软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2023,34(3):1148−1167 [doi: 10.13328/j.cnki.jos.006784]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于贡献度证明共识机制的去中心化联邦学习框架*

乔少杰[1], 林羽丰[1], 韩 楠[2], 杨国平[1], 李 贺[3], 袁 冠[4], 毛 睿[5], 元昌安[6], Louis Alberto GUTIERREZ[7]

[1](成都信息工程大学 软件工程学院, 四川 成都 610225)

[2](成都信息工程大学 管理学院, 四川 成都 610225)

[3](西安电子科技大学 计算机科学与技术学院, 陕西 西安 710071)

[4](矿山数字化教育部工程研究中心(中国矿业大学), 江苏 徐州 221116)

[5](深圳大学 计算机与软件学院, 广东 深圳 518060)

[6](广西人机交互与智能决策重点实验室(广西科学院), 广西 南宁 530100)

[7](Department of Computer Science, Rensselaer Polytechnic Institute, New York, USA)

通信作者: 韩楠, E-mail: hannan@cuit.edu.cn

摘 要: 在大数据背景下, 保证数据可信共享是数据联邦的基本要求. 区块链技术代替传统的主从架构, 可以提高联邦学习(federated learning, FL)的安全性. 然而, 现有工作中, 模型参数验证与数据持久化所产生的巨大通信成本和存储消耗, 已经成为数据联邦中亟待解决的问题. 针对上述问题, 设计了一种高效的去中心化联邦学习框架(efficient decentralized federated learning framework, EDFL), 能够降低存储开销, 并显著提升 FL 的学习效率. 首先, 提出了一种基于贡献度证明(proof-of-contribution)的共识机制, 使得区块生成者的选举基于历史贡献度而不采用竞争机制, 从而有效发避免了挖矿过程产生的区块生成延迟, 并以异步方式缓解模型参数验证中的阻塞问题; 其次, 提出了一种角色自适应激励算法, 因为该算法基于节点的工作强度和 EDFL 所分配的角色, 所以能够激励合法节点更积极地进行模型训练, 并有效地识别出恶意节点; 再者, 提出一种区块链分区存储策略, 使得多重局部修复编码块(local reconstruction code)可被均匀地分布到网络的各个节点上, 进而降低节点的本地存储代价, 并实现了较高的数据恢复效率; 最后, 在真实的 FEMNIST 数据集上, 对 EDFL 的学习效率、存储可扩展性和安全性进行了评估. 实验结果表明, EDFL 在以上 3 个方面均优于主流的基于区块链的 FL 框架.

关键词: 数据联邦; 区块链; 大数据安全管理; 共识机制; 存储策略

中图法分类号: TP18

中文引用格式: 乔少杰, 林羽丰, 韩楠, 杨国平, 李贺, 袁冠, 毛睿, 元昌安, Gutierrez LA. 基于贡献度证明共识机制的去中心化联邦学习框架. 软件学报, 2023, 34(3): 1148−1167. http://www.jos.org.cn/1000-9825/6784.htm

英文引用格式: Qiao SJ, Lin YF, Han N, Yang GP, Li H, Yuan G, Mao R, Yuan CA, Gutierrez LA. Decentralized Federated Learning Framework Based on Proof-of-contribution Consensus Mechanism. Ruan Jian Xue Bao/Journal of Software, 2023, 34(3): 1148−1167 (in Chinese). http://www.jos.org.cn/1000-9825/6784.htm

Decentralized Federated Learning Framework Based on Proof-of-contribution Consensus Mechanism

QIAO Shao-Jie[1], LIN Yu-Feng[1], HAN Nan[2], YANG Guo-Ping[1], LI He[3], YUAN Guan[4], MAO Rui[5], YUAN Chang-An[6], Louis Alberto GUTIERREZ[7]

[1](School of Software Engineering, Chengdu University of Information Technology, Chengdu 610225, China)

* 基金项目: 国家自然科学基金(61772091, 61802035, 61962006); 四川省科技计划(2021JDJQ0021, 2022YFG0186, 2021YZD0009, 2021ZYD0033); 成都市技术创新研发项目(2021-YF05-00491-SN, 2021-YF05-02414-GX, 2021-YF05-02413-GX, 2021-YF05-02420-GX, 2021-YF05-02424-GX); 成都市重大科技创新项目(2021-YF08-00156-GX, 2021-YF08-00159-GX); 成都市"揭榜挂帅"科技项目(2021-JB00-00025-GX)

本文由"大数据治理的理论与技术"专题特约编辑杜小勇教授、杨晓春教授和童咏昕教授推荐.

收稿时间: 2022-05-14; 修改时间: 2022-07-29; 采用时间: 2022-09-23; jos 在线出版时间: 2022-10-27

[2](School of Management, Chengdu University of Information Technology, Chengdu 610225, China)

[3](School of Computer Science and Technology, Xidian University, Xi'an 710071, China)

[4](Engineering Research Center of Mine Digitization (China University of Mining and Technology), Ministry of Education, Xuzhou 221116, China)

[5](College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China)

[6](Guangxi Key Laboratory of Human-machine Interaction and Intelligent Decision (Guangxi Academy of Sciences), Nanning 530100, China)

[7](Department of Computer Science, Rensselaer Polytechnic Institute, New York, USA)

Abstract: In the background of big data, ensuring credible data sharing is the basic requirement of data federation. Using blockchain technology to replace the traditional client-server architecture can improve the security of federated learning (FL). However, the huge communication cost and storage consumption generated by model parameter validation and data persistence in existing works have become problems that need to be solved urgently in data federation. To tackle these problems, an efficient decentralized federated learning framework (EDFL) is proposed, which can reduce the system overhead and significantly improve the learning efficiency of FL. Firstly, a consensus mechanism based on proof-of-contribution (PoC) is proposed where the election of the block generation is based on historical contribution instead of using the competition mechanism, thus, it can avoid the latency of the block generation caused by the mining process, and asynchronously alleviate the congestion in the model parameter validation. Secondly, a role-adaptive incentive algorithm is presented. Because the proposed algorithm is based on the work intensity of participating nodes and the role assigned by EDFL, it can motivate legitimate nodes to actively conduct model training and effectively identify malicious nodes. Thirdly, blockchain partition storage strategy is proposed. The proposed strategy enables multiple local reconstruction code chunks to be evenly distributed to nodes in the network, which reduces the local storage consumption and achieves higher efficiency of data recovery. Lastly, the learning efficiency, storage scalability, and security of EDFL are evaluated in real FEMNIST dataset. Experimental results show that EDFL outperforms the state-of-the-art blockchain-based FL framework from the above three aspects.

Key words: data federation; blockchain; big data security management; consensus mechanism; storage strategy

大数据驱动的智能应用发展迅速, 提高了人们生产和生活的效率, 同时引发了公众对于数据安全和隐私保护问题的讨论与担忧. 当数据生产者分布在异构网络中, 如何有效且合理地管理并使用这些数据, 成为人工智能领域中一个亟需解决的问题. 一方面, 大数据服务商需要为用户提供高效的智能化服务(如 Google 的翻译系统); 另一方面, 用户考虑到个人隐私, 不愿意向服务商共享其私有数据. 为了解决大数据环境下的数据孤岛问题, Google 在 2016 年首次提出了联邦学习框架(federated learning, FL)[1]. FL 是一种分布式的机器学习范式, 能够实现大数据隐私保护, 并进行多终端协同的数据共享和机器学习建模. 每个分布式设备不需要将其隐私数据上传至中心服务器, 而是在本地进行模型训练后, 发送少量的模型更新信息[2]. 然后, 由中心服务器聚合所有的更新信息并统一优化全局模型, 该优化后的模型将分发给各个设备并用于下一轮训练. FL 从技术上打破了数据孤岛, 使得海量用户数据能够被有效地共享和使用, 受到学术界和工业界的广泛关注.

然而, 传统的 FL 框架仍面临着诸多问题, 其中最为突出的问题是基于主从架构的通信模式. 这种通信模式易受到分布式环境中恶意节点的攻击, 极大地影响了 FL 中的模型鲁棒性. 例如, 横向联邦学习研究中一种常见的图像分类任务 FEMNIST, 其易受到网络中恶意节点的攻击, 即向本地模型中注入高斯噪声, 以降低全局模型的预测性能[3]. 由图 1 可得, 传统联邦学习框架(记为 vanilla FL)在更新全局模型时, 由于缺乏针对恶意攻击的检测机制, 导致当网络中存在 15%的恶意节点向本地模型中注入方差为 1 的高斯噪声时, Vanilla FL 中全局模型的预测精度在 100 轮迭代后下降了 88.45% (相比无恶意节点的情况). 因此, 不少学者开始研究如何引入一种新技术来提高数据联邦在大数据环境下的安全性. 区块链被认为是一种安全的分布式存储方案. 得益于链式的数据结构和共识机制, 它能够抵御网络中部分恶意节点的攻击, 并维持大多数节点本地数据一致性. 以比特币[4]为例, 集成了一种高安全性的共识算法, 称为工作量证明(proof-of-work, PoW). PoW 将工作量(参与设备的算力)作为各个节点间竞争区块链记账权的标准, 因此, 成功攻击比特币网络需要消耗巨大的算力. 此外, PoW 的安全性还与参与节点的数量呈正相关. 随着比特币网络的不断发展, PoW 不仅为其提供了更高的安全性, 还实现了更高程度的去中心化.

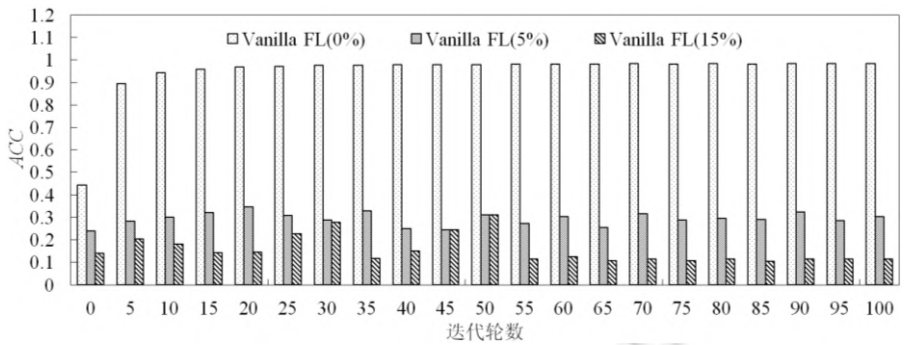


图 1 不同恶意节点的比例(0%, 5%, 15%)对于 Vanilla FL 全局模型预测精确度(*ACC*)的影响

当前, 不少学者开始研究基于区块链的 FL (blockchain-based FL, BFL)框架以提高 FL 的安全性. 然而, 现有的 BFL 框架为了提高全局模型的鲁棒性, 往往需要消耗大量的网络带宽以达成各个节点的共识结果[3,5−7]. 并且, 由于传统区块链中广泛采用的全复制策略, 使得系统的存储消耗不断增加. 在大数据背景下, BFL 框架需要更高效的共识机制和更轻量的存储策略, 以提高模型参数验证效率并减少系统的运维成本.

针对上述问题, 本文提出了一种高效的去中心化联邦学习框架(efficient decentralized federated learning framework, EDFL), 可以显著降低系统开销并提高 FL 的学习效率. 在每轮全局模型迭代开始时, EDFL 会根据历史贡献度将特定角色分配给网络中的设备: (1) 训练者(*T*)使用私有数据集训练本地模型并上传训练后的模型参数; (2) 评估者(*E*)负责评估训练者上传的模型参数; (3) 聚合者(*A*)负责聚合所有的合法参数以更新全局模型并生成新的区块. 每轮全局模型迭代结束时, 自适应奖励算法会根据各个节点的所属角色和工作强度给予相应奖励, 并更新其贡献度. 在下一轮迭代开始时, EDFL 基于各个节点的贡献度为其重新分配角色, 以保障系统中节点的流动性. 所有合法节点交替地执行模型训练、参数评估和区块生成任务, 使得全局模型能够被及时且正向地更新. 本文的主要贡献如下.

