The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21)

面向全局和个性化模型学习的联邦块坐标下降方案

吴瑞媛[1]、Anna Scaglione[2]、魏海陶[3]、Nurullah Karakoc[2]、Kari Hreinsson[2]、马永健[1]

[1]中国香港特别行政区香港中文大学电子工程系[2]美国亚利桑那州立大学电子计算机与能源工程学院

[3]香港特别行政区香港中文大学系统工程与工程管理系[1]{rywu, wkma}@ee.cuhk.edu.hk，[2]{Anna。Scaglione, nkarakoc, khreinss}@asu.edu，[3]htwai@se.cuhk.edu.hk

摘要

在联邦学习中，模型是从用户的数据中学习的，这些数据在他们的边缘设备中是私有的，通过将它们聚合在服务提供商的“云”中来获得全局模型。这样的全局模型在改善客户体验等方面具有巨大的商业价值。在本文中，我们将重点放在两个可能改进当前技术的领域。首先，我们考虑到用户习惯之间的差异，并提出了一个基于二次惩罚的公式，用于有效学习全局模型，从而允许个性化局部模型。其次，我们通过利用分层结构建模通信，不仅在云和边缘设备之间，而且在云内，解决了与边缘设备上异构训练时间相关的延迟问题。具体来说，我们设计了一个定制的基于块坐标下降的计算方案，并附带了同步和异步云设置的通信协议。我们描述了该算法的理论收敛率，并提供了一个在经验上表现更好的变体。我们还证明，受多智能体共识技术启发的异步协议，在边缘设备更新是间歇性的情况下，与同步设置相比，在延迟方面有很大的潜力。最后，提供的实验结果不仅证实了理论，而且还表明，与目前的技术水平相比，该系统可以更快地收敛边缘设备上的个性化模型。

Introduction

在过去几年中，*分布式学习*的一个新兴分支——联邦学习引起了越来越多的关注(McMahan等，2016;Li等人2018; Bonawitz et al. 2019;Yang et al. 2019)。它关注的是用户数据被处理用于低距离训练机器学习模型的场景，即在用户的边缘设备(如手机和可穿戴设备)上，使数据保持私密性。事实上，数据隐私是重中之重;用户愿意与可靠的服务提供商共享经过训练的模型，但不一定是原始数据。在这种情况下，服务提供商通过聚合这些局部模型来寻求一个全局模型——反过来，它可以增强用户的性能。这样的全局模型体现了“众智”，对服务有帮助服务商更好地了解客户偏好。联邦学习与传统的分布式学习(优化通常发生在稳定的数据中心)的区别在于以下几个方面(Li等人，2020):

· 用户不同边缘设备中可用的数据和计算能力的异质性;

· 边缘设备和云之间通信的间歇性(和昂贵)性质。

为了用数学术语来解释它，以下问题是联邦学习中的典型问题:

$$\min_{\boldsymbol{x}_i, \boldsymbol{z} \in \mathcal{X}} \sum_{i \in \mathcal{Q}} g_i(\boldsymbol{x}_i), \ \text{s.t. } \boldsymbol{x}_i = \boldsymbol{z}, \ \forall i \in \mathcal{Q}, \tag{1}$$

其中Q是边缘设备的集合，x_i和g_i是依赖于局部数据的第i个边缘设备上的模型参数和成本函数，X是the$_P$可行区域。具体来说，我们可以写gi(xi):= $_{|S1i|\ r\in S_i}$ h(xi;Sr)，其中h是训练损失函数，S_i是第i个边缘设备上训练数据的索引集，$|S_i|$是集合中元素的个数S_i，s_r是这样一个样本。为了解决问题(1)，联邦学习方法通常采用递归机制:边缘设备处理自己的训练数据以更新局部模型x_i’s，并引入云对全局模型z聚合x_i’s，并将所有局部模型与更新后的z同步。这一过程被认为是当前开发中的标准过程(McMahan等人，2016; Li等人，2018)。然而，我们注意到两个事实留下了一些改进的空间:

