计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2022,31(12):294−300 [doi: 10.15888/j.cnki.csa.008875]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 Reptile 的个性化联邦学习算法①

夏　雨, 崔文泉

(中国科学技术大学 管理学院 统计与金融系, 合肥 230026)
通信作者: 夏　雨, E-mail: xiayu01@mail.ustc.edu.cn

摘　要: 在联邦学习背景下, 由于行业竞争、隐私保护等壁垒, 用户数据保留在本地, 无法集中在一处训练. 为充分利用用户的数据和算力, 用户可通过中央服务器协同训练模型, 训练得到的公共模型为用户共享, 但公共模型对于不同用户会产生相同输出, 难以适应用户数据是异质的常见情形. 针对该问题, 提出一种基于元学习方法 Reptile 的新算法, 为用户学习个性化联邦学习模型. Reptile 可高效学习多任务的模型初始化参数, 在新任务到来时, 仅需几步梯度下降就能收敛到良好的模型参数. 利用这一优势, 将 Reptile 与联邦平均 (federated averaging, FedAvg) 相结合, 用户终端利用 Reptile 处理多任务并更新参数, 之后中央服务器将用户更新的参数进行平均聚合, 迭代学习更好的模型初始化参数, 最后将其应用于各用户数据后仅需几步梯度下降即可获得个性化模型. 实验中使用模拟数据和真实数据设置了联邦学习场景, 实验表明该算法相比其他算法能够更快收敛, 具有更好的个性化学习能力.

关键词: 联邦学习; 元学习; 个性化学习; 异质数据; 梯度下降; 隐私保护

引用格式: 夏雨,崔文泉.基于 Reptile 的个性化联邦学习算法.计算机系统应用,2022,31(12):294–300. http://www.c-s-a.org.cn/1003-3254/8875.html

Personalized Federated Learning Algorithm Based on Reptile

XIA Yu, CUI Wen-Quan

(Department of Statistics and Finance, School of Management, University of Science and Technology of China, Hefei 230026, China)

Abstract: In federated learning, due to barriers such as industry competition and privacy protection, users keep data locally and cannot train models in a centralized manner. Users can train models cooperatively through the central server to fully utilize their data and computing power, and they can share the common model obtained by training. However, the common model produces the same output for different users, so it cannot be readily applied to the common situation where users' data are heterogeneous. To solve this problem, this study proposes a new algorithm based on the meta-learning method Reptile to learn personalized federated learning models for users. Reptile can learn the initial parameters of models efficiently for multi-tasks. When a new task arrives, only a few steps of gradient descent are needed for convergence to satisfactory model parameters. This advantage is leveraged, and Reptile is combined with federated averaging (FedAvg). The user terminal uses Reptile to process multi-tasks and update parameters. After that, the central server performs the averaging aggregation of the parameters the user updates and iteratively learns better initial parameters of the model. Finally, after the proposed algorithm is applied to each user's data, personalized models can be obtained by only a few steps of gradient descent. In the experiment, this study uses simulated data and real data to set up federated learning scenarios. The experiment shows that the proposed algorithm can converge faster and offer a better personalized learning ability than other algorithms.

Key words: federated learning; meta-learning; personalized learning; heterogeneous data; gradient descent; privacy protection

① 基金项目: 国家自然科学基金 (71873128, 12171451)
收稿时间: 2022-04-19; 修改时间: 2022-06-01; 采用时间: 2022-06-13; csa 在线出版时间: 2022-08-19

1 引言

作为成功的人工智能范例, AlphaGo[1] 利用神经网络和树搜索, 于 2016 年打败世界围棋冠军, 鼓舞了众多人工智能研究者. 然而, 在 AlphaGo 成功的背后, 它使用了多达 30 万局对弈作为训练数据, 让我们不禁思考现实中大数据的质量和可获得性如何. 实际上, 情况并不乐观: 由于行业竞争、手续复杂等因素, 大多数行业只拥有有限或者低质的数据, 这阻碍了人工智能和机器学习技术的应用[2]. 同时, 数据泄露问题危害着社会权益, 对此国内外出台相关法律条例进行规范, 如中国先后发布《中华人民共和国网络安全法》《中华人民共和国数据安全法》和《中华人民共和国个人信息保护法》, 加强对数据安全和个人信息的保护[3]. 在本文中, 我们将拥有数据的实体称为用户. 随着国内外隐私监管的加强, 不同用户的数据流通受到阻碍, 使用户逐渐成为“数据孤岛”[4]. 对于以上场景, 用户的有限数据沟通不畅, 无法集中在一处训练, 因此用户的数据和算力得不到充分利用, 难以训练出表现良好的机器学习模型, 研究者开始探索如何在不泄露数据隐私的前提下破解“数据孤岛”困境[5].