(1) 提出了一种去中心化的 FL 模型参数验证机制. EDFL 将评估者和聚合者组成验证委员会, 对所有训练者上传的模型更新参数进行去中心化验证. 只有提交合法参数的训练者才能通过验证委员会中的交叉评估进程, 然后由聚合者聚合全部合法参数并更新全局模型. 在每轮全局模型迭代结束时, 角色轮换策略基于贡献度将角色重新分配给各个节点, 保证 3 种角色所包含节点的流动性, 以实现 EDFL 验证机制的去中心化.

(2) 提出了一种基于贡献度证明的共识机制. 在每轮迭代开始时, 历史贡献度排名前 c 的节点将组成验证委员会. 其中, 拥有最高贡献度的节点将成为本轮的聚合者, 剩余的节点成为评估者, 以确保模型参数验证和区块生成进程能够被合法执行. 在验证委员会中, 评估者的投票结果与其贡献度共同影响了训练者的合法性评估. 所有的模型参数与其对应的交叉评估结果交由聚合者统一进行收集、聚合、出块(区块生成), 以维持全局模型的正向更新.

(3) 为了提高验证委员会中交叉评估的效率, 提出了一种非阻塞的交叉评估机制. 评估者与聚合者之间通过异步通信的方式缓解了参数验证进程中的阻塞问题. 此外, 在验证委员会中引入了竞争机制, 进一步提升了去中心化验证进程的执行效率.

(4) 提出了一种区块链分区存储策略. EDFL 将拥有 K 个区块的区块链进行分区存储, 使得网络中参与节点的本地存储消耗从 O(K)下降至 O(1). 为了保障数据完整性, 设计了一种面向多副本纠删编码块的均匀分配规则, 以提高分区容错性和恢复效率.

(5) 为了更加精确地评估各个节点对全局模型的贡献度, 提出了一种角色自适应奖励算法. 针对不同的角色, 其获得的奖励将基于本轮迭代的工作强度计算得出. 采用自适应奖励算法, 不仅有助于激励合法节点的模型训练, 还能有效地识别出分布式环境中的恶意节点.

本文第 1 节综述基于区块链的联邦学习框架的相关工作. 第 2 节给出本文提及方法的背景知识和主要概念. 第 3 节介绍去中心化联邦学习框架 EDFL. 第 4 节介绍基于贡献度证明的委员会共识. 第 5 节介绍区块链分区存储策略与角色自适应奖励算法. 第 6 节对比不同 FL 框架的实验结果并进行分析. 第 7 节对本文工作进行总结并探讨未来的工作.

1 相关工作

作为一个分布式的机器学习框架, FL 的首要任务是通过协调网络的各个节点来学习一个表现良好的机器学习模型. 为了实现这一目标, FL 需要确保网络的参与设备能够进行模型训练, 并向中心服务器上传合法的模型参数. 然而在传统的 FL 框架中, 这个问题并没有得到有效的解决. 因此, 不少学者开始研究基于区块链的联邦学习框架, 用于验证和记录节点上传的模型更新参数, 以构建一个高鲁棒性开放式协作学习网络.

将区块链应用于 FL, 优势在于它可以提高网络的安全性. 为了解决单点故障问题, 区块链提供了一种去中心化的模型参数的聚合方案[6,8,9]. 因为区块链具有可追溯、不可篡改和可审计的特点, 使得它可以为 FL 提供可靠且可信的分布式学习方案[10−12]. Lyu 等人[12]提出一种交叉评估机制以提高本地模型的可信度. 为了保证算法的公平性, 提出一种 3 层加密方案, 保证了数据隐私并提高模型预测的准确性. 然而, 这项工作并没有关注共识协议. 如果网络中存在恶意节点并上传非法模型参数, 将严重影响全局模型预测精确度.

对于每个节点而言, 本地的区块链账本都由一种链式的数据结构组成, 以保证数据不被篡改. 对于整个网络来说, 每轮中的数据一致性需要由一系列分布式共识算法来保证, 以确保大多数节点的共识结果不会受到部分恶意节点的干扰. 因此, 近年来的多项 BFL 研究工作主要集中在共识机制上. 根据 CAP 原则[13], BFL 框架中的共识机制被分为两大类: 基于竞争的共识算法[3,14]具有高可用性和分区容错性, 这表明区块的生成先于节点间达成共识; 与之相反, 基于通信的共识算法[7,15,16]具有更强的数据一致性和分区容错性, 要求网络中的共识结果必须先于区块产生. 前者在提升参数验证效率和节点活跃性方面发挥着重要作用, 而后者可以提高全局模型的安全性和分布式环境中的数据一致性.

Kim 等人[14]通过分析端到端的模型预测精确度来提高区块链的吞吐量, 为深度学习应用设计了一种基于 PoW 共识的 BFL 框架. 但是, 如果网络中的节点数量达到一定规模, 这种方法将产生巨大的通信消耗和区块生成延迟. 另一项基于拜占庭容错的工作[7]对联盟链进行改进, 但它没有描述其具体的验证方法, 并且破坏了 FL 参与设备的匿名性. Chen 等人[3]提出了一种基于权益证明的 FL 框架(robust blockchained federated learning with model validation, VBFL), 首次将多角色的激励机制引入 FL 模型参数验证, 并在真实的 MNIST 数据集上进行模拟实验. 但是在分布式参数验证阶段, 该方法中不同角色之间产生了较大的通信消耗; 并且, 没有考虑由于过高奖励所导致的出块权被连续独占的问题. 实验结果表明, 随着系统的运行, VBFL 的部分节点会在中后期表现出中心化的特征.

由于已有的相关工作缺少一种更公平、更高效的参数验证机制, 本文综合考虑了两类共识机制的优点, 提出了一种新的去中心化 FL 框架, 称为 EDFL. 一方面, EDFL 基于节点贡献度在每轮中选择贡献度排名靠前的节点组成验证委员会, 其中, 拥有最高贡献度的聚合者负责聚合其他评估者发来的参数评估结果并最终达成强一致性. 聚合者收集所有模型更新参数, 并聚合其中合法的部分用于更新全局模型, 然后将本轮中的训练、验证、聚合阶段的中间结果记录在新区块中. 另一方面, EDFL 利用竞争机制提高了模型参数评估的效率, 提出了一种非阻塞的交叉评估方法, 能够跳过一部分非必要的评估过程并实现等效的合法性验证. 此外, 给予评估者的工作奖励是基于时序的, 将进一步提升去中心化参数验证进程的效率.

2 背景知识

2.1 数据联邦

数据联邦是一种数据集成方式, 旨在实现各种来源、格式、特点的数据在逻辑上进行有机地集成. 数据联邦的特点是将各方信息保留在本地以保护数据隐私, 并提供统一的数据视图实现数据共享. 数据联邦中一

个典型的上层应用是联邦学习, 其能够协调分布式环境中的各个设备共同参与机器学习建模, 而无须上传各参与方的本地数据. 联邦学习不仅能够有效利用多方数据驱动智能应用进行不断优化, 还解决了各参与方对于隐私保护和数据安全的诉求. 加上数据联邦与区块链技术在数据共享方面具有较高的契合度, 目前越来越多的学者针对基于区块链的联邦学习框架展开了深入研究.

2.2 区块链技术

区块链这一概念最早是在中本聪的比特币白皮书[4]中提出的. 比特币是一种点对点的电子现金系统, 它不仅具有完全开放和数据透明的特性, 而且在保护用户数字资产方面表现出了极高的安全性. 目前, 基于区块链技术的比特币已经成为全球最大的加密货币交易系统.

从本质上讲, 区块链是一种特殊的链式数据结构, 各区块通过父区块的哈希值相互连接[17]. 如图 2 所示, 每个区块都由区块头(header)和区块体(body)两部分组成. 其中, 区块头存储着前一个区块的哈希值(previous block Hash)、提交的时间戳(timestamp)、挖矿难度(nonce)、梅克尔根(Merkle root)以及区块的版本号(version); 区块体由一系列经过排序后的交易(transaction)组成, 每个交易都存储着若干有关业务对象属性的查询和修改操作. 区块间的数据通过链式结构相连接, 以维持区块链的稳定性; 区块内的数据通过树型结构相连接, 以保证交易数据不被篡改. 因此, 任何尝试修改区块链内容的操作都将消耗巨大的计算成本. 综上所述, 区块链可以被视为一种的分布式账本, 网络中的每个节点在本地保存着同一份账本的副本, 其中记录着一系列不可篡改的数据操作记录.

如图 2 所示, 一组具有特定顺序的交易通过稳定的哈希链和梅克尔树组织起来, 这使得存储在区块链上的数据具有以下特殊属性.

(1) 去中心化. 基于点对点网络, 使得区块链系统中的节点不需要借助第三方服务器或中心节点来进行消息传递. 在网络通信中, 所有的节点都处于平等的地位. 这种去中心化的特性, 有助于提高网络的鲁棒性和公平性, 极大地降低了系统被成功攻击的概率.

(2) 可溯源性. 区块链上的每一笔交易都可以通过链式和树型的数据结构追溯到其源头. 这为每个历史的数据操作记录提供了可达性, 也证明了区块链上数据的当前状态是可信的.

(3) 防篡改性. 任何尝试修改链上数据的操作都伴随着巨大的计算成本. 区块链采用安全散列算法(secure Hash algorithm, SHA)进行数据摘要的计算, 保证了运算过程的不可逆性. 如果试图篡改一个块的内容, 除非重写在它之前的所有区块, 否则链将会断开. 这种覆盖所有父区块的操作需要巨大的算力资源, 从而保证了各节点本地账本的不可篡改性.

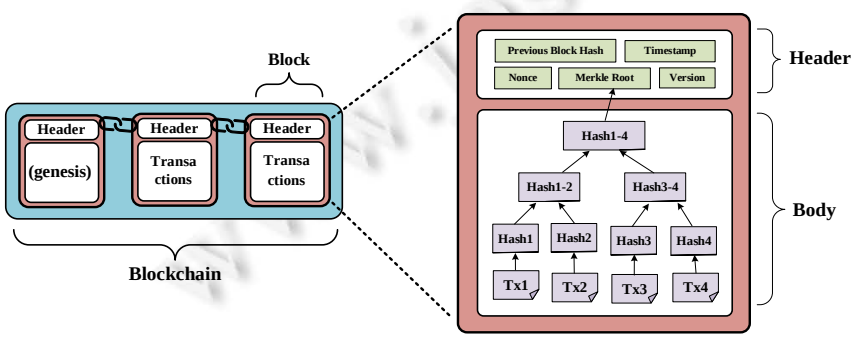


图 2 区块链的结构示意图

2.3 纠删码技术

将区块链技术应用于 FL 框架, 可以极大地提高其安全性. 但在大数据背景下, 传统区块链[3,6,8,14]中广泛使用的全复制存储策略将给 BFL 系统带来巨大的存储消耗. 一种有效的解决方案是引入纠删码技术(erasure coding)[18−21]. 这是一种前向纠错编码方法, 能够在本地生成少量的修复编码块, 以提高系统的存储可靠性.