(i)*在所有地方保持相同的模型是否有效*?数据在边缘设备上通常是非独立和同分布(i.i.d)的，因为它们反映了不同用户的使用习惯。有鉴于此，能够很好地处理用户感兴趣的数据(这也是非i.i.d)的边缘设备模型可能就足够了——换句话说，个性化模型可能更擅长于它们主要用于的任务。当维护单一模型时，用户可能会牺牲自己的客户体验，以改进对服务提供商更有利的全局模型，从而提升新用户的模型。

(ii)*同步云模式是否有效*?云由一组并行工作的服务器组成，一个边缘设备只需要与一个这样的云服务器进行通信。当将云视为唯一的计算资源时，

一个隐式强制某种形式的云同步，或同步云(可通过allreduce (Patarasuk and Yuan 2009)等技术实现);如图1(a)所示。在联邦学习中，这种同步可能会导致显著的更新延迟。要看到这一点，回想一下，由于通信和计算的容量异质性，来自边缘设备的可用性时间和本地训练时间可能会有很大差异。因此，一种可能的场景是，大多数云服务器被一些“缓慢”激活的边缘设备所困，这些设备甚至从未与它们交谈。

贡献

本文的重点是解决上述两个问题。我们的贡献和新颖性可以总结如下:

· 我们的工作为联邦学习提出了一个定制的分层通信架构。这种结构由主从和多智能体网络组成，尽管承认“令人惊讶”的熟悉外观，但以前从未被研究过。

· 我们提出了一种轻量级的块坐标下降计算方案，配备了明智设计的通信协议，这是第一个可以同时实现模型个性化和云服务器异步更新的工作-两个看似无关的问题实际上与联邦学习密切相关(cf . Remark 4)。

· 显示了所提出算法的亚线性收敛率。由于我们的通信架构包含两层信息交换，因此分析需要与现有工作不同的分析工具。

· 我们提供延迟分析来证明异步更新对联邦学习的有效性。我们的延迟分析框架实际上允许从边缘设备消息到达时间的分布中估计运行时，并将每次更新中涉及的云服务器数量与运行时连接起来，这是新的。

· 最后，在标准机器学习应用程序上进行了数值实验，以支持我们的说法。

相关工作

fedag (McMahan et al. 2016)是一种简单的迭代算法，被认为是联邦学习的第一部作品。在每一轮fedag中，云将全局模型发送到激活的部分边缘设备;然后，激活边缘设备对局部代价函数进行固定次随机梯度下降(SGD)，更新其局部模型;最后，云将上传的局部模型(通过加权求和)聚合为新的全局模型。大多数后续工作采用了与fedag引入的机制类似的机制(Li等人，2020)。例如，一个名为FedProx的最新变体(Li等人，2018)在边缘设备更新步骤上与fedag不同，在边缘设备更新步骤中，它向成本函数施加了额外的近似值项，并允许使用更快算法的时变epoch，例如加速SGD。

在联邦学习场景下，模型个性化是一个自然的，但有时被忽视的问题。在(Mohri, Sivek, and Suresh 2019)的工作中，作者认为，由于个人偏好，通过fedag训练的模型可能会偏向于大多数人的利益。他们提出AFL具有复杂的机制来确定云聚合中涉及的边缘设备模型的权重，而不是fedag中使用的简单数据大小比例。per - fedag (Fallah, Mokhtari和Ozdaglar 2020)将全球模型与边缘设备上的模型区分开来。他们的目标是寻求一个良好的“初始化”，作为全局模型，它可以很容易地在本地升级，通过几个简单的更新步骤，使每个边缘设备都是最优的。(Smith et al. 2017)将模型个性化视为一个多任务学习问题。他们提出的MOCHA可以为每个设备学习独立但相关的模型，同时通过多任务学习利用共享表示。证明模型个性化的重要性是本文准备和提交期间发表的工作(Jiang et al. 2019;Arivazhagan et al. 2019;Wu, He, and Chen 2020;Hu et al. 2020)，专门关注这个问题。另一方面，没有一项工作像我们一样考虑基于惩罚的方法。