联邦学习由谷歌提出, 作为面向数据孤岛和隐私保护的解决方案, 具体指在不泄露数据隐私的前提下基于分布在多个设备上的数据集构建机器学习模型[6]. 最初联邦学习引起了一些科技公司的兴趣, 比如谷歌和苹果拥有大量移动端用户, 因此希望通过部署联邦学习系统来改善它们的服务[7]. FedAvg[8] 作为联邦学习的基准算法, 已经得到广泛的研究与应用. FedAvg 在用户数据是独立同分布 (independent and identically distributed, IID) 且平衡的情况下表现良好, 而这种完美数据在现实场景中很少见. 有研究表明数据的异质性减缓了 FedAvg 的收敛速度[9]. 非 IID 是异质的典型情况, 即用户的数据并不服从同一分布, 若用户仍训练公共模型, 学习效果可能不佳. 除此之外, 用户数据量也可能是不平衡的.

有研究尝试通过创建为用户共享的数据子集以改进对非 IID 数据的训练[10], 但直接分配数据给用户有隐私泄露的风险, 不符合联邦学习的前提. 文献 [11] 提出了一种新算法, FedProx, 来克服联邦学习中的异质性, FedProx 在 FedAvg 的目标函数上增加了近端项, 限制每轮用户更新的参数不过多偏离全局参数. 文献 [12] 证明了局部随机梯度下降 (stochastic gradient descent, SGD) 在数据同质和异质时的复杂度, 并提供了最优步长和最优局部迭代次数的值. 但上述工作关注所有用户的数据, 学到的是公共模型, 不同用户应用公共模型时会产生相同输出, 难以学习它们之间的差异, 因此有必要探索能有效处理用户数据异质性的联邦学习算法.

为帮助解决上述问题, 有研究者提出个性化联邦学习, 即不同用户在联邦学习框架下学习不同模型[13]. 元学习是实现个性化联邦学习的有效方法, 思想是学习如何去学习, 可用来学习不同任务的最优初始化参数[14]. 一般而言, 优化模型时模型初始化参数作为超参数, 常需要人为设置, 再进行若干次迭代得到表现良好的模型参数, 但人为设置的初始化参数具有随意性, 可能离最优参数的距离很远, 需迭代成千上万次才能达到不错的效果, 而经学习的初始化参数更靠近最优参数, 可能仅需迭代几次就能获得好的模型.

文献 [15] 提出了一种模型不可知的元学习算法 (model-agnostic meta-learning, MAML), 该算法适用于任何使用 SGD 训练的模型. MAML 表现良好, 但它需要梯度的高阶信息, 这很耗费计算内存, 并且会减缓计算速度. 继 MAML 之后, 文献 [16] 提出了一种只需梯度一阶信息的元学习方法——Reptile. 结合 MAML 和 FedAvg, 文献 [17] 提出了一种新的算法, Per-FedAvg, 将 FedAvg 与 MAML 相结合为用户学习更优的模型初始化参数, 该算法在异质任务上实现了良好的性能, 但需要用户在每次迭代过程中计算 MAML 的梯度, 这涉及高阶导数信息, 对于用户的算力和内存有较高的要求.

联邦学习背景下, 数据孤岛的数据量往往较大, 而每个用户拥有的数据量较少, 应尽可能采取运算效率高的方法学习. 比如移动端训练时移动设备只拥有很小的算力和内存, 难以承载过于复杂的训练过程; 在线学习时应及时响应需求, 训练时间不宜过长. 考虑到用户使用 MAML 更新参数时需要耗费较多的计算资源, 而 Reptile 只需计算梯度的一阶信息, 较 MAML 有计算优势, 更适合上述计算速度和内存受限的场景. 本文结合 Reptile 与 FedAvg, 提出一种新的个性化联邦学习算法, 基于 Reptile 的个性化联邦学习算法 (personalized FedAvg algorithm based on Reptile, Per-FedAvg-Reptile). 中央服务器在每轮更新前随机抽取一定比例的用户参与训练, 被抽取的用户应用 Reptile 处理多任务并更新参数, 之后中央服务器将用户发来的参数平均聚合, 迭代学习合适的初始化参数. 然后将学到的初始化参数代入用户模型, 用户在自己的数据上进行若干步 (甚至

只需一步) 梯度下降后得到表现良好的个性化模型.

2 相关工作

对于元学习, 考虑优化问题: 找到一组初始化参数向量w, 使得对于随机采样的任务τ及相应的损失L_τ, 学习器在几次 SGD 后即可获得表现良好的模型. 其中τ表示具有相似信息的数据集, 这样我们才可能学习到任务之间的共同信息. 接下来, 将介绍两种最通用的元学习方法来实现这个目标, MAML 和 Reptile.

