计 算 机 研 究 与 发 展
Journal of Computer Research and Development
https://crad.ict.ac.cn
62(2): 472−487, 2025

基于用户级别本地化差分隐私的联邦学习方法

张啸剑　张雷雷　张治政
（河南财经政法大学计算机与信息工程学院　郑州　450016）
（xjzhang82@alu.ruc.edu.cn）

Federated Learning Method Under User-Level Local Differential Privacy

Zhang Xiaojian, Zhang Leilei, and Zhang Zhizheng
（*School of Computer & Information Engineering*, *Henan University of Economics and Law*, *Zhengzhou* 450016）

Abstract　Federated learning with user-level local differential privacy (ULDP) has attracted considerable research attention in recent years. The trade-off among federated data types, the mechanism of clipping local updates, the allocation of privacy budget, and user dropout directly constrain the accuracy of the global learning model. All existing federated learning methods are vulnerable to handling these problems. To remedy the deficiency caused by the current methods, we employ ULDP to propose an efficient algorithm, called ULDP-FED, to achieve global federated optimization. ULDP-FED can simultaneously handle IID and non-IID federated data types. Compared with those methods with fixed clipping thresholds, ULDP-FED uses a threshold dynamic decay strategy to balance the noise error caused by the Gauss mechanism and the bias caused by update clipping. To allocate the privacy budget of each user carefully, in each round, ULDP-FED relies on the similarity to replace the current local update with the historical noise updates. If the historical updates are obtainted, the user only sends the index of the historical update to the server, which can reduce the communication cost. ULDP-FED is compared with existing methods over MNIST and CIFAR 10 datasets. The experimental results show that our algorithm outperforms its competitors, and achieves the accurate results of federated learning.

Key words　differential privacy; user-level differential privacy (ULDP); federated learning; federated optimization; local gradient update

摘　要　基于用户级别本地化差分隐私的联邦学习得到了研究者的广泛关注，联邦数据的类型、本地更新的裁剪、隐私预算的分配以及用户掉线问题直接制约着全局联邦学习模型的精度. 针对现有方法难以有效应对该类问题的不足，提出了一种有效基于用户级别本地化差分隐私的联邦学习算法 ULDP-FED. 该算法能够同时处理同分布与非同分布的联邦数据类型；不同于现有的固定裁剪设置方法，ULDP-FED 算法采用裁剪阈值动态衰减策略来实现高斯机制造成的误差与裁剪造成的偏差之间的均衡；为了节省用户手中的隐私预算，该算法通过遍历用户所有历史本地噪音梯度更新来寻找当前轮本地梯度更新的替代更新. 若存在高度相似的历史更新，用户只需要上传该历史梯度更新的索引即可，进而减少了用户与服务器之间的通信代价. ULDP-FED 算法与现有算法在 MNIST 和 CIFAR 10 数据上的实验结果表明，其模型精度均高于同类方法.

关键词　差分隐私；用户级别差分隐私；联邦学习；联邦优化；本地梯度更新

收稿日期：2023-03-17；**修回日期**：2024-01-08

基金项目：国家自然科学基金项目（62072156, 61502146, 91646203, 91746115）；河南省高等学校重点科研项目计划基础研究专项（25ZX012）
This work was supported by the National Natural Science Foundation of China (62072156, 61502146, 91646203, 91746115) and the Basic Research Special Projects of Key Research Projects in Higher Education Institutes in Henan Province (25ZX012).

中图法分类号 TP392

DOI: 10.7544/issn1000-1239.202330167 **CSTR:** 32373.14.issn1000-1239.202330167

联邦学习[1]是打破数据孤岛、实现数据共享的常用分布式学习架构. 该框架遵循用户数据不离开本地且保护其数据隐私的原则, 协同完成某种学习任务. 在该学习框架中, 每轮被选中的用户需要把本地梯度或者本地梯度更新报告给服务端. 然而, 本地梯度或者本地梯度更新作为新型数据, 无论是在全局模型训练阶段还是模型推理阶段均面临多种攻击. 例如, 模型投毒攻击[2-3]、拜占庭攻击[4]、梯度反演攻击[5-6]等. 由于联邦学习训练过程需要服务器与部分用户进行多轮梯度或者梯度更新的交换, 恶意攻击者可以利用梯度反演攻击[5-6]重构某个用户的部分原始数据, 也可以利用中间交换的参数对用户数据进行成员攻击[7]. 为了保护用户隐私梯度或者梯度更新以及防止梯度反演攻击等, 系列基于示例级别的差分隐私(instance-level differential privacy, ILDP)[8−9]与基于用户级别的差分隐私(user-level differential privacy: ULDP)[10-13]的梯度更新保护方法被相继提出. 文献[8−9]中的方法通常假设每个用户仅贡献 1 个训练示例或者样本. 而实际的联邦系统中, 每个用户通常贡献多个样本, 例如, 某个病人在一家医院产生多条健康医疗记录, 以及通过该医院的 APP 产生挂号记录等, 该病人希望自己的所有记录得到保护. ULDP 保护模型更适合此类应用需求. 从用户级别来保护每个用户的所有数据, 而不是单个训练样本. 目前, 基于 ULDP 模型下的联邦学习方法包括 DP-FedAvg[10], DP-FedSGD[10], CS-DPFL[11], CRD[12], NbAFL[13], DP-FedAvgAC[14], D2P-Fed[15], DP-FedSAM[16-17], DP-FedAvg-blur[18], DP-FedAvg-blurs[18]等. DP-FedAvg, DP-FedSGD, CS-DPFL 这 3 种方法把 ULDP 模型引入联邦学习, 其主要区别在于 DP-FedAvg 与 DP-FedSGD 利用固定阈值裁剪梯度更新, 而 CS-DPFL 利用 L2 范式度量梯度更新幅值, 并用 L2 范式的中位数裁剪梯度更新. 不同于上述 3 种方法, CRD[19]利用矩审计机制实现了基于用户级别的本地差分隐私联邦学习, 然而该方法没有考虑裁剪阈值对梯度的影响. NbAFL 方法在服务器端和客户端均加入了高斯噪声来提高隐私保护程度, 然而, 该方法训练获得模型的可用性较低. DP-FedAvgAC 在 DP-FedAvg 的基础上考虑了裁剪阈值对模型可用性的影响, 提出了牺牲部分隐私预算计算自适应阈值的方法. 该方法裁剪阈值的自适应性提高了模型可用性, 却增加了算法的复杂性. DP-FedSAM 仅在 DP-FedAvg 的基础上引入 SAM 优化器, DP-FedAvg-blur 与 DP-FedAvg-blurs 分别在 DP-FedAvg 基础上增加了正则化与稀疏化操作. 尽管上述方法都是基于 ULDP 模型下的联邦学习方法, 但是依然存在 2 点不足: 1)大都直接将高斯噪声添加在梯度更新上, 没有考虑梯度更新在训练过程中的前后变化情况. 此外, 很难找到合适的固定裁剪阈值. 裁剪阈值设置过小导致摈弃梯度的信息过多, 偏差过大; 而裁剪阈值设置过大会导致本地添加的噪音过大. 2)大都采用基本序列组合定理、强组合定理或者矩审计定理确保每轮迭代满足 (ε, δ)-差分隐私. 然而, 基本序列组合定理不适合迭代总轮数 T 过大的情况, T 过大会造成过大的隐私损失, 即 $(T\varepsilon, T\delta)$-DP. 强组合定理虽然对隐私损失进行了定界, 然而这种定界仅给出理想的 ε 与 δ 的关系. 矩审计机制给出了比强组合定理更紧的界. 然而, 该方法容易受迭代轮数 T 的影响. 在实际联邦系统中, 参与每轮训练的用户时常会因为网络或设备故障无法上传自己的本地更新而发生掉线现象. 因此, 每个参与用户并非在 T 轮中每轮都贡献自己的数据.

为了探寻前后迭代轮次之间梯度更新的关系, 我们基于 MNIST 和 CIFAR 10 数据集测试了梯度更新符号之间的相似性. 设置裁剪阈值为 2, 用户总数为 1 000, 迭代轮次 T=1 000. 从图 1 中可以发现在 1 000 轮的迭代过程中, MNIST 数据集相邻 2 轮全局梯度更新的符号最大相似度为 0.82, 最小为 0.47, 超过 58% 左右的全局梯度更新的相似度大于 0.56. CIFAR 10 数据集在 1 000 轮的训练过程中, 相邻 2 轮的全局更新相似度最大为 0.96, 最小为 0.47, 超过 60% 左右的全局更新相似度大于 0.6.

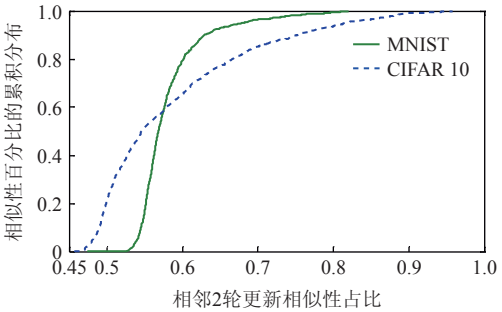


Fig. 1 Sign similarity changing of global updates

图 1 全局更新符号相似性变化

为了验证设置固定阈值的不合理性,我们基于MNIST和CIFAR 10数据集度量裁剪阈值与迭代轮数的关系.根据图2可以看出,随着迭代轮数的增加,无论是MNIST数据还是CIFAR 10数据,相邻2轮全局梯度更新的符号相似性逐渐减小.这是由于在训练初期,模型的损失很大,梯度符号包含着更多的信息;随着训练进行,模型逐渐收敛,梯度符号所包含的信息逐渐减少.如果采用恒定的裁剪阈值,不符合本地梯度更新的变化趋势.恒定的裁剪阈值会带来恒定的高斯噪音.由图2可知,训练初期梯度包含的信息较多,应该添加较大的噪音.而在训练后期,应添加较少的噪音.

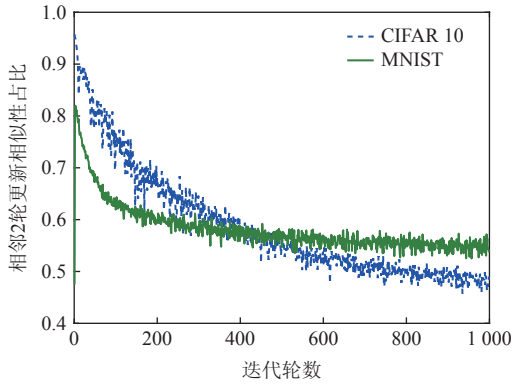


Fig. 2 Gradient changing based on MNIST and CIFAR 10

图2 基于MNIST与CIFAR 10的梯度变化

基于以上2组实验结果与现有方法的不足,我们提出了一种基于ULDP的联邦学习方法ULDP-FED.本文主要贡献有4点:

1)为了更好地利用矩审计机制,本文利用梯度更新回溯策略寻找历史记录中有哪些梯度更新可以替代当前轮的梯度更新.通过计算当前轮梯度更新与全局更新的相似性来判断是否触发替换操作.如果触发替换,当前轮在不消耗隐私预算的情况下仅发送给服务器1个标记信号即可.

2)不同于现有自适应裁剪策略,本文根据梯度更新的变化趋势,提出了一种动态衰减梯度更新裁剪策略.该策略结合前后迭代轮次梯度更新的变化情况,动态调整下一轮裁剪阈值的大小.

3)为了解决用户掉线问题,本文限制每个用户的参与训练轮数.我们假设每个用户最多参加r轮的训练,即每个用户以r/T概率参与每轮训练,进而减少训练的交互次数,降低通信代价.

4)理论分析了ULDP-FED的收敛性以及满足(ε, δ)-ULDP.通过真实数据实验分析,ULDP-FED具有较高的可用性.

1 定义与问题

1.1 用户级别本地化差分隐私

用户级别差分隐私(ε, δ)-ULDP实际上是本地差分隐私模型的一种拓展,其目的是保护所有与用户关联的数据.该模型的形式化定义如下所示.