相比之下，有大量关于异步多智能体网络中的分布式学习的文献;参见(Nedi´c, Olshevsky, and Rabbat 2018)最近的一项调查。虽然这样的实现通常比主从实现需要更多的迭代，但它们没有协调开销。最近(Assran et al. 2019)通过实验证明，开销的减少在运行时方面产生了显着的好处。当在大型分布式数据库上处理深度学习问题时，异步多代理算法比主从算法运行得更快，因为与传统的主从或增量架构相比，宽松的协调要求反过来有助于无滞后地完成每次更新。截至本文提交时，我们还没有看到在联邦学习的背景下对这一主题的研究，也没有关于如何表征性能趋势与算法运行时的理论，而不是简单地关注所需的迭代次数。虽然有工作提到了异步联邦学习(Chen et al. 2019;Lu et al. 2019)，他们专注于边缘设备和云之间的通信导致的异步更新——云仍然被视为唯一的点。

问题陈述

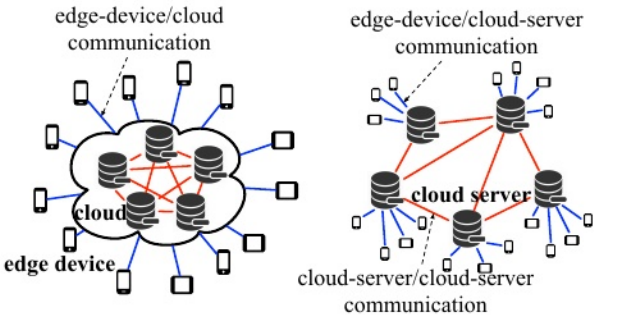

图1:两种通信架构。

我们将云服务器相互分离，同时考虑服务器/服务器和设备/服务器通信。为了解释，云服务器是由(可能)动态的多智能体图连接的，它们需要相互交谈以达成共识，无论是精确的还是渐近的;设备-副/服务器通信采用主从方式，并且与一般联邦学习一样，具有间歇性和随机性。我们称之为异步云架构，以区别于fedag采用的同步云架构;两者的区别见图1。异步云架构允许云服务器上的模型差异。正如我们稍后将展示的那样，可以利用这种灵活性来促进更有效的模型更新。

基于上述架构，我们提出的公式是

$$\begin{aligned}\min_{\substack{\{\boldsymbol{x}_i\}_{i\in\mathcal{Q}_n,n\in\mathcal{V}}\\ \{\boldsymbol{z}_n\}_{n\in\mathcal{V}}}}\ & \sum_{n\in\mathcal{V}}\sum_{i\in\mathcal{Q}_n}\left(g_i(\boldsymbol{x}_i)+\frac{\gamma_i}{2}\|\boldsymbol{x}_i-\boldsymbol{z}_n\|^2\right)\\ \text{s.t.}\quad & \boldsymbol{x}_i\in\mathcal{X},\forall i\in\mathcal{Q}_n,n\in\mathcal{V},\\ & \boldsymbol{z}_n=\boldsymbol{z}_m,\ \forall(n,m)\in\mathcal{E},\end{aligned}\tag{2}$$

式中，V为云服务器集合，Q_n为连接第n个云服务器的边缘设备集合，x_i’s为边缘设备模型，z_n’s为云服务器模型，E≠(V ×V)表示云服务器之间的通信，γ_i> 0为罚参数。在续集中，z_n’s将被称为全局模型，x_i’s将被称为个性化模型。我们的for-mulation区分了不同云服务器上的全局模型，并允许(但通过二次惩罚正则化器进行惩罚)个性化模型之间的偏差。