2.1 MAML

MAML 的目标函数是:

$$\min_{w} \mathbb{E}_\tau \left[L_{\tau,B}(\boldsymbol{U}_{\tau,A}(\boldsymbol{w}))\right] \tag{1}$$

其中, $\boldsymbol{w}$是要学习的初始化参数向量, τ划分成训练集A和测试集B, $\boldsymbol{U}_{\tau,A}$表示使用A的数据多次更新$\boldsymbol{w}$后的参数向量, 而$L_{\tau,B}$表示使用B的数据计算的损失. MAML 的工作原理是通过梯度下降优化式 (1), 即计算:

$$\begin{aligned}\boldsymbol{g}_{\text{MAML}} &= \frac{\partial}{\partial \boldsymbol{w}} L_{\tau,B}(\boldsymbol{U}_{\tau,A}(\boldsymbol{w})) \\ &= \boldsymbol{U}'_{\tau,A}(\boldsymbol{w}) \boldsymbol{L}'_{\tau,B}(\boldsymbol{U}_{\tau,A}(\boldsymbol{w}))\end{aligned} \tag{2}$$

式 (2) 中$\boldsymbol{U}_{\tau,A}(\boldsymbol{w}) = w - \alpha(\boldsymbol{g}_1 + \boldsymbol{g}_2 + \cdots + \boldsymbol{g}_k)$(如果$\boldsymbol{w}$使用 SGD 更新了$k$次), 其中, α是学习率, $\boldsymbol{g}_i$是第i次更新的梯度, $i \in \{1, \cdots, k\}$. $\boldsymbol{U}'_{\tau,A}(\boldsymbol{w})$是$\boldsymbol{U}_{\tau,A}(\boldsymbol{w})$的雅各比矩阵. 注意到, 测试集的最小损失可以通过限制对于训练集的若干次更新来实现, 因此 MAML 可快速学习新任务. $\boldsymbol{U}'_{\tau,A}(\boldsymbol{w})$涉及高阶导数信息, 其计算量很大. 不过, 其近似算法一阶 MAML (first-order MAML, FOMAML) 也可达到不错的学习效果. 具体来说, FOMAML 使用单位矩阵替换了$\boldsymbol{U}'_{\tau,A}(\boldsymbol{w})$, 即 FOMAML 使用的梯度是$\boldsymbol{g}_{\text{FOMAML}} = \boldsymbol{L}'_{\tau,B}(\boldsymbol{U}_{\tau,A}(\boldsymbol{w}))$.

2.2 Reptile

在本节中, 我们介绍另一个以一阶梯度为基础的元学习算法, 称为 Reptile. 像 MAML 一样, Reptile 学习一组初始化参数, 这样当我们在个性化应用阶段优化这些参数时, 学习速度会很快. Reptile 的细节如算法 1 所示.

算法 1. Reptile

输入: 任务集合T, 初始化参数向量的学习起点$\boldsymbol{w}$, 学习率ϵ, 迭代轮数K, 梯度下降步数k

输出: 训练好的初始化参数向量

1) 初始化$\boldsymbol{w}$ //其为初始化参数向量的学习起点
2) for 迭代轮次=1, 2, ⋯, K
3) 从T中抽样任务τ
4) 计算$\tilde{\boldsymbol{w}}=\boldsymbol{U}_\tau^k(\boldsymbol{w})$//代表$k$步梯度下降
5) 更新$\boldsymbol{w}\leftarrow\boldsymbol{w}+\epsilon(\tilde{\boldsymbol{w}}-\boldsymbol{w})$
6) end for

在每轮的最后一步中, 我们可以将$\boldsymbol{w} - \tilde{\boldsymbol{w}}$视为某种梯度. 文献 [16] 已经用泰勒近似证明了 MAML 和 Reptile 计算的梯度可以分为两部分: 一部分用来最小化期望损失, 提升任务的整体学习效果; 另一部分用来最大化任务间的泛化, 所以元学习可以学习到任务的共性. 算法 1 还可扩展为并行或批处理版本, 每次迭代考虑n个任务, 并将初始化参数更新为:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \epsilon \frac{1}{n} \sum_{i=1}^{n} (\tilde{\boldsymbol{w}}_i - \boldsymbol{w}) \tag{3}$$

其中, $\tilde{\boldsymbol{w}}_i = \boldsymbol{U}_{\tau_i}^k(\boldsymbol{w})$是第$i$个任务的更新参数. 并行版本和联邦学习处理多用户数据有相似之处, 为将 Reptile 应用于联邦学习场景提供了可能.

