙，虚拟机监控和其他安全领域<sup>[2]</sup>。PaaS以开发工具、框架、架构、程序和集成开发环境的形式提供服务。它的优势在于用户可以在云端获取各种开发工具，中间件以及数据库等资源，无需自己购买相关软硬件，关注开发的逻辑和功能即可。SaaS是远程计算服务的集合，它使得第三方供应商能够远程部署应用程序。云计算客户可以在云基础设施上通过网络获取云服务提供商的应用程序<sup>[3]</sup>。

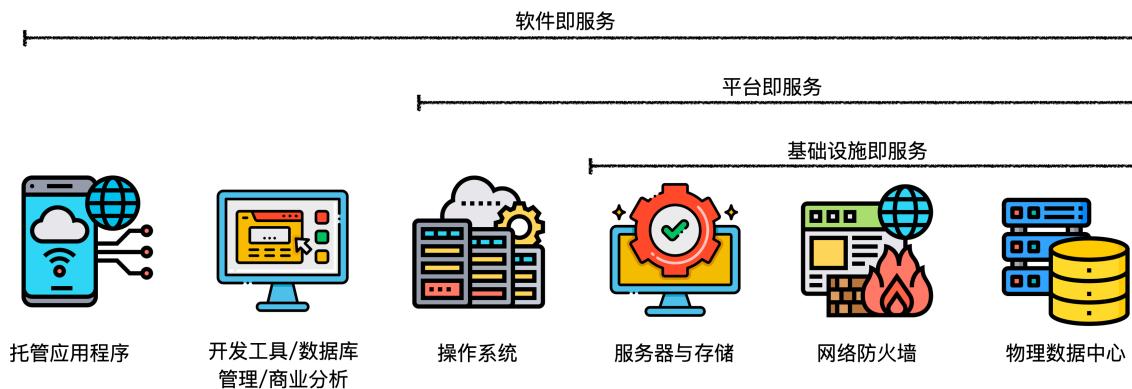


图 1.2 云计算服务交互模式

近年来，云计算场景下出现了一种新的服务类型，机器学习即服务（Machine Learning as a Service, MLaaS）<sup>[4]</sup>。随着研究的发展和设备的进步，机器学习从学术研究落地到生活的方方面面，但是大多数机器学习任务对于设备的性能要求较高，需要存储海量的数据才能取得较好的结果。大型公司有能力承担设备费用，利用机器学习的便利开展各种各样的研究与服务，但是资源有限的小公司和个人的需求常常难以被满足。

MLaaS基本上是一组基于云的工具的总称，这些工具以一种全新的方式支持科学家和数据工作者的日常工作，它们支持协作、版本控制、并行化和其他比较复杂的任务。此外，较大的云服务供应商提供了简单明了的方法来将他们的 MLaaS 服务与其他工具集成，方便用户进行自动化部署以及执行机器学习任务。MLaaS 包含很多种类，例如自然语言处理、图像和视频分析、计算机视觉、语音识别以及数据挖掘等等。随着机器学习技术的发展和进步，提供 MLaaS 的公司数量也在增加，主流的公司包括微软的 Azure、谷歌的 Google Cloud ML 等等。在 MLaaS 中，海量的数据需要被上传到云计算中心，这一过程也被称之为外包计算。由于

用户在数据上传后失去了对数据的完全控制，因此会更加关心数据隐私问题。云计算服务模型的复杂性、实时性、数据的多元异构以及终端资源有限等特点使得传统的数据隐私保护方法无法直接用来保护云计算中的大量数据<sup>[5]</sup>。

在 2018 年，脸书被爆出隐私泄露丑闻，近 5000 万选民的个人资料被泄露，据称被某政治咨询公司所使用。这一丑闻的发展不仅使得该公司的市值减少数十亿美元，同时还引发了公众的强烈不满和质疑，由此可见保护数据的隐私安全无论是对公司还是对用户都有深远意义。将一些敏感的数据进行外包以获取机器学习的结果可能会引发隐私泄漏问题，特别是在金融以及医疗领域。以医疗影像识别为例，在对数据不进行任何处理的前提下交付给云服务器进行训练，会直接造成用户私密医疗数据的泄漏，引发公众恐慌。即便是对用户的 data 进行了简单的脱敏，模型训练的中间结果也可能被恶意利用以获取信息。以 K-means 聚类为例，虽然原始 data 经过加密，但是中间结果，例如聚类簇的大小，能够直接揭露具有某种特征的群体有多少人，以及 data 的分布特点，特别有研究表明可以通过一些手段从中间结果恢复原始 data<sup>[6]</sup>。因此，如何能够进行安全的外包计算，在维护用户 data 安全的同时，正确的获取外包机器学习任务的结果成为一个研究热点。

聚类（Clustering）是一种非常流行的无监督机器学习技术，它能够将相似的输入元素划分到同一个簇（cluster）中。聚类的应用领域非常广泛，从业务分析到医疗保健等诸多领域。在许多这些应用场景中，敏感信息在被正确聚类的同时，也不应该被泄漏。此外，现在经常需要将不同来源的数据组合起来进行训练以提升分析质量，庞大的 data 量对资源有限的用户带来巨大的压力，因此，通常需要将复杂的计算外包给强大的云服务器进行处理，这就要求设计有效的外包隐私保护聚类方案<sup>[7]</sup>。目前，为了保护聚类过程中输入的敏感 data 的安全性，已经有了许多研究成果，涵盖各个方面。在设计隐私保护聚类协议时，通常基于两个不同的场景，即多方计算（图 1.4）与外包计算（图 1.3）。

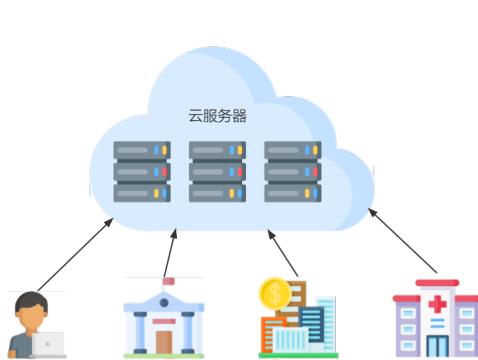


图 1.3 外包计算

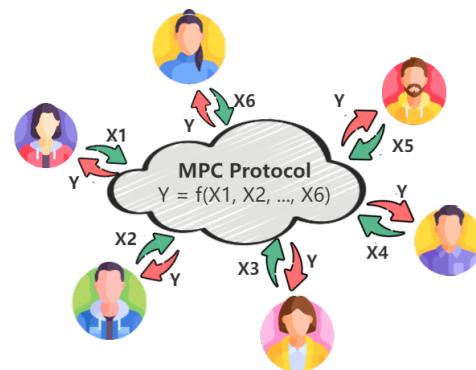


图 1.4 多方计算

在多方计算的场景下<sup>[8]</sup>，两个及两个以上的数据所有者共同执行安全计算协议，除了输出之外的任何内容都不会泄漏给彼此，比较常见的是两方计算。同时，一些研究也会在模型设计中引入半诚实参与方（通常是服务器）来辅助计算。相对应的，另一种场景则是外包计算<sup>[9]</sup>，一个或多个数据所有者将计算（或存储）外包给其他参与方，并假设这些参与方可以为数据所有者进行聚类而无需知道关于输入数据的任何信息。由于外包计算旨在使用外部资源，数据所有者通常不应该参与协议的执行，并处于离线状态，但是这一点通常难以完全实现。值得一提的是，一些多方计算协议（MPC）也可以用于外包计算场景，只需要数据所有者在多个不共谋的参与方之间秘密共享自己的数据，然后多个不共谋的参与方在秘密共享数据上执行安全协议实现聚类。但是，MPC 协议是否支持外包计算很大程度上取决于中间协议的设计，如果数据持有者需要在聚类过程中进行大量的明文计算或者在中间对数据进行解密，那就很难将协议转换到外包计算的场景中。

外包计算和多方计算各有其优势和缺点，对于资源有限的用户来说，外包计算更加具有实际意义和价值，能够有效的减轻用户的资源负担。基于外包计算的安全计算方法已有诸多研究，但是完全安全的聚类方案通常效率较低，即便是在性能较好的服务器上也需要花费难以接受的时间才能获得最终结果，效率较高的方案通常会牺牲一定的安全性，例如泄漏簇大小、簇中心这样的中间计算结果，带来一些隐私安全问题。因此如何设计出安全又高效的外包隐私保护聚类方案值得深入研究和探索。

综上所述，本文以云计算下的外包聚类为主要应用场景，深入分析其面临的基本问题和技术难点，以设计兼具效率与安全的方案为主要目的，针对典型的聚类算法（K-means, DBSCAN）展开研究，提出了完善的隐私保护方案，并通过理论分析和真实数据集测试，验证所述方案的安全性，高效性和正确性。通过本文的研究，期望能够为资源有限的用户提供一种可行的隐私保护外包聚类方法，为云服务提供商与独立用户提供合作的桥梁，使得用户能够专注于数据的挖掘分析，云服务提供商进一步提高资源利用率，各司其职，物尽其用，让更多人无需顾虑数据安全更加放心的享受科技进步带来的生活水平的提升。

## 1.2 云环境下聚类中隐私保护问题概述

本节首先给出云环境下聚类研究综述，详细介绍了云计算、聚类算法以及云环境中聚类算法的应用，然后介绍云环境下聚类应用中存在的数据安全问题。基于上述内容，总结归纳了隐私保护聚类研究中常用的技术手段，并探讨了隐私保护聚类方案设计的目标。

### 1.2.1 云环境下聚类相关研究概述

#### 1.2.1.1 云计算概述

云计算是指一种基于互联网的按需提供计算机系统资源，特别是数据存储（云存储）和计算能力，并且无需用户自行采购、配置或管理资源的计算模式<sup>[10]</sup>，其允许用户通过互联网使用分布在世界各地的服务器上的资源。云计算的部署模型通常可以分为三种，分别是公有云、私有云和混合云。公有云由第三方云服务提供商运营，使用户可以按照特定要求和业务目标安排资源部署。私有云由单个组织构建管理，以非公开的方式托管在本地，具有更强的数据控制、安全和管理功能。混合云是上述二者的结合体，使得用户既能利用公有云服务，同时保持私有云架构中常见的安全和合规功能。

云计算具有灵活性、成本低、高可靠、以及可共享等优点，带给用户巨大的便利，资源有限的用户无需关心如何部署、管理和维护 IT 基础设施，只需要付费即可获取想要的资源和服务。这项技术解放了用户的双手，使他们能够专注于本领域的研究与工作，提升效率。

#### 1.2.1.2 聚类算法概述

聚类是一种无监督机器学习算法，它能够将一组数据划分到不同的集合中，即簇。簇内部数据相似度较高，簇与簇之间数据相似度相对较低。聚类在实际的应用中非常广泛，常见于数据挖掘分析、生物信息处理、文本挖掘提取以及图像处理等领域。通过聚类能够找到不同数据集合特有的模式与结构，挖掘隐藏在背后的信息，从而进行分析和进一步处理。

常见的聚类算法包括：

- **基于划分的聚类方法：**也叫基于分区的聚类或基于距离的聚类，核心思想为假定数据集有  $n$  个样本，在满足样本间距离的前提下，最少将其划分为  $k$  个簇。簇内数据尽可能相似，不同簇的数据尽可能不相似。常见的基于划分的聚类方法包括 K-means 算法和 K-medoids 算法。
- **基于层次的聚类方法：**将数据对象划分为层次结构，可以采取自顶向下或者自底向上的顺序进行操作。自顶向下也称为分裂，将所有样本当作一个簇，找到最远的两个簇进行分割，直到满足预期条件。自底向上则是将每个点都看成一个独立的簇，找到距离最近的两个簇进行合并，迭代重复直到满足预期。常见的聚类算法包括 AGNES、DIANA 等。
- **基于密度的聚类方法：**根据样本的密度分布进行聚类，通过样本密度反映数据之间的可连接性，并通过可连接性不断扩展从而产生最终的聚类结果。算法认为密度较高的区域中数据对象属于同一个簇，密度低（包含较少或不包

含数据)的区域则形成了簇的边界。该方法能够发现任意形状和大小的簇,对噪声不敏感。常见的算法有 DBSCAN、OPTICS 等。

- **基于网格 (Grid) 的聚类方法:** 将数据空间划分为若干网格单元, 并将数据对象映射到网格中, 计算每个单元的密度, 通过预设阈值来判断网格是否为高密度单元, 邻近的稠密单元进行合并构成了簇。该方法可以大大减小聚类计算复杂度, 但是对网格的划分方式要求较高。典型的算法包括 STING、CLIQUE 等。
- **基于模型的聚类方法:** 该方法的核心思想为假设每个簇都符合某种概率模型, 例如正态分布, 然后借助最大似然估计法或者贝叶斯推断来估计模型参数并划分数据到所属簇中。该方法能够得到明确的聚类结果, 但是对模型和参数的选择要求较高。常见的算法有 EM、GMM 等。

综上所述, 聚类算法的划分一般可以基于划分、密度、网格和模型等方式。不同的具体聚类算法各有优缺点和适用场景。下面在表格1.1中, 给出几个常见的聚类算法, 以及它们在应用数据的规模、对噪声的抗干扰能力、适合的数据形状以及算法效率方面的特点。

表 1.1 常见聚类算法比较

算法类型	可用于大规模	对噪声抗干扰性	适合形状	算法效率
K-means	是	较差	球形	很高
DBSCAN	是	较好	任意形状	一般
BIRCH	否	较差	球形	很高
CURE	是	很好	任意形状	较高

本文主要基于 K-means 和 DBSCAN 聚类算法展开隐私保护方案相关研究。二者在对噪声的抗干扰性、适合的数据集形状以及算法效率方面存在一些区别。图1.5中可以看到 K-means 和 DBSCAN 在不同数据集上聚类划分结果的区别, 可以看到 DBSCAN 对于任意形状的数据集都适应的较好, 而 K-means 则更加适合球形数据集。

### 1.2.1.3 云环境下的聚类应用

云环境下聚类算法的实施通常依赖于外包计算这一计算模式, 即将计算任务和数据交给云服务提供商进行处理。外包计算的流程主要包括数据准备、任务分配、计算处理以及结果返回。它具有节省计算资源、提升计算效率、降低成本等优点。从用户角度来看, 外包聚类计算的应用主要可以细分为两种, 分别是单用户上传数据聚类, 多用户分别上传数据联合聚类。

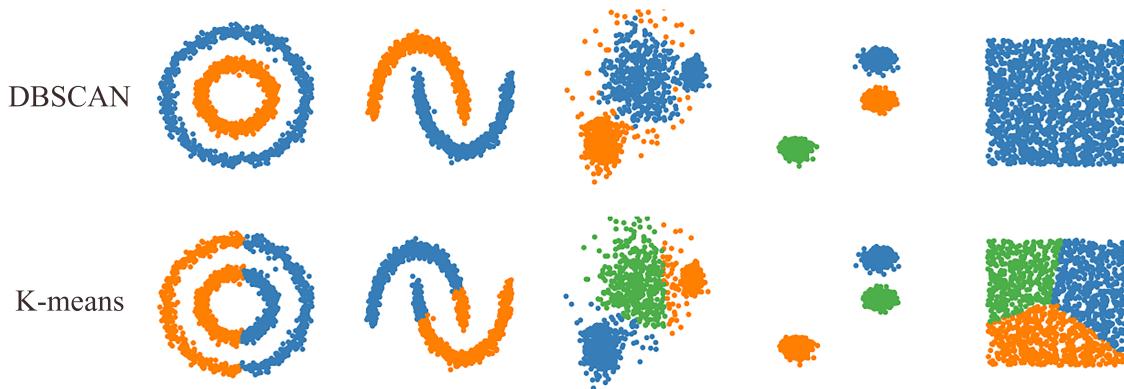


图 1.5 DBSCAN 与 K-means 聚类结果对比

单用户外包聚类，例如高校的学生老师或是小型的研究开发团队，通常希望能够对指定的数据集进行聚类，获取结果用于后续挖掘分析。这些用户拥有的计算和通信资源通常有限，难以在本地机器上对大型数据集进行处理，因此会选择将数据上传至云平台后，以外包计算这一形式获取结果。

多用户协同外包聚类，例如银行或医院等机构，因自身业务的开展，会产生极具研究价值的数据，深入挖掘分析这些数据，可以为医疗影像研究和用户信用评估等方面的研究提供指导。然而，独立机构产生的数据量有限，海量数据上的聚类通常效果更好。因此这些机构倾向于联合其他机构，将数据上传到云平台上共同聚类。以医院为例，医疗影像常用于各种疾病的辅助诊疗，通过对这些数据进行挖掘分析能够提取出各种特征用于研究，造福人类。

综上，聚类的数据来源和形式有所区别，但算法收敛后聚类结果以相同的方式通过互联网传递回用户手中以进行后续的研究分析。

### 1.2.2 云环境下聚类中的隐私问题

本小节主要就云环境下外包聚类计算中存在的隐私问题展开讨论，并总结了几种现有的典型隐私保护技术。如前所述，在外包计算中，用户需要将数据全部上传至云平台，云服务器利用这些数据进行聚类训练，并回传结果。

若外包计算模式下，云平台接收的数据并非源自网络公开数据集，而是来自独立用户或者组织机构上传。在这种情况下，数据本身通常具有较高的隐私保护要求。例如，医疗机构希望借助云计算强大的计算能力和海量的存储空间来对其医疗影像信息展开挖掘分析，从而在诊疗中辅助医生进行判断和诊断，其外包的数据通常包含患者的敏感信息，例如年龄、性别、籍贯、医疗影像以及相关医学检查结果。智能穿戴设备需要获取用户的各种私密数据，如心率、步数、运动状态以及睡眠状态等内容进行分析然后将结果反馈给用户。一旦发生数据泄露，会

造成巨大损失。此外，即便是对原始数据进行了加密操作，云平台在这些密文数据上进行聚类获取结果的同时，还会获取若干中间计算结果，这些数据的安全性也要纳入考虑。因此，对于云环境下隐私保护方案中数据安全的研究，需要考虑原始数据，中间结果以及划分结果等多个方面。

### 1.2.3 隐私保护聚类技术手段

云环境下隐私保护聚类方案的一大难点在于，如何在保证原始数据以及中间结果隐私性的前提下，完成安全高效的聚类运算。当下，聚类研究中常见的隐私保护技术手段大体上可以分为数据扰动、差分隐私、同态加密以及安全多方计算。本节将分别介绍上述几种技术的基本原理以及各自的优缺点。

#### 1.2.3.1 数据扰动

数据扰动（Data Perturbation）是一种隐私保护方法，允许在不泄漏原始数据信息的情况下，对数据进行一定的扰动，从而达到保护隐私的目的。数据扰动方式主要可以分为两种类型：

- **概率分布方式**：是数据扰动技术中一种重要手段，通过调整数据的概率分布来保护敏感信息。通常从相同的分布样本中或者从分布本身中获取数据进行替换。常见的实现的方式有拉普拉斯机制、高斯机制以及指数机制。但是若数据分布比较复杂，则难以用简单的概率分布来进行扰动。同时噪声的大小和分布需要根据具体的应用场景和应用特征来进行选择。
- **数值失真方式**：借助加法、乘法或其他随机过程扰乱数据。其中，添加加性随机噪声，使得攻击者无法获取单个数据的原始信息，但是由于没有影响数据的分布特征，因此仍能够通过分析挖掘出数据相关有效信息。乘性扰动则是指利用数据投影等方法，将原始数据映射到低维空间，例如可以使用随机映射技术将数据映射到一个随机选择的子空间中。

综上所述，数据扰动技术实施较简单，引入开销较小，适用场景广泛，但是存在着安全性较低、影响数据质量等问题。具体使用需要权衡数据隐私性和数据准确性，进行取舍。

#### 1.2.3.2 差分隐私

Dwork 等人<sup>[11]</sup>在 2006 年提出一种全新的隐私定义，即差分隐私。它能够确保在数据集合中添加或者删除某项内容，不会显著影响任何相关数据分析的结果<sup>[12]</sup>，将隐私泄露风险控制在可接受范围内，恶意攻击者无法通过观察运算结果来获取精确的单个数据信息。目前已有较多关于差分隐私的研究和应用，实际场景中通常采取本地化差分隐私技术保护隐私安全，苹果的 iOS 系统<sup>[13]</sup>、谷歌的 Chrome 浏览器<sup>[14]</sup>以及微软的 Windows 系统<sup>[15]</sup>均已应用。差分隐私的具体定义<sup>[16]</sup>如下：

**定义 1.1:** 设有随机算法  $\Gamma$ ,  $\Gamma$  所有可能输出构成的集合为  $O_\Gamma$ 。对于任意两个邻近数据集  $D$  和  $D'$  以及  $O_\Gamma$  的任何子集合  $S_\Gamma$ , 若算法  $\Gamma$  满足

$$\Pr [M(D) \in S_M] \leq \exp(\varepsilon) \times \Pr [M(D') \in S_M] \quad (1.1)$$

则称算法  $\Gamma$  能够提供  $\epsilon$ -差分隐私保护。

其中参数  $\epsilon$  称为隐私保护预算, 它用来控制算法  $\Gamma$  在两个相邻数据集上取得相同输出的概率比值, 反映了  $\Gamma$  提供的隐私保护水平, 当  $\epsilon$  越小时, 隐私保护的力度越大, 需要的噪声越多,  $\epsilon$  等于 0 时, 隐私保护水平最高。

通过在原始数据上添加噪声, 既保护了隐私又保留了原始数据的统计特征。以数据库查询为例, 利用差分隐私技术对数据库进行处理, 实现了数据库中具体某个记录发生变化但不影响数据发布的结果, 同时攻击者无法通过观察数据库发布的结果来推测用户的某条记录是否在数据库中, 进而实现隐私保护。

使用差分隐私技术能够有效避免数据泄露, 保护用户的隐私, 通过控制添加噪声的多少, 可以在数据的隐私保护与查询结果的准确性之间取得一定的平衡。此外, 差分隐私对于数据的保护能力不受具体数据特征的影响, 能够用于各种类型的数据和场景。然而差分隐私技术存在一定的局限性, 在处理较小数据集时, 添加噪声的影响变大, 可能会导致数据质量下降, 进而影响数据挖掘分析的结果。引入差分隐私, 会带来计算和时间成本, 降低系统的效率。

综上, 差分隐私既有隐私保护能力, 又有其受限制的地方, 需要结合实际应用场景的特点来选择性使用。

### 1.2.3.3 同态加密

同态加密 (Homomorphic Encryption) 是由 Rivest, Adleman 和 Dertouzos<sup>[17]</sup> 于 1978 年首次提出, Craig Gentry<sup>[18]</sup> 于 2009 年首次实现的一种加密方法。在 Gentry 提出一个可行的全同态加密方案之前, 同态加密的发展时间线<sup>[19]</sup> 如下图 1.6 所示。

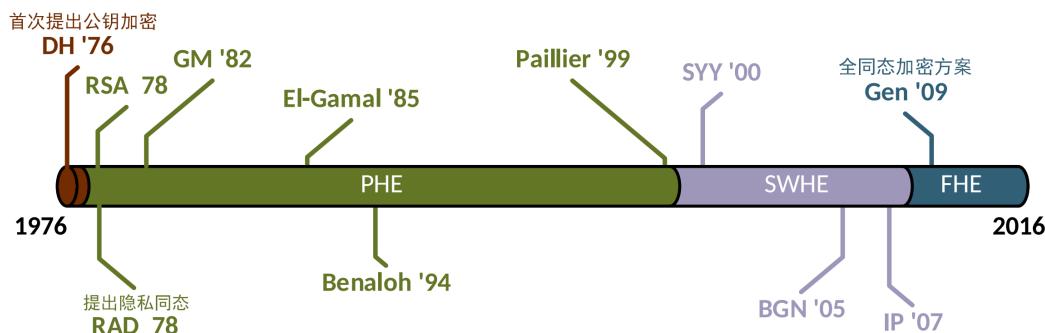


图 1.6 同态加密发展时间线

和经典的加密方式不同的是，同态加密允许直接在加密数据上进行计算，而不需要访问密钥。计算的结果仍然处于加密状态，并且可以通过密钥解密还原明文结果，目前同态加密相关研究已经取得了诸多成果。具体定义<sup>[19]</sup>如下：

**定义 1.2：**若一个加密方案支持如下公式，则称为操作  $\star$  上的同态

$$E(m_1) \star E(m_2) = E(m_1 \star m_2), \quad \forall m_1, m_2 \in M \quad (1.2)$$

其中  $E$  为加密算法， $M$  为所有可能信息的集合。

同态加密可以进一步细分为三类，分别是部分同态加密（Partially Homomorphic Encryption, PHE）、类同态加密（Somewhat Homomorphic Encryption, SWHE）以及全同态加密（Fully Homomorphic Encryption, FHE）。三者的区别如下表1.2所示：

表 1.2 同态加密方案对比

同态加密类型	支持的操作类型	操作次数	常见加密方案
PHE	加法或乘法	无限次	RSA、Goldwasser-Micali、El-Gamal 以及 Paillier 等
SWHE	加法和乘法	有限次	BGN
FHE	加法和乘法	无限次	GH11

同态加密目前已有诸多应用，例如加密搜索，即在同态加密的基础上使搜索引擎不知道用户搜索真正内容的前提下获取搜索结果。安全求交，也称为隐私集合求交，该技术允许多个参与方在不公开各自数据的前提下，协同查找出交集数据，且不泄露交集数据以外的隐私信息。多方联合建模能够在不泄露任何隐私的前提下，结合多方数据以提高模型效果，例如医疗机构协同诊断、银行联合反欺诈等等。同态加密具有诸多优点，在不受信任的环境下（例如云环境或者第三方）数据仍能保证安全和隐私，消除了数据可用性与数据隐私之间的权衡问题，无需为了数据的隐私而放弃任何数据特征，以及能够抵御量子攻击。

同态加密经过数十年的发展，从最初的半同态加密方案到现在提出的全同态加密方案，已经取得了长足进展，能够满足不同应用场景和复杂计算需求，但是由于该加密方案计算开销较大，效率较低，仍然难以广泛应用于实际生产环境中，性能仍有很大的优化空间。

#### 1.2.3.4 安全多方计算

安全多方计算（Secure Multi-Party Computation, SMPC）允许多个参与方之间协同进行安全的计算操作，而不泄露私有数据。每个参与方都只能看到自己输入的内容，无法获知其他参与方的输入。安全多方计算与外包计算的不同之处在于，协议的执行者同时也是数据的拥有者<sup>[20]</sup>。

安全多方计算已经有诸多隐私保护应用，例如姚式百万富翁问题<sup>[21]</sup>、安全拍卖、投票、安全机器学习<sup>[22]</sup>等。其中主流技术包括不经意传输（Oblivious Transfer, OT）、混淆电路（Garbled Circuits, GC）以及秘密共享（Secret Sharing, SS）。具体介绍如下：