定义 1. (ε, δ)-ULDP. 设D与D'为近邻数据集且相差1个用户所有数据,$D, D' \in \mathcal{D}$.给定随机算法M,$Range(M)$为M的输出域,若M在D与D'上任意输出结果$o(o \in Range(M))$的概率满足下列不等式,则M满足(ε, δ)-ULDP.

$$Pr\left[M(p_i)=o\right] \leqslant \mathrm{e}^{\varepsilon} \times Pr\left[M\left(p_j\right)=o\right]+\delta, \tag{1}$$

其中ε为隐私预算,δ为隐私损失概率.(ε, δ)-ULDP保护模型通常具有序列组合性质.

性质1[20]. 给定D与n个随机算法$M_1, M_2, \cdots, M_n$,且M_i满足$(\varepsilon_i, \delta_i)$-ULDP,则在$D$上的序列组合满足$(\varepsilon, \delta)$-ULDP,且$\varepsilon=\sum_{i=1}^{n}\varepsilon_i$, $\delta=\sum_{i=1}^{n}\delta_i$.

定义 2. L2-敏感度.给定函数$f: \mathcal{D} \to \mathbb{R}^d$,且$D, D' \in \mathcal{D}$.则$f$的L2-敏感度可表示为

$$\Delta S=\max_{D, D' \in \mathcal{D}} \|f(D)-f(D')\|_2. \tag{2}$$

定理 1. 高斯机制[21]. 给定函数f: $\mathcal{D} \to \mathbb{R}^d$,且$f$的L2-敏感度为$\Delta S$.高斯机制$M$可以表示为

$$M=f(D)+\mathcal{N}\left(0, \sigma^2 \Delta S^2 \boldsymbol{I}_d\right), \tag{3}$$

则M满足(ε, δ)-ULDP.其中,$\sigma^2 \Delta S^2 \boldsymbol{I}_d$表示高斯分布的方差,$\boldsymbol{I}_d$表示单位矩阵.

1.2 联邦优化

设$U=\{u_1, u_2, \cdots, u_n\}$是所有用户的集合,且$|U|=n$.$D=\{D_1, D_2, \cdots, D_n\}$表示所有用户数据组成的分布式数据集,其中$D_i$表示第$i$个用户的本地数据集.设$F_i$表示第$i$个用户的本地损失函数,$\boldsymbol{W}_i$为第$i$个用户的模型参数.第$i$个用户的本地损失函数可表示为

$$F_i(\boldsymbol{W}_i, D_i)=\frac{1}{|D_i|}\sum_{s \in D_i} F_i(\boldsymbol{W}_i, s), \tag{4}$$

其中s表示D_i中的一个样本.

基于分布式数据集D的联邦优化函数为:

$$\arg\min_{\boldsymbol{W}} F(\boldsymbol{W}, D)=\sum_{i \in U} p_i F_i(\boldsymbol{W}_i, D_i), \tag{5}$$

其中$p_i=|D_i|/|D|$,且$|D_i|$与$|D|$分别表示D_i和D的大小.$F(\cdot)$为全局损失函数,$\boldsymbol{W}$全局模型参数.

为了解决式(5)中的联邦优化问题,通常采用基

于随机梯度下降的联邦平均方法 FedAvg. 通过 T 轮迭代达到全局模型收敛，即可获得 $\boldsymbol{W}$. FedAvg 方法通常包括 4 个步骤：1）用户选择. 在第 t 轮迭代中，服务器随机选取$|U^t|$个用户. 2）本地更新. U^t集合中每个用户从服务器下载全局模型参数$\boldsymbol{W}^{t-1}$，结合自身本地数据 D_i 进行更新，进而获得本地模型参数 $\boldsymbol{W}_i^t$，如式(6)所示. 最后将 $\boldsymbol{W}_i^t$ 上传给服务器. 3）全局聚合. 服务器收集U^t中所有用户的模型参数进行聚合，如式(7)所示. 4）迭代. 重复 1)~3)过程.

$$\boldsymbol{W}_i^t = \boldsymbol{W}^{t-1} - \eta_i \nabla F_i(\boldsymbol{W}^{t-1}, D_i), \tag{6}$$

其中 η_i 表示本地学习率.

$$\boldsymbol{W}^t = \sum_{i \in U^t} p_i \boldsymbol{W}_i^t. \tag{7}$$

在联邦优化过程中，为了避免泄露用户的梯度更新或者本地模型参数，需要修改 FedAvg 方法中的步骤 2). 首先由式(6)获得第 t 轮的本地更新$\boldsymbol{\Delta}_i^t$，如式(8)所示：

$$\boldsymbol{\Delta}_i^t = \boldsymbol{W}_i^t - \boldsymbol{W}^{t-1}. \tag{8}$$

然后，对$\boldsymbol{\Delta}_i^t$进行裁剪，如式(9)所示：

$$\bar{\boldsymbol{\Delta}}_i^t = \boldsymbol{\Delta}_i^t \Big/ \max\left(1, \frac{\|\boldsymbol{\Delta}_i^t\|_2}{C}\right), \tag{9}$$

其中 C 与$\bar{\boldsymbol{\Delta}}_i^t$表示裁剪阈值和裁剪后的本地更新.

最后，结合定理 1 对$\bar{\boldsymbol{\Delta}}_i^t$添加高斯噪音，进而获得本地噪音更新$\tilde{\boldsymbol{\Delta}}_i^t$，如式(10)所示：

$$\tilde{\boldsymbol{\Delta}}_i^t = \bar{\boldsymbol{\Delta}}_i^t + \mathcal{N}(0, C^2\sigma^2 \boldsymbol{I}_d). \tag{10}$$

1.3 问题定义

根据式(9)(10)可知，每轮迭代均需要分割隐私预算 ε. 假设 T 轮迭代后全局模型收敛，则每轮迭代花费的隐私预算为 ε/T. 由于历史模型参数与当前轮的模型参数存在相似性，则可以利用此特性来节省部分隐私预算. 式(9)(10)中裁剪阈值 C 的设定是个大的挑战，不合适的 C 值通常会造成较大的高斯噪音和裁剪偏差. 此外，为了模拟现实环境用户掉线的情况，每个用户最多参与 r 轮训练，即每个用户以概率 r/T 参与每轮训练. 因此，本文要解决的问题是设计满足 (ε, δ)-ULDP 的联邦学习方法同时，要尽可能获得高可用性、降低通信代价以及安全的全局模型参数.

2 基于 (ε, δ)-ULDP 的联邦学习算法

在设计满足 (ε, δ)-ULDP 的联邦学习算法时需要考虑 3 条原则：1）保护用户数据安全与隐私；2）尽可能地提高模型可用性；3）尽可能地降低通信代价. 针对原则 1~3，本文采用本地差分隐私、基于历史本地更新的回溯技术以及裁剪阈值动态衰减技术来设计高可用的联邦学习算法. 基于上述 3 种原则，本文设计了基于 (ε, δ)-ULDP 的联邦学习算法 ULDP-FED，该算法的具体细节如算法 1 所示.

算法 1. ULDP-FED.

输入：数据集 $D=\{D_1, D_2, \cdots, D_n\}$，隐私预算 ε，全局学习率 η_g，本地学习率 η_i，本地完整的训练次数Γ，回溯阈值 τ；

输出：全局模型参数 $\boldsymbol{W}$.

服务器端：

① 广播全局初始化模型参数 $\boldsymbol{W}^0$，全局初始化更新$\boldsymbol{\Delta}^0$，以及迭代总轮数 T；

② for each iteration t=1, 2, ⋯, T do

③ for all user $u_i \in U^t$ do in parallel /*被采样的用户并行计算*/

④ $\tilde{\boldsymbol{\Delta}}_i^t \leftarrow UserUpdate\ (D_i, \boldsymbol{W}^{t-1}, \boldsymbol{\Delta}^{t-1})$;

⑤ $\tilde{\boldsymbol{\Delta}}^t \leftarrow \sum_{i \in U^t} p_i \tilde{\boldsymbol{\Delta}}_i^t$; /*计算全局噪音更新$\tilde{\boldsymbol{\Delta}}^t$*/

⑥ $\boldsymbol{W}^t = \boldsymbol{W}^{t-1} - \eta_g \tilde{\boldsymbol{\Delta}}^t$; /*更新全局模型参数*/

⑦ end for

⑧ end for

⑨ return $\boldsymbol{W}^{t-1}$.

用户端：

/*$UserUpdate$ 函数*/

⑩ $\boldsymbol{W}_i^t \leftarrow \boldsymbol{W}^{t-1}$, $\tilde{\boldsymbol{\Delta}}^{t-1}$; /*下载第 $t-1$ 轮的全局模型$\boldsymbol{W}^{t-1}$与全局噪音更新$\tilde{\boldsymbol{\Delta}}^{t-1}$*/

⑪ for each local epoch $e \in \{1, 2, \cdots, \Gamma\}$ do

⑫ for each batch b do

⑬ $\boldsymbol{W}_i^t \leftarrow \boldsymbol{W}_i^t - \eta_i \nabla F_i(\boldsymbol{W}_i^t, b)$;

⑭ end for

⑮ end for

⑯ $\boldsymbol{\Delta}_i^t \leftarrow \boldsymbol{W}_i^t - \boldsymbol{W}^{t-1}$; /*计算本地更新*/

⑰ $C^t = C^{t-1} \mathrm{e}^{-\beta k}$; /*进行阈值衰减*/

⑱ $\bar{\boldsymbol{\Delta}}_i^t \leftarrow \boldsymbol{\Delta}_i^t / \max\left(1, \frac{\|\boldsymbol{\Delta}_i^t\|_2}{C^t}\right)$; /*裁剪*/

⑲ $\tilde{\boldsymbol{\Delta}}_i^t \leftarrow \bar{\boldsymbol{\Delta}}_i^t + \mathcal{N}(0, \sigma^2 (C^t)^2 \boldsymbol{I}_d)$; /*加噪*/

⑳ $\tilde{\boldsymbol{\Delta}}_i^t \leftarrow CheckSimilarity(\tilde{\boldsymbol{\Delta}}_i^t, \tilde{\boldsymbol{\Delta}}^{t-1})$; /*计算相似性*/

㉑ return $\tilde{\boldsymbol{\Delta}}_i^t$.

/*$CheckSimilarity$ 函数*/

㉒ 利用式(11)或者式(13)计算$\tilde{\boldsymbol{\Delta}}_i^t$与$\tilde{\boldsymbol{\Delta}}^{t-1}$之间的相似度 $s(\tilde{\boldsymbol{\Delta}}_i^t, \tilde{\boldsymbol{\Delta}}^{t-1})$;

㉓ if $s(\tilde{\boldsymbol{\Delta}}_i^t, \tilde{\boldsymbol{\Delta}}^{t-1}) < \tau$ then /*小于阈值则回溯*/

㉔ $(i,k)=\{(i,k)|\,k\in\{1,2,\cdots,t-1\}\wedge\max\{s(\tilde{\Delta}_i^{t-1},\tilde{\Delta}^{t-1}),\cdots,s(\tilde{\Delta}_i^{k},\tilde{\Delta}^{t-1})\cdots\}\}$, 其中 i 与 k 分别表示第 i 用户, 用户 i 的历史本地更新中第 k 个本地更新的索引;

㉕ return (i,k);

㉖ else

㉗ return $\tilde{\Delta}_i^t$; /*大于阈值时返回当前轮本地更新*/

㉘ end if

ULDP-FED 算法包含服务器端与用户端 2 个角色. 服务器端聚合每轮用户上传的本地更新, 以及更新全局模型参数, 并将最新的全局模型参数广播给用户(行②~⑥). 参与每轮训练的所有用户利用 *UserUpdate* 函数计算当前轮的本地更新(行④). 在 *UserUpdate* 函数中, 每个用户首先下载上一轮的全局噪音更新与全局模型参数(行⑩), 利用本地数据更新全局模型(行⑬), 进而获得本地更新(行⑯). 利用动态衰减阈值裁剪该本地更新后, 添加相应的高斯噪音(行⑰~⑲). 为了节省用户的隐私预算, 利用 *CheckSimilarity* 函数(行⑳)在该用户历史本地噪音更新中寻找当前本地更新的替代品. 在 *CheckSimilarity* 函数中, 计算上一轮全局噪音更新与当前轮噪音本地更新之间的相似性(行㉒), 如果相似性值小于给定的阈值, 我们需要在该用户历史本地噪音更新中进行回溯操作. 即是在历史噪音更新中寻找与上一轮全局更新最相似的更新, 来替代当前轮的本地更新(行㉓~㉕). 否则该用户上传当前轮的本地噪音更新(行㉗).