3 基于 Reptile 的个性化联邦学习算法

在联邦平均中, 我们旨在解决这样一个问题:

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} \left\{ f(\boldsymbol{w}) \overset{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} f_i(\boldsymbol{w}) \right\} \tag{4}$$

其中, $\boldsymbol{w} \in \mathbb{R}^d$是需要学习的参数向量, $f_i : \mathbb{R}^d \to \mathbb{R}$是用户$i$的损失函数. 假设$\mathcal{D}_i = \{\mathcal{X}_i, \mathcal{Y}_i\}$是用户$i$拥有的数据, 通常我们取$f_i(\boldsymbol{w}) = \mathbb{E}_{(x,y)\sim p_i}[l_i(\boldsymbol{w}; x, y)]$, 其中$p_i$是$\mathcal{X}_i \times \mathcal{Y}_i$的分布, $l_i(\boldsymbol{w}; x, y)$衡量的是数据点$(x, y) \in \mathcal{D}_i$在模型参数为$\boldsymbol{w}$时的损失.

如图 1 所示, 对于 FedAvg, 中央服务器每次随机选取部分用户参与参数更新, 用户在本地通过 SGD 计算梯度信息并传送给中央服务器, 中央服务器平均聚合这些信息来更新模型参数. 注意到, 数据始终保留在用户那里, 所以符合隐私保护的要求.

文献 [17] 将 FedAvg 与元学习方法 MAML 相结合, 提出了 FedAvg 的个性化变体 Per-FedAvg, 在用户更新时用 MAML 替换 SGD, 与 FedAvg 学习的是模型参数不同, Per-FedAvg 学习的是模型的初始化参数, 用户在得到优化的初始化参数后还需在本地数据 SGD 一次才能获得个性化模型. 理论和实验证明了该算法的有效性. 然而, 如前所述, MAML 涉及高阶导数信息, 对于复杂模型, 这在计算代价上是昂贵的. 而

Reptile 只需要计算一阶导数, 局部更新步数也可灵活设置.

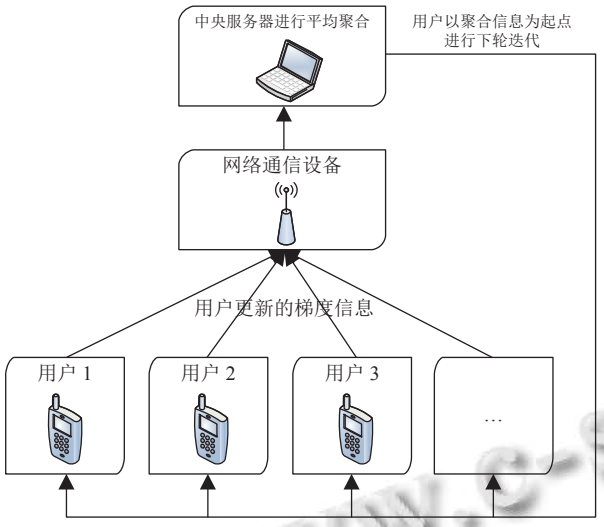


图 1 FedAvg 流程图

本文将 FedAvg 与 Reptile 结合起来, 提出新算法 Per-FedAvg-Reptile, 用户更新时只需计算一阶导数, 因此更适合移动端训练、在线学习等计算速度和内存受限的联邦学习场景. 该算法以 FedAvg 作为框架, 确保用户数据不出本地, 用户通过与中央服务器传递梯度来更新参数, 因此满足隐私保护的要求; 以 Reptile 作为用户的局部更新, 学习用户的模型初始化参数; 在学习完成之后, 用户以优化的初始化参数为起点, 基于本地的额外数据进行一步梯度下降就能获得个性化联邦学习模型. 新提出的算法细节如算法 2 所示, 并且可分为以下步骤:

步骤 1. 中央服务器初始化$\boldsymbol{w}_0$, 作为训练起点, 设定每轮抽取的用户比例为r, 对应算法 2 中的 2 行.

步骤 2. rn个用户被随机选择, 中央服务器向它们发送参数, 对应算法 2 中的 45 行. 考虑到用户可能会突然掉线或暂时退出, 这里不要求所有用户都参与每轮训练.

步骤 3. 对于选定的用户i, 接收参数并赋值给$\boldsymbol{w}_{\text{task}}$, 之后将数据$\mathcal{D}_i$划分为$\tau_{\text{out}}$批, 每批对应 Reptile 中的任务, 我们可以理解为学习用户数据的内部共性. 然后, 每批被划分为τ_{in}个更小批, τ_{in}次梯度下降的步骤会被执行, 我们得到更新的$\boldsymbol{w}_{\text{task_task}}$. $\boldsymbol{w}_{\text{task}}-\boldsymbol{w}_{\text{task_task}}$视为某种梯度, $\boldsymbol{w}_{\text{task}}$以学习率$\alpha$在该梯度上更新, 更新后的参数发给中央服务器, 对应算法 2 中的 11–22 行, 即用户执行的部分.

步骤 4. 中央服务器平均rn组参数作为更新的参数, 对应算法 2 中的 9 行, 这意味着整合$rn\times\tau_{\text{out}}$个任务的学习信息.

