基于多方安全计算的联邦密度聚类算法研究

申旭弘[1], 于海宁[1], 王孝余[2]

[1]哈尔滨工业大学网络空间安全学院, 哈尔滨 中国 150001
[2]国网黑龙江省电力有限公司电力科学研究院, 哈尔滨 中国 150001

摘要 聚类是一种流行的无监督机器学习技术, 它将相似的数据分组成簇。聚类被应用于众多领域的数据分析, 包括金融分析、医疗分析等。许多聚类应用都包含了敏感信息, 由于数据隐私保护政策, 这些敏感信息在聚类时不应被泄露。随着数据分析技术的发展, 现在通常需要对多个来源的数据进行聚类, 以提高分析的质量, 这需要高效的隐私保护聚类算法来保护各个参与方的隐私。然而, 现有的隐私保护聚类方法仅支持 2～3 个参与方共同聚类。本文实现了联邦密度聚类算法(Federated Density-Based Spatial Clustering of Applications with Noise, FDBSCAN)。FDBSCAN 是一种基于多方安全计算技术的支持任意多参与方的高效隐私保护聚类方法。FDBSCAN 算法基于 DBSCAN 算法对点的定义, 构造了多个聚类簇, 每个簇都由至少一个核心点和所有由它可达的点组成。在 FDBSCAN 中, 各参与方利用秘密共享加密数据并分享给其他参与方进行联合聚类, 避免了参与方隐私信息泄露与中间信息泄露问题。FDBSCAN 还基于秘密共享和二进制共享的混合协议实现了隐私保护的条件控制语句。实验表明, 与现有的隐私保护聚类算法相比, FDBSCAN 能够支持更多的参与方, 并且在聚类准确度与效率上表现更佳。在常见的聚类场景中, FDBSCAN 能够获得与明文 DBSCAN 算法相同的聚类准确度, 并且在计算效率上达到了可用水平。FDBSCAN 算法在双方联合聚类的场景中, 相较于已有的隐私保护聚类算法, 在真实环境数据集中表现出了更高的效率。本文还针对轨迹聚类这一应用场景实现了联邦轨迹聚类算法(Federated Trajectory Clustering, FTC)。FTC 使用 FDBSCAN 进行聚类, 并提出了高效隐私安全的轨迹距离计算方法, 获得了较好的轨迹聚类效果。实验结果表明, FTC 算法在轮廓系数等指标中的表现要优于现有的轨迹聚类算法。

关键词 聚类; 联邦学习; 多方安全计算; 秘密共享
中图法分类号 TP309.2 DOI 号 10.19363/J.cnki.cn10-1380/tn.2025.05.02

Research on Federated Density-Based Clustering Based on Secure Multi-Party Computation

SHEN Xuhong[1], YU Haining[1], WANG Xiaoyu[2]

[1] School of Cyberspace Science, Harbin Institute of Technology, Harbin 150001, China
[2] Electric Power Research Institute of State Grid Heilongjiang Electric Power Co., Ltd., Harbin 150001, China

Abstract Clustering is a widely-used unsupervised machine learning technique that groups similar data into clusters. Clustering is used in many areas for data analysis such as financial analysis and medical analysis due to the data privacy policy. Many of these applications contain sensitive information that should not be leaked when clustering. Moreover, with the development of data analysis techniques, it is often required to group data from multiple sources to increase the quality of data analysis, which requires efficient privacy-preserving clustering to preserver each participant's privacy. However, existing privacy-preserving clustering only support 2-3 participants clustering jointly and they performs poor effiency on clustering. In this paper, we implement Federated Density-Based Spatial Clustering of Applications with Noise (FDBSCAN). FDBSCAN is an efficient privacy-preserving clustering based on Multi-Party-Computation technique that supports an arbitrary number of participants. The FDBSCAN, based on the definition of points in the plaintext DBSCAN algorithm, forms multiple clusters and each cluster contains at least one core point and all points that are reachable from it. In FDBSCAN, each participant uses secret share to encrypt data and share them with other participants for joint clustering without disclosing private information and intermediate information. FDBSCAN implemented privacy-preserving conditional control statements based on a hybrid protocol of secret share and binary share. In our experiment, comparing with existing privacy-preserving clustering, FDBSCAN supports more participants and it shows better performance on clustering effiency and clustering quality. In common clustering application scenarios, FDBSCAN performs the same clustering quality as plaintext DBSCAN, and it reaches applicable effiency. FDBSCAN, in the scenario of joint clustering between two parties, performs higher effiency on real-world datasets compared to existing privacy-preserving clustering.

通讯作者: 于海宁, 博士, 研究员, Email: yuhaining@hit.edu.cn。
本课题得到国家自然科学基金项目(No. 62172123, No. 62302122)黑龙江省自然科学基金优秀青年项目(No. YQ2021F007)资助。
收稿日期: 2023-09-20; 修改日期: 2023-12-22; 定稿日期: 2025-03-07

For trajectory clustering, we implement Federated Trajectory Clustering(FTC). FTC uses FDBSCAN to cluster data, and it proposes efficient privacy-preserving trajectory-distance calculation algorithm and reaches good clustering performance. In experiment, compared with existing trajectory clustering like TRACLUS, FTC performs better in silhouette.
Key words cluster; federated learning; secure multi-party computation; secret share

1 引言

聚类是一种广泛应用于无监督机器学习的方法, 其主要目标是将具有相似性的数据点分组成簇, 以便揭示未标记输入数据的内在结构。聚类分析被广泛应用于医学分析[1]、金融分析[2]、图像处理[3]等领域中。例如, 在传染病学中, 研究人员通过对传染病患者的活动轨迹进行轨迹聚类, 可以迅速识别传染路径, 从而采取迅速的干预措施以遏制病毒传播。聚类算法的准确度受到数据量和数据质量的显著影响。

高质量数据集合能够显著提高聚类分析质量。然而在现实中, 由于隐私监管、行业竞争等原因, 数据往往分散在各个机构部门, 这种现象被称为数据孤岛现象。由于数据孤岛现象的存在, 数据的采集和共享变得更加困难, 聚类分析算法也难以获得高质量的数据集合。例如, 对某地区的用户通信数据进行聚类分析, 运营商 A 和运营商 B 分别持有部分用户数据, 但由于隐私监管与行业竞争的原因, 这些数据无法进行安全共享, 聚类分析的质量也会下降。

隐私保护聚类算法是一种解决数据孤岛下各方联合安全聚类的方法, 它能够在隐私安全的前提下实现各数据持有方联合聚类, 在没有隐私泄露的前提下提高了聚类质量。现有的隐私保护聚类算法主要使用两种技术来保证各参与方的数据安全: 同态加密[4]与多方安全计算[5-6]。部分隐私保护聚类算法也会混合使用这些技术, 以充分发挥各自的优势。同态加密是指: 对部分数据的加密结果进行运算后再解密, 得到的结果与这些数据未加密时执行某一运算所得到的结果一致。多方安全计算是解决数据使用和隐私保护之间矛盾的重要方法。多方安全计算是指: 在多方的参与下, 共同的完成某种协同计算, 但要求每一个参与者除了计算结果以外, 无法获得其他参与者的输入实例。目前, 多方安全计算技术主要包含混淆电路[7]与秘密共享[8], 它们可用于在多方协作中保护数据隐私。

现有的隐私保护聚类算法的思想大多基于 K-means[9]算法与 DBSCAN[10]算法。尽管 K-means 算法相对简单, 但它只能对特定类型的数据进行聚类, 且仅适用于凸数据集。此外, K-means 算法需要预先确定簇的数量 K, 这需要一定先验知识。在隐私保护聚类算法中, 数据分散在各个参与方, 如何协同各方确定参数 K 是一个难题。同时, K-means 算法对数据噪声异常敏感。在多方联合聚类场景中, 各方数据质量参差不齐, 这会降低基于 K-means 的隐私保护联合聚类质量。

基于 DBSCAN 算法思想的隐私保护聚类算法更加灵活。DBSCAN 算法能够对任何类型的数据集进行聚类, 且簇的数量可以灵活确定。同时 DBSCAN 算法不受异常值的影响, 且能将异常值标记为噪声。图 1 展示了在四个数据集中, DBSCAN 算法相对于 K-means 算法更具优势。在多方隐私保护联合聚类的场景下, 基于 DBSCAN 的隐私保护聚类在聚类效果表现优于基于 K-means 的隐私保护聚类。

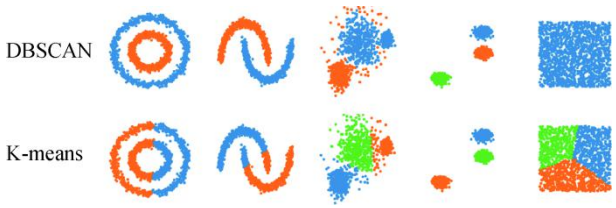

图 1 DBSCAN 与 K-means 算法的聚类结果对比
Figure 1 Comparsion of clustering results with K-means and DBSCAN

针对现有的隐私保护聚类算法的中间信息泄露问题以及参与方数量限制问题, 本文实现了支持任意多方的高效隐私保护聚类算法, 联邦密度聚类算法。同时, 针对现有研究中较少涉及到的隐私保护轨迹聚类场景, 本文实现了联邦轨迹聚类算法。

本文工作的主要贡献包括以下 2 个方面:

(1) 实现了支持任意多参与方的 FDBSCAN 算法。算法中各计算参与方将自身的数据通过秘密共享的方法分享给其他参与方进行联邦聚类, 在保护了各数据持有方数据隐私的同时, 还支持任意多参与方联合聚类。FDBSCAN 基于算术共享和二进制共享混合协议实现了隐私保护的条件控制, 避免了中间信息泄露和参与方隐私信息泄露。实验表明, 在常见的双方聚类场景中, 与现有的隐私保护聚类算法相比, FDBSCAN 在时间开销与聚类质量指标中表现更加优秀。在绝大多数应用场景中, FDBSCAN 可以仅通过现有隐私保护聚类算法一半甚至更少的时间开销, 实现与明文 DBSCAN 算法相同的聚类效果。

(2) 基于联邦密度聚类算法, 设计并实现了支持

任意多参与方的联邦轨迹聚类算法。算法设计了轨迹简化算法, 降低了各方计算开销。算法提出了两种基于秘密共享的轨迹距离计算方法, 包括轨迹近似距离计算方法以及轨迹编辑距离计算方法。算法通过上述两种轨迹距离计算方法, 通过联邦密度聚类算法进行聚类, 实现了支持任意多参与方的联邦轨迹聚类算法。算法在常见的轨迹数据集的测试结果表明, 其在轮廓系数等指标中表现甚至优于明文的 TRACLUS 算法[11]。

