高技术通讯 2023 年 第33 卷 第12 期:1265-1275

doi:10.3772/j.issn.1002-0470.2023.12.004

基于联邦元学习的安全移动边缘计算卸载框架①

杨仕成②* 陈保罗** 陈铁明** 黄 亮③**

(*浙江工业大学信息工程学院 杭州 310023)

(**浙江工业大学计算机科学与技术学院 杭州 310023)

摘 要 移动边缘计算(MEC)技术通过卸载部分计算任务到边缘服务器,可将第5代网络(5G)、云计算、大数据和人工智能等技术延伸到物联网终端。针对如何高效卸载计算任务和保障边缘数据隐私安全2个关键问题,在综述计算卸载性能优化研究基础上,本文提出了一种融合联邦学习和元学习的计算卸载应用框架,通过对计算任务的计算卸载以及计算资源的联合优化,从而实现系统加权时延和最小。在不泄露用户数据隐私的前提下,联合多个边缘服务器共同训练一个全局模型,并实现边缘服务器个性化计算卸载应用。在新的计算任务场景下,全局模型的网络参数仅用少量训练样本就能快速收敛。实验测试结果表明,本文提出的基于联邦元学习的计算卸载框架可适应未来边缘计算应用的隐私安全需求。

关键词 移动边缘计算(MEC);隐私保护;联邦学习;元学习

0 引 言

移动互联网技术的快速发展,推动了越来越多的时延敏感型和资源密集型应用和服务出现。然而,这些移动应用对无线设备的计算和存储资源需求大幅增加。移动边缘计算技术(mobile edge computing, MEC)为无线设备提供额外的计算资源,为该问题提供了一种很好的解决方案[1]。新兴的移动边缘计算技术是云计算、大数据、人工智能等技术在物联网终端应用的重要基础,推进“万物互联”到“万物智联”。自《“十四五”国家信息化规划》提出加快第5代(5G)网络规模化部署、前瞻性的布局第6代(6G)网络技术储备以来,国家进一步大力推进5G基建、大数据中心、人工智能、工业互联网等领域“新基建”计划。当前,电信运营商、网络设备供应商和云服务商等巨头企业都在布局迎接移动边缘计算带来的新机遇和挑战。因此,移动边缘计算也成为学术界近年来的研究热点。

1 相关研究

移动边缘计算技术通过将计算服务器配置在靠近物联网终端的网络边缘[2],可有效缩短计算任务传输时延并提升物联网终端的计算能力,深化物联网连接和计算的融合。简单而言,在移动边缘计算框架下,可使终端设备按需将计算任务卸载上传至边缘服务器,由边缘服务器完成计算任务并返回计算结果,以降低计算时延和设备能耗,提升终端设备的数据处理能力。如图1所示,移动边缘计算在物联网、端到端组网、无人机组网、车联网等领域有着广泛的应用场景[3]。

以计算卸载作为移动边缘计算的关键技术也引起了人们的广泛研究[4-5]。计算卸载问题是一个混

① 国家自然科学基金面上项目(62072410),浙江省自然科学基金(LD22F020002)和浙江省重点研发计划(2021C01117)项目资助。

② 男,1997年生,硕士生;研究方向:移动边缘计算;E-mail: scyang@zjut.edu.cn。

③ 通信作者,E-mail: lianghuang@zjut.edu.cn。

(收稿日期:2022-04-22)

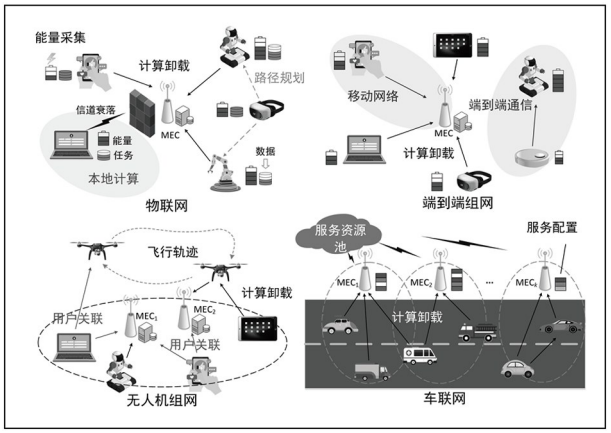


图 1 移动边缘计算网络中计算卸载应用场景[3]

合整数规划问题,并且一般是一个非确定多项式难题(non-deterministic polynomial hard, NP-hard),难以实时生成决策。一种可行的方案是利用分支定界法(branch-and-bound algorithm,BBA)[6]或动态规划法(dynamic programming,DP)[7]进行求解,然而,这些方法存在"维数灾难"(curse of dimensionality)问题,计算复杂度较高。另一种方案是将整型的 0/1 决策做连续线性松弛[8]或者用二次约束来近似整数约束并做半正定松弛[9],以比迭代搜索算法低的复杂度求得近似解方案,但这种方法牺牲了求解质量,通常得到的计算卸载决策都不是最优决策。此外,考虑到移动边缘计算网络的复杂性与动态性,通常可采用基于马尔可夫决策过程[10-11]或李雅普诺夫优化方法[12]设计动态计算卸载策略,并优化相关系统资源分配,最小化能耗、计算时延和网络效用。然而,上述解决方案都依赖于专家知识和精确的数学模型。因此,如何设计一种低复杂度的算法且能适用于时变环境的移动边缘计算网络仍是一个挑战。