步骤 5. 重复步骤 2–4 直到算法收敛. 通常我们不知道何时收敛, 会事先设置迭代轮数的上限.

若$r=1$, 每轮通信需要所有用户执行τ_{out}次本地更新步骤. $\boldsymbol{w}_t^i\in\mathbb{R}^d$被定义为用户$i$在时间$t$的参数向量, 对于用户$i$, 更新过程类似于文献 [18] 中的局部梯度下降, 可表示为 (如果通信从t_0开始):

$$\boldsymbol{w}_{t+1}^i=\begin{cases}\dfrac{1}{n}\displaystyle\sum_{j=1}^{n}\left(\boldsymbol{w}_t^j-\alpha\boldsymbol{g}_t^j\right), & t=t_{p\times\tau_{\text{out}}},\ p\in\{1,2,\cdots,K\}\\ \boldsymbol{w}_t^i-\alpha\boldsymbol{g}_t^i, & \text{其他}\end{cases}\tag{5}$$

其中, $\boldsymbol{g}_t^i=\boldsymbol{g}_{t,\tau_{\text{in}}}^i=\boldsymbol{g}_{t,\tau_{\text{in}}-1}^i+\nabla f_i\left(\boldsymbol{w}_t^i-\boldsymbol{g}_{t,\tau_{\text{in}}-1}^i\right)$且$\boldsymbol{g}_{t,1}^i=\nabla f_i\left(\boldsymbol{w}_t^i\right)$. 可以看到, 参数由所有用户进行 Reptile更新, 并在通信的时间点进行平均聚合. 算法 2 的优化方法是梯度下降, 所以可以应用于任何使用梯度下降的模型, 适用面较广.

算法 2. Per-FedAvg-Reptile

输入: n个用户的数据集合$\{\mathcal{D}_i=\{\mathcal{X}_i,\mathcal{Y}_i\}\}_{i=1}^n$, 初始化参数的学习起点$\boldsymbol{w}_0$, 学习率$\alpha$, 迭代轮数$K$, 主要超参数: r, τ_{out}, τ_{in}

输出: 训练好的初始化参数向量

1) 中央服务器执行:
2) 初始化$\boldsymbol{w}_0$, r
3) for k=1,⋯,K
4) $m\leftarrow\max(rn,1)$
5) $C_k\leftarrow$(随机抽取m个用户构成的集合)
6) for 每个用户$i\in C_k$并行
7) $\boldsymbol{w}_{k+1}^i\leftarrow UserUpdate(i,\boldsymbol{w}_k)$
8) end for
9) $\boldsymbol{w}_{k+1}\leftarrow\frac{1}{rn}\sum_{i\in C_k}\boldsymbol{w}_{k+1}^i$
10) end for
11) 用户执行$UserUpdate(i,\boldsymbol{w}_k)$:
12) $\boldsymbol{w}_{\text{task}}\leftarrow\boldsymbol{w}_k$
13) $\mathcal{B}\leftarrow$(将$\mathcal{D}_i$划分成τ_{out}个批)
14) for 批次$b\in\mathcal{B}$
15) $B\leftarrow$(将b划分成τ_{in}个更小批)
16) $\boldsymbol{w}_{\text{task\_task}}\leftarrow\boldsymbol{w}_{\text{task}}$
17) for 小批次$mb\in B$
18) $\boldsymbol{w}_{\text{task\_task}}\leftarrow\boldsymbol{U}_{mb}^1(\boldsymbol{w}_{\text{task\_task}})$//在 mb 上进行一次梯度下降
19) end for
20) $\boldsymbol{w}_{\text{task}}\leftarrow\boldsymbol{w}_{\text{task}}+\alpha(\boldsymbol{w}_{\text{task\_task}}-\boldsymbol{w}_{\text{task}})$
21) end for
22) 返回$\boldsymbol{w}_{\text{task}}$给中央服务器

4 实验分析

在本节中, 我们通过模拟数据和真实数据来比较个性化模型的效果. 考虑 3 种算法: (1) 本文提出的新算法 Per-FedAvg-Reptile; (2) 文献 [17] 将 FedAvg 与 FOMAML 结合的算法 Per-FedAvg-FO; (3) FedAvg with one update, 即进行一次额外更新的 FedAvg. 由于 (1) 和 (2) 训练获得良好的初始化参数后, 还需在用户端梯度下降一次才能获得个性化模型, 为了公平比较, 即使 FedAvg 学习的不是模型初始化参数, 也将训练好的参数在用户端进行一次梯度下降更新. 70% 的用户数据用于训练, 剩下的数据用于测试. 此外, 为了避免模型看到所有的测试数据, 我们将测试数据分成两部分: 30% 的测试数据用于梯度下降获得个性化模型, 70% 的测试数据用于测试模型的效果. 实验采用 PyTorch 作为学习框架.