里德-所罗门编码(Reed-Solomon coding, RSC)[22]是分布式存储系统中常用的一种纠删编码方法, 其包含一个编码器和一个解码器. 编码器为原始数据提供更多的冗余块, 使得用户能够在丢失部分数据的情况下, 从剩下的编码块中恢复原始数据. 在数据编码阶段, 给定一组长度为 K 的原始数据块, RSC 编码器能够将其编码成一组长度为 M 的冗余块. 在数据解码阶段, 因为生成了额外的 M 个冗余块, RSC 解码器能够使用 $K+M$ 个数据块中的任意 K 个数据块恢复原始数据. 给定 K 个原始数据$\{D_1,D_2,\ldots,D_k\}$, 基于柯西矩阵的 RSC 编码可以通过公式(1)计算出 M 个冗余块$\{R_1,R_2,\ldots,R_m\}$:

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ \frac{1}{x_1+y_1} & \frac{1}{x_1+y_2} & \cdots & \frac{1}{x_1+y_k} \\ \frac{1}{x_2+y_1} & \frac{1}{x_2+y_2} & \cdots & \frac{1}{x_2+y_k} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{x_m+y_1} & \frac{1}{x_m+y_2} & \cdots & \frac{1}{x_m+y_k} \end{bmatrix} \times \begin{bmatrix} D_1 \\ D_2 \\ \vdots \\ D_k \end{bmatrix} = \begin{bmatrix} D_1 \\ D_2 \\ \vdots \\ D_k \\ R_1 \\ R_2 \\ \vdots \\ R_m \end{bmatrix} \tag{1}$$

与传统的全复制策略相比, 将基于柯西矩阵的 RSC 编码应用于区块链中, 能够实现更高的存储可扩展性.

3 去中心化联邦学习框架

在只共享模型参数的前提下, FL 实现了多终端协作的机器学习模型训练, 可以有效地避免用户数据直接从本地节点传输到中心服务器过程中造成的隐私泄露. 对于传统的 FL 框架, 中心服务器负责收集、聚合和广播每轮迭代中的模型参数, 这种主从架构的通信模式增加了系统遭受单点故障和恶意攻击的概率. 一种直接的解决方法是采用去中心化的区块链代替传统的主从架构, 构建基于点对点通信的可信联邦学习网络.

尽管区块链能够提高 FL 框架中的安全性, 但它将产生额外的通信成本和存储消耗. 因此, 为了减轻大数据环境下的系统开销并且提高 FL 学习效率, 本文提出一种去中心化的联邦学习框架(EDFL). 在移除了中心服务器后, 每个分布式节点根据每轮分配到的角色分别执行模型训练、参数评估和聚合上链这 3 类任务. 采用基于贡献度证明的委员会共识, EDFL 在参数评估阶段可以激励合法节点上传本地的更新参数, 能够有效地防止恶意节点损害全局模型. 针对存储可扩展性, 本文提出了一种鲁棒的区块链分区存储策略. 它可以将节点的本地存储消耗降低至常数水平, 并显著提升其在数据恢复阶段的效率. 在下面的章节, 本文将详细介绍 EDFL 的主要工作流程.

3.1 参数定义及解释

定义 1(迭代周期). EDFL 使用一定的迭代周期去更新 FL 全局模型, 每个周期内的工作由模型训练、参数验证和聚合上链这 3 个部分组成. 迭代周期集合使用 $R=\{1,2,\ldots,j\}(j\in N_+)$表示, $|R|$是一个超参数, 表示 EDFL 中模型收敛所需要的最大迭代周期数.

定义 2(角色分配). EDFL 中的参与节点由集合 $N=\{n_1,n_2,\ldots,n_i\}(2\leqslant i\leqslant k)$表示, k 表示集合 N 的总个数$|N|$. 如图 3 所示, 在第 j 轮迭代开始时, 所有节点基于历史贡献度被划分成 3 种不同的角色($|T|+|E|+|A|=|N|$), 其中包括训练者 $T=\{t_1,t_2,\ldots,t_x\}(0<x<k-2)$训练本地模型、评估者 $E=\{e_1,e_2,\ldots,e_y\}(0<y<k-2)$评估由 T 发过来的模型参数, 聚合者 $A=\{a\}$聚合由 E 评估后的合法参数并生成新的区块.

定义 3(全局模型和本地模型). 在第 j 轮迭代中, EDFL 协调所有节点以更新 FL 全局模型, 记为 M_j. 在下一轮迭代中, M_j 将作为各个节点的本地模型, 用于进行模型训练和参数评估以更新 M_{j+1}, 如此往复.

定义 4(模型参数评估). 如图 3 所示, 评估者 E 需要使用本地数据集去评估训练者 T 发来的模型参数. 在

第 j 轮迭代中, e_y 对于 t_x 评估结果用投票 $v^{(j)}(y,x)$表示, 该投票结果将发送给聚合者等待最终验证.

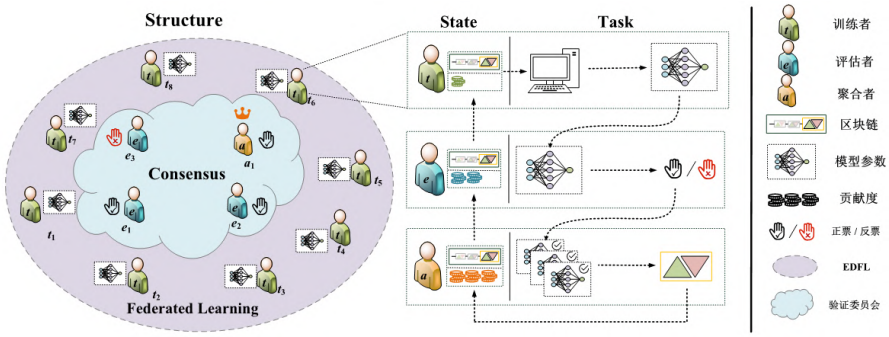


图 3 去中心化联邦学习框架 EDFL 结构图

定义 5(模型参数验证). 对于每个训练者 t_x, 聚合者需要在验证委员会中收集并聚合 t_x 有关的所有评估结果$\{v^{(j)}(y,x)\}$. 评估结果将最终决定 t_x 上传的模型更新是否能够通过委员会的合法性验证.

定义 6(节点的贡献度). 在 EDFL 中, 区块链保存有每个节点所获得的累计奖励, 称为其贡献度. 节点 n_i 在第 j 轮开始时的贡献度使用 $\mathcal{C}_i^{(j)}$ 表示, 若节点 n_i 在本轮中当选为评估者, 那么它的贡献度 $\mathcal{C}_i^{(j)}$ 也将作为验证委员会中的投票权重.

3.2 去中心化异步验证机制

如图 3 所示, EDFL 由外层的训练者和内部的验证委员会两部分组成, 其中, 验证委员会中历史贡献度最高的节点将选为本轮迭代中的聚合者, 其余节点为评估者. 首先, 外部的训练者从区块链中获取上一轮迭代的全局模型参数, 并将其更新至本地模型; 然后, 训练者使用私有数据集进行本地模型训练, 得到可用于更新本轮全局模型的模型参数并上传给评估者; 评估者接收到模型参数后, 使用本地数据集对接收到的所有模型参数进行评估和投票; 再者, 由聚合者以异步的方式地收集所有评估者的投票和对应的模型参数, 根据投票的结果以及投票者的贡献度动态决定该模型参数是否能够用于更新本轮的全局模型; 最后, 聚合者将本轮中所有合法的模型参数进行聚合, 并打包出新区块分发给其他节点.

在 EDFL 启动后, 系统中的每个参与节点需要执行以下初始化进程: (1) 数据集准备, 初始化一组本地数据样本用于训练本地模型; (2) 非对称加密密钥对生成, 使用 SHA256 算法生成一组属于该节点的公钥(PK)和私钥(SK), 公私钥之间可以互相认证以保证网络中消息的可靠性; (3) 角色分配, 根据历史贡献度从高到低, 所有节点将被分配到具体的角色(A,E,T). 如图 3 所示, 在第 j 轮迭代中, 训练者 t_x 使用其私有数据集训练本地模型 M_{j-1}. 训练完成后, t_x 将封装有模型参数的消息 $Msg_t(P_x^{(j)},SK_x)$ 使用私钥签名后发送给随机的评估者. 算法 1 给出了去中心化异步验证方法, 如下所示.

算法 1. 去中心化异步验证方法.

输入: 训练者 t_x 签名后的消息 $Msg_t(P_x^{(j)},SK_x)$, 参数验证哈希表 PT, 评估者投票队列 VQ, 区块链 $BC^{(j-1)}$.

输出: 区块链 $BC^{(j)}$.

1. **if** (评估者 e_y 接收到消息 $Msg_t(P_x^{(j)},SK_x)$)
2. **if** ($Msg_t(P_x^{(j)},SK_x)$ 能够通过本地公钥 PK_x 验证)
3. 将 $Msg_t(P_x^{(j)},SK_x)$ 广播给其他评估者 E;
4. **end if**
5. **for** $P_i^{(j)} \in \{Msg_t(P_x^{(j)},SK_x)\}$ (i=1; $i \leqslant |E|$; i++)
6. 向聚合者请求 $PT(P_i^{(j)})$;
7. **if** ($PT(P_i^{(j)})$ 没有通过验证)

8. $v^{(j)}(y,i)$=*ParametersEvaluate*($P_i^{(j)}$);
9. 向聚合者发送签名后的消息 $Msg_e(v^{(j)}(y,i), P_i^{(j)}, SK_y)$;
10. **end if**
11. **end for**
12. **end if**
13. **if** (聚合者接收到消息 $Msg_e(v^{(j)}(y,i), P_i^{(j)}, SK_y)$)
14. 初始化 *PT* 和 *VQ*;
15. **while** (在区块生成的时间阈值内)
16. **if** ($Msg_e(v^{(j)}(y,i), P_i^{(j)}, SK_y$)能够通过本地公钥 PK_y 验证且 $PT(P_i^{(j)}) \geqslant 0.5$
17. *VQ.Add*($v^{(j)}(y,i)$);
18. **if** ($v^{(j)}(y,i)$==True)
19. $\mathcal{C}_y^{(j)}$ =*FetchContribution*(e_y,$BC^{(j-1)}$);
20. *PT.Update*($P_i^{(j)}$,$\mathcal{C}_y^{(j)}$);
21. **end if**
22. **end if**
23. **end while**
24. $BC^{(j)}$=*GenerateBlock*($BC^{(j-1)}$,*PT*,*VQ*,SK_a);
25. 聚合者将 $BC^{(j)}$广播给其他节点;
26. **end if**

- 算法工作原理

对于评估者 e_y 而言，签名验证和参数评估是两项必须完成的任务. 如算法 1(第 1–4 行)所示，当评估者 e_y 接收到来自训练者 t_x 的消息 $Msg_t(P_x^{(j)}, SK_x)$ 时，需要通过本地的公钥 PK_x 去验证消息的签名，并在验证委员会之间广播合法的消息，使得每个评估者能够收集到全部训练者上传的模型参数 $\{Msg_t(P_i^{(j)}, SK_i)\}$. 然后，$e_y$ 通过向聚合者发送请求，以确定当前的模型参数 $P_i^{(j)}$ 是否已经通过验证(第 5、6 行): 如果没有通过验证，则将使用本地的数据集执行参数评估进程，并将评估者 e_y 评估结果以投票的形式发给聚合者(第 7–12 行),记为 $Msg_e(v^{(j)}(y,i), P_i^{(j)}, SK_y)$.

作为拥有最大贡献度的节点，聚合者负责收集并聚合其他节点的工作. 与先来先服务(FCFS)调度策略类似，聚合者将优先处理网络中先接收到消息(第 13–17 行). 如果参数 $P_i^{(j)}$ 此时已通过验证，则不会处理后续与之相关的投票. 聚合者基于收集到的评估投票和对应评估者的贡献度来判断参数 $P_i^{(j)}$ 是否通过了委员会的验证(第 18–20 行). 在完成所有聚合工作或达到区块生成的时间阈值时，聚合者需要在区块链上生成新的区块并广播给其他节点，以确保每一轮迭代中全局模型 M_j 能够被及时且合法地更新(第 21–24 行).