- **不经意传输：**是安全计算协议中一个非常重要的基础模块。标准的 1-out-of-2 OT 的定义涉及两方，发送方  $S$  持有两个秘密  $x_0, x_1$ ，接收方  $R$  持有一个选择比特  $b \in \{0, 1\}$ 。OT 允许  $R$  获得  $x_b$ ，同时不知道  $x_{1-b}$  的内容， $S$  无法得知  $R$  获得了什么内容。针对 OT 计算成本较高等特点，已有多种研究提出了改进方案。
- **混淆电路：**是最广为人知的多方计算技术。有许多协议和方案都是基于混淆电路设计，适用于各种计算操作，包括简单的加法乘法，复杂的比较排序等。但是混淆电路计算成本相对较高，需要每个参与方进行大量通信，因此通信复杂度较高，同时设计协议比较复杂。目前许多研究工作从降低通信开销<sup>[23]</sup>、减少电路执行时间<sup>[24]</sup> 以及减少电路门数<sup>[25]</sup> 来优化电路的使用。
- **秘密共享：**是一种将原始信息划分为  $n$  个部分，分配给不同的参与方，只有满足  $t$  个参与方合作才能恢复出原始秘密信息的重要密码学技术。基于秘密共享的复杂安全协议，通常需要参与方之间进行频繁通信，带来较大通信开销，限制了方案的可扩展性。为了能够扩展到多个参与方的场景，一些研究工作借助大数据计算框架进行并行计算以提升效率减少通信开销。

自安全多方计算被提出后，已经取得了长足进展，部署多方计算的开销下降了 3-9 个数量级。但是除了上述进展，在多方计算被大规模用于隐私保护应用之前还需要解决几个主要挑战。

- **开销：**多方计算协议的开销跨度非常大，从几乎可忽略到不能接受，主要取决于设定的威胁模型以及实现的功能。通常数据中心内部的带宽费用非常低，但是实际情况下，通常会将计算外包给不同的云服务提供商以防止数据被窃取，因此带宽开销不可忽视。
- **安全与效率权衡：**目前有一些研究选择在安全和效率之间折中。通过在核心协议中放弃一些强安全性保证来获取效率的极大提升。但是牺牲安全性保证会带来多大的数据安全问题仍有待研究。

在表格1.3中，总结归纳了几种不同的隐私保护技术的特点。随着研究的深入，越来越多学者提出了结合多种不同隐私保护技术的方案，扬长避短，并取得了不错的成果<sup>[26~29]</sup>。

表 1.3 隐私保护技术对比

技术名称	特点	优点	缺点	应用场景
差分隐私	添加噪声扰动数据	隐私性好，效率高	影响数据质量，可用性不高	算力较弱，对数据精度要求较低的场景
同态加密	允许在密文上进行计算	保护数据安全	计算开销大，复杂计算效率低	算力较强，隐私保护要求高
安全多方计算	数据持有方协同进行隐私计算	保护数据安全，计算结果准确	通信开销大，效率较低	分布式场景

#### 1.2.4 隐私保护聚类方案设计目标

云环境下设计隐私保护聚类系统要实现的目标主要有以下三点：

- **准确性：**用户使用秘密共享、同态加密等技术处理数据后上传到云平台进行聚类，虽然保障了数据的隐私安全，但是对聚类方案的设计带来了挑战。准确性要求在密文的基础上设计系列协议实现聚类的功能，最后得到和明文一致的准确结果。
- **安全性：**输入数据、计算中间结果、输出结果均与数据安全息息相关。一旦某一个环节出问题，都可能会导致隐私被泄露，带来巨大损失。因此隐私保护聚类方案要保护数据以及所有相关结果的安全性。外包计算场景中，执行协议的云服务器无法窥视数据。联合数据进行训练获取结果的每个用户无法获取其他用户的私密数据。
- **高效性：**随着互联网快速发展，用于聚类的数据量日益增加。在明文上进行聚类的耗时和开销已不可忽视，隐私保护方案通常计算和通信更加复杂，因此会进一步降低效率。除了要设计云平台可以高效执行的隐私保护方案，还要照顾到资源有限的用户，尽量降低用户的计算通信开销。

上述三个目标中，安全性与高效性通常难以兼顾，通常需要设计复杂的计算协议来实现完全安全的方案。一些隐私保护工具会损失数据的准确性，但是带来效率的提升，例如差分隐私技术。在设计方案时需要综合考虑，结合应用场景和实际需求，在三者之间做出取舍，达到平衡。

### 1.3 研究内容和创新点

云环境下的隐私保护聚类系统，需要兼顾效率与安全。本文针对当前研究中存在的数据泄露、聚类效率低等问题，面向具体的应用场景，基于聚类算法，隐私保护技术构建方案。本文主要研究内容以及创新点如下：

### 1.3.1 隐私保护外包 K-means 聚类方法研究

K-means 是机器学习算法中最为常见和流行的一种无监督的聚类算法，常用于数据挖掘分析和特征提取。因其包含的计算简单，便于实现，基于 K-means 的隐私保护聚类方案研究较多，种类丰富。然而现有的隐私保护 K-means 研究中，通常难以兼顾效率与安全，以近几年的前沿研究为例，高效的方案泄露了数据分布<sup>[30]</sup>，完全安全的方案则效率过低<sup>[31]</sup>。本文在论文<sup>[32]</sup>的基础上，研究了一种利用 Kd-tree 数据结构提升效率同时保障数据安全的隐私保护外包 K-means 聚类方案。

在本文的方案中，用户首先在本地基于明文数据构建 Kd-tree，通过秘密共享加密数据结构后分别上传至双云服务器，而后云服务器协同执行一系列安全协议，最终获得密文聚类结果。方案的创新点主要包括：（1）基于秘密共享技术和百万富翁协议<sup>[33]</sup> 构建了一系列安全计算协议，使得云平台可以在密文上完成比较、计算欧式距离、求解极值等运算；（2）与传统 K-means 算法不同，本文利用 Kd-tree 数据结构进行计算加速，提出一种全新的隐私保护方案；（3）与现有工作对比，本文的计算与通信开销均较小。

### 1.3.2 隐私保护 DBSCAN 系列方案研究

K-means 聚类过程简单，因此相关的隐私保护方案研究也非常丰富，但是 K-means 对数据集以及参数初始化有较高要求，需要人工评估簇个数  $k$ ，同时在非球形数据集上聚类表现较差，对异常数据比较敏感，适用场景有限。此外，本文提出的隐私保护外包 K-means 方案的应用场景为某一个用户持有所有原始数据，不支持多用户共同上传数据训练。为了解决上述问题，本文基于另一种聚类算法，即 DBSCAN，展开研究，针对不同的应用场景和实际需求提出了系列隐私保护 DBSCAN 方案。DBSCAN 算法具有诸多优点，应用场景广泛，包含计算比较简单，对该算法进行隐私保护实现具有较大现实意义。

#### 1.3.2.1 隐私保护 DBSCAN 方案

在传统明文 DBSCAN 方案<sup>[34]</sup> 的基础上，已有诸多隐私保护方案的研究<sup>[35~36]</sup>，但是方案大多效率较低且开销较大，同时没有详细的实验验证和分析。本文结合密度聚类算法的特点对聚类过程进行重新设计，构建了一个高效的隐私保护 DBSCAN 方案。

协同聚类的用户各自将数据进行秘密共享后，分别发送给不共谋的双云服务器，两个云平台协同运行安全协议，并将计算结果返回给用户。用户恢复明文后，运行简单的还原算法来获取最终的聚类结果。本文提出的隐私保护 DBSCAN 方案的创新点主要有：（1）提出一种全新的隐私保护 DBSCAN 的实现方式，构建临时

簇来降低计算轮次，将部分简单的结果还原工作移交给用户提升效率；（2）允许多个独立的参与方上传数据协同聚类，在不知道彼此数据的前提下获取最终聚类结果；（3）极大程度上提升了效率，耗时减少近 100 倍。

### 1.3.2.2 改进的隐私保护 DBSCAN 方案

明文 DBSCAN 方案存在聚类结果不稳定的问题，若数据同时满足被划分到多个簇的条件，则最终的划分结果取决于数据遍历的顺序，打乱顺序后再次排序可能得到不同的结果。因此本文在论文<sup>[37]</sup>的基础上，提出了一种改进的隐私保护 DBSCAN 方案。

方案的实现流程可以分为初始化，聚类以及还原结果三步。主要的创新点有：（1）针对结果稳定性问题首次提出隐私保护方案；（2）在优化聚类效果的基础上，效率远高于目前的前沿方案<sup>[29]</sup>，耗时缩短近 20 倍。

### 1.3.2.3 基于 DBSCAN 的隐私保护层次聚类

DBSCAN 聚类的效果受到关键参数的影响，而参数的设置需要分析数据分布特点，在多用户各自持有数据的场景下难以找到合适的方法确定参数值。同时，对于包含不同密度的数据集，传统 DBSCAN 方案难以获取合理的聚类结果，可能出现较小簇或者单个点的情况。为了解决上述问题，本文基于论文<sup>[38]</sup>的思想，借助  $k$  线图和  $k$  近邻算法，提出了一个基于 DBSCAN 的隐私保护层次聚类方案。

方案分为获取  $\text{eps}$  值、计算初始簇、合并初始簇还原结果三个步骤。用户上传加密数据到双云服务器后，云平台通过执行系列给定的安全协议，获取最终结果并发送给用户。本方案的创新点主要有：（1）基于秘密共享设计了安全排序协议；（2）无需人工给出关键参数，减少用户计算开销；（3）适应包含不同密度数据的数据集，应用更加广泛。

## 1.4 论文的组织结构

本文主要分为 5 个章节，组织结构如图1.7所示。

第一章是绪论，首先介绍了云计算和聚类的基本概念，通过剖析云环境下聚类的发展应用，引出了研究云环境下聚类所面临的数据安全和效率问题的意义。然后概述了设计隐私保护聚类方案可能利用的密码学技术及其优缺点，并给出了隐私保护聚类系统的设计目标。最后，简要介绍了本文的主要工作与贡献。

第二章是相关研究综述。本章调研了近年来隐私保护 K-means 方案以及隐私保护 DBSCAN 方案的相关研究，通过横向以及纵向对比，总结归纳了不同方案的优缺点。

第三章是隐私保护外包 K-means 聚类方案。现有的隐私保护 K-means 聚类方案大多无法兼顾效率与数据安全。为了在保证安全的基础上提升效率，减少用户

的计算和通信开销，本文借助 Kd-tree 数据结构，基于秘密共享设计了一系列安全协议，实现了安全高效的聚类方案。通过安全性分析、理论分析以及全面的实验评估，验证了方案的有效性和安全性，以及高效率。

第四章是隐私保护密度聚类方法研究。针对 K-means 聚类中存在的各种问题，提出了系列基于 DBSCAN 的隐私保护聚类方案，分别解决了 K-means 不适合非凸样本簇、传统 DBSCAN 聚类结果不稳定以及传统 DBSCAN 不适合多密度数据集的问题。通过系列分析，验证了方案的正确性、安全性和高效性。

第五章是总结与展望。本章首先总结了本文的研究工作，然后基于此对未来工作进行了展望。

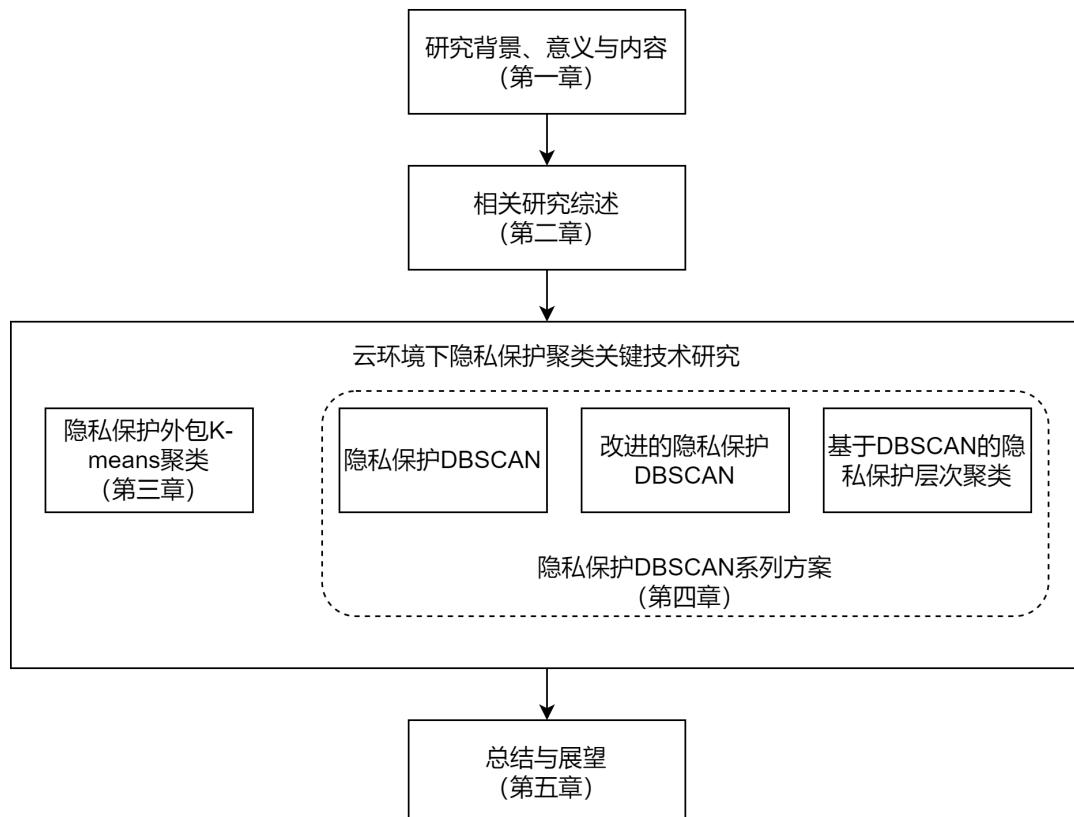


图 1.7 论文组织结构

## 第二章 相关研究综述

当前，已经有大量关于云环境下隐私保护聚类方案的相关研究，并取得了长足进展。结合本文的主要研究内容，本章总结归纳了国内外关于隐私保护 K-means 聚类方案和隐私保护 DBSCAN 聚类方案的研究成果，深入分析方案各自在安全性、高效性以及可应用性方面存在的优势和缺点。

### 2.1 隐私保护 K-means 聚类方案

近年来，在众多相关研究中使用的密码学工具主要可以分为两类，分别是同态加密和多方计算。本节将围绕基于同态加密的隐私保护 K-means 聚类方案以及基于多方计算的隐私保护 K-means 方案展开讨论。

#### 2.1.1 基于同态加密的 K-means 聚类方案

Liu 等人<sup>[39]</sup> 在全同态加密的基础上提出了一种外包聚类方案，由于密文不会保留距离的顺序特征，因此方案比较加密距离时需要用户提供陷门信息（Trapdoor Infomation），给用户带来巨大的开销。为了减少用户在聚类过程中的开销，Almutairi 等人<sup>[40]</sup> 提出一种更新距离矩阵（UDM）的办法来提升比较密文距离的效率。但是上述两种方案都向云服务器泄漏了部分信息，例如簇大小、数据之间的距离以及簇中心的值。此外，上述两种方案中使用的全同态加密方法在论文<sup>[41]</sup> 中证实不完全安全。外包计算中多用户上传数据存在单密钥限制以及低效频繁的客户端 - 服务器交流的问题，不利于大规模实际应用。为了解决上述问题，论文<sup>[42]</sup> 在 Spark 框架下提出基于双解密的公钥系统（PKCDD）提出了系列安全协议和外包聚类方案，但是该方案泄漏了中间结果，存在安全隐患。基于部分同态加密的协议通常涉及频繁的交互带来巨大的通信和计算开销，效率较低。对此，Wu 等人<sup>[30]</sup> 基于结合密文打包技术的全同态加密方案提出了安全高效的外包 K-means 聚类方案（SEOKC），通过对打包的密文进行并行计算，极大提升了方案的效率。然而协议中采用相同的噪声混淆数据后解密，泄漏了数据的分布信息存在安全性问题。

上述方案均没有满足语义安全性，为了解决这一问题，Rao 等人<sup>[43]</sup> 在 Paillier 同态加密方案的基础上提出了一个外包 K-means 聚类方案，方案允许在多用户场景下进行聚类，用户一旦上传数据后无需进行任何操作。在保护数据安全的同时，方案向两个不共谋的云服务器隐藏了数据访问模式。然而，在密文上执行大量交互协议引入了巨大的计算开销，使得方案无法在大型数据集上应用。Kim 等人<sup>[44]</sup> 认为方案<sup>[43]</sup> 之所以低效是因为采取了基于比特数组的比较和随机选择初始簇中心

的方法，为了解决这个问题，他们提供了一种全新的能够在密文上进行快速比较的方案并且根据数据分布来选簇中心，该方案计算效率相较于论文<sup>[43]</sup>提升3倍。为了解决全同态中计算除法的问题，论文<sup>[31]</sup>提出了一种自然编码方式，但是该方式效率非常低，因此作者对K-means方案进行修改避开除法以适应全同态方案的特点，最后为了进一步提升效率，作者以精确性为代价提出了一种近似的隐私保护K-means方案。

同态加密以其允许在密文上进行计算的优点受到广泛的关注和应用，但是基于同态加密的方案通常计算开销大以至于效率较低，难以应用于实际聚类场景中。关于全同态加密方案的研究仍在不断发展，随着研究的深入与提升，基于同态加密的隐私保护方案也会进一步优化。

### 2.1.2 基于多方计算的K-means聚类方案

基于多方计算的隐私保护K-means聚类方案通常使用秘密共享和混淆电路两种基础工具来构造协议。Doganay等人<sup>[45]</sup>提出了基于加性秘密共享的分布式隐私保护K-means聚类方案，相较于之前的工作效率有所提升，但是方案最后向所有参与方揭露了簇划分的信息。大部分相关研究通常基于半诚实模型，对此，Patel等人<sup>[46~47]</sup>等人分别在半诚实模型下基于Shamir秘密共享设计，在恶意模型下基于零知识证明提出了两种不同的聚类方案。Upmanyu等人<sup>[48]</sup>和Baby等人<sup>[49]</sup>基于中国剩余定理(CRT-SS)分别提出了分布式阈值秘密共享方案。然而，上述方案均泄漏了聚类相关信息，因此存在数据安全问题<sup>[50]</sup>。

此外，有许多相关研究将同态加密和多方计算结合起来设计协议，以结合二者的优势。Vaidya等人<sup>[51]</sup>提出首个基于安全多方计算的隐私保护K-means聚类方案<sup>[52]</sup>，应用于垂直划分的数据集，借助了同态加密、安全混淆<sup>[53]</sup>以及混淆电路等密码学技术，但是对每个数据都需要进行安全混淆使得效率较低。Jagannathan等人<sup>[27]</sup>提出将除法转换为乘以倒数，但是这种方式不满足准确性，无法正确实现K-means算法。举个简单的例子，在 $\mathbb{Z}_{21}$ 上用11除5，结果应该是约等于2，但是 $11 * 5^{-1} = 11 * 17 = 19$ ，同时该方案计算开销巨大<sup>[54]</sup>。Su等人<sup>[28]</sup>基于不经意多项式评估以及安全近似工具构建了相关方案，实现了安全数据标准化，但是除法操作存在问题，无法提供完整安全性。Sakuma等人<sup>[55]</sup>基于paillier加密系统，姚式混淆电路设计了一个能够用于混合划分数据集的隐私保护聚类方案，该方案可扩展能容错，但是泄漏了聚类结果中簇包含数据个数的信息。Jiang等人<sup>[56]</sup>提出了一个两方外包隐私保护K-means聚类方案，该方案需要云平台与用户进行多轮交互，同时更新的簇中心会被泄漏，存在安全性问题。

Bunn等人<sup>[54]</sup>基于同态加密和加性秘密共享设计了一系列协议，实现了完全

安全的隐私保护 K-means 聚类。解决了安全多方计算中除法的计算问题，提出了一个新的协议来随机选择 k 个初始簇中心。但是开销巨大。Mohassel 等人<sup>[57]</sup> 基于混淆电路、秘密共享和 1-out-of-n 不经意传输提出了一个兼顾安全与高效的两方隐私保护聚类方案，作者结合 K-means 算法的特点，需要计算一个固定点到多个簇中心的距离，设计了快速计算的协议提升效率。

基于安全多方计算技术的隐私保护聚类方案通常既可以用于外包计算场景，也可以迁移到数据持有者之间进行聚类的场景中。同时，秘密共享、混淆电路等技术计算开销较小效率较高，能够有效的减少聚类耗时，但是安全协议大多涉及大量的交互，带来较高的通信开销。此外，实现复杂功能的安全协议设计通常比较复杂，设计难度较大。

## 2.2 隐私保护 DBSCAN 聚类方案

过去关于隐私保护聚类的研究大多是基于 K-means 算法展开，鲜少有关于隐私保护 DBSCAN 的相关研究。这里综合过去数十年的研究成果进行总结分析。

Anikin 和 Gazimov<sup>[58]</sup> 在半诚实模型下，针对垂直划分数据集，设计了一种允许任意多个参与方加入的隐私保护 DBSCAN 协议。协议主要依赖于同态加密和加性秘密共享，加密操作的开销巨大。此外，聚类要求所有参与方都保持在线状态，因此难以应用到外包计算场景中。一个参与方负责主要的计算工作，并获取所有数据之间的明文距离信息、簇划分情况以及簇大小。作者既没有给出有说服力的协议实现也没有给出性能分析。

论文<sup>[35, 59~62]</sup> 针对垂直划分或者水平划分的数据集，而论文<sup>[61, 63]</sup> 则能够处理任意划分类型的数据集。上述方案用到了同态加密、随机混淆、部分方案引入了第三方来保护数据隐私安全。但是，所有方案均泄漏了聚类信息，例如簇大小，数据相邻关系<sup>[59~61, 63~64]</sup> 或数据之间的距离<sup>[35]</sup>。在特定的情况下，泄漏这样的信息会给原始数据带来泄露风险<sup>[60~61]</sup>。在论文<sup>[62]</sup> 中，当数据被划分到所属簇时，原始信息会被揭露。此外，上述方案均没有给出实现和实验性能分析。

Rahman 等人<sup>[64]</sup> 借助同态加密技术设计了一个外包场景下的隐私保护 DBSCAN 协议，不受信任的服务器在数据持有者的帮助下执行聚类过程。然而，该方案泄漏了簇大小和数据相邻关系信息给服务器。并且，作者没有对方案进行详细的实验分析。Bozdemir 等人<sup>[36]</sup> 在半诚实模型下基于 ABY 框架<sup>[26]</sup> 提出了一个隐私保护 DBSCAN 聚类方案，该方案既可以作为两方计算协议也可以用于外包计算场景，同时协议设计引入了并行计算提升效率。然而方案中迭代深度需要人工设定，对开销影响较大。同时，方案向服务器揭露 1 比特数据来判断是否满足迭代终止条件。Wang 等人<sup>[65]</sup> 针对数据准确性和计算开销的不同要求，提出了三种

数据预处理方案，基于同态加密设计了一种用户与云平台之间进行安全比较的协议。文章的实验部分显示提升了聚类效率，但是数据集较少，分析不够全面。

论文<sup>[66~67]</sup>中，作者借助差分隐私技术来保护用户的隐私数据安全。在这些研究中，数据持有者进行聚类，不受信任的服务器通过请求访问聚类结果。然而，该协议无法应用于数据来源于多个用户的场景。此外，每个参数维度添加到距离的噪声大小取决于具体的隐私要求，而且会不可避免的影响数据的实际意义，最终影响到聚类结果的质量。

综上所述，隐私保护 DBSCAN 方案中存在的问题主要可以分为如下几类：数据持有者高度参与聚类过程，不适合外包计算场景；基于差分隐私，牺牲准确性；不完全安全，泄漏部分信息；引入大量密码学技术，协议效率较低没有扩展性。

### 2.3 本章小节

本节主要分为两部分，第一部分归纳总结了隐私保护 K-means 聚类方案的国内外相关研究，基于同态加密和多方计算两种技术分别展开讨论。第二部分归纳总结了隐私保护 DBSCAN 的国内外相关研究，概述了最近数十年的研究发展，并探讨了研究中存在的部分问题。在表格2.1中，给出了所有完全安全的隐私保护方案以及简要的介绍信息。总体而言，当前云环境下隐私保护聚类方案的设计，需要权衡安全性、效率和准确性。

表 2.1 完全安全的隐私保护 K-means 和 DBSCAN 方案

算法	论文	隐私计算工具	场景	数据集划分
K-means	[54]	HE+ASS	两方	任意方式
	[43]	HE	双服务器外包	水平划分
	[31]	HE	单服务器外包	-
	[44]	HE	双服务器外包	-
	[57]	GC	双服务器外包或两方	水平划分
DBSCAN	[68]	GC	两方	水平划分
	[29]	GC+ASS	双服务器外包或两方	任意划分

## 第三章 隐私保护外包 K-means 聚类方法研究

### 3.1 引言

K-means 聚类是一种广泛应用的无监督机器学习算法，能够将相似的输入元素划分到同一个簇中，使不同簇之间的差异尽可能大，它的应用领域非常广泛，从业务分析，市场营销到医疗数据处理等等。随着现代社会数字化的不断演进，数据使用量呈指数级增加趋势，资源有限的独立用户与小型组织越来越难以在内部计算机系统上对大量数据进行聚类分析。同时，随着研究的深入，提升聚类算法的效果要求联合多个来源的数据进行训练，这也给传统的聚类模式带来了挑战。

云计算提供的机器学习即服务能够较好的解决上述问题，它为用户提供资源和服务来运行机器学习算法，允许多方上传数据并且价格合理方便快捷<sup>[4]</sup>。在 MLaaS 中，海量的数据需要被上传到云计算中心，这也带来了新的挑战。一方面，聚类所用数据可能包含敏感信息，直接将数据交给云平台进行聚类会带来潜在的隐私信息泄露风险；另一方面，在加密的数据上进行隐私保护聚类可能会导致耗时较长，计算与通信开销较大，使数据的可用性降低。

目前，云环境下的隐私保护外包 K-means 聚类方案已经有了较多研究成果<sup>[30~31, 42, 57]</sup>，Mohassel 等人<sup>[57]</sup> 基于混淆电路、不经意传输以及秘密共享提出了一种安全高效的 K-means 聚类方案，但是该方案通信开销巨大，要求云服务器进行多轮交互。Wu 等人<sup>[30]</sup> 基于能够进行密文打包的同态加密方案提出了一种外包 K-means 聚类方案，结合 K-means 聚类中计算的特点，对密文进行并行计算以获取最终结果，虽然该方案显著提升了隐私保护聚类方案的效率，缩短了耗时，但是中间协议泄露了数据分布特点，存在安全隐患。

针对上述研究现状，本文研究在外包计算场景下，如何在保护用户隐私数据和聚类中间结果以及保证效率可接受的同时，精确的完成 K-means 计算。首先，引入 Kd-tree 来加速聚类过程，Kd-tree 是一种空间划分数据结构，将空间上靠近的数据放在同一个节点中，广泛用于空间相关应用来加速计算。基于此，本文提出了一种基于 Kd-tree 的隐私保护 K-means 聚类方法。方案由独立用户和双云服务器参与，用户在本地明文基础上构造 Kd-tree，然后秘密共享所有数据分别发送给双云，云服务器在密文基础上运行系列安全协议来获取聚类结果后，将密文结果发送给用户，用户进行还原。

