软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2024,35(3):1418−1439 [doi: 10.13328/j.cnki.jos.006819]

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

理性与可验证的联邦学习框架*

吴柿红[1,2], 田有亮[1,2,3]

[1](公共大数据国家重点实验室 (贵州大学), 贵州 贵阳 550025)

[2](贵州大学 计算机科学与技术学院, 贵州 贵阳 550025)

[3](贵州大学 密码学与数据安全研究所, 贵州 贵阳 550025)

通信作者: 田有亮, E-mail: youliangtian@163.com

摘　要: 联邦学习作为解决数据孤岛问题的有效方法, 在服务器计算全部梯度的过程中, 由于服务器的惰性和自利性会存在全局梯度不正确计算问题, 因此需要验证全局梯度的完整性. 现有的基于密码算法的方案验证开销过大. 针对这些问题, 提出一种理性与可验证的联邦学习框架. 首先, 结合博弈论, 设计囚徒合约与背叛合约迫使服务器诚实. 其次, 所提方案使用基于复制的验证方案实现全局梯度的完整性验证, 且支持客户端离线. 最后, 经分析证明所提方案的正确性, 并经实验表明, 该方案与已有的验证算法相比, 客户端的计算开销降为 0, 一次迭代的通信轮数由原来的 3 轮优化到 2 轮, 且训练开销与客户端的离线率成反比.

关键词: 联邦学习; 博弈论; 囚徒合约; 背叛合约; 数据完整性

中图法分类号: TP18

中文引用格式: 吴柿红, 田有亮. 理性与可验证的联邦学习框架. 软件学报, 2024, 35(3): 1418–1439. http://www.jos.org.cn/1000-9825/6819.htm

英文引用格式: Wu SH, Tian YL. Rational and Verifiable Federated Learning Framework. Ruan Jian Xue Bao/Journal of Software, 2024, 35(3): 1418–1439 (in Chinese). http://www.jos.org.cn/1000-9825/6819.htm

Rational and Verifiable Federated Learning Framework

WU Shi-Hong[1,2], TIAN You-Liang[1,2,3]

[1](State Key Laboratory of Public Big Data (Guizhou University), Guiyang 550025, China)

[2](College of Computer Science and Technology, Guizhou University, Guiyang 550025, China)

[3](Institute of Cryptography & Date Security, Guizhou University, Guiyang 550025, China)

Abstract: Federated learning is an effective method to solve the problem of data silos. When the server calculates all gradients, incorrect calculation of global gradients exists due to the inertia and self-interest of the server, so it is necessary to verify the integrity of global gradients. The existing schemes based on cryptographic algorithms are overspending on verification. To solve these problems, this study proposes a rational and verifiable federated learning framework. Firstly, according to game theory, the prisoner contract and betrayal contract are designed to force the server to be honest. Secondly, the scheme uses a replication-based verification scheme to verify the integrity of the global gradient and supports the offline client side. Finally, the analysis proves the correctness of the scheme, and the experiments show that compared with the existing verification algorithms, the proposed scheme reduces the computing overhead of the client side to zero, the number of communication rounds in one iteration is optimized from three to two, and the training overhead is inversely proportional to the offline rate of the client side

Key words: federated learning; game theory; prisoner's contract; betrayal contract; data integrity

---

* 基金项目: 国家自然科学基金 (61662009, 61772008); 贵州省科技重大专项 (20183001); 国家自然科学基金联合基金 (U1836205); 贵州省科技计划 (黔科合基础 [2019]1098); 贵州省高层次创新型人才项目 (黔科合平台人才 [2020]6008); 贵阳市科技计划 (筑科合 [2021]1-5)
收稿时间: 2021-10-09; 修改时间: 2022-01-02, 2022-05-01; 采用时间: 2022-06-15; jos 在线出版时间: 2023-06-14
CNKI 网络首发时间: 2023-06-15

机器学习在工业生产[1], 自动驾驶[2,3], 医疗卫生[4], 零售业等行业中得到了广泛应用. 为了充分利用各个企业的数据进行模型训练从而得到更精确的结果同时保护数据的隐私, 联邦学习应运而生.

联邦学习作为解决数据孤岛问题的有效方法, 不需要把数据进行汇集就可实现把存储在不同设备上的数据充分利用起来进行模型训练[5]. 由于需要将局部梯度置于服务器进行聚合, 不可避免会导致一些问题[6], 例如: 通信开销大[7]和安全性弱[8]. 为降低模型的通信开销, Yin 等人[9]提出在梯度差达到设定阈值时只提交梯度差的方法减少通信的轮次, Sattler 等人[10]提出稀疏的二元压缩算法降低通信成本, Aji 等人[11]使用交换稀疏更新代替密集更新加快随机梯度下降的速度; 另一方面, 为保护训练数据的隐私不被泄露, 目前已存在的方案有的通过使用同态加密对局部梯度进行加密[12,13], 有的引入差分隐私对局部梯度进行扰动[14,15]或者基于安全多方计算来实现对局部梯度的保护[16,17], 但是这些方案以及联邦学习框架本身都是建立在服务器诚实的基础上, 如果服务器不诚实, 在服务器上聚合的全局梯度将不会有价值. 由于服务器本身具有惰性与自利性, 不会对局部梯度进行正确聚合, 而是任意选择随机梯度返回客户端. 如何约束服务器的行为, 解决联邦学习中数据的完整性问题仍是联邦学习的挑战之一[6].

针对数据完整性验证问题, 已发表的文献大多是基于密码算法[18,19], 客户端在本地计算局部梯度的哈希, 并使用单个服务器向其他客户端传递该哈希值, 在客户端对所有客户端的哈希值进行聚合, 通过与全局梯度的哈希进行对比, 如果结果不正确, 密码算法能确保客户端以很高的概率拒绝该全局梯度. 但是, 基于密码算法的验证开销远远高于任务本身开销[20,21]. 基于复制的方法需要至少两台服务器执行相同的计算任务, 客户端仅需要向服务器提供相同任务的副本, 通过对比执行相同任务的服务器计算结果的一致性判断服务器是否正确执行计算任务[22], 该方法能够弥补基于密码算法的不足, 不需要客户端提供额外的计算开销用于验证, 其主要的开销来源于通信开销, 与基于密码算法的验证方法相比, 基于复制的验证方法更实际、效率更高、实用性更强、计算开销更低[23,24]. 本文基于复制的方法, 提出一种理性与可验证的联邦学习框架, 使用两台服务器承担聚合任务, 在以太坊上创建智能合约, 约束服务器行为, 通过对两份全局梯度进行对比实现全局梯度的完整性验证. 本文的贡献如下.

1) 提出理性与可验证的联邦学习框架. 基于复制的验证方式, 使用双服务器执行全局梯度的计算任务, 结合博弈论提出囚徒合约对理性服务器的行为进行约束, 引入背叛合约避免服务器间的合谋, 让服务器向能使自身收益最大化的方向选择策略, 进而实现联邦学习中全局梯度的完整性验证.

2) 设计支持客户端离线的方案. 通过只使用公共客户端的局部梯度作为计算任务的输入实现客户端离线. 方案允许客户端以一定的比率离线, 提高模型的可扩展性.

3) 对方案的正确性进行分析与证明, 经实验表明该方案在联邦学习框架中的可行性, 与已有的验证算法相比, 客户端的计算开销由 $O(di)$ 降为 0, 一次迭代的通信轮数由 3 轮优化到 2 轮, 且训练开销与客户端的离线率成反比.

1 相关工作

2016 年, 谷歌给出联邦学习的定义——将训练数据分散在客户端, 通过将本地更新进行聚合来训练全局模型[25]. Yang 等人[5]对联邦学习做出了更全面的介绍, 其中包含联邦学习的架构和应用. 由于联邦学习是机器学习的分布式训练方式, 其中的局部梯度上传阶段与模型训练阶段不可避免会造成安全问题, 周俊等人[26]对联邦学习中的安全与隐私保护问题进行总结, 着重分析联邦学习中存在的投毒攻击, 对抗攻击和隐私泄露问题. 其一, 针对投毒攻击, Cao 等人[27]利用正常数据与中毒数据间梯度的欧几里德距离具有明显差距的特性划分正常与中毒客户端, 方案中女巫可以操纵梯度的大小以实现差异性, 但梯度的方向无法在不降低攻击有效性的情况下改变. 其二, 针对对抗攻击, Zantedeschi 等人[28]使用数据处理技术引入平滑空间滤波和标量量化测试对抗样本. 其三, 针对隐私泄露问题, Zhou 等人[29]将同态加密与差分隐私相结合, 对训练数据进行扰动, 使用盲因子与 Paillier 算法对局部梯度进行保护.

联邦学习中的安全问题不仅于此[26], Mothukuri 等人[6]对联邦学习中现阶段存在的问题进行更详尽的描述, 除

文献 [26] 中的问题外, 全局梯度的完整性验证也是联邦学习中的一大挑战. 目前, 对联邦学习中全局梯度完整性验证的研究较少, 存在的解决全局梯度完整性验证问题的方法主要基于密码算法. Xu 等人[18]提出使用零知识证明与同态哈希相结合验证服务器计算结果的完整性, 方案中, 客户端计算局部梯度的同态哈希并上传服务器, 服务器对验证信息进行聚合并把聚合后的信息传递给各个客户端, 最终通过在本地对比判断全局梯度是否正确. 然而基于维度的零知识证明验证通信开销高, 其通信成本线性依赖于梯度的维度, 现实状况下训练得到的梯度是高维度的. 针对方案 [18] 中存在的问题, Guo 等人[19]提出使用线性同态散列与承诺方案相结合, 通过摊销降低计算开销且该方案的通信开销独立于梯度的维度, 方案中, 客户端在上传局部梯度前向其他客户端传递其梯度向量的线性同态哈希以及哈希的承诺字符串, 通过接收到的哈希与随机数解除承诺, 判断全局梯度是否进行正确聚合. 但是该方案中, 由于秘密重建算法的开销大, 数据的批量大小不能设置太大, 否则会出现严重的离线客户端数据的积累; 其次, 客户端需要生成自己的承诺字符串, 该承诺字符串需要在其他客户端与该客户端之间往返传递, 这意味着, 发送承诺字符串的通信开销与客户端数量呈正相关. 该方案加入一轮数据传输, 通过秘密共享方案实现客户端离线, 计算开销再次增加.

除通过密码学方法对数据完整性进行验证外, 在委托计算中, 理性的服务器被广泛使用[22,30], 存在方案通过复制的方式对数据的完整性进行验证, 2017 年 Dong 等人[21]针对外包计算中如何验证服务器计算结果的正确性问题提出把客户端的计算任务委托给两台服务器, 通过服务器返回的结果是否相等来验证服务器计算结果的正确性. 该方案利用服务器的自利性设计囚徒合约迫使服务器诚实, 囚徒合约由服务器与客户端签订, 合约规定, 签订合约需要支付押金, 服务器诚实计算将获得奖励, 同时押金将会被退回, 不诚实者押金将会被没收. 方案证明服务器可以通过合谋获得奖励并避开惩罚和计算开销, 针对服务器间的合谋进一步设计背叛者合约, 鼓励服务器向客户端报告合谋, 如果确实存在服务器不诚实, 则诚实服务器将获得奖励, 此时, 两台服务器都知道当他们发起合谋时, 另一台服务器一定会报告, 因此, 背叛者合约对合谋造成了威胁, 服务器不会合谋, 最终通过服务器返回结果的一致性判断计算结果的正确性. 最后在太坊上建立智能合约实现囚徒合约与背叛者合约. 本文把双服务器思想应用于联邦学习, 设计符合联邦学习场景的囚徒合约与背叛合约迫使服务器诚实执行聚合任务, 同时, 方案也支持客户端离线, 其次, 使用博弈论分析服务器的行为, 最后在以太坊上建立智能合约, 约束服务器行为, 最终实现全局梯度的完整性验证.

2 基础知识

本文所提方案主要基于博弈论与联邦学习, 下面就相关概念和基本知识进行介绍.

2.1 联邦学习

联邦学习可以实现各个客户端不将数据汇集就可用客户端的数据进行模型训练. 具体操作是各个客户端从服务器下载全局参数并使用自己的本地数据进行模型训练, 把训练得到的局部梯度上传中央服务器, 中央服务器聚合各个客户端的局部梯度后把全局梯度传送给客户端进行下一轮的模型训练, 直到模型收敛.