- 算法时间复杂度分析

为了提高 EDFL 中的模型参数验证效率，本文提出了一种去中心化异步验证算法. 它允许聚合者忽略部分非必要的评估者投票，提高了分布式验证进程的并行效率. 因为聚合者的选择是基于历史贡献度而非竞争机制，所以 EDFL 可以有效地避免挖矿过程带来的区块生成延迟. 与时间复杂度为 O(M×T×E)的 VBFL[3]相比，本文提出的去中心化异步验证算法能够将时间复杂度下降至 O(T×E)，其中，T 表示训练者的数量，E 表示评估者的数量，M 表示矿工的数量.

此外，本文提出了角色自适应奖励算法以进一步提高参数验证进程的效率. 在第 j 轮迭代中，不同角色所获得的预期奖励 $r_t^{(j)}$, $r_e^{(j)}$, $r_a^{(j)}$将通过区块链中对应的智能合约计算得出. 具体的奖励算法将在第 5.2 节中详细介绍.

4 基于贡献度证明的委员会共识机制

FL 的首要任务是: 基于数据联邦, 训练具有高性能的全局模型. 为了实现这一目标, FL 框架不仅需要激励合法节点积极上传模型参数, 而且需要具有较强的恶意节点识别能力. 因此, 本文提出一种基于贡献度证明的委员会共识机制, 能够保证分布式环境下的数据一致性, 并有效地降低恶意节点对全局模型的负面影响.

4.1 验证委员会

如定义 6 所述, 本文使用 $\mathcal{C}_i^{(j)}$ 来表示每个节点对全局模型的贡献度, 以此量化分布式环境中各节点的合法性. 在每轮迭代开始时, 验证委员会由贡献度排名前 c 的节点组成, 其中, 持有最高贡献度的聚合者将成为委员会的临时领导者. 为了保证大多数节点的共识结果不受到少数恶意节点的干扰, 本文利用公式(2)来验证 EDFL 委员会的安全性.

$$P_s=\sum_{i=0}^{\frac{\lceil c\times n\rceil}{2}-1}\frac{\binom{n(1-m)}{i}\times\binom{n\times m}{n\times c-i}}{\binom{n}{n\times c}} \tag{2}$$

在节点数为 n 的网络中, 系统被成功攻击的概率 P_s 由公式(2)计算得到, 其中, m 和 c 分别表示恶意节点和委员会节点在 EDFL 中的比例.

本文将系统被成功攻击定义为恶意节点占据了委员会中的大部分席位. 为了更加直观地展示 m 和 c 的不同取值对于系统安全性的影响, 本文在公式(2)的基础上进行实验, 实验结果见表 1.

表 1 在 m 和 c 的不同取值下, 概率 P_s 的分布

m	c			
	0.2	0.4	0.6	0.8
0.2	0.013 4	0	0	0
0.4	0.153 1	0.113 2	0.054 2	0
0.6	0.465 4	0.611 7	0.740 4	0.897 8
0.8	0.838	0.985 6	1	1

从总体上看, 与变量 c 相比, 变量 m 对委员会共识的影响更大. 如表 1 所示, 随着恶意节点比例 m 的变化, 委员会被成功攻击的概率 P_s 朝着 0 和 1 两个极端发展; 此外, 当恶意节点比例 m 大于 0.5 时, 随着委员会节点比例 c 的增加, 委员会被成功攻击的概率将会进一步提升. 这一现象与 PoW 共识算法[4]中的“51%算力攻击”结论吻合. 进而得出结论: 当系统中的恶意节点占据了超过半数委员会席位时, 验证委员会的共识结果将面临着被操控的风险.

4.2 贡献度证明共识机制

如上所述, 委员会共识能够为 BFL 系统提供 50%的容错能力, 但其在模型参数交叉评估过程中将产生 $|T|\times|E|$的通信消耗. 为了提高去中心化参数验证的效率, 本文提出一种非阻塞的交叉评估方法, 允许聚合者忽略一部分非必要的评估过程, 并实现等效的合法性验证. 如算法 1(第 20 行)中所示, 聚合者通过维护一个存有各个训练者上传模型参数的哈希表 PT, 来验证并聚合其中的合法参数. 对于训练者 t 而言, 如下两种验证结果在哈希表 PT 中是等效的, 例如 $PT(P_t^{(j)})$ ={True, True}和{True,True,False}. 这表明在验证委员会中, 如果当前正向投票数量超过评估者数量的一半时, 后续的评估过程将是不必要的.

因此, 为了提高验证委员会中交叉评估的效率和鲁棒性, 本文基于贡献度为每个评估者设计了不同的投票权重. 通过区块链上存储的可靠记录, 评估者 e_y 的投票权重将通过其贡献度 $\mathcal{C}_y^{(j)}$ 计算得出. 这使得一些恶意节点在验证委员会中始终保持着较低的话语权. 与公式(2)不同, 改进后的验证结果不仅取决于正向投票的数量, 还取决于评估者的历史贡献度, 这能够进一步降低委员会被成功攻击的概率. 具体来讲, 聚合者利用公式(3)来判断训练者上传的模型参数是否能够通过合法性验证.

$$PT(P_t^{(j)}) = \frac{\exp(\mathbb{C}_y^{(j)} + \beta) \times v^{(j)}(y,t)}{\sum_{i=0}^{|E|} \exp(\mathbb{C}_i^{(j)} + \beta)} + PT(P_t^{(j)}) \tag{3}$$

本文采用归一化指数函数来量化验证委员会对训练者上传参数的认可度. 哈希表 $PT(P_t^{(j)})$ 存储着训练者 t 上传参数 $P_t^{(j)}$ 的验证结果. 聚合者通过本轮中收集到的投票 $v^{(j)}(y,t)$和该投票所属评估者的贡献度 $\mathbb{C}_y^{(j)}$ 来更新 $PT(P_t^{(j)})$. 公式(3)描述了具体的更新方法, 其中, $PT(P_t^{(j)})$ 和 $PT(P_t^{(j)})$ 分别表示聚合者对模型参数 $P_t^{(j)}$ 更新前和更新后的验证结果. β是一个超参数, 用于解决 EDFL 运行中后期由于过高贡献度而导致的变量内存溢出问题. 在 $PT(P_t^{(j)})$ 完成更新后, 聚合者根据更新后的 $PT(P_t^{(j)})$ 来标记 $P_t^{(j)}$ 是否通过验证: 如果 $PT(P_t^{(j)}) \geqslant 0.5$, 则表示委员会中多数合法的评估者已经认可训练者 t 在本轮中上传的模型更新参数 $P_t^{(j)}$; 否则, 在本轮迭代中, 聚合者将不能使用参数 $P_t^{(j)}$ 来更新全局模型.

5 区块链分区存储策略与角色自适应奖励算法

在 EDFL 中, 验证委员会验证每轮迭代中上传的模型参数, 然后区块链将合法性验证结果存储在可靠的分布式账本上. 通过散列的链式数据结构, 使得存储在本地账本的共识结果变得不可篡改, 从而提高了全局模型的鲁棒性. 然而, 区块链为 FL 提供更高安全性的同时, 也给每个节点带来了巨大的本地存储消耗. 为了解决这一问题, 本文提出一种区块链分区存储策略. 为了进一步激励合法节点并且有效地识别出网络中恶意设备, 本文还提出一种角色自适应奖励算法来评估每轮中不同节点对全局模型的贡献度.

5.1 分区存储策略

在大数据背景下, 传统区块链的全复制策略不适用于有大量参与节点的 FL 框架[3,4,6,7]. 本文提出的区块链分区策略能够显著提高各个节点在数据联邦环境下的存储扩展性, 并在较低的恢复延迟下, 保证了原数据的完整性. 通过基于柯西矩阵的局部修复编码(local reconstruction code, LRC)[23], 有效地解决了 RSC 编码中数据恢复效率较低的问题. 与 RSC 的编码阶段类似, $LRC(K,M,Q)$首先通过公式(1)将 K 个原始数据块编码成 $K+M$ 个纠删编码块, 其中, K 表示原始数据块(data chunk)的数量, M 表示全局编码块(global code chunk)的数量. 在此基础上, LRC 再将 K 个原数据块平均划分为两个局部编码组, 每个编码组通过子柯西编码矩阵分别计算出 $Q/2$ 个局部编码块(local code chunk). 在丢失任意一个编码块的情况下, 不同于 RSC 至少需要 K 个编码块, LRC 仅需要 $K/2$ 个编码块即可恢复原始数据.

- 算法工作原理

在每轮迭代结束后, EDFL 将生成唯一的合法区块; 每 K 轮迭代后, 对长度为 K 的区块链执行分区存储策略. 因此, 本文将 K 轮迭代定义为一个分区周期. 如算法 2 所示, 每 K 轮迭代结束时, 各个节点准备执行区块链分区存储策略. 为了保证分区结果的一致性, 如果节点的本地区块链副本没有 j 个区块, 该节点将从聚合者请求最新的区块链 $BC^{(j)}$(第 1–4 行). 在初始化阶段, $LRC(K,M,Q)$将 $BC^{(j)}$平均分割成 K 个原始数据块(第 5 行); 然后, 再通过基于柯西矩阵的编码器生成额外的 M 个全局编码块(第 6 行). 每个全局编码块均为所有数据块的线性组合, 因此可以视为任意一个数据块的冗余替换. 本文将$\{D_1,D_2,\ldots,D_k,G_1,G_2,\ldots,G_m\}$统一表达为 $K+M$ 个编码块$\{C_1,C_2,\ldots,C_{k+m}\}$, 以便于分区到各个节点上.

算法 2. 区块链分区存储策略.

输入: $LRC(K,M,Q)$, 区块链 $BC^{(j)}$, 编码块副本数量 p, 节点索引 i.

输出: 节点 n_i 本地存储的编码块集合 $\Omega_i^{(j)}$.

