基于主从博弈的分层联邦学习激励机制研究

贾云健[①] 黄 宇[①] 梁 靓*[①] 万杨亮[②] 周继华[③]

[①](重庆大学微电子与通信工程学院 重庆 400044)

[②](95696部队 重庆 400030)

[③](重庆金美通信有限责任公司复杂环境通信重庆市重点实验室 重庆 400030)

摘 要：为了优化分层联邦学习(FL)全局模型的训练时延，针对实际场景中终端设备存在自私性的问题，该文提出一种基于博弈论的激励机制。在激励预算有限的条件下，得到了终端设备和边缘服务器之间的均衡解和最小的边缘模型训练时延。考虑终端设备数量不同，设计了基于主从博弈的可变激励训练加速算法，使得一次全局模型训练时延达到最小。仿真结果显示，所提出的算法能够有效降低终端设备自私性带来的影响，提高分层联邦学习全局模型的训练速度。

关键词：分层联邦学习；博弈论；激励机制

中图分类号：TN92 **文献标识码**：A **文章编号**：1009-5896(2023)04-1366-08

DOI: 10.11999/JEIT220175

Research on Hierarchical Federated Learning Incentive Mechanism Based on Master-Slave Game

JIA Yunjian[①] HUANG Yu[①] LIANG Liang[①] WAN Yangliang[②] ZHOU Jihua[③]

[①](*School of Microelectronics and Communication Engineering, Chongqing University, Chongqing* 400044, *China*)

[②](95696 *Troops*, *Chongqing* 400030, *China*)

[③](*Chongqing Key Laboratory of Complex Environment Communication*, *Chongqing Jinmei Communication Co. Ltd. Chongqing* 400030, *China*)

Abstract: In order to optimize the training delay of the hierarchical Federated Learning (FL) global model, focusing on the selfishness of the terminal devices in the actual scene, an incentive mechanism based on game theory is proposed. Under the condition of limited incentive budget, the equilibrium solution between terminal devices and edge servers and the minimum edge model training delay are obtained. Considering the different number of terminal devices, a variable incentive training acceleration algorithm based on Stackelberg game is designed to minimize the training delay of a global model. Simulation results demonstrate that the proposed algorithm can effectively reduce the impact of terminal devices selfishness and improve the training speed of hierarchical federated learning global model.

Key words: Hierarchical Federated Learning (FL); Game theory; Incentive mechanism

1 引言

随着各种智能设备的不断普及，依赖数据和计算能力的机器学习技术得到了迅速发展。为了解决机器学习模型训练面临的数据安全问题，2017年谷歌提出了一种新的分布式机器学习方法——联邦学习(Federated Learning, FL)[1]。在联邦学习的架构中，设备用户的原始数据不会上传至数据中心，而是留在设备本地进行模型训练，设备只上传训练出的模型参数[2]。联邦学习将机器学习与在中心服务器中获取、存储和训练数据分离开来，实现了用户数据隐私保护[3]。

自从谷歌提出联邦学习这个概念后，联邦学习就成为机器学习领域的一个研究热点。McMahan等人[4]提出了一个基于模型平均的联邦学习实用模型，并进行了广泛的实证评估，这篇文章提出的联

收稿日期：2022-02-25；改回日期：2022-06-27；网络出版：2022-08-16

*通信作者： 梁靓 liangliang@cqu.edu.cn

基金项目：国家自然科学基金(62071075, 61971077)，重庆市自然科学基金(cstc2020jcyj-msxmX0704)

Foundation Items: The National Natural Science Foundation of China (62071075, 61971077), The Natural Science Foundation of Chongqing (cstc2020jcyj-msxmX0704)

邦平均算法(Federated Average algorithm, FedAvg)成为一个经典的联邦学习算法。在此基础上，研究者针对FedAvg算法优化，进行了一系列研究。Li等人[5]在FedAvg的基础上引入了一个修正项，提出了FedProx (FedAvg with the Proximal term)算法，它允许在设备之间局部地执行可变量的工作，并且依赖这个修正项来确保方法的稳定性，解决了联邦学习固有的系统异质性和统计异质性问题。Mills等人[6]采用分布式Adam优化技术和模型压缩技术，提出了一种改进的FedAvg算法CE-FedAvg (Communication-Efficient FedAvg)，该算法可以减少达到目标精度所需的通信轮次和每一轮需要加载的数据量，解决了联邦学习在物联网边缘计算的高效通信问题。Li等人[7]则从算法的公平性角度出发，提出了q-Fair FL算法，它重新权衡了FedAvg算法中的目标函数，在损失函数中分配更高的权重给损失较高的设备，从而使训练精度分布更加均匀。为了进一步优化联邦学习的性能，有研究者对传统联邦学习框架进行了改进。Liu等人[8]将基于边缘和基于云的联邦学习结合起来，提出了分层联邦学习架构，该分层联邦学习架构与基于云的联邦学习架构相比，模型训练时间和终端设备的能耗都得到了降低。Abad等人[9]通过分簇法研究了在蜂窝网中的分层联邦问题，优化了分层联邦学习的全局通信时延。