深度学习利用深度神经网络[13]从数据样本学习,最终生成状态空间到动作空间的最优映射,擅长处理大状态空间[14]和动作空间[15],被广泛应用于无线通信网络领域。近年来,将深度学习应用于移动边缘计算卸载的研究逐渐增多。监督学习将计算卸载问题建模为多标签分类问题,通过离线训练和线上部署深度神经网络,提升卸载决策的响应速度。监督学习算法需要预生成大量的训练数据,常用遍历搜索[16]或者传统分析优化方法[17]获得特定网络场景的最优决策。然而,当网络场景改变时,需要重新生成训练数据并训练深度神经网络,该方法不适用于动态的网络场景。深度强化学习可通过搜索计算卸载动作空间并自主学习实现最优决策。当前,基于深度 Q 学习的计算卸载研究[18-21]较为广泛,其将状态/动作空间离散化并通过在线学习最优化计算卸载决策及系统资源优化。随着大型神经网络模型的兴起,模型对系统内存的需求也在逐步增高,分布式架构[22]通过并行的数据处理来减少系统内存开销,这一方法也引起了研究人员的广泛关注。联邦学习利用其分布式网络架构联合多个边缘服务器端进行数据训练,在提升模型泛化性的同时,最大限度地减少用户隐私泄露,进一步提高了计算卸载的安全性与有效性[23]。其局限性是这些方法大多数是基于静态交互的移动边缘计算环境,一旦移动边缘计算场景发生变化,在收敛到新的场景时,难以收集到足够的训练样本。针对计算任务场景动态变化以及上述深度学习方法网络参数可移植差等问题,基于元学习[24]的计算卸载方法可以有效克服这一问题,面对一个新的计算任务场景,基于元学习的计算卸载方法通过找到一个全局模型,使模型执行一个或几个梯度下降步骤来快速收敛于该场景[25]。然而,现有算法考虑的是集中式移动边缘计算网络模型,存在隐私泄露的风险,因此,如何设计一种算法能在保护用户隐私的前提下,适用于时变环境的移动边缘计算仍是一个挑战。

为解决上述问题,本文提出一种基于联邦元学习的计算卸载(federated meta-learning based offloading,FEMO)框架。一方面联邦学习可以有效解决移动边缘计算的数据隐私和数据孤岛问题,并且基于联邦学习的计算卸载还能有效降低通信带宽需求,减轻了远程云服务器的存储和计算负载,并降低模型更新相应延时[26]。另一方面,为了满足不同移动端的服务质量(quality of service, QoS)需求,本算法融合了模型无关元学习(model-agnostic meta-learning, MAML)思想,在动态变化的边缘计算网络中,仍能取得较高的卸载效率。具体来说,本文的贡献点如下。

(1)将整个移动边缘计算网络的系统效用模型

化为所有无线设备的任务完成时延的加权和,为了使系统效用最小化,本文提出了一个针对移动边缘计算网络的联合计算卸载与计算资源分配问题。该问题联合优化了每个边缘服务上的无线设备的卸载决策和每个设备上计算资源分配。

(2)提出一种基于联邦元学习的计算卸载框架,通过联合多个边缘服务器上的数据共同训练学习,得到一个泛化能力更强的神经网络模型。并且进一步考虑到边缘服务器上个性化计算卸载应用,为隐私保护的个性化计算卸载提供一种新的思路。

(3)数值结果进一步验证了所提算法的有效性,针对一个新的计算任务场景,基于联邦元学习的计算卸载框架仅需少量的微调训练步数(少于 10 步),就能达到 0.995 以上的性能。

2 系统模型和问题建模

如图 2 所示,本文考虑一个移动边缘计算网络包含 K 台边缘服务器和一个云端服务器,其中每个边缘服务器都含有 N 台有计算任务的无线设备(wireless device,WD)。N 台无线设备用集合 $N=\{1,2,\cdots,N\}$ 表示,K 台边缘服务器用集合 $K=\{1,2,\cdots,K\}$ 表示,每个无线设备决定将每个任务卸载到对应的边缘服务器或者本地处理,即为二进制卸载策略。

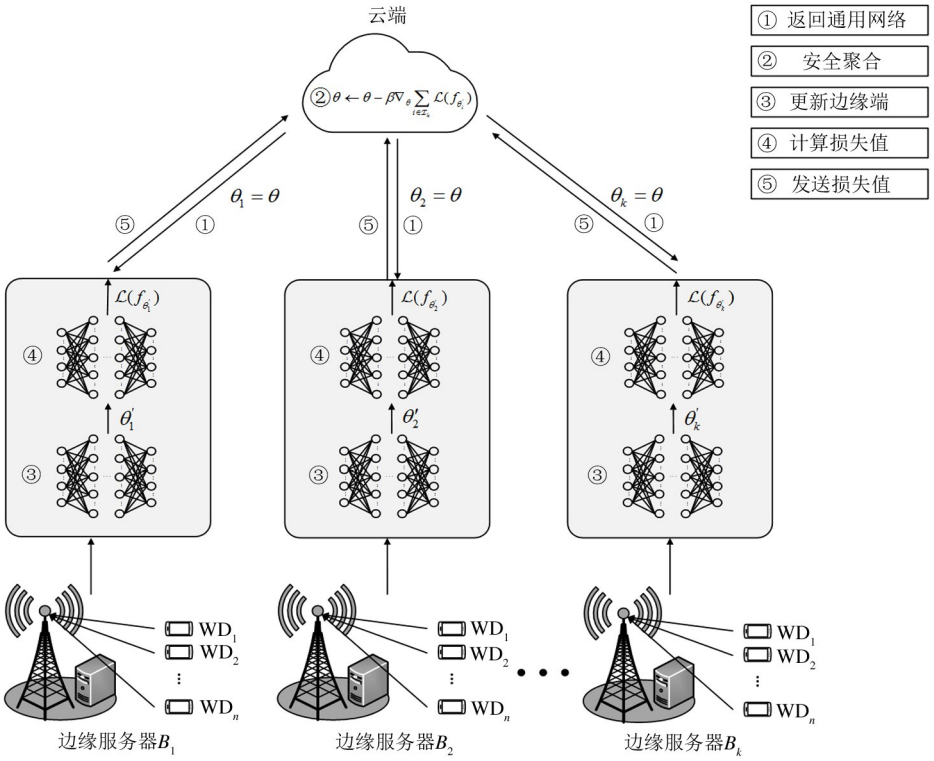


图 2 基于联邦学习的隐私保护计算卸载算法应用框架

每个边缘服务器的用户数据仅保存在该服务器。本文主要研究了边缘计算系统的计算资源配置优化问题,并通过用户的权重优先级系数来构造动态的计算任务场景。本文考虑了由于无线设备与服务器之间的通信而导致的延迟优化问题。