1. **if** (迭代轮数 j mod K==0)
2. **if** ($BC^{(j)}.height$!=j)
3. 向聚合者请求最新区块链 $BC^{(j)}$;
4. **end if**

5. $\{D_1,D_2,\ldots,D_k\}=Split(BC^{(j)},K)$;
6. $\{G_1,G_2,\ldots,G_m\}=GlobalEncode(\{D_1,D_2,\ldots,D_k\},M)$;
7. $\Omega_i^{(j)}=Partition(\{C_1,C_2,\ldots,C_{k+m}\},i,p)$;
8. **if** ($GetAssignRole(n_i)==A$) //A 表示聚合者
9. $\{L_1,L_2,\ldots,L_q\}$=LocalEncode($\{D_1,D_2,\ldots,D_k\},Q$);
10. **end if**
11. **end if**
12. **if** (节点 n_i 使用 $\Omega_i^{(j)}$ 准备恢复原始数据)
13. 向聚合者请求$\{L_1,L_2,\ldots,L_q\}$;
14. **if** ($\Omega_i^{(j)}$ 不能恢复原始数据)
15. 根据贡献度排名依次向其他节点请求余下的(KpQ)个数据块;
16. **end if**
17. **if** ($\Omega_i^{(j)}$ 不能恢复原始数据)
18. 根据贡献度排名依次向其他节点请求余下的 M 个编码块;
19. **end if**
20. $\{D_1,D_2,\ldots,D_k\}=Decode(\Omega_i^{(j)})$;
21. $BC^{(j)}=Merge(\{D_1,D_2,\ldots,D_k\})$;
22. **end if**

为了提高分区存储后的容错性和可恢复性, 一种有效的方法是为每个编码块引入 p 个副本. 在全局编码进程结束后, 每个节点都保存有完整的编码块集合$\{C_1,C_2,\ldots,C_{k+m}\}$(第 6 行). 本文提出一种编码块均匀分配方法, 以减少节点的本地存储消耗. 每个节点 n_i 仅需要保留完整编码块的子集, 其最终存储的编码块集合 $\Omega_i^{(j)}$ 通过公式(5)计算可得(第 7 行). 其中, 分区因子 F_δ是 p 个不同的正整数($\delta\in[1,p]$), 用于控制 $p\times(K+M)$个编码块能够被均匀地分布到 N 个节点上. 为了提高数据恢复的效率, 分区因子 F_δ的分布应当尽可能地均匀, 以确保每个节点存储的编码块分别来自两个局部编码组. 本文利用公式(4)计算分区因子 F_δ, 其中, h_δ表示当前最新区块哈希值的第δ位.

$$F_\delta=\begin{cases}h_\delta \bmod (N/2), & \delta\in[1,\ p/2]\\ N-F_{(\delta-p)}, & \delta\in(p/2,\ p]\end{cases} \tag{4}$$

$$\Omega_i^{(j)}=\{C_\gamma\mid\gamma=(i+F_\delta) \bmod N+(j\times N),\delta\in[1,\ p]\} \tag{5}$$

在经过分区存储后, 每个编码块将被分发到 p 个不同的节点, 每个节点将在本地存储 p 个不同的编码块. 为了更好地利用 LRC 具有高恢复效率的特性, 本文选择将局部编码块存储在聚合者本地, 以便其他节点快速恢复原始数据(第 8–11 行). LRC 将数据块均匀地划分成两个分组, 分别使用柯西矩阵对两个分组进行局部纠删编码: $\{L_1,L_2,\ldots,L_{q/2}\}$表示第 1 个分组$\{D_1,D_2,\ldots,D_{k/2}\}$的局部编码块, $\{L_{(q/2)+1},L_{(q/2)+2},\ldots,L_q\}$表示第 2 个分组$\{D_{(k/2)+1},D_{(k/2)+2},\ldots,D_k\}$的局部编码块.

如图 4 给出的示例, 当网络中存在 6 个参与节点时(即 N=6), 使用 LRC(4,2,2)对数据进行分割和纠删编码(K=4, M=2, Q=2). 每个编码块引入 3 个副本(p=3)并配合均匀分配方法, 使得各个节点在本地保留 3 个不同的编码块(聚合者将保留额外的 2 个局部编码块), 这能够显著地提高原数据的恢复效率(第 12 –22 行). 如图 4 所示, 通过对应的分区因子 F 和本轮中分配到的角色, 节点 n_3 本地存储的编码块 $\Omega_3^{(j)}$ 能够根据其索引计算得出.

- 算法时间复杂度分析

为了减少节点的本地存储消耗, 本文在 EDFL 中集成了多副本的 LRC 纠删编码分区策略. 该分区存储策略可以将节点的本地存储消耗从 O(K)降低至 O(1), 并且拥有更高的数据恢复效率. 该分区存储策略在分区存储过程中的时间复杂度为 O(1), 在数据恢复过程中的时间复杂度为 O(K+M), 其中, K 表示原始数据块的数量,

M 表示冗余数据块的数量.

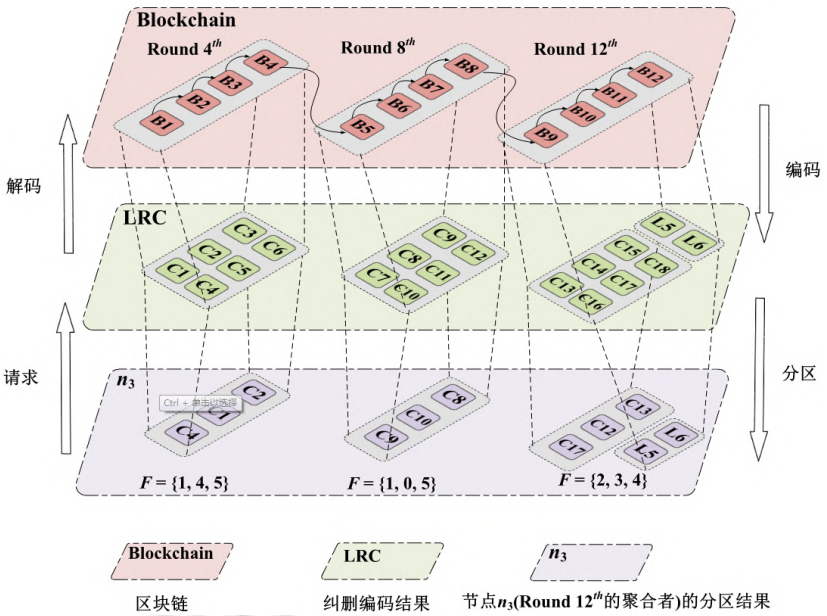


图 4 EDFL 中, 区块链分区存储与数据恢复流程图(以节点 n_3 为例)

5.2 角色自适应奖励算法

作为一种基于数据联邦的分布式机器学习范式, FL 可以实现多终端并发的机器学习模型训练. 由于面向大数据的异构网络具有数据匿名性的特点, 因此无法保证分布式环境中所有参与节点都是可靠的. 传统 FL 框架不具备恶意行为的检测机制, 这将导致全局模型容易遭受部分恶意节点的攻击而最终表现出较低的预测性能. 在 BFL 系统中, 设计合适的挖矿奖励函数, 能够有效地激励节点上传合法的模型更新参数. 作为对全局模型贡献的量化评估, 累计奖励数值的高低, 间接地反映了该节点的合法性. 在 EDFL 中, 不同于传统的单一挖矿奖励模式, 本文提出了一种基于角色工作强度的自适应奖励算法.

(1) 训练者的奖励

在 EDFL 中, 训练者是 FL 系统中的重要组成部分, 经过其训练后的参数, 将直接影响全局模型的预测性能. 因为不同节点的本地数据样本质量参差不齐, 使用低质量或者恶意的模型参数去更新全局模型, 可能会造成预测准确度的严重下降. 因此, 分配给训练者的奖励应根据其上传参数的合法性和训练强度共同决定.

如公式(6)所示, 训练者 t_x 在第 j 轮迭代中获得的预期奖励, 将基于其数据集大小和训练轮数计算得到. 在经过去中心化参数验证后, t_x 上传参数的验证结果 $PT(P_x^{(j)})$ 由委员会验证并记录在区块链上. 因此, 训练者 t_x 最终是否能够获得其预期奖励 $r_x^{(j)}$, 取决于其上传参数的合法性验证结果 $PT(P_x^{(j)})$. 在公式(6)中, s_x 表示 t_x 的本地数据集大小, $epochs$ 表示 t_x 的训练轮数, 奖励单元 $\hat{r}$ 是一个超参数:

$$r_x^{(j)}=\begin{cases}|s_x|\times|epochs|\times\hat{r}, & PT(P_x^{(j)})\geqslant 0.5\\ 0, & PT(P_x^{(j)})<0.5\end{cases} \tag{6}$$

(2) 评估者的奖励

去中心化参数验证对于 EDFL 的可靠运行是至关重要的, 它确保全局模型能够在一定周期内进行良性迭代. 与训练者工作内容不同, 评估者使用代理评估方式来评估接收到的模型参数. 评估者仅需使用私有数据集对本地模型训练一个 $epoch$, 以模拟合法训练者使用本地数据集训练全局模型后会表现出的合理下降[3]. 此外, 给予验证者 e_y 的奖励 $r_y^{(j)}$ 还将基于该节点在委员会中投票的时间顺序. 这不仅缓解了聚合者参数验证进

程中的阻塞问题, 而且提高了评估者执行交叉评估的并行效率. 如算法 1 所示, VQ 是一个存储所有评估者投票顺序的队列, 其由聚合者维护并记录在区块链中. 根据投票的索引值, $r_y^{(j)}$ 利用公式(7)进行求取.

$$r_y^{(j)}=\sum_i^{|T|}\frac{|VQ|-VQ.FetchIndex(v^{(j)}(y,i))}{|VQ|}\times|s_y|\times\hat{r} \tag{7}$$

(3) 聚合者的奖励

与传统区块链中的激励机制不同, EDFL 不采用矿工间的竞争机制来生成区块, 而是在每轮迭代结束时, 由聚合者将本轮生成的中间结果打包并上链. EDFL 给予聚合者较低的奖励值, 一方面因为聚合者的任务是低工作强度的, 其工作不包括模型参数的训练和评估; 另一方面, 给予聚合者较低的奖励有助于验证委员会中节点的流动性. 考虑到聚合者的运算和存储开销主要来自对参数验证哈希表 PT 和评估者投票队列 VQ 的维护, 因此, 其奖励 $r_x^{(j)}$ 利用公式(8)进行计算.

$$r_a^{(j)}=|PT|\times|VQ|\times\hat{r} \tag{8}$$

综上所述, EDFL 为训练者、评估者、聚合者设置了自适应的奖励算法, 反映了其不同角色的工作量. 如图 3 所示, 从训练者到聚合者, 分配对应角色的具体节点数量呈逐渐减少的趋势($|T|>|E|>|A|=1$). 因此, 为了确保所有参与节点能够在不同角色间进行有效切换, EDFL 应当为 3 种角色设置依次递减的奖励函数.

6 实验结果与分析

6.1 实验设置

本文所提 EDFL 框架与最主流的 BFL 框架(VBFL[3])保持相同的实验环境和可比性, 基于 IBM 区块链实现了 EDFL 框架, 在此基础上集成了 LevelDB 用于数据持久化. EDFL 中的算法基于 Python 3.7 的异步编程范式实现, 采用基于 SHA-256 的非对称加密算法对网络中的消息进行签名和验证. 本文实现了基于柯西矩阵的 LRC 纠删编码方法, 包括一个基于有限域上的编码器和解码器.

本文实验采用联邦学习中常用的基准测试数据集 FEMINIST[3,7], 用于实现一个分布式环境下的图像分类任务. FEMINIST 包含 62 种不同字符的灰度图片, 图片大小为 28×28 像素, 样本数为 805 263. 以下部分, 本文将采用联邦学习效率来评估不同 FL 框架在 FEMINIST 分类任务下的工作效率, 其包括全局模型训练效率和区块生成效率两个部分. 其中, 全局模型训练效率的定义为 FL 框架经过一定迭代轮数的训练后, 全局模型的预测准确率. 预测准确率是指模型在运行完一次 FEMINIST 测试集后所表现出的准确率(ACC). 区块生成效率的定义为: FL 框架在完成一次全局模型迭代, 并在区块链上生成新区块所需要的时间(s).

本文实验选用 Vanilla FL[1]、VBFL[3]和 EDFL 这 3 种 FL 框架进行对比. 其中, Vanilla FL 作为 Google 提出的 FL 原型框架, 是目前所有 FL 框架的实现基础; VBFL 是目前综合综合性能最优的 FL 框架, 具有较高的理论研究价值和优秀的实验性能表现. 所有实验都运行在配备了英特尔至强 W-2245 3.90 GHz CPU、英伟达 RTX3090 GPU、64 GB RAM 的服务器上, 网络带宽为 1 GB/s. 对于 FL 模型训练, 设置了与 Vanilla FL[1]和 VBFL[3]相同的实验环境, 模拟了 20 个分布式节点(14 个训练者、5 个评估者、1 个聚合者)的 FL 任务. 整个 FEMNIST 数据集以独立同分布的方式随机分配给 20 个节点. 每个节点使用 MNIST_CNN[1]网络来训练本地模型, 聚合者使用 FedAvg[1]方法来聚合合法的模型参数. FL 模型训练中的学习率(learning rate)设置为 0.01、批大小(batch size)设置为 10、epoch 设置为 5. 此外, 通过设置一定比例的恶意节点来评估系统的安全性, 恶意行为包括向模型中注入方差为 1 的高斯噪声、投票结果取反、在分区恢复过程中拒绝响应其他节点的请求. 所有实验执行 3 次, 将所有实验结果的平均值作为最终实验结果.