然而不管是针对传统联邦学习框架还是分层联邦学习框架，目前已有的研究大多集中在优化联邦学习算法以提高模型训练性能上，用于激励终端设备参加模型训练的激励机制在很大程度上却被忽视了。大多数流行的分布式训练算法都是使用小批量随机梯度下降[10]，这在实际训练中，需要等待每一个同步批次中最慢的设备，导致随机优化的完全同步往往很慢，即受到“掉队效应”[11]的影响，这在异构网络中更为明显。同时，目前的大多数研究都做出了一个乐观的假设，即所有的终端设备在受到邀请时，都将无条件地参与联邦学习。这在现实世界中是不实际的，因为在联邦模型训练过程中，终端设备在计算和通信方面承受着相当大的开销[12]，如果没有精心设计的激励机制，具有自私性的终端设备将不会拿出足够的资源甚至不愿意加入到联邦学习任务中来，这将导致十分严重的“掉队效应”，使得模型的训练时间大大增加，影响联邦学习的使用。

针对上述问题，本文在分层联邦学习框架下，考虑实际场景中每个边缘服务器下连接的终端设备数量不同，首先对模型训练过程进行了建模分析，得出一次全局模型训练的时间消耗和资源消耗。然后在终端设备和边缘服务器之间设计了两层主从博弈(即Stackelberg博弈[13])，通过调整分配给每个边缘服务器的激励预算值，提出了基于主从博弈的可变激励训练加速算法。该算法能够刺激终端设备更加积极地参与到联邦学习的任务中来，有效地减小“掉队效应”的影响，从而最小化全局模型训练时间。

2　系统模型

如图1所示在分层FL架构中，假设有一个云服务器C，边缘服务器集合为$\mathcal{N}=\{i:i=1,2,\cdots,N\}$，边缘服务器$i$下连接的终端设备集合为$\mathcal{M}_i=\{m:m=1,2,\cdots,M\}$，每个终端设备集合大小不相同。考虑完全同步的FL，完成一次全局训练过程如下：

终端设备端。终端设备m基于本地的数据集来进行本地模型训练，假设所有的终端设备的本地数据集大小相同，为了达到相同的本地模型精度$\theta\in(0,1)$，终端设备m需要进行迭代的次数可以表示为[14]

$$L(\theta)=\mu\log_2\left(\frac{1}{\theta}\right) \tag{1}$$

其中，μ是一个取决于数据集大小和机器学习任务的参数。$f_{i:m}$表示边缘服务器i下的终端设备m进行本地计算时的CPU频率，C_m表示完成1次本地迭代计算任务需要的总的CPU转圈数。那么完成$L(\theta)$次本地迭代，边缘服务器i下的终端设备m所消耗的能量和时间可以分别表示为

$$e_{i:m}^{\mathrm{cmp}}=L(\theta)kC_m(f_{i:m})^2 \tag{2}$$

$$t_{i:m}^{\mathrm{cmp}}=L(\theta)\frac{C_m}{f_{i:m}} \tag{3}$$

其中，k是一个取决于芯片结构的系数。为了使终端设备投入更多的计算资源以减小本地训练模型所花的时间，边缘服务器端会引入激励机制，即边缘服务器i向其下面的终端设备提供奖励，$q_{i:m}$表示边缘服务器i对其服务范围内的终端设备m提供的CPU频率单价，那么终端设备m迭代一次获得的收入为$q_{i:m}f_{i:m}$。本地模型精度达到θ之后，终端设备

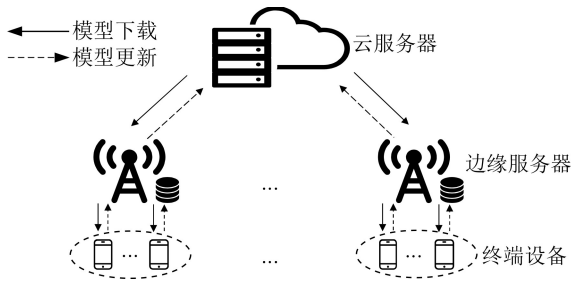

图 1 系统模型图

向对应的边缘服务器上传训练得到的参数。假设每个边缘服务器能够分配给终端设备的信道总带宽均为B，边缘服务器将其均分给下面的每一个终端设备，那么传输率可以表示为

$$r_{m\to i}=\frac{B}{m}\log_2\left(1+\frac{h_m u_m}{N_0}\right) \tag{4}$$

其中，N_0为噪声功率，u_m为传输功率，h_m为信道增益。终端设备m完成一次参数传输至边缘服务器i所需要的时间为

$$t_{m\to i}^{\mathrm{com}}=\frac{d_m}{r_{m\to i}} \tag{5}$$

其中，d_m表示终端设备m传输的参数量，所需要的能量为

$$e_{m\to i}^{\mathrm{com}}=t_{m\to i}^{\mathrm{com}}u_m \tag{6}$$

边缘服务器端。边缘服务器i在收到终端设备传来的参数后，会进行聚合然后再分发下去。以上的过程会迭代多次，直到所有的边缘服务器达到一个相同边缘服务器模型精度ε。为了达到所需精度，对于一个凸的机器学习任务，边缘服务器需要迭代的次数可以表示为[14]

$$I(\varepsilon,\theta)=\frac{\delta\left(\log_2\left(\frac{1}{\varepsilon}\right)\right)}{1-\theta} \tag{7}$$