神经网络可以表示为函数 $f(x,w)=y'$, w 为模型参数, x 为客户端的输入, y' 表示使用参数 w 和函数 f 得到的输出. 整个训练集表示为 $D=\{\langle x_i,y_i\rangle, i=1,2,\ldots,T\}$, T 表示数据条数, 数据集 D 的平均损失函数公式[18]如下:

$$\xi_f(D,w)=\frac{1}{|D|}\sum_{(x_i,y_i)\in D}\xi_f(x_i,y_i,w) \tag{1}$$

对于特定的损失函数 l, $\xi_f(x,y,w)=l(y,f(x,w))=l(y,y')$.

联邦学习的训练目标是通过改变 w 最小化损失函数, 其每轮迭代的计算公式[18]如下:

$$w^{j+1}=w^j-\lambda\Delta\xi_f(D^j,w^j) \tag{2}$$

其中, λ 表示学习率, w^{j+1} 表示第 $j+1$ 轮训练后的模型参数, D^j 表示从数据集 D 中随机选择的子集.

在服务器端, 对每个客户端进行平均梯度聚合, 公式[25]如下:

$$w^{j+1}=\frac{\sum_{j=1}^{N}\left|D_i^j\right|w_i^j}{\sum_{j=1}^{N}\left|D_i^j\right|} \tag{3}$$

其中, w_i^j 表示第 j 轮训练后客户端 i 的模型参数, w^{j+1} 表示第 $j+1$ 轮训练后的全局模型, D_i^j 表示第 j 轮训练客户端 i 参与训练的数据集, D_i 表示客户端 i 的本地数据集, $i=1,2,\ldots,N$, $D=\{D_1,D_2,\ldots,D_N\}$ 表示联合数据集, N 表示客户端的数量.

2.2 博弈论

定义 1. 完全不完美信息动态博弈 $G=\{P,A,B,E,I,U\}$, P 为玩家集合, A 为玩家在信息集上的一组有限的可选行动, B 为非终端, E 为终端, I 为信息集, U 为收益函数.

定义 2. 在博弈 G 中, 信念系统 $\beta=(\beta_i)_{i\in P}$ 的解释为: 对于每个玩家 $P_i\in P$, β_i 为信息集 $I_{ij}\in I_i$ 上的每个节点分配概率, 对于节点 $b\in I_{ij}$, 玩家 P_i 的 $\beta_i(b)=\Pr[b|I_{ij}]$.

定义 3. 在博弈 G 中, 玩家 $P_i\in P$ 的策略 s_i 意味着对于每个信息集 $I_{ij}\in I_i$, 在玩家 P_i 的每个可选行动集 $A_{ij}\in A_i$ 上分配概率.

定义 4. 在博弈 G 中, 策略集 s 是所有玩家策略 $s=(s_i)_{i\in P}$ 的列表, 例如: 一个策略集可以表示为 $s=(s_i,s_{-i})$, 其中 $s_{-i}=(s_1,s_2,\ldots,s_{i-1},s_{i+1},\ldots,s_n)$, n 表示玩家数量.

定义 5. 在博弈 G 中, 玩家 P_i 在策略集 s 中的非终端节点 b 的收益为他可达的终端集 e 中各节点收益的和, 即:

$$u_i(s,b)=\sum_{e\in E}u_i(e)\cdot\Pr[e|(s,b)] \tag{4}$$

其中, $\Pr[e|(s,b)]$ 为当策略集为 s 时从节点 b 到达终端节点 e 的概率, $u_i(e)_{e\in E}$ 表示玩家 P_i 在可达终端节点上的收益.

定义 6. 在博弈 G 中, 假设信念系统为 β, 当玩家 P_i 到达信息集 I_{ij} 时, 他的收益为处于信息集 I_{ij} 中所有节点 $b\in I_{ij}$ 收益的和, 即:

$$u_i(s,\beta,I_{ij})=\sum_{b\in I_{ij}}\beta_i(b)\cdot u_i(s,b) \tag{5}$$

定义 7. 在博弈 G 中, 当存在一系列完全混合的策略 $(s^k)_{k\in N}$ 收敛于 s, 源于 $(s^k)_{k\in N}$ (使用贝叶斯法则) 的一系列信念 $(\beta^k)_{k\in N}$ 收敛于信念系统 β, 则判断 (s,β) 是序贯一致的, 即:

$$\begin{cases}\lim\limits_{k\to\infty}s^k\to s\\ \lim\limits_{k\to\infty}\beta^k\to\beta\end{cases} \tag{6}$$

定义 8. 在博弈 G 中, 信念系统为 β, 对于任意玩家 P_i, 当 $\forall s_i'\neq s_i, u_i((s_i',s_{-i}),\beta,I_{ij})\leqslant u_i(s,\beta,I_{ij})$ 时, 则称策略集 $s=(s_i,s_{-i})$ 在信息集 I_{ij} 处是理性的.

定义 9. 对于任意玩家 $P_i\in P$ 和任意信息集 $I_{ij}\in I$, 策略集 s 在信息集 $I_{ij}\in I$ 上是理性的, 则称 (s,β) 为序贯理性的.

定义 10. 在博弈 G 中, 当 (s,β) 是序贯理性且序贯一致的, 则称 (s,β) 序贯均衡[31].

定义 11. 在博弈 G 中, $G(T)$ 表示 G 重复进行 T 次的重复博弈, 玩家 P_i, 在下一次博弈开始前, 之前阶段的博弈都被观测到. 如果 G 有唯一的均衡, 则 $G(T)$ 的收益为 G 进行 T 次博弈收益的简单相加[32]. 即:

$$u(G(T))=T\cdot u(G) \tag{7}$$

为更形象地表示动态游戏[32], 使用博弈的扩展式对游戏进行描述, 博弈的扩展式事实上是树形结构, 即博弈树.

3 系统模型

3.1 系统框架

系统框架由 3 个部分组成, 分别是区块链 (智能合约载体), 客户端, 服务器, 如图 1 所示.

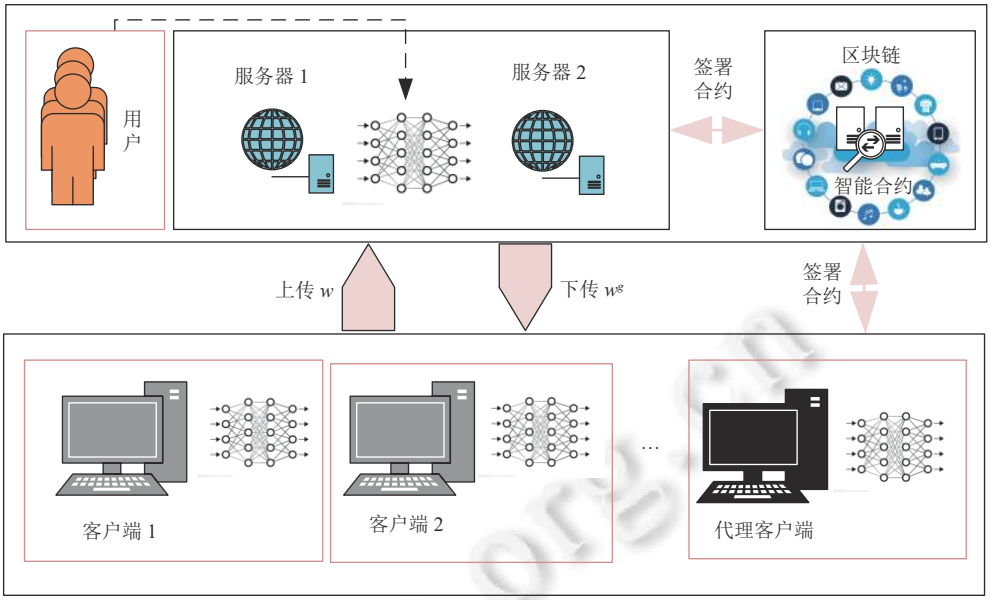

图 1 系统框架

(1) 区块链: 为客户端和服务器提供合约, 存放合约押金.

(2) 客户端: 客户端分为普通客户端 (客户端) 和代理客户端.

1) 普通客户端: 向服务器上传训练梯度, 并使用本地数据对模型进行训练.

2) 代理客户端: 代表所有客户端与两服务器签署合约, 同时使用本地数据对模型进行训练并向服务器上传训练梯度, 此外, 在存在客户端离线时传递公共客户端名单.

(3) 服务器: 系统中有两台服务器参与模型训练, 负责计算全局梯度, 与客户端签署合约, 同时根据收益最大化原则选择是否向另一服务器签署合约.

3.2 敌手模型与假设

(1) 由于服务器的自利性和惰性, 支付服务器费用让其执行任务时, 服务器为避开计算开销会随机返回不经计算的值. 因此, 服务器的自利性和惰性是联邦学习中服务器正确计算全局梯度的威胁, 本文使用智能合约约束服务器的行为, 使服务器诚实计算全局梯度.

(2) 基于加密货币在以太坊上建立智能合约, 使用博弈论建立博弈模型求解唯一的纳什均衡, 使各方参与者选择能使自身收益最大化的策略. 本文设计囚徒合约, 服务器在模型训练前需要签署该合约, 只有服务器诚实计算全局梯度时才会得到奖励, 相反, 不诚实计算将受到惩罚, 从而约束服务器行为, 但是由于计算会产生开销. 服务器为了使自身的收益最大, 在寻找与他执行相同任务的服务器的开销低于计算开销时, 服务器倾向于以合谋的方式避开计算开销得到更高收益. 合谋的两服务器协商一个合理的值 (实际上, 产生合理的合谋值会产生开销, 在模型中, 假设产生合谋值不产生开销) 返回给客户端, 客户端接受服务器传来的相同且合理的结果并支付服务器奖励, 因此, 合谋将成为囚徒合约的威胁, 方案通过允许服务器背叛抑制合谋合约的签署.

4 方案设计

方案基于博弈论设计囚徒合约约束服务器的自利与惰性行为, 通过背叛合约阻止服务器间的合谋, 使用两台服务器执行相同的计算任务, 即对相同客户端的局部梯度进行聚合, 通过比较服务器返回结果的一致性判断服务器是否诚实 (服务器返回结果相同则两服务器都诚实, 返回结果不相同则存在不诚实的服务器). 从而实现全局梯度的完整性验证. 表 1 为本文涉及的符号和符号意义.

表 1 符号定义

参数	描述
Q_{ij}	在第 i 轮训练中成功上传梯度给服务器 j 的客户端名单
Q_i	在第 i 轮训练中成功上传局部梯度的公共客户端名单
d	签署囚徒合约时服务器与客户端需支付的押金
o	签署合谋合约时服务器需支付的押金
q	成功计算全局梯度时服务器获得的奖励
q_z	服务器获得的总奖励
r	验证全局梯度需要支付的开销
g	服务器计算全局梯度需要的开销
h	正确计算全局梯度客户端得到的收益
v	服务器发起合谋寻找另一台服务器的开销
w_i^j	第 i 轮训练客户端 j 得到的局部梯度
w_i^g	第 i 轮训练的全局梯度
di	向量维度

表 1 中, 当 $q>g$ 时, 服务器才愿意参与训练, 当 $2q<r$ 时, 客户端才愿意使用双服务器进行验证, 当 $h-2q>0$ 时, 客户端才愿意参与训练. 此外, 上述参数还满足 $o>d+q-g$, $d>g+r$.

方案通过只使用成功上传梯度的客户端的交集 (公共客户端) 进行梯度聚合, 实现客户端离线. 具体由 4 个阶段组成, 分别为初始化, 签署合约, 模型训练, 奖金结算.

(1) 初始化: 确定全局模型、训练次数、客户端数量、客户端离线率、押金、奖金、时间参数, 系统挑选代理客户端.

(2) 签署合约: 只有当两台服务器都与客户端签署囚徒合约时, 才开始进行模型训练.

第 1 步: 代理客户端向服务器发起签署囚徒合约请求并提交押金.

第 2 步: 服务器同意签署囚徒合约, 并在指定时间内提交押金.

(3) 模型训练: 客户端下载全局梯度进行训练, 并向服务器上传局部梯度, 服务器计算全局梯度并返回客户端. 其中, 为处理客户端离线, 引入一轮通信 (第 3 步、第 4 步) 获取公共客户端名单.

第 1 步: 客户端对全局梯度进行对比验证.

第 2 步: 客户端训练模型并在规定时间内向服务器上传局部梯度.