设置一个元组 $(\alpha_n,\beta_n,\gamma_n)$ 来表示 WD$_n$ 的计算任务,其中 $n\in N$,α_n、β_n、γ_n 分别表示上行传输数据大小、服务器返回数据大小、完成该计算任务所需要的中央处理器(central processing unit,CPU)周期数。当 WD$_n$ 的任务分配给边缘服务器时,将 WD$_n$ 与边缘服务器之间的上行和下行链路传输速率量化为

$$C_n=B_n\log_2\left(1+\frac{P_n h_n(t)}{\omega_0}\right) \tag{1}$$

其中,B_n 表示上行和下行传输链路的带宽,ω_0 表示白噪声功率,P_n 表示 WD$_n$ 将其计算任务卸载到边缘服务器的发射功率以及边缘服务器将计算结果回传的发射功率,$h_n(t)$ 表示对应的信道增益并假设该值在传输过程中保持不变。那么 WD$_n$ 的总通信

时延可以推导为上行和下行传输时延之和,并假设上行和下行传输速率相等,则通信时延 T_n^{Comm} 为

$$T_n^{\text{Comm}} = \frac{\alpha_n}{C_n} + \frac{\beta_n}{C_n} \tag{2}$$

对于计算时延,用 f_k 表示边缘服务器每秒能执行的 CPU 周期数(cycle/s),其中 $k \in K$, 并在本文中假设各边缘服务器计算资源相同;用 f_0 表示无线设备每秒能执行的 CPU 周期数,且 f_0 远小于 f_k, 这是由于边缘服务器上的计算资源更丰富。当 2 个或多个任务被卸载到同一台边缘服务器时,边缘服务器的计算资源被所有任务共享,计算资源的分配策略表示为 $\boldsymbol{f}_t = \{f_n(t) \mid n \in N_t\}$, 其中 $f_n(t)$ 表示边缘服务器分配给 WD_n 的计算资源, $N_t = \{n \in N \mid a_n(t) = 1\}$ 是指在时间帧 t, 计算任务被卸载到边缘服务器的所有无线设备集合。则 WD_n 在边缘服务器上处理计算任务的延迟 T_n^{Comp} 表示为

$$T_n^{\text{Comp}} = \frac{\gamma_n}{f_n(t)},\ n \in N_t \tag{3}$$

相应地, WD_n 在本地计算任务的计算延迟 T_n^L 可表示为

$$T_n^L = \frac{\gamma_n}{f_0} \tag{4}$$

相应地, WD_n 在边缘服务器执行计算任务所需要的时延 T_n^C 可表示为

$$T_n^C = T_n^{\text{Comm}} + T_n^{\text{Comp}} \tag{5}$$

综上所述, WD_n 完成其计算任务所需的计算任务总传输的计算时延 T_n 可表示为

$$T_n = (T_n^{\text{Comm}} + T_n^{\text{Comp}})a_n(t) + T_n^L(1 - a_n(t)) \tag{6}$$

本文的研究目标是最小化边缘计算系统的加权时延和,问题建模如下:

$$Q^*(\boldsymbol{a}_t, \boldsymbol{f}_t, \boldsymbol{h}_t, \boldsymbol{w}_t) = \min_{\boldsymbol{a}_t, \boldsymbol{f}_t} \sum_{n \in N} T_n(t) w_n(t) \tag{7}$$

$$\text{约束:}\ a_n(t) = \{0, 1\},\ \forall n \in N \tag{8}$$

$$\sum_{n \in N_t} f_n(t) \leqslant f_k,\ \forall k \in K \tag{9}$$

$$f_n(t) > 0, \forall n \in N_t \tag{10}$$

上述问题为混合整数规划问题,难以直接求解。本文将上述问题进行了分层处理,其中底层问题利用神经网络学习来生成最优的卸载决策,一旦得到最优的卸载决策,将其代入到顶层问题中求解最优的计算资源分配,最终使得整个边缘计算系统的加权时延和最小。

将无线信道增益 $\boldsymbol{h}_t$ 与最优卸载决策 $\boldsymbol{a}_t^*$ 之间的映射关系用 π 表示:

$$\pi: \boldsymbol{h}_t \mapsto \boldsymbol{a}_t^* \tag{11}$$

在得到最优卸载决策 $\boldsymbol{a}_t^*$ 之后,可以进一步求解资源分配问题 $\boldsymbol{f}_t^*$:

$$\boldsymbol{f}_t^* = \arg\min_{f_t} Q(\boldsymbol{f}_t \mid \boldsymbol{a}_t^*, \boldsymbol{h}_t, \boldsymbol{w}_t) \tag{12}$$

$$\text{约束:}\ \sum_{n \in N_t} f_n(t) \leqslant f_k \tag{13}$$

$$f_n(t) > 0, \forall n \in N_t \tag{14}$$

显然,上述优化问题是一个凸优化问题,本文采用了 KKT(Karush-Kuhn-Tucker)方法来求解该问题,其拉格朗日函数 Ω 可表示为

$$\Omega = \sum_{n \in N} (T_n^{\text{Comm}} + \frac{\gamma_n}{f_n(t)}) w_n(t) + \nu(\sum_{n \in N_t} f_n(t) - f_k) \tag{15}$$

其中, ν 为拉格朗日系数。通过对该拉格朗日函数中的计算资源 $f_n(t)$ 和拉格朗日系数 ν 求偏导,令其导数为零,解方程组即可得到约束条件下函数的最优解。对该函数进行求导,可得如下表达式:

$$\frac{\partial L}{\partial f_n(t)} = -\frac{\gamma_n w_n(t)}{f_n^2(t)} + \nu,\ \forall n \in N_t \tag{16}$$

当该导函数取到 0 时,可通过式(17)来计算最优的资源分配结果。

$$f_n^*(t) = \sqrt{\frac{\gamma_n w_n(t)}{\nu^*}}, \forall n \in N_t \tag{17}$$

由于 $\gamma_n w_n(t)$ 为大于 0 的实数,所以 ν^* 大于 0,因此,最优的资源分配方案满足式(18)。

$$\sum_{n \in N_t} f_n^*(t) = f_k, \forall k \in K \tag{18}$$

将式(17)代入到式(18)中,可获得拉格朗日因子的最优值:

$$\nu^* = \left(\frac{1}{f_n(t)} \sum_{n \in N_t} \sqrt{\gamma_n w_n(t)}\right)^2 \tag{19}$$

最后,将式(19)代入式(17),则边缘服务器的计算资源分配问题可获得如式(20)所示的最优解。

$$f_n^*(t) = \frac{f_k \sqrt{\gamma_n w_n(t)}}{\sum_{n \in N_t} \sqrt{\gamma_n w_n(t)}}, \forall n \in N_t, \forall k \in K \tag{20}$$