其中，δ是取决于学习任务的参数。由于边缘服务器通常拥有强大的计算能力和稳定的能量供给，所以边缘服务器端的模型参数聚合和分发所产生的时间与能量消耗在本文中没有考虑。对于终端设备来说，接收分发下来的参数所消耗的时间和能量相比于它上传参数要小得多，在这里本文也不予考虑。因此，边缘服务器i完成$I(\varepsilon,\theta)$次迭代，所需要的时间为

$$T_i=I(\varepsilon,\theta)[\max_{m\in\mathcal{M}_i}(t_{i:m}^{\mathrm{cmp}})+t_{m\to i}^{\mathrm{com}}] \tag{8}$$

终端设备m消耗的总能量为

$$E_{i:m}=I(\varepsilon,\theta)(e_{i:m}^{\mathrm{cmp}}+e_{m\to i}^{\mathrm{com}}) \tag{9}$$

边缘服务器会向云服务器上传满足精度要求的边缘服务器模型参数，假设边缘服务器的传输率都为r_i，上传的参数量为d_i，那么上传1次参数，边缘服务器的时间和能量消耗分别为

$$t_{i\to C}^{\mathrm{com}}=\frac{d_i}{r_i} \tag{10}$$

$$e_{i\to C}^{\mathrm{com}}=u_i t_{i\to C}^{\mathrm{com}} \tag{11}$$

其中，u_i表示边缘服务器的传输功率。

云服务器端：在接收到边缘服务器端传来的模型参数后，云服务器端进行参数聚合并更新模型，这样就完成了一次全局迭代。相较于其他环节，这个聚合时间非常短，本文也不考虑。那么，一次全局迭代，所需要的总时间为

$$T=\max_{i\in\mathcal{N}}T_i+t_{i\to C}^{\mathrm{com}} \tag{12}$$

云服务器负责分配激励预算给每个边缘服务器，总预算为V。边缘服务器i所分配到的激励预算为W_i。定义边缘服务器i对于一次全局迭代的时间贡献度为$\frac{1}{T_i}$，时间贡献度越大，表示边缘服务器i迭代到所要求的精度的时间越短。云服务器会基于每个边缘服务器的时间贡献度来给予边缘服务器奖励，云服务器端总的奖励为R_c，边缘服务器i从云服务器端获得的奖励定义为$R_c\frac{1}{T_i}$。

3 基于主从博弈的激励机制

3.1 博弈问题定义

本文考虑信息对称场景，即每个终端设备会在训练开始前向其所连接的边缘服务器报告自己能够提供的最大算力、本地数据集大小等先验信息。在终端设备层和边缘服务器层之间引入主从博弈(即Stackelberg博弈)。将终端设备作为跟随者(follower)，边缘服务器作为领导者(leader)。边缘服务器i决定给每个终端设备m的CPU频率单价$q_{i:m}=q_{i:1}$，$q_{i:2},\cdots,q_{i:m}$。根据报价，每个边缘服务器服务范围内的各个终端设备向其报告自己用于参与训练的CPU频率$f_{i:m}=f_{i:1},f_{i:2},\cdots,f_{i:m}$，然后边缘服务器再调整单价$q_{i:m}$。边缘服务器$i$下的终端设备$m$的效用函数可以表示为

$$U_{i:m}=I(\varepsilon,\theta)L(\theta)q_{i:m}f_{i:m}-\eta E_{i:m} \tag{13}$$

其中，第1项为终端设备m完成一次全局迭代获得的激励奖励，第2项为总的计算和传输能耗。系数η用于匹配效用函数前后两项的数量级。边缘服务器i的效用函数可以表示为

$$U_i=R_c\frac{1}{T_i}-\sum_{m\in\mathcal{M}_i}q_{i:m}f_{i:m}-\eta e_{i\to C}^{\mathrm{com}} \tag{14}$$

其中，第1项为边缘服务器i从云服务器获得的奖励，第2项为边缘服务器i向下面的终端设备支出的激励总和，第3项为边缘服务器i上传参数的传输能耗。

博弈框架如图2所示。首先明确引入激励机制的目的是减小完成一次全局迭代的时间。云服务器作为激励预算分配者(Allocator)，负责为每个边缘服务器分配用于激励的预算W_i，然后每个边缘服务器和其服务范围内的终端设备形成1个博弈簇，进行Stackelberg博弈。在1个博弈簇内，每个终端

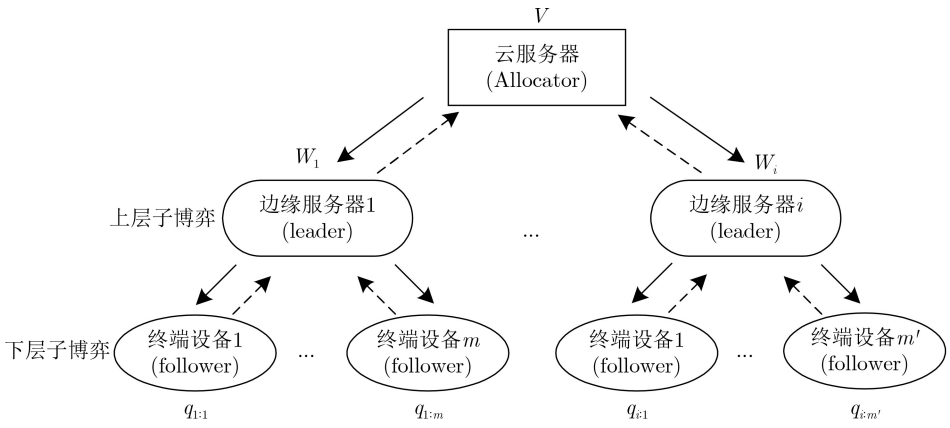

图 2 博弈框架图

设备$m \in \mathcal{M}_i$根据边缘服务器i的激励报价，决定自己投入训练任务的CPU频率，从而最大化自己的效用函数。故会产生m个下层子博弈问题。边缘服务器i根据其服务范围内的所有终端设备的CPU频率投入情况，再重新调整自己给出的激励报价，最大化自己的效用函数。故产生一个上层子博弈问题。一个博弈簇内上下两层子博弈反复进行，直到达到纳什均衡点。