2.1 本地更新回溯

结合图 2 与 ULDP-FED 算法可知, 全局模型在训练初期, 本地更新蕴含着丰富的信息. 而在模型训练后期, 本地更新趋于平滑. 基于此, 为了提高全局模型的可用性、节省用户手中的隐私预算, 以及减少用户与服务器之间的通信代价, 本文设计了一种本地更新回溯机制. 该机制的主要思想是利用本地更新的历史数据替代当前轮的本地更新. 而如何在历史本地更新中寻找替代更新面临 2 个问题: 1)为了与全局更新收敛方向保持一致, 我们不能直接计算当前本地更新与历史本地更新的相似性. 而是利用上一轮的全局更新作为介质, 来计算当前本地更新、历史更新与该介质的相似性, 进而判断是上传历史更新还是当前轮的本地更新. 2)如何计算本地更新与上一轮全局更新相似性, 以及历史本地更新与上一轮全局更新的相似性. 为了解决上述 2 种问题, 本文设计了 2 种相似性计算方法, 具体操作为:

由于本地更新和全局更新本质是张量且具有方向性, 我们采用文献 [22] 中的符号计数方法或者余弦计算方法来度量两者之间的相似性. 具体表示如式(11)(13)所示. 假设当前轮为第 t 轮, 用户 i 在第 t 轮的本地噪音更新为 $\tilde{\Delta}_i^t$. $\tilde{\Delta}^{t-1}$ 表示第 $t-1$ 轮的全局噪音更新.

$$s(\tilde{\Delta}_i^t,\tilde{\Delta}^{t-1})=\frac{1}{l}\sum_{j=1}^{l}I(\mathrm{sgn}(\tilde{\Delta}_{i,j}^t)=\mathrm{sgn}(\tilde{\Delta}_j^{t-1})),\tag{11}$$

其中 l 表示本地更新张量的元素个数, $\mathrm{sgn}(\tilde{\Delta}_{i,j}^t)$与 $\mathrm{sgn}(\tilde{\Delta}_j^{t-1})$分别表示张量 $\tilde{\Delta}_{i,j}^t$ 与 $\tilde{\Delta}_j^{t-1}$ 第 j 位对应的符号 (−, +, 0). $I(\cdot)$ 为标识函数, 当 $\mathrm{sgn}(\tilde{\Delta}_{i,j}^t)=\mathrm{sgn}(\tilde{\Delta}_j^{t-1})$时, $I(\cdot)$=1, 否则 $I(\cdot)$=0.

同理, 在用户 i 的历史本地噪音更新中寻找与第 $t-1$ 轮全局噪音更新的相似性如式(12)所示:

$$s(\tilde{\Delta}_i^k,\tilde{\Delta}^{t-1})=\frac{1}{l}\sum_{j=1}^{l}I(\mathrm{sgn}(\tilde{\Delta}_{i,j}^k)=\mathrm{sgn}(\tilde{\Delta}_j^{t-1})),\tag{12}$$

其中 k 表示历史本地更新中第 k 轮的噪音更新索引.

此外, 我们利用余弦函数度量本地噪音更新与全局噪音更新之间的相似性, 以及度量历史本地噪音更新与第 $t-1$ 轮全局噪音更新的相似性. 具体计算方式如式(13)所示:

$$s(\tilde{\Delta}_i^t,\tilde{\Delta}^{t-1})=\cos(\tilde{\Delta}_i^t,\tilde{\Delta}^{t-1})=\frac{\tilde{\Delta}_i^t\cdot\tilde{\Delta}^{t-1}}{\|\tilde{\Delta}_i^t\|\times\|\tilde{\Delta}^{t-1}\|}.\tag{13}$$

由式(11)~(13)可知, 若上一轮的全局噪音更新或者用户 i 的历史本地噪音更新能够替代当前轮中该用户的本地噪音更新, 我们只需要发送 (i,k) 给服务器即可(参阅 *CheckSimilarity* 的行㉔), 不需要发送维度为 d 的整个张量. 进而可以节省本轮隐私预算, 减少用户与服务器之间的通信代价. 如若仅发送 (i,k) 给服务器, 则该轮的通信代价为 $O(1)$.

2.2 隐私预算分配

如果用户 i 在当前轮利用式(11)(13)寻找到自身本地更新的替代更新, 则用户 i 本轮的隐私预算就会被节省下来. 然而, 节省下来的隐私预算如何应用到将来的迭代中是需要解决的问题. 假设每个用户在 T 轮迭代中最多参加 r 轮迭代. t 表示迭代的当前轮, a 表示用户 i 当前已参与的轮数($a<r$). 给定隐私预算 ε, 在规定的 r 轮迭代中, 每轮预分配的隐私预算为 ε/r. 假设 j 表示用户 i 在 $\{1,2,\cdots,a\}$轮中首次发生回溯的轮次索引. 用户 i 在 r 轮迭代中的隐私预算分配可分为如下情况进行计算:

1)用户 i 首次参与全局训练, 则 i 的可用轮数

为 r.

①当 $r\leqslant T-t+1$ 时，算法整体可训练的轮数覆盖了用户 i 的可参与轮数 r，此时该用户当前的隐私预算为 ε/r；

②当 $r>T-t+1$ 时，用户 i 可参与训练的轮数大于整体剩余的训练轮数，此时该用户最多可参与的轮数为 $T-t+1$，当前消耗的隐私预算为 $\varepsilon/T-t+1$.

2）用户 i 非首次参与全局训练，该用户可用轮数为 $r-a$. 此时，算法整体运行到第 t 轮. 假设在 $\{1, 2, \cdots, a\}$ 轮中，用户 i 首次在第 $j(j\in\{1,2,\cdots,a\})$ 轮发生了本地更新回溯，即节省了第 j 轮的隐私预算. 那么在 $j+1$ 轮时，用户 i 手中可用的隐私预算为

$$\varepsilon' = \varepsilon - \frac{\varepsilon(j-1)}{r}, \tag{14}$$

则用户 i 在 $j+1$ 轮参与训练的情况有 2 种，即发生回溯或者不发生回溯. 隐私预算为

$$\varepsilon'' = \begin{cases} \varepsilon' - \dfrac{\varepsilon'}{r-j}, & \text{不发生回溯,} \\ \varepsilon', & \text{发生回溯,} \end{cases} \tag{15}$$

其中 ε'' 为第 $j+1$ 轮执行后用户 i 所剩的隐私预算.

不失一般性，如果按照式(14)(15)计算第 $a+1$ 轮用户 i 可用的隐私预算，我们同样可以表示为 ε''. 此时，用户 i 在剩余的 $r-a$ 轮迭代中隐私预算分配情况为：

①当 $r-a\leqslant T-t+1$ 时，算法整体可训练的轮数覆盖了用户 i 的可参与轮数，此时该用户当前的消耗隐私预算为 $\varepsilon''/r-a$；

②当 $r>T-t+1$ 时，用户 i 可参与训练的轮数大于算法整体剩余的训练轮数，此时该用户最多可参与的轮数为 $T-t+1$，当前消耗的隐私预算为 $\varepsilon''/T-t+1$.

为了说明 2.1 节与 2.2 节中回溯策略和隐私预算分配策略，本文给出例子进行解释. 假设 $T=8$, $t=4$, $r=4$. 如果用户 i 从当前轮 $t=4$ 首次参加全局训练如图 3(a)所示，此种情况下 $a=0$, $j=0$. 给定隐私预算 ε，由于 $r\leqslant T-t+1$(即 $r=4\leqslant T-t+1=5$)，则在当前轮 $t=4$ 时，该用户不发生回溯，所分配的隐私预算为 $\varepsilon/4$；如果用户 i 在当前轮 $t=4$ 时并非首次参与训练，此种情况下 $a\neq 0$, $j\neq 0$. 如图 3(b)所示，假设用户 i 在 $t=1$ 首次参与训练. $a=3$, $j=3$. 该用户在 $j=3$ 时发生回溯，即在该轮节省 $\varepsilon/4$. 则在 $j+1=4$ 即 $t=4$ 时，用户 i 手中的隐私预算为 $2\varepsilon/4$. 该用户不发生回溯，全部消耗掉 $2\varepsilon/4$，剩余的隐私预算为 0.

2.3 裁剪阈值的动态衰减

根据 ULDP-FED 算法中 *UserUpdate* 函数的行⑰~⑲可知，设置过大的裁剪阈值 C 会较好地保护本地更新，并且裁剪操作产生的更新偏差较小. 然而，过大的 C 值导致过大的高斯噪音，高斯机制产生的误差会很大，进而造成最终聚集的全局模型可用性较低；设置过小的 C 值尽管产生较小的高斯噪音，且由高斯机制产生的误差比较小，但是会对用户的本地更新过度裁剪，进而导致本地更新信息损失严重，即由裁剪操作产生的更新偏差会很大. 同样也会对最终全局模型的可用性造成严重影响. 因此，每个用户如何选择合适的裁剪阈值来有效均衡高斯机制产生的误差与裁剪操作产生的偏差是一个大的挑战.

图 4 描述了 MNIST 数据集上的本地更新与裁剪阈值变化情况. 可以看出，本地更新在训练初期包含的信息更多，随着迭代轮数的增加本地更新趋于平滑. 因此，裁剪阈值的设定应该能够符合本地更新的变化趋势. 基于此，本文提出一种基于动态衰减机制的裁剪阈值设置方法，如图 4 中 $C=\mathrm{e}^{-\beta k}C$ 所示，其中 $k\in\{1, 2, \cdots, r\}$, β 表示衰减因子. 为了阐述动态衰减阈值设置的合理性，本文给出定理 2 进行解释.

定理 2. 给定参数 r 与 k，且 $k\in\{1, 2, \cdots, r\}$. 在用户 i 参与的 r 轮迭代中，每一轮均可获得合适的裁剪阈值 $C=\left\|\Delta_i^t\right\|/\mathrm{e}^{-\beta k}+\dfrac{2\sigma^2}{|U^t|}d$，使得高斯噪音误差与裁剪产生的偏差均衡. 其中 d 表示本地更新的维度，U^t 表

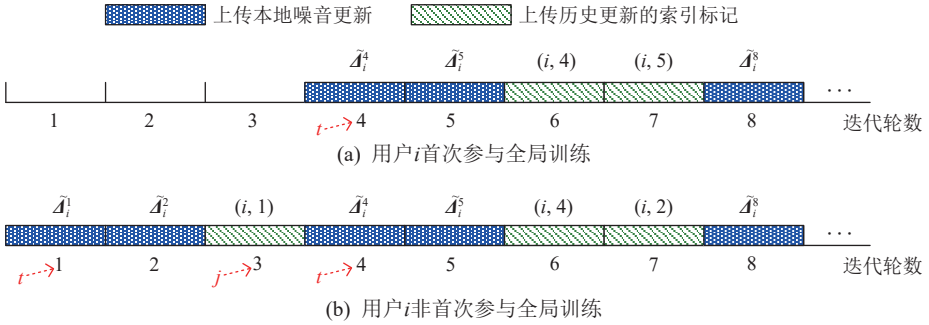


Fig. 3 An illustration of local update recall

图 3 本地更新回溯示意图

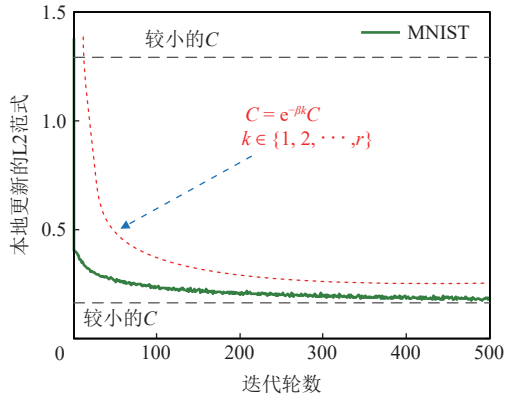


Fig. 4 Selection of clipping threshold of local update

图 4 本地更新裁剪阈值选择

示当前参与训练的用户集合, σ^2 表示高斯机制的方差.