备注1*最近在不同的分布式学习文献中重新审视了看似na¨ve二次惩罚。在(Zhang, Choromanska, and LeCun 2015)中,这种惩罚用于在底层优化具有许多局部最优的深度学习任务中寻求更好的解决方案。这种二次惩罚也在对抗性场景中进行了研究,在对抗性场景中,全局模型有望抵御恶意设备的攻击(Yang, Gang, and Bajwa 2020)。*

Remark 2 *(Global and Personalized Mod-els的动机)考虑智能手机键盘的应用。用户想要准确的下一个单词预测,量身定制的个性化模型可以更好地实现这一点。然而,每个用户产生的用于训练的数据非常有限,因此,其局部个性化模型可能无法适用于新的场景,而这正是来自全局模型的综合知识提供帮助的地方(在我们的信息中,这是通过鼓励个性化模型接近其全局对应模型来实现的)。此外,全局模型可以作为新用户的无偏初始化。*

联邦块坐标下降

在本节中，我们将解决(2)中定义的问题。我们的开发基于经典的块坐标下降(BCD)方案(Bertsekas 1997):

$$\{\boldsymbol{x}_i^{(t+1)}\}_{\substack{i\in\mathcal{Q}_n\\ n\in\mathcal{V}}}=\operatorname*{argmin}_{\boldsymbol{x}_i\in\mathcal{X}} g_i(\boldsymbol{x}_i)+\frac{\gamma_i}{2}\|\boldsymbol{x}_i-\boldsymbol{z}_n^{(t)}\|^2,\tag{3a}$$

$$\{\boldsymbol{z}_n^{(t+1)}\}_{n\in\mathcal{V}}=\operatorname*{argmin}_{\{\boldsymbol{z}_n\}_{n\in\mathcal{V}}}\sum_{n\in\mathcal{V}}\sum_{i\in\mathcal{Q}_n}\frac{\gamma_i}{2}\|\boldsymbol{z}_n-\boldsymbol{x}_i^{(t+1)}\|^2\tag{3b}$$

$$\text{s.t.}\quad \boldsymbol{z}_n=\boldsymbol{z}_m,\ \forall(n,m)\in\mathcal{E}.$$

我们后续的工作可以解释为使BCD更新(3)适应同步云和异步云架构。与fedag相同，假设每轮只有部分边缘设备可用，用$Q_{n(t)}$ Q_n表示第t轮激活的边缘设备集(即可进行局部训练且通信状态稳定的边缘设备)。在续集中，我们将分别对云服务器和边缘设备更新进行阐述。

边缘设备上的更新。如果边缘设备i未激活，即i∈/ $Q_{n(t)}$，则有xi(t+1) = x(t)i。否则，则应用加速随机投影梯度(ASPG)的时变epoch。具体来说，让t¯be在第i个边缘设备被激活时的最后一轮，并给定initial-$_{(t^-,K}$

假设xi(t,0) = xi$_{(}$$_{t}$$_{)}$且x(t,i−1)= x(ti−，Ki−1)，则边缘器$^{(t-)}$件递归执行:

$$\boldsymbol{x}_{i,\text{ex}}^{(t,k)}=\boldsymbol{x}_i^{(t,k)}+\zeta(\boldsymbol{x}_i^{(t,k)}-\boldsymbol{x}_i^{(t,k-1)}),\tag{4a}$$

$$\boldsymbol{x}_i^{(t,k+1)}=\tag{4b}$$

$$\Pi_{\mathcal{X}}\left(\boldsymbol{x}_{i,\text{ex}}^{(t,k)}-\eta_x\left(\nabla\tilde{g}_i(\boldsymbol{x}_{i,\text{ex}}^{(t,k)})+\gamma_i(\boldsymbol{x}_{i,\text{ex}}^{(t,k)}-\boldsymbol{z}_n^{(t)})\right)\right),$$