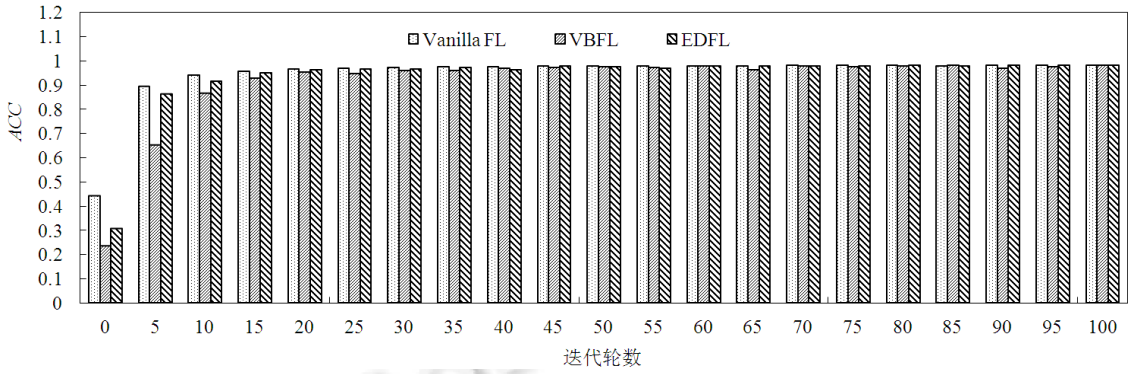
6.2 联邦学习效率

为了评估不同 FL 框架的学习效率, 本文将高效的 FL 框架定义为能够使用更少的迭代轮数训练出具有更高预测性能的全局模型. 因此, 本节主要围绕全局模型预测性能和区块生成效率两个方面进行实验分析.在以下实验中, 将在每个 FL 框架中分别随机注入 5%和 15%的恶意节点, 验证去中心化异步验证机制的有效性.

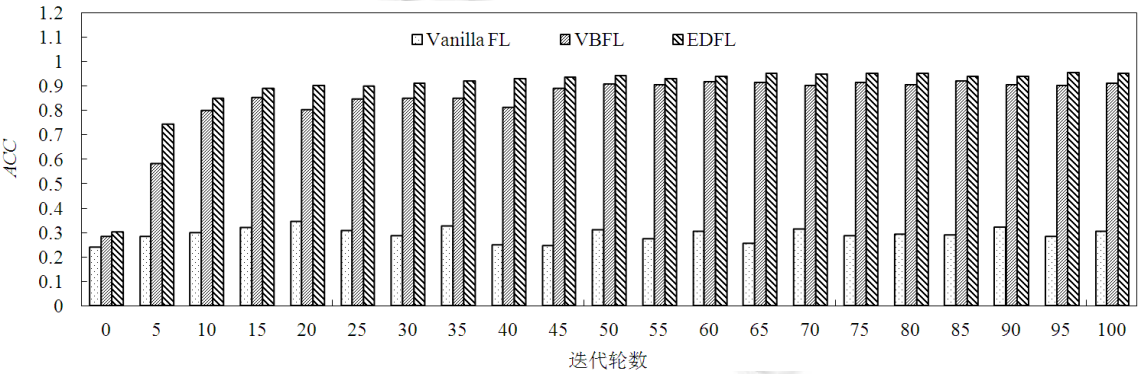
6.2.1 全局模型训练效率

考虑到分布式环境中可能存在的恶意行为, 提出了一种去中心化验证机制来评估各个节点上传模型参数的合法性. 验证委员会对训练者上传的参数进行交叉评估, 以提高 FL 全局模型的鲁棒性.

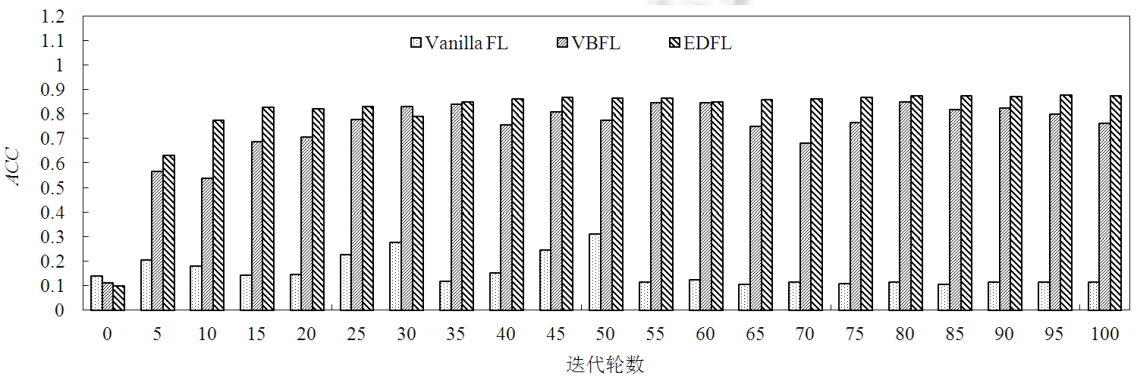
如图 5 所示, 本文在 100 轮迭代中进行了 12 组实验, 并记录下每轮迭代结束时, 全局模型在 FEMNIST 数据集下的预测准确率 *ACC*.



(a) 20 个参与节点, 且无恶意节点

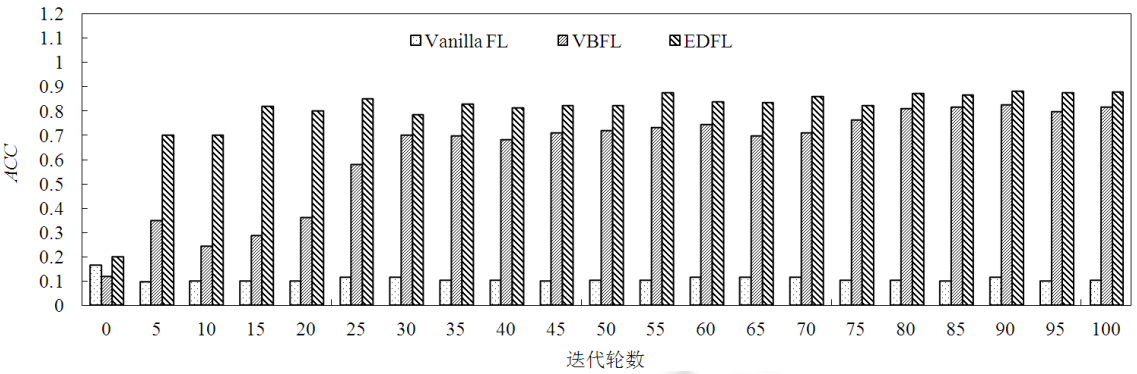


(b) 20 个参与节点, 其中 5%为恶意节点



(c) 20 个参与节点, 其中 15%为恶意节点

图 5 在 100 轮迭代中, 不同 FL 框架的全局模型预测性能对比



(d) 40 个参与节点，其中 15%为恶意节点

图 5　在 100 轮迭代中，不同 FL 框架的全局模型预测性能对比(续)

如图 5(a)所示，在 20 个参与节点均为合法节点的情况下，3 种 FL 框架训练出的全局模型均表现出了良好的预测性能. 在第 20 轮迭代结束时，3 种框架的全局模型均实现了 95%以上的预测准确率，但是三者却有着不同的收敛效率. 对于 Vanilla FL，因为网络中的所有节点都参与了模型训练，所以在不存在恶意节点的情况下，它拥有最好的 FL 学习性能. 对于 VBFL，它通过矿工间的竞争机制来生成新的区块. 这种方法虽然提高了系统的安全性，但也使得更多的合法节点去争夺区块链的记账权而不是进行模型训练. 因此，VBFL 在全局模型更新中消耗了一部分节点的算力，从而表现出较低的 FL 学习效率. 相比之下，EDFL 综合考虑了模型训练准确性和系统鲁棒性两个方面. 基于贡献度证明的委员会共识机制，使得区块生成过程变得更加高效，不依赖设备的算力而是基于节点的历史贡献度来生成合法区块. 如图 5(a)所示，相比于 VBFL，EDFL 在不存在恶意节点的情况下，全局模型准确率更接近 Vanilla FL. 其略差于 Vanilla FL 的原因在于，网络中的所有节点都参与了模型训练，而 VBFL 和 EDFL 中的部分节点用于分布式模型参数的验证.

此外，在网络中存在恶意节点的情况下，得益于模型参数验证机制，EDFL 和 VBFL 中，全局模型的预测性能相较于 Vanilla FL 都存在着明显的优势. 从图 5(b)可以观察到，随着训练轮数的增加，EDFL 表现出更好的稳定性. 因为 EDFL 在前 10 轮迭代中就能有效地识别出恶意节点，并使它们的贡献度始终维持在一个较低的水平. 这使得验证委员会可以更加高效地辨别出合法的模型参数并将其传递给聚合者，减少了恶意节点对全局模型的负面影响. 并且当网络中恶意节点的比例增加时，EDFL 的去中心化异步验证机制能够更加有效地提高全局模型的训练效率和鲁棒性. 对比图 5(b)和图 5(c)可以得出，当恶意节点的比例从 5%增加至 15%，EDFL 经过相同迭代轮数的训练后，其全局模型预测准确率(*ACC*)均高于 VBFL. 此外，通过纵向对比同一框架(恶意节点比例为 5%)的实验数据发现，EDFL(恶意节点比例为 15%)在第 100 轮迭代结束时的 *ACC* 仅下降了 8.65%，而 VBFL(恶意节点比例为 15%)则下降了 16.61%. 这表明，基于贡献度证明共识的验证委员会能够在恶意节点的数量增加时，实现更高的鲁棒性，使得 EDFL 仍保持着较高的全局模型训练效率.

本文还对比了相同恶意节点比例在不同参与节点数量下的全局模型训练效率. 对比图 5(c)和图 5(d)可以得出，当网络中的参与方数量增加时，VBFL 的挖矿竞争机制导致更多的节点参与了区块生成任务，而未进行模型训练任务. 这使得 VBFL 在 40 个参与节点的情况下的训练效率远低于 EDFL. 从图 5(d)可以看出，经过相同轮数的迭代后，EDFL 能够训练出具有更高 *ACC* 的全局模型. 此外，通过纵向对比同一框架(节点数量为 20)的实验数据发现：因为 EDFL 采用了基于贡献度的共识机制和异步的模型参数验证方法，能够在参与节点增加的情况下缓解去中心化验证进程中的阻塞问题，进而有效地提高了全局模型的训练效率.

6.2.2　区块生成效率

评估联邦学习效率的第 2 项工作是分析 EDFL 中的区块生成效率，其中包括参数验证效率和参数聚合效率两个部分. 首先，本文将验证去中心化异步验证机制的有效性，并记录下两种 FL 框架在第 0、25、50、75、100 轮中的参数验证时间.

如图 6 所示, EDFL 的验证时间约为 VBFL 的 0.31 倍. 因为 VBFL 没有对分布式验证机制进行优化, 每轮迭代中都将产生$|T|\times|E|$次参数评估过程. 相比之下, EDFL 采用了非阻塞的异步验证机制, 它能够忽略一部分非必要的参数评估过程, 并实现等效的合法性验证. 得益于贡献度证明共识机制, 随着系统的持续运行, EDFL 的验证时间将不断减少. 具体来说, 第 25 轮验证时间比第 0 轮低 46.1%, 因为在第 25 轮中, 验证委员会基于节点的贡献度能够很容易地识别出网络中的恶意节点, 进一步提高了去中心化验证过程的效率.