证明. 设 Δ_i^t 为用户 i 未裁剪未加噪的本地更新, $\overline{\Delta}_i^t$ 表示该用户已裁剪未加噪的本地更新, $\tilde{\Delta}_i^t$ 表示该用户已裁剪已加噪的本地更新. 根据均方误差度量标准可知如下式(16)成立.

$$
\begin{aligned}
&E\left[\frac{1}{d}\left\|\tilde{\Delta}_i^t-\Delta_i^t\right\|_2^2\right]=\frac{1}{d}E\left[\left\|\overline{\Delta}_i^t-\Delta_i^t\right\|_2^2\right]+\frac{1}{d}E\left[\left\|\tilde{\Delta}_i^t-\overline{\Delta}_i^t\right\|_2^2\right]=\\
&\quad\frac{1}{d}E\left[\left\|\Delta_i^t\min\left(1,\frac{Ce^{-\beta k}}{\left\|\Delta_i^t\right\|}\right)-\Delta_i^t\right\|_2^2\right]+\\
&\quad\frac{1}{d}E\left[\left\|\mathcal{N}\left(0,\frac{\sigma^2(Ce^{-\beta k})^2}{|U^t|}I_d\right)\right\|_2^2\right]=\\
&\quad\frac{1}{d}E\left[\left\|\frac{\Delta_i^t}{\left\|\Delta_i^t\right\|}\min\left(0,Ce^{-\beta k}-\left\|\Delta_i^t\right\|\right)\right\|_2^2\right]+\\
&\quad\frac{1}{d}dE\left[\mathcal{N}\left(0,\frac{\sigma^2(Ce^{-\beta k})^2}{|U^t|}\right)^2\right]=\\
&\quad\frac{1}{d}\min\left(0,Ce^{-\beta k}-\left\|\Delta_i^t\right\|\right)^2+E\left[\mathcal{N}\left(0,\frac{\sigma^2(Ce^{-\beta k})^2}{|U^t|}\right)^2\right]=\\
&\quad\frac{1}{d}\max\left(0,\left\|\Delta_i^t\right\|-Ce^{-\beta k}\right)^2+\frac{\sigma^2(Ce^{-\beta k})^2}{|U^t|}.
\end{aligned}
\tag{16}
$$

由式(16)可知, $\left\|\Delta_i^t\right\|$, σ, $|U^t|$, d 为常数, 可以优化的参数为 C 与 β. 为了计算方便, 把式(16)表示成式(17).

$$
\begin{aligned}
f(C)=&E\left[\frac{1}{d}\left\|\tilde{\Delta}_i^t-\Delta_i^t\right\|_2^2\right]=\\
&\frac{1}{d}\max\left(0,\left\|\Delta_i^t\right\|-Ce^{-\beta k}\right)^2+\frac{\sigma^2(Ce^{-\beta k})^2}{|U^t|}.
\end{aligned}
\tag{17}
$$

式(17)两侧同时对 C 进行求导, 可得

$$
f'(C)=\begin{cases}\dfrac{2}{d}\left(Ce^{-\beta k}-\left\|\Delta_i^t\right\|\right)e^{-\beta k}+\dfrac{2\sigma^2e^{-\beta k}}{|U^t|}C, & C\leqslant\dfrac{\left\|\Delta_i^t\right\|}{e^{-\beta k}},\\ \dfrac{2\sigma^2e^{-\beta k}}{|U^t|}C, & C>\dfrac{\left\|\Delta_i^t\right\|}{e^{-\beta k}},\end{cases}
$$

当 $f'(C)=0$ 时, 函数 $f(C)$ 在 $C=\dfrac{\left\|\Delta_i^t\right\|}{e^{-\beta k}+\dfrac{2\sigma^2}{|U^t|}d}$ 取得最小值.

因此, 用户 i 在参与的 $\{1,2,\cdots,r\}$ 轮次中, 每轮均可寻找到合适的裁剪阈值 C 来均衡高斯噪音误差与裁剪产生的偏差. 证毕.

2.4 ULDP-FED 算法的隐私性度量

由于 ULDP-FED 算法通过反复迭代求解全局模型, 每个参与用户需要合理地分配隐私预算才能确保每轮满足差分隐私, 进而保证每个用户在 r 轮后满足 (ε,δ)-ULDP. 本文通过差分隐私的基本组合定理、强组合定理以及矩审计来阐述 ULDP-FED 算法的隐私性. ULDP-FED 算法中只有 *UserUpdate* 函数消耗隐私预算, 只要证明其满足 (ε,δ)-ULDP, 则 ULDP-FED 算法同样满足. 为了阐述方便, 我们给出 *UserUpdate* 函数的隐私损失定义, 如式(18)所示:

$$
c_M(o;M,aux,D,D')=log\frac{Pr[M(aux,D)=o]}{Pr[M(aux,D')=o]},
\tag{18}
$$

其中 M 表示 *UserUpdate* 函数执行 r 轮的整体算法, 即 $M=(M_1,M_2,\cdots,M_r)$, D 表示用户拥有的训练数据, D' 表示 D 的近邻数据集, aux 表示辅助输入, o 表示 M 的输出结果.

若式(18)$\leqslant\varepsilon$, 即可说明 ULDP-FED 算法满足 (ε,δ)-ULDP. 我们重写式(18)后获得式(19).

$$
\begin{aligned}
c_M(o;M,aux,D,D')=&\log\frac{Pr[M(aux,D)=o]}{Pr[M(aux,D')=o]}=\\
&\log\frac{Pr[M_1(aux,D)=o,\cdots,M_r(aux,D)=o]}{Pr[M_1(aux,D')=o,\cdots,M_r(aux,D')=o]}=\\
&\log\frac{Pr[M_1(aux,D)=o]\times\cdots\times Pr[M_r(aux,D)=o]}{Pr[M_1(aux,D')=o]\times\cdots\times Pr[M_r(aux,D')=o]}.
\end{aligned}
\tag{19}
$$

结合式(19), 设置第 k 轮的隐私损失, 如式(20)所示:

$$
log\frac{Pr[M_k(aux,D)=o]}{Pr[M_k(aux,D')=o]}=c\left(o;M_k,o_{1:(k-1)},D,D',g_k\right),
\tag{20}
$$

其中 g_k 表示第 k 轮是否发生回溯.

结合式(19)(20), 可知式(21)成立.

$$
c_M(o;M,aux,D,D')\leqslant\sum_{k=1}^{r}c\left(o;M_k,o_{1:(k-1)},D,D',g_k\right).
\tag{21}
$$

1)基于基本组合定理的隐私性度量. 结合式(21), 对 $c(o;M_k,o_{1:(k-1)},D,D',g_k)$ 进行定量分析. 对于参与整体训练的任意用户, 该用户有效参与轮次 $1,2,\cdots,r$ 中任意一个轮次 j, 其隐私损失为 $c(o;M_j,o_{1:(j-1)},D,D',g_j)$. 当 $g_j=1$ 时, 说明第 j 轮发生回溯, 则此时第 j 轮的隐私损失 $c(o;M_j,o_{1:(j-1)},D,D',g_j)=0$. 否则, 结合式(14)

(15)可获得 $c(o; M_j, o_{1:(j-1)}, D, D', g_j)=\varepsilon_j$. 根据本文的本地更新回溯规则、隐私预算分配策略，以及基本组合定理可知，$\sum_{k=1}^{r} c\left(o; M_k, o_{1:(k-1)}, D, D', g_k\right) \leqslant \varepsilon$. 进而可知 $c_M(o; M, aux, D, D') \leqslant \varepsilon$. 结合式(18)与文献[19]可知ULDP-FED算法满足 (ε, δ)-ULDP.

2)基于强组合定理的隐私性度量. 根据文献[19]可知，强组合定理通常计算隐私损失 $c(o; M_k, o_{1:(k-1)}, D, D', g_k)$ 的期望值，即 $E(c(o; M_k, o_{1:(k-1)}, D, D', g_k))$. 结合本地更新回溯规则与隐私预算分配策略可知，如果第 j 轮发生回溯，则 $E(c(o; M_j, o_{1:(j-1)}, D, D', g_j))=0$，否则，$E(c(o; M_j, o_{1:(j-1)}, D, D', g_j)) \leqslant \varepsilon_j(\mathrm{e}^{\varepsilon j}-1)$. 根据文献[19]可知，强组合定理的隐私损失边界比较松散，即ULDP-FED满足 $(O(q\varepsilon\sqrt{r\log(1/\delta)}), (rq+1)\delta)$-ULDP.

3)基于矩审计机制的隐私性度量. 不同于基本组合定理与强组合定理，文献[19]给出的矩审计机制利用矩母函数重新定义每轮隐私损失 $c(o; M_k, o_{1:(k-1)}, D, D', g_k)$. 结合马尔可夫不等式与尾界理论给出 $c(o; M_k, o_{1:(k-1)}, D, D', g_k)$ 更紧的边界. 为了使得矩审计机制符合实际的联邦学习系统，本文把该机制拓展到允许部分用户掉线的情况，即每个用户最多参与 r 轮训练. 对于有效参与训练的每个用户，我们能够精确地计算出 r 轮中每一轮的高斯噪音尺度 σ_k，如定理3所示.

定理 3. 给定采样概率 $q=|U^t|/|U|$，整体算法迭代轮次 T，每个用户最多参与的训练轮次 r. 若第 i 个用户满足 $(\varepsilon_i, \delta_i)$-ULDP，对于任意轮次 $k\in\{1, 2, \cdots, r\}$，该轮次中高斯机制的噪声尺度 σ_k 满足下列不等式：

$$\sigma_k=\begin{cases}\dfrac{2qC\mathrm{e}^{-\beta k}\sqrt{r\log(1/\delta_i)}}{\varepsilon_i}, & T-t\geqslant r,\\ \dfrac{2qC\mathrm{e}^{-\beta k}\sqrt{(T-t)\log(1/\delta_i)}}{\varepsilon_i}, & T-t<r,\end{cases}$$

其中 U^t 表示当前轮 t 中参与训练的用户集合，U 表示全部的用户集合，β 表示裁剪阈值的衰减因子.

证明. 为了证明方便，令 μ_0 表示高斯机制 $\mathcal{N}(0,\sigma_k^2)$ 的概率密度函数，μ_1 表示 $\mathcal{N}(C\mathrm{e}^{-\beta k},\sigma_k^2)$ 的概率密度函数，μ 为混合高斯分布 $q\mathcal{N}(C\mathrm{e}^{-\beta k},\sigma_k^2)+(1-q)\mathcal{N}(0,\sigma_k^2)$ 的概率密度函数. 根据文献[19]与式(18)，设置 M_k 表示 *UserUpdate* 函数执行第 k 轮的整体过程，M_k 隐私损失的 λ 阶对数矩母函数如式(22)所示：

$$\alpha_{M_k}(\lambda; aux, D, D')=\log E\left(\mathrm{e}^{(\lambda c(o; M_k, aux, D, D'))}\right). \tag{22}$$

根据文献[19]可知，式(23)成立.

$$\alpha_M(\lambda)\leqslant\sum_{k=1}^{r}\alpha_{M_k}(\lambda). \tag{23}$$

结合式(22)(23)可知，式(24)成立.

$$\alpha_{M_k}(\lambda):=log(\max\{E_0, E_1\}), \tag{24}$$

其中 $E_0=E_{z\sim\mu_0}\left[\dfrac{\mu_0(z)}{\mu(z)}\right]^{\lambda}=E_{z\sim\mu_0}\left[\dfrac{\mu(z)}{\mu_0(z)}\right]^{-\lambda}$，$E_1=E_{z\sim\mu}\left[\dfrac{\mu(z)}{\mu_0(z)}\right]^{\lambda}=E_{z\sim\mu_0}\left[\dfrac{\mu(z)}{\mu_0(z)}\right]^{\lambda+1}$.