4.1 模拟数据

为了生成异质的模拟数据, 我们参考了文献 [11] 提出的生成方法. 具体来说, 样本(X,Y)根据模型$y=\arg\max(\mathrm{Softmax}(\boldsymbol{W}\boldsymbol{x}+\boldsymbol{b}))$生成, 其中$\boldsymbol{x}\in\mathbb{R}^{60}$, $\boldsymbol{W}\in\mathbb{R}^{10\times 60}$, $\boldsymbol{b}\in\mathbb{R}^{10}$. 对于用户$i$, 从$\mathcal{N}(u_i,1)$抽样出$W_i$和$b_i$的每个元素, 其中$u_i\sim\mathcal{N}(0,\delta)$; $\boldsymbol{x}_i\sim\mathcal{N}(\boldsymbol{v}_i,\boldsymbol{\Sigma})$, 其中$\boldsymbol{\Sigma}$是对角元素为$\boldsymbol{\Sigma}_{j,j}=j^{-1.2}$的对角矩阵, $\boldsymbol{v}_i$的每个元素从$\mathcal{N}(B_i,1)$抽样, $B_i\sim N(0,\theta)$. 用户数据量由均匀分布$U(100,1000)$抽样取整确定.

通过仔细观察, 可以看出δ的大小影响u_i的方差, u_i决定用户i模型参数的均值, 因此δ控制用户模型之间的差异, 其值越大, 用户模型之间的差异越大; 而θ的大小影响B_i的方差, B_i决定$\boldsymbol{v}_i$的均值, $\boldsymbol{v}_i$决定$\boldsymbol{x}_i$的均值, 因此θ控制用户数据之间的差异, 其值越大, 用户数据的差异越大. 我们改变δ, θ的值来生成 4 个异质程度不同的数据集, 表示为模拟数据 (δ,θ), 如图 2 所示.

假设有$n=30$个用户协同学习模型, 并且在每次迭代中随机选择rn个用户参与训练, 其中$r=0.2$, τ_{out}和τ_{in}都被设置为 4, 学习率$\alpha=0.01$. 请注意, 算法结果对应的是运行一步梯度下降后所有用户的平均测试精度, 并且我们运行 10 次实验以获得具有标准差的平均性能[19], 图 2 中误差棒的中点为 10 次实验的平均测试准确率, 数值越大意味着测试准确率越高, 中点到两端的距离为 10 次测试准确率的标准差, 长度越长意味着测试准确率的波动性越大.