本章的组织结构如下：第3.2节介绍了安全性定义、基于 Kd-tree 的 K-means 聚类以及基于秘密共享的安全多方计算。第3.3节中描述了系统模型、安全模型以及设计目标。第3.4节中提出了一系列基于秘密共享的隐私保护计算模块。第3.5节

中提出了基于 Kd-tree 的隐私保护 K-means 聚类方案。第3.6节从理论上分析了方案的正确性和安全性。第3.7节中对方案进行了全面的实验评估。最后，在3.8节中对本章进行了总结。

## 3.2 预备知识

### 3.2.1 半诚实模型

半诚实模型（Semi-honest Model），也称为诚实且好奇的模型（Honest-but-curious Model），指的是参与方会诚实的执行所有协议，但会竭尽所能利用已有的信息获取尽可能多的内容。半诚实的参与方通常是被动的，因为他们除了通过观察执行协议的过程无法采取其他任何行动来获取隐私数据。这样的模型广泛应用于诸多研究来支持密文数据上交互协议的研究。具体定义<sup>[69]</sup>如3.1所示：

**定义 3.1：**假设参与方  $C_i$  的输入为  $x_i$ , 对于任意协议  $\pi$ ,  $C_i$  的执行镜像为  $\Pi_i(\pi)$ , 并且  $C_i$  根据协议  $\pi$  的计算输出为  $y_i$ 。若可以根据  $x_i$  和  $y_i$  模拟  $\Pi_i(\pi)$ , 且它的模拟镜像分布与实际执行镜像分布计算不可区分（Computationally Indistinguishable）, 则可以认为  $\pi$  是安全的。

### 3.2.2 基于 Kd-tree 的 K-means 聚类方法

K-means 聚类是最常用的聚类算法之一，给定簇个数  $k$ ，它能够将大小为  $n$  的数据集划分成  $k$  个内容不相交的子集。每个簇都有一个中心点，对单个数据而言，该数据点到最终所属簇中心的距离相较于其他簇更短。K-means 聚类算法有多种不同的实现方式，以使聚类更加高效便捷。论文<sup>[32]</sup>的核心思想是以 Kd-tree 为主要数据结构，设计了一种高效的过滤算法来聚类数据。

#### 3.2.2.1 Kd-tree 数据结构

Kd-tree 是一种空间划分数据结构，将空间上靠近的点划分到树的同一个节点中<sup>[70]</sup>，通常应用于加速与点空间相关的计算，例如 k 近邻算法和创建点云。

这里，本文遵循如下规则来创建 Kd-tree:

- 在  $k$  维数据集中，计算数据每个维度的方差，选择方差最大的维度  $d_{max}$  来划分数据
- 找到  $d_{max}$  维度数据的中位数  $m$  作为基准来划分数据集，得到两个子集合
- 对划分出的子集重复上述过程直到 Kd-tree 构造完成

Kd-tree 数据结构有两个主要的优势，一方面，它能够将可能属于同一个数据集的点划分到同一个树节点中。另一方面，它采取一种高效的方式来将初始空间划分为两个子空间以加速后续数据处理。

### 3.2.2.2 过滤算法

本小节详细介绍论文<sup>[32]</sup>的核心，过滤算法。给定  $n$  个点，构造的 Kd-tree 包含  $O(n)$  个节点，高度为  $O(\log n)$ 。对于 Kd-tree 中每个节点  $u$ ，计算包含的数据个数  $u.count$  以及加权质心  $u.wgtCent$ ，即包含点数据之和。中心的计算方式为  $u.wgtCent/u.count$ ，在构造 Kd-tree 的过程中可以连带计算上述内容。初始簇中心采取从数据集中随机选择的方式。

---

#### 算法 3.1 过滤算法

---

输入：Kd-tree 节点  $u$ ，候选集合  $Z$

输出：聚类后的簇

```

1:  $C \leftarrow u.cell$ 
2: if  $u$  为叶子节点 then
3:      $z^*$  为  $Z$  中距离点  $u$  最近的簇
4:      $z^*.wgtCent \leftarrow z^*.wgtCent + u.point$ 
5:      $z^*.count \leftarrow z^*.count + 1$ 
6: else
7:      $z^*$  为  $Z$  中距离  $u$  的中心点最近的簇
8:     for each  $z$  in  $Z \setminus \{z^*\}$  do
9:         if  $z.isFarther(z^*, C)$  then
10:             $Z \leftarrow Z \setminus \{z\}$ 
11:        end if
12:    end for
13:    if  $|Z| = 1$  then
14:         $z^*.wgtCent \leftarrow z^*.wgtCent + u.wgtCent$ 
15:         $z^*.count \leftarrow z^*.count + u.count$ 
16:    else
17:        Filter( $u.left, Z$ )
18:        Filter( $u.right, Z$ )
19:    end if
20: end if

```

---

对于 Kd-tree 中每个节点  $u$  维护一个候选簇集合  $Z$ ，该集合中的簇均有可能为节点中所有数据的最终所属簇。对于根节点，候选簇为随机选择的  $k$  个簇。按照算法3.1中的方式来处理每个点的候选簇集合：对于每个节点  $u$ ，令  $C$  为包含数据集合， $Z$  为候选簇集合。若  $u$  为叶子节点，则直接找到  $Z$  中最近的候选簇  $z^*$ ，然后将该叶子节点内包含的数据划分到  $z^*$  中。若为 Kd-tree 中的非叶子节点，则需要进行更复杂的处理。首先计算  $C$  的中心点，然后找到候选簇中距离中心最近的

簇  $z^* \in Z$ 。接下来，对比剩余的候选簇  $z \in Z \setminus \{z^*\}$ ，若  $C$  中所有数据均距离  $z^*$  更近，则认为  $z$  不可能为节点  $u$  中任何一个数据的最近所属簇，因此进行剪枝，即从候选簇中删除  $z$ 。若经过上述处理， $u$  仅剩一个候选簇 ( $|Z| = 1$ )，即  $z^*$ ，则认为  $z^*$  就是  $u$  中所有数据的所属簇，可以将这些数据划分进  $z^*$  中，并且将相关的  $u.wgtCent$  和  $u.count$  添加到  $z^*$  中。否则，对其子节点重复上述过程。

一旦确定了节点中所有数据的最终所属簇，即可不用再对子节点进行处理，减少了冗余计算并提升聚类效率。Kd-tree 数据结构的特点使得相似的数据点会被划分到同一个节点中，能够加速算法收敛。

### 3.2.3 基于秘密共享的安全多方计算

安全多方计算 (SMPC) 首先由<sup>[71]</sup> 提出，它能够使多个参与方在不泄露自身输入数据的前提下协同进行计算获取结果，广泛用于各种隐私保护方案中。秘密共享是安全多方计算中一个常用工具，最早由 Shamir<sup>[72]</sup> 和 Blakley<sup>[73]</sup> 于 1979 年分别提出。秘密共享的基本思路是：将秘密  $s$  划分为  $n$  份，分发给  $n$  个不同的参与方，至少需要  $t$  个不同的参与者才能够重构秘密，否则失败。接下来，本节以两个参与方  $A$  与  $B$  为例，详细介绍秘密共享相关计算。

假设所有数据的范围在环  $\mathbb{Z}_P$  上，对于  $x$  的加性秘密共享值  $\langle x \rangle$ ，有  $\langle x \rangle^A + \langle x \rangle^B = x \pmod P$ 。参与方  $A$  拥有  $\langle x \rangle^A$ ，参与方  $B$  同理。重构秘密  $x$  ( $Rec(\cdot, \cdot)$ )，其中一个参与方将分配给自己的秘密发送给另一方，然后计算  $x = \langle x \rangle^A + \langle x \rangle^B (\pmod P)$ 。接下来的说明中，为了简化表示省略  $\pmod P$  说明。

#### 3.2.3.1 加性秘密共享加法

为计算两个秘密共享值  $\langle x \rangle$  ( $\langle x \rangle^A, \langle x \rangle^B$ ) 和  $\langle y \rangle$  ( $\langle y \rangle^A, \langle y \rangle^B$ ) 之和，参与方  $A$  与  $B$  分别计算  $\langle z \rangle^A = \langle x \rangle^A + \langle y \rangle^A$  和  $\langle z \rangle^B = \langle x \rangle^B + \langle y \rangle^B$ 。对于秘密共享值  $\langle x \rangle$  与常量  $c$  的加法，参与方  $A$  在本地计算  $\langle z \rangle^A = \langle x \rangle + c$ ，参与方  $B$  计算  $\langle z \rangle^B = \langle x \rangle^B$ 。

#### 3.2.3.2 秘密共享乘法

秘密共享的安全乘法协议首先由 Beaver<sup>[74]</sup> 提出，为计算乘积  $\langle z \rangle = \langle x \rangle \cdot \langle y \rangle$ ，需要一个预计算的乘法三元组  $\langle c \rangle = \langle a \rangle \cdot \langle b \rangle$ 。其中参与方  $i$  计算  $\langle e \rangle^i = \langle x \rangle^i - \langle a \rangle^i$  和  $\langle f \rangle^i = \langle y \rangle^i - \langle b \rangle^i$ ，其中  $i \in \{A, B\}$ 。参与方均运行  $Rec(\cdot, \cdot)$  来恢复  $e$  和  $f$ 。最后，参与方  $A$  令  $\langle z \rangle^A = f \cdot \langle a \rangle^A + e \cdot \langle b \rangle^A + \langle c \rangle^A$ ，参与方  $B$  令  $\langle z \rangle^B = e \cdot f + f \cdot \langle a \rangle^B + e \cdot \langle b \rangle^B + \langle c \rangle^B$ ，即计算完成。本文用 MUL 来表示乘法操作，若输入为两个秘密共享值，则输出对应乘法结果；若为两个秘密共享数组相乘，则输出对应元素相乘的秘密共享结果数组；若输入为多个秘密共享值，则输出为计算的多个数据连乘的结果。

值得注意的是，乘法主要分为两个阶段。离线阶段中，预计算的乘法三元组每次进行乘法时都要更新，但是三元组生成的过程是离线的，可以由两方通过不

经意传输（Oblivious Transfer）生成<sup>[75]</sup>，也可以由可信第三方提供<sup>[76]</sup>。更多关于预计算乘法三元组的细节可以参考<sup>[74]</sup>。在线阶段中，两个参与方相互通信以获得最终的计算结果。

### 3.3 问题描述

#### 3.3.1 系统与安全模型

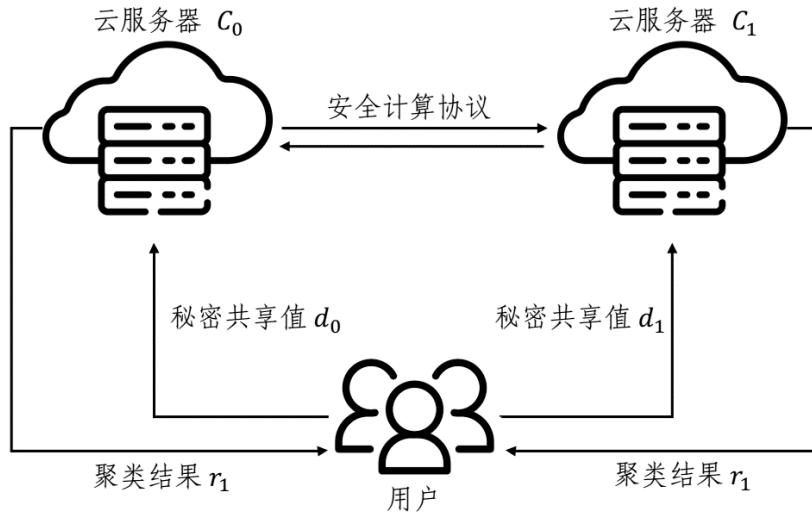


图 3.1 系统模型

如图3.1所示，本方案的系统由两个部分组成：用户和云服务器。这样的系统模型广泛用于隐私保护研究中<sup>[30, 54]</sup>。这里的客户指的是，持有隐私数据需要聚类来进行数据分析和挖掘的独立用户，他们通常没有足够的计算资源来聚类海量数据，因此需要将计算任务外包以获取结果。常见的实际用户有医疗机构、金融机构等。服务器端通常指的是拥有大量计算资源，提供付费计算服务的云服务器运营厂商例如阿里云、腾讯云等，他们提供机器学习即服务（MLaaS）这一付费模式，用户上传数据后即可离线等待机器学习计算结果。下面详细介绍方案中的每一部分：

- 双云服务器：在系统中有两个不共谋的云服务器，标识为  $C_0$  和  $C_1$ ，而且均持有用户秘密共享后的数据。他们会在此基础上执行一系列安全协议来实现隐私保护 K-means 聚类，最后将秘密共享的结果发送给用户。
- 用户：用户首先在原始数据上构造 Kd-tree，然后选择随机数来将数据划分为两份秘密共享值，分别发送给双云服务器。在 K-means 聚类结束后，获取服务器返回的秘密共享结果，运行  $Rec(\cdot, \cdot)$  来还原最终结果。

这里，本文假定  $C_0$  和  $C_1$  为半诚实模型下不共谋的两个云服务器，在实际场

景中，可以将这两个云服务器部署在不同的云服务提供商上，例如阿里云和腾讯云，为了公司各自的声誉和利益，二者可以做到保持相互独立不谋，因此这里的假定可以在实际场景中实现。同时，两个云服务器为诚实但好奇的个体，意味着它们会如实的执行执行的协议，但也会企图通过各种不谋的方式来推测感兴趣的隐私或敏感信息。

### 3.3.2 设计目标

本节所述隐私保护外包 K-means 聚类方案设计目标如下：

- **安全性：**所有外包数据例如中间计算结果、最终计算结果都不应泄露给双云服务器。同时，服务器无法从秘密共享值推断出原始数据内容以及相关数据特征，例如数据分布、距离大小关系等。
- **高效性：**方案应具备高效性，整个聚类过程主要由双云服务器进行，二者具有丰富的计算和通信资源，能够对海量数据进行处理，应当显著降低用户计算和通信开销。
- **正确性：**密文方案应不损失计算精度，例如距离计算结果不产生误差、中间协议的结果正确。双云服务器应当能够返回和明文聚类方案一致的计算结果。

## 3.4 基于秘密共享的隐私保护计算模块

为了使用户与云端能够进行安全的交互，本文基于秘密共享技术设计了一系列基本的计算模块。用户通过产生随机值将明文数据拆分为两份密文发送给双云服务器，两方在秘密共享值的基础上进行系列计算和交互，获取最终结果。

### 3.4.1 安全欧式距离计算协议

计算簇  $z$  到点  $x$  之间的欧式距离的计算公式如下：

$$dist = \sqrt{\sum_{j=1}^m (z[j] - x[j])^2} \quad (3.1)$$

其中下标  $j$  标识数据的第  $j$  个维度，所有数据均为加性秘密共享值。在所有点均被划分到对应簇后，通过计算平均值获得簇的中心点。假设簇  $z'_i$  包含点个数为  $|z'_i|$ ，包含的数据为  $x_1, \dots, x_{|z'_i|}$ ，那么簇  $z'_i$  的中心点计算方式如下：

$$z'_i[j] = \frac{x_1[j] + \dots + x_{|z'_i|}[j]}{|z'_i|} = \frac{s'_i[j]}{|z'_i|}, 1 \leq j \leq m \quad (3.2)$$

其中  $s'_i[j]$  表示簇  $z'$  中所有点第  $j$  个维度数据之和。在秘密共享值上进行除法

比较困难，因此采用文献<sup>[30]</sup>中提到的放缩方法来将距离计算中的除法转变为乘法。首先，计算全局缩放因子  $\alpha$  以及簇  $z_i$  对应的  $\alpha_i$ ，计算方式为：

$$\alpha = \prod_{j=1}^k |z_j|, \alpha_i = \prod_{j=1 \wedge i \neq j}^k |z_j| \quad (3.3)$$

为了方便计算，省略公式3.1中的根号计算，将整个公式改写为：

$$dist = \sum_{j=1}^m (\alpha x[j] - \alpha_i z_i[j])^2 \quad (3.4)$$

根据秘密共享方案的性质，可以针对公式3.4做出改进，简化计算。在一轮迭代中，无论计算什么距离，缩放因子  $\alpha$  和  $\alpha_i$  的值以及相关的计算结果都是不变的，因此可以在每轮迭代开始计算这些固定值，减少后续冗余计算。同时，参数不相关的计算可以并行进行，减少云服务器交互的次数。

### 3.4.2 安全比较协议

安全比较场景如下，参与方  $p_i$  拥有加性秘密共享值  $\langle x_i \rangle$  和  $\langle y_i \rangle$ ，其中  $i \in \{0, 1\}$ ，期望能够在不泄露  $x$  和  $y$  明文值的前提下，获取二者的大小关系  $\delta = LT(x, y)$ ， $\delta = \langle \delta \rangle_0 + \langle \delta \rangle_1$ ，其中  $\delta = LT(x, y)$  的具体含义如下：

$$LT(x, y) = \begin{cases} 1, & \text{if } x < y \\ 0, & \text{if } x \geq y \end{cases} \quad (3.5)$$

在文献<sup>[33]</sup>中，作者提出了一个高效的安全比较协议来解决百万富翁问题，该方案能够同时比较多对数据，通信开销较小，计算复杂度低。经过多轮不经意传输（Oblivious Transfer）后，参与方  $p_0$  和  $p_1$  分别获得布尔秘密共享结果  $\delta = LT(x, y)$ ， $\delta = \langle \delta \rangle_0^B \oplus \langle \delta \rangle_1^B$ 。

由于本研究中安全比较的输入与输出均为加性秘密共享值，本协议在上述百万富翁协议的基础上添加一些改进以构造安全比较协议。具体过程如算法3.2所示：

百万富翁协议解决的是双方各自持有明文在不泄露自身数据的情况下进行比较的问题，无法直接比较秘密共享值，因此这里进行如下改造，将秘密共享值之间的比较转换为明文上的比较。首先，令云服务器  $C_0$  产生随机数  $a, a \in \mathbb{Z}_n$ ，将秘密共享值  $\langle x \rangle$  与  $\langle y \rangle$  之间的比较转换为明文  $a$  与  $x - y + a$  之间的比较，若  $x > y$ ，则  $x - y + a > a$ 。上述转换只需要一轮交互即可完成，通信和计算开销均较小。

针对协议的输出，需要将布尔秘密共享值 ( $v = \langle v \rangle_0^B \oplus \langle v \rangle_1^B$ ) 转变为加性秘密共享值 ( $v = \langle v \rangle_0^A + \langle v \rangle_1^A$ )。根据如下观察， $v^A = \langle v \rangle_0^B + \langle v \rangle_1^B - 2\langle v \rangle_0^B \langle v \rangle_1^B$  即可将布尔共享值转变为加性共享值。为了计算  $\langle v \rangle_0^B \langle v \rangle_1^B$ ，首先令云服务器  $C_i$  与  $C_{1-i}$

秘密共享  $\langle v \rangle_i^B$ , 然后, 进行计算乘法。最后, 云服务器在本地计算最终密文比较结果。这里的示例为一对数据进行比较, 但是百万富翁协议实际上可以同时比较多对数据, 只这里值得注意的是需要生成不同的随机数  $a$ , 涉及乘法的部分三元组也需要进行更新。

---

**算法 3.2 SC  $\rightarrow (\langle \delta \rangle_0, \langle \delta \rangle_1)$** 


---

**输入:**  $C_0, C_1$  输入  $\langle x \rangle_i, \langle y \rangle_i, i \in [0, 1]$ .

**输出:**  $C_0, C_1$  获得  $\langle \delta \rangle_0, \langle \delta \rangle_1$

- 1:  $C_0$  生成随机数  $a, a \in \mathbb{Z}_N, \langle r \rangle_0 = \langle x \rangle_0 - \langle y \rangle_0 + a$ , 将  $\langle r \rangle_0$  发送给  $C_1$
  - 2:  $C_1$  计算  $\langle r \rangle_1 = \langle x \rangle_1 - \langle y \rangle_1$ , 还原  $r$  值  $r = \langle r \rangle_0 + \langle r \rangle_1 = x - y + a$
  - 3:  $C_0$  和  $C_1$  使用百万富翁协议来比较  $a$  和  $r$ , 获得结果  $\langle v \rangle_i^B, i \in [0, 1]$ ,  $\langle v \rangle_i^B$  代表  $\langle LT(a, r) \rangle_i^B$
  - 4:  $C_i$  选择随机数  $t_i \in \{0, 1\}$ , 计算  $\langle v' \rangle_{1-i}^B = \langle v \rangle_i^B \oplus t_i$ ,  $C_i$  将  $\langle v' \rangle_{1-i}^B$  发送给  $C_{1-i}$ , 因此  $C_0$  和  $C_1$  获取  $\langle v \rangle_i^B$  的秘密共享值
  - 5:  $C_0$  和  $C_1$  计算  $\langle \mu \rangle_i^b \leftarrow \text{MUL}(\langle v \rangle_1^B, \langle v \rangle_0^B)$
  - 6: 两方计算  $\langle \delta \rangle_i = \langle v \rangle_i^B - 2\langle \mu \rangle_i^b, i \in \{0, 1\}$
- 

### 3.4.3 安全极值计算协议

安全极值计算协议可以找到一组加性秘密共享值中极值的位置, 给定  $n$  个秘密共享的数据, 经过计算后, 输出仅包含 0、1 秘密共享值的  $n$  维数组, 其中极值位置存放 1, 其他位置均为 0。若需要计算极值的具体数值, 将结果数组与原始数组逐个进行乘法运算后累加结果即可。安全极值比较可以分为求解极大值和极小值, 二者方法类似, 这里以安全求解极小值为例进行阐述, 求解极大值只需将协议中求解  $LT(x, y)$  改为  $LT(y, x)$  即可。

在  $n$  个数据中找到极值的传统方式为通过冒泡排序逐个比较大小, 从而获得结果。上述方法需要进行  $n - 1$  次比较, 并且由于冒泡排序前后结果相互影响因此不能进行并行计算, 效率较低。本文提出的安全比较协议能够并行比较多对数据, 基于此, 针对数据集大小的不同场景分别设计了两种极值计算协议。

针对小数据集, 可以通过增加并行比较的数据对, 来减少比较的次数。以一个包含三个整数的数据集为例  $x_i \in \mathbb{Z}_N, i \in [1, 3]$ , 令每个  $x_i$  与其他两个不同的数据比较。现在有 6 个比较数据对  $(x_i, x_{j \neq i}), i, j \in [1, 3]$  进行一轮数据比较。具体构造流程如下图3.2所示:

结果标识为  $\{\langle \delta_1 \rangle, \dots, \langle \delta_6 \rangle\}$ , 由云服务器  $C_0$  和  $C_1$  秘密共享。然后利用 **MUL** 来累乘与  $x_i$  相关的比较结果, 获得  $\{\langle m_1 \rangle, \langle m_2 \rangle, \langle m_3 \rangle\}$ 。假设  $x_1 < x_2 < x_3$ , 与  $x_1$  相关的比较结果全部为 1, 同时与  $x_2, x_3$  相关的比较结果至少包含一个 0。因此,

假设我们有  $x_1 < \dots < x_n$ ,  $x_i \in \mathbb{Z}_N$

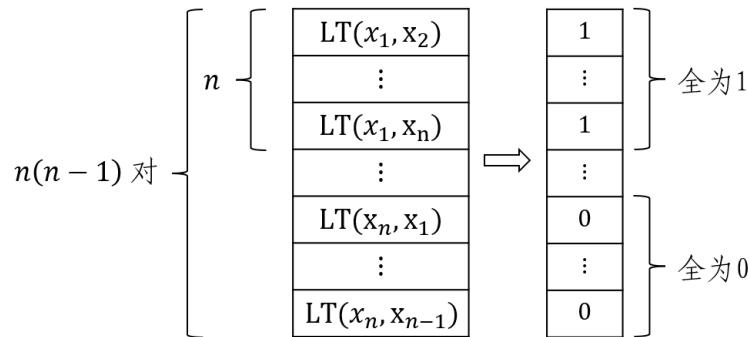


图 3.2 小数据集上的安全极值协议 (SMin (S))

对应的累乘结果为  $m_1 = 1, m_2 = 0, m_3 = 0$ 。详细描述如算法3.3所示，首先用两个数组分别存储待比较的内容，长度为  $n(n - 1)$ ，然后运行安全比较算法获取结果  $\langle t_i \rangle, i \in [1, n(n - 1)]$ ，最后将每一个元素  $x_i$  相关的  $n - 1$  个比较结果相乘得到最终极值结果  $\langle r_i \rangle, i \in [1, n]$ 。

---

**算法 3.3 SMin(S)  $\rightarrow \{\langle r_1 \rangle, \dots, \langle r_n \rangle\}$**

---

输入:  $C_0, C_1$  输入  $\{\langle x_1 \rangle, \dots, \langle x_n \rangle\}$

输出:  $C_0, C_1$  获得  $\{\langle r_1 \rangle, \dots, \langle r_n \rangle\}$

```

1: 构造  $n(n - 1)$  对比较数据，分别存储在  $l_1$  和  $l_2$  中
2: for  $i = 1$  to  $n$  do
3:   for  $j = 1$  to  $n, j \neq i$  do
4:      $l_1[i * (n - 1) + j] \leftarrow \langle x_i \rangle$ 
5:      $l_2[i * (n - 1) + j] \leftarrow \langle x + j \rangle$ 
6:   end for
7: end for
8: 比较大小获取结果  $\langle t \rangle \leftarrow \text{SC}(l_1, l_2)$ 
9: for  $i = 1$  to  $n$  do
10:   将  $\langle x_i \rangle$  相关比较结果相乘  $\langle r_i \rangle \leftarrow \text{MUL}(\langle t_{i*(n-1)+1} \rangle, \dots, \langle t_{i*n} \rangle)$ 
11: end for

```

---

针对大型数据集，上述方法增加的比较数据对以平方级剧增，带来大量冗余的通信和计算开销。因此，这里放弃增加比较数据对的思路，采取树型结构来减少比较轮次，在冒泡排序需要  $n - 1$  轮比较的基础上，将迭代轮次减少为  $\log n$ 。

假设数据集包含  $n$  个数据，首先比较  $n/2$  对数据，然后用比较结果与原始数据相乘后相加，以获取一对比较数据中的较小值。以  $x, y$  为例，比较结果为  $\delta = LT(x, y)$ ，较小值的计算方式为  $r = \text{MUL}(\delta, y) + \text{MUL}(1 - \delta, x)$ 。经过上述操

作数据集的大小由  $n$  变为  $n/2$ , 保留了所有比较数据对中的较小值。反复进行上述操作, 直到集合仅包含一个数据, 即极小值, 比较过程如图3.3所示。

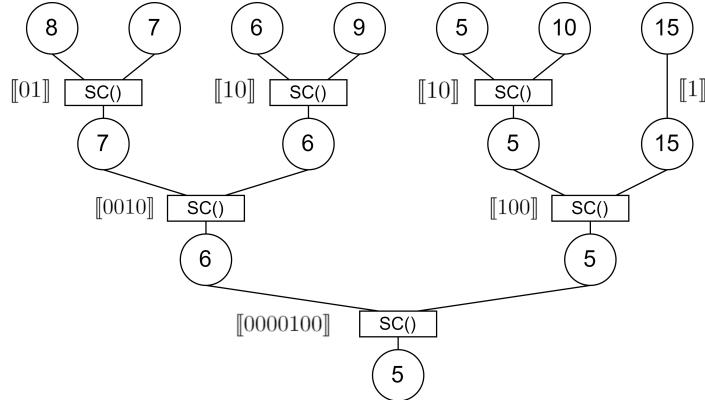


图 3.3 大数据集上的安全极值协议 (SMin(L))