由文献[11]可知，$E_1>E_0$ 成立. 利用二项展开式对 E_1 展开可获得式(25).

$$E_1=E_{z\sim\mu_0}\left[\left(\frac{\mu(z)}{\mu_0(z)}\right)^{\lambda+1}\right]=E_{z\sim\mu_0}\left[\left(1+\frac{\mu(z)-\mu_0(z)}{\mu_0(z)}\right)^{\lambda+1}\right]=\sum_{a=0}^{\lambda+1}\binom{\lambda+1}{a}E_{z\sim\mu_0}\left[\left(\frac{\mu(z)-\mu_0(z)}{\mu_0(z)}\right)^{a}\right]. \tag{25}$$

根据式(25)可知，E_1 二项展开式的第1项为1，第2项如式(26)所示：

$$E_{z\sim\mu_0}\left[\frac{\mu(z)-\mu_0(z)}{\mu_0(z)}\right]=\int_{-\infty}^{+\infty}\mu_0(z)\frac{\mu(z)-\mu_0(z)}{\mu_0(z)}\mathrm{d}z=0. \tag{26}$$

由于 $u(z)\geqslant(1-q)u_0(z)$，为了便于计算式(25)，我们把 E_1 展开式的第3项变换为：

$$E_{z\sim\mu_0}\left[\left(\frac{\mu(z)-\mu_0(z)}{\mu_0(z)}\right)^2\right]=E_{z\sim\mu_0}\left[\left(\frac{q\mu_1(z)+(1-q)\mu_0(z)-\mu_0(z)}{\mu_0(z)}\right)^2\right]=q^2E_{z\sim\mu_0}\left[\left(\frac{\mu_1(z)-\mu_0(z)}{\mu_0(z)}\right)^2\right]=q^2E_{z\sim\mu_0}\left[\left(\mathrm{e}^{\left(\frac{2zC\mathrm{e}^{-\beta k}-\left(C\mathrm{e}^{-\beta k}\right)^2}{2\sigma_k^2}\right)}-1\right)^2\right]=$$

$$q^2\left\{1-2E_{z\sim\mu_0}\left[\mathrm{e}^{\frac{2zC\mathrm{e}^{-\beta k}-\left(C\mathrm{e}^{-\beta k}\right)^2}{2\sigma_k^2}}\right]+E_{z\sim\mu_0}\left[\mathrm{e}^{\frac{4zC\mathrm{e}^{-\beta k}-2\left(C\mathrm{e}^{-\beta k}\right)^2}{2\sigma_k^2}}\right]\right\}=q^2\left[1-2\mathrm{e}^{\frac{\left(C\mathrm{e}^{-\beta k}\right)^2}{2\sigma_k^2}}\times\mathrm{e}^{\frac{-\left(C\mathrm{e}^{-\beta k}\right)^2}{2\sigma_k^2}}+\mathrm{e}^{\frac{4\left(C\mathrm{e}^{-\beta k}\right)^2}{2\sigma_k^2}}\times\mathrm{e}^{\frac{-2\left(C\mathrm{e}^{-\beta k}\right)^2}{2\sigma_k^2}}\right]=q^2\left[\mathrm{e}^{\frac{\left(C\mathrm{e}^{-\beta k}\right)^2}{\sigma_k^2}}-1\right].$$

因此 E_1 展开式的第3项如式(27)所示：

$$\binom{1+\lambda}{2}E_{z\sim\mu_0}\left[\left(\frac{\mu(z)-\mu_0(z)}{\mu_0(z)}\right)^2\right]=\frac{\lambda(1+\lambda)q^2}{2}\left[\mathrm{e}^{\frac{\left(C\mathrm{e}^{-\beta k}\right)^2}{\sigma_k^2}}-1\right]\leqslant\frac{\lambda(1+\lambda)q^2\left(C\mathrm{e}^{-\beta k}\right)^2}{\sigma_k^2}. \tag{27}$$

结合式(27)(24)可得式(28)成立.

$$\alpha_{M_k}(\lambda)\leqslant\frac{\lambda(1+\lambda)q^2\left(C\mathrm{e}^{-\beta k}\right)^2}{\sigma_k^2}+O\left(\frac{\lambda^3q^3\left(C\mathrm{e}^{-\beta k}\right)^3}{\sigma_k^3}\right)\approx\frac{\lambda(1+\lambda)q^2\left(C\mathrm{e}^{-\beta k}\right)^2}{\sigma_k^2}. \tag{28}$$

结合式(23)，可知 r 轮总的隐私损失如式(29)

所示:

$$\alpha_M(\lambda)\leqslant\sum_{k=1}^{r}\alpha_{M_k}(\lambda)=\frac{r\lambda(1+\lambda)q^2\left(C\mathrm{e}^{-\beta k}\right)^2}{\sigma_k^2}. \tag{29}$$

根据尾界理论可得: $\delta_i\geqslant\mathrm{e}^{\alpha_M(\lambda)-\lambda\varepsilon_i}$成立. 当$\delta_i=\min\limits_{\lambda}\mathrm{e}^{\alpha_M(\lambda)-\lambda\varepsilon_i}$时, 算法 M 获得隐私损失最紧的上界.

然后把式(29)带入$\delta_i=\min\limits_{\lambda}\mathrm{e}^{\alpha_M(\lambda)-\lambda\varepsilon_i}$, 可知式(30)成立.

$$\min_{\lambda}\mathrm{e}^{\alpha_M(\lambda)-\lambda\varepsilon_i}\leqslant\min_{\lambda}\mathrm{e}^{\frac{r\lambda(1+\lambda)q^2\left(C\mathrm{e}^{-\beta k}\right)^2}{\sigma_k^2}-\lambda\varepsilon_i}. \tag{30}$$

当$\lambda=\dfrac{\varepsilon_i\sigma_k^2}{2rq^2(C\mathrm{e}^{-\beta k})^2}-\dfrac{1}{2}$时, 不等式(31)成立.

$$\delta_i\geqslant-\frac{\varepsilon_i^2\sigma_k^2}{4rq^2(C\mathrm{e}^{-\beta k})^2}+\frac{\varepsilon_i}{2}-\frac{rq^2\left(C\mathrm{e}^{-\beta k}\right)^2}{4\sigma_k^2}. \tag{31}$$

对不等式(31)两边同时取倒数后再取对数, 可获得不等式(32).

$$\log\left(\frac{1}{\delta_i}\right)\leqslant\frac{\varepsilon_i^2\sigma_k^2}{4rq^2(C\mathrm{e}^{-\beta k})^2}-\frac{\varepsilon_i}{2}+\frac{rq^2\left(C\mathrm{e}^{-\beta k}\right)^2}{4\sigma_k^2}. \tag{32}$$

由于 $\delta_i\in(0,1]$, 则不等式(33)成立.

$$\frac{r\lambda(1+\lambda)q^2\left(C\mathrm{e}^{-\beta k}\right)^2}{\sigma_k^2}-\lambda\varepsilon_i<0. \tag{33}$$

由于 $\lambda>0$, 则可以把不等式(33)放缩成$\dfrac{rq^2\left(C\mathrm{e}^{-\beta k}\right)^2}{\sigma_k^2}<\varepsilon_i$. 然后, 把$\dfrac{rq^2\left(C\mathrm{e}^{-\beta k}\right)^2}{\sigma_k^2}<\varepsilon_i$带入式(32), 进而获得不等式(34).

$$\log\left(\frac{1}{\delta_i}\right)<-\frac{\varepsilon_i}{4}+\frac{\varepsilon_i^2\sigma_k^2}{4rq^2(C\mathrm{e}^{-\beta k})^2}<\frac{\varepsilon_i^2\sigma_k^2}{4rq^2(C\mathrm{e}^{-\beta k})^2}. \tag{34}$$

结合式(34)可获得不等式(35)成立.

$$\sigma_k\geqslant\frac{2qC\mathrm{e}^{-\beta k}\sqrt{r\log(1/\delta_i)}}{\varepsilon_i}. \tag{35}$$

由于参数 r 是每个用户参与的最多轮数, 则 r 与当前轮 t 以及全局总轮数 T 存在如下关系:

当 $T-t<r$ 时, 说明用户 i 最多参与的轮数为 $T-t$. 此时, $\sigma_k=\dfrac{2qC\mathrm{e}^{-\beta k}\sqrt{(T-t)\log(1/\delta_i)}}{\varepsilon_i}$.

当 $T-t\geqslant r$ 时, 说明用户 i 在剩余的轮数 $T-t$ 中可以完成 r 轮的迭代. 此时, $\sigma_k=\dfrac{2qC\mathrm{e}^{-\beta k}\sqrt{r\log(1/\delta_i)}}{\varepsilon_i}$.

证毕.

2.5 ULDP-FED 算法的收敛性

联邦学习算法的收敛性是衡量该类算法的主要标准之一. 为了说明 ULDP-FED 算法的收敛性, 基于 1.2 节联邦优化问题, 对优化函数给出相应假设: 1)$F_i(\boldsymbol{W})$ 为强凸函数; 2)$F_i(\boldsymbol{W})$ 满足 L-Lipschitz 光滑, 对任意的 $\boldsymbol{W}$, $\boldsymbol{W}'$存在 $\|\nabla F_i(\boldsymbol{W})-\nabla F_i(\boldsymbol{W}')\|\leqslant L\|\boldsymbol{W}-\boldsymbol{W}'\|$, 其中 L 是大于 0 的常数; 3)用户中的随机参数都有界 $E[\|\nabla F_i(\boldsymbol{W},s)-\nabla F_i(\boldsymbol{W})\|^2]\leqslant\gamma_l^2$, 其中 $s\in D_i$; 4)对于对全局参数和局部参数满足不等式$\dfrac{1}{n}\sum\limits_{i=1}^{n}\|\nabla F_i(\boldsymbol{W})-\nabla F(\boldsymbol{W})\|^2\leqslant\gamma_g^2$; 5)用户本地损失函数的梯度对于任意的 $s\in D_i$ 满足 $\|\nabla F_i(\boldsymbol{W},s)\|\leqslant G$. 根据上述 5 种假设, 本文给出定理 4.

证明. **定理 4.** 在假设 1)~5)的条件下, 当$\eta_l\leqslant\dfrac{1}{\sqrt{60}\Gamma L}$时, 算法 ULDP-FED 满足下列不等式:

$$\frac{1}{T}\sum_{t=1}^{T}E\left[\bar{\alpha}^t\left\|\nabla F(\boldsymbol{W}^t)\right\|^2\right]\leqslant O\left(\frac{1}{\eta_g\eta_l\Gamma T}+\eta_l^2\Gamma^2+\frac{\eta_g\eta_l}{|U^t|}+\frac{\eta_g\sigma^2\left(C\mathrm{e}^{-\beta k}\right)^2 d}{\eta_l\Gamma|U^t|^2}\right),$$

其中Γ表示本地完整的训练次数, η_l表示本地学习率, η_g表示全局学习率.

$$\bar{\alpha}^t=\frac{1}{n}\sum_{i=1}^{n}\alpha_i^t,\ \alpha_i^t=\frac{C\mathrm{e}^{-\beta k}}{\max\left(C\mathrm{e}^{-\beta k},\eta_l\left\|\sum\limits_{e=1}^{\Gamma}g_i^{t,e}\right\|\right)}.$$

根据文献 [18] 与文献 [23] 可知, 定理 4 成立.

证毕.

3 实验结果与分析

3.1 实验平台与数据集

本文实验平台是 4 核 Intel CPU(4 GHz), 16 GB 内存, Win7 系统, 代码采用 Python 语言实现, 深度学习框架为 PyTorch. 实验采用了 2 种数据集, 分别是 MNIST 和 CIFAR 10. 这 2 种数据集均是从 PyTorch 框架下载, 其中 MNIST 为手写数字数据集, 共 70 000 张灰度图像, 大小为 28×28, 类别为 10 分类 0~9; CIFAR 10 数据集是大小为 32×32 的彩色图片, 共 60 000 张, 同样拥有 10 个类别, 包括飞机、汽车、鸟类等. 2 种数据集具体细节如表 1 所示.

为了模拟真实环境, 我们将 MNIST 数据集非独立同分布地平均分配给 1 000 个用户. 根据数据集的标签对数据集做了排序处理, 并将其分成了 200 份, 每份拥有 300 张图片, 每个用户从其中任意选择 2 份, 即每个用户拥有 600 条数据集并且只拥有 0~9 中的

Table 1 Characteristics of Datasets

表 1 数据集的属性

数据集	大小	通道数	训练集张数	测试集张数
MNIST	28×28	1	60 000	10 000
CIFAR 10	32×32	3	50 000	10 000

2个数字. 对于CIFAR 10数据集, 将50 000张数据图片随机均匀分配给1 000个用户, 平均每个用户500张数据图片. 这种随机均匀分配方式符合独立同分布.