对于k = 0，⋯，Ki(t)−1，其中ηx是步长，ζ≥0是动量权，Π_X是X上的投影，∇g⊠$_i$是小批量随机梯度，使得∇⊠g_i= $_1$ PR
− $_{R\,R\,=\,1}$ ∇h (x)；ξ$_{(t,k))}$与ξ $_{(t,\,k)}$ $_{t\,t}$是

与边缘设备和R_{w}是批量大小。最终更新$_{(t+1)}$ $_{(t,K}$
$_{i}$是xi= xi)，如果ζ= 0, update(4)缩小到标准SPG。经验上，观察到ASPG更新收敛得更快(Beck and Teboulle 2009)。在云服务器上进行更新。我们为同步云和异步云架构导出了(3b)的更新规则。在同步情况下，与FedAvg相同，我们可以将所有参与云服务器上的全局模型表示为z_n= z，对于n∈V。然后，给定初始化z(t)，云更新

(同步云)

$$\boldsymbol{z}^{(t+1)}=\boldsymbol{z}^{(t)}-\eta_z\sum_{n\in\mathcal{V}}\sum_{i\in\mathcal{Q}_n}\gamma_i(\boldsymbol{z}^{(t)}-\boldsymbol{x}_i^{(t+1)}),\tag{5}$$

其中η_z为步长。

在异步设置中，我们建议使用分布式梯度下降(DGD)算法(Ram, Nedi´c, and Veeravalli 2010)。类似地，给定初始化$z_n^{(t)}$，对于n∈V，每个云服务器都运行

(异步云)

$$\boldsymbol{w}_n^{(t)}=\sum_{m\in\mathcal{V}}a_{n,m}^{(t)}\boldsymbol{z}_m^{(t)},\tag{6a}$$

$$\boldsymbol{z}_n^{(t+1)}=\boldsymbol{w}_n^{(t)}-\eta_z\sum_{i\in\mathcal{Q}_n}\gamma_i(\boldsymbol{w}_n^{(t)}-\boldsymbol{x}_i^{(t+1)}),\tag{6b}$$

其中$a_{n,m}$是由$_{(t)}$决定的混合系数
第t轮的服务器/服务器通信链路。

Algorithm 1 Pseudocode of FedBCD for problem (2)

1: **for** $t = 0, 1, \ldots, T-1$ **do**
2: **for** $n \in \mathcal{V}$ **do**
3: cloud server sends $\boldsymbol{z}_n^{(t)}$ to activated edge device $i \in \mathcal{Q}_n^{(t)}$.
4: activated edge device updates $\boldsymbol{x}_i^{(t)}$ by $K_i^{(t)}$ epochs of ASPG (4) on $g_i(\boldsymbol{x}_i) + \frac{\gamma_i}{2}\|\boldsymbol{x}_i - \boldsymbol{z}_n^{(t)}\|^2$ and obtain $\boldsymbol{x}_i^{(t+1)}$; other edge device has $\boldsymbol{x}_i^{(t+1)} = \boldsymbol{x}_i^{(t)}$.
5: cloud server receives the uploaded $\boldsymbol{x}_i^{(t+1)}$'s from activated edge devices.
6: cloud servers update $\boldsymbol{z}_n^{(t+1)}$'s by the Sync-cloud/Async-cloud update (5) or (6)

结合式(4)-式(6)，得到FedBCD格式。由于篇幅限制，我们只给出了算法1中FedBCD的伪代码，准确版本在补充文档中给出。在第一部分中，我们将描述FedBCD的理论收敛保证。

FedBCD的同步云和异步云通信协议

我们将介绍两种通信协议，以阐明如何应用FedBCD，以及异步云协议如何允许更有效的更新。

· *Sync-cloud协议*。该协议与(Bonawitz et al. 2019)中的协议类似，有两个阶段:(i)新一轮t从“响应和更新”阶段开始。在这一阶段，云服务器接受来自边缘设备的更新请求，将全局模型$z^{(t)}$发送给这些激活的边缘设备，并等待它们执行本地更新(4)。对于每个云服务器，这一阶段在接收到$xi(t+1)$'s时结束