#### 3.4.4 安全过滤协议

本节将介绍如何在秘密共享的 Kd-tree 数据结构上进行聚类, 该方案的正确性验证由论文<sup>[32]</sup>给出, 明文算法的详细过程在章节3.2.2.2给出。

自根节点开始, 依次遍历所有节点来判断当前节点中包含的所有数据点是否可以被划分到某一个簇中。针对每个节点, 维护一个候选簇集合, 该集合中所有簇都可能是节点内所有数据的最终所属簇。在迭代中, 不断删除不符合条件的候选簇直到仅剩一个候选簇, 即节点所属簇。

具体过程如算法3.4所示。首先, 计算 Kd-tree 中当前节点到每个候选簇中心的距离, 然后借助安全极值协议找到距离最近的候选簇并标记为  $z^*$ 。然后, 执行一系列子协议来排除不符合候选条件的簇, 排除依据为, 与候选簇  $z_j$ ,  $z_j \neq z^*$  相比, 节点中包含所有数据点在空间上都离  $z^*$  更近, 则认为  $z_j$  非候选簇, 更加详细的规则解释在论文<sup>[32]</sup>中给出。

在此之后, 通过乘法将被排除的候选簇标记为 0, 累加结果并执行安全比较协议来判断候选簇集合是否仅剩一个簇。如果是, 则将节点中所有数据划分到该候选簇中, 并停止向下迭代, 否则继续对子节点重复上述操作。若聚类过程即将收敛, 那么能够快速排除其他簇进行划分, 子节点可以不用进行冗余的距离计算和比较, 节省大量计算资源。然而上述方案泄露了一比特数据来判断是否终止子节点迭代, 这样少量的数据仅仅轻微泄露了迭代过程的信息, 但是带来了效率的巨大提升。论文<sup>[29, 77]</sup>中采取了类似的方法作为一个交换来获取性能提升。

值得一提的是, 不同节点之间包含的数据不相交, 并且划分的过程和结果不会相互影响, 因此可以引入并行处理来加速不同节点之间处理和计算, 进一步提

---

**算法 3.4 SF**

---

输入: Kd-tree  $\langle tr \rangle$  以及簇  $\langle z \rangle_j, j \in [k]$

输出: 新的簇中心  $\langle z' \rangle_j, j \in [k]$

```

1: for  $i = 1$  to  $n$  do
2:   for  $j = 1$  to  $k$  do
3:      $\langle d \rangle_j \leftarrow \text{SED}(\langle z \rangle_j, \langle c \rangle_i)$ 
4:   end for
5:    $\{\langle r \rangle_1, \dots, \langle r \rangle_k\} \leftarrow \text{SMin}(\langle d \rangle_1, \dots, \langle d \rangle_k)$ 
6:   计算最近候选簇  $\langle z^* \rangle \leftarrow \sum_{p=1}^k \text{MUL}(\langle r \rangle, \langle z \rangle_p)$ 
7:   for  $j = 1$  to  $k$  do
8:      $\langle r \rangle \leftarrow \text{SC}(\langle u \rangle, 0), \langle \mu \rangle \leftarrow \langle z^* \rangle - \langle z \rangle_j$ 
9:      $\langle v \rangle \leftarrow \text{MUL}(\langle r \rangle, \langle c \rangle_{min}) + \text{MUL}(1 - \langle r \rangle, \langle c \rangle_{max})$ 
10:     $\langle d^* \rangle \leftarrow \text{SED}(\langle z^* \rangle, \langle v \rangle), \langle d_j \rangle \leftarrow \text{SED}(\langle z \rangle_j, \langle v \rangle)$ 
11:     $\langle r \rangle_j \leftarrow \text{SC}(\langle d^* \rangle, \langle d_j \rangle)$ 
12:  end for
13:   $\langle f \rangle \leftarrow \text{SC}(\sum_{j=1}^k \langle r \rangle_j, 1)$ 
14:  if  $\langle f \rangle_0 + \langle f \rangle_1 == 1$  then
15:     $\langle \mu \rangle_j \leftarrow \langle \mu \rangle_j + \text{MUL}(\langle r \rangle_j, \langle c \rangle), j \in [k]$ 
16:  else
17:     $\langle z \rangle_j \leftarrow \text{MUL}(\langle z \rangle_j, \langle r \rangle_j), j \in [k]$ 
18:    将候选集  $\{\langle z \rangle_1, \dots, \langle z \rangle_k\}$  传递给子节点, 并重复上述过程
19:  end if
20: end for

```

---

提升安全过滤算法的效率。

### 3.5 基于 Kd-tree 的隐私保护 K-means 方案

本节主要介绍基于 Kd-tree 的隐私保护 K-means 方案 (PPOKC) 的具体细节。假设数据在双云服务器  $C_0$  和  $C_1$  上加性秘密共享, 并且二者不可以合谋。云服务器在密文基础上进行系列安全协议获取聚类结果, 整个过程可以被划分为两个阶段:

- **初始化:** 首先, 用户基于拥有的明文数据构建 Kd-tree。然后将数据秘密共享为两份, 并分别发送给云服务器  $C_0$  和  $C_1$ 。此外, 原始数据也会在被划分后发送给云服务器以支持其他计算。此后用户不再参与聚类过程。
- **聚类:** 对 Kd-tree 中节点执行过滤操作, 根节点的候选簇集合包含所有初始簇。针对树中的节点, 执行过滤操作, 直到所有数据都被划分到对应的簇。在每轮迭代结束, 更新簇中心并且将新簇与旧簇相比较来判断是否满足聚类收敛条件。

### 3.5.1 算法详述

PPOKC 方案的细节在算法3.5中给出，虽然这里提供了一种判断是否停止迭代的方法，在稍后的实验中会采取固定迭代次数的方式来测试性能。不同的数据集迭代收敛所需的轮次通常不同，目前许多相关隐私保护聚类研究采取的迭代终止策略通常存在安全问题，即泄露数据集迭代收敛所需轮次或者泄露了一定的数据。同时，K-means 聚类算法在迭代足够多轮次后一定会收敛<sup>[57]</sup>。

---

#### 算法 3.5 隐私保护外包聚类算法

---

输入: 明文数据  $x_{ij}, i \in [n], j \in [m]$   
 输出: 秘密共享的簇中心  $\langle z'_j \rangle, j \in [k]$

```

1: 初始化 (用户)
2: 在明文基础上构造 Kd-tree
3: 加性秘密共享数据  $x_{ij}$  以及 Kd-tree, 随机选择初始簇中心  $z_j, j \in [k]$ , 上述内
容分别发送给  $C_0$  和  $C_1$ 
4: 聚类 ( $C_0, C_1$ )
5:  $\{\langle z'_1 \rangle, \dots, \langle z'_k \rangle\} \leftarrow \text{SF}(\langle z_1 \rangle, \dots, \langle z_k \rangle)$ 
6:  $\langle r \rangle \leftarrow \sum_{i=1}^n \sum_{j=1}^m \text{SC}(\langle z_{ij} \rangle - \langle z'_{ij} \rangle, 1)$ 
7: if  $\langle r \rangle_0 + \langle r \rangle_1 == 1$  then
8:     终止迭代, 返回结果给用户
9: else
10:     $\{\langle z_1 \rangle, \dots, \langle z_k \rangle\} \leftarrow \{\langle z'_1 \rangle, \dots, \langle z'_k \rangle\}$ 
11:    回到步骤 5
12: end if

```

---

### 3.5.2 讨论

最近，论文<sup>[30]</sup>提出了一种双云环境下基于密文打包技术的高效隐私保护 K-means 聚类方案。然而，虽然方案非常高效，但是存在一些安全性问题。在论文<sup>[30]</sup>的 S3ED 算法中 11 行，可以看到  $Dst_{aj} \leftarrow Dst_{aj} * r_1^a + r_1^a$ , for  $j \in [k]$ ，意味着距离矩阵中行  $a$  中所有值均被添加相同的噪声。根据论文<sup>[78]</sup>章节 7.3 的分析，上述方式无法保护数据的分布信息。在本方案中，不同的距离均为秘密共享值，在进行安全比较时，添加了互异的随机噪声，因此不会泄露原始数据任何相关信息。

## 3.6 安全性分析

本节中，根据论文<sup>[79]</sup>中的标准模拟理论，假设双云服务器  $C_0$  和  $C_1$  为潜在的攻击者，并提供对于安全欧式距离、安全比较协议、安全极值协议、以及安全过

滤协议的书面安全性证明。然后，根据组合理论，证明 PPOKC 方案在半诚实模型下是安全的。为了论证方案的安全性，首先介绍半诚实模型下的安全性定义<sup>[80]</sup>：

**定义 3.2：**参与方  $A$  和  $B$  在半诚实模型下计算函数  $f$ ，若存在一对非均匀的概率多项式模拟器  $S_A, S_B$ ，以使得对于每一个安全参数  $n$  和所有输入  $x, y \in \{0, 1\}^n$ ，都有以下内容成立：

$$\begin{aligned} \{\mathcal{S}_A(x, f(x, y)), f(x, y)\} &\approx \{e \leftarrow [A(x) \leftrightarrow B(y)] : \text{View}_A(e), \text{Out}_B(e)\} \\ \{\mathcal{S}_B(y, f(x, y)), f(x, y)\} &\approx \{e \leftarrow [A(x) \leftrightarrow B(y)] : \text{View}_B(e), \text{Out}_A(e)\} \end{aligned} \quad (3.6)$$

此外，证明中需要用到如下引理：

**引理 3.1：**如果一个随机元素  $c$  在  $\mathbb{Z}_N$  上是随机分布的，且与  $\mathbb{Z}_N$  中的元素  $x$  无关，则  $c \pm x$  在  $\mathbb{Z}_N$  也是均匀随机分布的，与  $x$  的值无关。

接下来开始证明相关协议的安全性：

**定理 3.1：**安全欧式距离计算协议在半诚实模型下是安全的。

**证明：**安全欧式距离计算协议中，大部分计算由云服务器  $C_0$  和  $C_1$  分别在本地进行，没有泄露可能性。交互操作仅在乘法部分，该方法的安全性证明已经由论文<sup>[74]</sup>给出。因此，根据组合理论，本协议的安全性得证。

**定理 3.2：**安全比较协议在半诚实模型下是安全的

**证明：**该算法是基于文献<sup>[33]</sup>中的百万富翁协议，对输入和输出进行改造。 $C_0$  用不同的随机数打乱数据，然后发送给  $C_1$ ，这种方式根据引理3.1是安全的。输出部分，由布尔秘密共享值转换为加性秘密共享值的安全证明依赖于乘法的安全性。因此，证明安全比较协议是安全的。

**定理 3.3：**安全极值协议在半诚实模型下是安全的。

**证明：**小数据集上的安全极值协议是安全比较协议的变形，唯一的区别是在协议最后添加了一些乘法操作，因此安全性可以由安全比较协议推导出。对于大数据集上的安全极值协议，令云服务器  $C_1$  的执行镜像为  $\Pi_A(SC) = \{x_i^A, r_i^A, \alpha_i^A\}$ ，其中  $x_i^A$  为输入值， $r_i^A$  为中间值以及  $\alpha_i^A$  为比较结果，其中  $i, j \in [n]$ 。上述所有值均为秘密共享值。假设  $C_1$  的模拟镜像表示为  $\Pi_A^S(SC) = \{x'_i, r'_i, \alpha'_i\}$ ，模拟镜像中所有值都是  $\mathbb{Z}_N$  上的随机值。由于秘密共享均为  $\mathbb{Z}$  上的随机值，因此  $\{x_i^A, r_i^A, \alpha_i^A\}$  与  $\{x'_i, r'_i, \alpha'_i\}$  计算上不可区分。根据上述分析，可以认为  $\Pi_A(SC)$  与  $\Pi_A^S(SC)$  计算上不可区分。 $C_1$  和  $C_0$  执行的相同的工作，所以在这里省略重复分析。

---

**定理 3.4:** 本文所述的隐私保护外包 K-means 聚类方案在半诚实模型下是安全的。

证明：从3.5中可以看出，所述 PPOKC 方案是由上述安全协议组成的。在前面已经给出了基础安全协议的安全性证明，根据组合原理<sup>[79]</sup>，可以认为 PPOKC 方案在半诚实模型下是安全的。此外，由于云平台  $C_0$  和  $C_1$  是不能合谋的，因此它们也无法获知每个数据具体划分到哪个簇中，所以数据的访问模式也得到了保护。由于 Kd-tree 的数据结构特点，决定了每个节点中包含数据的个数已知，但是具体包含哪些数据是未知的，因此云平台无法跟踪数据与树节点的关系。

### 3.7 系统评估与性能分析

前述内容详细的描述方案中涉及协议的具体涉及以及相关知识，本节在依据方案实现系统的基础上详细进行理论分析和实验评估，在多个人工合成的数据集上进行实验以观察系统的实用性和可扩展性。同时，也在现实数据集上进行了充分的实验，以和目前最先进的隐私保护 K-means 聚类方案<sup>[30, 57]</sup>进行实验对比。

#### 3.7.1 理论分析

本节中，主要从理论层面分析方案的计算和通信开销。安全欧式距离仅包含简单的秘密共享加法和乘法运算，因此不纳入分析。安全比较算法主要基于论文<sup>[33]</sup>中的百万富翁协议，详细的效率分析已经在论文中给出，这里不再赘述。因此主要分析安全极值协议以及安全过滤协议。值得注意的是，乘法所需三元组的计算是离线进行的，因此这里生成乘法三元组的过程不纳入计算与通信开销分析。

##### 3.7.1.1 计算开销

考虑到上述两个协议，主要由安全比较和秘密共享乘法加法组成，乘法与加法相对安全比较协议来说计算开销可以忽略不计，因此在以下的论述中，以安全比较协议的运行次数为基准给出分析结果，如表3.1所示：

表 3.1 计算开销

协议	安全极值协议(大数据集)	安全极值协议(小数据集)	安全过滤协议
安全比较协议	$\log n$	1	$2kn$

借助安全比较协议，可以运行一轮协议比较多对数据，效率非常高。然而，随着比较数据的增加，计算开销也随之剧增。虽然小数据集上的安全极值协议只需要运行一次安全比较协议，但是当数据量变得很大时，协议运行非常耗时。因此设计了另一种能够减少比较数据量略微增加比较次数的协议。

### 3.7.1.2 通信开销

假设数据集大小为  $n$ , 数据为  $l$  比特, 这里仅分析乘法的通信开销, 因为安全比较协议的通信开销分析比较复杂, 并且在论文<sup>[33]</sup> 中已经给出。在安全比较协议, 以及小数据集上的安全极值协议, 通信开销分别为  $3nl$ ,  $7n(n - 1)l$  比特。对于大数据集上的安全极值协议, 开销最多不超过  $nl \log n$  比特。因为安全过滤协议的开销分析非常复杂, 这里仅给出近似值  $O(n^2 \log n)l$  比特。

### 3.7.2 实验分析

本节通过实验评估本文所述方案的性能, 我们基于 c++ 编程语言和论文<sup>[33]</sup> 中的函数库实现了整个系统, 并对用户与云平台以及云平台之间的交互进行了仿真模拟。我们主要关注系统在不同的数据集上的计算和通信开销, 以及与论文<sup>[30~31, 57]</sup> 在相同数据集上的实验表现进行对比。

为了与前沿研究论文<sup>[30, 57]</sup> 的实验进行直观的对比, 我们选择完全相同的实验机器, 配置为 Inter Core i7-7700HQ 2.80 GHz CPU 和 16 GB RAM。然而, 上述机器内存有限, 为了进一步考察方案的可扩展性, 我们选择在配置为 Intel Core i9-10980XE 3.00GHz CPU 和 64GB RAM 的机器上对人工合成的数据集进行系列实验, 考察簇个数  $k$ 、数据量  $n$  和数据维度  $m$  对于系统性能的影响。

#### 3.7.2.1 数据集

表 3.2 数据集详细信息

数据集名称	数据集大小 $n$	簇个数 $k$	维度 $d$
Lsun	400	3	2
KEGG	8192	3	5
Synthetic	{10000, 60000}	{3, 15}	{2, 20}

为了能够进行全面的对比, 这里使用了表格3.2所示数据集, 其中每一个都在之前的研究工作中出现过:

- KEGG: 是一个集成的数据库, 主要包含 15 个不同的数据库资源, 包含的内容主要是分子系统的计算机表示, 从基因信息到化学信息等。本方案选择的 KEGG Metabolic Reaction Network (Undirected), 是收录于 UCI KDD 条目的现实世界数据集<sup>[81]</sup>。本文从中提取 8192 条数据, 5 个维度 ( $n = 8192, m = 5$ ), 令簇个数  $k = 3$  进行实验, 以和论文<sup>[30]</sup> 的实验设置保持一致。
- Lsun: 数据集是来自论文<sup>[82]</sup> 的一个人造二维数据集, 该数据集常用于聚类研究以进行对比分析和展示。其中包含 400 个数据, 3 个簇。将用于与论

文<sup>[31]</sup> 的比较。

- **人工合成数据集：**本文构造了数据量为  $n$ , 维度为  $m$  的人造数据集, 随机值为整数, 均匀分布在  $[0, 1000]$  内。簇的个数在 3 到 15 之间以方便进行实验。

### 3.7.2.2 现实数据集上的实验

**与论文<sup>[30]</sup> 方案对比：**为了进行最直接的比较, 这里使用相同的数据集和一致的实验机器运行本系统。实验数据如表格3.3所示, 其中给出了 PPCOM<sup>[42]</sup>、SEOKC<sup>[30]</sup> 以及本方案 PPOKC 进行一轮迭代的通信和计算开销。

表 3.3 通信与计算开销对比 ( $n = 8192, m = 5, k = 3$ )

实验表现	PPCOM	SEOKC	PPOKC
计算开销	401 min	17 s	20 s
通信开销	1430 MB	148 MB	405MB

从计算耗时角度来看, PPCOM 方案和 SEOKC 方案迭代一轮分别需要 401 分钟和 17s, 而本文的 PPOKC 方案则需要 20s, 对比来看, 比 PPCOM 方案要快 1203 倍, 比 SEOKC 方案慢 3s。从通信开销角度来看, PPCOM 需要 1430MB 而 SEOKC 需要 148MB, 本方案则需 405MB。

综合来看, 本文的方案的表现远远超过 PPCOM, 接近目前最高效的 SEOKC 方案。值得一提的是, 从章节3.5.2可以看到, SEOKC 方案虽然高效但是存在不容忽视的安全性问题, 采取的随机噪声方法会暴露数据分布。而 PPOKC 方案则更加安全, 不会泄露相关信息, 因此性能上略微逊色是可以接受的。

**与论文<sup>[31]</sup> 的比较：**为了进行公平的比较, 这里在 Lsun 数据集上进行实验。对比论文的机器为 Inter i7-3770 3.4GHz 20GB RAM, 本文所用前述实验机器要慢 1.3 倍。

论文<sup>[31]</sup> 提供了 3 种不同的隐私保护 K-means 聚类方案, 分别是精确的、稳定的以及近似的方案。上述方案进行 15 轮迭代分别用时为 545.91 天、48.9 小时、15.47 小时。在本文的系统中, 完成相同的迭代轮次总共需要 20.94 秒, 用时远远少于前述三种方案。

此外, 实验设置相同的初始簇中心初始化, 在 Lsun 数据集上分别运行明文 K-means 方案以及本文提出的 PPOKC 方案, 分类结果如图3.4和图3.5所示。本方案所有安全协议的正确性均以论文<sup>[32]</sup> 中提出的基于 Kd-tree 的改进 K-means 方案为支撑。因此, 最终聚类结果和明文方案完全一致。

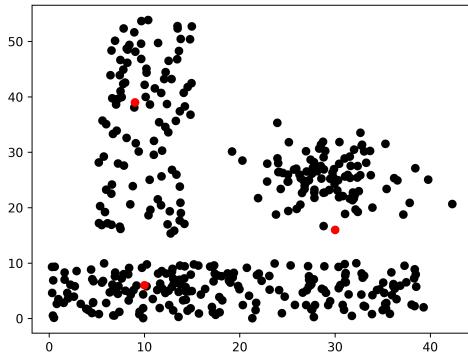


图 3.4 明文 K-means 聚类结果

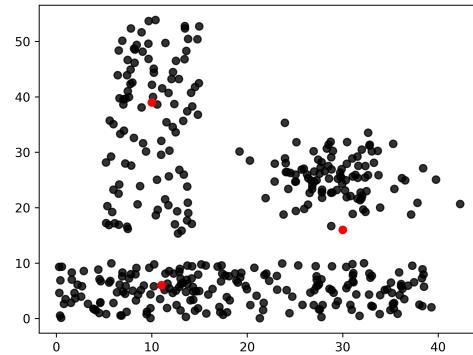


图 3.5 隐私保护 K-means 聚类结果

### 3.7.2.3 人造数据集上的实验

本节，在数据量跨度为  $n \in [10000, 60000]$  的人造数据集上进行系列对比实验。在生成数据时控制簇的个数在 3-15 之间，彼此之间没有重叠的部分。同时，聚类终止条件选择达到指定迭代次数即停止聚类比较结果。该选择主要原因如下：不同数据集收敛所需迭代次数不固定，受初始簇中心的影响较大，若按照算法中所给出的判断是否收敛来停止迭代，会导致时间分析不准确。

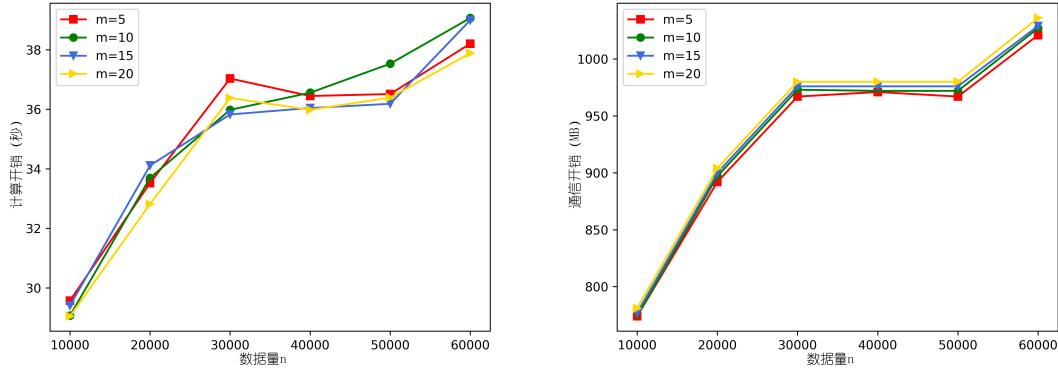
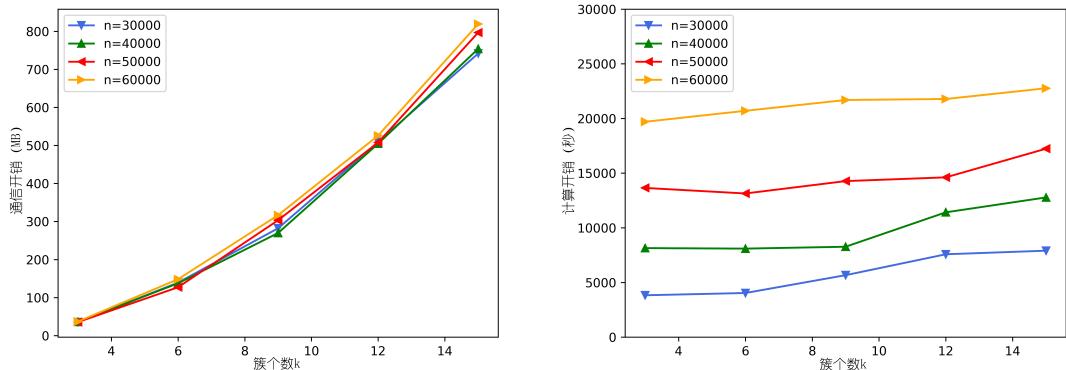
这里考虑三个影响聚类的参数，分别是数据集大小  $n$ 、数据的维度  $m$  以及簇的个数  $k$ 。为了准确的分析参数对于方案的性能影响，本文的策略是固定一个参数，然后变化另外两个参数的值来运行系统。实验运行的计算和通信开销如图3.6和图3.7所示。

对于图3.6，首先从曲线的变化可以看到计算开销和通信开销的变化趋势相似，其次二者均随着数据量  $n$  的增加而增加，但是从跨度来看，随着  $n$  的变化，计算开销的跨度较小，原因在于人工合成的数据集，数据分布比较规范，同时初始簇中心选择在每个生成的簇内，因此算法会快速收敛，冗余计算大大减少。维度  $m$  对通信开销的影响几乎可以忽略不计，而对于计算开销的影响则较小。具体原因在于，维度的变化只会对密文计算中的加法和乘法次数产生影响，不会影响安全比较协议的运行次数，而后者才是隐私保护聚类方案的瓶颈，此外，乘法操作的次数还可以通过引入并行操作进行优化。

对于图3.7，计算开销随着簇个数  $k$  的增加而呈现线性变化趋势，不同数据量之间的变化则较小，也符合上面的观察和解释。通信开销则对  $k$  的变化并不敏感，对数据量的变化计算开销随着  $k$  的变化呈线性变化，而  $k$  对通信开销的影响则较小。值得一提的是，给定相同的  $k$  值，不同的数据量对于计算开销的影响较小。

通过分析实验结果，本文提出的方案兼具高效与安全性，并且能够适应大型数据集。与研究<sup>[30, 42]</sup>相比，本文的方案在  $k$  较大时，实验表现更好。基于 Kd-tree

的改进 K-means 算法允许减少大量冗余计算，进一步加速聚类过程，此外，SIMD 操作可以很轻松被引入方案来提升效率。

图 3.6  $k = 3, T = 20$ 图 3.7  $m = 5, T = 20$ 

### 3.8 本章小结

隐私保护 K-means 聚类方案的研究通常难以兼具效率与安全性，实现完全隐私保护的方案耗时通常难以接受<sup>[31]</sup>，高效的方案则可能存在泄露隐私的风险<sup>[30]</sup>。过去研究常用到的密码学工具为秘密共享<sup>[57]</sup> 和同态加密<sup>[31][30]</sup>，二者各有优缺点。秘密共享技术进行加法和乘法运算较快，但是通信开销较大，而同态加密则计算开销较大，通信开销较小。

为了能够设计兼具安全与高效的隐私保护 K-means 聚类方案，本文在秘密共享的基础上，提出了一个基于 Kd-tree 的隐私保护外包 K-means 聚类方案。方案利用了 Kd-tree 数据结构来减少冗余计算，提升聚类效率，同时保证聚类的正确性。此外，提出了基于秘密共享设计了一系列隐私保护的计算协议，这些协议独立于