3.2 模型结构与评价方法

本文采用的机器学习算法是卷积神经网络(convolution neural network, CNN), 由于MNIST和CIFAR 10数据集的通道数不同, 分别设计了不同的网络结构. 对于MNIST数据集, 网络模型由2个卷积层(卷积核大小是5×5, 第1层通道是10, 第2层通道是20)、2个池化层(池化窗口大小为2×2)、2个全连接层(神经元个数为50, 激活函数为ReLu)以及1个softmax输出层组成. CIFAR 10数据集的模型结构则是由2个卷积层(卷积核大小是5×5, 第1层通道是6, 第2层通道是16, 激活函数为ReLu)、2个池化层(池化窗口大小为2×2)、3个全连接层(神经元个数为120, 84, 激活函数为ReLu)以及1个softmax输出层组成. 由于MNIST和CIFAR 10数据集都是10分类问题, 本文采用测试准确率(test accuracy)作为评价指标. 对于通信代价, 则采用相同准确率下, 对算法所需的迭代轮数进行对比.

3.3 对比算法介绍

本文在实验阶段将ULDP-FED算法拆分为基于符号比较的DSDP-FedAvg算法和基于余弦比较的DCDP-FedAvg算法, 并且引入了相似性比较、本地更新回溯以及动态衰减策略. 为了探索算法准确率的提高是源自于回溯策略还是动态衰减策略或是两者的共同作用, 本文又设计了DDP-FedAvg, SDP-FedAvg, CDP-FedAvg算法用于消融实验. 各算法的具体细节如表2所示.

3.4 基于序列组合性算法对比

1)基于MNIST数据集的算法比较

图5描述了DSDP-FedAvg, DCDP-FedAvg, DDP-FedAvg这3种算法的实验结果, 实验参数设置为: 用户采样率q=0.1, 初始裁剪阈值C=2, 算法迭代总轮数T=300, 用户最多参与轮数r=50, ε=10, 衰减因子$\beta\in\{0.02, 0.04, 0.06, 0.08, 0.10\}$. 图5(a)描述了DDP-FedAvg在MNIST数据集上的测试准确率的变化情况. 当β=0.06时, 该算法的测试准确率达到0.941 2. 其原因是当β过大时, 对用户本地更新的裁剪过大, 导致本地更新的原始信息损失过多; 而β过小时, 会导致本地更新所加入的噪声较大. 图5(b)描述了当ε=8, r改变时, DDP-FedAvg测试准确率变化情况. 当r=50, 60, 70时, 该算法的测试准确率逐渐降低. 其原因是用户参与的轮数越多, 每一轮分到的ε越小, 导致添加的高斯噪音越多. 图5(c)描述了DDP-FedAvg在r=60, ε改变情况下的测试准确率变化情况. 从图5(c)可发现, 随着ε增大, 该算法的测试准确率逐渐增加.

图5(d)~(f)与图5(g)~(i)分别描述了DSDP-FedAvg, DCDP-FedAvg在r, ε, τ变化情况下的实验结果. 图5(d)(g)描述了r=50, ε=10, β=0.06时, 回溯阈值τ如何影响这2种算法的测试准确率. 从图5(d)(g)可知, τ分别取值0.25, 0.006时, 这2种算法的测试准确率最高, 其原因是回溯操作节省了部分隐私预算. 图5(e)(h)(f)(i)描述了固定β=0.06, r与ε分别变化时, DSDP-FedAvg与DCDP-FedAvg测试准确率的变化情况. 结果显示, 固定ε时, r值越大, 算法的测试准确率越低; 固定r时, ε值越大, 算法的测试

Table 2 Introduction of Comparative Algorithms

表2 对比算法介绍

算法	算法介绍
FedAvg[24]	原始的联邦平均算法，不添加噪声
DP-FedAvg[10]	在FedAvg算法基础上增加了差分隐私保护
NbAFL[13]	在FedAvg算法基础上增加了本地与全局加噪方法
DP-FedAvg-blur[18]	在DP-FedAvg算法基础上增加了稀疏化操作
DP-FedAvg-blurs[18]	在DP-FedAvg算法基础上增加了正则化与稀疏化操作
DP-FedSAM[16-17]	在DP-FedAvg算法基础上引入SAM优化器
DDP-FedAvg	在DP-FedAvg算法基础上引入了动态衰减策略
SDP-FedAvg	在DP-FedAvg算法基础上引入了回溯策略，相似计算方式为符号
CDP-FedAvg	在DP-FedAvg算法基础上引入了回溯策略，相似计算方式为余弦函数
DSDP-FedAvg	在DP-FedAvg算法基础上引入了动态衰减和回溯策略，相似性计算方式为符号相似
DCDP-FedAvg	在DP-FedAvg算法的基础上引入了动态衰减和回溯策略，相似性计算方式为余弦函数

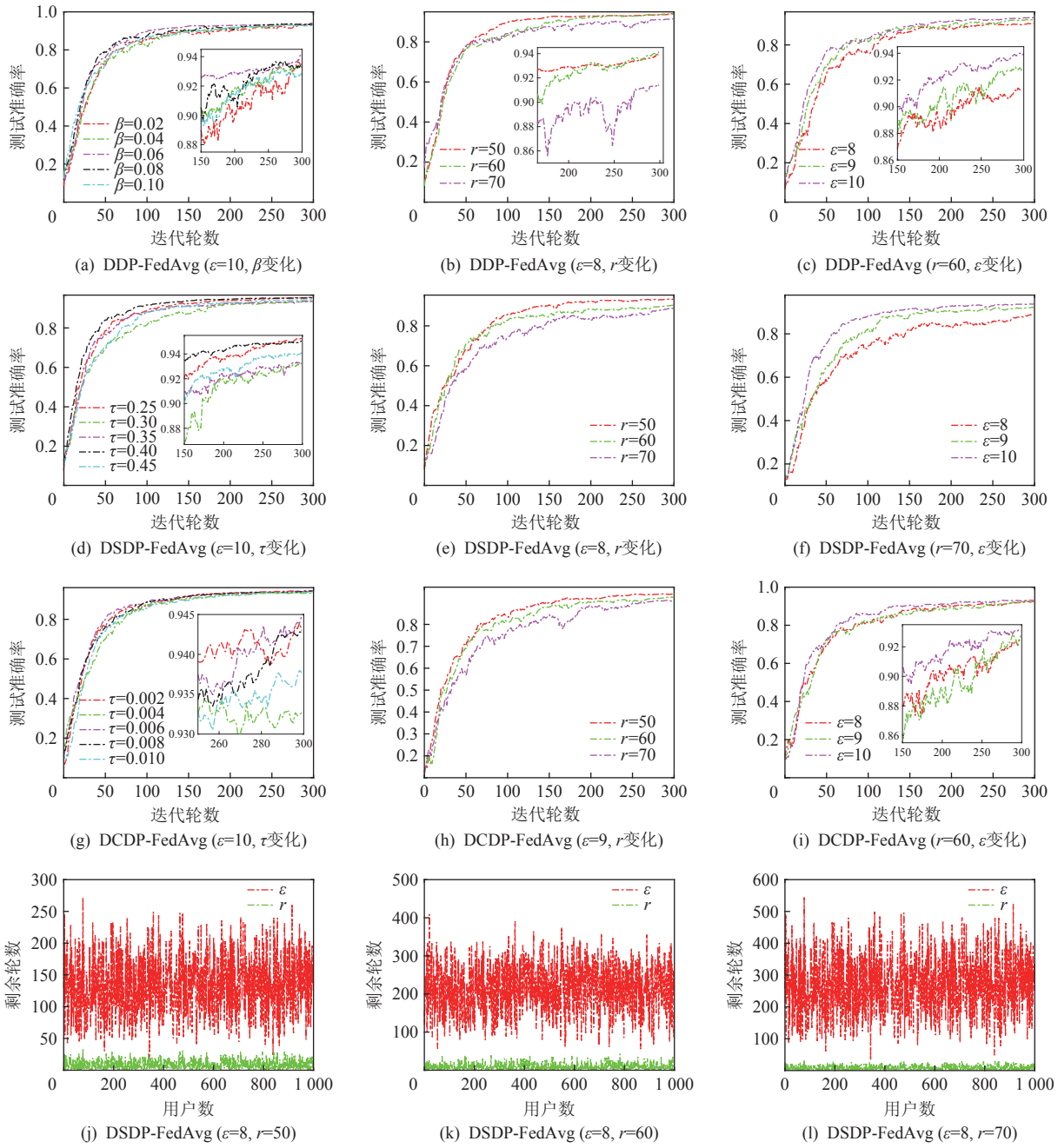


Fig. 5 Algorithms comparison based on MNIST dataset

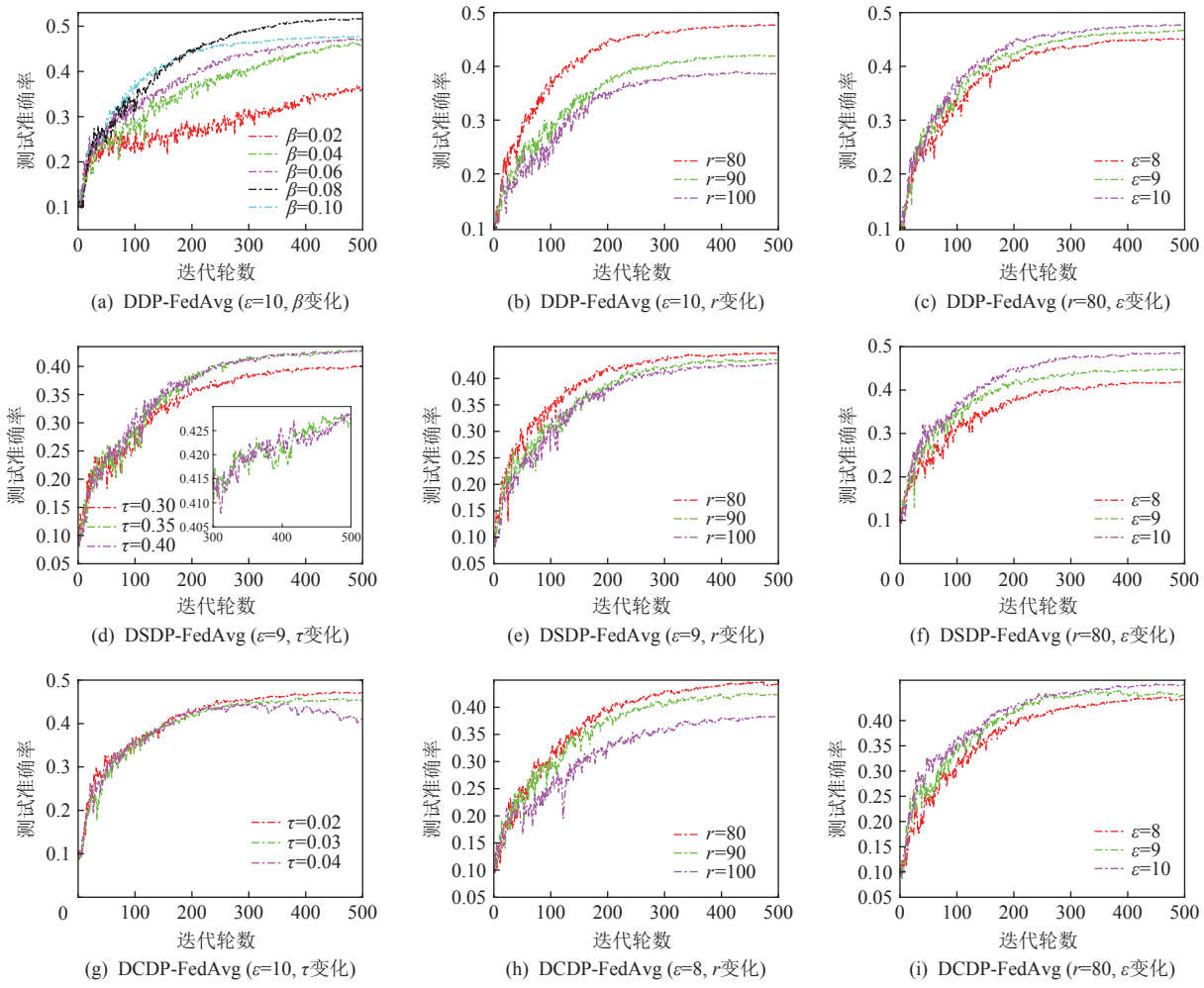
图 5 基于 MNIST 数据集的算法对比

准确率越高. 其原因是 ε 值越大, 每轮分到的预算越大.