3 基于联邦元学习的算法设计

以本文研究的边缘计算系统为例,考虑到用户的隐私安全问题以及动态变化的计算任务场景,为了使得每个 WD_n 获得更高质量的个性化模型,本节提出了一种基于联邦元学习的计算卸载算法。具体来说,根据各个边缘服务器本地数据共同训练一个全局模型,在个性化微调测试中,模型仅需少量训练样本就能快速适应该场景。

3.1 训练全局模型

基于联邦元学习的隐私保护计算卸载算法体系结构如图 2 所示,每个边缘服务器都含有等量计算任务场景 $\{\Psi_i^k \mid i \in K, k \in K\}$,其中 $I = \{1,2,\cdots,L\}$,且每个场景中都含有 $M = \{1,2,\cdots,M\}$ 个数据样本,每个数据样本又由信道增益集合和最优卸载决策集合组合而成。整体的训练流程包含 5 个步骤。

(1)边缘服务器 $B_k, k \in \{1,2,\cdots,K\}$,从云端上下载云端网络模型参数,复制并更新边缘端网络模型,即模型参数 $\theta_k = \theta$。

(2)对于每个边缘服务器 B_k 并行训练各自本地的模型参数 θ_k。首先,各边缘服务器从本地任务场景 $\{\Psi_i^k \mid i \in K, k \in K\}$ 中随机抽取一个计算任务场景来迭代训练各自的神经网络;以单个边缘服务器为例,针对单一的计算任务场景 $\Psi_i^{'k}$,从该场景中不重复的随机抽取一批数据样本 M_b,其中 $D_i^k = \{(\boldsymbol{h}_m, \boldsymbol{a}_m)_i^k \mid m \in M_b\}$;并将该样本表示为 $M_b \subset M$,其中 $\boldsymbol{h}_m$ 表示时变无线信道增益,$\boldsymbol{a}_m$ 表示卸载策略。随后,通过最小化均方误差来计算与预测值与真实标签之间的损失值,其计算公式如式(21)所示,在得到相应的误差损失后,同样采用式(22)更新网络参数 θ_k。

$$L(f_{\theta_k}) = \sum_{m \in M_b} \| f_{\theta_k}(\boldsymbol{h}_m) - \boldsymbol{a}_m \|_2^2 \tag{21}$$

$$\theta_k^{'} = \theta_k - \lambda_1 \nabla_{\theta_k} L(f_{\theta_k}) \tag{22}$$

(3)当边缘服务器上的神经网络参数更新后,基于更新后的网络参数 $\theta_k^{'}$,各边缘服务器在各自的场景 $\Psi_i^{'k}$ 中重新抽取一批样本,具体可表示为 $D_i^{'k} = \{(\boldsymbol{h}_m, \boldsymbol{a}_m)_i^k \mid m \in M \backslash \{M_b\}\}$,并计算该样本下的损失 $L(f_{\theta_k^{'}})$。

(4)各边缘服务器将得到的损失值 $L(f_{\theta_k^{'}})$ 上传至云端服务器。

(5)云端服务器通过聚合所有边缘服务上的训练损失 $\{L(f_{\theta_k^{'}}) \mid k \in K\}$,进一步更新全局模型的参数 θ。全局网络模型通过累加损失进行梯度运算,并更新其参数如下:

$$\theta - \lambda_2 \nabla_\theta \sum_{k \in K} L(f_{\theta_k^{'}}) \rightarrow \theta \tag{23}$$

其中,λ_2 是超参数。

重复上述操作,直至全局网络模型收敛。基于各边缘服务器共同训练得到的神经网络模型具有更强的泛化性,面对新的计算任务场景,能凭借少量的训练数据做到快速适应。FEMO 的算法伪代码如算法 1 所示。

算法 1 FEMO 算法训练流程

输入:边缘服务器 $\{B_k \mid k \in K\}$ 和相应的计算任务场景 $\{\Psi_i^k \mid i \in K, k \in K\}$,超参数 λ_1、λ_2

输出:训练完成的全局网络模型

1. 云端网络随机初始化全局神经网络模型 θ
2. 当边缘服务器训练集 $I = \{1,2,\cdots,L\}$ 未使用完时,执行步骤 3,否则本轮训练结束
3. 各边缘服务器 B_k 以并行的方式执行步骤 4 至步骤 6
4. 从 I 中不重复地随机抽取 Ψ_i^k,并创建本地模型,其结构和全局模型一致,参数记为 θ_k
5. 将全局神经网络参数赋值给各本地模型,即 $\theta_k = \theta$
6. 从场景 Ψ_i^k 中随机抽取数据 $D_i^k = \{(\boldsymbol{h}_m, \boldsymbol{a}_m)_i^k \mid m \in M_b\}$
7. 利用式(21)来计算预测值与真实标签之间的损失值:$L(f_{\theta_k}) = \sum_{m \in M_b} \| f_{\theta_k}(\boldsymbol{h}_m) - \boldsymbol{a}_m \|_2^2$

3.2 个性化场景下执行微调训练

在使用基于联邦元学习的计算卸载策略算法时,每个边缘服务器不止于简单复制运行云端共享模型,可以进一步个性化微调网络模型。如图 3 所示,首先边缘服务器 B_k 从云端下载共享模型,并创建本地模型令 $\theta_k = \theta, \forall k \in K$。根据新的计算任务场景 $\{\Psi_i^{k'} \mid i \in I' \backslash \{I\}\}$,利用该场景下的数据样本进行微调训练,使得本地模型能够快速适应该场景,满足不同边缘服务器的个性化需求。其中,对于本地模型参数的更新,仍然采用梯度下降更新方式,具

体如式(24)所示。

$$\theta_{k'}^{'} = \theta_{k'} - \lambda_3 \nabla_{\theta_{k'}} L(f_{\theta_{k'}}) \tag{24}$$