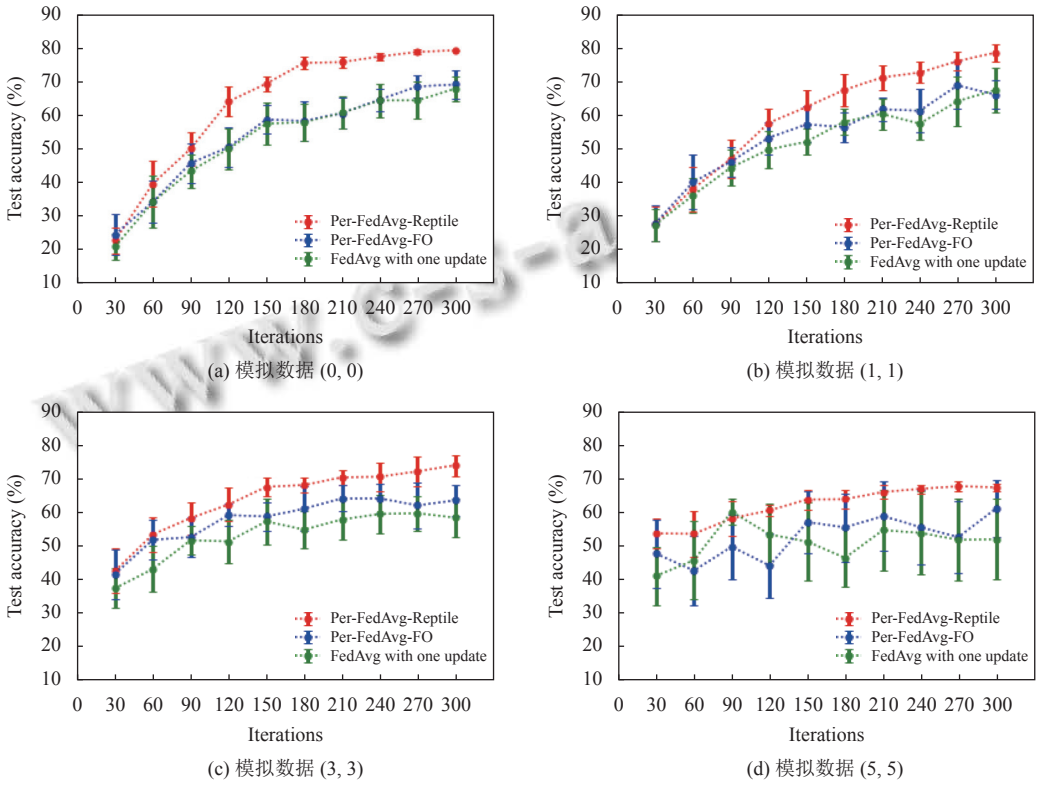


图 2 模拟数据实验

在图 2 中, 横坐标为迭代次数, 对应算法 2 中的迭代轮数K; 纵坐标为测试准确率, 衡量算法性能. 从图 2(a)到图 2(d), 由于(δ,θ)逐渐增大, 用户数据和用户模型的异质程度增加, 学习难度随之加大. 我们可以看到, Per-FedAvg-Reptile 的学习折线一直处于上方, 说明新算法在测试准确率上优于其他两种算法; 而且 Per-FedAvg-Reptile 的误差棒较短, 说明 10 次实验的测试准确率相当, 性能更具鲁棒性. 当异质程度很显著时, 如图 2(d), Per-FedAvg-FO 和 FedAvg with one update的误差棒长度明显增长, 测试准确率的波动加大, 说明这两种算法缺乏鲁棒性, 而 Per-FedAvg-Reptile 仍能快速收敛到良好的测试准确率.

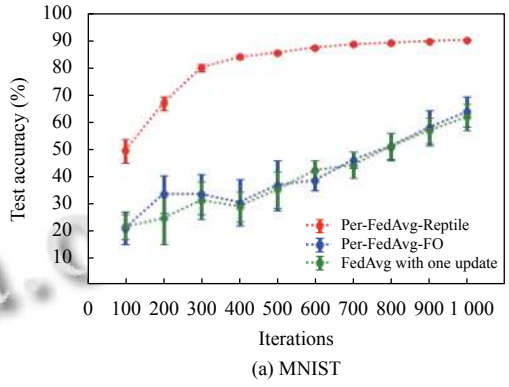
4.2 真实数据

为继续验证新方法, 我们在标准数据集 MNIST[20] 和 CIFAR10[21] 上构建了联邦学习场景. MINST 是包括 60 000 个训练样本和 10 000 个测试样本的黑白图像集, 内容是手写数字 0–9; CIFAR10 是包括50 000 个训练样本和 10 000 个测试样本的彩色图像集, 内容是 10 类生活图片. MINST 和 CIFAR10 来源于 torchvision 中的 torchvision.datasets 库.

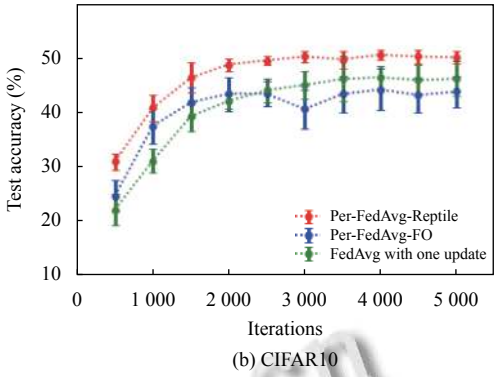
我们考虑文献 [19] 提出的异质数据划分方法, 为用户分配规模大小和类别比例都不平衡的数据. 具体而言, 假设随机向量$\boldsymbol{p}_k \sim Dir_n(0.5)$, $p_{k,i}$为$\boldsymbol{p}_k$的第i个元素, 将$p_{k,i}$比例的类k样本分配给用户i.

模型采用包含两个隐藏层的全连接神经网络, 规模分别为 80 个神经元和 60 个神经元, 指数线性单元 (exponential linear unit, ELU) 作为激活函数. 选用更复杂的模型可获得更高精度, 而本文关注的是算法之间的差异, 所以只采用两层神经网络. 我们假设有$n = 50$个用户协同学习模型, 其他超参数与模拟数据相同, 所有算法的结果也是对应运行一步梯度下降后所有用户的平均测试精度, 运行 10 次试验以获得具有标准差的平均性能. 与 MNIST 相比, 学习 CIFAR10 时迭代更多次, 因为它作为彩色图像集更难学习.

观察图 3, 我们可以得出一些结论. 在学习黑白图像集 MNIST 时, Per-FedAvg-Reptile 的学习折线处于上方, 在测试准确率方面明显优于其他两种算法, 而且折线率先趋于平稳说明收敛速度更快; 而在学习彩色图像集 CIFAR10 时, 3 种算法的性能较接近, 我们猜测这和任务的难度有关, 对于更难学的 CIFAR10, Per-FedAvg-Reptile 的相对优势被压缩了. 但当迭代轮数为 5 000 时, Per-FedAvg-Reptile、FedAvg with one update 和 Per-FedAvg-FO 的误差棒中点分别为 50.09%、46.13% 和 43.77%, 新算法在测试准确率方面仍优于另外两种算法, 且收敛速度更快. 总之, Per-FedAvg-Reptile 在收敛速度和测试准确率上都表现更优.