图 5(j)~(l)描述了 q=0.1, ε=8, r=50, 60, 70 时, DSDP-FedAvg 训练结束后, 1 000 个用户所剩余的隐私预算 ε 和参与轮数 r 的分布情况. 假设用户按照顺序参与训练, 用户最终剩余的轮数为 20, 30, 40. 例如, 在图 5(j)中, 当 r=50 时, 用户的剩余轮数在 20 左右, 这是由于 MNIST 数据集具有 Non-IID 属性, 用户数据的差异会导致回溯发生.

2)基于 CIFAR 10 数据集的算法比较

图 6 描述了 DDP-FedAvg, DSDP-FedAvg, DCDP-FedAvg 这 3 种算法的实验结果, 实验参数为: 用户采样率 q=0.1, 初始裁剪阈值 C=2, 算法迭代总轮数 T=500. 图 6(a)描述了 r=80, ε=10, $\beta \in$ 0.02, 0.04, 0.06, 0.08, 0.1时, DDP-FedAvg 的测试准确率变化情况. 当 β=0.08 时, 该算法的测试准确率最高, 达到 0.512 9. 图 6(b)(c)描述了分别固定 ε 和 r 时, DDP-FedAvg 的测试准确率表现. 当 ε=10 时, 该算法的测试准确率随着 r 的增加逐渐降低. 当 r 分别取值 80, 90, 100 时, 测试准确率为 0.476, 0.420 7, 0.387 3. 图 6(d)~(f)和图 6(g)~(i)分别描述了 DSDP-FedAvg 和 DCDP-



(a) DDP-FedAvg (ε=10, β变化) (b) DDP-FedAvg (ε=10, r变化) (c) DDP-FedAvg (r=80, ε变化)

(d) DSDP-FedAvg (ε=9, τ变化) (e) DSDP-FedAvg (ε=9, r变化) (f) DSDP-FedAvg (r=80, ε变化)

(g) DCDP-FedAvg (ε=10, τ变化) (h) DCDP-FedAvg (ε=8, r变化) (i) DCDP-FedAvg (r=80, ε变化)

Fig. 6 Algorithms comparison based on CIFAR 10 dataset

图 6 基于 CIFAR 10 数据集的算法对比

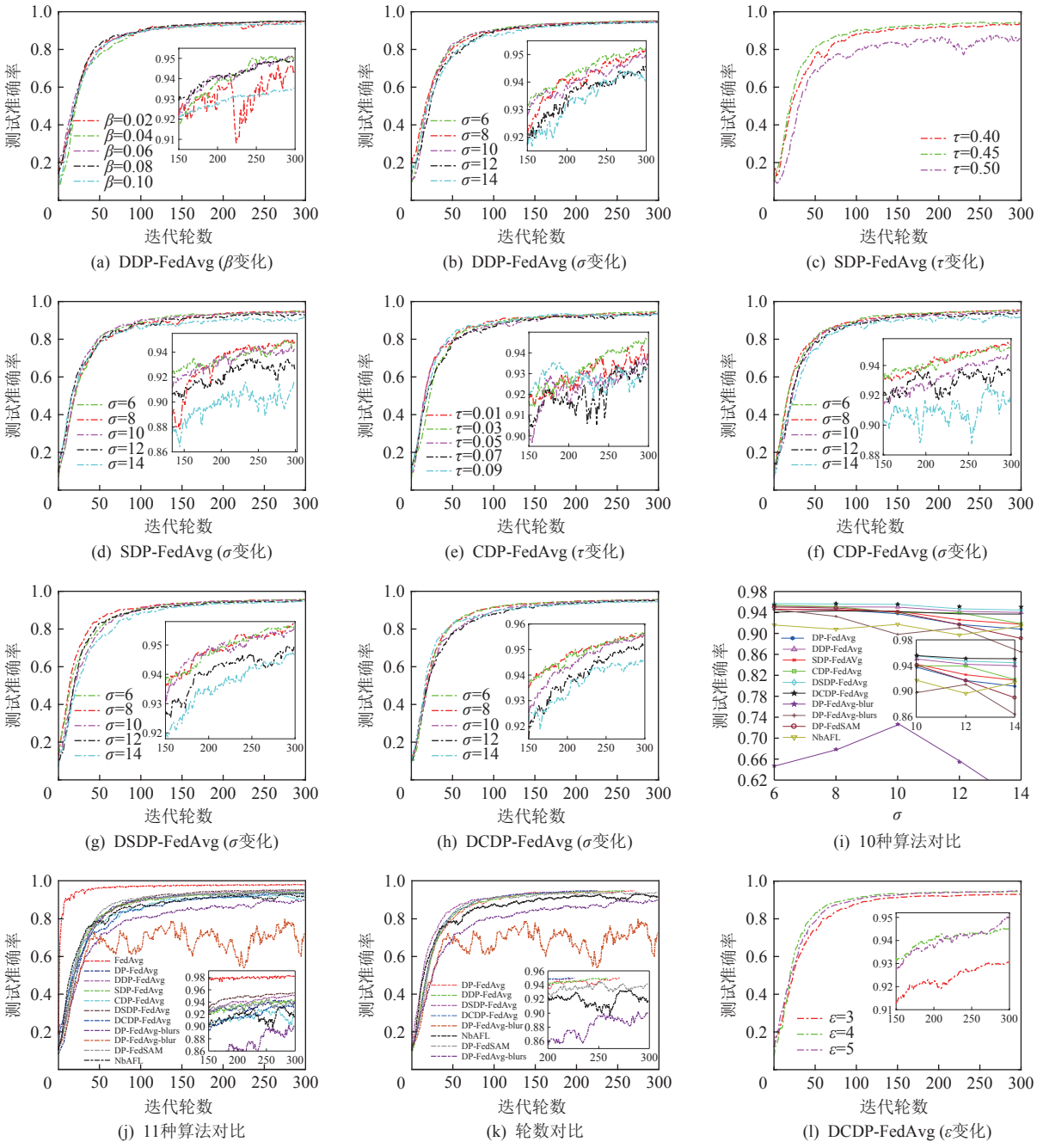
FedAvg 在 r, ε, τ 变化情况下的实验结果. 当 r=100, ε=9, β=0.08 时, DSDP-FedAvg 的最优回溯阈值 τ=0.4; 当 r=90, ε=10, β=0.08 时, 该算法的最优回溯阈值 τ=0.02. 从实验结果中可以发现, 在 τ 值的探索过程中, τ 值的改变对算法准确率影响不是很显著, 这是因为 CIFAR 10 数据集满足 IID 属性; 而图 5 中的 MNIST 数据集具有 Non-IID 属性, τ 值的改变对算法准确率的影响比较明显.

3.5 基于矩审计策略的算法对比

1)基于 MNIST 数据集的算法比较

图 7 描述了表 2 中 11 种算法的实验结果. 图 7(a)描述了 q=0.1, r=50, 噪音参数 σ=10, β=0.02, 0.04, 0.06, 0.08, 0.1 时, DDP-FedAvg 的测试准确率变化情况. 当 β=0.06 时, 该算法的测试准确率最高, 其原因是 β=0.06 时高斯噪音误差与裁剪偏差最为平衡. 从图 7(b)可以发现, σ 增大会导致 DDP-FedAvg 的测试准确率降低, 其原因是 σ 越大导致高斯噪音过大. 图 7(c)~(f)描述了 SDP-FedAvg 和 CDP-FedAvg 准确率变化情况. 同样可知 σ 越大导致测试准确率越低. 图 7(c)(e)分别给出了这 2 种算法最优回溯阈值 τ=0.45 和 τ=0.03.

图 7(g)(h)描述了 β=0.06, τ=0.45, τ=0.03 时, DSDP-FedAvg 和 DCDP-FedAvg 测试准确率受 σ 变化的影响情况, σ 越大导致添加的高斯噪音越多, 进而测试准确率越低. 图 7(i)描述了 DP-FedAvg, NbAFL, DP-FedAvg-blur, DP-FedAvg-blurs, DP-FedSAM, DDP-FedAvg, SDP-FedAvg, CDP-FedAvg, DSDP-FedAvg, DCDP-FedAvg 这 10 种算法的准确率对比情况. σ 从 6 变化到 14 时, 这 10 种算法的测试准确率整体呈下降趋势. DSDP-FedAvg 和 DCDP-FedAvg 的测试准确率变化趋于平缓, 其主要原因是这 2 种算法均同时采用了裁剪阈值动态衰减和回溯策略. 尽管 NbAFL 采用中心化与本地化高斯噪音处理本地模型与聚集模型、DP-FedAvg-blur 与 DP-FedAvg-blurs 采用正则化与稀疏化处理本地模型、DP-FedSAM 采用 SAM 优



(a) DDP-FedAvg (β变化) (b) DDP-FedAvg (σ变化) (c) SDP-FedAvg (τ变化)

(d) SDP-FedAvg (σ变化) (e) CDP-FedAvg (τ变化) (f) CDP-FedAvg (σ变化)

(g) DSDP-FedAvg (σ变化) (h) DCDP-FedAvg (σ变化) (i) 10种算法对比

(j) 11种算法对比 (k) 轮数对比 (l) DCDP-FedAvg (ε变化)

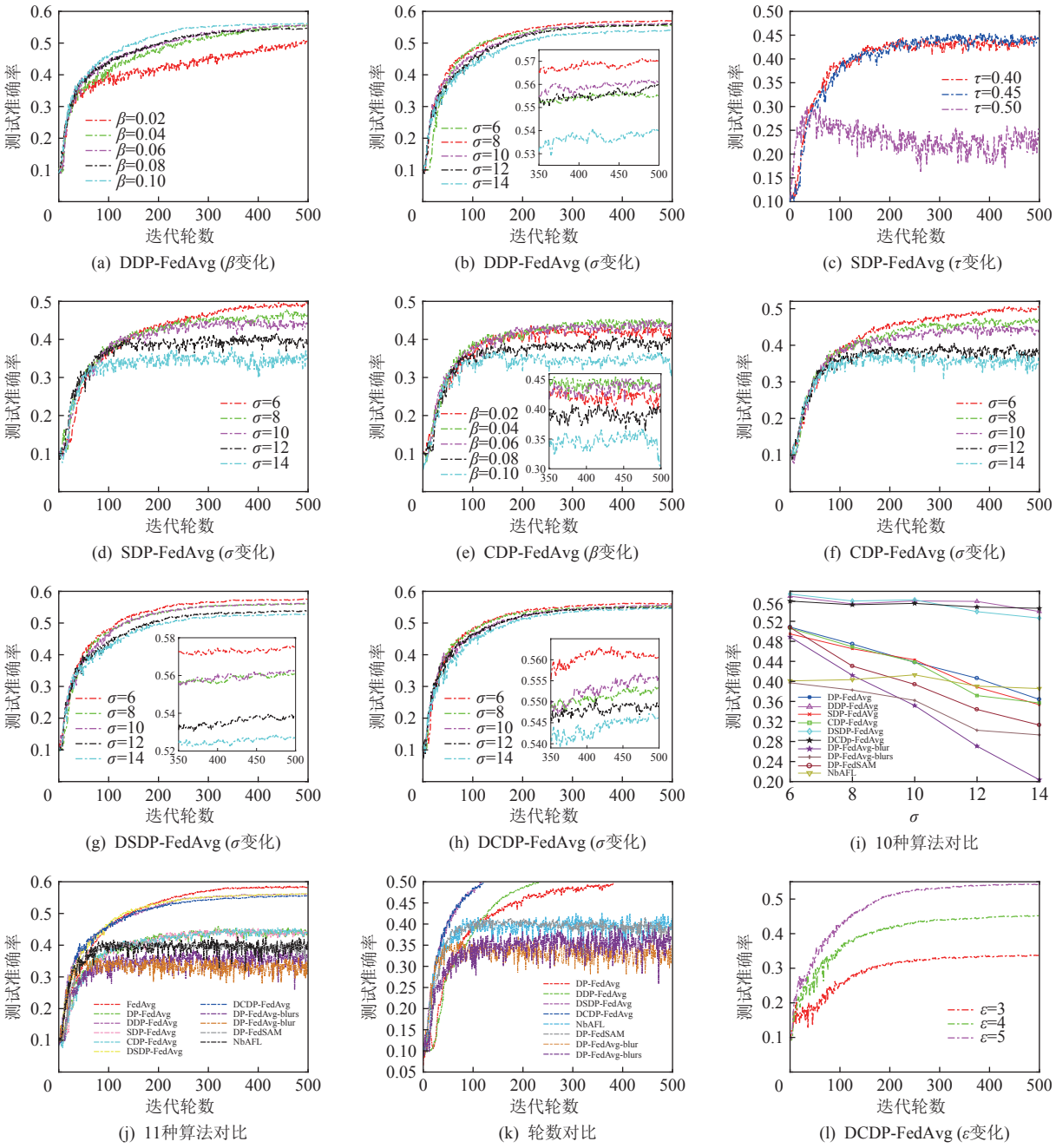
Fig. 7 Algorithms comparisons based on MNIST dataset