第 3 步: 服务器返回客户端成功上传梯度的客户端名单 Q_{i1} 和 Q_{i2}.

第 4 步: 代理客户端收到 Q_{i1} 和 Q_{i2} 后向服务器上传它们的交集 Q_i.

第 5 步: 服务器接收到 Q_i 后, 只使用 Q_i 中的客户端梯度进行计算并在规定时间内返回全局梯度.

(4) 奖金结算:

在第 i 轮训练中, 如果客户端上传局部梯度, 则默认 i 轮之前两服务器都诚实. 若服务器不诚实, 客户端将拒绝向服务器传递局部梯度, 系统结算服务器本轮训练之前的奖励并没收不诚实服务器押金, 同时抛弃不诚实服务器; 若服务器在训练周期内都诚实, 奖励与押金将在训练结束后结算.

4.1 仲裁策略

由于玩家会选择使自身利益最大化的行为, 服务器为避开计算开销可能会不计算全局梯度, 同时在服务器正确计算全局梯度后, 客户端为避开奖励可能会接受全局梯度并否认结果的正确性, 为解决上述两类问题, 方案通过允许玩家提出仲裁, 验证其他玩家的行为. 由于网络情况的复杂性, 仲裁不对玩家实施惩罚, 但是, 提出仲裁请求需要支付开销 r, 不诚实玩家的押金将会作为仲裁发起者的补偿传递给仲裁发起者.

服务器与客户端的可选行为有: $A=\{verify, not\text{-}verify\}$, $verify$代表发起仲裁请求, $not\text{-}verify$代表不发起冲裁请求.

(1) 对于客户端, 需要考虑当收到相同结果与不同结果时是否对结果进行验证, 输出结果相同, 则服务器都诚实. 当输出结果不同时, 肯定存在服务器不诚实.

1) 当服务器都诚实时, 客户端的收益有:

$$u(verify)=h-2q-r,\ u(not\text{-}verify)=h-2q \tag{8}$$

因为 $u(verify)<u(not\text{-}verify)$, 即客户端收到相同结果时不验证可获得最大收益.

2) 当有一台服务器不诚实时, 客户端的收益有:

$$u(verify)=h-q-r+d,\ u(not\text{-}verify)=-2q \tag{9}$$

所以有 $u(verify)>u(not\text{-}verify)$, 即客户端收到不同结果时验证可获得最大收益.

3) 当有两台服务器都不诚实时, 客户端的收益有:

$$u(verify)=2d-r,\ u(not\text{-}verify)=-2q \tag{10}$$

所以有 $u(verify)>u(not\text{-}verify)$, 即客户端收到不同结果时验证可获得最大收益.

(2) 对于服务器, 为防止客户端抵赖服务器正确计算的结果, 服务器同样可以发起仲裁, 请求获得正确计算全局梯度的奖励.

1) 对于诚实服务器有:

$$u(verify)=q-g-r+d,\ u(not\text{-}verify)=-g-d \tag{11}$$

所以有 $u(verify)>u(not\text{-}verify)$, 即诚实服务器选择验证获得最大收益.

2) 对于不诚实服务器有:

$$u(verify)=-d-r,\ u(not\text{-}verify)=-d \tag{12}$$

所以有 $u(verify)<u(not\text{-}verify)$, 即不诚实服务器选择不验证获得最大的收益.

根据上述两种情况可知, 收到不相同的结果时, 客户端发起仲裁请求, 收到相同结果时, 客户端判断服务器诚实; 当客户端抵赖服务器计算结果的正确性时, 服务器可以发起仲裁请求, 获取自己正确计算全局梯度的证明, 从而获得奖励.

4.2 囚徒合约

联邦学习模型训练过程涉及多个客户端, 系统挑选代理客户端与服务器签署囚徒合约. 在两台服务器签署囚徒合约并提交押金后, 规定时间内没有返回正确的全局梯度, 则经过验证后, 不诚实服务器将会被惩罚失去押金 d, 诚实服务器的押金将会退回, 同时客户端将支付诚实服务器奖励 q. 只有当服务器的计算开销小于计算得到的奖励时, 服务器才愿意签署囚徒合约.

在算法 1 中, 只有当两服务器都与客户端成功签署囚徒合约时, 才能进行模型训练. 由于联邦学习中用户过少会影响模型训练的精确度, 因此, 需要根据模型训练的要求提供可允许的最大用户离线率, 只有当公共客户端数量不少于可允许的最小在线用户数量时, 服务器才会计算全局梯度, 否则客户端重新上传训练梯度, 同时, 向服务器上传局部梯度的客户端的数量不少于可允许的最小在线用户数量时, 客户端才不会被惩罚, 且存在客户端离线时代理客户端与服务器必须在规定的时间内传递客户端名单, 否则押金将会被没收.

算法 1. 囚徒合约.

初始化: *PC*.*t*1, *PC*.*t*2, *PC*.*t*3, *PC*.*t*4, *PC*.*t*5, 离线率 x, 客户端数量 N.

要求: *PC*.status=false.

```
1  代理客户端分别向两服务器发起囚徒合约并提交押金 d
2      if 服务器都同意签署合约
3          在 PC.t1 前提交押金 d
4          PC.status=true
5      else
6          退回客户端押金
7      if PC.status==true
```

```
8        if 在 PC.t2 前向服务器传递局部梯度的客户端的数量|Q_i1|<(1–x)N || |Q_i2|<(1–x)N
9          扣留客户端押金
10         退回其他玩家押金
11       elif 存在客户端离线时服务器没有在 PC.t3 前返回 Q_i1 和 Q_i2
12         扣留该服务器的押金
13         退回其他玩家押金
14       elif 存在客户端离线时代理客户端没有在 PC.t4 前返回 Q_i
15         扣留客户端押金
16         退回其他玩家押金
17       else
18         if  |Q_i| ⩾ (1 − x)N
19             if  两服务器在 PC.t5 前正确的计算全局梯度并返回
20                 客户端支付服务器奖励 q
21                 退回客户端与服务器的押金
22             elif 只有一台服务器在 PC.t5 前正确的全局梯度并返回
23                 诚实服务器发起验证请求并提交验证开销 r
24                 客户端支付诚实服务器计算奖励 q
25                 把不诚实服务器的押金给诚实服务器
26                 退回诚实服务器和客户端的押金
27             else
28                 客户端发起验证请求并提交验证开销 r
29                 把服务器的押金给客户端并退回客户端押金
30         else
31             退回到步骤 8, 并重新计时
32   end
```

在图 2 的博弈树中, 玩家 $P=\{P_1,P_2,P_3\}$, P_1 表示客户端, P_2 和 P_3 分别代表两服务器, 玩家的信息集为 $I=\{I_1,I_2,I_3\}$, $I_1=\{I_{11}\}$, $I_2=\{I_{21}\}$, $I_3=\{I_{31}\}$, $I_{11}=\{n_0\}$, $I_{21}=\{n_1\}$, $I_{31}=\{n_2,n_3\}$, 玩家可选行动集 $A=\{A_1,A_2,A_3\}$, $A_1=\{A_{11}\}$, $A_2=\{A_{21}\}$, $A_3=\{A_{31}\}$, $A_{11}=\{sign,not\text{-}sign\}$, $A_{21}=\{honest,dishonest\}$, $A_{31}=\{honest,dishonest\}$.

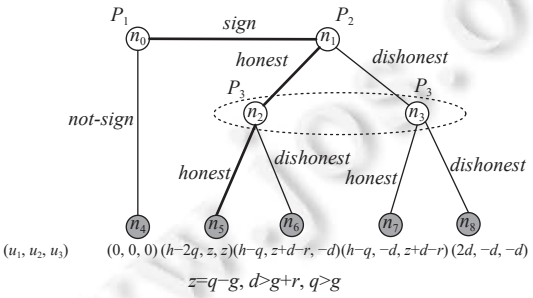

图 2 囚徒合约的博弈树

合约签署阶段, 客户端可选行动为: $\{sign,not\text{-}sign\}$, $sign$ 代表客户端发起囚徒合约并成功签署, $not\text{-}sign$ 则相反. 合约签署成功后, 服务器可选行动为: $\{honest,dishonest\}$, $honest$ 代表服务器诚实计算全局梯度, $dishonest$ 则相反.

图 2 展示了两服务器与客户端签订囚徒合约 (算法 1) 后服务器的行为以及该行为获得的收益, 通过对定理 1 的证明可知, 客户端发起囚徒合约并与两服务器成功签署合约后才获得收益, 合约签署成功后, 如果两服务器都不

诚实, 则两服务器的押金将会被传递给客户端, 如果存在一台服务器不诚实, 诚实服务器将发起验证从而获得计算奖励, 不诚实服务器的押金将会被分配给诚实服务器, 如果两服务器都诚实则都会获得计算奖励, 因此, 诚实成为服务器的最优选择. 总之, 客户端发起囚徒合约并与两服务器成功签署合约后才获得收益, 两服务器只有签订合约并诚实计算全局梯度才能获得最大收益, 博弈会沿着图 2 中的加粗路径进行. 所以客户端会选择发起囚徒合约, 两服务器会选择签订囚徒合约并诚实的执行聚合任务, 最终, 客户端通过两服务器返回结果的一致性判断是否存在不诚实的服务器, 进而实现全局梯度的完整性验证.

定理 1. 当 $d>g+r$, $q>g$ 时, 囚徒合约中客户端选择发起合约, 服务器选择诚实来获得最大收益, 唯一的序贯均衡为 $\{s,\beta\}$.

$$\begin{cases} s_1=([1(sign),0(not\text{-}sign)]) \\ s_2=([1(honest),0(dishonest)]) \\ s_3=([1(honest),0(dishonest)]) \\ \beta_1=([1(n_0)]) \\ \beta_2=([1(n_1)]) \\ \beta_3=([1(n_2),0(n_3)]) \end{cases}.$$

证明:

$$\begin{cases} s_1=([\gamma 1(sign),\gamma 2(not\text{-}sign)]) \\ s_2=([\alpha 1(honest),\alpha 2(dishonest)]) \\ s_3=([\rho 1(honest),\rho 2(dishonest)]) \\ \beta_1=([1(n_0)]) \\ \beta_2=([1(n_1)]) \\ \beta_3=([\alpha 1(n_2),\alpha 2(n_3)]) \end{cases},$$

有 $\rho 1+\rho 2=1,\alpha 1+\alpha 2=1,\gamma 1+\gamma 2=1$.

当 P_3 到达信息集 I_{31} 时 $u_3(s,\beta,I_{31})=\alpha 1\cdot u_3(s,n_2)+\alpha 2\cdot u_3(s,n_3)$.

因为 $\begin{cases} u_3(s,n_2)=\rho 1\cdot u_3(n_5)+\rho 2\cdot u_3(n_6) \\ u_3(s,n_3)=\rho 1\cdot u_3(n_7)+\rho 2\cdot u_3(n_8) \end{cases}$, 且 $\begin{cases} u_3(n_5)>u_3(n_6) \\ u_3(n_7)>u_3(n_8) \end{cases}$, 所以有 $\rho 1=1,\rho 2=0$ 时 P_3 在信息集 I_{31} 处得到最大收益.

当 P_2 到达信息集 I_{21} 处时 $u_2(s,\beta,I_{21})=\gamma 1\cdot u_2(s,n_1)=\gamma 1\cdot(\alpha 1\cdot u_2(s,n_1)+\alpha 2\cdot u_2(s,n_3))$.

因为 $\begin{cases} u_2(s,n_2)=\rho 1\cdot u_2(n_5)+\rho 2\cdot u_2(n_6) \\ u_2(s,n_3)=\rho 1\cdot u_3(n_7)+\rho 2\cdot u_3(n_8) \end{cases}$, 且 $\rho 1=1,\rho 2=0$, $u_2(n_5)>u_2(n_7)$, 所以 $\alpha 1=1,\alpha 2=0$ 时 P_2 在信息集 I_{21} 处取得最大收益.

当 P_1 到达信息集 I_{11} 处时 $u_1(s,\beta,I_{11})=u_1(s,n_0)=\gamma 2\cdot u_1(s,n_4)+\gamma 1\cdot u_1(s,n_1)$.

因为 $u_1(s,n_1)=\alpha 1\cdot u_1(s,n_2)+\alpha 2\cdot u_2(s,n_3)$ 且 $\rho 1=1,\rho 2=0,\ \alpha 1=1,\alpha 2=0,\ u_1(n_5)>u_1(n_4)$, 所以 $\gamma 1=1,\gamma 2=0$ 时 P_1 在信息集 I_{11} 处可取得最大收益.