下层子博弈问题定义为

$$\left.\begin{aligned}&\max_{f_{i:m}}\ U_{i:m}=I(\varepsilon,\theta)L(\theta)q_{i:m}f_{i:m}-\eta E_{i:m}\\&\text{s.t. }\ f_{i:m}^{\min}\le f_{i:m}\le f_{i:m}^{\max}\end{aligned}\right\}\tag{15}$$

上层子博弈问题定义为

$$\left.\begin{aligned}&\max_{q_{i:m}}\ U_i=R_c\frac{1}{T_i}-\sum_{m\in\mathcal{M}_i}q_{i:m}f_{i:m}-\eta e_{i\to C}^{\mathrm{com}}\\&\text{s.t. }\sum_{m\in\mathcal{M}_i}q_{i:m}f_{i:m}\le W_i\\&\quad\sum_{i\in\mathcal{N}}W_i\le V\end{aligned}\right\}\tag{16}$$

3.2 博弈均衡解分析

3.2.1 下层子博弈求解

为了找到下层子博弈的均衡解，首先对$U_{i:m}$求1阶导数，可以得到

$$\frac{\partial U_{i:m}}{\partial f_{i:m}}=I(\varepsilon,\theta)L(\theta)q_{i:m}-2\eta I(\varepsilon,\theta)L(\theta)kC_mf_{i:m}\tag{17}$$

因为

$$\frac{\partial^2U_{i:m}}{\partial f_{i:m}^2}=-2\eta I(\varepsilon,\theta)L(\theta)kC_m<0\tag{18}$$

所以$U_{i:m}$是严格凹的，保证了纳什均衡解存在且唯一。可得均衡解为

$$f_{i:m}^*(q_{i:m})=\max\left\{\frac{q_{i:m}}{2\eta kC_m},f_{i:m}^{\max}\right\}\tag{19}$$

3.2.2 上层子博弈求解

边缘服务器i的效用函数式(14)由3部分组成，(1)$R_c\dfrac{1}{T_i}$，(2)$-\sum\limits_{m\in\mathcal{M}_i}q_{i:m}f_{i:m}$，(3)$-\eta e_{i\to C}^{\mathrm{com}}$。由前面系统模型的描述中可知，$-\eta e_{i\to C}^{\mathrm{com}}$可视为一个常量，所以下面分析(1)和(2)的凹凸情况。

分析(1)，令$h=R_c\dfrac{1}{T_i}$并代入$f_{i:m}^*$可得表达式

$$h(q_{i:m})=R_c\frac{1}{I(\varepsilon,\theta)\left[\max\limits_{m\in\mathcal{M}_i}\left(2\eta kC_m^2L(\theta)/q_{i:m}\right)+t_{m\to i}^{\mathrm{com}}\right]}\tag{20}$$

$h(q_{i:m})$的黑塞矩阵定义为

$$\boldsymbol{A}=\begin{pmatrix}\dfrac{\partial^2h}{\partial q_{i:1}^2}&\cdots&\dfrac{\partial^2h}{\partial q_{i:1}\partial q_{i:m}}\\\vdots&\ddots&\vdots\\\dfrac{\partial^2h}{\partial q_{i:m}\partial q_{i:1}}&\cdots&\dfrac{\partial^2h}{\partial q_{i:m}^2}\end{pmatrix}\tag{21}$$

由于

$$\frac{\partial^2h}{\partial q_{i:m}^2}=\frac{-4\eta kR_cL(\theta)I(\varepsilon,\theta)^2C_m^2t_{m\to i}^{\mathrm{com}}}{\left(2\eta kL(\theta)I(\varepsilon,\theta)C_m^2+I(\varepsilon,\theta)t_{m\to i}^{\mathrm{com}}q_{i:m}\right)^3}\tag{22}$$

$$\frac{\partial^2h}{\partial q_{i:m}\partial q_{i:n}}=0\ (m\ne n)\tag{23}$$

所以$h(q_{i:m})$的黑塞矩阵可进一步表示为

$$\boldsymbol{A}=\begin{pmatrix}\dfrac{-4\eta kR_cL(\theta)I(\varepsilon,\theta)^2C_m^2t_{1\to i}^{\mathrm{com}}}{\left(2\eta kL(\theta)I(\varepsilon,\theta)C_m^2+I(\varepsilon,\theta)t_{1\to i}^{\mathrm{com}}q_{i:1}\right)^3}&\cdots&0\\\vdots&\ddots&\vdots\\0&\cdots&\dfrac{-4\eta kR_cL(\theta)I(\varepsilon,\theta)^2C_m^2t_{m\to i}^{\mathrm{com}}}{\left(2\eta kL(\theta)I(\varepsilon,\theta)C_m^2+I(\varepsilon,\theta)t_{m\to i}^{\mathrm{com}}q_{i:m}\right)^3}\end{pmatrix}\tag{24}$$