来自预先确定数量的边缘设备。然后，这个云服务器将$xi_{(t)}$'s的本地副本替换为$xi(t+1)$'s
(对于未激活的边缘设备，它只需将$xi(t+1) = xi_{(t)}$)，然后发送$P\ x_{(t+1)}$

$i \in Q_n i$

给一个保持最新的协调器

全局模型z(t)。(ii)一旦收到所有云服务器的消息，该协调器进入“聚合”阶段，执行(5)以获得z(t+1)并将其广播回来。接收到$z^{(t+1)}$后，云服务器再次可用，进行下一轮更新。

· *异步云协议*。这个协议需要更仔细的实现。(i)在第t轮，完成“响应和更新”阶段的云服务器将其模型z_n's发送给协调器。(ii)协调器等待，直到它$_{(t)}$

接收到前B个云服务器$V_{(t)}$[在图2(B)中为$|V_{(t)}|=B=2$]，并进入“聚合”阶段i。然后，该协调器计算$w(t) = Pn \in V_{(t)}\ zn_{(t)/B}$，并将w(t)发送给$V_{(t)}$中的云服务器。(iii) $V_{(t)}$中的云服务器进入“聚合”阶段II，它们使用接收到的$w^{(t)}$作为wn并更新zn

(6);其他云服务器sim-

Ply有$z_{n\ (t+1)} = z_n$。回到我们的FedBCD方案，这$_{(t)}$
协议对应如下设置:$a_{n,m} = 1/B$ $_{(t)}$
对于$n,m \in V_{(t)}$，$a(t)n,n = 1$;对于$n,m \in / V_{(t),\ an,m\ (t)} = 0$;对于$n \in /V_{(t),\ Qn\ (t)} = \varnothing$，$\eta z(t)n = 0$。

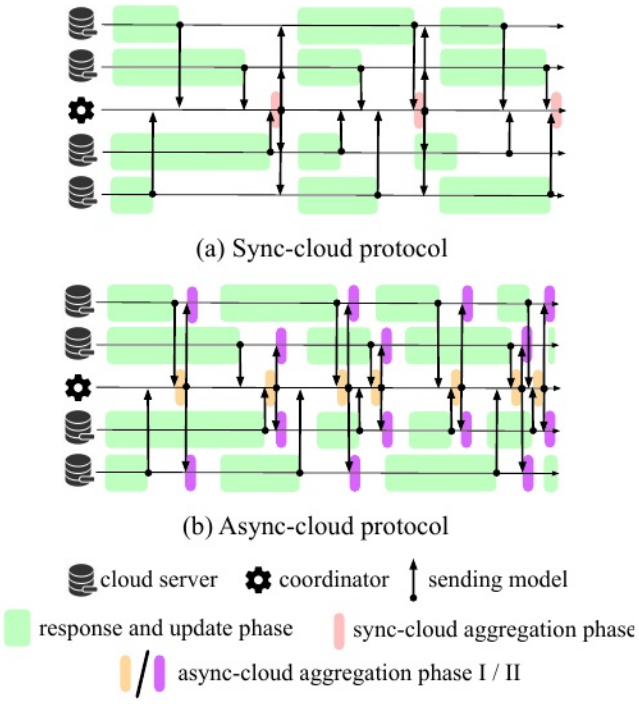

图2:同步/异步云架构下的协议。

在同步云协议中，模型聚合发生在所有云服务器完成“响应和更新”阶段时;而在第二种协议中，云服务器异步工作，模型聚合以“先到先得”的方式进行。通过构建基于先准备好的云服务器动态混合模型的云服务器邻域，异步云协议，如下面分析的，有望提供更有效的更新;如图2所示。