本文设计完全混合得序列 $s^k=(s_1^k,s_2^k,s_3^k)$, 有:

$$\begin{cases} s_1=\left(\left[\dfrac{k-1}{k}(sign),\dfrac{1}{k}(not\text{-}sign)\right]\right) \\ s_2=\left(\left[\dfrac{k-1}{k}(honest),\dfrac{1}{k}(dishonest)\right]\right) \\ s_3=\left(\left[\dfrac{k-1}{k}(honest),\dfrac{1}{k}(dishonest)\right]\right) \end{cases}.$$

所以信念系统为:

$$\begin{cases} \beta_1 = ([1(n_0)]) \\ \beta_2 = ([1(n_1)]) \\ \beta_3 = ([1(n_2), 0(n_3)]) \end{cases},$$

有:

$$\begin{cases} \lim\limits_{k\to\infty} \dfrac{1}{k} = 0 \\ \lim\limits_{k\to\infty} \dfrac{k-1}{k} = 1 \end{cases} \to \begin{cases} \lim\limits_{k\to\infty} s^k \to s \\ \lim\limits_{k\to\infty} \beta^k \to \beta \end{cases}.$$

4.3 合谋合约

由于服务器计算全局梯度会存在开销 g, 为了得到更大的收益, 服务器会选择合谋返回相同的随机值. 由于服务器在众多服务器中寻找与他做相同计算任务的服务器需要一定的开销 v, 当 $q-g>q-v$ 时, 服务器选择诚实会得到更大的收益; 当 $q-g<q-v$ 时, 服务器可以通过选择合谋避开计算开销获得更大的收益. 当在算法 2 中, 合谋接受者 (跟随者) 收到合谋发起者 (领导者) 的合谋请求后, 有两个可选行动: $\{sign, not\text{-}sign\}$. 选择行动 *not-sign*, 跟随者面对领导者提出的合谋不做处理, 领导者会消耗向跟随者发起合谋请求寻找跟随者的开销. 跟随者选择 *sign* 后, 两服务器签署合约, 提交押金 o, 领导者和跟随者有 3 个可选行动: $\{w^g, con, \text{others}\}$. 由于 others 是严格劣战略, 选择该策略不仅会受到囚徒合约的惩罚, 也会受到合谋合约的惩罚, 如表 2, 对于理性玩家, 他们不会选择该策略, 本文不对选择 others 的情形进行赘述, 即领导者和跟随者仅两个可选行动: $\{w^g, con\}$. w^g 代表服务器诚实的计算全局梯度, *con* 代表服务器合谋.

表 2 行动收益对照表

P_x 行动	P_y 选 others 收益	P_y 选 w^g 收益	P_y 选 *con* 收益	P_y 最低收益行动
con	$-d-o$	$q-g-o$	q	others
w^g	$-d-o$	$q-g-o$	$-d$	others
others	$-d-o$	$q-g-o$	$-d$	others

在算法 2 中, 当合谋合约签署成功, 若存在玩家不执行合谋合约内容, 即不返回 *con*, 则该玩家的押金 o 将会被没收. 若玩家执行合谋合约内容返回 *con*, 该玩家的押金 o 将会被退回.

算法 2. 合谋合约.

初始化: *CC.t*1, *CC.t*2.
要求: *PC*.status=true, *CC*.status=false.
1 领导者发起合谋请求并提交押金 o
2 if 跟随者同意签署
3 在 *CC.t*1 之前提交押金 o
4 两服务器协商 *con* 伪造成全局梯度
5 *CC*.status=true
6 else
7 退回领导者押金 o
8 if *CC*.status==true
9 if 两服务器在 *CC.t*2 前返回 *con*
10 客户端支付两服务器奖励 q
11 退回服务器押金 o 和 d

```
12          else
13              没收没有返回 con 的服务器押金 o
14  end
```

在图 3 的博弈树中, 玩家 $P=\{P_1,P_2\}$, P_1 代表合谋跟随者, P_2 代表合谋发起者, 信息集 $I=\{I_1,I_2\}$, $I_1=\{I_{11},I_{12}\}$, $I_2=\{I_{21},I_{22}\}$, $I_{11}=\{n_1\}$, $I_{12}=\{n_3,n_4\}$, $I_{21}=\{n_0\}$, $I_{22}=\{n_2\}$. 可选行动集 $A=\{A_1,A_2\}$, $A_1=\{A_{11},A_{12}\}$, $A_2=\{A_{21},A_{22}\}$, $A_{11}=\{sign,not\text{-}sign\}$, $A_{12}=\{con,w^g\}$, $A_{21}=\{sign,not\text{-}sign\}$, $A_{22}=\{con,w^g\}$.

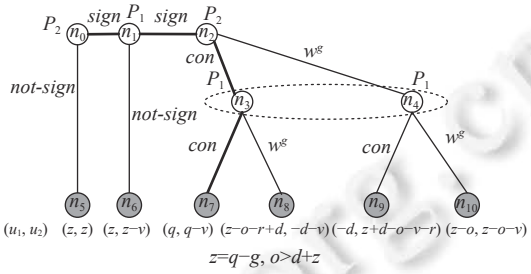

图 3 合谋合约的博弈树

定理 2. $q-g<q-v$ 时, 领导者肯定发起合谋合约请求, 跟随者一定签署合谋合约, 当 $o>d+z$ 时两服务器不敢背叛合约, 并选择 *con* 获得最大收益. 合谋合约存在唯一的序贯均衡 $\{s,\beta\}$.

$$\begin{cases} s_1=([1(sign),0(not\text{-}sign)],[1(con),0(w^g)]) \\ s_2=([1(sign),0(not\text{-}sign)],[1(con),0(w^g)]) \\ \beta_1=([1(n_1)],[1(n_3),0(n_4)]) \\ \beta_2=([1(n_0)],[1(n_2)]) \end{cases}.$$

证明:

$$\begin{cases} s_1=([\rho 1(sign),\rho 2(not\text{-}sign)],[\rho 3(con),\rho 4(w^g)]) \\ s_2=([\gamma 1(sign),\gamma 2(not\text{-}sign)],[\gamma 3(con),\gamma 4(w^g)]) \\ \beta_1=([1(n_1)],[\gamma 3(n_3),\gamma 4(n_4)]) \\ \beta_2=([1(n_0)],[1(n_2)]) \end{cases},$$

有 $\rho 1+\rho 2=1$, $\rho 3+\rho 4=1$, $\gamma 1+\gamma 2=1$, $\gamma 3+\gamma 4=1$.

当 P_1 到达信息集 I_{12} 时, $u_1(s,\beta,I_{12})=\gamma 3\cdot u_1(s,n_3)+\gamma 4\cdot u_1(s,n_4)$.

因为 $\begin{cases} u_1(s,n_3)=\rho 3\cdot u_1(n_7)+\rho 4\cdot u_1(n_8) \\ u_1(s,n_4)=\rho 3\cdot u_1(n_9)+\rho 4\cdot u_1(n_{10}) \end{cases}$, 且 $\begin{cases} u_1(n_7)>u_1(n_8) \\ u_1(n_9)>u_1(n_{10}) \end{cases}$, 所以 $\rho 3=1,\rho 4=0$ 时 P_1 在信息集 I_{12} 处取得最大收益.

当 P_2 到达信息集 I_{22} 时 $u_2(s,\beta,I_{22})=\rho 1\cdot u_2(s,n_2)$.

因为 $\rho 3=1,\rho 4=0$, 有 $u_2(s,n_2)=\gamma 3\cdot u_2(s,n_7)+\gamma 4\cdot u_2(s,n_9)$, 且 $u_2(n_7)>u_2(n_9)$, 所以 $\gamma 3=1$, $\gamma 4=0$ 时 P_2 在信息集 I_{22} 处取得最大收益.

当 P_1 到达信息集 I_{11} 时, $u_1(s,\beta,I_{11})=\gamma 1\cdot u_1(s,n_1)$.

因为 $\rho 3=1,\rho 4=0,\gamma 3=1,\lambda 4=0$, 有 $u_1(s,n_1)=\rho 1\cdot u_1(n_7)+\rho 2\cdot u_1(n_6)$, 所以 $\rho 1=1,\rho 2=0$ 时 P_1 在信息集 I_{11} 处取得最大收益.

当 P_2 到达信息集 I_{21} 时, $u_1(s,\beta,I_{21})=\gamma 1\cdot u_2(s,n_1)+\gamma 2\cdot u_2(s,n_5)$.

因为 $\rho 3=1,\rho 4=0,\gamma 3=1,\lambda 4=0$, $\rho 1=1,\rho 2=0$ 所以 $u_2(s,\beta,I_{21})=\gamma 1\cdot u_2(n_7)+\gamma 2\cdot u_2(n_5)$, 因为 $u_2(n_7)>u_2(n_5)$,

所以 $\gamma 1=1,\gamma 2=0$ 时 P_2 在信息集 I_{21} 处取得最大收益.

本文设计完全混合序列 $s^k=(s_1^k,s_2^k)$, 有:

$$\begin{cases} s_1=\left(\left[\frac{k-1}{k}(sign),\frac{1}{k}(not\text{-}sign)\right],\left[\frac{k-1}{k}(con),\frac{1}{k}(w^g)\right]\right) \\ s_2=\left(\left[\frac{k-1}{k}(sign),\frac{1}{k}(not\text{-}sign)\right],\left[\frac{k-1}{k}(con),\frac{1}{k}(w^g)\right]\right) \end{cases}.$$

所以信念系统为:

$$\begin{cases} \beta_1=\left([1(n_1)],\left[\frac{k-1}{k}(n_3),\frac{1}{k}(n_4)\right]\right) \\ \beta_2=([1(n_0)],[1(n_2)]) \end{cases},$$

有:

$$\begin{cases} \lim\limits_{k\to\infty}\frac{1}{k}=0 \\ \lim\limits_{k\to\infty}\frac{k-1}{k}=1 \end{cases}\to\begin{cases} \lim\limits_{k\to\infty}s^k\to s \\ \lim\limits_{k\to\infty}\beta^k\to\beta \end{cases}.$$

图 3 展示了两服务器选择合谋 (算法 2) 避免计算开销与诚实执行聚合任务会获得的收益, 通过对定理 2 的证明可知, 合谋发起者发起合谋后, 跟随者会为了更高的收益选择签订合谋合约, 此时服务器受到囚徒合约与合谋合约的约束, 在之后的博弈中, 如果两服务器都遵循合谋合约, 客户端将收到两份相同的计算结果, 从而判断两服务器诚实, 服务器在不做聚合任务的同时也获得了计算奖励, 避免了计算开销. 如果存在一台服务器违背合谋合约选择诚实的执行聚合任务, 则其将会受到合谋合约的惩罚而失去押金 o, 同时获得囚徒合约的奖励. 由于 $o>d+z$, 服务器不会违背合谋合约, 博弈将沿着图 3 中的加粗路径进行, 总之, 在囚徒合约与合谋合约的约束下合谋将是服务器的最优选择.

4.4 背叛合约

背叛合约能够有效地防止服务器之间的合谋, 它由服务器与客户端签署, 发起背叛合约的服务器称为背叛者. 一旦签署背叛合约, 服务器在背叛阶段将获得两次向客户端传递全局梯度的机会. 从而使背叛者可以使用第 1 次机会向客户端传递 con, 避开合谋合约的惩罚, 使用第 2 次机会传递 w^g 避开囚徒合约的惩罚并获得奖励. 一旦在第 1 次结果中传递了 w^g, 客户端将不接受第 2 次传递的结果, 博弈结束, 背叛者将会受到合谋合约的惩罚. 所以背叛者第 1 次机会传输 con, 第 2 次机会传输 w^g, 将是背叛者的最优选择[22]. 背叛合约的签署说明了系统中出现了不诚实玩家, 所以在背叛阶段, 诚实计算全局梯度的玩家都需要发起仲裁请求对全局梯度进行验证, 否则客户端不接受未经验证的结果.

算法 3. 背叛合约.