本文组织结构如下: 第 2 节介绍了当前隐私保护聚类的研究现状; 第 3 节介绍了本文使用到的多方安全计算协议与尝试解决的信息泄露问题; 第 4 节介绍了本文针对的问题与算法整体结构; 第 5 节介绍了本文提出的联邦密度聚类算法的算法细节; 第 6 节对本文提出的算法进行了实验分析; 第 7 节讨论了算法在效率与聚类结果的表现及原因; 第 8 节总结全文并展望未来方向。

2 相关工作

2.1 隐私计算框架平台

随着多方安全计算技术的发展, 不少研究工作实现了各类隐私计算框架平台。

Daniel Demmler 等人[12]提出了 ABY 框架。ABY 框架支持三种不同类型的共享, 分别是算术共享、布尔共享、姚氏共享, 并且在半诚实模型下使用不经意传输协议实现了三种不同类型的共享的互相转换。ABY 框架使用混合协议进行计算, 获得了相比单个协议更好的性能, ABY 框架支持双方安全计算。

Payman Mohassel 等人[13]提出了 ABY3 框架。ABY3 框架在三方联合计算的场景下实现了算术共享、布尔共享、姚氏共享的互相转化, 并且框架同时适用于半诚实模型和恶意模型。ABY3 可以看作在 ABY 框架在双方计算场景下的实现, 并且在通信阶段做出了一定优化, 其在效率上表现良好。

MP-SPDZ 是 Keller 等人[14]实现的多方安全计算虚拟机。MP-SPDZ 涵盖了多种多方安全计算协议, 包括同态加密、秘密共享、混淆电路协议等, 所有这些协议都可以用同一个高级接口来使用。MP-SPDZ 针对不同的安全模型与参与方数量选择适当多方安全计算协议。

CrypTen 框架是由 Brian Knott 等人[15]实现的多方安全计算机器学习框架, 其底层是由秘密共享实现的。其实现了算数共享与二进制共享的相互转换, 在半诚实模型下实现了任意多参与方的共同计算, 并优化实现了许多非线性函数。

目前, 一些隐私保护聚类算法选择自己实现多方安全计算协议, 大多隐私保护聚类算法都是依赖于成熟的隐私计算框架平台实现。本文提出的 FDBSCAN 算法可以在多个隐私计算框架平台上实现, 这需要隐私计算框架平台实现秘密共享的私有加法、乘法、比较协议等。本文最终使用 CrypTen 框架实现 FDBSCAN 算法。

2.2 基于 K-means 的隐私保护聚类

基于 K-means 的隐私保护聚类算法在底层安全计算技术使用了例如同态加密, 双方安全计算, 多方安全计算等。算法要求各个数据持有者都参与计算。然而, 许多已经提出的算法都产生了中间信息泄露问题, 例如 Vaidya 等人[16]提出了基于同态加密的 K-means 计算方案, 其在协议的执行过程中暴露了中心点的信息。Gheid 等人[17]的方案则会在计算过程中暴露聚类大小的信息。一些方案[18]为了更好的安全性和效率也使用了差分隐私的技术进行聚类, 但此类方案会损失聚类准确度。

Paul Bunn 等人[19]提出了基于 Paillier[20]同态加密方案与双方安全计算技术的安全双方 K-means 聚类算法, 能够在不泄露中间信息的情况下得到聚类结果, 然而同态加密方案的计算开销较大, 使得计算方案几乎不可应用。Mohassel 等人[21]提出了高效的 K-means 的隐私保护聚类算法, 其底层基于混淆电路实现, 在没有中间信息泄露的情况下表现出了较高的效率, 但算法仅支持双方的联合聚类。

2.3 基于 DBSCAN 的隐私保护聚类

基于 DBSCAN 的隐私保护聚类算法在底层安全计算技术上使用了同态加密与多方安全计算。一些算法会暴露潜在的中间计算信息, 从而造成信息泄露。例如 Anikin 等人[22]基于通过同态加密实现的支持多方参与的隐私保护聚类算法, 算法在实际计算中暴露了点之间的距离信息, 造成了中间信息泄露。

Bozedemir 等人[23]通过 ABY 框架[12]下的算术共享与逻辑电路的混合协议实现了双方隐私保护密度聚类算法, 算法在无中间信息泄露的情况下, 在可接受的时间开销内获得了较好的聚类效果, 但算法仅支持双方的联合聚类。一些方案[24-25]也尝试实现基于差分隐私的联合密度聚类算法, 但这些方案对数据添加了噪声, 降低了聚类效果。

现有的基于 DBSCAN 的隐私保护聚类算法存在参与方数量限制与中间信息泄露问题。本文提出的 FDBSCAN 算法能够在无中间信息泄露的情况下支持任意多参与方联合聚类。

2.4 隐私保护的轨迹聚类问题

较少的研究关注隐私保护的轨迹聚类问题。现

有的工作[26-27]大多聚焦于如何安全获得轨迹数据以及如何安全共享轨迹数据。Bozedemir 等提出了隐私安全的基于 TRACLUS 的轨迹聚类算法, 其实现了安全的轨迹聚类, 然而算法时间开销较大。

本文提出的联邦轨迹聚类算法使用轨迹简化算法降低各参与方的计算开销, 并提出了隐私保护的轨迹距离计算方法, 获得了较好的聚类效果。

3 预备知识

3.1 隐私保护聚类的信息泄露问题

隐私保护聚类算法尝试解决多方联合聚类场景下原始数据隐私泄露问题与中间信息泄露问题。现有的隐私保护聚类算法采用了例如同态加密、秘密共享等技术对原始数据进行了安全分享, 保护了参与方的原始数据隐私。但由于技术实现以及效率原因, 部分隐私保护聚类协议在执行时会暴露一些中间信息, 导致参与方隐私泄露。中间信息泄露问题是隐私保护聚类算法中的常见问题, 一些攻击者可以通过泄露的中间信息推测出参与方的原始数据隐私。中间信息的泄露可能发生在各个环节, 例如聚类的大小信息暴露后, 攻击者可以通过聚类中数据的数量获得其他参与方的部分数据信息, 一些关键中间信息的泄露甚至会导致原始数据隐私的泄露。

在基于 DBSCAN 的隐私保护聚类算法中, 数据点之间的关系信息是重要的聚类划分依据, 数据点之间的关系信息泄露也会造成原始数据隐私的泄露。Liu 等人[28]的研究展示了一种基于 DBSCAN 的隐私保护聚类中的中间信息泄露问题, 攻击者可以通过泄露的中间信息来推测数据持有者的隐私数据。如图 2 所示, 假设两个数据持有方 Alice 和 Bob, 其各自持有一个水平分区的数据集。Alice 持有的数据点中, 存在三个中心数据点a_1,a_2,a_3。Alice 和 Bob 采用了某种可能会产生中间信息泄露的隐私保护聚类协议共同进行计算。在聚类过程中, 该协议泄露了部分信息, 这些信息包括 Bob 有一个数据点和a_1,a_2,a_3都是密度直达的(即 Bob 的一个数据记录在a_1,a_2,a_3的ε邻域内)。Alice 此时知道自己的数据点a_1,a_2,a_3, 并且知道 Bob 有一个数据处于a_1,a_2,a_3的邻域内。图 2 中的红色部分表示了 Alice 持有的数据a_1,a_2,a_3的ε邻域的交集尽管 Alice 并不直接了解 Bob 数据集合的信息, Alice 却可以 Bob 的一个数据点在数据a_1,a_2,a_3的ε邻域的交集这一信息推断出 Bob 的原始数据信息。领域的交集越小, Alice 对于 Bob 的原始数据信息的了解程度越高。

图 2 表述的中间信息泄露问题反映了数据点之间关系被泄露的情况, 可能的泄露内容包括点之间的距离或者点之间是否为直接密度可达。事实上, 中间信息的泄露可能发生在协议的任何阶段, 任何明文的信息泄露都有可能带来中间信息泄露, 这使得攻击者能够通过这些泄露的中间信息推测出其他数据持有方的原始数据信息。一种解决中间信息泄露问题的思路是: 所有的数据以及中间计算结果均采用密文进行分享且仅在结果获取阶段进行对聚类结果进行解密。因此, 本文设计的 FDBSCAN 算法的所有数据以及中间计算结果均采用秘密共享的形式分享给其他参与方, 各参与方只能获得以秘密共享形式表达的中间信息, 无法获得任何明文信息, 进而避免了中间信息泄露。

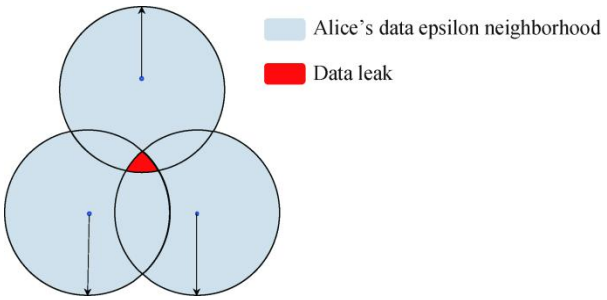

图 2 基于密度聚类的隐私保护聚类算法中的中间信息泄露问题

Figure 2 Intermediate information leak problem of privacy-preserving density-based clustering

3.2 秘密共享与算术共享

秘密共享是一种常见的多方安全计算协议。图 3 展示了秘密共享的原理, 秘密共享可用于任意数量的参与方, 每个参与方将自己的输入作为秘密值分享给计算中的所有参与方, 存在某个阈值 d, 至少需要 d 个参与方才能恢复出秘密值。多方安全计算中使用的秘密分享方案通常都是加法同态的秘密分享, 即在分享值上做加法等价于在秘密值上做加法。较早的秘密共享方案是 Shamir 秘密共享方案[8], 其将秘密值分享为 k–1 次多项式的零次项系数, 每个参与方获得一个该多项式上的点作为分享, 任意 k 个参与方即可恢复出秘密值。不过这种方案在模某个大素数 p 的同余环上进行计算, 实现中计算量比较大。现有的秘密共享实现方案将秘密值分享为模 $2k$ 同余群中 n 个随机元素的和, 具备加法同态的性质。同时, 算数共享借助于乘法 triple 来实现乘法, 较为常见的是 Beaver triples 协议[29]。所有使用 Beaver triples 来完成乘法计算的秘密共享方案, 在进行一次乘法操作时, 所有的参与方都会进行一次通信, 这也导致秘密共享体系下的乘法操作通信开销较高。目前, 不

少安全计算框架都支持秘密共享, 这主要是因为它的加法同态和乘法同态性质。

图 3 秘密共享示意

Figure 3 Diagram of secret share

算术共享是秘密共享的一种, 它同时支持加法与乘法。假设有两个参与方: Alice 和 Bob。假设 Alice 持有一个数据 x, Bob 持有一个数据 y, 对于 x,y 有 $x=x_0 \bmod 2^k+x_1 \bmod 2^k$, $y=y_0 \bmod 2^k+y_1 \bmod 2^k$。由此 Alice 获得了$x_0$和$y_0$, Bob 获得了$x_1$和$y_1$, Alice 和 Bob 分享了 x 和 y 的内容, 并且 Alice 不知道 y 的原始值, Bob 不知道 x 的原始值, 由此算术共享安全地实现了数据共享。