矩阵$\boldsymbol{A}$的所有特征值均小于等于0，矩阵$\boldsymbol{A}$负定，所以$h(q_{i:m})$为凹函数。

分析(2)，令$g=-\sum\limits_{m\in\mathcal{M}_i}q_{i:m}f_{i:m}$，代入$f_{i:m}^*$可得表达式

$$g(q_{i:m})=-\sum_{m\in\mathcal{M}_i}\frac{1}{2\eta kC_m}{q_{i:m}}^2 \tag{25}$$

$g(q_{i:m})$对$q_{i:m}$求2阶导可得

$$\frac{\partial^2 g}{\partial q_{i:m}^2}=-\frac{1}{\eta kC_m}<0 \tag{26}$$

所以$g(q_{i:m})$为严格凹的。根据以上分析可知，边缘服务器i的效用函数U_i为一个凹函数。

下面求解上层子博弈。式(16)等效于

$$\left.\begin{aligned}&\min_{q_{i:m}}\ U_i=-R_c\frac{1}{T_i}+\sum_{m\in\mathcal{M}_i}q_{i:m}f_{i:m}^*+\eta e_{i\to C}^{\mathrm{com}}\\&\text{s.t. }y_1=\sum_{m\in\mathcal{M}_i}q_{i:m}f_{i:m}^*-W_i\le 0\\&\quad y_2=\sum_{i\in\mathcal{N}}W_i-V\le 0\end{aligned}\right\} \tag{27}$$

上述问题变成一个凸问题，可以用拉格朗日乘数法来解。其拉格朗日函数可以表示为

$$\begin{aligned}L(q_{i:m},\lambda_1,\lambda_2)=&-R_c\frac{1}{T_i}+\lambda_1\left(\sum_{m\in\mathcal{M}_i}q_{i:m}f_{i:m}^*-W_i\right)\\&+\lambda_2\left(\sum_{i\in\mathcal{N}}W_i-V\right)\end{aligned} \tag{28}$$

通过KKT条件，可以推导出充分必要条件

$$\begin{aligned}\frac{\partial L}{\partial q_{i:m}}=&-\frac{2\eta kR_cL(\theta)I(\varepsilon,\theta)C_m^2}{\left(2\eta kL(\theta)I(\varepsilon,\theta)C_m^2+I(\varepsilon,\theta)t_{m\to i}^{\mathrm{com}}q_{i:m}\right)^2}\\&+\lambda_1\frac{1}{\eta kC_m}q_{i:m}=0\end{aligned} \tag{29}$$

$$\begin{aligned}\lambda_1y_1&=\lambda_1\left(\sum_{m\in\mathcal{M}_i}q_{i:m}f_{i:m}^*-W_i\right)\\&=\lambda_1\left(\sum_{m\in\mathcal{M}_i}\frac{1}{2\eta kC_m}q_{i:m}^2-W_i\right)=0\end{aligned} \tag{30}$$

$$\lambda_1\ge 0 \tag{31}$$

由条件式(31)可得

$$\lambda_1=\frac{2\eta^2k^2R_cL(\theta)I(\varepsilon,\theta)C_m^3}{q_{i:m}\left(2\eta kL(\theta)I(\varepsilon,\theta)C_m^2+I(\varepsilon,\theta)t_{m\to i}^{\mathrm{com}}q_{i:m}\right)^2}>0 \tag{32}$$

所以

$$\sum_{m\in\mathcal{M}_i}\frac{1}{2\eta kC_m}q_{i:m}^2-W_i=0 \tag{33}$$

表明解是存在的。当边缘服务器i给每一个终端设备m的$q_{i:m}$相同时，可解得

$$q_{i:m}=\sqrt{\frac{2\eta kC_m}{m}W_i} \tag{34}$$

这个特殊解的实际意义在于如果$q_{i:m}$不相同，那么时间会由算得最慢的那个终端设备决定，从而使得边缘服务器i的效用函数降低。只有当边缘服务器i将激励预算W_i全部使用，且分配下去的$q_{i:m}$相等时，边缘服务器i以及终端设备m这一簇的时间消耗T_i才能达到最小，从而使得边缘服务器i的效用函数达到最大值。

3.3 博弈执行过程

初始化阶段，云服务器将总预算V均分给每个边缘服务器，此时$W_1=W_2=\cdots=W_i=\dfrac{V}{i}$。

步骤1 边缘服务器i刚开始随意分配激励单价$q_{i:m}$，终端设备m会提供相应的算力，使得自己的效用函数$U_{i:m}$达到最大。此时边缘服务器i会发现它完成$I(\varepsilon,\theta)$次迭代的时间是受限于计算最慢的那个终端设备，故它会重新分配$q_{i:m}$，激励最慢的终端设备增加算力，从而减小T_i，增加自己的效用函数U_i。最终边缘服务器i会根据终端设备的个数，均分W_i，此时的$q_{i:m}$即为前面求出的特殊解式(34)。

步骤2 当每个边缘服务器和它连接的终端设备达到均衡点后，边缘服务器i得到了T_i和最终的效用函数U_i。此时每个边缘服务器i会将最终的T_i上报给云服务器。为了使得一次全局迭代的总时间达到最小，云服务器会根据每个边缘服务器的T_i大小值调整激励预算分配比例。