整个方案，可以灵活的用于其他各种基于秘密共享的隐私保护方案中。实验结果证明，本文的方案在保证聚类结果正确的基础上，保护了数据的隐私安全，减少了计算和通信开销，为用户提供了一种实际可行的外包聚类方案。

然而，聚类通常是对海量数据进行分析挖掘，数据量级可以达到亿万级别。在该场景下，即便是目前最高效的隐私保护聚类方案所需时间也是不可接受的。因此需要探索新的方案设计方向，即近似的隐私保护方案。机器学习对于计算的精度要求通常较宽松，因此可以考虑通过设计近似安全计算协议来简化计算过程，提升效率。

## 第四章 隐私保护密度聚类方法研究

### 4.1 引言

目前，已有许多关于隐私保护聚类方案的研究，其中数量最多的是与 K-means 相关的隐私保护研究<sup>[50]</sup>。然而，K-means 聚类存在诸多限制与缺点，其计算过程相对比较简单、聚类结果受到初始簇中心影响较大并且只能检测到凸型簇，因此适用场景极其有限。此外，簇的数量  $k$  必须要根据专业领域知识提前给出，如果只了解数据集的子集难以确定  $k$  的值。同时，K-means 聚类不包含噪声的概念，聚类的结果对异常值非常敏感因为每一个输入都要被划分到某一个簇中，影响该簇的中心位置（簇中所有数据的平均值）。

因此，一些研究人员转向基于 DBSCAN 的隐私保护聚类方案研究。DBSCAN 是一种由论文<sup>[83]</sup> 提出的更加灵活的聚类算法，能够检测到任意形状的簇。此外，簇的数量根据数据集的特点灵活产生，无需人工确定。该算法对异常值不敏感，会将其标记为噪声。关于隐私保护 DBSCAN 的相关研究相对较少，Rahman 等人<sup>[64]</sup> 基于同态加密技术提出了一种外包隐私保护 DBSCAN 聚类方案，然而该方案泄露了簇大小和数据相邻关系，存在安全隐患。Bozdemir 等人<sup>[29]</sup> 基于 ABY 安全计算框架提出了一种高效的隐私保护 DBSCAN 方案，显著提升了聚类的效率减少了耗时，但是方案设计不够完善，引入了人工设置参数。

为了解决上述问题，本文提出了三种不同的隐私保护 DBSCAN 聚类方案，分别解决了不同的问题。方案一是一种安全高效的隐私保护 DBSCAN 方案，该方案针对明文算法进行改进以适应安全计算的特点，对比前沿研究将密文方案复杂度降低为  $O(n^2)$ ，显著降低聚类所需时间。传统 DBSCAN 方案存在数据划分结果不稳定的问题，即最终划分结果取决于算法运行中数据遍历的顺序，将数据打乱后重新进行聚类，同一个点可能会被划分到不同的簇中。为了解决该问题，在论文<sup>[37]</sup> 思想的基础上，本文在方案二中提出了改进的隐私保护 DBSCAN，以获取稳定的数据划分结果，提升聚类质量。尽管 DBSCAN 具有诸多优点，其聚类过程涉及两个重要参数  $\text{minPts}$  和  $\epsilon$ ，这两个参数的取值与数据分布密切相关，通常需要人工分析后给出。为解决该问题，在论文<sup>[38]</sup> 的基础上，本文在方案三中提出了基于 DBSCAN 的隐私保护层次聚类，借助  $k$  近邻算法和  $k$  线图来决定聚类参数，并进行多轮聚类来适应不同密度的簇，获取尽可能全面的聚类结果，提高算法对于数据集的适应性。

本章的组织结构如下：第4.2节介绍了 DBSCAN 的相关内容。第4.3节中描述了系统模型、安全模型以及设计目标。第4.4节中增加了一些基于秘密共享的

隐私保护计算模块。第4.5节对方案一隐私保护 DBSCAN 方案进行了详细介绍。第4.6节中提出了方案二改进的隐私保护 DBSCAN 方案。第4.7节中阐述了方案三基于 DBSCAN 的隐私保护层次聚类方案。第4.8节中对方案进行了实验评估。最后，在4.9节对本章进行了总结。

## 4.2 预备知识

本节主要介绍聚类质量衡量指标、DBSCAN 的明文算法、改进的 DBSCAN 算法以及基于 DBSCAN 的层次聚类方案。

### 4.2.1 调整兰德系数

调整兰德系数 (Adjusted Rand Index, ARI)<sup>[84]</sup> 用于评估聚类模型的性能，它的前身是兰德系数 (Rand Index, RI)。兰德系数的取值范围为 [0, 1]，值越大意味着聚类结果与真实划分情况越吻合。假设  $U$  为外部评价标准 (即真实划分结果)，而  $V$  为聚类结果，这里设定 4 个不同统计量：

- $a$  为在  $U$  为同一类，在  $V$  也为同一类别的数据元素对数
- $b$  为在  $U$  中为同一类，但在  $V$  中属于不同类别的数据元素对数
- $c$  为在  $U$  中不在同一类，但在  $V$  中为同一类别的数据元素对数
- $d$  为在  $U$  中不在同一类，且在  $V$  中也不属于同一类别的数据元素对数

具体情况可以由如下表格概括：

表 4.1 数据集具体信息

分类	同一簇	不同簇	累加
同一簇	$a$	$b$	$a + b$
不同簇	$c$	$d$	$c + d$
累加值	$a + c$	$b + d$	$a + b + c + d$

因此，最终兰德系数计算方式如公式4.1所示：

$$RI = \frac{a + d}{a + b + c + d} \quad (4.1)$$

然而上述计算方式无法保证随机划分的聚类结果 RI 值接近 0，因此 Hubert 和 Arabie 等人在 1985 年提出了 ARI。下面给出该指标的计算方式，首先构造表格4.2，其中值  $n_{ij}$  表示数据集中数据同时位于簇  $X$  和簇  $Y$  的个数。

ARI 的计算方式如4.2所示，ARI 的范围为  $[-1, 1]$ ，该值越大则意味着聚类结果与真实情况越吻合。

表 4.2 ARI 指标计算中间值

分类	$Y_1$	$Y_2$	...	$Y_s$	累加值
$X_1$	$n_{11}$	$n_{12}$	...	$n_{1s}$	$a_1$
$X_2$	$n_{21}$	$n_{22}$	...	$n_{2s}$	$a_2$
...	...	...	...	...	...
$X_r$	$n_{r1}$	$n_{r2}$	...	$n_{rs}$	$a_r$
累加值	$b_1$	$b_2$	...	$b_s$	

$$ARI = \frac{\sum_{ij} \binom{n_{ij}}{2} - \left[ \sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2} \right] / \binom{n}{2}}{\frac{1}{2} \left[ \sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2} \right] - \left[ \sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2} \right] / \binom{n}{2}} \quad (4.2)$$

ARI 衡量指标的优点主要有，对于任意数量的聚类中心和样本数量，随机聚类结果的 ARI 值均非常接近于 0，同时还可以用于比较不同聚类算法的效果差异。但是缺点也非常明显，计算该值需要已知数据集正确划分结果。

#### 4.2.2 DBSCAN

DBSCAN 是一种基于密度的聚类算法，在密集区域聚集在一起的数据点被划分到同一个簇中，稀疏区域的数据被标记为噪声<sup>[85]</sup>。算法要求两个重要参数：

- $\epsilon$ : 也称为 eps，确定两个点被视为相邻的最大距离
- minPts: 确定邻域内至少包含多少点才能构成一个簇

DBSCAN 围绕每个数据点以  $\epsilon$  为半径构建圈，并将数据点划分为核心点、边界点以及噪声点。若圈内不包含任何点，则认为是噪声点。若数据点圈内包含点的数量至少为 minPts 个，则认为是核心点，否则为边界点。围绕上述定义展开，DBSCAN 还包含如下概念：

假设样本集为  $D = (x_1, x_2, \dots, x_m)$ :

- 密度直达：若  $x_i$  位于  $x_j$  的  $\epsilon$ -邻域内，且  $x_j$  为核心对象，则称  $x_i$  由  $x_j$  密度直达，反之不一定成立。
- 密度可达：对于  $x_i$  和  $x_j$ ，若存在样本序列  $p_1, p_2, \dots, p_t$ ，满足  $p_1 = x_i, p_T = x_j$ ，且  $p_{t+1}$  由  $p_t$  密度直达，则称  $x_j$  由  $x_i$  密度可达。
- 密度相连：对于  $x_i$  和  $x_j$ ，如果存在核心点  $x_k$ ，使得  $x_i$  和  $x_j$  均由  $x_k$  密度可达，则称  $x_i$  和  $x_j$  密度相连。

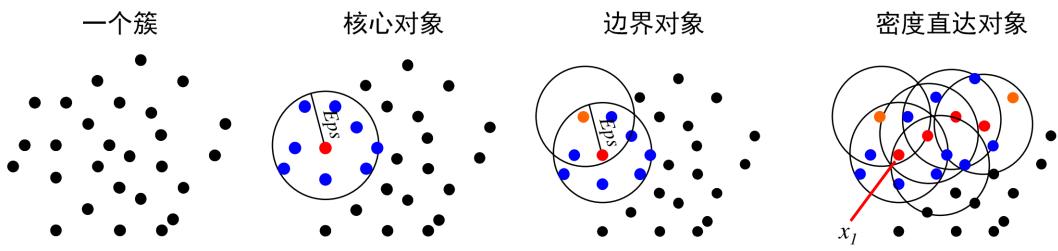


图 4.1 DBSCAN 中核心定义图示

下面具体阐述 DBSCAN 聚类算法的流程：

1. 初始化核心点集合  $\Omega = \emptyset$ , 初始簇数量  $k = 0$ , 初始化未访问点集合  $\Gamma = D$ , 簇划分  $C = \emptyset$
2. 遍历所有点  $i = 1, 2, \dots, m$ , 按如下方式找到所有核心点:
  - (a) 计算距离, 找到样本  $x_i$  的  $\epsilon$ -邻域内所有点集合  $N_\epsilon(x_i)$
  - (b) 如果集合包含数据点个数满足  $|N_\epsilon(x_i)| \geq minPts$ , 将  $x_i$  加入核心点集合:  $\Omega = \Omega \cup \{x_i\}$
3. 若  $\Omega = \emptyset$ , 则算法结束, 否则转入步骤 4
4. 在  $\Omega$  中, 随机选择一核心点  $o$ , 初始化当前簇包含核心点集合  $\Omega_c = \{o\}$ , 簇序号为  $k = k + 1$ , 当前簇包含点集合  $C_k = \{o\}$ , 更新未访问样本集合  $\Gamma = \Gamma - \{o\}$
5. 若  $\Omega_{cur} = \emptyset$ , 则簇  $C_k$  聚类完毕, 更新簇集合  $C = \{C_1, C_2, \dots, C_k\}$ , 更新核心点集合  $\Omega = \Omega - C_k$
6. 从  $\Omega_{cur}$  中取出一个核心点  $o'$ , 通过邻域距离  $\epsilon$  找到所有  $\epsilon$ -邻域子集  $N_\epsilon(o')$ , 令  $\Delta = N_\epsilon(o') \cap \Gamma$ , 更新当前簇包含点集合  $C_k = C_k \cup \Delta$ , 更新未访问点集合  $\Gamma = \Gamma - \Delta$ , 更新  $\Omega_{cur} = \Omega_{cur} \cup (\Delta \cap \Omega) - o'$ , 转入步骤 5
7. 输出结果为: 簇划分  $C = \{C_1, C_2, \dots, C_k\}$

DBSCAN 能够应用于任意维度的数据集。明文算法最坏情况下的复杂度为  $O(n^2)$ , 其中  $n$  为数据的数量。设置好  $\epsilon$  和  $minPts$  后, 无需指定  $k$  值, 运行算法即可自动划分成簇, 噪声点对聚类结果影响较小, 最后不会被划分到任何簇中。

### 4.2.3 改进的 DBSCAN

正如提出 DBSCAN 的论文<sup>[83]</sup> 中所说，该算法在检测相邻簇的边界数据点时划分结果不稳定。若核心点  $x_1$  与  $x_2$  分别属于不同的簇  $C_1$  和  $C_2$ ，二者邻域范围内均包含边界点  $x_3$ ，则依据 DBSCAN 划分标准， $x_3$  既可以被划分到  $C_1$ ，也可以被划分到  $C_2$  中，划分结果主要取决于  $x_3$  在算法运行时被处理的顺序。

下面给出一个具体的示例，假设二维数据集中包含两个簇，分别包含约 250 个数据，球形簇中间有部分数据相邻。图4.2为测试数据集的正确划分结果，图4.3为首次运行明文 DBSCAN 的结果，集合 2 中部分数据被划分到集合 1 中。图4.4为打乱数据集中数据顺序后运行明文 DBSCAN 的结果，可以看到划分结果发生了变化，右侧集合中部分数据被划分到左侧簇中。图4.3和图4.4中数据划分均产生一定的误差。

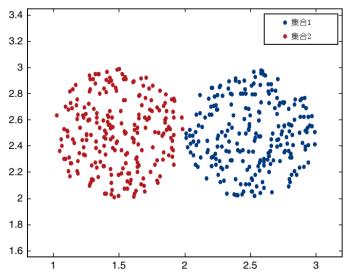


图 4.2 测试数据集

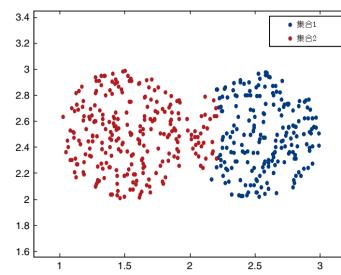


图 4.3 DBSCAN 聚类结果

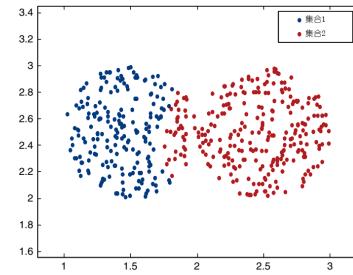


图 4.4 数据打乱后，DBSCAN 聚类结果

在论文<sup>[37]</sup> 中，作者认为边界对象对于传统 DBSCAN 算法簇的扩展没有贡献，因此可以通过改造算法的形式，以使得边界点的划分结果稳定。文章提出在聚类过程中，暂时不处理边界对象，直到所有核心对象都被分配到对应的簇中。最后，将所有未被划分的边界对象划分到最近的核心对象所属簇中即可。在相同的数据集上，改进的 DBSCAN 算法划分结果如图4.5所示：

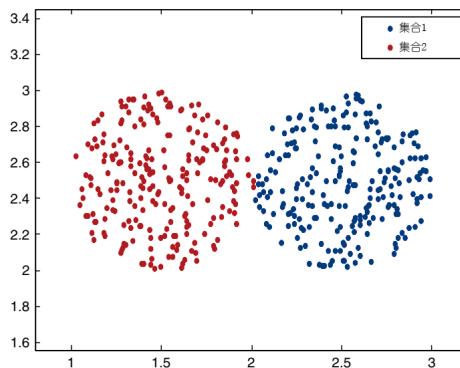


图 4.5 改进的 DBSCAN 算法划分结果

#### 4.2.4 基于 DBSCAN 的层次聚类

传统 DBSCAN 算法中， $\text{minPts}$  和  $\epsilon$  的取值对于聚类效果有极大影响，但同时又难以确定，因为二者受数据分布的影响较大。为了解决这个问题，论文<sup>[38]</sup> 提出一种基于 DBSCAN 的层次聚类方案。首先借助  $k$  近邻算法和  $k$  线图来决定  $\epsilon$  的值。实际应用中数据大多不是均匀分布的包含多个密度，因此需要计算不同的  $\epsilon$  值，然后根据不同的  $\epsilon$  值进行多次聚类。最后，若最终结果所得簇个数  $|C|$  超过了用户实际需要的数量  $k$ ，则反复选择最近的簇进行合并，直到满足要求。

关于参数  $\text{minPts}$ 、 $k$ 、以及  $\epsilon$  的获取依据见论文<sup>[38]</sup>。这里概括核心步骤并进行简要阐述， $\text{minPts}$  值越小，形成的簇个数越多，相反则越少。如果  $\text{minPts}=1$ ，则会创建密度为 1 的簇，不合理。若  $\text{minPts}=2$ ，则 DBSCAN 的结果近似于层次聚类的结果，因此  $\text{minPts}$  的最小值应为 3，最大值则是通过数据集大小  $M$  和簇的个数  $C$  给出 ( $\text{minPts}_{\max} = M \div C$ )。经过分析，若  $\text{minPts}$  取值为 3，则  $k$  的取值为必须为 2，具体策略在论文<sup>[38]</sup> 的 3.1 节给出。最后，由于数据集中具体包含多少个不同的密度是无法获知的，因此采取用系列  $\epsilon$  值迭代聚类的策略来获取最终结果， $\epsilon$  的取值主要借助  $k$  线图。方法在算法4.1 中给出：

首先，获取数据集的大小  $m$ ，在计算点之间的距离后，找到每个点第二近的距离。按照升序排列获取的距离值  $Dist$ ，然后以  $\sqrt{m}$  为间隔在  $Dist$  中取值，即所需  $\epsilon$  值。

---

#### 算法 4.1 获取 $\epsilon$ 值

---

输入：数据集  $D$

输出： $\epsilon$  值集合  $E$

```

1:  $m = |D|$ 
2: for  $i = 1$  to  $m$  do
3:      $Dist[i] =$  距离  $i$  第二近的距离
4: end for
5: 以升序排列  $Dist$ 
6:  $j = \sqrt{m}$ 
7: while  $j < m$  do
8:      $\{E\} = \{E\} + Dist[j]$ 
9:      $j = j + \sqrt{m}$ 
10: end while

```

---

其次，获取划分的初始簇，具体流程如算法4.2所示。每次取最小的  $\epsilon$  值来执行 DBSCAN，设置  $\text{minPts}$  的值为 3。然后，将本轮划分好的数据从原始数据集中剔除，不再参与后续聚类过程。最后，将本轮迭代所用的  $\epsilon$  值剔除，重复上述过

---

程，直到遍历完所有  $\epsilon$  值。

---

#### 算法 4.2 划分初始簇

输入：数据集  $D$ ,  $\epsilon$  值集合  $E$

输出：初始簇

```

1: while  $|E| > 0$  do
2:    $e = \min(\{E\})$ 
3:   执行 DBSCAN( $D, e, 3$ )
4:    $D'$  为已经被划分所属簇的点
5:    $\{D\} = \{D\} - D'$ 
6:    $\{E\} = \{E\} - e$ 
7: end while

```

---

最后，将初始簇进行合并获取最终结果。如算法4.3所示，首先计算所有簇的中心，即累加簇中所有数据的值，然后取平均。然后计算簇中心之间的距离，选择最近的两个簇合并为一个簇，形成一个新的簇，重新计算簇中心，并且更新数据点的划分关系，初始簇集合大小减一。重复上述过程，直到剩余簇个数为给出的  $k$  个。

---

#### 算法 4.3 合并初始簇

输入：目标簇个数  $k$ , 初始簇

输出：聚类结果

```

1: 初始簇个数为  $n$ 
2: 计算每个簇的中心
3: while  $k > n$  do
4:   找到最近的两个簇  $C$  和  $C'$ 
5:   合并  $C$  和  $C'$ 
6:   计算新簇的中心
7:    $n = n - 1$ 
8: end while

```

---

### 4.3 问题描述

如图4.6所示，系统由两个部分组成：服务器端和用户端。与图3.1类似，服务器端为两个云服务器，获取用户数据后执行一系列安全协议，最后分发结果。

不同的是，图3.1中客户端为某一个持有全部明文数据的组织机构，本方案系统模型中的用户既可以是单一机构，也可以是各自持有不共享数据的不同机构。第二种场景下，用户可以是不同的医疗机构，期望能够在不泄露患者信息的情况下，

联合起来对医疗诊断或者症状信息进行聚类来做数据挖掘与分析。用户将数据进行秘密共享后发送给双云服务器即可。在获取到聚类结果后，不同机构各自执行简单的算法来还原结果。

#### 4.3.1 系统模型

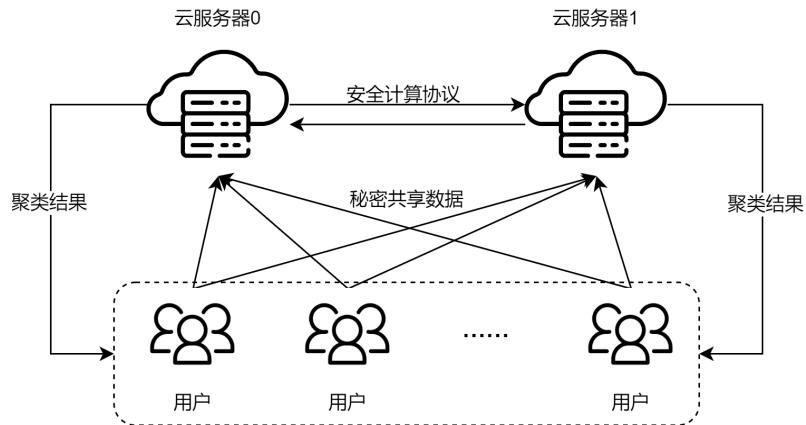


图 4.6 隐私保护 DBSCAN 方案系统模型

#### 4.3.2 安全模型

本方案基于半诚实模型的假设，细节说明见3.2.1中的说明。具体来说：  
对用户来说，持有数据的不同用户可能会根据聚类结果和云平台给予的信息，推断其他用户输入数据的具体内容。

对服务端来说，不共谋的云平台可能会试图根据秘密共享的数据、中间计算结果以及聚类结果推测原始内容，挖掘关键信息。

#### 4.3.3 设计目标

针对三个解决不同问题的隐私保护 DBSCAN 方案，设计目标不尽相同。对于方案一，本文期望

- **高效性：**遵循明文 DBSCAN 算法的思路，设计一个效率较高的密文方案，使得通信和计算开销均较小。与前沿隐私保护 DBSCAN 方案相比，在不损失安全性的前提下更加高效。
- **准确性：**我们的首要目标是密文方案与明文方案聚类结果一致，设计可靠的安全计算协议，获取精度不受损失的聚类划分结果。
- **安全性：**我们重点关注如何在密度聚类的过程中，保护用户原始数据、中间计算结果和最终聚类结果中包含的隐私或敏感信息，防止云服务器或其他用户获取相关信息。

在上述目标的基础上，方案二的设计目标为，在效率损失可接受的情况下，设计一个能够获取稳定聚类划分结果的隐私保护 DBSCAN 方案，划分结果与明文保持一致，不损失任何安全性。

方案三的设计目标为，设计一个能够不依赖人工设置参数的隐私保护 DBSCAN 方案，方案能够有效划分包含不同密度数据的数据集，同时无需对数据集进行前置数据分析以获取适合的  $\epsilon$  和 minPts 值，最后获得近似的划分结果。

#### 4.3.4 系统输入

数据持有者首先利用加性秘密共享来将数据划分为两份，分别发送给两个云服务器。采用加性秘密共享分发数据的方式允许系统中加入任意多个数据持有者提供聚类数据。此外，本文所述系统能够支持任何数据划分方式，例如水平划分、垂直划分以及混合划分。水平划分方式指的是，不同用户分别持有完整但是互异的数据<sup>[86]</sup>，垂直划分方式则是指不同用户持有相同数据记录的不同参数<sup>[45]</sup>，混合划分方式则是上述两种方式的结合<sup>[87]</sup>。

### 4.4 基于秘密共享的隐私保护计算模块

在3.4节所述算法的基础上，根据隐私保护 DBSCAN 方案的特点设计了如下计算模块。

#### 4.4.1 安全排序协议

安全排序协议主要由安全比较协议构成，用户输入秘密共享的数组后，期望能够得到按照升序排列的密文结果。协议的核心思想为，将  $n$  个数据上的排序拆分为  $n$  次求解序列最值。

---

#### 算法 4.4 安全排序协议

---

输入:  $D = \{\langle x \rangle_1, \langle x \rangle_2, \dots, \langle x \rangle_n\}$

输出: sorted result  $D' = \{\langle x' \rangle_1, \langle x' \rangle_2, \dots, \langle x' \rangle_n\}$

```

1: for  $i = 1$  to  $n$  do
2:    $\{\langle t \rangle_1, \dots, \langle t \rangle_n\} \leftarrow \text{smin}(D)$ 
3:    $res \leftarrow 0$ 
4:   for  $j = 1$  to  $n$  do
5:      $res \leftarrow res + \text{mul}(\langle t \rangle_j, D[j])$ 
6:      $D[j] \leftarrow \text{mul}(D[j], 1 - \langle t \rangle_j) + \text{mul}(\text{maxv}, \langle t \rangle_j)$ 
7:   end for
8:    $r[i] \leftarrow res$ 
9: end for

```

---

具体过程如算法4.4所示，进行 $n$ 次迭代，首先求解最小值，结果为秘密共享值 $\langle t \rangle_i \in \{0, 1\}$ ,  $i \in [1, n]$ ，数组中最小值的位置为1，其余均为0。然后，开始遍历每个结果， $res$ 通过累加记录最小值的具体值，并更新原始数组，通过将最小值变为全局最大值，以消除对后续计算的影响。若当前元素为最小值（即 $\langle t \rangle_i = 1$ ），则对应 $D[i]$ 中的值变为最大值 $\text{maxv}$ ，若不是则保留原值，不做修改。

## 4.5 隐私保护 DBSCAN 方案

在正式介绍本文设计的隐私保护 DBSCAN 方案之前，首先通过一个示例来详细阐述方案的核心思想。在 DBSCAN 算法的基础上，本文引入了临时簇的概念，令核心对象与其 $\epsilon$ 范围内未被划分的边界对象组成独立的临时簇，临时簇内所有数据对象的簇标识保持一致。在遍历的过程中对于核心对象，修改其 $\epsilon$ 范围内未被划分的边界对象的簇标识，与当前核心对象保持一致。对于核心对象 $\epsilon$ 范围内的其他核心对象，记录二者的相邻关系。在还原聚类结果时，根据相邻关系构造邻接矩阵（相邻对象对应值大于0），通过深度优先遍历算法找到相邻的临时簇合并，不相邻的临时簇不受影响。

假设当前有一个包含10个点的数据集 $\{p_1, \dots, p_{10}\}$ ，令 $\text{minPts}$ 为2，用圆圈框出所有数据对象的 $\epsilon$ 邻域，数据集可视化后如图4.7所示，其中虚线框为边界对象 $\epsilon$ 邻域，实线框为核心对象 $\epsilon$ 邻域。在图4.8中，我们用蓝色表示核心对象，绿色表示边界对象，噪声点在聚类过程中不会进行任何处理，为了简化表示，示例数据集没有噪声点。