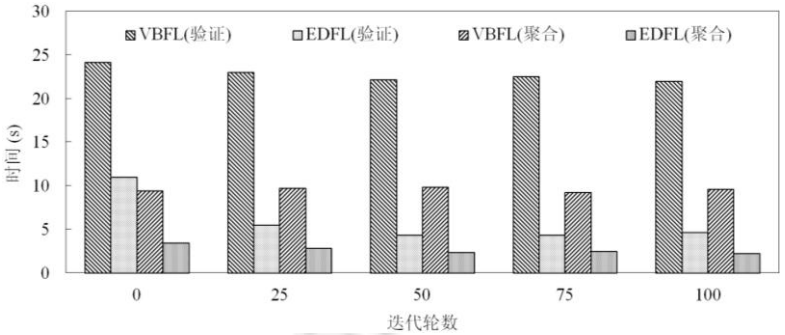


图 6 第 0、25、50、75、100 轮中, 两种 FL 框架的参数验证和参数聚合时间对比

在参数聚合过程中, VBFL 采用基于竞争的共识机制, 要求矿工们相互竞争来获取每轮迭代中的出块权. 如图 6 所示, 在每一轮迭代中, VBFL 需要 3 名矿工(因为其相比于 EDFL, VBFL 采用基于竞争的共识机制来实现更高的安全性, 但是会产生 3 倍的聚合时间)挖掘出属于自己的区块并参与竞争. 然后, 竞争成功的节点将丢弃剩余的 2 个区块. 与之不同, EDFL 采用基于贡献度的共识机制, 在每轮迭代开始时选择具有最高贡献度的节点作为聚合者, 这将极大地减少区块生成过程中由于挖矿竞争所造成的额外聚合延迟.

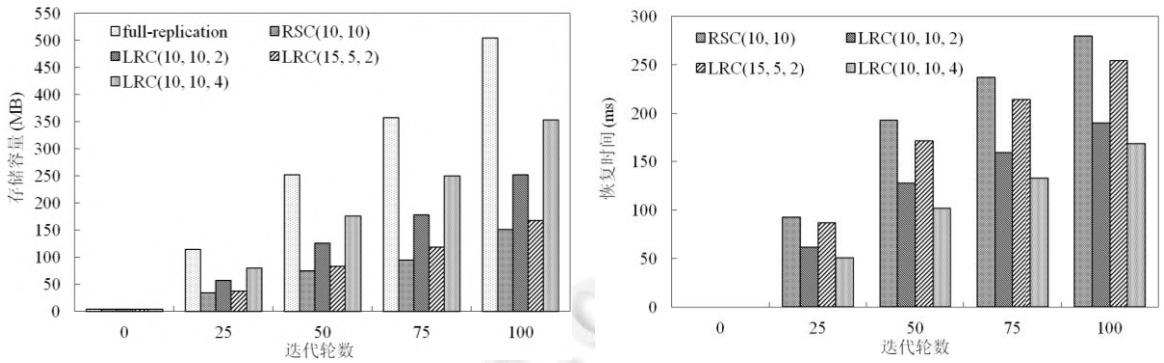
除此之外, 如果某个节点在验证委员会中被累计标记为恶意超过 6 轮, 那么它将被添加到各节点的本地黑名单中. 如图 6 所示, 随着迭代轮数的增加, EDFL 的聚合时间呈现出逐渐下降的趋势. 这是因为在第 25 轮及以后的迭代中, 聚合者根据本地存储的黑名单, 仅需维护具有更小存储容量的 *PT* 和 *VQ*. 此外, 角色自适应激励算法能够提高 EDFL 的安全性, 确保每轮迭代中的区块生成者可以进行有效地轮换, 并实现更高的去中心化. EDFL 的角色自适应奖励算法将在第 6.4 节中进行详细的实验分析.

6.3 存储可扩展性

本节将评估 EDFL 在存储可扩展性方面的性能. 为了验证区块链分区存储策略的有效性, 分别从存储消耗和恢复时间两个方面对 3 种存储策略(全复制、RSC 编码、LRC 编码)进行对比. 因为 RSC 编码与 LRC 编码对于非聚合者节点的存储策略是相同的, 所以在图 7(a)中仅分析具有差异性的聚合者本地存储消耗. 此外, 设置了 3 种不同的 *LRC*(*K*,*M*,*Q*)参数并进行了实验分析, 反映了 EDFL 在本地存储消耗和数据恢复时间之间进行取舍. 所有的分区存储实验都是在每个编码块拥有 3 个副本的情况下进行的(p=3, 类似于 Hadoop[24]中默认的三副本存储策略). 因为引入更多的副本数量将实现更高的容错性和恢复效率, 但会线性地增加节点的本地存储消耗, 所以将不会针对不同的 p 副本数量进行额外的实验.

如第 3 节中所述, EDFL 中每轮生成区块的内容主要由 3 部分组成: 训练者上传的模型参数、评估者的投票集合以及聚合者的聚合结果. 如图 7(a)中的第 0 轮所示, 每个区块的容量大小在 4.6–5.2 MB 之间. 在传统的全复制策略下, 节点的本地存储消耗随着迭代轮数的增加而线性增加, 因为每个节点均在本地保留了完整区块链的副本. 与之相比, EDFL 采用分区存储策略, 每个参与设备上仅需要存储 p 个不同的纠删编码块(聚合者需要存储额外的局部编码块). 如图 7(a)所示, 随着迭代次数的增加, 区块链分区存储策略在存储可扩展性方面的优势相较于全复制策略在逐渐增大. 在第 100 轮结束时, 采用 *LRC*(10,10,2)纠删编码方法的本地存储消耗相比采用全复制策略下降了 51.2%. 与 *RSC*(10,10)编码相比, 在第 100 轮结束时, 采用 *LRC*(10,10,2)编码的本地存储消耗将提升 1.65 倍. 因为 *LRC*(10,10,2)编码在聚合者本地保留了额外的 2 个局部编码块, 使得它能够

在数据恢复阶段更快地恢复原始数据. 此外, 具有更长分区周期 K 的 LRC 编码能够实现更小的本地存储消耗, 因为每个编码块所占的存储容量更小. 例如, 采用 *LRC*(15,5,2)编码的本地存储容量相较于 *LRC*(10, 10,2)平均下降了 32.8%. 但是在数据恢复阶段, 因为前者拥有更大的 K 值, 所以需要请求更多的编码块用于恢复原始数据.



(a) 存储消耗 (b) 恢复时间

图 7 第 0、25、50、75、100 轮中, 不同存储策略的本地存储消耗和恢复时间对比

由上述可知, 采用区块链分区存储策略的系统相较于传统的全复制策略在存储可扩展性方面表现出了显著的优势, 但是它需要花费额外的时间来进行原数据的恢复. 对于图 7(b)中的 4 种不同纠删编码方案, 因为 K 值的不同, 它们的第 1 次分区操作分别发生在第 10 轮和第 15 轮迭代结束时, 所以在第 0 轮结束时, 4 种方案均采用全复制策略, 节点间不会发送数据恢复请求, 所以不会产生恢复延迟. 在 25 轮之后, 所有的方案将请求不同的编码块来执行数据恢复进程. 其中, LRC 在聚合者本地保存了额外的局部编码块, 它能够有效减少恢复请求中消息的转发次数. 并且在收集到足够的数据块后, 能够执行效率更高的局部解码进程. 如图 7(b)所示, 3 种 LRC 编码方案相较于 RSC 编码在数据恢复效率上均存在着一定的优势. 在第 100 轮结束时, *LRC*(10, 10,2)编码的恢复时间比 *RSC*(10,10)下降了 32.8%. 如上文所述, 如果 LRC 编码选择在本地保留更多的局部编码块(例如 *LRC*(10,10,4)编码方案), 它的数据恢复效率将进一步提高. 此外, 具有更长分区周期的 *LRC*(15,5,2)编码方案在数据恢复阶段需要请求更多的编码块, 所以相较于 *LRC*(10,10,2), 需要花费更长的恢复时间.

综上所述, 区块链分区存储策略能够显著降低 FL 参与节点的本地存储消耗, 但是它需要花费额外的时间去恢复原始数据. 如果想要提高数据的恢复效率, 可以适当调低 *LRC*(*K*,*M*,*Q*)编码中的参数 K 并调高参数 Q, 同时也会增加系统的存储消耗, 所以, 分区存储策略需要根据具体的实际应用进行参数调整.

6.4 安全性

在本节中, 将分析两种 FL 框架的奖励算法在恶意节点识别有效性和去中心化程度两个方面的性能. 如图 8 所示, EDFL 的角色自适应奖励算法能够有效地识别出网络中的恶意节点. 经过 100 轮迭代后, 3 个恶意节点(n_3, n_5, n_8)所获得的累计奖励均保持在一个较低的水平, 它们与拥有最低奖励的合法节点(n_{16})之间仍保持着平均 9.54 倍的差距, 这使得恶意节点能够更明显地被检测出来并加入其他节点的黑名单中. 相比之下, 因为没有采用基于贡献度证明的委员会共识机制, VBFL 中部分的恶意节点(n_5)依然能够在参数验证过程中获得奖励. 此外, 因为 VBFL 在分布式参数验证过程中的验证结果仅基于投票的数量而并未引入节点贡献度作为投票的权值, 使得恶意的评估者在验证阶段蓄意地将部分合法节点(n_{19})的训练工作评价为不合法的, 从而出现合法节点被提前加入黑名单的情况.

为了更加具体地分析累计奖励对于系统安全性的影响, 本文使用两种评价指标来衡量两种 FL 框架中各节点的奖励分布. 本文定义标准差 s 为累计奖励排名前三的节点的奖励数值的标准差. 在每轮迭代中, 标准差 s 反映了系统能否有效地进行区块生成者的轮换. 如图 8 所示, 若一个系统拥有较高的标准差 s, 表明该系统

中存在中心节点的可能性更大. 一旦某个节点能够在连续数轮迭代, 可预期地当选为区块的生成者, 会使得系统中的共识结果受到“non-democracy side effect”影响[25], 这违背了区块链的去中心化特性. 例如, 持有过高奖励的节点连续且可预期地独占区块链中的记账权, 那么在该系统中发生中心节点被攻击或者成为恶意节点的可能性将大大增加. 该系统将面临与主从架构相同的安全性问题, 使得聚合者容易遭受外部攻击, 或者独占区块的生成权利, 从而导致 FL 中的全局模型无法被及时且合法地更新.