图 7 基于 MNIST 数据集的算法对比

化器处理本地模型, 然而这些方法均没有考虑裁剪阈值的衰减与隐私预算的节省. 图 7(j)是在上述 10 种算法的基础上增加了 FedAvg 算法. 从实验结果可知, 随着迭代轮数的增加, 模型的测试准确率增加. 非隐私 FedAvg 的测试准确率高达 0.981 6, 本文提出 5 种算法的测试准确率均在 0.92 以上. 特别是 DSDP-FedAvg 算法, 在 200 轮测试准确率达到 0.95, 且优于其他 9 种算法. 其主要原因是裁剪阈值的动态衰减和本地梯度回溯能够有效提升整体模型精度. 图 7(k)描述了 DSDP-FedAvg, DCDP-FedAvg, DDP-FedAvg, DP-FedAvg 测试准确率首次达到 0.95 时所需要的迭代轮数, 分别为 226, 224, 259 与 271. 相比于 DP-FedAvg 的轮数, DSDP-FedAvg, DCDP-FedAvg 分别减少了 45 轮和 47 轮, 进而减少了通信代价. 而 NbAFL, DP-FedAvg-blur, DP-FedAvg-blurs, DP-FedSAM 在 300 轮内均没有达到 0.95 的测试准确率. 图 7(l)描述了不同 ε 取值下 DCDP-FedAvg 的测试准确率变化情况, 结果显示 ε 增大, 算法的测试准确率逐渐增大.

2）基于 CIFAR 10 数据集的算法比较

为了验证文中算法的健壮性，在 CIFAR 10 做了类似的实验，结果如图 8 所示. 表 2 中 11 种算法在 CIFAR 10 数据集上的变化趋势与 MNIST 数据集的类似. 然而，CIFAR 10 与 MNIST 数据集的实际分布对回溯策略的效果影响较大. 由于 CIFAR 10 是由 10 个类别彩色图片组成的数据集，本文提出 5 个算法的测试准确率在 0.50 左右，而与其对比分析的 5 个算法的测试准确率在 0.40 左右. 图 8(i)描述了本文提出的 5 种算法与 DP-FedAvg, NbAFL, DP-FedAvg-blur, DP-FedAvg-blurs, DP-FedSAM 的测试准确率对比分析. 当噪音参数 σ 从 6 变化到 14 时，DSDP-FedAvg 与 DCDP-FedAvg 的测试准确率趋于平滑，而剩余算法的精度趋于下降，其原因是 σ 引起的. 图 8(j)描述了表 2 中 11 种算法关于迭代轮数对其测试准确率的影响. 从实验结果可知，随着迭代轮数的增加，模型的测试准确率增加. FedAvg 的测试准确率达到 0.582 9，DSDP-FedAvg 与 DCDP-FedAvg 的测试准确率均在 0.50 以上，DSDP-FedAvg 算法的测试准确率在 300 轮时达到 0.552 4. 其原因是本文提出的算法采用了裁



(a) DDP-FedAvg (β变化)　(b) DDP-FedAvg (σ变化)　(c) SDP-FedAvg (τ变化)

(d) SDP-FedAvg (σ变化)　(e) CDP-FedAvg (β变化)　(f) CDP-FedAvg (σ变化)

(g) DSDP-FedAvg (σ变化)　(h) DCDP-FedAvg (σ变化)　(i) 10种算法对比

(j) 11种算法对比　(k) 轮数对比　(l) DCDP-FedAvg (ε变化)

Fig. 8　Algorithms comparison based on CIFAR 10 dataset

图 8　基于 CIFAR 10 数据集的算法对比

剪阈值动态衰减与梯度回溯策略. 图 8(k)描述了在测试准确率为 0.50 时, 算法 DP-FedAvg, DDP-FedAvg, DSDP-FedAvg 以及 DCDP-FedAvg 之间的训练轮数差异性. 结果显示当测试准确率达到 0.50 时, DSDP-FedAvg 和 DCDP-FedAvg 只需要迭代 120 轮左右, 而 DP-FedAvg 则需要迭代 380 轮左右. 而 NbAFL, DP-FedAvg-blur, DP-FedAvg-blurs, DP-FedSAM 在 500 轮内测试准确率均没有达到 0.50. 图 8(l)描述了不同 ε 取值下 DCDP-FedAvg 的测试准确率变化情况, 结果显示 ε 增大其准确率逐渐增大, 其原因是 ε 越大高斯噪音越小.

4 结束语

针对用户级别本地化差分隐私保护下联邦学习问题, 本文结合现有方法存在的不足, 提出了一种有效的联邦学习算法 ULDP-FED. 该算法通过动态衰减策略修正每一轮本地更新的裁剪阈值, 进而较好地平衡高斯噪音误差与裁剪偏差. 结合本地噪音更新之间的相似性, 利用历史本地更新替代当前轮的本地更新, 进而能够节省部分隐私预算. 从基本组合定理、强组合定理以及矩审计机制的角度证明 ULDP-FED 满足用户级别本地化差分隐私. 最后通过 2 种数据集验证了 ULDP-FED 算法的学习精度. 未来工作考虑瑞丽差分隐私保护下的联邦学习问题.

作者贡献声明: 张啸剑负责论文的算法设计、相关定理设置及论证, 并负责整篇论文的撰写; 张雷雷和张治政负责论文算法的实现与实验结果分析.

参 考 文 献

[1] Kairouz P, McMahan H B, Avent B, et al. Advances and open problems in federated learning[J]. Foundations and Trends® in Machine Learning, 2021, 14(1/2): 1−210

[2] Fang Minghong, Cao Xiaoyu, Jia Jinyuan, et al. Local model poisoning attacks to Byzantine-robust federated learning[C]//Proc of the 29th USENIX Security Symp (S&P 2020). Berkeley, CA: USENIX Association, 2020: 1623−1640

[3] Cao Di, Chang Shan, Lin Zhijian, et al. Understanding distributed poisoning attack in federated learning[C]//Proc of the 25th Int Conf on Parallel and Distributed Systems (ICPADS 2019). Piscataway, NJ: IEEE, 2019: 233−239

[4] So J, Güler B, Avestimehr A S. Byzantine-resilient secure federated learning[J]. IEEE Journal on Selected Areas in Communications, 2020, 39(7): 2168−2181

[5] Zhu Ligeng, Liu Zhijian, Han Song. Deep leakage from gradients[C]//Proc of the 33rd Neural Information Processing Systems (NIPS 2019). Cambridge, MA: MIT, 2019: 14774–14784

[6] Zhao Bao, Mopuri K R, Bilen H. iDLG: Improved deep leakage from gradients[J]. arXiv preprint, arXiv: 2001.02610, 2020

[7] Nasr M, Shokri R, Houmansadr A. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning[C]//Proc of the 40th Symp on Security and Privacy (SP 2019). Piscataway, NJ: IEEE, 2019: 739−753

[8] Liu Ruixuan, Cao Yang, Chen Hong, et al. Flame: Differentially private federated learning in the shuffle model[C]//Proc of the 35th Association for the Advance of Artificial Intelligence (AAAI 2021). Palo Alto, CA: AAAI, 2021: 8688−8696

[9] Sun Lichao, Qian Jianwei, Chen Xun, et al. Ldp-FL: Practical private aggregation in federated learning with local differential privacy[C]//Proc of the 30th Int Joint Conf on Artificial Intelligence (IJCAI 2021). San Francisco, CA: Morgan Kaufmann, 2021: 1571−1578

[10] McMahan H B, Ramage D, Talwar K, et al. Learning differentially private recurrent language models[J]. arXiv preprint, arXiv: 1710.06963, 2017

[11] Geyer R C, Klein T, Nabi M. Differentially private federated learning: A client level perspective[J]. arXiv preprint, arXiv: 1712.07557, 2017

[12] Wei Kang, Li Jun, Ding Ming, et al. User-level privacy-preserving federated learning: Analysis and performance optimization[J]. IEEE Transactions on Mobile Computing, 2021, 21(9): 3388−3401

[13] Wei Kang, Li Jun, Ding Ming, et al. Federated learning with differential privacy: Algorithms and performance analysis[J]. IEEE Transactions on Information Forensics and Security, 2020, 15: 3454−3469

[14] Andrew G, Thakkar O, McMahan B, et al. Differentially private learning with adaptive clipping[C]//Proc of the 35th Neural Information Processing Systems (NIPS 2021). Cambridge, MA: MIT, 2021: 17455−17466

[15] Wang Lun, Jia Ruoxi, Song Dawn. D2P-Fed: Differentially private federated learning with efficient communication[J]. arXiv preprint, arXiv: 2006.13039, 2020

[16] Shi Yi, Wei Kang, Li Shen, et al. Towards the flatter landscape and better generalization in federated learning under client-level differential privacy[J]. arXiv preprint, arXiv: 2305.00873, 2023

[17] Shi Yi, Liu Yingqi, Wei Kang, et al. Make landscape flatter in differentially private federated learning[J]. arXiv preprint, arXiv: 2303.11242, 2023

[18] Cheng Anda, Wang Peisong, Jian Cheng et al. Differentially private federated learning with local regularization and sparsification[C]//Proc of the 36th Computer Vision and Pattern Recognition (CVPR 2022). Los Alamitos, CA: IEEE Computer Society, 2022: 10112−10121

[19] Abadi M, Chu Andy, Goodfellow I, et al. Deep learning with differential privacy[C]//Proc of the 2016 ACM SIGSAC Computer

and Communications Security(CCS 2016). New York: ACM, 2016: 308−318

[20] Frank M. Privacy integrated queries: An extensible platform for privacy-preserving data analysis[C]//Proc of the 53rd ACM SIGMOD Int Conf on Management of Data (SIGMOD 2009). New York: ACM, 2009: 19–30

[21] Dwork C, Roth A. The algorithmic foundations of differential privacy[J]. Foundations and Trends in Theoretical Computer Science, 2014, 9(3/4): 211−407

[22] Wang Luping, Wang Wei, Li Bo. CMFL: Mitigating communication overhead for federated learning[C]//Proc of the 39th Int Conf on Distributed Computing Systems (ICDCS 2019). Piscataway, NJ: IEEE, 2019: 954−964

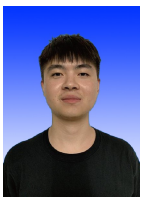
[23] Zhang Xinwei, Chen Xiangyi, Hong Mingyi, et al. Understanding clipping for federated learning: Convergence and client-level differential privacy[C]//Proc of the 19th Int Conf on Machine Learning (ICML 2022). New York: ACM, 2022: 26048−26067

[24] McMahan B, Moore E, Ramage D, et al. Communication-efficient learning of deep networks from decentralized data[C]//Proc of the 20th Artificial Intelligence and Statistics (AISTATS 2017). Cambridge, MA: MIT, 2017: 1273−1284



Zhang Xiaojian, born in 1980. PhD, professor, master supervisor. His main research interests include differential privacy, data mining, and graph data management.

张啸剑, 1980 年生. 博士, 教授, 硕士生导师. 主要研究方向为差分隐私、数据挖掘、图形数据管理.



Zhang Leilei, born in 1997. Master candidate. His main research interests include differential privacy and federated learning.

张雷雷, 1996 年生. 硕士研究生. 主要研究方向为差分隐私、联邦学习.



Zhang Zhizheng, born in 1996. Master candidate. His main research interests include differential privacy and federated analytics.

张治政, 1996 年生. 硕士研究生. 主要研究方向为差分隐私、联邦分析.