初始化: *BC*.*t*1, *BC*.*t*2.
要求: *CC*.status=true, *BC*.status=false.
1 if 服务器在 *BC*.*t*1 前向客户端发起背叛合约
2 if 客户端在 *BC*.*t*2 前同意背叛
3 *BC*.status= true
4 else
5 执行合谋合约
6 end

由于合谋合约由两服务器签署, 客户端只能通过背叛者提供的地址知道合谋合约的存在, 无法判断合谋合约的发起者, 所以为提供公平的网络环境, 允许领导者和跟随着都可以与客户端签署背叛合约. 在算法 3 中, 领导者

和跟随者在背叛阶段的可选行动都为: $\{betrayal, not\text{-}betrayal\}$. 在算法 4 中, 当背叛合约签署, 背叛者将获得两次传输聚合结果的机会, 可选行动为: $\{con, w^g\}$.

算法 4. 背叛策略选择.

初始化: *BC.t*.

要求: *CC*.status=true.

```
if  CC.status==true
    if 领导者和跟随者都背叛并在 BC.t 前返回 con 和 w^g
        客户端支付两服务器奖励 q
        退回服务器押金 d 和 o
        服务器支付验证开销 r
    elif 只有一台服务器背叛并在 BC.t 前返回 con 和 w^g
        客户端支付诚实计算全局梯度的服务器奖励 q
        诚实服务器支付验证开销 r
        if 不背叛者返回 con
            退回两服务器押金 o 并将不背叛者押金 d 给背叛者
            退回背叛者的押金 d
        else 不背叛者返回 w^g
            退回两服务器押金 d 并没收不背叛者押金 o
            退回背叛者的押金 o
    else 都不背叛
        执行合谋合约
end
```

在图 4 的博弈树中, 参与游戏的玩家有 $P=\{P_1,P_2\}$, P_1 代表合谋跟随着, P_2 代表合谋发起者, 信息集为 $I=\{I_1,I_2\}$, $I_1=\{I_{11},I_{12},I_{13},I_{14},I_{15},I_{16}\}$, $I_2=\{I_{21},I_{22},I_{23},I_{24}\}$, $I_{11}=\{n_1\}$, $I_{12}=\{n_3,n_4\}$, $I_{13}=\{n_7,n_8\}$, $I_{14}=\{n_9,n_{10}\}$, $I_{15}=\{n_{13},n_{14}\}$, $I_{16}=\{n_{15},n_{16}\}$, $I_{21}=\{n_0\}$, $I_{22}=\{n_2\}$, $I_{23}=\{n_5,n_6\}$, $I_{24}=\{n_{11},n_{12}\}$. 可选行动集有 $A=\{A_1,A_2\}$, $A_1=\{A_{11},A_{12},A_{13},A_{14},A_{15},A_{16}\}$, $A_2=\{A_{21},A_{22},A_{23},A_{24}\}$, $A_{11}=\{sign,not\text{-}sign\}$, $A_{12}=\{betrayal,not\text{-}betrayal\}$, $A_{13}=A_{14}=A_{15}=A_{16}=\{con,w^g\}$, $A_{21}=\{sign,not\text{-}sign\}$, $A_{22}=\{betrayal,not\text{-}betrayal\}$, $A_{23}=A_{24}=\{con,w^g\}$

定理 3. 当 $o>d+z$ 时, 服务器一定不会发起合谋从而得到更高收益. 存在唯一序贯均衡 $\{s,\beta\}$.

$$\begin{cases} s_1=([0(sign),1(not\text{-}sign)],[1(betrayal),0(not\text{-}betrayal)],[1(con),0(w^g)][1(con),0(w^g)],[1(con),0(w^g)],[1(con),0(w^g)] \\ s_2=([0(sign),1(not\text{-}sign)],[1(betrayal),0(not\text{-}betrayal)][1(con),0(w^g)],[1(con),0(w^g)]) \\ \beta_1=([1(n_1)],[1(n_3),0(n_4)],[1(n_7),0(n_8)],[1(n_9),0(n_{10})],[1(n_{13}),0(n_{14})],[1(n_{15}),0(n_{16})]) \\ \beta_2=([1(n_0)],[1(n_2)],[1(n_6),0(n_5)],[1(n_{11}),0(n_{12})]) \end{cases}$$

证明:

$$\begin{cases} s_1=\left(\begin{array}{l}[\rho1(sign),\rho2(not\text{-}sign)],[\rho3(betrayal),\rho4(not\text{-}betrayal)],[\rho5(con),\rho6(w^g)][\rho7(con),\rho8(w^g)], \\ [\rho9(con),\rho10(w^g)],[\rho11(con),\rho12(w^g)]\end{array}\right) \\ s_2=([\gamma1(sign),\gamma2(not\text{-}sign)],[\gamma3(betrayal),\gamma4(not\text{-}betrayal)][\gamma5(con),\gamma6(w^g)],[\gamma7(con),\gamma8(w^g)]) \\ \beta_1=([1(n_1)],[\gamma3(n_3),\gamma4(n_4)],[\gamma5(n_7),\gamma6(n_8)],[\gamma5(n_9),\gamma6(n_{10})],[\gamma7(n_{13}),\gamma8(n_{14})],[\gamma7(n_{15}),\gamma8(n_{16})]) \\ \beta_2=([1(n_0)],[1(n_2)],[\rho3(n_6),\rho4(n_5)],[\rho3(n_{11}),\rho4(n_{12})]) \end{cases}.$$

$\rho 1+\rho 2=1, \rho 3+\rho 4=1, \rho 5+\rho 6=1, \rho 7+\rho 8=1, \rho 9+\rho 10=1, \rho 11+\rho 12=1, \gamma 1+\gamma 2=1, \gamma 3+\gamma 4=1,$
$\gamma 5+\gamma 6=1, \gamma 7+\gamma 8=1.$

当 P_1 到达信息集 I_{13} 时 $u_1(s,\beta,I_{13})=\gamma 5\cdot u_1(s,n_7)+\gamma 6\cdot u_1(s,n_8)$.

因为 $\begin{cases} u_1(s,n_7)=\rho 5\cdot u_1(n_{17})+\rho 6\cdot u_1(n_{18}) \\ u_1(s,n_8)=\rho 5\cdot u_1(n_{19})+\rho 6\cdot u_1(n_{20}) \end{cases}$，且 $\begin{cases} u_1(n_{17})>u_1(n_{18}) \\ u_1(n_{19})>u_1(n_{20}) \end{cases}$，所以 $\rho 5=1, \rho 6=0$ 时，P_1 在信息集 I_{13} 处取得最大收益.

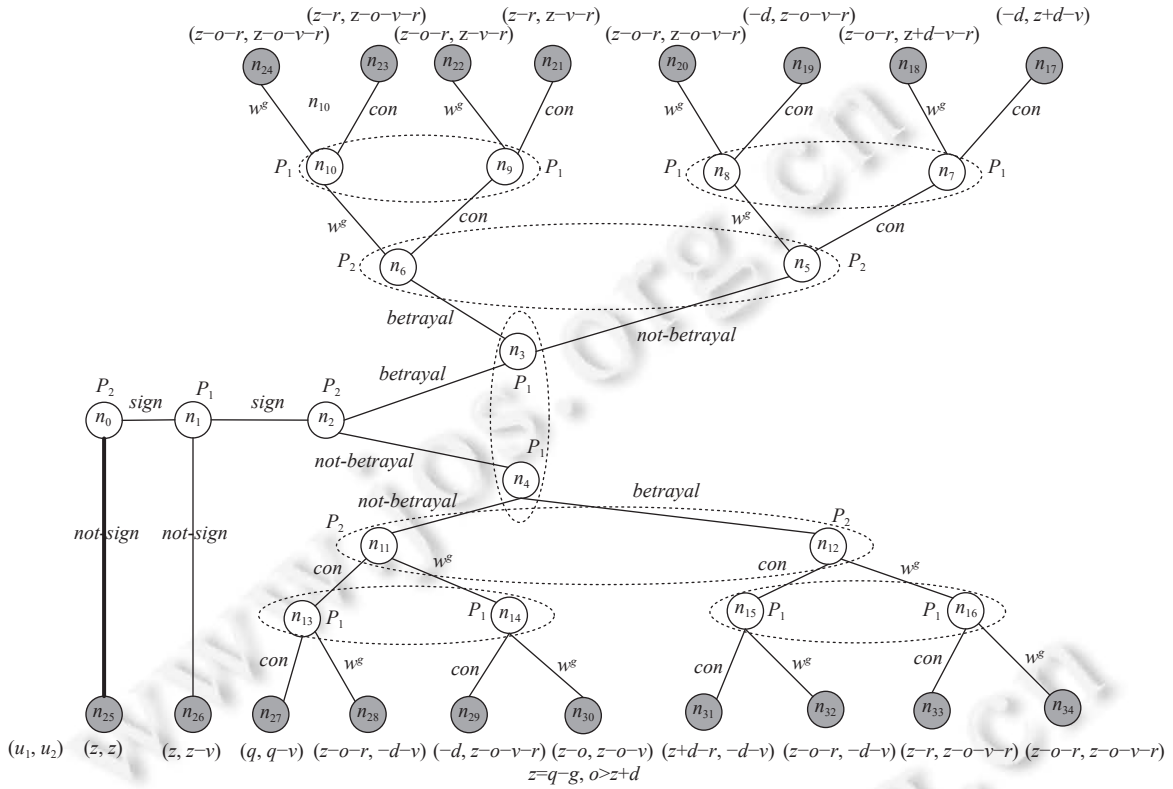

图 4 背叛合约的博弈树

当 P_1 到达信息集 I_{14} 时 $u_1(s,\beta,I_{14})=\gamma 5\cdot u_1(s,n_9)+\gamma 6\cdot u_1(s,n_{10})$.

因为 $\begin{cases} u_1(s,n_9)=\rho 7\cdot u_1(n_{21})+\rho 8\cdot u_1(n_{22}) \\ u_1(s,n_{10})=\rho 7\cdot u_1(n_{23})+\rho 8\cdot u_1(n_{24}) \end{cases}$，且 $\begin{cases} u_1(n_{21})>u_1(n_{22}) \\ u_1(n_{23})>u_1(n_{24}) \end{cases}$，所以 $\rho 7=1, \rho 8=0$ 时，P_1 在信息集 I_{14} 处取得最大收益.

当 P_1 到达信息集 I_{15} 时 $u_1(s,\beta,I_{15})=\gamma 7\cdot u_1(s,n_{13})+\gamma 8\cdot u_1(s,n_{14})$.

因为 $\begin{cases} u_1(s,n_{13})=\rho 9\cdot u_1(n_{27})+\rho 10\cdot u_1(n_{28}) \\ u_1(s,n_{14})=\rho 9\cdot u_1(n_{29})+\rho 10\cdot u_1(n_{30}) \end{cases}$，且 $\begin{cases} u_1(n_{27})>u_1(n_{28}) \\ u_1(n_{29})>u_1(n_{30}) \end{cases}$，所以 $\rho 9=1, \rho 10=0$ 时，P_1 在信息集 I_{15} 处取得最大收益.

当 P_1 到达信息集 I_{16} 时 $u_1(s,\beta,I_{16})=\gamma 7\cdot u_1(s,n_{15})+\gamma 8\cdot u_1(s,n_{16})$.

因为 $\begin{cases} u_1(s,n_{15})=\rho 11\cdot u_1(n_{31})+\rho 12\cdot u_1(n_{32}) \\ u_1(s,n_{16})=\rho 11\cdot u_1(n_{33})+\rho 12\cdot u_1(n_{34}) \end{cases}$，且 $\begin{cases} u_1(n_{31})>u_1(n_{32}) \\ u_1(n_{33})>u_1(n_{34}) \end{cases}$，所以 $\rho 11=1, \rho 12=0$ 时，P_1 在信息集 I_{16} 处取得最大收益.

当 P_2 到达信息集 I_{23} 时 $u_2(s,\beta,I_{23})=\rho 4\cdot u_2(s,n_5)+\rho 3\cdot u_2(s,n_6)$.

因为 $\begin{cases} u_2(s,n_5)=\gamma 5\cdot u_2(n_7)+\gamma 6\cdot u_2(n_8) \\ u_2(s,n_6)=\gamma 5\cdot u_2(n_9)+\gamma 6\cdot u_2(n_{10}) \end{cases}$，且 $\rho 7=1, \rho 8=0$，$\rho 5=1, \rho 6=0$，所以有 $u_2(s,\beta,I_{23})=\rho 4\cdot(\gamma 5\cdot u_2(n_{17})+\gamma 6\cdot u_2(n_{19}))+\rho 3\cdot(\gamma 5\cdot u_2(n_{21})+\gamma 6\cdot u_2(n_{23}))$.