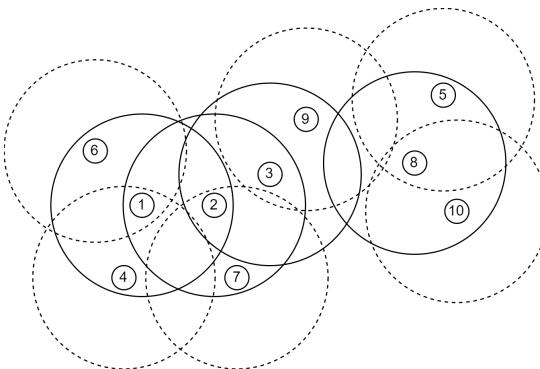


图 4.7 数据可视化

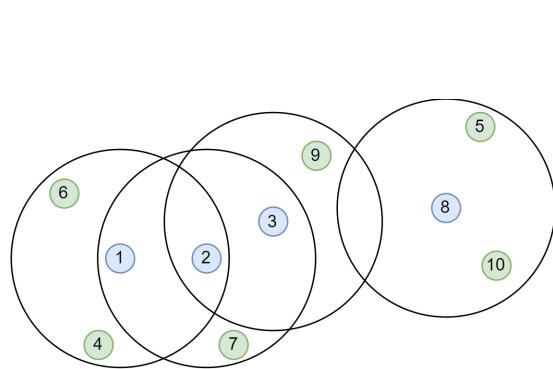


图 4.8 核心对象与边界对象

接下来，在上述示例数据集的基础上，将详细介绍聚类的过程。下述过程按照编号 $\{1, 2, \dots, 10\}$ 进行遍历，每一个数据对象的初始簇标识为图中序号。首先，对 $p_1$ 进行处理，在其 $\epsilon$ 范围内，有边界对象 $p_6$ ,  $p_4$ 和核心对象 $p_2$ 。针对边界对象，我们修改其编号与 $p_1$ 保持一致；针对核心对象，我们记录相邻关系；不在 $\epsilon$ 范围内的点（例如 $p_3$ 和 $p_9$ ），不做任何处理。对 $p_1$ 遍历完所有数据对象，结果

如图4.9.a 所示，这里对处理过的内容进行颜色加深方便展示。其他核心对象 ( $p_2$ ,  $p_3$ ,  $p_8$ ) 的处理和  $p_1$  一致（详细过程见图4.9中的 b、c、d），遍历完成的结果如图4.9.d 所示。

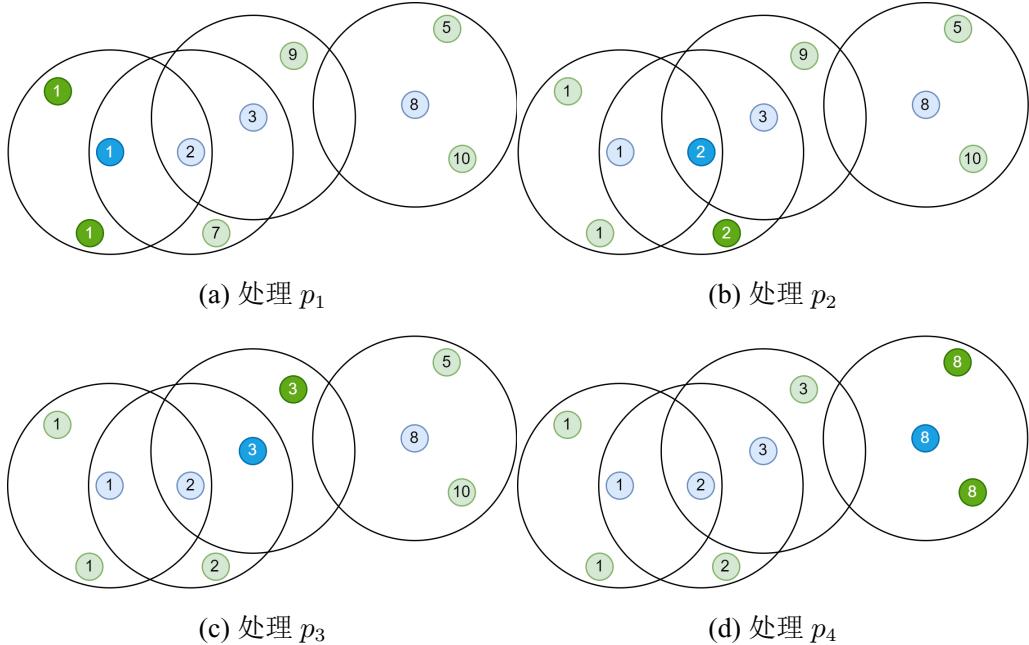


图 4.9 聚类处理过程

接下来，我们根据遍历过程中存储的相邻关系构造矩阵，结果如图4.10所示，可以看到只有核心点之间的存在相连关系（灰色部分）。基于该邻接矩阵，运行深度优先遍历算法，可以将临时簇进行合并。

	1	2	3	4	5	6	7	8	9	10
1	0	2	0	0	0	0	0	0	0	0
2	2	0	2	0	0	0	0	0	0	0
3	0	2	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

图 4.10 相连关系

经过合并后，得到最终聚类结果，如图4.11所示，可以看到数据集包含两个簇，其中黄色为簇 1，紫色为簇 2。以上即是本节所述密文方案的具体示例，方案

的复杂度为  $O(n^2)$ ，与明文方案一致。我们通过限制临时簇只能包含未被划分的边界对象，以避免边界对象被反复划分到不同的临时簇中的问题。

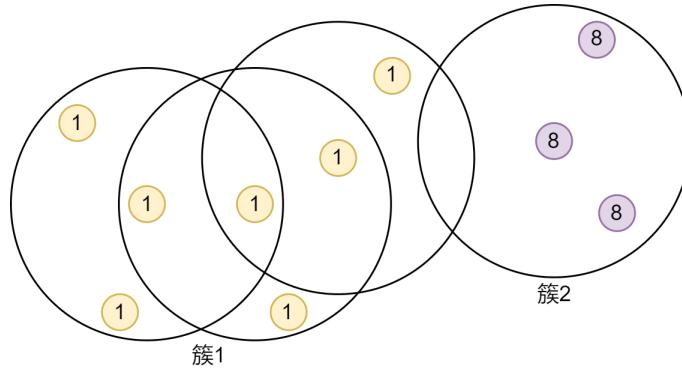


图 4.11 还原聚类结果

明文 DBSCAN 算法采取的策略是随机选择一个未被划分的核心对象，不断的找到密度可达的其他核心对象并将所有相邻的边界对象纳入当前簇集合，直到没有新的核心对象能加入当前簇集合。若沿用上述思路直接设计密文方案，会导致方案复杂度达到  $O(n!)$ ，计算开销过大难以应用于实际场景中。因此本文引入临时簇的概念，将方案的迭代深度减少为 2 次，极大的降低了密文方案的复杂度。记录相连关系本质上是在记录核心对象之间的密度可达关系，密度不可达的核心对象以及边界对象在邻接矩阵中的值都为 0。最后我们让用户在明文上基于相邻关系运行深度优先遍历将相邻的临时簇进行合并，来获取最终聚类结果。各自持有数据的独立用户，无法知道彼此拥有的明文数据，仅知道簇的划分情况，达到了保护数据安全的目的。

接下来详细介绍本文提出的隐私保护 DBSCAN 方案的具体过程，主要分为三个阶段，具体介绍如下：

- **预处理元素：**通过获取的密文数据，初始化存储在云服务器上的自定义数据结构体（以下简称元素），然后计算不同元素之间的欧式距离关系，并判断每个元素是否为核心点。
- **密度聚类：**根据第一阶段获取的信息，提出了一种全新的聚类方法，引入临时簇这一概念，结合不同核心对象之间的相连关系，可以将相连的临时簇进行合并。聚类过程即是标记临时簇，并记录相连信息。
- **还原聚类结果：**用户获取聚类记录的临时簇相连信息和簇标记信息，通过深度优先遍历算法来还原数据簇划分结果。

### 4.5.1 预处理元素

云平台在获取的秘密共享数据基础上，进行初始化。对于每个秘密共享的数据，构造 sharedElem 对象。cluId, cluId  $\in [1, n]$  为初始簇中心，值取该数据在数组中的顺序，比如第一个数据初始化 cluId 为 1，最后一个数据的 cluId 则为 n。isMark, isMark  $\in [0, 1]$  表示该数据对象是否已经被划分到簇中，0 表示未被划分到簇中，1 则表示已经被划分。data 则存储该数据对应的秘密共享值，是一个大小为 m 的向量。最后 isCore, isCore  $\in [0, 1]$  标识数据对象是否为核心点，主要用于在聚类时参与计算，详细内容如算法4.5所示。

---

#### 算法 4.5 sharedElem 数据结构

---

```

1: class sharedElem(sdata,idx):
2:     cluId = idx
3:     isMark = 0
4:     data = sdata
5:     isCore = 0
6:     resCId = 0

```

---

构造完 sharedElem 对象后，得到了长度为 n 的 sharedList 数组。在初始化阶段，我们需要们计算对象之间的距离，获取相邻关系，同时判断每个元素是否为核心点。具体过程如算法4.6所示。

---

#### 算法 4.6 预处理元素

---

输入：数组 sharedList 简写为  $\{x_1, \dots, x_n\}$   
输出：相邻关系二维数组 u，更新后的 x

```

1: cnt  $\leftarrow [0, \dots, 0]_n$ 
2: for i = 1 to n do
3:     for j = 1 to n do
4:          $d_j \leftarrow \text{DIST}(x[i], x[j])$ 
5:     end for
6:      $u_i \leftarrow \text{SC}(d, [\epsilon, \dots, \epsilon]_n)$ 
7:     cnt  $\leftarrow \text{sum}(u_{ij})$  for  $j \in [n]$ 
8: end for
9: r  $\leftarrow \text{SC}(cnt, [minPts, \dots]_n)$ 
10: for i = 1 to n do
11:     x[i].isCore = r[i]
12: end for

```

---

首先，构造数组 cnt 存储每个元素邻域范围内包含其他元素的个数。然后，通

过 DIST 计算元素之间的欧式距离，去掉根号便于计算，将第  $i$  个元素与其他所有元素的距离  $d_j, j \in [n]$  与  $\epsilon$  进行一次安全比较，判断二者是否相邻，结果存储到二维数组  $u_{ij}, i, j \in [n]$  中，最后累加  $u_i$  的值，统计元素  $i$  邻域范围内包含点的个数，存储到 cnt 中。最后，并行比较存储邻域点个数的 cnt 数组与长度为  $n$  的值均为 minPts 的数组之间的大小关系，并更新 sharedList 的 isCore 属性。

值得一提的是，这里的  $\epsilon$  值与明文 DBSCAN 算法中的不同，对欧式距离计算舍去根号，因此  $\epsilon$  值在原来的基础上取平方，为了方便表示，这里省略了详细说明。

#### 4.5.2 密度聚类

初始化之后，获得了元素的 isCore 信息，以及元素之间的相邻关系  $u_{ij}, i, j \in [n], u_{ij} \in \{0, 1\}$ 。接下来，这里将详细说明如何根据预处理过程中已经获取的信息聚类的过程。

在本节的开头提到了基于 DBSCAN 设计隐私保护聚类方案的难点，论文<sup>[29]</sup>为了解决这个问题，提出了两种解决办法，一方面可以人工设置了最大迭代深度，但是该方法不具备普适性，每次都需要结合数据集的特点给出最大迭代深度；另一方面则是通过还原 1 比特的数据来判断是否需要终止迭代。

本文为了解决这一问题，引入了临时簇，在遍历过程中，记录临时簇之间的相连关系（即核心对象之间的距离小于  $\epsilon$ ），同时核心对象会将邻域范围内所有未被标记的边界对象的 cluId 修改为与自身一致。在获取聚类结果阶段，构造邻接矩阵，将相连的临时簇进行合并，获取划分结果。

相连信息存储在 info 数据结构中，具体内容如算法4.7所示，其中 id1 和 id2 记录 sharedElem 中的 cluId 信息，isConn 则记录临时簇之间的相连关系，取值为 0 或 1。

---

#### 算法 4.7 info 数据结构

---

```

1: class info(cluId1,cluId2,conn):
2:     id1 = cluId1
3:     id2 = cluId2
4:     conn = isConn

```

---

在初始化的过程中，默认每个元素的初始 cluId 为下标  $\{1, 2, \dots, n\}$ 。在聚类的过程中，对于核心对象，修改其  $\epsilon$  范围内所有未被标记的边界对象的 cluId，并将状态修改为已标记。对于所有非核心点，无法修改其他点的属性，并且被标记后，无法再被其他核心点标记，噪声不会受到任何影响。经过上述过程，数据集已经被划分为若干临时簇，同一临时簇内所有数据对象的簇标识 cluId 一致。

---

**算法 4.8 密度聚类**

---

输入: 数组 sharedList 简写为  $x$ , 相邻关系  $u_{ij}, i, j \in [n]$

输出: 簇连接关系  $plist$

```

1: for  $i = 1$  to  $n$  do
2:   for  $j = 1$  to  $n, j \neq i$  do
3:      $m_1 \leftarrow \text{MUL}(x[i].isCore, u_{ij}, 1 - x[j].isCore)$ 
4:      $m_2 \leftarrow \text{MUL}(m_1, 1 - x[j].isMark)$ 
5:      $x[j].cluId \leftarrow \text{MUL}(m_2, x[i].cluId) + \text{MUL}(1 - m_2, x[j].cluId)$ 
6:      $conn \leftarrow \text{MUL}(x[i].isCore, x[j].isCore, u_{ij})$ 
7:     add new info item  $[x[i].cluId, x[j].cluId, conn]$  to  $plist$ 
8:      $x[j].isMark \leftarrow m_1 + x[j].isMark - \text{MUL}(m_1, x[j].isMark)$ 
9:   end for
10: end for

```

---

具体流程如算法4.8所示,为了简化说明,将输入数组简写为  $d$ ,并将相连关系的结果记录在  $plist$  数组中作为输出结果,每个元素的结构为  $info$  类型。

在整个算法的执行过程中,通过各种运算来实现逻辑与和逻辑或,借助计算一些辅助标识来控制是否修改内容。首先,获取标识  $m_1$  来辅助后续计算,若  $x[i]$  为核心对象、 $x[j]$  为非核心对象并且二者相邻,则  $m_1$  为 1,否则为 0。若在  $m_1$  为 1 的情况下,  $x[j]$  未被标记,则  $m_2$  为 1。这里根据  $m_2$  来控制是否修改  $x[j]$  的  $cluId$ ,若不满足条件,则  $x[j]$  的  $cluId$  保持原样,否则修改为与  $x[i]$  的  $cluId$  一致。 $conn$  反映的是  $x[i]$  与  $x[j]$  是否相连,若  $x[i]$  和  $x[j]$  均为核心对象并且  $u_{ij} = 1$  则  $conn$  等于 1,否则为 0,然后新增  $info$  类型的数据到  $plist$  中。值得一提的是,这里无论是否相邻,都会记录新的  $info$  信息,因此最终  $plist$  的长度为  $n(n - 1)$ 。最后,更新  $x[j]$  的  $isMark$  标识,若  $x[j]$  已被划分 ( $isMark=1$ ) 或者本次迭代过程中被划分 ( $m_1 = 1$ ),则  $x[j]$  的  $isMark$  标识为 1,否则为 0。算法中的计算方式体现的是逻辑或关系。

论文<sup>[29]</sup> 中提出的 ppDBSCAN 方案将复杂度从  $O(n^3)$  降为  $O(\text{maxIterations} \cdot n^2)$ ,本节中,上述聚类过程的复杂度为  $O(n^2)$ ,并且不会产生任何精度损失也不会泄露任何密文信息,方案更加安全高效。

#### 4.5.3 还原聚类结果

聚类后,在  $plist$  中记录了所有临时簇的相邻关系。云服务器分别将自己持有的份额分别发送给用户。用户运行  $\text{Rec}(\cdot, \cdot)$  还原结果后,在本地分别运行还原算法以获取最终结果。方案较简单,对计算资源要求较低。在算法4.9和算法4.10中将详细说明如何还原结果。

#### 算法 4.9 还原结果

输入: 明文 plist

输出: 获取 resCId 的 plist

```
1: 构造二维数组 matrix, 大小为  $n \times n$  存储相连关系
2: 构建 resmap 存储初始簇 id 到结果簇 id 的映射
3: for each p in plist do
4:     matrix[p.id1][p.id2] += p.isConn
5: end for
6: 构造访问标识数组 visited = [false]n
7: mark = 1
8: for i = 1 to n do
9:     if visited[i] 为真 then
10:        不再继续访问
11:    end if
12:    visited[i] = true
13:    mark += 1
14:    resmap[plist[i].cluId] = mark
15:    执行 dfs(i, mark, visited, plist, matrix, resmap)
16: end for
17: for i = 1 to n do
18:     if x[i].cluId in resmap then
19:         x[i].resCId ← resmap[x[i].cluId]
20:     end if
21: end for
```

---

#### 算法 4.10 深度优先遍历 (DFS)

输入: 起始位置 idx, 归属簇标识 mark, 访问标识数组 visited, 元素数组 plist, 二维矩阵 matrix, 映射关系 resmap

输出: 修改后的输入值

```
1: for i = 1 to n do
2:     if visited[i] 为真 then
3:         该元素已经被划分好, 不再继续
4:     end if
5:     if matrix[idx][i] > 0 then
6:         resmap[plist[i].cluId] = mark
7:         visited[i]=true
8:         dfs(i, mark, visited, plist, matrix, resmap)
9:     end if
10: end for
```

---

首先，在算法4.9中构造大小为  $n \times n$  的全零邻接矩阵，来存储邻接关系。根据 plist 信息来还原相连信息，若相邻则邻接矩阵中，对应位置的值大于 0。然后，初始化一个数组 visited 来标记每个数据是否已经被划分到最终所属簇中，若已经被划分，则不再继续处理，这里用 mark 表示最终结果的簇编号，自 1 开始累加。随后，开始遍历，若元素  $i$  已被划分，则跳出当前循环，开始下一次。若未被划分，则更新 mark 值自增 1，然后开始构建新簇，从当前元素  $i$  开始执行深度优先遍历（DFS）找到所有与  $i$  相连的元素纳入到新簇中。DFS 的过程如算法4.10所示，边递归边记录映射关系，将相连的簇进行合并。最后，根据 resmap 映射关系，更新所有元素的最终簇编号 resCId，这一步操作可以将属于同一个簇的所有临时簇中的数据点的 cluId 修改一致。

深度优先算法在这里应用的目的是将相连的临时簇合并成一个整体，从而获取最终的结果。在遍历的过程中，若判断相连，则建立该数据 cluId 到最终结果 mark 的映射关系。

上述过程由用户独立运行，这里算法的复杂度最大为  $O(n^2)$ ，但是可以通过一些改进方法将深度优先遍历算法的复杂度降低为  $O(n)$ ，对用户来说，执行上述算法的开销较小。

## 4.6 改进的隐私保护 DBSCAN

本节将在上一节的基础上详细阐述一种改进的隐私保护 DBSCAN 方案，该方案能够在数据集被打乱的情况下，使满足条件可以被划分到多个簇的数据点稳定分类到最近的簇中，只需要在方案一的基础上引入一些比较和计算操作，即可提升聚类质量。

在前一章4.5可视化示例的基础上，我们简述方案一和方案二的主要差异。初始数据集如图4.12所示，然后开始遍历，仅处理非核心对象，这里没有噪声点，因此将所有边界对象的簇标识 cluId 都改成与最近的核心对象一致，核心对象不做任何修改，结果如图4.13所示，所有边界对象均修改了簇标识（深绿色点）。对于噪声点，是否修改 cluId 都对最终结果没有任何影响。

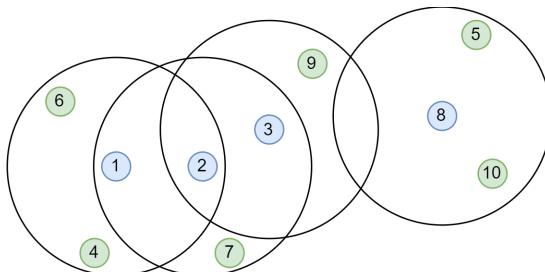


图 4.12 初始数据集

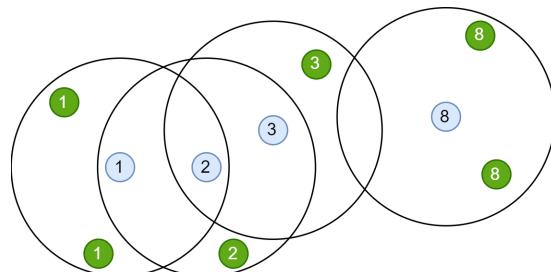


图 4.13 处理边界对象

经过上述处理，使得边界对象的划分结果稳定，与最近的核心对象所属簇保持一致。同时减少了聚类算法的计算复杂度，无需再对边界对象的簇标识进行处理，只要记录相邻关系，最后还原即可。

本方案包含三个步骤，初始化、聚类以及还原结果。还原结果的过程与章节4.5.3所述方式一致，这里不多做说明。主要区别在于初始化和聚类过程，下面将着重介绍这两个部分。

#### 4.6.1 初始化

不同于方案一的初始化过程，这里需要对每个非核心对象求最近的核心对象，并修改 `cluId`，与最近的核心对象保持一致。上述操作需要运行安全极值协议，计算和通信开销相较于方案一略微增加。

具体流程如算法4.11所示：

---

##### 算法 4.11 初始化

---

输入：数组 `sharedList` 简写为 `x`

输出：距离关系 `u`

```

1: for  $i = 1$  to  $n$  do
2:     for  $j = 1$  to  $n$  do
3:          $dist[j] \leftarrow \text{DIST}(x[i].data, x[j].data)$ 
4:     end for
5:      $u[i] \leftarrow \text{SC}([\epsilon_n], d)$ 
6:      $s[i] \leftarrow \sum \text{sum}(u[j])$  for  $j \in [n]$ 
7: end for
8:  $r \leftarrow \text{SC}(u, [minPts_n])$ 
9:  $x[i].isCore \leftarrow r[i]$  for  $i \in [n]$ 
10: for  $i = 1$  to  $n$  do
11:      $m \leftarrow 0$ 
12:     for  $j = 1$  to  $n$  do
13:          $t1[j] \leftarrow \text{MUL}(x[j].isCore, u[i][j])$ 
14:          $t2[j] \leftarrow 1 - t1[j]$ 
15:          $m = t1[j] + m - \text{MUL}(m, t1[j])$ 
16:     end for
17:      $r1 \leftarrow \text{MUL}(x[i], t1) + \text{MUL}([max_n], t2)$ 
18:      $r2 \leftarrow \text{SMin}(r1)$ 
19:      $nId \leftarrow \text{sum}(\text{MUL}(r2[j], x[j].cluId))$  for  $j \in [n]$ 
20:      $x[i].CluId \leftarrow \text{MUL}(nId, m) + \text{MUL}(1 - m, x[i].cluId)$ 
21: end for

```

---

首先，计算每个元素到其他元素的距离，同时获取邻域关系存储在  $u$  中，并且记录每个元素邻域范围内包含点的个数  $s[j], j \in [n]$ 。然后，并行比较  $s$  与  $\text{minPts}$  的大小关系，并更新每个元素的 `isCore` 属性。计算完 `isCore` 信息后，开始本节算法的核心部分，即更新非核心对象的 `cluId`。预设标识  $m$  来判断是否对元素更新 `cluId`，判断依据是若邻域范围内存在核心点，则可以修改。若  $x[j]$  为核心点且与  $x[i]$  相邻，则  $t1[j]$  为 1，否则  $t2[j]$  为 1。根据该值计算出  $r1$ ，过滤出所有相邻核心对象的距离，无关距离设为最大值，避免影响求解距离最近的核心对象。最后，通过逐个相乘后累加来求解最近相邻核心对象的 `cluId`，并根据前述  $m$  值来更新 `cluId`。

值得指出的是，对于核心对象， $m = 1$  需要修改 `cluId`，但是最近相邻核心点为自身，因此 `cluId` 未改变。对于边界对象，依据算法将修改 `cluId` 与最近相邻核心对象保持一致。对于噪声， $m = 0$  因此 `cluId` 不会发生改变。

#### 4.6.2 核心对象聚类

在初始化的过程中，所有边界元素均已被划分到最近相邻核心点所属簇中。因此相较于算法4.8，本方案的聚类过程更加简单便捷，在算法4.12中只要记录相连的临时簇信息即可，当且仅当  $x[i]$  与  $x[j]$  均为核心对象并且距离小于  $\epsilon$ ，则认为二者相连。

---

#### 算法 4.12 聚类

---

输入：数组 `sharedList` 简写为 `x`，相邻关系  $u_{ij}, i, j \in [n]$

输出：相连关系 `plist`

```

1: for  $i = 1$  to  $n$  do
2:     for  $j = 1$  to  $n$  do
3:          $r \leftarrow \text{MUL}(x[i].isCore, x[j].isCore, u_{ij})$ 
4:         add new item to plist( $x[i].cluId$ ,  $x[j].cluId$ ,  $r$ )
5:     end for
6: end for

```

---

#### 4.7 基于 DBSCAN 的隐私保护层次聚类

方案一4.5和方案二4.6均是在用户给出  $\epsilon$  和  $\text{minPts}$  值的基础上，实现隐私保护 DBSCAN。但是现实情况中，多个独立用户在不知道彼此拥有的具体数据信息时，难以根据数据分布特点给出适合的参数，从而导致聚类效果较差。

为了解决上述问题，尽可能减少用户对聚类的影响，根据论文[38]中的思想设计了一种基于 DBSCAN 的隐私保护层次聚类方案。前述方案的  $\epsilon$  均为固定值，没

有考虑到数据集中的数据密度可能不同这一情况，导致在个别数据集上划分效果不佳的问题。本方案考虑数据集密度不同的特点，设置不同的  $\epsilon$  值进行聚类以提升聚类效果。

整个方案主要可以分为三个阶段：

- **获取  $\epsilon$  值：**通过计算数据之间的距离，绘制  $k$  线图并固定间隔取值来计算系列  $\epsilon$  值。
- **计算初始簇：**根据上述获取的  $\epsilon$  值，升序取  $\epsilon$  并据此反复执行 DBSCAN 过程，已被划分的点不再参与后续过程，直到遍历完所有  $\epsilon$  的值。
- **合并初始簇：**在初始簇的基础上，不断将相距最近的初始簇进行合并，直到簇个数满足要求  $k$ 。

#### 4.7.1 获取 $\epsilon$ 值

本小节将详细阐述如何通过  $k$  线图来获取所有聚类所需的  $\epsilon$  值， $k$  线图的绘制方式为，首先计算每个点到其最近的第  $k$  个点的距离，然后按照升序方式排列，最后根据这些排序值绘图。

由于现实数据集的复杂性，目前没有办法准确找到每个数据集具体包含多少种不同的密度，DBHC<sup>[38]</sup> 经过分析，选择在  $k$  线图中固定间隔来选取  $\epsilon$  的值，数据集的大小为  $n$ ，则间隔为  $\sqrt{n}$ 。因此 DBSCAN 的执行次数也为  $\sqrt{n}$  次。在算法4.13中，将详细说明计算流程。

首先，计算元素与元素之间的距离，由于距离是相对的  $d[i][j] = d[j][i]$ ，因此缩减了一半的计算量。然后，取  $k = 2$ ，计算每个点第二近的距离来绘制  $k$  线图。 $d$  为二维矩阵， $d[i]$  存储元素  $i$  所有距离值，首先求距离最小值，然后通过乘法和加法使得  $d[i]$  中的最小值化为最大值，在此之后，再求解一次最小值，即可获得第  $k$  小的值。通过乘法使得数组中只保留第  $k$  小的值，其他值均为 0，然后通过计算数组累加值来获取最终结果存储到  $d2$  中。其中，借助安全排序算法4.4 来对  $d2$  数组进行升序排列，结果存放到  $sdist$  中。