算数共享具有加法同态的性质。Alice 持有x_0和y_0, Bob 持有x_1和y_1, Alice 可以在本地计算$z_0=x_0+y_0$。Bob 可以在本地计算$z_1=x_1+y_1$。根据模运算的性质可以得到: $z=z_0 \bmod 2^k+z_1 \bmod 2^k$。

算数共享可以通过可信第三方(Trusted Third Party, TTP)可以实现乘法同态。算术共享的乘法操作的主流实现是 Beaver triples 协议。Beaver triples 协议开始前由可信第三方产生一个三元组$[a],[b],[c]$, $c=a*b$。其中 a,b 对所有参与方是保密的。Alice 获得a_0,b_0, 并且公开$\alpha_0=x_0-a_0$, $\beta_0=y_0-b_0$。Bob 获得a_1,b_1, 并且公开$\alpha_1=x_1-a_1$, $\beta_1=y_1-b_1$。由于公开了$\alpha_0,\alpha_1,\beta_0,\beta_1$, Alice 和 Bob 均获得了$\alpha,\beta$。Alice 可以计算$z_0=c_0+\alpha*b_0+\beta*a_0+\alpha*\beta$, 同理 Bob 可以计算$z_1=c_1+\alpha*b_1+\beta*a_1+\alpha*\beta$。将 Alice 获得的$z_0$与 Bob 获得的$z_1$利用秘密共享函数进行计算, 可知$z_0, z_1$为$z=x*y$ 的秘密共享值。算数共享乘法引入了可信第三方来产生一个三元组, 并且可信第三方需要将对应的秘密共享值分享给 Alice 和 Bob。Alice 和 Bob 也需要分别将$\alpha_0, \alpha_1, \beta_0, \beta_1$进行共享。

对于一个标量值x, $x\in Z/QZ$(模Q同余类加法群), 每一个参与方可以获得一个算术共享值$[x]=\{[x]_p\}_{p\in P}$, 并且 $x=\sum_{p\in P}[x]_p \bmod Q$。本文后续部分将统一使用$[x]$代表 x 的算数共享。算数共享的实现分为整数算数共享与浮点数算术共享两种类别。算术共享还引入了二进制共享来方便地实现私有比较协议。

整数算术共享: 持有x的参与方会为各参与方生成一个伪随机的零共享(Zero Share), 这些零共享的和为 0。真正持有 x 的参与方会将 x 添加到自己的零共享上, 随后丢弃 x。由此, 每个参与方都共享了 x。对于每一个参与方 $p\in P$, p 持有$[x]_p$, 有 $x=\sum_{p\in P}[x]_p$。

浮点数算术共享: 浮点数算术共享需要建立起浮点数和整数之间的映射。对于一个浮点数x_R, 数据持有方会生成一个较大的缩放因子 B, $x=\text{nearest}(B*x_R)$, 即 x 是距离 $B*x_R$最接近的整数。数据持有方按照整数算术共享的操作共享 x。$B=2^L$, L 是对应的比特长度。如需解码 x, $x_R\approx x/B$。

二进制共享: 二进制共享是算术共享的特殊形式。二进制共享下的所有操作都是在环 $Z/2Z$ 下进行的。对于一个标量 x, 其二进制共享$\langle x\rangle$是对其所有比特在 $Q=2$ 下的算术共享值。对于每一个参与方 $p\in P$, p 持有$\langle x\rangle_p$, 有 $x=\oplus_{p\in P}\langle x\rangle_p$。

本文使用的算术共享下的私有运算协议有如下的实现:

(1) 私有加法协议: 算术共享支持加法协议, 算术共享具有加法同态性质。对于$[z]=[x]+[y]$, 要计算$[z]$的算术共享, 对于每一个参与方 $p\in P$, p 持有$[x]_p$、$[y]_p$, 由于算术共享加法同态的性质, 参与方 p 在本地计算$[z]_p=[x]_p+[y]_p$, $[z]_p$即为加法共享。

(2) 私有乘法协议: 本文使用 Beaver triples 协议实现算术共享下的乘法。协议首先利用可信第三方生成一个三元组$([a],[b],[c])$, $c=a*b$。每一个参与方计算$[\alpha]_p=[x]_p-[a]_p$,$[\beta]_p=[y]_p-[b]_p$。之后利用掩码形式在不造成信息泄露的基础上计算出α和β。每一个参与方计算$[x*y]_p=[c]_p+\beta*[b]_p+\alpha*[a]_p+\alpha*\beta$。

(3) 私有比较协议: 通过算术共享和二进制共享的互相转换可以实现算术共享的比较协议。如果要计算$[z<0]$, 首先将算数共享$[z]$转化为二进制共享$\langle z\rangle$, 之后计算二进制共享$\langle z\rangle$的标志位$\langle b\rangle=\langle z\rangle\gg(L-1)$, 最后再将二进制共享$\langle b\rangle$转化为算术共享$[b]$, $[b]$即为$[z<0]$的算术共享结果。若要计算$[x<y]$, 则可将问题转化为$[z]=[x]-[y]$, $[z<0]$。

(4) 算术共享转化为二进制共享: 算术共享的比较操作依赖于将算数共享转化为二进制共享。各参与方持有算术共享$[x]_p$, 计算$[x]_p$的各个比特位的共享, 并计算$\langle x\rangle_p=\sum_{p\in P}\langle[x]_p\rangle$。目前主流的转化协议是 Damgård 等人实现的超进位(Carry-Lookahead)多方安全计算比较通信协议[30]。

(5) 二进制共享转化为算术共享: 算术共享的比较操作需要将二进制共享下的比较结果转化为算术共享。参与方通过计算$[x]=\sum_{b=1}^{B}2^{b}[\langle x\rangle^{(b)}]$, 其中$\langle x\rangle^{(b)}$代表了二进制共享$\langle x\rangle$的第 b 个比特位, B 代表了二进制共享的总体比特数量。各参与方为了计算$\langle x\rangle^{(b)}$的算术共享$[\langle x\rangle^{(b)}]$, 需要一个可信第三方构建随机比特位$r^{(b)}$, 各参与方会获得$r^{(b)}$的算数共享$[r^{(b)}]$与$r^{(b)}$的二进制共享$\langle r^{(b)}\rangle$, 并通过$\langle r^{(b)}\rangle$掩盖$\langle x\rangle^{(b)}$,获得掩码计算结果$z^{(b)}$。各参与方之后计算$\left[\langle x\rangle^{(b)}\right]=\left[r^{(b)}\right]+z^{(b)}-2\left[r^{(b)}\right]z^{(b)}$, 并通过该计算结果获得二进制共享转化为算数共享的结果。

根据实验结果与私有运算协议原理分析发现, 私有乘法协议与私有比较协议的耗时较大。对于私有乘法协议, 其依赖于可信第三方生成Beaver triples, 并且有可信第三方的通信。对于私有比较操作, 其在实现上需要算术共享与二进制共享的互相转化, 并且在二进制转化为算术共享时, 需要通过可信第三方构建多个随机比特位$r^{(b)}$, 并且需要分享给各参与方, 通信次数较多。因此, 从时间开销与通信开销上来说私有比较协议开销最大。

算数共享下有一些协议通过近似值实现了除法与一些非线性函数, 但非线性函数计算协议开销较大且仅为近似实现。本文实现的 FDBSCAN 算法与 FTC 算法使用了私有加法、乘法、比较协议。

4 算法概述

4.1 问题定义

本文实现的 FDBSCAN 采用秘密共享协议实现各参与方的数据交换与通信。如图 4 所示, 各计算参与方在决定算法的参数并且准备自己持有的数据后, 会利用秘密共享协议与其他参与方交换数据。交换数据后, 所有的计算均是基于秘密共享协议实现, 并且计算的中间结果也是秘密共享形式的, 包括比较等操作的结果均为秘密共享形式。各个参与方会严格遵守基于 FDBSCAN 算法中的计算协议。FDBSCAN 算法的参数包括$\varepsilon, minpts, maxIteration$。各参与方在实际开始聚类之前需要协商确定各参数。

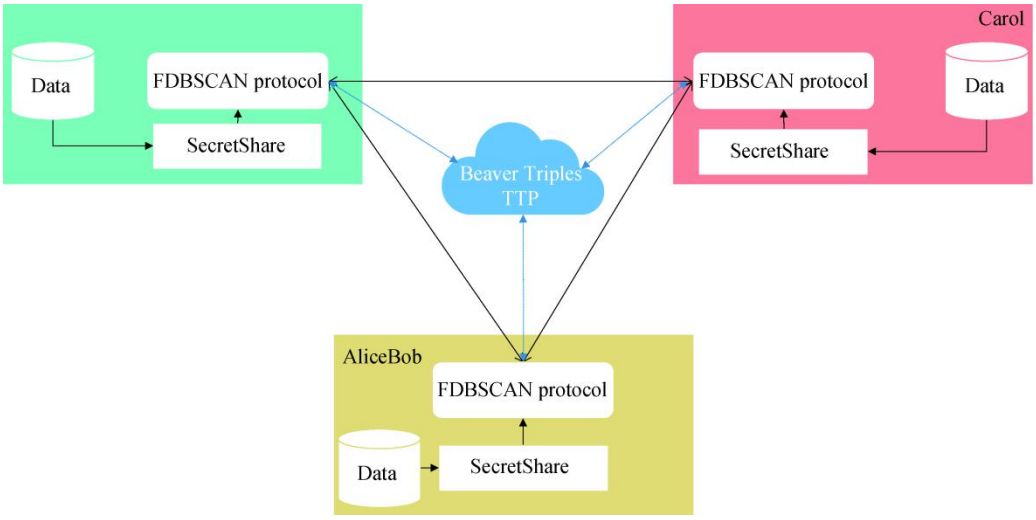

图 4 FDBSCAN 算法结构

Figure 4 Framework of FDBSCAN

为了更加清晰地描述 FDBSCAN 算法实现的聚类效果, 本文对 FDBSCAN 算法有如下的定义:

(1) 中心点: 如果点 p 在距离为ε^2的范围内有至少 $minpts$ 个点, 则称 p 为中心点。

(2) 直接可达: 如果点 p 是中心点, 则在其欧式平方距离为ε^2的范围内所有点, p 到这些点是直接可达的。没有任何点是非中心点直接可达的。

(3) 可达: 如果存在一条路径 $p_m, p_{m+1}, \cdots, p_n$,对于路径上的每一个点$p_i$, p_{i+1}都是p_i的直接可达点, 则称p_n是p_m的可达点。没有任何点是非中心点可达的。

(4) 噪声: 所有不由任何点可达的点都被称为噪声点。

(5) 连结性: 如果存在一个点 o, 点 p,q 都是由 o 可达的, 则称点 p,q 是连结的。

FDBSCAN 算法使用欧式平方距离作为点之间的距离度量方法。在 FDBSCAN 算法中获得的聚类结果中, 假设聚类结果为 $C=\{c_1,c_2,\cdots,c_i\}$, 其中c_m