步骤3 得到重新分配的激励预算后，每个边缘服务器再重复步骤1，然后云服务器重复步骤2，直到每个边缘服务器的T_i值最终相等。整个系统达到稳定的状态，且完成一次全局迭代的时间达到最短。

根据上述过程，本文提出了全局模型训练的加速算法，如算法1所示。

4 仿真结果与分析

本文用MNIST数据集[15]来评估所提出的激励机制的性能。随机分配相同数量的10个种类的训练数据给每个终端设备，用随机梯度下降来训练本地模型，学习率为0.1。其他一些参数为：完成1次本地迭代计算任务需要的总的CPU转圈数$C_m=5\,000$，边缘服务器i下的终端设备m进行本地计算时的CPU频率$f_{i:m}\in[1,10^9]$ Hz，信道总带宽$B=10^6$ Hz，噪声功率$N_0=10^{-8}$ W，传输功率$u_m=200$ mW，终

端设备m传输的参数量$d_m=2\,500$ bit，终端设备m进行迭代的次数$L(\theta)=2$，边缘服务器需要迭代的次数$I(\varepsilon,\theta)=5$，芯片结构的系数$k=2\times10^{-28}$，系数$\eta=10^{18}$。

(1) 边缘服务器与终端设备的博弈迭代过程。为了评估这个过程，本文对一个博弈簇进行仿真实验，在激励预算足够的情况下，探究边缘服务器给出的激励单价对终端设备效用以及边缘服务器自身效用的影响，同时验证引入激励机制能够刺激终端设备投入更多的计算资源，减少本地计算时间。

图3显示的是边缘服务器给出的CPU激励单价对单个终端设备效用值的影响。可见，单个终端设备的效用值随着激励单价的增加逐渐上升并最终达到最大值稳定下来。这是因为随着激励单价q的增加，终端设备得到的奖励越来越多，导致其效用不断增加。当激励单价增加到某个值，如图中的$q=20$时，终端设备会将所有的计算资源都投入到本地模型训练任务中，此时它获得的奖励值达到最大，故效用值也达到最大。而如果q继续增加，由于该终端设备之前就已投入了全部的CPU频率资源，所以边缘服务器为了节省激励成本，实际上并不会继续增加真正给到该终端设备的激励单价，故该终端设备的效用值保持最大值不变。

图3虽然是显示的单个终端设备效用与激励单价的关系，但是该博弈簇中的所有终端设备都遵循这个关系，当每个终端设备效用值达到最大时，下层子博弈达到均衡点。

图4显示的一个边缘服务器给出的激励单价对其效用值的影响。可以看到，边缘服务器的效用值随着激励单价的增加而增加，最后达到最大值并稳定下来。这是因为增加激励单价q，能够刺激终端设备提供更多的算力到模型训练任务中来，从而减小时间T_i，增大了该边缘服务器对于一次全局迭代的时间贡献度。虽然增加激励单价会导致边缘服务器的激励支出增加，但是它能够从云服务器端获得更多的奖励，故其效用值最终增加。当激励单价增加到某个值，比如图中的$q=20$时，终端设备将全部算力投入计算，时间T_i达到最小，该边缘服务器效用值达到最大。同样地，之后激励单价如果继续增加，边缘服务器为了节约激励成本，实际上并不

算法1　基于主从博弈的可变激励训练加速算法

输入：$\mathcal{N}=\{i:i=1,2,\cdots,N\}$，$\mathcal{M}_i=\{m:m=1,2,\cdots,M\}$，计算任务Task，数据集$D$，$W_1=W_2=\cdots=W_i=\dfrac{V}{i}$。

输出：激励预算分配的均衡点$\mathcal{W}_i=\{W_i,i\in\mathcal{N}\}$，$\mathcal{Q}_{i:m}=\{q_{i:m},m\in\mathcal{M}_i\}$，终端设备$m$提供的算力均衡解$\mathcal{F}^*_{i:m}=\{f^*_{i:m},m\in\mathcal{M}_i\}$。

(1) **repeat**

(2) **for** i **in** N **do**

(3) 边缘服务器i分配激励单价$\{q_{i:m},m\in\mathcal{M}_i\}$，激励单价需满足条件$\sum\limits_{m\in\mathcal{M}_i}q_{i:m}f_{i:m}\le W_i$。终端设备$m$会提供相应的算力$\left\{f^*_{i:m}=\max\left\{\dfrac{q_{i:m}}{2\eta kC_m},f^{\max}_{i:m}\right\},m\in\mathcal{M}_i\right\}$，使得自己的效用$U_{i:m}$达到最大。