最后,可以基于微调过的本地模型生成卸载决策,并进一步利用式(20)得到相应的计算资源分配,最终得到相应的 Q 值。需要说明的是,为了将 FEMO 算法部署在边缘计算网络,当全局模型训练时的损失不再变化则认为该模型收敛。而一旦模型收敛后,其参数被固定不再变化。一方面,本文更看重部署后的长远效益,当模型收敛无需动态维护该模型,大大节省了计算资源;另一方面 FEMO 是一种分布式算法,通过客户端与云端之间模型参数的交互进行训练,通过在云端聚合各客户端的模型参数来进一步提升模型的性能。另外,在客户端与云端之间只涉及到模型参数的交互,而不含原始数据的信息,降低了通信成本,也保护了各客户端数据的隐私泄露。

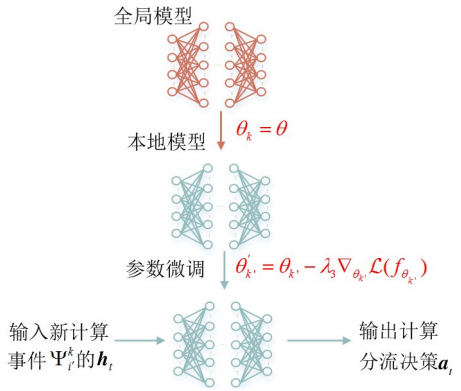


图 3　新场景下执行微调训练示意图

4　算法仿真设置

4.1　算法参数设置

不失一般性,本文考虑了一个 $N = 10$ 个无线设备和 $K = 5$ 个边缘服务器所组成的边缘计算系统,且每台无线设备的可选权重系数被设置为 $\mathbb{W} = \{1.0, 1.5\}$,本实验的数据集包含了 2^{10} 种不同计算任务场景。其中,80% 的场景将被作为训练集用于训练全局网络模型,每个边缘服务器上含有等量的计算任务场景,其中 $L = 164$。而余下 20% 的场景将用来测试 FEMO 算法的性能。本文的信道增益可利用与空间相关的路径衰落模型生成,其具体的表达式为 $H_{[\mathrm{dB}]} = 103.8 + 20.9 \log_{10} d_{[\mathrm{km}]}$ [27]。

一旦得到无线信道增益 $\boldsymbol{h}_t$ 以及相应的计算任务权重 $\boldsymbol{w}_t$ 后,可以通过枚举法求解最优卸载决策 $\boldsymbol{a}_t^*$。$\alpha_n = 0.5$ MB,$\beta_n = 0.28 \times \alpha_n$ 分别表示上行、下行数据大小;$\gamma_n = 330$ cycles/byte $\times \alpha_n$ [28] 为完成该计算任务需要的 CPU 周期数。本节以 Gzip 应用为计算任务类型,并设置每台无线设备的处理速率为 $f_0 = 2.84 \times 10^9$ cycle/s、边缘服务器的处理速率为 $f_e = 100 \times 10^9$ cycle/s,上下行的传输信道带宽设置为 $B_n = 10$ MHz。

本节使用的网络结构均为全连接神经网络,且神经网络的结构由 1 个输入层、2 个隐藏层和 1 个输出层组成,并且神经元参数分别为 N、120、80、N,每层的非线性函数为 ReLU。其他参数为:$\lambda_1 = 0.010$,$\lambda_2 = 0.001$,$\lambda_3 = 0.010$,$|M_b| = 10$。为了验证本文提出算法的有效性及其性能,本节的实验软件平台为基于 CPU 的 PyTorch 深度学习框架,编程语言为 Python。

4.2　算法性能评价指标

为了更直观地展现本文所提出 FEMO 算法的收敛性能,将神经网络的预测输出对应的系统加权时延和作了归一化,如式(25)所示。

$$\hat{Q} = \frac{Q^*(\boldsymbol{h}, \boldsymbol{w}, \boldsymbol{a}^*)}{Q(\boldsymbol{h}, \boldsymbol{w}, \boldsymbol{a})} \tag{25}$$

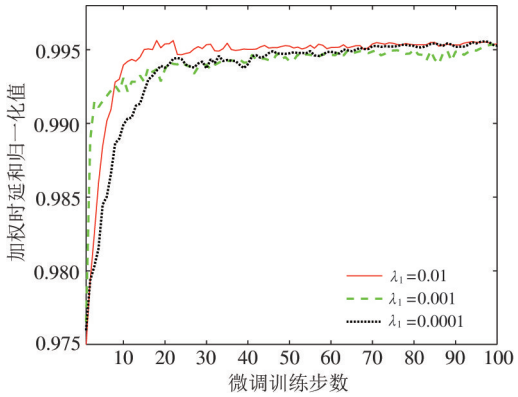
$Q^*(\boldsymbol{h}, \boldsymbol{w}, \boldsymbol{a}^*)$ 是由枚举法获得的真实最优卸载决策对应的系统加权时延和,而 $Q(\boldsymbol{h}, \boldsymbol{w}, \boldsymbol{a})$ 为 FEMO 算法获得的预测卸载方案对应的系统加权时延和。在预测值远离最优值时,分母大于分子,整体的比值 $\hat{Q}$ 小于 1;而当其越接近于真实值,$\hat{Q}$ 越接近于 1。

5　算法结果分析

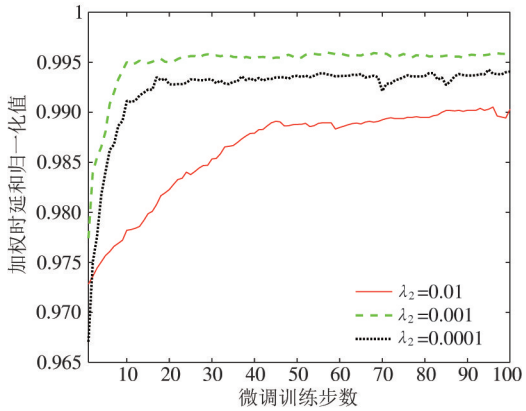
5.1　算法收敛性分析

如图 4 所示,本文研究了不同学习率下 FEMO 算法,在改变学习率 λ_1 参数时,其他训练率参数保持不变。如图 4(a)所示,学习率越大,模型的收敛速度越快,因此,将学习率 λ_1 设置为 0.010 0。又如图 4(b)所示,学习率 λ_2 设置范围在0.010 0到0.000 1之间,而学习率过大时,模型会陷入局部最优,所以当 λ_2 设置为 0.001 0 时,模型的收敛效果更好。为