满足聚类中至少有一个中心点p_m, 聚类中任意点都是互相连结的, 且如果一个点 p 是由一个聚类里的点 q 可达的, 那么 p 也在 q 的聚类中。当算法协议结束时, 如果存在点没有被划分到任一聚类中, 则称该点为噪声点。FDBSCAN 算法能最终能够获得数据集中的聚类与其中的数据噪声点。

4.2 安全模型

FDBSCAN 算法实现了半诚实模型, 即所有的协议参与方在协议执行过程中都遵守协议的规则, 但试图利用在协议中合法获得的信息, 恢复出有关其他参与方的输入信息。FDBSCAN 算法将所有需要分享给其他参与方的信息都封装为算术共享形式, 且中间计算结果也通过算术共享形式进行封装, 各参与方无法获得其他参与方的信息, 避免了中间信息泄露, 实现了半诚实模型。

算法的协议中, 各参与方之间的通信过程使用 CrypTen[15]框架实现, CrypTen 框架底层实现了本文使用的算数共享以及其相关的私有加法协议、私有乘法协议(基于 Beaver Triples 实现)、私有比较协议。CrypTen 框架实现了安全的可信第三方生成以及安全的通信协议, 在各方通信中不会泄露信息, 实现了半诚实模型。

4.3 算法结构

算法的整体结构分为参与方与可信第三方，其中参与方的数量相比于过去已实现的隐私保护聚类算法可以提升至任意多参与方。通常来说，已实现的隐私保护聚类算法仅支持 2 个参与方。

算法的参与方同时扮演了计算参与方与数据提供方的角色。各参与方都持有一份相同的算法程序副本，各参与方按照程序副本执行 FDBSCAN 算法协议中的交互、计算。算法的开始阶段，各个参与方会读入需要聚类的数据。各参与方会将数据转化为算数共享上分享给其他参与方。

算法中需要可信第三方参与的部分包含两部分: 基于 Beaver Triples 的私有乘法协议、私有比较协议。在私有乘法协议中, 算法需要可信第三方生成多个 Beaver triples 用于安全的算术共享乘法, Beaver triples 协议依赖于可信第三方生成随机数。在私有比较协议中, 算法需要将算数共享转化为二进制共享, 转换的过程中需要可信第三方生成随机比特位。

5 联邦密度聚类算法

5.1 隐私保护的距离计算

在进行聚类之前, 各数据持有方需要将自身持有的数据通过算术共享形式分享给其他参与方。算法首先会计算这些数据的距离。由于目前没有较好的算术共享除法协议, 算法采用的是欧式平方距离, 该距离计算方法仅用到了算数共享中的加法协议和乘法协议, 并且和传统欧氏距离存在映射关系, 距离度量效果最好。

公式 1 展示了算术共享下的欧式平方距离计算, 其支持高维度的点距离计算。在完成了距离度量计算之后, 为了减少聚类过程中的重复距离计算时间, 算法在聚类开始前计算点数据之间的关系, 包括是否与其他点是否为直接密度可达。在聚类过程中, 算法并不显式地计算出点之间的关系, 只需要知道数据点是否为中心点且其他点是否在其邻域内。为了保护数据持有方的数据隐私并且防止中间信息泄露, 这些信息都采用算术共享的形式分享给其他数据持有方。

$$\begin{aligned}\mathrm{SED}(s_0,s_1)&=(s_1[0]-s_0[0])^2\\&+(s_1[1]-s_0[1])^2+\cdots+(s_1[i]-s_0[i])^2\end{aligned}\tag{1}$$

其中, s_0、s_1 表示两个算数共享形式下的数据点, 可以表达为 $s_0=[[a],[b],\cdots[i]]$, 其各个维度的数据均采用了算数共享形式表达。公式 1 的最终计算结果仍为算术共享形式。

公式 1 可以计算出数据点之间欧式平方距离的算术共享形式, 且兼容了多维度数据的欧式平方距离计算。同时, 在 FDBSCAN 算法中, 距离计算与聚类过程是解耦合的, FDBSCAN 也兼容基于算术共享实现的其他距离度量方式, 例如汉明距离、曼哈顿距离等。欧式平方距离的优点是, 与欧氏距离存在映射关系, 这使得聚类的参数设定更加直观。同时, 欧式平方距离只运用到了私有加法、乘法协议, 私有加法、乘法协议在算术共享下是精确实现的, 无需使用到不精确的非线性协议实现。

5.2 隐私保护的条件表达式

DBSCAN 算法在执行时需要通过布尔表达式作为条件的判断来选择性地执行或者不执行某些语句, 例如, 判断数据点的直接密度可达点的数量是否超过了 *minPts*, 从而更新数据点是否为中心点的信息。然而, 为了确保 FDBSCAN 算法在协议执行时不泄露任何中间计算信息, 协议不能显式地知道条件表达式的值, 例如在协议执行过程中, 各参与方不能知道某一数据点的直接密度可达点的数量是否超过了 *minPts*, 否则各参与方可以通过该信息推测出其他参与方持有的数据点的信息。因此, FDBSCAN 算法通过私有比较操作产生的算术共享[0], [1], 并将其加入条件表达式控制的计算过程中, 进而实现隐

私保护的条件表达式。

在 FDBSCAN 算法的执行过程中, 假设某项数据值需要更新, 其旧值为$[a]$, 其需要更新的值为$[b]$, 更新条件的隐私保护的条件表达式为$[k]$。那么, FDBSCAN 算法会更新该数据值为$[a]$*([1] − $[k]$)+$[b]$* $[k]$。当更新条件不满足时, $[k]$=[0], 此时数值更新结果为保留旧值$[a]$; 当更新条件满足时, $[k]$=[1], 此时数值更新结果为新值$[b]$。FDBSCAN 算法通过条件表达式的算术共享值加入计算中, 实现隐私保护的条件表达式。FDBSCAN 算法不再涉及实际的条件判断, 而是通过将条件表达式的算术共享值加入到控制流下的数据值更新计算中, 实现了隐私保护的条件表达式。在各参与方的视角下, FDBSCAN 算法执行了每一条数据值更新的指令, 没有根据显式条件表达式选择性执行指令, 所有的中间信息都通过算术共享表达, 实现了无中间信息泄露。

5.3 FDBSCAN 算法

5.3.1 数据共享结构

FDBSCAN 算法设计了共享结构(Shared Element, SE)来封装算法协议中需要各方共享的数据以及可能应用到的中间信息。 在 FDBSCAN 算法中产生的中间信息中, 除了表示数据持有方的归属信息以外, 其余数据均采用算术共享形式表达, 任何中间信息的计算结果也采用算术共享形式表达, 各方获得的中间信息都是由算术共享形式封装的, 避免了潜在的中间信息泄露问题。

SE 结构存储了单个数据点在在进行密度聚类时需要共享或者计算的信息。表 1 展示了 SE 结构存储的变量信息, 其中需要共享的信息为数据点自身信息, 需要计算获得的属性包括是否为中心点、点与其他点的关系、所属的聚类、是否经过处理、是否为噪声点等信息。为了获得这些信息, 算法会在实际聚类开始前计算点之间的距离, 距离的计算结果采用秘密共享形式封装。在计算了点之间的距离后, 通过秘密共享比较协议, 算法还能获得任意两点之间是否直接密度可连的秘密共享信息, 并且还能获得点是否为中心点的秘密共享信息。算法在各参与方的交互过程中, 仅暴露了数据的持有方信息, 其他任何数据相关的信息均采用秘密共享的形式进行封装, 其他参与方均只能获得数据归属方的信息, 而无法获得数据的其他任意信息。

5.3.2 初始聚类

在聚类开始前, FDBSCAN 算法会计算各个点之间算术共享下的欧式平方距离, 并将其和参数$[\varepsilon]$ 进行比较, 从而得到两点之间的关系, $[\varepsilon]$表示点之间的密度ε的秘密共享形式。算法通过 $p.DAN$ 来表示点 p 和其他点的距离关系, 当 FDBSCAN 计算了某一点 p 和p_j的计算共享下的欧式平方距离$[d]$后, 比较$[d]$与$[\varepsilon]$, 并更新 $p.DAN$ $[j]$=$[d]$<$[\varepsilon]$。进一步地, 当某一点 p 获得了和其他所有点的关系时, 算法将所有比较结果相加并和$[minPts]$进行比较, 从而判断该点是否为核心点, 并将 $p.isCenter$ 更新为比较结果, 即更新 $p.isCenter$=sum$(p.DAN)$<$[minPts]$。其中$[minPts]$表示簇中最少元素个数 $minPts$ 的秘密共享形式。算法 1 的所有计算结果都通过算数共享形式表达, 在算法运行过程中, 没有明文的中间信息泄露。

表 1 SE 结构存储的变量信息

Table 1 Variable information in SE

变量名	变量含义	类型	存储形式
isCenter	是否为中心点	中间信息	算术共享
clusterID	当前所属的聚类	中间信息	算术共享
processed	是否经过处理	中间信息	算术共享
data	数据点坐标	原始信息	算术共享
party	数据归属方	原始信息	明文
DAN	ε邻接矩阵	中间信息	算术共享

算法 1.更新点之间的信息

输入: 包含各计算参与方秘密共享数据的列表 *SEL*, 聚类密度参数ε。

输出: 更新点间关系信息后的秘密共享数据列表*SEL*

1. FOR ALL p IN *SEL*
2. 计算 p 与其他秘密共享数据 q 之间的欧式平方距离 d
3. 比较 d 与ε
4. 更新 p 与 q 的密度可达信息
5. END FOR
6. FOR ALL p IN SEL
7. 计算 p 的直接密度可达点的数量
8. 更新 p 的中心节点信息为 p 的直接密度可达点数量是否大于 $minPts$
9. $p.isCenter$=sum$(p.DAN)$>$minPts$
10. END FOR

FDBSCAN 算法在完成点之间信息的计算后, 会对各数据点进行初始聚类。初始聚类是指, 在给定一个数据集后, 对于其中所有元素p_i以及和它直接密度连接的元素都划分为一个簇。尽管初始聚类获得的结果和明文 DBSCAN 算法获得的聚类效果相同,

但该结果已经获得了一部分聚类信息, 这也为后续的聚类合并提供了初始的聚类信息。

算法目标是从输入的数据集中建立一个初始的聚类簇。对数据集中的每个元素p_i, p_i有且只有在是核心点且未处理的情况下才能分配新簇, 如果p_i是核心点且未处理, 则创建一个新的簇分配给p_i, 并且将p_i标记为已处理。由于所有的值都用算术共享的形式表达, 算法无法显式地判断p_i是否已经处理, 因此算法构建了算术共享表达式来计算p_i应该划分的簇。一个问题是, 如何通过算术共享表达式来表达这些条件控制语句。