(4) **if** $t^{\mathrm{cmp}}_{i:m}\ne t^{\mathrm{cmp}}_{i:n}(m\ne n)$ **then**

(5) 边缘服务器i重新分配$q_{i:m}$，使得自己的效用函数U_i达到最大，同时得到时间T_i。

(6) **end for**

(7) **if** $T_i<T_j(i\ne j)$ **then**

(8) 云服务器重新分配V(减小W_i，增大W_j)。

(9) **Until** $T_i=T_j(i\ne j)$

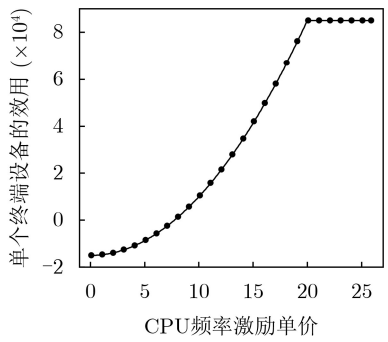

图 3 激励单价q与单个终端设备效用的关系

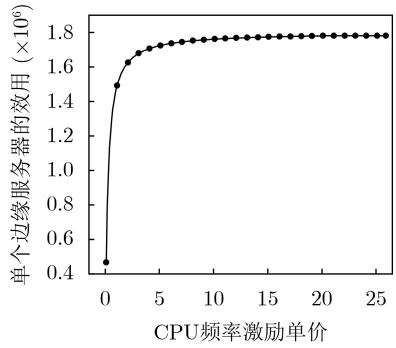

图 4 激励单价q与单个边缘服务器效用的关系

会给予已经完全激励了的终端设备更高的激励单价，故终端设备的效用值保持最大值不变。

当一个博弈簇中的边缘服务器的效用达到最大时，上层子博弈达到均衡点。当上下两层子博弈同时达到均衡点时，该博弈簇内的Stackelberg博弈达到纳什均衡点。

图5显示的是单个终端设备的本地计算时间与边缘服务器给出的激励单价的关系。可以看到，终端设备的本地计算时间随着激励单价q的增加逐渐减少，因为终端设备提供了更多的本地计算资源。当激励单价q增大到某个值，比如图中的$q=20$时，该终端设备贡献出全部的算力，本地计算时间达到最小。此后q继续增加，由于没有多余的计算资源，本地计算时间不会再继续减小，故保持最小值不变。

以上的仿真实验虽然是对一个博弈簇，在激励预算分配足够的情况下进行的实验，但是所有的博弈簇中的博弈迭代过程都遵循上述分析的结果。并且在激励预算不够的情况下，边缘服务器最终会根据其服务范围内的终端设备个数，均分激励预算，即满足前文解出的特殊解$q_{i:m}=\sqrt{\dfrac{2\eta kC_m}{m}}W_i$，从而博弈达到纳什均衡点，最小化本地计算时间$T_i$。

(2) 云服务器激励预算分配达到平衡点的过程。假设边缘服务器数量为2，边缘服务器1下连接的终端设备数量为1 000个，边缘服务器2下连接的数量为1 200个，云服务器能够分配的总激励预算$V=40\,000$。刚开始时分配情况为$W_1=W_2=V/2=20\,000$，然后云服务器会调整分配情况，一部分分给W_1，剩余部分全部分给W_2，即$W_2=V-W_1$。

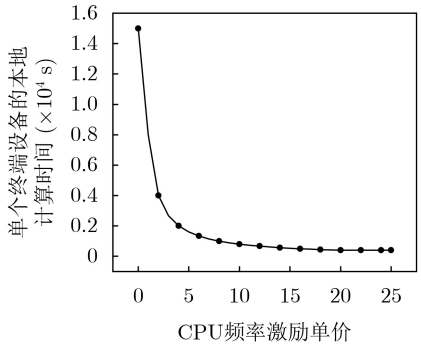

图 5 激励单价q与单个终端设备本地计算时间的关系

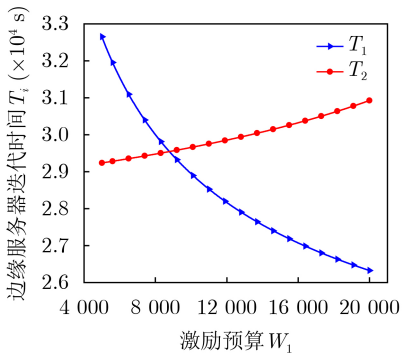

图 6 不同W_1的值与T_1, T_2的关系

从图6可以看到，刚开始均分时，由于边缘服务器1连接的终端设备数量更少，故$T_1<T_2$，全局训练时间取决于较大者。然后云服务器开始减小W_1，同时等量增大W_2。T_1随着W_1的减小而增大，因为边缘服务器1连接的终端设备得到的激励在减小，用于计算的CPU频率降低，本地计算时间增加，导致最终T_1增大。W_1减小，相应地W_2在增加，边缘服务器2获得更多的激励预算，故T_2逐渐减小。当云服务器分配$W_1=8\,700$，$W_2=31\,300$时，$T_1=T_2$，如图6中的两条曲线的交点处所示，此时全局训练时间达到最小，激励预算分配达到均衡点。

(3)将本文提出的基于主从博弈的可变激励训练加速算法与没有设计激励机制的算法[8]进行比较。边缘服务器个数设置为2个，每个边缘服务器下连接不同数量的终端设备。两个边缘服务器连接的终端设备总数分别设置为800个、1 000个、1 200个。

在没有设计激励机制的算法中，为了反映真实情况下终端设备的自私性，本文假设会有一定比例的终端设备不会完全贡献自己的资源甚至不愿意参与到联邦学习中。

从图7可以看到，在没有激励机制的情况下，全局模型的训练时延随着终端设备数的增加而减小，这是因为虽然设备存在自私性，但是随着设备的基数增加，总的算力资源是增加的。而在本文提出的激励机制算法中，由于激励预算是一定的，所以随着设备数量的增加，每个设备能够分到的激励量在减小，所以总的算力在减小，导致总时间增加。但是，在不同终端设备总数的情况下，与没有激励机制的算法相比，本文提出的带有激励机制的算法均能够有效地刺激终端设备积极参与到训练中