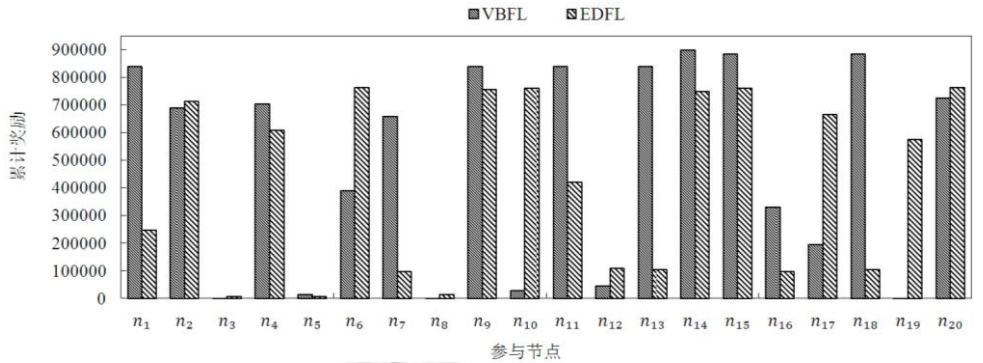
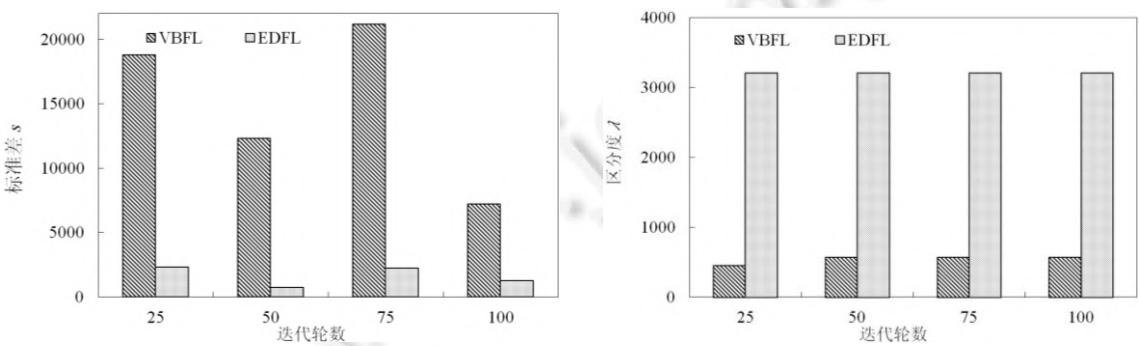


图 8 在第 100 轮迭代结束时, 两种 FL 框架中, 20 个节点累计奖励的对比
(其中, n_3, n_5, n_8 为恶意节点)

对于恶意节点识别的有效性, 本文定义区分度λ为 3 个拥有最低奖励的合法节点累计奖励的几何平均数与 3 个恶意节点累计奖励的几何平均数之间的比值, 其量化了合法节点和恶意节点之间的可区分性. 如图 9(b)所示, 拥有较高区分度λ的 FL 框架能够在系统运行的前期有效地识别出网络中的恶意节点, 并将其列入黑名单. 这同样解释了图 5 中 EDFL 比 VBFL 拥有更高的 FL 学习准确率的原因, 因为 EDFL 的验证委员会能够在前 10 轮迭代结束时识别出全部的恶意节点, 并在后续的迭代中显著降低其对全局模型的负面影响. 综上所述, 如图 9 所示, EDFL 拥有着更低的标准差 s (平均为 VBFL 的 0.12 倍)以及更高的区分度λ (平均为 VBFL 的 5.91 倍). 这表明, EDFL 相较于 VBFL, 在去中心化程度和恶意节点识别有效性方面存在着显著优势.



(a) 存储消耗 (b) 恢复时间

图 9 第 25、50、75、100 轮, 不同 FL 框架中的累计奖励标准差 s 和区分度λ的对比

7 结论与展望

在大数据背景下, 为了提高数据联邦中的学习效率并降低节点的本地存储消耗, 提出了一种高效的去中心化联邦学习框架 EDFL. 首先, 提出了一种基于贡献度证明的委员会共识机制, 并集成了角色自适应奖励算法, 使得 EDFL 能够增强 FL 全局模型鲁棒性, 并提高全局模型的训练效率. 其次, 为了减少节点的本地存储

消耗, 提出一种新的区块链分区存储策略, 通过多副本的 LRC 编码块均匀分区方法, 能够将本地存储消耗降低至常数水平, 并提升原始数据的恢复效率. 最后, 在真实的 FEMNIST 数据集上, 对 EDFL 的学习效率、存储可扩展性和安全性进行了评估实验. 结果表明, 与最先进的基于区块链的 FL 框架 VBFL 相比, EDFL 能够在更低的系统消耗上实现更高的 FL 学习准确率. 同时, 所提出的角色自适应奖励算法在恶意节点识别和去中心化方面取得了更好的性能.

未来的工作包括: (1) 利用消息队列技术将去中心化参数验证机制扩展到多轮全局模型迭代, 进一步提高 EDFL 中参数验证进程的并发性; (2) 在验证委员会中引入聚合者工作的合法性评估方法, 进一步提高 EDFL 的安全性; (3) 设计一种纠删码流水线修复(repair pipelining)机制, 减少解码过程中数据恢复时间.

致谢 感谢所有参与本课题研究和本文工作但没有在文中署名的课题组成员以及对本项目提供技术支持的专家和学者; 感谢所有评阅本文的匿名评审人以及他们对本文提出的宝贵修改意见.

References:

[1] McMahan B, Moore E, Ramage E, Hampson S. Communication-efficient learning of deep networks from decentralized data. In: Proc. of the 20th Int'l Conf. on Artificial Intelligence and Statistics. PMLR, 2017. 1273−1282.

[2] Zhou P, Wang KH, Guo LK, Gong SM. A privacy-preserving distributed contextual federated online learning framework with big data support in social recommender systems. IEEE Trans. on Knowledge and Data Engineering, 2021, 33(3): 824−838. [doi: 10.1109/TKDE.2019. 2936565]

[3] Chen H, Asif SA, Park J, Shen CC, Bennis M. Robust block chained federated learning with model validation and proof-of-stake inspired consensus. In: Proc. of the 35th AAAI Conf. on Artificial Intelligence. AAAI, 2021. 1−9.

[4] Nakamoto S. Bitcoin: A Peer-to-peer Electronic Cash System. Bitcoin White Paper, 2008.

[5] Sheller M, Edwards B, Reina GA, Martin J. Federated learning in medicine: Facilitating multi-institutional collaborations without sharing patient data. Scientific Reports, 2020, 10(1): 1−12. [doi: 10.1038/s41598-020-69250-1]

[6] Awan S, Li FJ, Luo B, Liu M. Poster: A reliable and accountable privacy-preserving federated learning framework using the blockchain. In: Proc. of the 14th ACM SIGSAC Conf. on Computer and Communications Security. New York: ACM, 2019. 2561−2563. [doi: 10.1145/3319535.3363256]

[7] Li YZ, Chen C, Liu N, Huang H. A blockchain-based decentralized federated learning framework with committee consensus. IEEE Network, 2021, 35(1): 234−241. [doi: 10.1109/MNET.011.2000263]

[8] Ramanan P, Nakayama K. BAFFLE: Blockchain based aggregator free federated learning. In: Proc. of the IEEE Int'l Conf. on Blockchain. Washington: IEEE, 2020. 72−81. [doi: 10.1109/Blockchain50366.2020.00017]

[9] Zhang ZB, Dong DJ, Ma YH, Ying YL. Refiner: A reliable incentive-driven federated learning system powered by blockchain. Proc. of the VLDB Endowment, 2021, 14(12): 2659−2662. [doi: 10.14778/3476311.3476313]

[10] Bonawitz K, Eichner H, Grieskamp W. Towards federated learning at scale: System design. In: Proc. of the 3rd Conf. on Machine Learning and Systems. MLSys, 2019. 374−388.

[11] Islam S, Sobhan S, Valero M, Shahriar H. Framework for collecting data from specialized IoT devices—An application to enhance healthcare systems. In: Proc. of the IEEE Int'l Conf. on Digital Health. Washington: IEEE, 2021. 231−233. [doi: 10.1109/ICDH52753.2021.00045]

[12] Lyu L, Yu JS, Nandakumar K, Li YT. Towards fair and privacy-preserving federated deep models. IEEE Trans. on Parallel Distributed Systems, 2020, 31(11): 2524−2541. [doi: 10.1109/TPDS.2020.2996273]

[13] Fox A, Brewer E. Harvest, yield and scalable tolerant systems. In: Proc. of the 7th Workshop on Hot Topics in Operating Systems. Washington: IEEE, 1999. 174−178. [doi: 10.1109/HOTOS.1999.798396]

[14] Kim H, Park J, Bennis M, Kim S. Blockchained on-device federated learning. IEEE Communications Letters, 2020, 24(6): 1279−1283. [doi: 10.1109/LCOMM.2019.2921755]

[15] Kang JW, Xiong ZH, Niyato D, Wang P. Incentivizing consensus propagation in proof-of-stake based consortium blockchain networks. IEEE Wireless Communications Letters, 2019, 8(1): 157−160. [doi: 10.1109/LWC.2018.2864758]

[16] Kang JW, Xiong ZH, Niyato D, Zou YZ. Reliable federated learning for mobile networks. IEEE Wireless Communications, 2020, 27(2): 72−80. [doi: 10.1109/MWC.001.1900119]

[17] Dinh T, Liu R, Zhang MH, Chen G, Ooi B, Wang J. Untangling blockchain: A data processing view of blockchain systems. IEEE Trans. on Knowledge and Data Engineering, 2018, 30(7): 1366−1385. [doi: 10.1109/TKDE.2017.2781227]

[18] Qi XD, Zhang Z, Jin CQ, Zhou AY. A reliable storage partition for permissioned blockchain. IEEE Trans. on Knowledge and Data Engineering, 2021, 33(1): 14−27. [doi: 10.1109/TKDE.2020.3012668]

[19] Tao YC, Li B, Jiang JJ, Ng HC, Wang C, Li BC. On sharding open blockchains with smart contracts. In: Proc. of the 36th IEEE Int'l Conf. on Data Engineering. Washington: IEEE, 2020. 1357−1368. [doi: 10.1109/ICDE48307.2020.00121]

[20] Dang H, Dinh TTA, Loghin D, Chang EC, Lin Q, Ooi B. Towards scaling blockchain systems via sharding. In: Proc. of the ACM SIGMOD/PODS Conf. New York: ACM, 2019. 123−140. [doi: 10.1145/3299869.3319889]

[21] Weatherspoon H, Kubiatowicz J. Erasure coding vs. replication: A quantitative comparison. In: Proc. of the 1st Int'l Workshop on Peer-to-peer Systems. Berlin: Springer, 2002. 328−338. [doi: 10.1007/3-540-45748-8_31]

[22] Reed I, Solomon G. Polynomial codes over certain finite fields. Journal of the Society for Industrial and Applied Mathematics, 1960, 8(2): 300−304. [doi: 10.2307/2098968]

[23] Huang C, Simitci H, Xu YK. Erasure coding in windows azure storage. In: Proc. of the USENIX Annual Technical Conf. Boston: USENIX Association, 2012. 15−26.

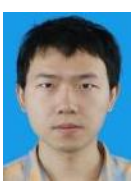
[24] Ghemawat S, Gobioff H, Leung S. The Google file system. In: Proc. of the 19th ACM Symp. on Operating Systems. New York: ACM, 2003. 29−43. [doi: 10.1145/945445.945450]

[25] Qu XD, Wang SL, Hu Q, Cheng XZ. Proof of federated learning: A novel energy-recycling consensus algorithm. IEEE Trans. on Parallel Distributed Systems, 2021. 32(8): 2074−2085. [doi: 10.1109/TPDS.2021.3056773]

乔少杰(1981－), 男, 博士, 教授, CCF 杰出会员, 主要研究领域为区块链数据库, 人工智能数据库.

林羽丰(1998－), 男, 硕士生, 主要研究领域为区块链, 联邦学习.

韩楠(1984－), 女, 博士, 副教授, 主要研究领域为数据库, 数据挖掘.



杨国平(1997－), 男, 硕士生, CCF 学生会员, 主要研究领域为数据库查询优化.

李贺(1983－), 男, 博士, 副教授, CCF 专业会员, 主要研究领域为数据挖掘, 机器学习, 大数据处理.

袁冠(1982－), 男, 博士后, 教授, CCF 高级会员, 主要研究领域为数据挖掘, 机器学习.

毛睿(1975－), 男, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为大数据.

元昌安(1964－), 男, 博士, 教授, 博士生导师, CCF 专业会员, 主要研究领域为数据库.

Louis Alberto GUTIERREZ(1980 －), 男, 博士, Researcher, 主要研究领域为人工智能.