算法中的一些控制条件可以转化为算术共享表达式, 算法利用秘密共享下的乘法协议, 隐式地通过这些算数共享表达式控制了最终计算结果。例如, 对数据集中的每个元素p_i分配新簇的条件可以表达为: 1. p_i必须是核心点; 2. p_i未经处理。针对条件 1, 其算数共享表达式为$p_i.isCenter$, 表达式算术共享计算结果仅包含[0], [1]; 针对条件 2, 其算数共享表达式为$p_i.Processed$, 表达式的算术共享计算结果仅包含[0], [1]。因此, 算法通过 $res=p_i.isCenter*([1]-p_i.Processed)$来控制条件表达式。当且仅当$res=[1]$时, p_i可以分配新簇的条件成立。因此, 尽管算法也无法知道 res 的内容, 但可以通过 res 控制计算结果。假设$[ID_{new}]$为新分配的簇, $[ID_{old}]$为p_i的之前的簇 ID。那么, 更新p_i的簇 ID 的算数共享表达式可以构造为: $p_i.clusterID=res*[ID_{new}]+([1]-res)*[ID_{old}]$。当条件成立时,算法为$p_i$更新簇 ID, $p_i.clusterID=[ID_{new}]$; 当条件不成立时, 算法不会更新p_i的簇 ID, $p_i.clusterID=[ID_{old}]$。由此, 算法通过算术共享表达式实现了条件控制。

同时, 当分配了一个新的簇 ID 给中心点后, 中心点内的所有密度可达且未被处理的点都应该被划分进入这个簇。算法仍然使用算术共享表达式来控制条件表达式。对于一个刚分配新簇的中心点p_i, 对于所有的其他数据点p_j, 如果p_j需要被纳入新簇中, 那么其需要满足以下两个条件: 1. p_j未经过处理; 2. p_j是在p_i的邻域范围内。条件 1 的算术共享表达式为$p_j.Processed$, 其可能的算术共享计算结果仅包含[0], [1]。条件 2 的算术共享表达式为$p_i.DAN[j]$, 其中$p_i.DAN$ 表示p_i-ε邻域信息。因此, 条件 1 和条件 2 可以表达为如下的算数共享表达式: $temp=p_i.DAN[j]*([1]-p_i.Processed)$, p_j的簇 ID 可以表达为: $p_j.clusterID=temp*pi.clusterID+(1-temp)*p_j.clusterID$。

算法 2.初始聚类算法

输入: 包含点间信息的秘密共享数据列表 SEL
输出: 包含聚类信息的秘密共享数据列表 SEL

1. 更新 $ClusterID=[1]$
2. FOR ALL p IN SEL
3. $b_1=p.Processed$ AND $p.isCenter$
4. 为 p 分配一个新的簇 c
5. $p.clusterID=ClusterID*b_1$
6. $ClusterID=ClusterID+b_1$
7. FOR ALL q IN SEL
8. b_2= NOT $q.Processed$ AND $p.DAN[q]$
9. 将 q 划分到 p 所属的簇
10. $q.clusterID=c.ID$
11. END FOR
12. END FOR

算法 2 中所有布尔运算均采用隐式的隐私保护的条件控制语句实现, 避免了显式地执行条件控制语句导致中间信息泄露问题。算法 2 中所有的运算均为算术共享下的私有运算, 包括私有加法、私有乘法等。算法 2 为还未经处理的中心点分配新的聚类簇, 同时将该点邻域内所有的点都划分到这个簇中, 直到所有的中心点都经过了处理。算法 3 保证了所有的中心点都会被划分到一个簇, 而一些噪声点就不会被划分到这些聚类簇中。

初始聚类算法获得的结果是很多小型聚类。这些小型聚类并非最终的聚类结果。同时, 初始聚类算法提供了例如中心点, 数据噪声等聚类信息。算法将在后续的聚类合并中使用这些聚类信息生成最终的聚类结果。

5.3.3 聚类合并

算法执行初始聚类后, 各个点会被划分到各个聚类中。这些聚类并非算法的最终聚类结果。由于初始聚类算法只具有局部收敛的性质, 因此可能会出现两个点之间距离小于ε, 但并没有被划分到同一簇中的情况。假设根据初始聚类算法获得了两个聚类$cluster_1, cluster_2$, 其中$cluster_1$的边缘处有一点p_m, p_m是一个核心点, 同时 $cluster_2$的边缘处有p_{n+1},p_{n+2},p_{n+3}, 且它们都是核心点, 且都在p_m的ε邻域内, 即它们互为直接密度可达点。初步聚类算法很容易出现这个情况, 设$p_n \in cluster_2$,且p_n为一个核心点, 并且p_{n+1}, p_{n+2}, p_{n+3}都是p_n的直接密度可达点, 初步聚类算法首先探测到了p_n尚未处理且为核心点, 因此会为p_n创建一个新簇$cluster_2$且在内层循环中将p_{n+1},p_{n+2},p_{n+3}分类为$cluster_2$。算法之后会检测到

p_m为核心点且未处理, 因此会为其分配一个新簇$cluster_1$, 但在对p_m的ε邻域内的点划分簇时, 由于p_{n+1}, p_{n+2}, p_{n+3}是已经处理的状态, 所以不会被划入$cluster_1$。但在明文 DBSCAN 算法中, 所有密度相连点都会被聚类为一个类。p_{n+1},p_{n+2},p_{n+3}是p_m的直接密度可达点, 所以任意 $p\in cluster_2$, p 都是p_m的密度相连点。因此两个簇应该合并为一个簇。本文采用了如算法 3 所示的簇合并的算法来解决初始聚类的错误的簇划分问题。算法 3 中所有布尔运算均采用隐式的隐私保护条件控制语句实现, 所有运算均为算术共享下的私有运算。

算法 3.聚类合并算法

输入: 包含初始聚类信息的秘密共享数据列表 SEL

输出: 包含最终聚类结果的秘密共享数据列表 SEL

1. FOR ALL p IN SEL
2. FOR ALL q IN SEL
3. b_1=$p.isCenter$ AND $q.Processed$ AND $p.clusterID$<$q.ClusterID$
4. $q.clusterID$=$p.clusterID$*b_1
5. END FOR
6. END FOR

聚类合并算法目标是根据初始聚类信息获得最中聚类结果。在算法中, 所有的信息都利用算术共享形式表达, 因此算法无法显式地获得簇的各项信息, 例如簇的中心点, 簇的 ID 等。因此, 算法采用了扩散传播的思想, 将一个簇尽可能地扩散到满足条件的地方。然而, 扩散思想可能会带来循环扩散问题, 例如当簇 A 和簇 B 可以进行合并, 簇 B 中的部分点已经被纳入了簇 A, 而剩下的簇 B 的点仍然有可能将这些点重新划分为簇 B, 这会导致簇 A 和簇 B 始终无法得到合并。算法在实际进行聚类合并时会以簇 ID 相对大小作为是否可以进行聚类合并的依据。通过簇 ID 相对大小的秘密共享表达式, 可以作为秘密共享下的布尔表达式来判断是否可以进行聚类合并。

算法 3 在本质上是一个簇之间进行合并的过程。对于常见的数据集来说, 进行 2～3 次聚类合并算法就可以获得和明文 DBSCAN 算法相同的准确度。然而对于某些特殊情况, 例如数据集中数据点的重合度较高, FDBSCAN 算法需要执行多次聚类合并算法才能获得与明文 DBSCAN 算法接近的聚类准确度。图 5 展示了聚类的簇在聚类合并算法中的变化情况。初始聚类算法获得了许多较小的簇。聚类合并算法会将这些小簇不断进行合并, 并获得最终的聚类结果。

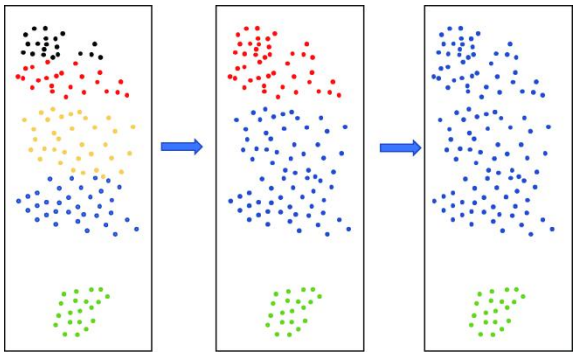

图 5　聚类合并算法中各簇的演变过程

Figure 5　Cluster changes in cluster merge algorithm

多个参与方进行 FDBSCAN 算法的流程如下:

(1) 各个参与方确认算法的各项参数, 包括 $minPts$,ε,$maxIteration$; $minPts$ 表示簇最少拥有点的数量; ε表示簇的密度; $maxIteration$ 表示最大可迭代次数, 该参数主要用于限制聚类合并操作的迭代次数。各参与方确定参数后, 会开始执行计算协议。

(2) 各个参与方将自身持有的数据转化为算数共享形式, 通过 CrypTen 框架的通信协议分享给其他参与方。

(3) 各参与方执行计算协议。算法首先会计算各个算术共享下数据点的欧式平方距离, 并获得点之间的信息关系。算法执行初始聚类算法, 并执行 $maxIteration$ 轮次的聚类合并算法, 并获得最终的聚类结果。

(4) 各方对自己持有的点进行解密, 从而获得每个点从属的簇, 进而获得聚类结果。

FDBSCAN 算法中的数据通过算术共享形式在各参与方之间共享, 并且算法的中间计算结果也通过算数共享形式表达, 因此算法不存在原始数据泄露以及中间信息泄露的风险。

5.4　基于 FDBSCAN 的联邦轨迹聚类算法

FTC 算法实现了基于算术共享的多方隐私保护轨迹聚类算法。FTC 算法首先通知各计算执行方在本地对明文的轨迹数据应用轨迹简化算法, 之后将所有简化后的轨迹数据转化为算术共享形式并存储在对应数据结构中。各参与方定义轨迹或线段之间的距离公式, 计算各轨迹或各线段之间的距离。之后可以直接调用已实现的 FDBSCAN 算法进行聚类并输出轨迹聚类结果。

5.4.1　轨迹简化算法

FTC 算法考虑了使用轨迹简化算法来降低轨迹距离的计算量。轨迹数据的数据量庞大, 单条轨迹会包含数个坐标点, 过多的坐标点会加大算术共享的运算量, 这会对算法的实际效率产生影响。因此, 本