值得一提的是，这里的固定间隙值  $\sqrt{n}$  在明文上进行计算，因为数据集的大小  $n$  对于云平台是已知的。最后，从  $sdist$  中按照固定间隙  $\sqrt{n}$  取  $\epsilon$  的值存储到  $elist$  中。

#### 4.7.2 计算初始簇

本节将详细介绍，如何在获取了所有  $\epsilon$  值后对不同密度的数据进行聚类。算法的基本逻辑是，对每个  $\epsilon$  值（按升序）执行一次 DBSCAN 算法，总共  $|E|$  次。 $\epsilon$  值较小时，无法覆盖所有数据，因此剩余部分数据没有被划分，为了使剩余数据被划分的同时，已经划分的数据不受影响，将这些数据移出数据集。经过上述过

---

**算法 4.13 获取  $\epsilon$  值**

---

输入: 秘密共享的元素数组  $\{x_1, \dots, x_n\}$   
 输出: 秘密共享的  $\epsilon$  值  $elist$

```

1: for  $i = 1$  to  $n$  do
2:   for  $j = i + 1$  to  $n$  do
3:      $d[i][j] \leftarrow \text{DIST}(x[i].data, x[j].data)$ 
4:      $d[j][i] \leftarrow d[i][j]$ 
5:   end for
6: end for
7: for  $i = 1$  to  $n$  do
8:    $r \leftarrow \text{SMin}(d[i])$ 
9:    $t[j] \leftarrow 1 - r[j]$  for  $j \in [n]$ 
10:   $d[i] \leftarrow \text{MUL}(d[i], t) + \text{MUL}([max]_n, r)$ 
11:   $r \leftarrow \text{SMin}(d[i])$ 
12:   $d2[i] \leftarrow \text{sum}(\text{MUL}(d[i], r))$ 
13: end for
14:  $sdist \leftarrow \text{SSORT}(d2)$ 
15:  $j \leftarrow \sqrt{n}$ 
16: for  $i = j$  to  $n$  do
17:   add  $sdist[i]$  to  $elist$ 
18:    $i \leftarrow i + j$ 
19: end for

```

---

程会产生较多簇，在下一节，将进行合并操作以获取预期的聚类划分结果。在算法4.14中，将详细说明密文方案中算法的实现细节。

根据算法4.12排序所得  $\epsilon$  值为升序排列，因此这里逐个遍历  $elist$  值，针对每个  $\epsilon$  值进行聚类即可。首先，进行数据处理工作，在每次聚类前将数据的 `isCore` 信息清空置 0。然后计算  $x[i]$  与  $x[j]$  之间的距离， $d[i][j]$  在获取  $\epsilon$  值时已经算出，无需重复计算。只有当  $x[i]$  与  $x[j]$  均没有被划分 ( $x[i].isMark = 0, x[j].isMark = 0$ ) 时，距离为  $d[i][j]$ ，否则为数值范围内最大值  $max$ 。然后，比较距离和  $\epsilon$  大小关系，并记录到  $u$  中，同时累加邻域范围内的点记录到  $nei$  中。最后，比较邻域范围内点个数与 `minPts` 的大小关系，并据此更新  $x[i]$  的 `isCore` 属性。最后根据上述信息，执行 DBSCAN 聚类。

在算法4.15中将详细解释执行聚类的过程。首先更新数据的 `cluId`，只有在  $x[i]$  为未被标记的核心点， $x[j]$  未被标记，且二者相邻  $u[i][j] = 1$  的情况下，才将  $x[j]$  的 `cluId` 修改为与  $x[i]$  一致。`m[j]` 记录是否更新  $x[j]$  的 `isMark` 信息，若  $x[j]$  的 `cluId` 被修改过，则认为该数据已被被划分。然后，记录核心点相连信息，在上述判断

---

**算法 4.14 计算初始簇**

---

输入: 秘密共享的元素数组  $\{x_1, \dots, x_n\}$ , 秘密共享的  $\epsilon$  值  $elist$

输出: 初始簇列表  $cluList$

```

1: for each  $\epsilon$  in  $elist$  do
2:   for  $i = 1$  to  $n$  do
3:      $x[i].isCore \leftarrow 0$ 
4:     for  $j = 1$  to  $n$  do
5:        $t \leftarrow (1 - x[j].isMark) * (1 - x[i].isMark)$ 
6:        $d'[j] \leftarrow \text{MUL}(t, d[i][j]) + \text{MUL}(1 - t, max)$ 
7:     end for
8:      $u[i] \leftarrow \text{SC}(\epsilon, d')$ 
9:      $nei[i] \leftarrow \text{sum}(u[i])$ 
10:    end for
11:     $r \leftarrow \text{SC}(nei, minPts)$ 
12:     $x[i].isCore \leftarrow r[i]$  for  $i \in [n]$ 
13:    执行算法4.15进行聚类
14: end for

```

---

**算法 4.15 聚类**

---

输入: 秘密共享的元素数组  $\{x_1, \dots, x_n\}$

输出: 更新信息后的元素数组

```

1: for  $i = 1$  to  $n$  do
2:   for  $j = 1$  to  $n$  do
3:      $t1 \leftarrow \text{MUL}(x[i].isCore, 1 - x[i].isMark)$ 
4:      $t2 \leftarrow \text{MUL}(u[i][j], 1 - x[j].isMark)$ 
5:      $t \leftarrow \text{MUL}(t1, t2)$ 
6:      $x[j].cluId \leftarrow \text{MUL}(t, x[i].cluId) + \text{MUL}(1 - t, x[j].cluId)$ 
7:      $m[j] \leftarrow m[j] + t - \text{MUL}(m[j], t)$ 
8:      $r \leftarrow \text{MUL}(t, x[j].isCore)$ 
9:     记录新的元素 ( $x[i].cluId$ ,  $x[j].isCore$ ,  $r$ ) 到  $plist$  中
10:    end for
11: end for
12: for  $i = 1$  to  $n$  do
13:    $x[i].isMark \leftarrow m[i] + x[i].isMark - \text{MUL}(m[i], x[i].isMark)$ 
14: end for

```

---

是否修改  $cluId$  逻辑的基础上, 加入  $x[j]$  为重心点的限制即可, 随后构造新元素并将结果计入  $plist$  中。最后, 更新所有元素的  $isMark$  属性, 若元素在之前迭代过程中已被聚类 ( $isMark=1$ ) 或者在本轮迭代中被划分 ( $m=1$ ) 则  $isMark$  属性标记为

---

1。isMark 的作用是让本轮以及之前被划分好的数据不再参与后续聚类。

### 4.7.3 合并初始簇

本节的主要工作由用户独立完成。在算法4.16中，给出详细描述。首先根据算法4.9将临时簇进行合并还原成初始簇，并更新 plist 每个数据的 resCId，即初始簇标识。然后对 plist 进行处理获取每个初始簇包含点的个数，以及簇中心。反复求解集合  $C$  中最近的两个簇  $c_a$  和  $c_b$  然后将数据进行合并，更新参数直到簇的个数为  $k$  个。

---

#### 算法 4.16 合并初始簇

---

输入：秘密共享的聚类结果  $plist$ ，期望簇个数  $k$

输出： $k$  个簇信息

- 1: 根据4.9还原聚类结果得到获取 resCId 的 plist
  - 2: 根据 plist 获取初始簇集合  $C$ ，每个簇包含的数据个数 num 和中心 cen
  - 3: 开始合并初始簇
  - 4: while  $|C| > k$  do
  - 5:     计算簇与簇之间的距离  $d$
  - 6:     求解距离最近的两个簇  $c_a$  和  $c_b$  进行合并
  - 7:     更新簇参数 num 和 cen
  - 8: end while
- 

## 4.8 实验分析

在本节中，将分别给出隐私保护 DBSCAN（方案一）、改进的隐私保护 DBSCAN（方案二）以及基于 DBSCAN 的隐私保护层次聚类（方案三）的理论分析和实验结果。本文将会在多个数据集上进行实验，比较聚类的质量和基本开销，此外实验中会和其他隐私保护 DBSCAN 相关研究进行对比来说明本文方案的实用性。

### 4.8.1 理论分析

本小节，对上述三个方案给出对应的复杂度分析，方案主要有安全计算子协议组成，例如安全比较（SC）、安全排序（SeSort）、安全乘法（MUL）等。其中安全排序和安全极值协议由安全比较和安全乘法组成。因此将方案进行拆分，可以分为安全比较协议和安全乘法协议两部分。

安全比较协议基于百万富翁协议<sup>[33]</sup>，计算与通信均比较复杂。安全乘法协议根据 Beaver 乘法三元组<sup>[74]</sup>实现，其中预计算的内容均由用户或者第三方机构离线产生，不产生聚类开销，与安全比较协议相比，运算开销较小。因此在下面的

分析过程中，以安全比较协议的执行次数来衡量方案的复杂度，省略乘法次数的分析。

具体的比较如下图4.3所示：

表 4.3 方案复杂度理论分析

方案	方案一	方案二	方案三
比较次数	$n + 1$	$2n + 1$	$3n + \sqrt{n}(n + 1)$

安全比较协议3.4.2一次可以比较多对数据，在安全极值协议中3.4.3中，给出了两种实现方式，分别是小数据量上的最值协议和大数据量上的树状比较，前者通过增加比较数据对减少比较次数， $n$ 个数据进行一轮比较（ $n^2$ 对数据），后者通过树状比较结构减少比较对数，进行  $\log n$  轮比较。在本节的分析中，选择基于小数据量上的比较，因此运行一次安全极值协议进行一次比较。

从表格中可以看到，方案二相比方案一复杂度没有量级上的差异，因此在后面的实验中也可以看到开销的差异并不是非常显著。但是方案三的复杂度则提升了一个量级，因此开销大大增加。具体的耗时对比，会在下面的具体实验中给出。

#### 4.8.2 实验设置

**服务器配置：**方案实现基于 c++ 和论文<sup>[33]</sup> 提供的百万富翁协议，位宽设置为 64 位。在一台主机上模拟两个云平台进行交互，机器的配置为 Inter Core i7-7700HQ 2.80 GHz CPU 和 16 GB RAM。

**场景：**隐私保护 DBSCAN 方案既可以应用于外包计算也可以应用于两方安全计算的场景。此外，聚类的数据可以采取任意划分形式，例如水平划分、垂直划分或者是混合划分。在以下实验中，基于外包计算的场景，用户将数据秘密共享后分别发送给两个不共谋的云服务器。值得一提的是，外包计算场景下，参与聚类的用户数量不受限制，本文假设单个用户上传所有数据。方案可以直接应用到用户更多的场景中，只要数据、参数和簇个数和实验保持一致。

**数据集：**为了更好地评估聚类方案，在多个不同的数据集上进行了实验。其中两个是为了和论文<sup>[29]</sup> 对比方案的效率和实用性。

- Lsun：图4.14(a)中的 Lsun 数据集<sup>[88]</sup> 在论文<sup>[29~30]</sup> 分别用于评估隐私保护 K-means 和隐私保护 DBSCAN 方案的性能。它的正确划分结果中包含 3 个簇，分别拥有不同的方差和簇距离。三个簇分别包含 200、100 和 100 个元素
- S1：图4.14(b)中的 S1<sup>[82]</sup> 数据集是一个人工合成的二维数据集，包含 5000 个样本，可以划分为 15 个不同的球形高斯簇，分别包含 300 到 350 个元素。其中簇之间有 9% 的重叠部分。该数据集在论文<sup>[28~29, 57]</sup> 中被用于分析聚类效

果。

- Aggregation：图4.14(c)中的 Aggregation 数据集<sup>[89]</sup>为人工合成的二维数据集，包含 788 个数据，7 个分类。在后续实验中，将通过该数据集分析 K-means 和 DBSCAN 聚类结果的差异。

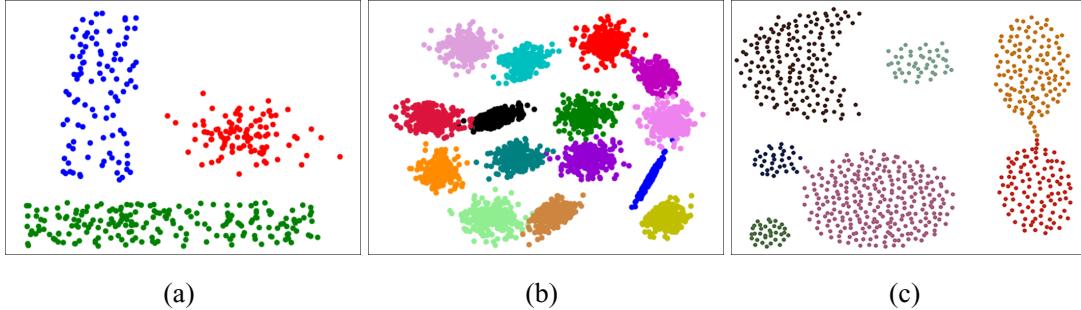


图 4.14 Lsun、S1 和 Aggregation 数据集正确划分结果

**编码：**Lsun 数据集由有理数组成。参考前述工作<sup>[31]</sup>，将数据放大  $10^6$  全部转化为整数后执行协议。相应的，Aggregation 数据集放大  $10^2$  倍。S1 数据集由整数构成不进行放缩。

**参数：**方案一与方案二均需要人工设定  $\epsilon$  和 minPts 的值，参考论文<sup>[29]</sup> 中的设置，并根据编码过程中的放缩值对参数进行修改，同时，由于算法中计算欧式距离去掉了根号，因此也对  $\epsilon$  值做出了相应的处理。具体信息在4.4中给出。方案三不需要提前根据数据特点分析给出  $\epsilon$  和 minPts 的值，在本节的实验中，令  $k=3$ 。

表 4.4 数据集具体信息

数据集	维度	簇个数	数据量	$\epsilon$	minPts
Lsun	2	3	400	$2 \times 10^{11}$	4
S1	2	15	5000	$2.25 \times 10^9$	50
Aggregation	2	7	788	$3.5 \times 10^4$	3

### 4.8.3 聚类质量

本小节，将根据调整兰德系数（ARI）在 Lsun、S1 和 Aggregation 数据集上进行实验来比较上述几个方案与论文<sup>[29]</sup> 中提出的 ppDBSCAN 方案、K-means 算法的聚类质量。

由于聚类是一种无监督机器学习算法，没有办法计算准确率，但是 Lsun、S1 与 Aggregation 为人工合成数据集，主要用于对机器学习算法进行基准测试，所以

正确的分类结果是已知的。因此，能够基于这些数据集对聚类算法用 ARI 指标来衡量聚类质量，这是一个广泛应用的外部聚类质量指标<sup>[90~91]</sup>，详细的计算方法在小节4.2.1给出。

为了进行公平合理的比较，对每个方案都选择了最佳的参数设置。对于 K-means，选择 sklearn 库中的标准 K-means 聚类方法<sup>[92]</sup> 来获取结果，提供准确的簇个数，同时为了使算法更快收敛和获取高质量的 K-means 聚类结果，选择 K-means++ 参数来优化初始簇中心的选择（random 为随机选择初始簇中心）。对于 DBSCAN，按照表格4.4来设置参数运行所有方案。论文<sup>[29]</sup> 没有给出基于 Aggregation 的实验结果，因此这里不进行对比分析。

表 4.5 聚类质量评估

数据集	K-means	ppDBSCAN <sup>[29]</sup>	方案一	方案二	方案三
Lsun	0.4386	1.0	1.0	1.0	0.9973
S1	0.9254	0.9757	0.9766	0.9778	0.9371
Aggregation	0.7621	-	0.8089	0.8089	0.7784

在表格4.5中，可以看到基于 ARI 的聚类质量评估结果。

- 与 K-means 比较：在所有数据集上，本文所述三个方案的划分结果均优于 K-means 聚类的划分结果。特别在 Lsun 这种非球形数据集上，K-means 的聚类质量显著落后于 DBSCAN。S1 为球形数据集，因此 K-means 方案的划分结果的 ARI 值与方案一二三相近。Aggregation 数据集同时包含球形和非球形的数据集，因此方案一二三比 K-means 表现更好。值得一提的是，K-means 算法的聚类效果受到初始簇中心的影响较大，若随机生成初始簇中心，ARI 指标的计算值会更小。
- 与 ppDBSCAN 比较：方案一的聚类质量与 ppDBSCAN 近似，因为方案一基于传统 DBSCAN 算法，输出结果与明文 DBSCAN 一致。但是方案一在 S1 数据集上表现更好，原因是 S1 数据集有部分重叠，论文<sup>[29]</sup> 为了简化密文方案复杂度，没有完全遵循明文 DBSCAN 的思路，设置了最大迭代深度来限制次数，该参数主要受到数据分布影响。
- 方案一二三比较：在 Lsun 数据集上，方案一二结果一致，均能够正确的划分簇，但是方案三对于极个别数据划分产生错误。在 S1 这种包含重叠部分的数据集上，方案 2 的表现最好，原因在于该方案能够获取稳定的数据划分结果，使得在数据集包含较多重叠部分时，让数据被划分到最近的簇中，提升聚类质量。方案三在包含不同密度数据的数据集上表现会更好。

**可视化比较：**图4.15中，给出了聚类的可视化结果，行依次对应 Lsun、S1 和 Aggregation 数据集，列依次为方案一、二、三和明文 K-means 的聚类结果。其中黑色为 DBSCAN 结果中的噪声点。可以看到方案一二均划分的较好，但是方案三在簇内部产生较多噪声点，结果误差较大。K-means 在非球形数据集上容易将单一簇划分为两个，效果较差 (4.15.d/4.15.l)。对比图4.15.e 和4.15.f 可以看到在不同簇之间的重叠部分划分结果不一样，方案二打乱数据后，重复运行结果保持一致，方案一则会产生变化。

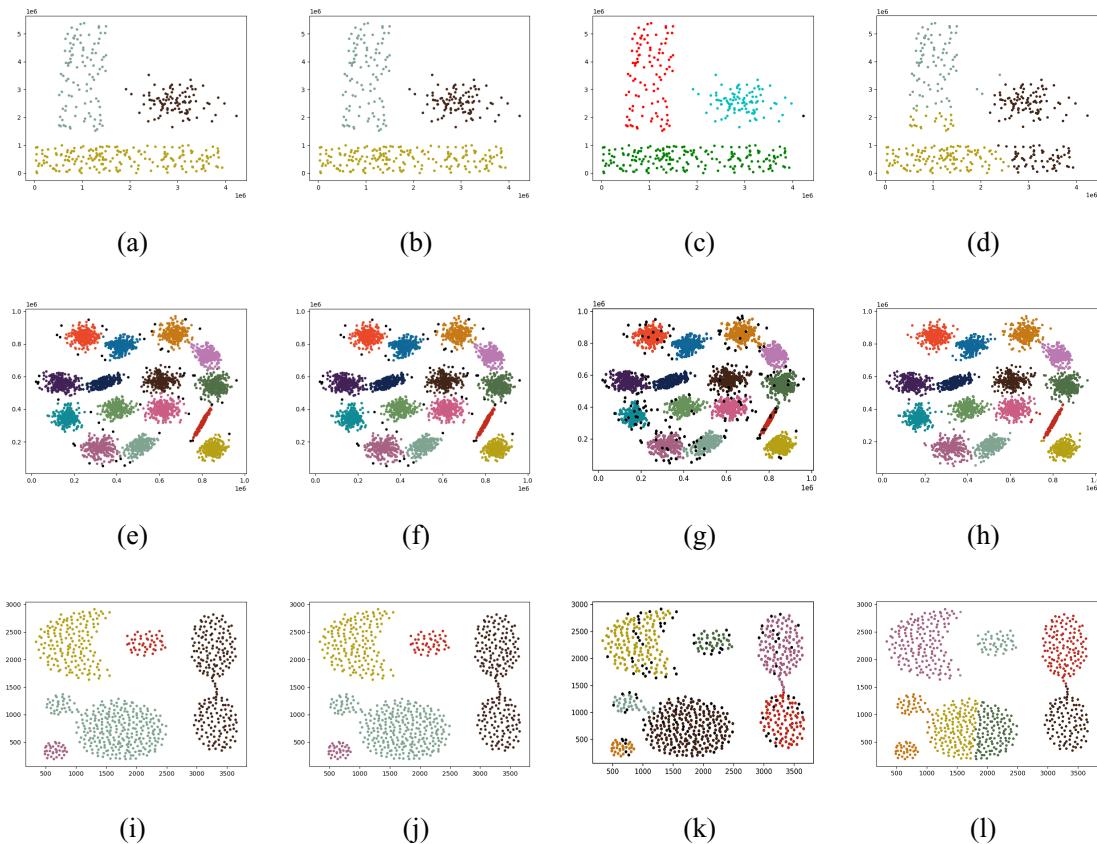


图 4.15 聚类结果可视化比较

#### 4.8.4 运行开销与分析

本小节，将详细讨论提出的系列方案与前沿隐私保护 K-means 方案<sup>[57]</sup> 以及隐私保护 DBSCAN 方案<sup>[29]</sup> 的耗时对比。上述两个方案均较好的平衡了效率与安全，论文<sup>[57]</sup>在多方计算场景下提出了一种完全安全的高效隐私保护 K-means 方案，但是受限于 K-means 聚类本身的缺陷，聚类的效果较差，适用场景较少。而论文<sup>[29]</sup>所提出的方案虽然高效但是方案可能会存在轻微的数据泄露问题。其他隐私保护 DBSCAN 相关研究均存在较大的数据安全问题，同时没有给出具体的实现和详细

的实验分析，因此这里不进行对比分析。

在表格4.6中，给出了上述方案在 Lsun、S1 和 Aggregation 数据集上的用时。其中论文<sup>[57]</sup> 实验机器配置为 Intel Core i7 2.6 GHz 12 GB RAM，模仿网络传输 10Gbit/s 和 0.02ms 的往返时间。而论文<sup>[29]</sup> 的实验机器配置则是 Intel Core i9-7960X CPUs 2.8 GHz 128 GB RAM，两个独立的服务器在 10Gbit/s 和 0.02ms 往返时间的网络上传输。两个方案的实验配置均与本实验的配置近似。

表 4.6 耗时对比

隐私保护方案	隐私保护 K-means 聚类			隐私保护 DBSCAN 方案		
	数据集	Mohassel 等人 <sup>[57]</sup>	Bozdemir 等人 <sup>[29]</sup>	方案一	方案二	方案三
Lsun	22.21s	420.72s	4.12s	27.90s	2508.27s	-
S1	1472.6s	620,912.70s	301.12s	1002.55s	-	-
Aggregation	-	-	103.77s	639.10s	19620.2s	-

从图中，可以看到在所有数据集上，隐私保护 DBSCAN 方案（方案一）的耗时均最短，相较于论文<sup>[57]</sup> 和论文<sup>[29]</sup>（ppDBSCAN）分别有近 5 倍和 100 倍的提升，在耗时上远远优于当下的前沿方案。而改进的隐私保护 DBSCAN 方案（方案二）的耗时在 Lsun 数据集上则略逊色于论文<sup>[57]</sup>，同时领先 ppDBSCAN 约 14 倍，方案二为了稳定聚类划分的结果引入了更多比较操作，因此耗时相对于方案一略有增加，但同时也提升了聚类质量。方案三在耗时上没有显著优势，但是该算法的研究意义在于无需提供人工设定的参数  $\epsilon$  和 minPts，减少了前置数据分析的工作量，使得算法的运行更加独立。

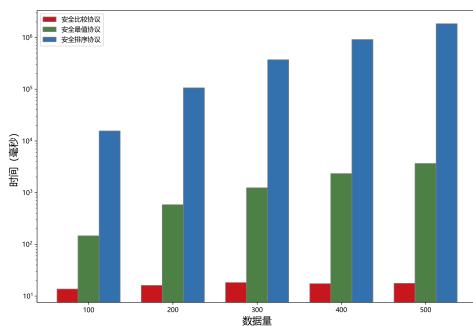


图 4.16 耗时对比

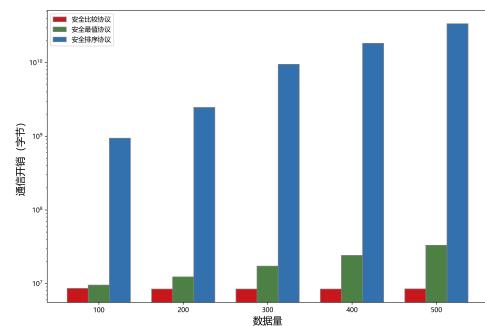


图 4.17 通信开销对比

接下来，根据图4.17和4.16来对安全比较协议、安全极值协议以及安全排序协议从耗时和通信开销两个角度进行对比分析。从图4.16可以看到，三个协议随着

数据量的增加耗时均有所增加，其中安全比较协议的耗时远远优于其他两个协议，主要原因是安全极值协议和安全排序协议由安全比较协议和其他操作组合而成，其中比较操作是性能瓶颈。

为了保证协议的顺利运行，设置数据传输的默认大小为 $2^{20}$ 。在此基础上，可以看到安全排序协议的通信开销远超其他两个协议，原因在于对 $n$ 个数据进行安全排序，需要执行 $n$ 次安全极值协议，而每一次执行安全极值协议都需要进行一次安全比较，并行比较 $n(n - 1)$ 对数据。由上述关系可以反映图中的变化规律，数据量的变化对安全比较协议所耗费通信开销的增长速度影响较小，其他协议则不然。

综上所述，安全排序协议无论是从耗时还是通信角度，开销都非常大。在大数据上表现尤为明显，因此有其局限性。而安全比较协议和安全极值协议则表现较好，能够迁移到数量较大的场景中。

最后简要分析，用户承担的还原聚类结果的工作所需计算开销。选择在数据量范围为[1000, 4000]的人造数据集上执行方案一进行实验。方案一与方案二的还原算法一致，方案三在原来算法的基础上，增加了合并初始簇的过程，复杂度为 $O(m)$ ， $m$ 为初始簇个数。计算比较简单，这里不进行单独分析。

如图4.18所示，可以看到还原结果所需时间显著低于聚类所需时间，用户承担计算量的复杂度在 $O(n)$ 量级，因此认为隐私保护DBSCAN系列方案对用户带来的计算负担较小，能够帮助用户处理在保障数据安全的前提下，获取聚类结果。