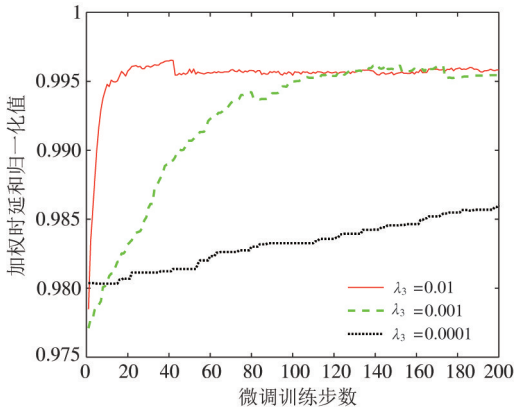
了选择一个合适的学习率参数,如图 4(c)所示,将 λ_3 设置为 0.010 0。



(a)学习率 λ_1



(b)学习率 λ_2



(c)学习率 λ_3

图 4　不同学习率对算法收敛性能的影响

5.2　算法有效性分析

如图 5 所示,从测试集中随机抽取了一个新的计算任务场景来测试 FEMO 算法在新场景中的表现。在评估之前,已经利用所有边缘端下的计算任务场景下来训练云端模型和单一模型数据集下训练得到的本地模型,实验展示了基于联邦元学习的计算卸载算法在边缘服务器的测试性能。

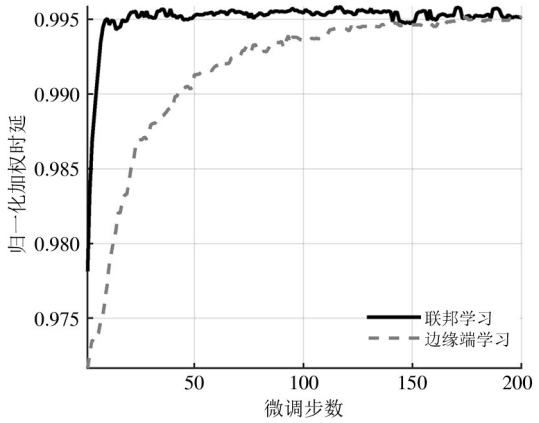


图 5　FEMO 算法在新场景下的测试表现

实验中基于联邦元学习算法和对比算法都采用相同的神经网络模型,即神经网络模型只学习了 0-1 计算卸载动作,但对比算法仅使用单一边缘服务器上的训练数据。以下实验结果均为对应测试场景下 100 个数据样本的均值。训练收敛后,2 种算法都获得了不错的初始化性能,分别达到理想卸载方案的 0.978 和 0.972。在相同的个性化微调测试中,针对一个新移动边缘计算应用场景,FEMO 算法仅需少量的微调训练步数(少于 10 步),就能达到0.995 以上的性能,而单一边缘服务器的数据集下训练的模型需要更多的迭代步数(200 步)才能收敛。因此,FEMO 算法在保护各边缘服务器数据隐私的前提下,当遇到一个新的任务场景时,仅需少量的训练样本就能快速适应该场景。

为了进一步研究在小样本下 FEMO 算法的有效性,针对一个全新的场景,本文将 FEMO 算法与其他 2 种具有代表性的基准方法进行了比较。

(1)边缘端学习。通过利用单一边缘服务器上的数据集训练本地模型,与联邦学习的过程类似,边缘端学习也是根据不同的计算任务场景创建相应的与本地模型网络结构相同的事件模型,通过汇总不同计算任务场景下的损失值来更新整个本地模型。

(2)随机初始化学习。神经网络在针对新的场景前并没有训练过,其网络参数是随机初始化的,在 PyTorch 框架中全连接层的参数初始化满足均方分

布。其网络结构和其他 2 种算法使用的模型结构完全一致。

在实验过程中把 3 种算法的学习率以及批处理大小等实验参数保持一致。如图 6 所示，由于训练数据的匮乏，基准方法的性能是有限的。特别地，对于基于 FEMO 算法得到的初始化参数在仅有 1 个训练样本下能通过微调达到 0.984，而基于单一边缘服务器数据集训练的本地模型虽然最终能获得近似 FEMO 的性能，但由图 4 可知，需要更多的迭代步数。使用随机初始化的神经网络模型，一开始的初始化参数就远小于其他 2 种方法的初始化参数，则需要更多的样本数量以及更多的微调步数才能达到相似的性能。这表明 FEMO 算法可以快速从历史的场景中学到先验知识，并迅速将其转移至新的计算任务场景。

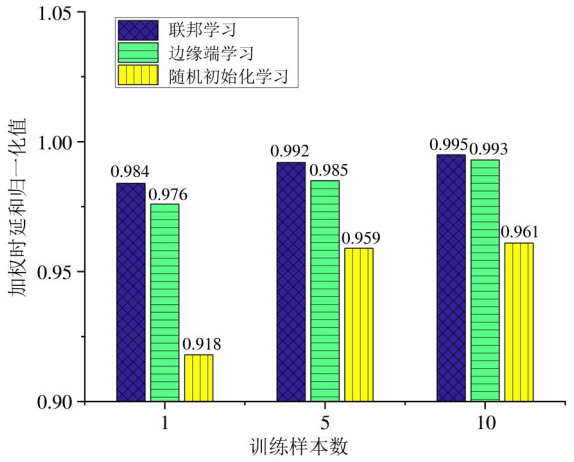


图 6　几种少量样本下基准方法的加权时延和归一化值

为了进一步评估所提算法的有效性与优越性，本文将所提算法与以下其他计算卸载算法进行比较。

(1) DSLO[29]：基于监督学习的计算卸载算法，通过标签样本训练的神经网络，预测卸载决策。

(2) DROO[30]：基于强化学习的计算卸载算法，并利用经验回放技术指导神经网络学习。该算法通过将预测的卸载决策量化成多个二进制决策，将决策代入目标函数计算系统加权时延和，从中找出使时延取最小值的分流决策。

(3) 基于线性规划松弛的计算卸载[31] (linear relaxation, LR)：所有的二进制卸载策略松弛到0 ~ 1之间的一个实数，用 $\hat{a}_n \in [0,1]$ 表示。然后该松弛约束的优化问题(P1)相对于原卸载策略 可以用凸优化工具箱解决。一旦 $\hat{a}_n$ 给出，则可通过下式计算出相应的二进制卸载策略：

$$a_n = \begin{cases} 1 & \hat{a}_n \geqslant 0.5 \\ 0 & \text{其他} \end{cases} \tag{26}$$