文实现了轨迹简化算法, 能够在几乎不影响轨迹聚类效果的前提下通过轨迹坐标点集合中的数个点代替原有轨迹。在多个计算参与方开始进行隐私保护轨迹聚类之前, 每个计算参与方都需要在本地执行轨迹化简的工作。轨迹简化算法的目标是在尽可能保留轨迹特征的同时减少轨迹数据中坐标点的数量。轨迹简化算法通过寻找轨迹中的两点, 如果这两点的连线可以代表这两点之间所有顺序相连的线段, 则用这两点之间的连线替换掉该线段组。而在判断两点连线是否能够代表线段组时, 会充分考虑到线段组的长度、角度等因素。本文采用公式 2 的方法判断线段$p_m p_n$能否代表线段组$\{p_m,p_n\}$。

$$\begin{aligned} &\mathrm{L}(p_m,p_n)= \\ &\log_2\Big(\sum_{(i=m)}^{(n-1)}(\mathrm{len}(p_i,p_{(i+1)})*\sin\theta_i)\Big)+ \\ &\log_2\Big(\sum_{(i=m)}^{(n-1)}\mathrm{cendist}(p_i,p_{(i+1)},p_m,p_n)\Big) \end{aligned} \tag{2}$$

式中, $\{p_m,p_n\}$表示一组以p_m开始, 以p_n结束的轨迹, $\mathrm{cendist}(p_i,p_{i+1},p_m,p_n)$表示线段$p_i p_{i+1}$的中点到线段$p_m p_n$的距离, θ_i表示线段$p_i p_{i+1}$与线段$p_m p_n$的夹角。

算法会按照公式 2 的方法计算$\mathrm{L}(p_m,p_n)$, 并设置一个超参数γ来控制轨迹简化效果, γ越大, 轨迹简化效果越好, 但这也会丢失更多的轨迹特征。FTC 算法会找到轨迹中尽可能长的线段$p_m p_n$, 且满足$\mathrm{L}(p_m,p_n)<\gamma$, 用线段$p_m p_n$来代替$p_m,p_n$之间的线段组。图 6 展示了轨迹简化算法的一种典型场景, 算法直观地将弯曲程度较小的线段组合并为一条线段, 在保留了轨迹特征的同时实现了轨迹简化。

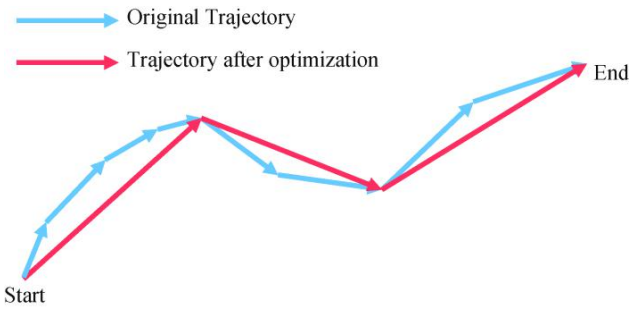

图 6　轨迹简化算法

Figure 6　Trajectory optimization algorithm

5.4.2　隐私保护的轨迹距离计算

本文总共实现了两种轨迹距离计算方法, 一种是轨迹近似距离度量方法, 另一种是隐私安全的编辑距离算法。本文在算术共享协议下, 实现了隐私保护下的两种距离计算方法。上述的隐私保护的轨迹距离计算方法不会泄露中间信息。

TRACLUS 算法对于轨迹之间的距离由平行距离、角度距离、垂直距离三部分组成。但在算术共享下, 这三个距离很难精确实现。因此, 本文考虑采用一种简化的近似距离度量, 在较少的计算量情况下实现较好的线段距离度量效果。

$$\begin{aligned} &\mathrm{dist}(l_i,l_j)=\mathrm{sed}(s_i,s_j)+\mathrm{sed}(s_i,e_j) \\ &+\mathrm{sed}(e_i,s_j)+\mathrm{sed}(e_i,e_j) \end{aligned} \tag{3}$$

式中, s_i,e_i表示线段l_i的开始点与结束点。sed 表示 FDBSCAN 算法中实现的隐私安全欧式平方距离。

公式 3 展示了轨迹近似距离度量方法。轨迹近似距离度量方法实际是对各个线段进行聚类。轨迹近似距离度量方法仅使用了已实现的隐私安全的欧式平方距离算法实现, 因此该算法也是轨迹安全的。

本文还实现了隐私安全的编辑距离算法。两个轨迹tr_1,tr_2之间的编辑距离(EDR 距离)表征了tr_1通过增加、删除、变换等操作变化为tr_2的最小操作次数。EDR 距离基于点位模糊匹配的思想, 如果两个点之间的距离在一个阈值内, 则认为这两个点之间可以相互变化(即相互变化的开销为 0)。算法用一个算术共享邻接矩阵来存储两个轨迹之间每两个点是否可以相互转化。轨迹编辑距离的计算使用到的算子包括私有加法、私有乘法、私有比较协议, 这些协议都有算术共享实现。因此算法实现的轨迹编辑距离也是隐私安全的。

6　实验分析

本文对联邦密度聚类算法以及联邦轨迹聚类算法进行了关于准确度和效率的实验, 其中针对点数据聚类效果采用聚类纯度进行衡量, 针对轨迹数据的聚类效果采用轮廓指数进行衡量。

本文实现的联邦密度聚类算法与联邦轨迹聚类算法均支持任意多参与方, 因此本文的实验部分还对参与方数量对算法效率的影响进行了分析。任意参与方都配置了相同的处理器与存储, 且均配置在局域网中。各参与方的通信协议由 CrypTen 框架实现。算法的底层使用到了算术共享的私有乘法协议, 在 CrypTen 中, 私有乘法协议采用 Beaver triples 协议实现, CrypTen 提供了多种生成 Beaver triples 的方案, 其中包含了无需第三方的 TFP 模式, 即由参与多方安全计算的某一方提供 Beaver triples, 也提供了需要可信第三方的 TTP 模式, 在 TTP 模式下, Beaver triples 依赖于一个可信第三方进行生成分发管理。无论哪种模式下, CrypTen 的 Beaver triples 生成策略都是在离线阶段大量生成, 因此 Beaver triples 的生成时间并不会对算法的整体流程有较大影响。为了与

其余依赖可信第三方的隐私保护聚类算法进行对比，本文配置了一个包含多个参与方可信第三方的实验环境。

6.1 联邦密度聚类算法评估

6.1.1 准确度评估

本文采用聚类准确度作为聚类准确度的衡量指标。本文实验后续部分中使用到的 Lsun 数据集与 S1,S2,S3 数据集都包含了原始数据的标签，采用调整兰德指数(Adjusted Rand Index, ARI)作为聚类性能评估指标。ARI 需要获得真实聚类信息来评估聚类结果, ARI 的取值范围为[–1,1], ARI 的值越接近 1，代表算法的聚类效果越好。

FDBSCAN 算法的聚类过程包括一次初始聚类与之后可以通过参数控制次数的聚类合并过程。初始聚类仅能获得数据集中部分聚类信息。聚类合并会在初始聚类获得的信息上合并聚类，获得更多聚类信息。本节将对聚类准确度与聚类合并迭代次数以及数据集之间的关系进行实验。

Lsun[31]数据集包含了 400 个 2 维数据点，这些数据源自于真实宇宙的行星坐标。400 个点实际由 3 个簇组成，每个簇分别由 200，100，100 个点，每个簇之间具有不同的方差和均值。算法中，设置 $minPts$=4，ε=3e+11，并且模拟双方安全计算环境(即将数据集水平划分为两个相等大小的数据集)首先设置 $maxIteration = 0$，该参数的设置表明了算法仅执行初始聚类步骤，并不展开进行聚类合并算法。算法对 Lsun 数据集的聚合结果如图 8 所示，算法在不进行聚类合并的情况下，ARI 仅为 0.3325。算法在进行一次聚类合并后，ARI 即可上升到 0.925，达到与 DBSCAN 算法相同的聚类效果。从图 7 的实验结果中可以看到，大多数据集仅 1 次迭代即可获得与 DBSCAN 算法相同的聚类效果。对于较为复杂的数据集，FDBSCAN 当也可以在有限次迭代中获得与 DBSCAN 算法相同的聚类效果。

Lsun 数据集是典型的离散点数据集，即数据所属的聚类之间没有重叠部分。实验结果表明，对于离散点数据集, FDBSCAN 算法仅需要较少轮次的迭代即可获得与 DBSCAN 算法相同的效果。然而对于聚类之间有重叠的数据集来说，算法无法快速达到与 DBSCAN 相同的聚类下效果。表 2 展示了对于不同重合度的数据表下 FDBSCAN 算法在一轮迭代下的 ARI 指标。数据集使用了 S1,S2,S3 数据集，这三个数据集是 Fränti 等人提出的用于衡量 K-means 聚类准确度的数据集[32]，数据集对于其他聚类算法也具有适用性。其中 S1,S2,S3 数据集的数据使用高斯分布生成，通过控制高斯分布的均值与标准差参数，这些数据集具有不同的重合度，同时 S1,S2,S3 数据集的数据个数均为 5000。对于 S1 数据集的聚类结果如图 9 所示。实验结果表明，对于重合度较高的数据集, FDBSCAN 算法无法通过一轮聚类合并获得与 DBSCAN 算法相同的聚类效果。

表 2 展示了 DBSCAN 算法与 FDBSCAN 算法在不同数据集下的调整兰德系数，其中 FDBSCAN 算法控制聚类合并次数为 1。表 2 的实验结果表明，对于重合度较低的数据集, 1 次聚类合并即可获得与 DBSCAN 算法相同的聚类效果。图 7 与表 2 的实验结果也表明，对于重合度较大的数据集, FDBSCAN 算法需要更多次的聚类合并以使得其聚类效果稳定接近明文 DBSCAN 算法取得的聚类效果。这主要是因为，当数据集中数据点重合度较大时，算法难以