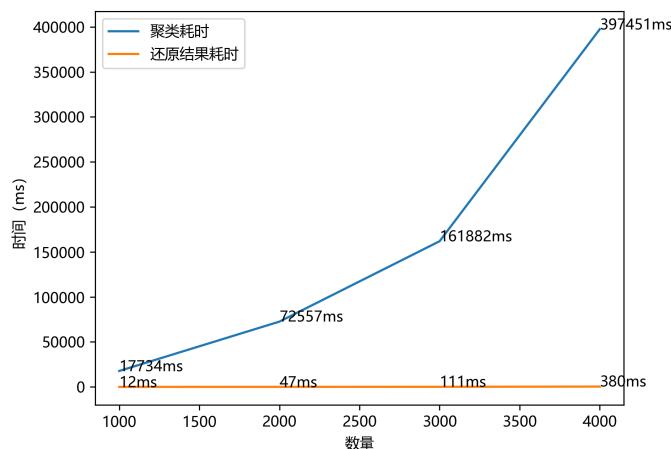


图 4.18 聚类与还原结果耗时比较

## 4.9 本章小结

现有的隐私保护 DBSCAN 相关研究相对隐私保护 K-means 聚类研究较少，许多研究不够完善，没有给出代码实现和实验分析。最近的前沿方案<sup>boz</sup> 则存在数据泄漏风险，因此需要设计一个兼顾效率与安全的隐私保护 DBSCAN 聚类方案。

本章，从三个角度设计了系列安全高效的隐私保护 DBSCAN 方案。首先，基于传统 DBSCAN 的思想，对算法进行深入分析和改造设计了一种全新的密文聚类方式，聚类时获取临时簇并记录相连关系，最后根据深度优先遍历算法还原结果，该算法开销远低于<sup>[29]</sup>。其次，为了解决 DBSCAN 中聚类划分结果不稳定的问题，提出了一种改进的隐私保护 DBSCAN 方案，该算法能够获取稳定的聚类结果以及提升聚类质量。最后，为了减少对人工设置参数的依赖，提出了一个基于 DBSCAN 的隐私保护层次聚类方案，该方案无需根据数据分布设置关键参数，同时能够聚类不同密度的数据。

虽然本节的方案均兼顾了安全与高效，方案一和方案二具备良好的扩展性和可用性，在减少计算开销和提升聚类质量方面取得长足进展。但是方案三因其复杂性耗时较多，无法推广至大数据集，同时在某些数据集上聚类效果有劣势。仍需进一步改进，从多方面进行提升。

## 第五章 总结与展望

### 5.1 工作总结

云环境下的聚类应用赋能大数据挖掘与分析，为人类生活与工作带来了诸多便利，解放了资源受限用户的生产力，让用户可以关注分析与发现。然而，伴随而来的数据安全问题也引发了广泛的关注和重视。数据用户在享受便捷服务的同时，上传的隐私数据与聚类获取的划分结果可能会被泄漏。过去几年来，因为数据泄漏而引发的问题不胜枚举，用户对于提供云服务的公司信心受损，公司蒙受巨大损失。为此，本文针对隐私保护聚类方案中存在的安全问题，效率问题以及准确性问题展开研究。本文的主要的研究内容和贡献包括以下几个方面：

- 提出了一种基于 Kd-tree 的隐私保护外包 K-means 聚类方案。针对目前隐私保护 K-means 聚类中存在的无法兼顾安全和高效的问题，设计了一种全新的解决方案。方案基于秘密共享技术，在不共谋的双云服务器协作的场景下，设计了一系列可扩展可迁移的安全计算协议。通过构造 kd-tree 数据结构加速聚类过程，减少了冗余计算，加速算法收敛。方案能够迁移到大型数据集上，在即将收敛时，效率远远超过传统 K-means 方案。在详细的安全分析和理论分析的基础上，通过实验验证，我们认为方案不仅能够在兼顾效率和安全的情况下完成聚类，同时还保证了结果的准确性。
- 提出了隐私保护 DBSCAN 系列方案。针对隐私保护 DBSCAN 相关研究中存在的效率较低、划分结果不稳定以及依赖人工参数设置等问题分别提出了对应的解决方案。在双云服务器交互的场景下，多用户秘密共享数据后分别上传给云平台，通过运行系列安全计算协议，获取最终的聚类结果并还原。所述方案不仅可以应用于外包场景还能够迁移到多方计算的场景中，具备可扩展性。经过全面的实验分析，证明方案一兼顾效率与安全性，方案二提升了在保证高效与安全的同时提升了聚类质量，方案三能够在特定数据集上发挥较好的效果。

### 5.2 研究展望

目前云环境下隐私保护聚类方案已有较多相关研究，但是大多聚类方案只适用于数据量较小的情况，难以应用于需要进行实时分析的大数据场景。由于自身能力有限与时间不足，对相关问题的研究不够深入。因此在本文研究的基础上，接下来可以在如下两个方面进一步研究探索：

本文设计的隐私保护聚类方案虽然能够在兼顾效率与安全的情况下，在中小

规模数据集上获取高质量的聚类结果，但是在海量数据上聚类耗时较长，无法满足实际应用的需求，因此存在较大局限性。由于数据挖掘分析本身的特点，允许在聚类精度可接受的情况下对算法进行修改，获取近似结果以减少计算量提升效率、扩展性和易用性。同时，随着云计算和大数据的进一步发展，如何高效的对海量数据进行隐私保护挖掘分析具有较大的现实意义。

目前关于云环境下的隐私保护聚类研究大多基于 K-means 算法、其次是 DBSCAN 算法和层次聚类算法。聚类算法还包含很多其他分支，例如 BIRCH 算法聚类速度快能够识别噪声点、GMM 算法获取划分到每个类的概率以及 Affinity Propagation 算法对于初始值不敏感等等。上述聚类算法分支都有其适用的场景和优缺点，但是相关隐私保护方案则研究较少。对于这些算法设计相关隐私保护实现方案具有应用和现实意义，能够进一步提升隐私保护数据挖掘与分析的能力。

## 致 谢

论文进入尾声，我的研究生涯也将结束。从为科大而准备考研起，我就一直过着充实丰富又踏实的生活，在这个过程中，我遇到了厉害又负责的老师，遇到了友好又可爱的朋友们，遇到了耐心又有趣的师门同学，也遇到了我的另一半，这一段人生旅程是我一生的美好回忆与力量源泉。

感谢我的导师付绍静老师，老师在生活和学习上都给予我们充分的关心和照拂，研一时常询问我们是否在学习生活上有困难，在和老师聊天的过程中，我也慢慢找到了努力的方向。老师言传身教，总是在清晨和深夜看到老师忙碌的身影，我深受鼓舞。老师积极的生活态度，严谨的治学态度以及如沐春风的待人态度，是我永远的学习方向。

感谢柳林老师和罗玉川老师对我的指导。罗玉川老师总是在组会上提出针对性的问题，让我发掘知识盲点。柳老师指导我找到了研究方向，告诉我如何看论文，写论文。老师总能在我研究卡壳的时候，给我提供新的思考角度和突破方向。感谢老师不辞辛苦，耐心指教，时时敦促，我才能一直投入学习不敢松懈。

感谢师门中的各位同学，师兄刘开放、张富成和杨璐铭，师姐范书珲、黄雪伦和冯丹，同级好友陈丽杰、赵侠男、王寅秋和杨旭，师弟师妹黄欣怡、杨钰婷和刘琨鹏等人。和大家一起留下了很多难忘的回忆，师兄们常和我们聊天分享各种经验，雪伦师姐常在我沮丧时悉心开导我，犹记一起在301实验室学习到深夜，丽杰、侠男、寅秋和小杨都是非常有趣善良的人，我们一起努力奋斗，感谢他们充实丰富了我的读研生活，小黄师妹认真踏实乐于分享。与罗淞巍相遇于科大，相识于师门，相熟于网安课程，相恋于初春，相守至今，我们一起阅读健身学习工作，感谢一路的陪伴和支持，今后也要一起奋进美好新生活。

感谢我的家人和朋友们，父母给我提供了莫大的支持，让我没有后顾之忧，母亲一直以来都是我最坚强的后盾，没有她，就没有我的今天。她乐观的态度，坚强的性格和真诚的处世之道，是我一生最宝贵的财富。父母文化水平有限，一生辛勤工作不求回报，只为拖举着我去看更大的世界。感谢相识逾十年的好友小姚小佳，我们一起分享生活分享快乐。感谢大学室友冬倩萧萧舒舒，相离未相忘，你们的陪伴让我的生活趣味良多，感谢我的室友朱姝和许诗瑶，小朱同学打游戏总是给我带来很多快乐，小许同学认真的性格上进的态度让我受益良多。

凡是过往，皆为序章。感恩相遇，希望今后我们都心有所想，事有所成，在各自的人生道路上发光发热。

## 参考文献

- [1] Sadiku M N, Musa S M, Momoh O D. Cloud computing: opportunities and challenges[J]. IEEE potentials, 2014, 33(1): 34~36.
- [2] Manvi S S, Shyam G K. Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey[J]. Journal of network and computer applications, 2014, 41: 424~440.
- [3] Antonopoulos N, Gillam L. Cloud computing[M]. Springer, 2010.
- [4] Ribeiro M, Grolinger K, Capretz M A. Mlaas: Machine learning as a service[C]// 2015 IEEE 14th international conference on machine learning and applications (ICMLA). 2015: 896~902.
- [5] Hunt T, Song C, Shokri R, et al. Chiron: Privacy-preserving machine learning as a service[J]. ArXiv preprint arXiv:1803.05961, 2018.
- [6] Liu Y, Li X, Zhang M H A, et al. When Machine Learning Meets Privacy: A Survey and Outlook[J]. ACM Computing Surveys (CSUR), 2021, 54(2): 1~38.
- [7] Ahmed M, Seraj R, Islam S M S. The k-means algorithm: A comprehensive survey and performance evaluation[J]. Electronics, 2020, 9(8): 1295.
- [8] Cramer R, Damgård I B, et al. Secure multiparty computation[M]. Cambridge University Press, 2015.
- [9] Li P, Li J, Huang Z, et al. Privacy-preserving outsourced classification in cloud computing[J]. Cluster Computing, 2018, 21: 277~286.
- [10] Montazerolghaem A, Yaghmaee M H, Leon-Garcia A. Green cloud multimedia networking: NFV/SDN based energy-efficient resource allocation[J]. IEEE Transactions on Green Communications and Networking, 2020, 4(3): 873~889.
- [11] Dwork C. Differential privacy[C]//Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II 33. 2006: 1~12.
- [12] Dwork C. Differential privacy: A survey of results[C]//Theory and Applications of Models of Computation: 5th International Conference, TAMC 2008, Xi’ an, China, April 25-29, 2008. Proceedings 5. 2008: 1~19.
- [13] Team A, et al. Learning with privacy at scale[J]. Apple Mach. Learn. J, 2017, 1(8): 1~25.
- [14] Erlingsson Ú, Pihur V, Korolova A. Rappor: Randomized aggregatable privacy-preserving ordinal response[C]//Proceedings of the 2014 ACM SIGSAC conference on computer and communications security. 2014: 1054~1067.
- [15] Ding B, Kulkarni J, Yekhanin S. Collecting telemetry data privately[J]. Advances in Neural Information Processing Systems, 2017, 30.

- [16] Dwork C. A firm foundation for private data analysis[J]. Communications of the ACM, 2011, 54(1): 86~95.
- [17] Rivest R L, Adleman L, Dertouzos M L, et al. On data banks and privacy homomorphisms[J]. Foundations of secure computation, 1978, 4(11): 169~180.
- [18] Gentry C. Fully homomorphic encryption using ideal lattices[C]//Proceedings of the forty-first annual ACM symposium on Theory of computing. 2009: 169~178.
- [19] Acar A, Aksu H, Uluagac A S, et al. A survey on homomorphic encryption schemes: Theory and implementation[J]. ACM Computing Surveys (Csur), 2018, 51(4): 1~35.
- [20] Evans D, Kolesnikov V, Rosulek M. A Pragmatic Introduction to Secure Multi-Party Computation[J]. Foundations & Trends® in Privacy & Security, 2018, 2(2-3): 70~246.
- [21] Yao A C. Protocols for secure computations[C]//Proc. of the 23rd Annual IEEE Symposium on Foundations of Computer Science, 1982. 1982.
- [22] Jian L, Juuti M, Yao L, et al. Oblivious Neural Network Predictions via MinIONN Transformations[C]//The 2017 ACM SIGSAC Conference. 2017.
- [23] Goyal V, Mohassel P, Smith A. Efficient two party and multi party computation against covert adversaries[C]//Advances in Cryptology–EUROCRYPT 2008: 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings 27. 2008: 289~306.
- [24] Malkhi D, Nisan N, Pinkas B, et al. Fairplay-Secure Two-Party Computation System.[C]//USENIX security symposium: vol. 4. 2004: 9.
- [25] Pinkas B, Schneider T, Smart N P, et al. Secure two-party computation is practical[C]//Advances in Cryptology–ASIACRYPT 2009: 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings 15. 2009: 250~267.
- [26] Demmler D, Schneider T, Zohner M. ABY - A Framework for Efficient Mixed-Protocol Secure Two-Party Computation[C]//Network & Distributed System Security Symposium. 2015.
- [27] Jagannathan G, Wright R N. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data[C]//Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining. 2005: 593~599.
- [28] Su C, Bao F, Zhou J, et al. Privacy-preserving two-party k-means clustering via secure approximation[C]//21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07): vol. 1. 2007: 385~391.
- [29] Bozdemir B, Canard S, Ermis O, et al. Privacy-preserving density-based clustering[C]//Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security. 2021: 658~671.

- [30] Wu W, Liu J, Wang H, et al. Secure and efficient outsourced k-means clustering using fully homomorphic encryption with ciphertext packing technique[J]. IEEE Transactions on Knowledge and Data Engineering, 2020, 33(10): 3424~3437.
- [31] Jäschke A, Armknecht F. Unsupervised machine learning on encrypted data[C]//Selected Areas in Cryptography—SAC 2018: 25th International Conference, Calgary, AB, Canada, August 15–17, 2018, Revised Selected Papers. 2019: 453~478.
- [32] Kanungo T, Mount D M, Netanyahu N S, et al. An efficient k-means clustering algorithm: Analysis and implementation[J]. IEEE transactions on pattern analysis and machine intelligence, 2002, 24(7): 881~892.
- [33] Rathee D, Rathee M, Kumar N, et al. CrypTFlow2: Practical 2-party secure inference[C]//Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. 2020: 325~342.
- [34] Ester M. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise[J]. Proc.int.conf.knowledg Discovery & Data Mining, 1996.
- [35] Amirbekyan A, Estivill-Castro V. Privacy Preserving DBSCAN for Vertically Partitioned Data[C]//Proceedings of the 4th IEEE international conference on Intelligence and Security Informatics. 2006.
- [36] Bozdemir B, Canard S, Ermis O, et al. Privacy-preserving Density-based Clustering[C]//ASIA CCS '21: ACM Asia Conference on Computer and Communications Security. 2021.
- [37] Tran T N, Drab K, Daszykowski M. Revised DBSCAN algorithm to cluster data with dense adjacent clusters[J]. Chemometrics and Intelligent Laboratory Systems, 2013, 120: 92~96.
- [38] Latifi-Pakdehi A, Daneshpour N. DBHC: A DBSCAN-based hierarchical clustering algorithm[J]. Data & Knowledge Engineering, 2021, 135: 101922.
- [39] Liu D, Bertino E, Yi X. Privacy of outsourced k-means clustering[C]//Proceedings of the 9th ACM symposium on Information, computer and communications security. 2014: 123~134.
- [40] Almutairi N, Coenen F, Dures K. K-means clustering using homomorphic encryption and an updatable distance matrix: secure third party data clustering with limited data owner interaction[C]//Big Data Analytics and Knowledge Discovery: 19th International Conference, DaWaK 2017, Lyon, France, August 28–31, 2017, Proceedings 19. 2017: 274~285.
- [41] Wang Y. Notes on two fully homomorphic encryption schemes without bootstrapping[J]. Cryptology ePrint Archive, 2015.
- [42] Rong H, Wang H, Liu J, et al. Privacy-Preserving-Means Clustering under Multiowner Setting in Distributed Cloud Environments[J]. Security and Communication Networks, 2017, 2017.

- [43] Rao F Y, Samanthula B K, Bertino E, et al. Privacy-preserving and outsourced multi-user k-means clustering[C]//2015 IEEE conference on collaboration and internet computing (CIC). 2015: 80~89.
- [44] Kim H J, Chang J W. A privacy-preserving k-means clustering algorithm using secure comparison protocol and density-based center point selection[C]//2018 IEEE 11th International Conference on Cloud Computing (CLOUD). 2018: 928~931.
- [45] Doganay M C, Pedersen T B, Saygin Y, et al. Distributed privacy preserving k-means clustering with additive secret sharing[C]//Proceedings of the 2008 international workshop on Privacy and anonymity in information society. 2008: 3~11.
- [46] Patel S, Garasia S, Jinwala D. An efficient approach for privacy preserving distributed k-means clustering based on shamir' s secret sharing scheme[C]//Trust Management VI: 6th IFIP WG 11.11 International Conference, IFIPTM 2012, Surat, India, May 21-25, 2012. Proceedings 6. 2012: 129~141.
- [47] Patel S, Patel V, Jinwala D. Privacy preserving distributed k-means clustering in malicious model using zero knowledge proof[C]//Distributed Computing and Internet Technology: 9th International Conference, ICDCIT 2013, Bhubaneswar, India, February 5-8, 2013. Proceedings 9. 2013: 420~431.
- [48] Upmanyu M, Namboodiri A M, Srinathan K, et al. Efficient privacy preserving k-means clustering[C]//Intelligence and Security Informatics: Pacific Asia Workshop, PAISI 2010, Hyderabad, India, June 21, 2010. Proceedings. 2010: 154~166.
- [49] Baby V, Chandra N S. Distributed threshold k-means clustering for privacy preserving data mining[C]//2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI). 2016: 2286~2289.
- [50] Hegde A, Möllering H, Schneider T, et al. Sok: Efficient privacy-preserving clustering[J]. Cryptology ePrint Archive, 2021.
- [51] Vaidya J, Clifton C. Privacy-preserving k-means clustering over vertically partitioned data[C]//Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. 2003: 206~215.
- [52] Meskine F, Bahloul S N. Privacy preserving k-means clustering: a survey research.[J]. Int. Arab J. Inf. Technol., 2012, 9(2): 194~200.
- [53] Du W, Atallah M J. Privacy-preserving cooperative statistical analysis[C]//Seventeenth Annual Computer Security Applications Conference. 2001: 102~110.
- [54] Bunn P, Ostrovsky R. Secure two-party k-means clustering[C]//Proceedings of the 14th ACM conference on Computer and communications security. 2007: 486~497.
- [55] Sakuma J, Kobayashi S. Large-scale k-means clustering with user-centric privacy-preservation[J]. Knowledge and Information Systems, 2010, 25: 253~279.

- [56] Jiang Z L, Guo N, Jin Y, et al. Efficient two-party privacy-preserving collaborative k-means clustering protocol supporting both storage and computation outsourcing[J]. *Information Sciences*, 2020, 518: 168~180.
- [57] Mohassel P, Rosulek M, Trieu N. Practical privacy-preserving k-means clustering[J]. *Cryptology ePrint Archive*, 2019.
- [58] Anikin I V, Gazimov R M. Privacy preserving DBSCAN clustering algorithm for vertically partitioned data in distributed systems[C]//2017 International Siberian Conference on Control and Communications (SIBCON). 2017: 1~4.
- [59] Jiang D, Xue A, Ju S, et al. Privacy-preserving DBSCAN on horizontally partitioned data[C]//2008 IEEE International Symposium on IT in Medicine and Education. 2008: 1067~1072.
- [60] Kumar K A, Rangan C P. Privacy preserving DBSCAN algorithm for clustering[C]//Advanced Data Mining and Applications: Third International Conference, ADMA 2007 Harbin, China, August 6-8, 2007. Proceedings 3. 2007: 57~68.
- [61] Liu J, Huang J Z, Luo J, et al. Privacy preserving distributed DBSCAN clustering[C]//Proceedings of the 2012 Joint EDBT/ICDT Workshops. 2012: 177~185.
- [62] Xu W j, Huang L s, Luo Y l, et al. Protocols for privacy-preserving dbscan clustering[J]. *International Journal of Security and Its Applications*, 2007, 1(1).
- [63] Almutairi N, Coenen F, Dures K. Secure Third Party Data Clustering Using Data: Multi-User Order Preserving Encryption and Super Secure Chain Distance Matrices (Best Technical Paper)[C]//Artificial Intelligence XXXV: 38th SGAI International Conference on Artificial Intelligence, AI 2018, Cambridge, UK, December 11–13, 2018, Proceedings. 2018: 3~17.
- [64] Rahman M S, Basu A, Kiyomoto S. Towards outsourced privacy-preserving multiparty DBSCAN[C]//2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC). 2017: 225~226.
- [65] Wang M, Zhao W, Cheng K, et al. Homomorphic Encryption Based Privacy Preservation Scheme for DBSCAN Clustering[J]. *Electronics*, 2022, 11(7): 1046.
- [66] Ni L, Chao L, Xiao W, et al. DP-MCDBSCAN: Differential Privacy Preserving Multi-Core DBSCAN Clustering for Network User Data[J]. *IEEE Access*, 2018, PP(99): 1~1.
- [67] Wei-Min W U, Huang H K, Computer S O. A DP-DBScan clustering algorithm based on differential privacy preserving[J]. *Computer Engineering & Science*, 2015.
- [68] Zahur S, Evans D. Circuit structures for improving efficiency of security and privacy tools[C]//2013 IEEE Symposium on Security and Privacy. 2013: 493~507.
- [69] Oded G. Foundations of cryptography: volume 2, basic applications[Z]. 2009.

- [70] El Malki N, Ravat F, Teste O. KD-means: clustering method for massive data based on kd-tree[C]//22nd International Workshop On Design, Optimization, Languages and Analytical Processing of Big Data-DOLAP 2020:- vol. 2572. 2020.
- [71] Yao A C C. How to generate and exchange secrets[C]//27th annual symposium on foundations of computer science (Sfcs 1986). 1986: 162~167.
- [72] Rivest R L, Shamir A, Tauman Y. How to leak a secret[C]//Advances in Cryptology—ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings 7. 2001: 552~565.
- [73] Blakley G R. Safeguarding cryptographic keys[C]//Managing Requirements Knowledge, International Workshop on. 1979: 313~313.
- [74] Beaver D. Efficient multiparty protocols using circuit randomization[C]//Advances in Cryptology—CRYPTO’ 91: Proceedings 11. 1992: 420~432.
- [75] Schneider T, Zohner M. GMW vs. Yao? Efficient secure two-party computation with low depth circuits[C]//Financial Cryptography and Data Security: 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers 17. 2013: 275~292.
- [76] Riazi M S, Weinert C, Tkachenko O, et al. Chameleon: A hybrid secure computation framework for machine learning applications[C]//Proceedings of the 2018 on Asia conference on computer and communications security. 2018: 707~721.
- [77] Boldyreva A, Tang T. Privacy-Preserving Approximate-Nearest-Neighbors Search that Hides Access, Query and Volume Patterns[J]. Proceedings on Privacy Enhancing Technologies, 2021, 2021(4): 549~574.
- [78] Liu L, Su J, Liu X, et al. Toward highly secure yet efficient KNN classification scheme on outsourced cloud data[J]. IEEE Internet of Things Journal, 2019, 6(6): 9841~9852.
- [79] Goldreich O. Encryption Schemes. The Foundations of Cryptography, vol. 2[Z]. 2004.
- [80] Bogdanov D, Laur S, Willemson J. Sharemind: A framework for fast privacy-preserving computations[C]//Computer Security-ESORICS 2008: 13th European Symposium on Research in Computer Security, Málaga, Spain, October 6-8, 2008. Proceedings 13. 2008: 192~206.
- [81] Naeem M, Asghar S. Kegg metabolic reaction network data set[J]. The UCI KDD Archive, 2011.
- [82] Fränti P, Sieranoja S. K-means properties on six clustering benchmark datasets[J]. Applied intelligence, 2018, 48: 4743~4759.
- [83] Ester M, Kriegel H P, Sander J, et al. A density-based algorithm for discovering clusters in large spatial databases with noise.[C]//Kdd: vol. 96: 34. 1996: 226~231.

- [84] Hubert L, Arabie P. Comparing partitions[J]. Journal of classification, 1985, 2: 193~218.
- [85] Khan K, Rehman S U, Aziz K, et al. DBSCAN: Past, present and future[C]//The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014). 2014: 232~238.
- [86] Gheid Z, Challal Y. Efficient and privacy-preserving k-means clustering for big data mining[C]//2016 IEEE Trustcom/BigDataSE/ISPA. 2016: 791~798.
- [87] Yu T K, Lee D, Chang S M, et al. Multi-party k-means clustering with privacy consideration[C]//International symposium on parallel and distributed processing with applications. 2010: 200~207.
- [88] Ultsch A. Clustering wih som: U\* c[C]//Proc. Workshop on Self-Organizing Maps (Jan. 2005)(zitiert auf S. 53). 2005.
- [89] Gionis A, Mannila H, Tsaparas P. Clustering aggregation[J]. Acm transactions on knowledge discovery from data (tkdd), 2007, 1(1): 4~es.
- [90] Vinh N X, Epps J, Bailey J. Information theoretic measures for clusterings comparison: is a correction for chance necessary?[C]//Proceedings of the 26th annual international conference on machine learning. 2009: 1073~1080.
- [91] Arbelaitz O, Gurrutxaga I, Muguerza J, et al. An extensive comparative study of cluster validity indices[J]. Pattern recognition, 2013, 46(1): 243~256.
- [92] Buitinck L, Louppe G, Blondel M, et al. API design for machine learning software: experiences from the scikit-learn project[J]. ArXiv preprint arXiv:1309.0238, 2013.

## 作者在学期间取得的学术成果

### 发表的学术论文

- [1] Yang Y, Ren T L, Zhang L T, et al. Miniature microphone with silicon-based ferroelectric thin films. *Integrated Ferroelectrics*, 2003, 52: 229~235. (SCI 收录, 检索号:758FZ)
- [2] 杨轶, 张宁欣, 任天令, 等. 硅基铁电微声学器件中薄膜残余应力的研究. *中国机械工程*, 2005, 16(14): 1289~1291. (EI 收录, 检索号:0534931 2907)
- [3] 杨轶, 张宁欣, 任天令, 等. 集成铁电器件中的关键工艺研究. *仪器仪表学报*, 2003, 24(S4): 192~193. (EI 源刊)
- [4] Yang Y, Ren T L, Zhu Y P, et al. PMUTs for handwriting recognition., In press. (已被 *Integrated Ferroelectrics* 录用. SCI 源刊.)
- [5] Wu X M, Yang Y, Cai J, et al. Measurements of ferroelectric MEMS microphones. *Integrated Ferroelectrics*, 2005, 69: 417~429. (SCI 收录, 检索号:896KM.)
- [6] 贾泽, 杨轶, 陈兢, 等. 用于压电和电容微麦克风的体硅腐蚀相关研究. *压电与声光*, 2006, 28(1): 117~119. (EI 收录, 检索号:06129773469.)
- [7] 伍晓明, 杨轶, 张宁欣, 等. 基于 MEMS 技术的集成铁电硅微麦克风. *中国集成电路*, 2003, 53: 59~61.

### 研究成果

- [1] 任天令, 杨轶, 朱一平, 等. 硅基铁电微声学传感器畴极化区域控制和电极连接的方法: 中国, CN1602118A. (中国专利公开号.)
- [2] Ren T L, Yang Y, Zhu Y P, et al. Piezoelectric micro acoustic sensor based on ferroelectric materials: USA, No.11/215, 102. (美国发明专利申请号.)