因为 $\begin{cases}u_2(n_{17})>u_2(n_{19})\\ u_2(n_{21})>u_2(n_{23})\end{cases}$, 所以 $\gamma 5=1,\gamma 6=0$ 时, P_2 在信息集 I_{23} 处取得最大收益.

当 P_2 到达信息集 I_{24} 时 $u_2(s,\beta,I_{24})=\rho 4\cdot u_2(s,n_{11})+\rho 3\cdot u_2(s,n_{12})$.

因为 $\begin{cases}u_2(s,n_{11})=\gamma 7\cdot u_2(n_{13})+\gamma 8\cdot u_2(n_{14})\\ u_2(s,n_{12})=\gamma 7\cdot u_2(n_{15})+\gamma 8\cdot u_2(n_{16})\end{cases}$, 且 $\rho 9=1,\rho 10=0,\rho 11=1,\rho 12=0$, 所以有 $u_2(s,\beta,I_{24})=\rho 4\cdot(\gamma 7\cdot u_2(n_{27})+\gamma 8\cdot u_2(n_{29}))+\rho 3\cdot(\gamma 7\cdot u_2(n_{31})+\gamma 8\cdot u_2(n_{33}))$.

因为 $\begin{cases}u_2(n_{27})>u_2(n_{29})\\ u_2(n_{31})>u_2(n_{33})\end{cases}$, 所以 $\gamma 7=1,\gamma 8=0$ 时, P_2 在信息集 I_{24} 处取得最大收益.

当 P_1 到达信息集 I_{12} 时 $u_1(s,\beta,I_{12})=\gamma 3\cdot u_1(s,n_3)+\gamma 4\cdot u_1(s,n_4)$.

因为 $\begin{cases}u_1(s,n_3)=\rho 4\cdot u_1(n_5)+\rho 3\cdot u_1(n_6)\\ u_1(s,n_4)=\rho 4\cdot u_1(n_{11})+\rho 3\cdot u_1(n_{13})\end{cases}$, 且 $\gamma 5=\lambda 7=1,\gamma 6=\lambda 8=0,\rho 5=\rho 7=\rho 9=\rho 11=1,\rho 6=\rho 8=\rho 10=\rho 12=0$, 所以有 $\begin{cases}u_1(s,n_3)=\rho 4\cdot u_1(n_{17})+\rho 3\cdot u_1(n_{21})\\ u_1(s,n_4)=\rho 4\cdot u_1(n_{27})+\rho 3\cdot u_1(n_{31})\end{cases}$, 因为 $\begin{cases}u_1(n_{21})>u_1(n_{17})\\ u_1(n_{31})>u_1(n_{27})\end{cases}$, 所以 $\rho 3=1,\rho 4=0$ 时, P_1 在信息集 I_{12} 处取得最大收益.

当 P_2 到达信息集 I_{22} 时 $u_2(s,\beta,I_{22})=\rho 1\cdot u_2(s,n_2)$, 有 $u_2(s,n_2)=\gamma 3\cdot u_2(n_3)+\gamma 4\cdot u_2(n_4)$.

因为 $\gamma 5=\lambda 7=1,\gamma 6=\lambda 8=0,\rho 3=\rho 5=\rho 7=\rho 9=\rho 11=1,\rho 4=\rho 6=\rho 8=\rho 10=\rho 12=0$, 所以有 $u_2(s,n_2)=\gamma 3\cdot u_2(n_{21})+\gamma 4\cdot u_2(n_{31})$, 因为 $u_2(n_{21})>u_2(n_{31})$, 所以 $\gamma 3=1,\gamma 4=0$ 时, P_2 在信息集 I_{22} 处取得最大收益.

当 P_1 到达信息集 I_{11} 时 $u_1(s,\beta,I_{11})=\rho 1\cdot u_1(s,n_2)+\rho 2\cdot u_1(s,n_{26})$.

因为 $\lambda 3=\gamma 5=\lambda 7=1,\ \gamma 4=\gamma 6=\lambda 8=0,\rho 3=\rho 5=\rho 7=\rho 9=\rho 11=1,\ \rho 4=\rho 6=\rho 8=\rho 10=\rho 12=0$, 所以有 $u_1(s,n_2)=u_1(s,n_{21})$, 因为 $u_1(n_{26})>u_1(n_{21})$, 所以 $\rho 2=1,\rho 1=0$ 时, P_1 在信息集 I_{11} 处取得最大收益.

当 P_2 到达信息集 I_{21} 时 $u_2(s,\beta,I_{21})=\gamma 1\cdot u_2(s,n_1)+\gamma 2\cdot u_2(s,n_{25})$.

因为 $\lambda 3=\gamma 5=\lambda 7=1,\gamma 4=\gamma 6=\lambda 8=0,\rho 2=\rho 3=\rho 5=\rho 7=\rho 9=\rho 11=1,\rho 1=\rho 4=\rho 6=\rho 8=\rho 10=\rho 12=0$, 所以有 $u_2(s,n_1)=u_2(s,n_{26})$ 因为 $u_2(s,n_{25})>u_2(s,n_{26})$, 所以 $\gamma 2=1,\gamma 1=0$ 时, P_2 在信息集 I_{21} 处取得最大收益.

本文设计完全混合的序列 $s^k=(s_1^k,s_2^k)$, 有:

$$\begin{cases}s_1=\left(\left[\frac{1}{k}(sign),\frac{k-1}{k}(not\text{-}sign)\right],\left[\frac{k-1}{k}(betrayal),\frac{1}{k}(not\text{-}betrayal)\right],\left[\frac{k-1}{k}(con),\frac{1}{k}(w^g)\right],\left[\frac{k-1}{k}(con),\frac{1}{k}(w^g)\right],\right.\\ \left.\left[\frac{k-1}{k}(con),\frac{1}{k}(w^g)\right],\left[\frac{k-1}{k}(con),\frac{1}{k}(w^g)\right]\right)\\ s_2=\left(\left[\frac{1}{k}(sign),\frac{k-1}{k}(not\text{-}sign)\right],\left[\frac{k-1}{k}(betrayal),\frac{1}{k}(not\text{-}betrayal)\right],\left[\frac{k-1}{k}(con),\frac{1}{k}(w^g)\right],\left[\frac{k-1}{k}(con),\frac{1}{k}(w^g)\right]\right)\end{cases}.$$

所以信念系统为:

$$\begin{cases}\beta_1=\left([1(n_1)],\left[\frac{k-1}{k}(n_3),\frac{1}{k}(n_4)\right],\left[\frac{k-1}{k}(n_7),\frac{1}{k}(n_8)\right],\left[\frac{k-1}{k}(n_9),\frac{1}{k}(n_{10})\right],\left[\frac{k-1}{k}(n_{13}),\frac{1}{k}(n_{14})\right],\left[\frac{k-1}{k}(n_{15}),\frac{1}{k}(n_{16})\right]\right)\\ \beta_2=\left([1(n_0)],[1(n_2)],\left[\frac{k-1}{k}(n_6),\frac{1}{k}(n_5)\right],\left[\frac{k-1}{k}(n_{11}),\frac{1}{k}(n_{12})\right]\right)\end{cases},$$

有:

$$\begin{cases}\lim\limits_{k\to\infty}\frac{1}{k}=0\\ \lim\limits_{k\to\infty}\frac{k-1}{k}=1\end{cases}\to\begin{cases}\lim\limits_{k\to\infty}s^k\to s\\ \lim\limits_{k\to\infty}\beta^k\to\beta\end{cases}.$$

图 4 展示了当客户端提供服务器背叛合谋合约 (算法 3) 的机会时, 合谋发起者与合谋跟随者的选择以及该选择会获得的收益. 通过对定理 3 的证明可知, 在签订合谋合约后服务器如果都不发起背叛, 博弈将沿着合谋合约博弈路径进行, 如果存在一台服务器 (背叛者) 背叛合谋合约并利用第 1 次机会向客户端传递合谋协商的全局梯度, 利用第 2 次机会传递正确的全局梯度, 则该服务器将避开合谋合约的惩罚并获得计算奖励, 此时, ① 如果另一台服务器不背叛合谋合约, 则背叛者将获得额外的奖励 d (来自不背叛者签署囚徒合约的押金), ② 如果另一台服务

器也选择背叛, 则合谋发起者的收益为 $q-g-v-r$, 跟随者的收益为 $q-g-r$, 最终收益反而低于只签署囚徒合约的收益, 因此, 合谋合约一旦签订, 服务器一定会选择背叛, 两服务器都背叛时得到的收益低于只签署囚徒合约的收益, 不合谋将成为服务器的最优选择, 博弈将沿着图 4 中的粗线路径执行. 总之, 背叛合约的存在会使服务器不合谋, 并诚实的执行聚合任务, 从而证明基于复制的方法使用双服务器执行聚合任务不会受到合谋的威胁最终实现全局梯度的完整性验证.

4.5 模型训练

模型训练中一旦发现服务器不诚实, 该服务器会被系统抛弃, 系统重新随机为模型分配服务器. 在完全不完美信息动态博弈中, 当基础博弈有唯一的均衡路径时, 他的重复博弈为基础博弈的重复, 收益为基础博弈收益的简单相加[32]. 囚徒合约只需要在模型训练前签署, 只有在出现不诚实行为时才会重新签署, 否则在模型训练的整个过程中都生效. 在没有出现合谋和背叛时, 客户端与服务器签署囚徒合约提交的押金在上一轮训练结束之后不用退回, 作为下一轮训练的押金, 且奖励在训练结束后结算.

算法 5. 训练流程.

初始化: 模型参数 w, 训练次数 n, 学习率 λ, i=0, 离线率 x, 客户端数量 N, 囚徒合约押金 d, 时间 t, $t1$, $t2$, $t3$, $t4$, $T=t+system.t$, $T1=T(T4)+t1$, $T2=T1+t2$, $T3=T2+t3$, $T4=T3+t4$.

1 客户端向服两务器发起囚徒合约并提交押金 d
2 $system.t=now()$
3 if 两服务器都同意签署合约
4 在 T 前提交押金 d
5 PC.status=true
6 else
7 退回客户端押金
8 if PC.status=true
9 下载全局梯度
10 while $i<n$
11 客户端训练模型, 在 $T1$ 前向服务器上传局部梯度 w_i^j
12 服务器在 $T2$ 前向代理客户端发送 (Q_{i1}, Q_{i2})
13 代理客户端通过 (Q_{i1}, Q_{i2}) 提取 Q_i, 在 $T3$ 前向两服务器发送 Q_i
14 if $|Q_i|\geqslant(1-x)N$
15 服务器使用 Q_i 中的客户端梯度计算全局梯度并在 $T4$ 前返回全局梯度
16 客户端对比两份全局梯度 w_i^g
17 if 全局梯度相同
18 客户端更新模型参数
19 $i++$
20 else
21 执行合约惩罚和奖励
22 重新签署囚徒合约
23 else
24 跳转到第 11 步
25 退回押金并结算奖励 $q\cdot n$
26 end

在算法 5 中, 呈现了模型训练的完整过程. 模型训练需要在合约签署之后才能开始, 每轮训练客户端需要对两份全局梯度进行对比. 若结果不一致则结算奖金, 并对押金进行分配, 之后重新选择服务器签署合约并训练模型. 训练分为 4 个阶段.

1) 初始化: 确定 ① 训练轮数 n, ② 客户端数量 N, ③ 客户端离线率 x, ④ 押金 d, ⑤ 奖金 q, ⑥ 时间 t, $t1$, $t2$, $t3$, $t4$, 其中, t 为客户端发起囚徒合约到服务器签署合约的最大时间间隔, $t1$ 为客户端训练模型到服务器收到局部梯度的最大时间间隔, $t2$ 为服务器收到局部梯度到客户端收到客户端名单的最大时间间隔, $t3$ 为客户端收到客户端名单到服务器收到公共客户端名单的最大时间间隔, $t4$ 为服务器收到公共客户端名单到客户端收到全局梯度的最大时间间隔, $now()$ 为系统的当前时间.

2) 签署合约: 代理客户端在 $system.t$ 时发起囚徒合约, 服务器在 T 前给予回应.

3) 模型训练: 客户端和服务器必须在指定 $T1$, $T2$, $T3$, $T4$ 时间前进行对应的操作, 否则押金将会被没收. 只有在第 1 轮训练时 $T1=T+t1$. 服务器对局部梯度进行聚合得到全局梯度, 客户端在下一轮训练中对两份全局梯度进行对比验证, 结果相同则更新模型并进行该轮训练.