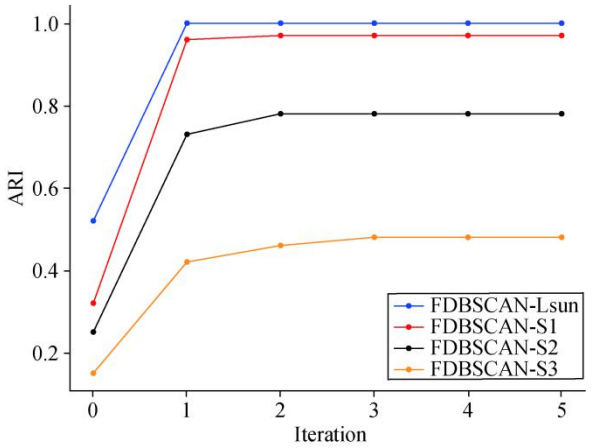

图 7 各数据集下算法迭代次数与聚类准确率的关系

Figure 7 Iteration and accuracy of FDBSCAN in datasets

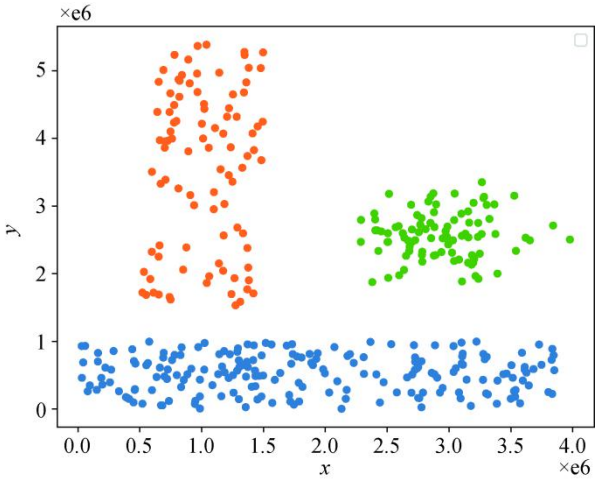

图 8 Lsun 数据集在 FDBSCAN 中的聚类结果

Figure 8 Clustering result of dataset Lsun in FDBSCAN

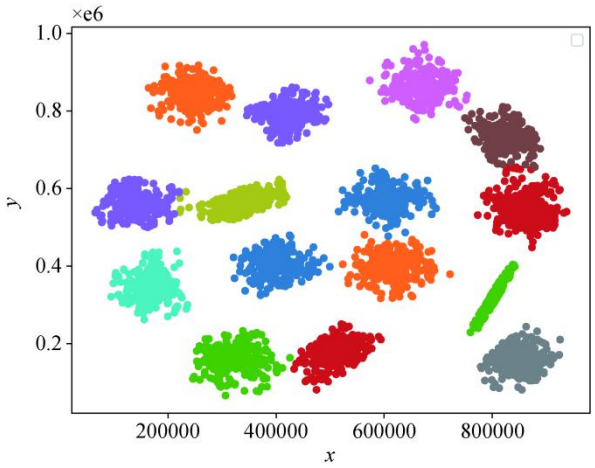

图 9　S1 数据集在 FDBSCAN 中的聚类结果

Figure 9　Clustering result of data set S1 in FDBSCAN

表 2　FDBSCAN 算法与 DBSCAN 算法的调整兰德系数

Table 2　ARI of FDBSCAN and DBSCAN

数据表名	重合度	DBSCAN	FDBSCAN
S1	9%	0.9757	0.9757
S2	22%	0.7824	0.7824
S3	41%	0.4834	0.4227

获得准确的聚类信息, 数据集中的重合部分对于算法的聚类合并产生了一定干扰, 因此需要更多轮次的聚类合并来获得更加准确的聚类信息。

6.1.2　性能评估

FDBSCAN 算法耗时主要由计算参与方个数与数据集大小确定(实验中, 假定每个参与方持有的数据集大小相同)。实验首先统计了计算参与方在 2, 3, 5 的情况下以及每个参与方数据在 10, 50, 100, 500 的情况下的距离计算耗时, 在每次评估中, 以耗时最大的参与方为准(任一参与方计算完成后, 需要等待未完成计算任务的参与方完成计算, 共同解密获得的聚类结果)。各方的数据读取与共享是同时进行的, 因此算法在数据读取阶段并不会有显著的耗时变化。本文对 FDBSCAN 算法中的距离计算与单轮聚类的耗时情况进行了实验。实验结果如表 3～表 4 所示。

实验情况表明, 随着参与方数量以及各方持有的数据量的增多, 算法的耗时逐渐增加。参与方的增多带来的耗时具有多个原因。首先, 尽管已经在距离计算阶段进行了优化, 算法仍然会使用到基于 Beaver triples 的私有乘法协议, 参与方数量的增多会显著地增加距离计算时间。其次, 在聚类合并阶段, 算法涉及了比较操作, 私有比较协议涉及多轮通信以及与多个参与方的通信, 因此时间开销增大。在距离计算与聚类中, 算法会使用到私有乘法协议, 私有乘法协议需要通过网络通信进行数据的交互, 如果参与方数量较多, 各参与方持有的数据量较大, 私有乘法协议会造成较大的时间开销。在参与方数量较小的情况下, 算法能够获得较好的效率。

表 3　距离计算耗时与参与方和数据量的关系

Table 3　Distance calculation time cost

参与方数量	各参与方持有数据量			
	10	50	100	500
2	4.7236s	112.53s	409.23s	4982.74s
3	27.18s	679.84s	2813.44s	32389s
5	152.53s	3923.28s	20417.90s	—

表 4　聚类耗时与参与方和数据量的关系

Table 4　Time cost of one-turn clustering

参与方数量	各参与方持有数据量			
	10	50	100	500
2	6.65s	132.76s	549.41s	6174.32s
3	52.38s	1008.33s	4672.30s	58722s
5	234.19s	8249.75s	43428.72s	—

上述实验结果也表明, 当算法的计算参与方控制在 2 至 3 个时, 算法能够获得较好的可用性。而当计算参与方上升到 5 个及以上时, 算法的时间开销会增大, 这主要是多个参与方之间通信开销增大带来的时间开销。

6.1.3　对比实验

现有的隐私保护聚类算法大多支持双方联合聚类。本文在相同的实验环境中, 对参与方数量限制为 2, 分别对这些算法的时间开销与聚类准确度进行了对比, 使用的数据集包括 Lsun 数据集与 S1 数据集, FDBSCAN 使用的聚类合并轮次控制在 3 轮。同时本文还设计了 UCI[33]真实数据集下的聚类对比实验。

在表 5～表 6 中, ppDBSCAN 表示 Bozedemir[23]等人提出的隐私保护聚类算法, FDBSCAN-3 表示聚类合并轮次为 3 的 FDBSCAN 算法, FDBSCAN-1 表示聚类合并轮次为 1 的 FDBSCAN 算法。表 5～表 6 均采用了相同的实验环境, 均控制参与方数量为 2。其中 ppDBSCAN 算法的耗时包含了 ABY 框架中的姚氏电路构建耗时, FDBSCAN 算法的耗时包含了 CrypTen 框架离线生成 Beaver triples 的耗时。可以看到, FDBSCAN 算法在时间开销上优于现有的隐私保护聚类算法。同时在 ARI 指标中, FDBSCAN 在 Lsun 数据集与 S1 数据集中的聚类准确度也要优于基于

K-means 思想的隐私保护聚类算法。实验表明, FDBSCAN 在时间开销与聚类效果上都要优于已有的隐私保护密度聚类算法。

表 5 隐私保护聚类算法的时间开销

Table 5 Time cost of privacy-preserving clustering algorithms

数据集	隐私保护聚类算法		
	FDBSCAN-1	ppDBSCAN	FDBSCAN-3
Lsun	322.01s	623.73s	582.39s
S1	523,971.74s	1,110,372.21s	929,710.40s
S2	522,899.92s	1,134,739.34s	921,826.82s
S3	—	1,122,382.47s	922,745.62s

表 6 聚类算法的调整兰德系数

Table 6 ARI of clustering algorithms

数据集	隐私保护聚类算法		
	FDBSCAN-1	DBSCAN	FDBSCAN-3
Lsun	1.0	1.0	1.0
S1	0.9757	0.9757	0.9757
S2	0.7824	0.7824	0.7824
S3	0.2182	0.4834	0.4834

图 7 与表 6 的实验结果表明, 即便针对簇之间重合度较高的数据集例如 S3, FDBSCAN 算法可以在 3 次聚类合并下达到与 DBSCAN 算法相同的效果。然而, 在数据点重叠度较高的数据集中, DBSCAN 算法的聚类效果也表现较差。因此, DBSCAN 算法的常用场景仍然为簇之间重合度较低的数据集。为了探究 FDBSCAN 算法在簇之间重合度较低数据集的情况, 本文选择 Lsun,S1,S2 三个簇重合度较低的数据集进行测试, 并将 FDBSCAN 的聚类合并轮次控制在 1 轮。

表 7 UCI 数据集下隐私保护聚类算法的调整兰德系数

Table 7 ARI of privacy-preserving clustering on UCI dataset

数据集	聚类算法		
	K-means	DBSCAN	FDBSCAN-1
Iris	0.583	0.732	0.732
Wine	0.830	0.661	0.661
Seeds	0.704	0.686	0.686

表 5～表 6 中的实验结果都表明, 在簇之间重合度较低的数据集中, FDBSCAN 可以通过控制聚类合并次数的方式, 在几乎不损失聚类效果的情况下大幅减少时间开销。实验结果表明, 在适用于 DBSCAN 算法的场景下, FDBSCAN 算法获得了较好的时间开销。

本文之前实验部分使用的 Lsun 数据集、S1,S2,S3 数据集是常用于评估 DBSCAN 算法的数据集。本文还选取了来源于真实世界的 UCI 数据集, 以更好地分析 FDBSCAN 的适用场景。

在表 7～表 9 中, FDBSCAN-1 表示聚类合并轮次为 1 的 FDBSCAN 算法。ACC 表示聚类准确度, 其值上界为 1, 越高代表聚类效果越好。实验结果表明, 在 UCI 的 Iris,Wine,Seeds 数据集中, FDBSCAN 算法可以在 1 次聚类合并的情况下获得与 DBSCAN 算法相同的聚类效果, 并表现出了相比于其他隐私保护聚类算法在时间开销上的优越性。从 ACC 和 ARI 的指标来看, 在真实环境的数据集下, FDBSCAN 算法能够在远优于已有隐私保护聚类算法时间开销的情况下获得与明文 DBSCAN 算法的聚类效果。

表 8 UCI 数据集下隐私保护聚类算法的聚类准确度

Table 8 ACC of privacy-preserving clustering on UCI dataset

数据集	聚类算法		
	K-means	DBSCAN	FDBSCAN-1
Iris	0.795	0.732	0.732
Wine	0.905	0.876	0.876
Seeds	0.890	0.881	0.881

表 9 UCI 数据集下各隐私保护聚类算法的时间开销

Table 9 Time cost of privacy-preserving clustering in UCI dataset

数据集	简化后聚类效果	
	ppDBSCAN	FDBSCAN-1
Iris	8247.52s	4593.74s
Wine	6559.48s	3661.59s
Seeds	6222.75s	3297.72s

6.2 联邦轨迹聚类算法评估

在前文中已经完成了对联邦密度聚类算法的各项时间开销的测量, 本章节将对轨迹简化算法的简化效果以及轨迹距离计算的时间开销进行评价。本章节采用了麋鹿轨迹数据集和飓风移动数据集两种轨迹数据集进行测试。

6.2.1 轨迹简化效果评估

本小节通过计算应用轨迹简化算法前后的轨迹数据大小来度量轨迹简化算法的效果。测试数据集来自于文献[11]提供的飓风移动轨迹数据集(Hurrica

ne1950—2006)与麋鹿轨迹数据集(Deer)。

如表 10 所示, 轨迹简化算法对不同数据集有不同的简化效果。这主要是由于, 轨迹简化算法的核心思想是用一条线段来代替一段轨迹, 从而降低数据集大小。飓风移动数据集中, 大部分轨迹较为平缓, 能够用一条线段来代表一段轨迹, 因此轨迹简化算法获得了较好的效果。