为避免偶然性，在整个实验过程中，均保证了神经网络结构与超参数设置保持一致。从测试集中随机选取了 10 个场景下测试的均值作为实际的测试效果。如图 7 所示，LR 算法产生的加权时延和归一化值 $\hat{Q}$ 始终保持为 0.972，不会随着训练样本数增多而改变。FEMO 算法在 20 个微调步数内，其产生的加权时延和归一化值 $\hat{Q}$ 能超过 0.990，可以快速适应新的计算任务场景。DSLO 算法随着训练样本数量的增加，在 120 个微调步数内，其产生的加权时延和归一化值 $\hat{Q}$ 能超过 0.990，但是显然在 200 个微调步数内还没有收敛到最优。因此，相较于 FEMO 算法，DSLO 算法需要更多的训练样本数以及更多的微调步数。值得一提的是，DROO 算法由于缺乏标签数据，相较于 DSLO 算法和 FEMO 算法需要更多的微调步数才能适应新的计算任务场景，在 200 个微调步数内，其产生的加权时延和归一化值 $\hat{Q}$ 为0.968。

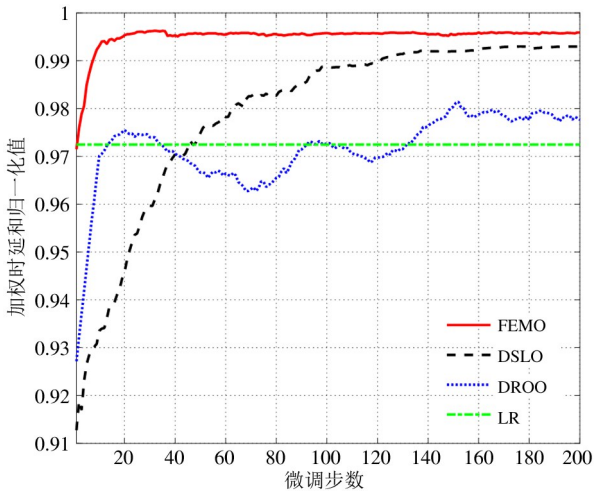


图 7　不同算法的加权时延和归一化值

总的来说，FEMO 算法通过学习历史场景中的先验知识，并且在少量训练样本的情况下可以达到比其他计算卸载算法更好的模型初始化参数。该算法可以迅速适应新的计算任务场景。

5.3 FEMO 泛化性能分析

如表 1 所示,本文对比了在不同计算卸载算法下加权时延和归一化值。本小节将本文提出的 FEMO 算法与随机卸载算法和线性松弛卸载算法进行了比较。由于 FEMO 算法是利用多个计算任务场景中的先验知识来得到一个较好的神经网络初始化参数,因此,在个性化微调测试中,针对一个新的计算任务场景,在不做微调测试的情况下,其加权时延和归一化值为 0.976 7,并优于其他 2 种对比算法的卸载决策,最终在 10 步微调测试内收敛于该场景。

表 1 不同计算卸载算法性能对比

计算卸载方法	加权时延和归一化值
随机卸载决策	0.908 4
基于线性松弛的计算卸载	0.972 5
FEMO 不经过微调训练	0.976 7
FEMO 执行 1 步微调训练	0.983 4
FEMO 执行 5 步微调训练	0.990 5
FEMO 执行 10 步微调训练	0.995 1

本实验一共生成了 1024 个训练任务场景,其中 80% 的计算任务场景作为训练集。如图 8 所示,本文利用不同数量的计算任务场景训练云端模型,在边缘服务器数量 K 不变的情况下,将 L 分别设置原训练集的 40%、60%、80%、100%。其中,每个边缘服务器在仅有原训练集 40% 的计算任务且每个场景只有 10 个训练样本的前提下,针对新的计算任务场景,FEMO 算法通过 10 次迭代训练后,加权时延和归一化值能超过 0.985。然而,更多的计算任务场景意味着更多的先验知识,使得神经网络模型有着更高的初始化参数。

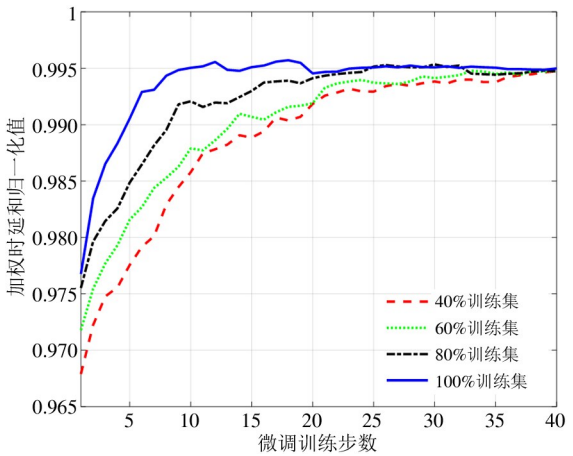


图 8 不同规模的训练数据对算法性能的影响

6 结 论

本文面向 5G 环境下的万物智联应用场景,阐述了移动边缘计算卸载是融合计算和通信的关键技术。针对 5G 实际应用场景要求计算卸载决策实时性和边缘数据隐私保护安全性等挑战,本文融合联邦学习和元学习,实现了一种兼顾性能优化与隐私保护的边缘计算卸载通用框架。测试实验表明,FEMO 算法仅需少量的微调训练步数就能达到 0.995以上的性能,可为未来 5G 移动边缘计算应用提供关键技术支撑。

参考文献

[1] MACH P, BECVAR Z. Mobile edge computing: a survey on architecture and computation offloading[J]. IEEE Communications Surveys & Tutorials, 2017, 19(3): 1628-1656.

[2] SATYANARAYANAN M. The emergence of edge computing[J]. Computer, 2017,50(1):30-39.

[3] LI X, HUANG L, WANG H, et al. An integrated optimization-learning framework for online combinatorial computation offloading in MEC networks[J]. IEEE Wireless Communications, 2022,29(1):170-177.

[4] 龙隆, 刘子辰, 石晶林, 等. 移动边缘计算中计算卸载与资源分配的联合优化策略[J]. 高技术通讯, 2020,30(8):765-773.