4) 奖金结算: 训练 n 轮后, 结算总奖励 $q_z = q \cdot n$, 并退回各玩家押金.

5 实验分析

5.1 性能分析

文献 [19] 是最新关于联邦学习全局梯度完整性验证的研究, 因此, 本文与该方案进行了对比. 文中提出的基于复制的验证方案与基于密码算法的验证方案最大的不同在于该方案不需要客户端计算用于验证的验证信息, 从而把客户端的计算开销从文献 [19] 中的 O(di) 降为 0. 其次, 方案在有客户端离线时, 一次迭代的通信轮数仅需两轮 (一轮用于传递梯度, 另一轮用于客户端离线时获取公共客户端名单), 在没有客户端离线时仅需要一轮通信, 而文献 [19] 则需要 3 轮通信 (第 1 轮用于传递验证信息, 第 2 轮用于传递梯度, 第 3 轮用于出现客户端离线时传递哈希值与随机数的密钥份额). 最后, 从第 5.3 节中的实验可知, 所提方案用于验证的时间低于训练总时间的 1/2, 而文献 [19] 提出的方案用于验证的时间高于训练总时间的 1/2, 说明本文提出的验证方案训练开销更低. 当存在客户端离线时, 所提方案仅需要服务器向代理客户端传递成功向其传递局部梯度的客户端名单, 训练开销与离线率成反比, 而文献 [19] 需要服务器向所有在线客户端传递离线客户端的哈希值与随机数, 训练开销随离线率的增加而增加, 此时文献 [19] 的通信复杂度高于本文方案, 离线率越高时更明显.

通过与文献 [19] 中的验证算法分别在客户端计算开销, 一次迭代的通信轮数, 存在客户端离线时一次迭代的通信轮数, 验证时间占比这 4 个方面进行对比, 如表 3 所示, 本文提出的方案优势有如下几点: 客户端的计算复杂度由原来的 O(di) 降到 0, 通信轮数减少一轮, 用于验证的时间占比不到训练总时间的 1/2.

表 3 性能对比

方案	客户端计算开销	离线时通信轮数	通信轮数	验证时间占比
文献[19]	O(di)	3	2	>1/2
本文	0	2	1	<1/2

5.2 合约开销

根据第 4 节中的分析可知, 服务器不会合谋且正确计算全局梯度, 使自身收益最大化. 在囚徒合约, 合谋合约, 背叛合约中, 只有囚徒合约会被签署, 部署与执行合约的经济学开销如表 4 所示.

(1) 囚徒合约涉及的函数如下.

1) 初始化: 创建囚徒合约, 并初始化合约各个参数.

2) 签署合约: 客户端与服务器签署囚徒合约并提交押金.

3) 返回名单: 返回客户端名单.

4) 聚合: 服务器计算全局梯度并返回全局梯度.

5) 结算: 根据两服务器返回的全局梯度进行奖惩和押金分配.

(2) 合谋合约涉及的函数如下.

1) 初始化: 创建合谋合约, 并初始化合约各个参数.

2) 签署合约: 两服务器签署合谋合约并提交押金.

3) 协商: 两服务器协商合谋值并返回.

4) 结算: 根据两服务器返回的结果进行奖惩和押金分配.

(3) 背叛合约涉及的函数如下.

1) 初始化: 创建背叛合约, 并初始化合约各个参数.

2) 背叛: 服务器向客户端举报合谋.

3) 返回: 服务器返回全局梯度.

虽然联邦学习需要多次训练, 但是合约的初始化只需要进行一次, 同时在没有出现不诚实行为时签署合约操作也只需执行一次. 表 4 中的聚合操作是执行一次该操作的费用, 总的聚合费用会根据训练的次数和梯度的复杂度改变, 这与没有验证功能需要支付的费用一致. 若进行 n 次训练, 没有验证功能的总开销为 $n\cdot 0.4503$, 具有验证功能的总开销为 $(0.1718+0.0327+n\cdot(0.0041+0.4503)+0.0174)\cdot 2$, 由于聚合开销占主要开销, 当训练次数增加时, 具有验证功能的经济学开销几乎等于不具有验证功能的 2 倍, 如图 5 所示.

表 4 合约开销

合约	操作	开销 (gas)	开销 (美元)
囚徒合约	初始化	983 961	0.171 8
	签署合约	187 180	0.032 7
	返回名单	23 483	0.004 1
	聚合	2 578 400	0.450 3
	结算	99 704	0.017 4
合谋合约	初始化	623 506	0.109
	签署合约	93 461	0.016 3
	协商	23 527	0.004 1
	结算	83 722	0.014 6
背叛合约	初始化	301 605	0.052 7
	背叛	55 513	0.009 7
	返回	42 922	0.007 5

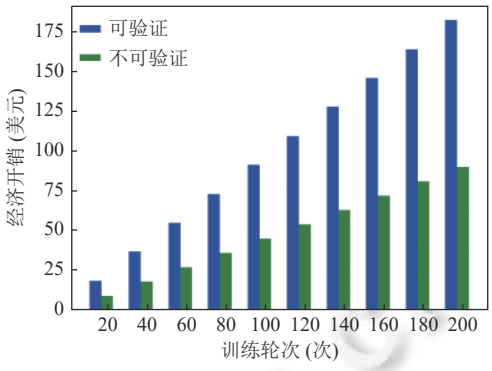

图 5 经济开销

5.3 训练开销

实验使用 100 台移动设备训练数据, 并使用两台性能相同的服务器执行全局梯度计算任务. 训练数据集采用手写数字数据集 (MNIST), 样本数据被均分置于各个移动设备用于模型训练, MNIST 训练集中共有 60 000 个训练样本和 10 000 个测试样本, 每个样本为 28×28 的图像. 训练时把每个样本图像转换为向量作为输入. 每次训练使用 1 KB 的数据作为输入, 主要验证的是通信与计算的时间开销, 不涉及数据训练阶段的开销, 所以数据训练阶段的开销不做考虑, 文中所提的总开销不包含本地训练时间.

1) 聚合开销

由于方案引入两台服务器计算全局梯度, 因此在实验中加入聚合开销的对比实验, 检测使用双服务器对聚合开销的影响. 如图 6 所示, 分别与不具备验证功能的联邦学习框架、文献 [19] 进行对比实验, 得到的时间开销与不可验证的联邦学习框架、文献 [19] 的聚合开销几乎相同, 即本文提出的方案实现验证不需要额外的聚合开销, 改变客户端的数量结论也同样成立. 此外, 客户端不需要额外的开销用于计算验证信息, 这明显优于基于密码算法的联邦学习验证方案[18,19].

2) 总开销

实验中, 在客户端对客户端上传局部梯度到收到全局梯度的时间进行测试 (除本地训练开销). 总开销与客户端数量呈正相关, 同时, 对不具备验证功能的联邦学习框架和文献 [19] (剔除了本地训练时间以及对局部梯度进行加密的时间) 进行测试, 存在相同的结论. 此外, 所提方案的总开销没有达到不具有验证功能的联邦学习框架的两倍, 因为方案中, 客户端需要上传两份局部梯度, 网络中的信息量变为原来的 2 倍, 但是通信开销并不会上升到原来的 2 倍. 这是因为向两台服务器传输数据是同时进行的, 受网络带宽的影响, 通信开销会保持在不具有验证功能的联邦学习框架的 1–2 倍之间. 因此, 验证开销在总开销中的占比低于 1/2, 而文献 [19] 的总开销高于本文, 且验证开销在总开销中的占比高于 1/2, 如图 7 所示.

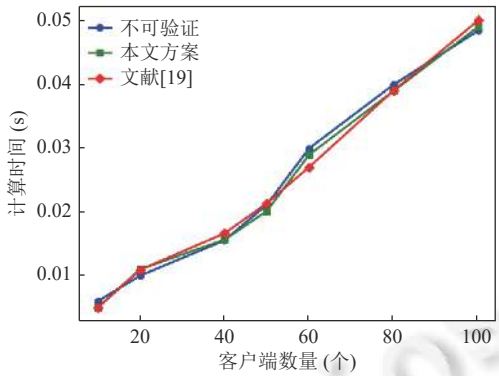

图 6 聚合开销

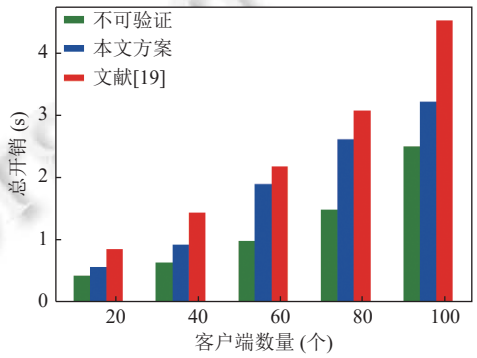

图 7 总开销

3) 客户端离线开销

验证方案支持客户端离线, 系统使用 100 个客户端参与训练. 存在两种类型的离线: ① 客户端加入模型训练, 但是在将要训练时客户端退出训练; ② 客户端只向一个服务器发送局部梯度后退出训练. 由于方案中只使用公共客户端的局部梯度计算全局梯度, 所以两种情况下客户端都没有对模型训练提供帮助, 将不会获得模型训练产生的收益. 文献 [19] 中, 客户端离线率增加时, 需要在线客户端提供更多重建离线客户端承诺的信息, 这会使通信开销增大. 相反, 在本文提出的方案中, 离线会使总开销减少, 如图 8、图 9 所示, 离线客户端越多, 总开销越低, 这是由于客户端离线, 需要聚合的局部梯度变少, 网络中的信息量减少. 图 10 展示了不同客户端离线率下验证开销在总开销中的占比, 3 种离线情况下, 验证开销不会超过总开销的 1/2. 在客户端数量为 100 时, 不同情况下各阶段的开销如表 5 所示.

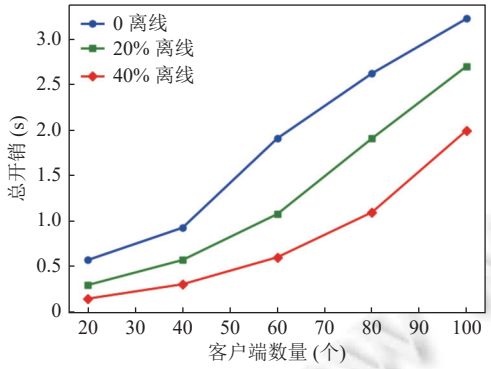

图 8 不同离线率下的总开销

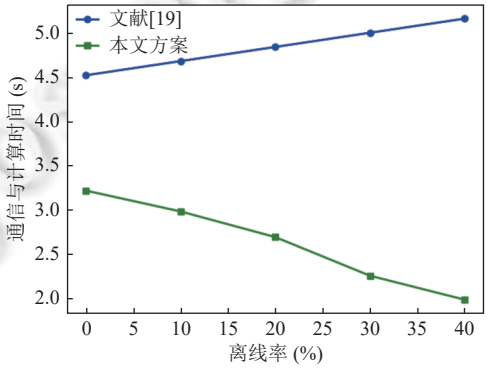

图 9 N=100 时不同离线率下的开销

4) 通信开销

实验中, 每个客户端每轮训练传递的信息为 7 KB, 如图 11 所示, 具有验证功能的联邦学习框架需要传递的数

据是没有验证功能的 2 倍. 客户端数量越多, 传入网络中的信息越多. 图 12 为每个客户端分别训练不同的轮次向服务器传递的总的信息量. 随着训练轮次的增加, 客户端向服务器发送的信息量呈线性增加.

表 6 为当 N=100 时, 不同离线率下为解决客户端离线需要的通信开销, 离线率越高获取公共客户端需要的通信开销越低, 文献 [19] 为解决客户端离线, 需要传递秘钥份额用于重建承诺, 其通信开销随离线率上升而增加.