为了探究轨迹简化算法对最终聚类准确度的影响, 本文使用近似距离与编辑距离对轨迹简化后的聚类效果进行了实验。如表 11 所示, 轨迹聚类效果采用轮廓系数(Silhouette Coefficient)进行度量, 轨迹简化算法对于聚类效果的影响较小。轨迹简化算法的本质是将轨迹中相似的一组线段用单条线段来表示, 在尽可能保留轨迹原有特征的同时实现了轨迹的简化。

表 10 轨迹简化效果

Table 10 Effect of trajectory predigestion

数据集	精简前点数量	精简后点数量	精简比例
hurricane1950—2006	17736	9664	45.41%
deer	20772	18273	12.03%

表 11 轨迹简化对聚类效果影响

Table 11 Effect of trajectory predigestion on clustering

距离计算方法	简化前聚类效果	简化后聚类效果
近似距离	0.57	0.58
编辑距离	0.52	0.52

6.2.2 聚类效果评估

轨迹聚类效果难以直接量化, 因此本文将在 hurricane 数据集上使用 TRACLUS 算法与 FTC 算法进行对比, 得到 FTC 算法的实际聚类效果。实验中, 评判轨迹聚类效果的指标包括聚类获得的簇个数, 轮廓系数(Silhouette Coefficient)。实验结果如表 12 所示, 表中 FTC 表示使用简化的近似距离度量方法的 FTC 算法, FTC-E 表示使用编辑距离的 FTC 算法。可以看到, 在轮廓系数指标中, FTC 算法甚至要优于 TRACLUS 算法, 这主要是 FTC 算法在轨迹距离的计算上和 FTC 算法不同。实验结果表明, FTC 算法已经获得了较好的轨迹聚类效果。

7 讨论

本小节将探讨 FDBSCAN 算法的通用性、效率与聚类效果等问题。

表 12 Hurricane 数据集下 TRACLUS 算法与 FTC 算法的聚类效果对比

Table 12 Cluster result of TRACLUS and FTRACLUS in data set Hurricane

算法	簇个数	轮廓系数
TRACLUS	2	0.44
FTC	3	0.58
FTC-E	4	0.52

本文提出的 FDBSCAN 算法可以在多个隐私计算框架平台上实现, 这需要隐私计算框架平台实现秘密共享的私有加法、乘法、比较协议等。从理论上, FDBSCAN 算法可以使用任何支持上述私有计算协议的隐私计算框架平台实现。本文最终考虑使用 CrypTen 实现主要考虑到其对多个及以上的参与方联合计算支持良好, 并且在一些算子上有较好的优化实现, 且能在后续使用 GPU 加快效率, FDBSCAN 算法也可以在此基础上迁移至 GPU 计算实现更高效率。

FDBSCAN 算法为了尽可能提高聚类效果, 保留了许多有必要的私有比较操作, 包括数据点之间距离与密度参数ε的比较, 聚类簇内数据点数量与簇最小数据点数量 $minPts$ 的比较。私有比较协议的耗时较高, 本文为了平衡算法的效率与聚类结果以及出于安全性的考虑, 将聚类算法分为了初始聚类与聚类合并阶段, 通过控制聚类合并次数来平衡算法的时间开销与聚类结果。实验表明, 对于大多数较为简单、数据点重复度较低的数据集, FDBSCAN 算法在双方联合聚类的情境下, 通过较好的时间开销获得与 DBSCAN 算法相同的聚类准确度。对于复杂、数据点重复度较高的数据集, FDBSCAN 算法也可以通过一定的时间开销, 获得更好的聚类效果。

8 结论

本文提出了支持任意多参与方的隐私保护多方聚类算法 FDBSCAN。FDBSCAN 算法基于 DBSCAN 的思想, 通过算术共享的形式在各参与方之间共享数据, 并且算法中的中间计算结果均使用算术共享的形式来表达, 避免了中间信息泄露。同时, 本文还针对轨迹这一数据类型实现了支持任意多参与方的隐私保护轨迹聚类算法 FTC。FTC 算法通过轨迹简化算法在保留了轨迹特征的同时降低了各参与方计算开销, 同时算法实现了两种隐私保护的距离计算方式, 包括近似距离与编辑距离。实验表明, 在绝大多数双方计算环境中, 相比于已有的隐私保护聚类算法, FDBSCAN 算法可以在大幅减少的时间开销下

达到与 DBSCAN 相同的聚类结果。且 FDBSCAN 算法能够支持更多参与方进行联合聚类。FTC 算法在轮廓系数指标上甚至优于 TRACLUS 算法。未来工作将通过并行计算等方式进一步提高算法效率。

参考文献

[1] Wisaeng K. Breast Cancer Detection in Mammogram Images Using K-Means++ Clustering Based on Cuckoo Search Optimization[J]. *Diagnostics*, 2022, 12(12): 3088.

[2] Ahmed M, Mahmood A N, Islam M R. A Survey of Anomaly Detection Techniques in Financial Domain[J]. *Future Generation Computer Systems*, 2016, 55: 278-288.

[3] Coleman G B, Andrews H C. Image Segmentation by Clustering[J]. *Proceedings of the IEEE*, 1979, 67(5): 773-785.

[4] Gentry C. Fully Homomorphic Encryption Using Ideal Lattices[C]. *The Forty-First Annual ACM Symposium on Theory of Computing*, 2009: 169-178.

[5] Goldreich O. Secure multi-party computation[J]. *Manuscript. Preliminary version*, 1998, 78(110).

[6] Du W L, Atallah M J, et al. Secure Multi-Party Computation Problems and Their Applications[C]. *The 2001 Workshop on New Security Paradigms*, 2001: 13-22.

[7] Bellare M, Hoang V T, Rogaway P, et al. Foundations of Garbled Circuits[C]. *The 2012 ACM Conference on Computer and Communications Security*, 2012: 784-796.

[8] Shamir A. How to Share a Secret[J]. *Communications of the ACM*, 1979, 22(11): 612-613.

[9] Hartigan J A, Wong M A. Algorithm AS 136: A K-Means Clustering Algorithm[J]. *Journal of the Royal Statistical Society Series C* (*Applied Statistics*), 1979, 28(1): 100-108.

[10] Ester M, Kriegel H P, Sander J, et al. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise[C]. *The Second International Conference on Knowledge Discovery and Data Mining*, 1996: 226-231.

[11] Lee J G, Han J W, Whang K Y, et al. Trajectory Clustering[C]. *The 2007 ACM SIGMOD International Conference on Management of Data*, 2007: 593-604.

[12] Demmler D, Schneider T, Zohner M. ABY - a Framework for Efficient Mixed-Protocol Secure Two-Party Computation[J]. *22nd Annual Network and Distributed System Security Symposium*, *NDSS 2015*, 2015.

[13] Mohassel P, Rindal P. ABY3: A mixed protocol framework for machine learning[C]. *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018: 35-52.

[14] Keller M. MP-SPDZ: A Versatile Framework for Multi-Party Computation[C]. *The 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020: 1575-1590.

[15] Knott B, Venkataraman S, Hannun A, et al. Crypten: Secure multi-party computation meets machine learning[J]. *Advances in Neural Information Processing Systems*, 2021, 34: 4961-4973.

[16] Vaidya J, Clifton C, et al. Privacy-Preserving K-Means Clustering over Vertically Partitioned Data[C]. *The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003: 206-215.

[17] Gheid Z, Challal Y. Efficient and Privacy-Preserving K-Means Clustering for Big Data Mining[C]. *2016 IEEE Trustcom/ BigDataSE/ISPA*, 2016: 791-798.

[18] Uri S. Locally Private K-Means Clustering[J]. *Journal of Machine Learning Research*, 2021, 22: 7964-7993.

[19] Bunn P, Ostrovsky R, et al. Secure Two-Party K-Means Clustering[C]. *The 14th ACM Conference on Computer and Communications Security*, 2007: 486-497.

[20] Paillier P. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes[C]. *Advances in Cryptology- EUROCRYPT '99*, 1999: 223-238.

[21] Mohassel P, Rosulek M, Trieu N. Practical Privacy-Preserving K-Means Clustering[J]. *Proceedings on Privacy Enhancing Technologies*, 2020, 2020(4): 414-433.

[22] 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*, 2017: 1-4.

[23] Bozdemir B, Canard S, Ermis O, et al. Privacy-Preserving Density-Based Clustering[C]. *The 2021 ACM Asia Conference on Computer and Communications Security*, 2021: 658-671.

[24] Ni L N, Li C, Wang X, et al. DP-MCDBSCAN: Differential Privacy Preserving Multi-Core DBSCAN Clustering for Network User Data[J]. *IEEE Access*, 2018, 6: 21053-21063.

[25] Wei-min W, Huan-kun H. A DP-DBSCAN clustering algorithm based on differential privacy preserving[J]. *Computer Engineering & Science/Jisuanji Gongcheng yu Kexue*, 2015, 37(4).

[26] Pelekis N, Gkoulalas-Divanis A, Vodas M, et al. Private-HERMES: A Benchmark Framework for Privacy-Preserving Mobility Data Querying and Mining Methods[C]. *The 15th International Conference on Extending Database Technology*, 2012: 598-601.

[27] Aïvodji U M, Huguenin K, Huguet M J, et al. SRide: A Privacy-Preserving Ridesharing System[C]. *The 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, 2018: 40-50.

[28] Liu J F, Xiong L, Luo J, et al. Privacy Preserving Distributed DBSCAN Clustering[J]. *Transactions on Data Privacy*, 2013, 6(1): 69-85.

[29] Pullonen P. Actively secure two-party computation: Efficient beaver triple generation[J]. *Instructor*, 2013.

[30] Damgård I, Fitzi M, Kiltz E, et al. Unconditionally Secure Constant-Rounds Multi-Party Computation for Equality, Comparison, Bits and Exponentiation[C]. *Theory of Cryptography*, 2006: 285-304.

[31] Ultsch A. Clustering wih som: U* c[C]. *Proc. Workshop on Self-Organizing Maps, 2005, 53*.

[32] Fränti P, Sieranoja S. K-Means Properties on Six Clustering Benchmark Datasets[J]. *Applied Intelligence*, 2018, 48(12): 4743-4759.

[33] Markelle Kelly, Rachel Longjohn, Kolby Nottingham.The UCI Machine Learning Repository, https://archive.ics.uci.edu.

申旭弘 于 2022 年在哈尔滨工业大学信息安全专业获得学士学位。现在哈尔滨工业大学网络空间安全学院网络空间安全专业攻读硕士学位。研究兴趣包括: 多方安全计算、数据安全等。Email: 22S003121@stu.hit.edu.cn

王孝余 于 2011 年在哈尔滨工业大学计算机技术专业获得硕士学位。现任国网黑龙江电科院数字化技术中心主任。研究领域为网络安全、大数据应用等。Email: wxyLxiaoyu@163.com

于海宁 于2013年在哈尔滨工业大学信息安全专业获得博士学位。现任哈尔滨工业大学网络空间安全学院研究员。CCF 会员。研究领域为数据安全、隐私计算、AI 安全等。Email: yuhaining@hit.edu.cn