[5] 徐旭,钱丽萍,吴远. 基于移动边缘计算的区块链计算资源分配和收益分享研究[J]. 计算机科学, 2021,48(11):124-132.

[6] NARENDRA P M, FUKUNAGA K. A branch and bound algorithm for feature subset selection[J]. IEEE Transactions on Computers, 1977,26(9):917-922.

[7] BERTSEKAS D. Dynamic programming and optimal control: volume I[M]. Massachusetts: Athena Scientific, 2012.

[8] 李邱苹, 赵军辉, 贡毅. 移动边缘计算中的计算卸载

和资源管理方案[J]. 电信科学, 2019,35(3):36-46.
[9] 张依琳, 梁玉珠, 尹沐君, 等. 移动边缘计算中计算卸载方案研究综述[J]. 计算机学报, 2021,44(12):2406-2430.
[10] 刘婷, 罗喜良. 移动边缘计算中的在线任务卸载方法[J]. 中国科学院大学学报, 2022,39(2):267-274.
[11] 范艳芳, 袁爽, 蔡英, 等. 车载边缘计算中基于深度强化学习的协同计算卸载方案[J]. 计算机科学, 2021,48(5):270-276.
[12] BI S, HUANG L, WANG H, et al. Lyapunov-guided deep reinforcement learning for stable online computation offloading in mobile-edge computing networks[J]. IEEE Transactions on Wireless Communications, 2021, 20(11):7519-7537.
[13] LECUN Y, BENGIO Y, HINTON G. Deep learning[J]. Nature, 2015,521(7553):436-444.
[14] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015,518(7540):529-533.
[15] DULAC-ARNOLD G, EVANS R, VAN HASSELT H, et al. Deep reinforcement learning in large discrete action spaces[EB/OL]. (2015-12-24)[2022-04-23]. https://arxiv.org/pdf/1512.07679.pdf.
[16] YU S, WANG X, LANGAR R. Computation offloading for mobile edge computing: a deep learning approach[C]//Proceedings of 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications. Piscataway: IEEE, 2017:1-6.
[17] LUONG N C, XIONG Z, WANG P, et al. Optimal auction for edge computing resource management in mobile blockchain networks: a deep learning approach[C]// Proceedings of 2018 IEEE International Conference on Communications. Piscataway: IEEE Press, 2018:1-6.
[18] MIN M, XIAO L, CHEN Y, et al. Learning-based computation offloading for IoT devices with energy harvesting[J]. IEEE Transactions on Vehicular Technology, 2019, 68(2):1930-1941.
[19] HUANG L, FENG X, ZHANG C L, et al. Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing[J]. Digital Communications and Networks, 2019, 5(1):10-17.
[20] ALE L, ZHANG N, FANG X, et al. Delay-aware and energy-efficient computation offloading in mobile edge computing using deep reinforcement learning[J]. IEEE Transactions on Cognitive Communications and Networking, 2021,7(3):881-892.
[21] 杨乐, 李萌, 叶欣宇, 等. 融合边缘计算与区块链的工业互联网资源优化配置研究[J]. 高技术通讯, 2021,30(12):1253-1263.
[22] HADIDI R, CAO J, RYOO M S, et al. Toward collaborative inferencing of deep neural networks on Internet-of-Things devices[J]. IEEE Internet of Things Journal, 2020,7(6):4950-4960.
[23] LIM W Y B, LUONG N C, HOANG D T, et al. Federated learning in mobile edge networks: a comprehensive survey[J]. IEEE Communications Surveys & Tutorials, 2020,22(3):2031-2063.
[24] WANG J, HU J, MIN G, et al. Fast adaptive task offloading in edge computing based on meta reinforcement learning[J]. IEEE Transactions on Parallel and Distributed Systems, 2020,32(1):242-253.
[25] HUANG L, ZHANG L, YANG S, et al. Meta-learning based dynamic computation task offloading for mobile edge computing networks[J]. IEEE Communications Letters, 2020,25(5):1568-1572.
[26] CHEN Z, WANG X. Decentralized computation offloading for multi-user mobile edge computing: a deep reinforcement learning approach[J]. EURASIP Journal on Wireless Communications and Networking, 2020(1):1-21.
[27] DING M, LOPEZ-PEREZ D, CLAUSSEN H, et al. On the fundamental characteristics of ultra-dense small cell networks[J]. IEEE Network, 2018,32(3):92-100.
[28] MIETTINEN A P, NURMINEN J K. Energy efficiency of mobile clients in cloud computing[C]//The 2nd USENIX Workshop on Hot Topics in Cloud Computing. Boston: USENIX Association, 2010:1-4.
[29] YU S, WANG X, LANGAR R. Computation offloading for mobile edge computing: a deep learning approach[C]//IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications. Montreal: IEEE, 2017:1-6.
[30] HUANG L, BI S, ZHANG Y J A. Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks[J]. IEEE Transac-

tions on Mobile Computing, 2019,19(11):2581-2593.
[31] GUO S, XIAO B, YANG Y, et al. Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing[C]// The 35th Annual IEEE International Conference on Computer Communications. San Francisco: IEEE, 2016:1-9.

Secure mobile edge computation offloading framework via federated meta-learning

YANG Shicheng*, CHEN Baoluo**, CHEN Tieming**, HUANG Liang**
(*College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023)
(**College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023)

Abstract

Emerging mobile edge computing(MEC) technologies bring 5G, cloud computing, big data, and artificial intelligence technologies close to IoT terminals by offloading computing tasks to edge servers. Aiming to efficiently offload computing tasks and protect user privacy, the existing computing offloading technologies are investigated and a secure computational offloading application framework that integrates federated learning and meta-learning is proposed. Through the joint optimization of computational tasks offloading and computational resource allocation, the proposed algorithm minimizes the weighted sum delay of the system. To protect from revealing user data, jointly training a general model with multiple edge servers, personalized computing offloading on each edge server is achieved. Under a new computational scenario, the parameters of the general model can quickly converge with few training samples. Simulation results show that the proposed federated meta-learning-based computing offloading framework can guarantee privacy security for future mobile edge computing applications.

Key words: mobile edge computing(MEC), privacy protection, federated learning, meta-learning