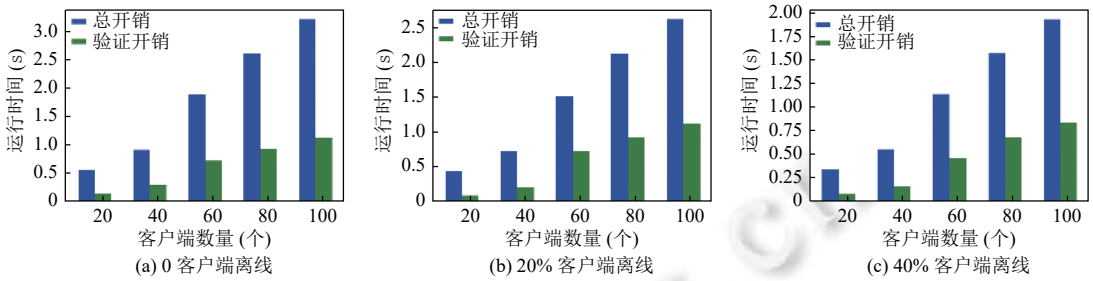

图 10 离线时验证开销在总开销中的占比

表 5 N=100 时训练总开销

客户端数量	离线率 (%)	第1轮通信开销 (ms)	聚合开销 (ms)	第2轮通信开销 (ms)	总开销 (ms)
60	0	1 872.97	29.93	0	1 902.90
	20	1 523.56	22.97	12.09	1 558.62
	40	1 239.37	17.53	11.99	1 268.89
80	0	2 578.41	39.92	0	2 618.33
	20	2 041.16	31.16	12.29	2 084.61
	40	1 538.21	23.12	11.98	1 573.31
100	0	3 173.58	48.37	0	3 221.95
	20	2 649.31	38.94	12.34	2 700.59
	40	1 906.29	29.59	12.13	1 948.01

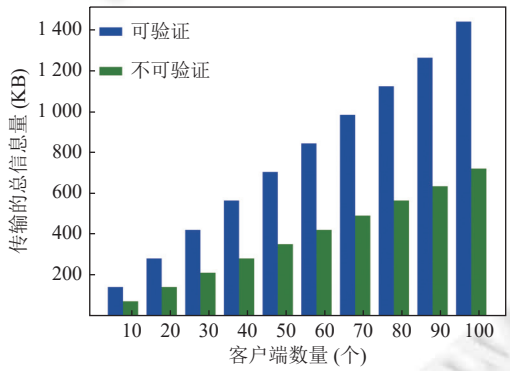

图 11 每轮训练传递的信息量

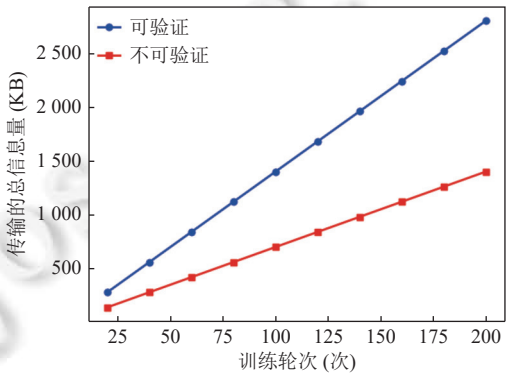

图 12 传递的总信息量

表 6 N=100 时不同离线率下的通信开销 (KB)

方案	10%	20%	30%
文献[19]	1.5	2.7	4
本文	0.26	0.24	0.2

6 总 结

本文基于复制的方式, 使用两台服务器承担全局梯度的计算任务, 通过判断全局梯度的一致性实现全局梯度的完整性验证. 其中, 通过囚徒合约阻止服务器的自利与惰性行为, 并通过背叛合约阻止服务器间的合谋. 此外, 方案通过只使用公共客户端的局部梯度进行聚合实现客户端离线. 通过理论分析和实验验证了方案在计算开销与通信轮次上都优于已有的联邦学习验证算法.

在未来的工作中, 将致力于完善方案使两台服务器不用同时实时在线, 从而解决服务器占用与网络信息量大的问题, 同时, 设计合约使服务器面对外部攻击时选择诚实执行聚合任务以获得更高奖励.

References:

[1] Ge ZQ, Song ZH, Ding SX, Huang B. Data mining and analytics in the process industry: The role of machine learning. IEEE Access, 2017, 5: 20590–20616. [doi: 10.1109/ACCESS.2017.2756872]

[2] Mohassel P, Zhang YP. SecureML: A system for scalable privacy-preserving machine learning. In: Proc. of the 2017 IEEE Symp. on Security and Privacy. San Jose: IEEE, 2017. 19–38. [doi: 10.1109/SP.2017.12]

[3] Zhang Y, Xu CX, Li HW, Yang K, Zhou JY, Lin XD. HealthDep: An efficient and secure deduplication scheme for cloud-assisted eHealth systems. IEEE Trans. on Industrial Informatics, 2018, 14(9): 4101–4112. [doi: 10.1109/TII.2018.2832251]

[4] Ahiska K, Ozgoren MK, Leblebicioglu MK. Autopilot design for vehicle cornering through icy roads. IEEE Trans. on Vehicular Technology, 2018, 67(3): 1867–1880. [doi: 10.1109/TVT.2017.2765245]

[5] Yang Q, Liu Y, Chen TJ, Tong YX. Federated machine learning: Concept and applications. ACM Trans. on Intelligent Systems and Technology, 2019, 10(2): 12. [doi: 10.1145/3298981]

[6] Mothukuri V, Parizi RM, Pouriyeh S, Huang Y, Dehghantanha A, Srivastava G. A survey on security and privacy of federated learning. Future Generation Computer Systems, 2021, 115: 619–640. [doi: 10.1016/j.future.2020.10.007]

[7] Sattler F, Wiedemann S, Müller KR, Samek W. Robust and communication-efficient federated learning from Non-i.i.d. data. IEEE Trans. on Neural Networks and Learning Systems, 2020, 31(9): 3400–3413. [doi: 10.1109/TNNLS.2019.2944481]

[8] Nasr M, Shokri R, Houmansadr A. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In: Proc. of the 2019 IEEE Symp. on Security and Privacy. San Francisco: IEEE, 2019. 739–753. [doi: 10.1109/SP.2019.00065]

[9] Yin LH, Feng JY, Xun H, Sun Z, Cheng XC. A privacy-preserving federated learning for multiparty data sharing in social IoTs. IEEE Trans. on Network Science and Engineering, 2021, 8(3): 2706–2718. [doi: 10.1109/TNSE.2021.3074185]

[10] Sattler F, Wiedemann S, Müller KR, Samek W. Sparse binary compression: Towards distributed deep learning with minimal communication. In: Proc. of the 2019 Int'l Joint Conf. on Neural Networks. Budapest: IEEE, 2018. 1–8. [doi: 10.1109/IJCNN.2019.8852172]

[11] Aji AF, Heafield K. Sparse communication for distributed gradient descent. In: Proc. of the 2017 Conf. on Empirical Methods in Natural Language Processing. Copenhagen: Association for Computational Linguistics, 2017. 440–445. [doi: 10.18653/v1/D17-1045]

[12] Fang C, Guo YB, Hu YJ, Ma BW, Feng L, Yin AQ. Privacy-preserving and communication-efficient federated learning in Internet of Things. Computers & Security, 2021, 103: 102199. [doi: 10.1016/j.cose.2021.102199]

[13] Phong LT, Aono Y, Hayashi T, Wang LH, Moriai S. Privacy-preserving deep learning via additively homomorphic encryption. IEEE Trans. on Information Forensics and Security, 2018, 13(5): 1333–1345. [doi: 10.1109/TIFS.2017.2787987]

[14] Triastcyn A, Faltings B. Federated learning with Bayesian differential privacy. In: Proc. of the 2019 IEEE Int'l Conf. on Big Data. Los Angeles: IEEE, 2019. 2587–2596. [doi: 10.1109/BigData47090.2019.9005465]

[15] Chamikara MAP, Bertok P, Khalil I, Liu D, Camtepe S. Privacy preserving distributed machine learning with federated learning. Computer Communications, 2021, 171: 112–125. [doi: 10.1016/j.comcom.2021.02.014]

[16] Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan HB, Patel S, Ramage D, Segal A, Seth K. Practical secure aggregation for privacy-preserving machine learning. In: Proc. of the 2017 ACM SIGSAC Conf. on Computer and Communications Security. Dallas: AMC, 2017. 1175–1191. [doi: 10.1145/3133956.3133982]

[17] Abadi M, Chu A, Goodfellow I, McMahan HB, Mironov I, Talwar K, Zhang L. Deep learning with differential privacy. In: Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security. Vienna: ACM, 2016. 308–318. [doi: 10.1145/2976749.2978318]

[18] Xu GW, Li HW, Liu S, Yang K, Lin XD. VerifyNet: Secure and verifiable federated learning. IEEE Trans. on Information Forensics and

Security, 2019, 15: 911–926. [doi: 10.1109/TIFS.2019.2929409]
[19] Guo XJ, Liu ZL, Li J, Gao JQ, Hou BY, Dong CY, Baker T. VeriFL: Communication-efficient and fast verifiable aggregation for federated learning. IEEE Trans. on Information Forensics and Security, 2021, 16: 1736–1751. [doi: 10.1109/TIFS.2020.3043139]
[20] Walfish M, Blumberg AJ. Verifying computations without reexecuting them. Communications of the ACM, 2015, 58(2): 74–84. [doi: 10.1145/2641562]
[21] Dong CY, Wang YL, Aldweesh A, McCorry P, van Moorsel A. Betrayal, distrust, and rationality: Smart counter-collusion contracts for verifiable cloud computing. In: Proc. of the 2017 ACM SIGSAC Conf. on Computer and Communications Security. Dallas: ACM, 2017. 211–227. [doi: 10.1145/3133956.3134032]
[22] Chen ZR, Tian YL, Xiong JB, Peng CG, Ma JF. Towards reducing delegation overhead in replication-based verification: An incentive-compatible rational delegation computing scheme. Information Sciences, 2021, 568: 286–316. [doi: 10.1016/J.INS.2021.03.047]
[23] Kotla R, Alvisi L, Dahlin M, Clement A, Wong E. Zyzzyva: Speculative byzantine fault tolerance. In: Proc. of the 21st ACM SIGOPS Symp. on Operating Systems Principles. Stevenson: ACM, 2007. 45–59. [doi: 10.1145/1294261.1294267]
[24] Distler T, Cachin C, Kapitza R. Resource-efficient byzantine fault tolerance. IEEE Trans. on Computers, 2016, 65(9): 2807–2819. [doi: 10.1109/TC.2015.2495213]
[25] McMahan B, Moore E, Ramage D, Hampson S, Arcas BA. Communication-efficient learning of deep networks from decentralized data. In: Proc. of the 20th Int'l Conf. on Artificial Intelligence and Statistics. Fort Lauderdale: PMLR, 2017. 1273–1282.
[26] Zhou J, Fang GY, Wu N. Survey on security and privacy-preserving in federated learning. Journal of Xihua University (Natural Science Edition), 2020, 39(4): 9–17 (in Chinese with English abstract).
[27] Cao D, Chang S, Lin ZJ, Liu GH, Sun DH. Understanding distributed poisoning attack in federated learning. In: Proc. of the 25th IEEE Int'l Conf. on Parallel and Distributed Systems (ICPADS). Tianjin: IEEE, 2019. 233–239. [doi: 10.1109/ICPADS47876.2019.00042]
[28] Zantedeschi V, Nicolae MI, Rawat A. Efficient defenses against adversarial attacks. In: Proc. of the 10th ACM Workshop on Artificial Intelligence and Security. Dallas: ACM, 2017. 39–49. [doi: 10.1145/3128572.3140449]
[29] Zhou CY, Fu AM, Yu S, Yang W, Wang HQ, Zhang YQ. Privacy-preserving federated learning in fog computing. IEEE Internet of Things Journal, 2020, 7(11): 10782–10793. [doi: 10.1109/JIOT.2020.2987958]
[30] Li QX, Tian YL. Rational delegation computing using information theory and game theory approach. In: Proc. of the 26th Int'l Conf. on Multimedia Modeling. Daejeon: Springer, 2020. 669–680. [doi: 10.1007/978-3-030-37734-2_54]
[31] Kreps DM, Wilson R. Sequential equilibria. Econometrica, 1982, 50(4): 863–894. [doi: 10.2307/1912767]
[32] Gibbons R. A Primer in Game Theory. Harlow: Prentice Education Limited, 1992. 1–267.

附中文参考文献:

[26] 周俊, 方国英, 吴楠. 联邦学习安全与隐私保护研究综述. 西华大学学报(自然科学版), 2020, 39(4): 9–17.

吴柿红(1995－), 女, 硕士, 主要研究领域为联邦学习, 密码学.

田有亮(1982－), 男, 博士, 教授, 博士生导师, CCF 专业会员, 主要研究领域为博弈论, 密码学与安全协议, 大数据隐私保护.