(a) MNIST



(b) CIFAR10

图 3 真实数据实验

5 结论与展望

本文结合联邦平均和 Reptile, 提出了一种新算法 Per-FedAvg-Reptile, 旨在学习良好的模型初始化参数, 以期快速学习到个性化联邦学习模型. 我们在模拟数据和真实数据上验证了该算法的有效性, 与现有的两种算法相比, Per-FedAvg-Reptile 在收敛速度和测试准确率上表现更好, 可以学习到更加个性化的用户模型. 在未来的研究中, 会考虑 3 种方向的改进: 现实中, 用户的重要程度可能各不相同, 所以中央服务器可以在聚合用户信息时采取不同权重, 比如按照样本量比例分配权重; 探究新算法中超参数的作用和选择; 分析新算法的收敛性理论.

参考文献

1 Silver D, Huang A, Maddison CJ, *et al*. Mastering the game

of Go with deep neural networks and tree search. Nature, 2016, 529(7587): 484–489. [doi: 10.1038/nature16961]
2 Yang Q, Liu Y, Chen TJ, *et al*. Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology, 2019, 10(2): 12. [doi: 10.1145/3298981]
3 陈磊, 刘文懋. 合规视角下的数据安全技术前沿与应用. 数据与计算发展前沿, 2021, 3(3): 19–31. [doi: 10.11871/jfdc.issn.2096-742X.2021.03.003]
4 Yang Q, Liu Y, Cheng Y, *et al*. Federated learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, 2019, 13(3): 1–207. [doi: 10.2200/S00960ED2V01Y201910AIM043]
5 杨强. AI 与数据隐私保护: 联邦学习的破解之道. 信息安全研究, 2019, 5(11): 961–965. [doi: 10.3969/j.issn.2096-1057.2019.11.003]
6 McMahan HB, Moore E, Ramage D, *et al*. Federated learning of deep networks using model averaging. arXiv: 1602.05629, 2016.
7 Kairouz P, McMahan HB, Avent B, *et al*. Advances and open problems in federated learning. Foundations and Trends® in Machine Learning, 2021, 14(1–2): 1–210. [doi: 10.1561/2200000083]
8 McMahan HB, Moore E, Ramage D, *et al*. Communication-efficient learning of deep networks from decentralized data. Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. Fort Lauderdale: PMLR, 2017. 1273–1282.
9 Li X, Huang KX, Yang WH, *et al*. On the convergence of FedAvg on Non-IID data. Proceedings of the 8th International Conference on Learning Representations. Addis Ababa: ICLR, 2020. 1–26.
10 Zhao Y, Li M, Lai LZ, *et al*. Federated learning with Non-IID data. arXiv:1806.00582, 2018.
11 Li T, Sahu AK, Zaheer M, *et al*. Federated optimization in heterogeneous networks. Proceedings of Machine Learning and Systems 2020. Austin: MLSys, 2020. 429–450.
12 Khaled A, Mishchenko K, Richtárik P. Tighter theory for local SGD on identical and heterogeneous data. Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics. Palermo: PMLR, 2020. 4519–4529.
13 Dinh CT, Tran NH, Nguyen TD. Personalized federated learning with moreau envelopes. Proceedings of the 34th International Conference on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2020. 1796.
14 Vanschoren J. Meta-learning. Automated Machine Learning: Methods, Systems, Challenges. Cham: Springer, 2019. 35–61. [doi: 10.1007/978-3-030-05318-5_2]
15 Finn C, Abbeel P, Levine S. Model-agnostic meta-learning for fast adaptation of deep networks. Proceedings of the 34th International Conference on Machine Learning. Sydney: PMLR, 2017. 1126–1135.
16 Nichol A, Achiam J, Schulman J. On first-order meta-learning algorithms. arXiv:1803.02999, 2018.
17 Fallah A, Mokhtari A, Ozdaglar A. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. Proceedings of the 34th International Conference on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2020. 300.
18 Khaled A, Mishchenko K, Richtárik P. First analysis of local GD on heterogeneous data. arXiv:1909.04715, 2019.
19 Yurochkin M, Agarwal M, Ghosh S, *et al*. Bayesian nonparametric federated learning of neural networks. Proceedings of the 36th International Conference on Machine Learning. Long Beach: PMLR, 2019. 7252–7261.
20 Deng L. The MNIST database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine, 2012, 29(6): 141–142. [doi: 10.1109/msp.2012.2211477]
21 Krizhevsky A. Learning multiple layers of features from tiny images. Toronto: University of Toronto, 2009.

(校对责编: 牛欣悦)