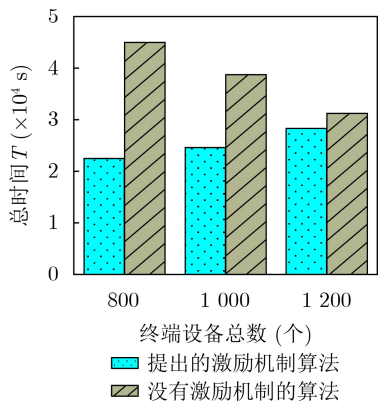

图 7 两种算法在不同终端设备总数下训练总时间的比较

来，提供更多的资源到训练过程中，从而有效降低了全局模型训练的总时延。

5　结束语

本文面向分层联邦学习框架，针对联邦学习中终端设备存在自私性而影响全局模型训练时间的问题，设计了基于主从博弈的激励机制。该机制通过对分层联邦学习模型训练过程进行建模，利用可变激励训练加速算法，不断调整分配给边缘服务器的激励预算，使得边缘服务器和终端设备达到纳什均衡点的同时，最小化1次全局模型训练时间。仿真结果表明，本文所提算法能够有效地降低终端设备自私性带来的影响，优化分层联邦学习全局模型的训练时延。

参 考 文 献

[1] MCMAHAN B and RAMAGE D. Federated learning: Collaborative machine learning without centralized training data[EB/OL]. https://starrymind.tistory.com/180, 2017.

[2] NIKNAM S, DHILLON H S, and REED J H. Federated learning for wireless communications: Motivation, opportunities, and challenges[J]. *IEEE Communications Magazine*, 2020, 58(6): 46–51. doi: 10.1109/MCOM.001.1900461.

[3] LIU Yi, PENG Jialiang, KANG Jiawen, *et al*. A secure federated learning framework for 5G networks[J]. *IEEE Wireless Communications*, 2020, 27(4): 24–31. doi: 10.1109/MWC.01.1900525.

[4] McMAHAN B, MOORE E, RAMAGE D, *et al*. Communication-efficient learning of deep networks from decentralized data[C]. The 20th International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, USA, 2017: 1273–1282.

[5] LI Tian, SAHU A K, ZAHEER M, *et al*. Federated optimization in heterogeneous networks[C]. Machine Learning and Systems 2020, Austin, USA, 2020, 2: 429–450.

[6] MILLS J, HU Jia, and MIN Geyong. Communication-efficient federated learning for wireless edge intelligence in IoT[J]. *IEEE Internet of Things Journal*, 2020, 7(7): 5986–5994. doi: 10.1109/JIOT.2019.2956615.

[7] LI Tian, SANJABI M, BEIRAMI A, *et al*. Fair resource allocation in federated learning[C]. The 8th International Conference on Learning Representations (ICLR), Addis Ababa, Ethiopia, 2019: 1–27.

[8] LIU Lumin, ZHANG Jun, SONG S H, *et al*. Client-edge-cloud hierarchical federated learning[C]. 2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 2020: 1–6. doi: 10.1109/ICC40277.2020.9148862.

[9] ABAD M S H, OZFATURA E, GUNDUZ D, *et al*. Hierarchical federated learning ACROSS heterogeneous cellular networks[C]. 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 2020: 8866–8870. doi: 10.1109/ICASSP40776.2020.9054634.

[10] SUN Haifeng, LI Shiqi, YU F R, *et al*. Toward communication-efficient federated learning in the internet of things with edge computing[J]. *IEEE Internet of Things Journal*, 2020, 7(11): 11053–11067. doi: 10.1109/JIOT.2020.2994596.

[11] SHI Yuanming, YANG Kai, JIANG Tao, *et al*. Communication-efficient edge AI: Algorithms and systems[J]. *IEEE Communications Surveys & Tutorials*, 2020, 22(4): 2167–2191. doi: 10.1109/COMST.2020.3007787.

[12] LUO Siqi, CHEN Xu, WU Qiong, *et al*. HFEL: Joint edge association and resource allocation for cost-efficient hierarchical federated edge learning[J]. *IEEE Transactions on Wireless Communications*, 2020, 19(10): 6535–6548. doi: 10.1109/TWC.2020.3003744.

[13] KHAN L U, RAJ PANDEY S, TRAN N H, *et al*. Federated learning for edge networks: Resource optimization and incentive mechanism[J]. *IEEE Communications Magazine*, 2020, 58(10): 88–93. doi: 10.1109/MCOM.001.1900649.

[14] TRAN N H, BAO Wei, ZOMAYA A, *et al*. Federated learning over wireless networks: Optimization model design and analysis[C]. IEEE INFOCOM 2019-IEEE Conference on Computer Communications, Paris, France, 2019: 1387–1395. doi: 10.1109/INFOCOM.2019.8737464.

[15] LECUN Y, BOTTOU L, BENGIO Y, *et al*. Gradient-based learning applied to document recognition[J]. *Proceedings of the IEEE*, 1998, 86(11): 2278–2324. doi: 10.1109/5.726791.

贾云健：男，博士，教授，研究方向为新一代移动通信网络、网络内生安全.

黄　宇：男，硕士生，研究方向为联邦学习.

梁　靓：女，博士，副教授，研究方向为移动通信网络、可信网络.

万杨亮：女，硕士，工程师，研究方向为计算机网络与无线通信.

周继华：男，博士，研究员，研究方向为无线通信.

责任编辑：马秀强