注释3一些读者可能会认为,在两种协议中使用协调器不利于多智能体通信。然而,与传统设置不同的是,这里的协调器只涉及轻量级计算。我们认为,随着软件定义网络的出现,可以很自然地假设专门的服务器通信,如多代理消息交换,可以以有效的方式进行编排,以支持特定的应用程序。完全消除协调的方法需要重新规范化权重(Assran等人,2019);通常算法需要更长的时间才能收敛。

延迟的分析。在这里，我们提供了一种获得分析洞察力的方法，通过假设间隔更新时间的一定分布。设$\tau_{n\ (t)}$，$n \in V$为云服务器n累积$Q^{(t)}_n$边缘设备更新所花费的随机时间;设$\tau_{n\ (t)}$为i.i.d，均值和标准差分别为$\mu\tau$，$\sigma\tau$。每个$\tau_{n\ (t)}$等于更新请求竞争时间和与$Q^{(t)}_n$边缘设备相关的计算时间之和。用$|V| = N$，用$\tau(k)(t)$表示，$k = 1, \cdots, N$样本的有序统计量$\tau_{n\ (t)}$按递增顺序表示，即$\tau(1)_{(t)} \leqslant \tau(2)_{(t)} \leqslant \cdots \leqslant \tau(N)_{(t)} = \max n \in V\ \tau\ N_{(t)}$，令

同步云和异步云协议分别为$\tau_{SC}^{(t)}$和τAC(t)。由此可见,τ SC (t) =τ(N) (t)和τAC (t) =τ(B) (t)。因此，每轮平均持续时间的减少为:

$$r_{B,N} := \mathbb{E}[\tau_{(B)}^{(t)}]/\mathbb{E}[\tau_{(N)}^{(t)}] \equiv \mathbb{E}[\tau_{\mathrm{AC}}^{(t)}]/\mathbb{E}[\tau_{\mathrm{SC}}^{(t)}]. \quad (7)$$

设β= B/N表示每次更新所涉及的网络节点的百分比，并考虑$r_{B,N}$⩽rB,N−1。对于大N，我们可以利用经典结果(Mosteller 2006)，显示大样本量下有序统计量的渐近正态性。特别是通过去记Fτ−n1 (u)随机变量τ_n的分位数函数，我们有

$$\tau_{(\beta N)}^{(t)} \sim \mathcal{N}\left(F_{\tau_n}^{-1}(\beta), \frac{\beta(1-\beta)}{N\left[f_{\tau_n}\left(F_{\tau_n}^{-1}(\beta)\right)\right]^2}\right) \quad \Rightarrow$$

$$r_{\beta N,N} \lessapprox \frac{F_{\tau_n}^{-1}(\beta)}{F_{\tau_n}^{-1}(1-1/N)}, \text{ for } N \gg 1. \quad (8)$$

τn (t)与样本空间(0,+∞)Fτ−1 n(1−1 / n)→+∞

则潜伏期的增益是无限大的。然而，对于典型的分布，速率很慢。例如，对于$\tau_n^{(t)}$遵循威布尔分布，具有形状参数k(不考虑尺度参数λ)，我们有rβN,N /−(ln(1−β))$_k^{1}$ (log(N))$_{-k^{1}}$ − −

对于具有有限支持[0，τ]且τ< +∞的随机时间分布，则还原饱和到Fτ−n1 (β)/τ。还要注意的是，一般来说，每次更新β中网络节点的百分比越小，每轮更新进度就越慢，从而在减少延迟和增加更新进度之间做出权衡，从而得出β的最佳选择。最后，应该指出的是，我们只提供了一个简单的解决方案来说明异步云架构的潜力。像DGD这样的多智能体共识技术已经在分布式学习中得到了很好的研究(Assran等人，2019)，为进一步的进步打开了大门，以促进更有效的更新。

Remark 4*模型个性化和异步更新乍一看似乎是两个独立的主题。然而,我们强调这两者本质上与联邦学习的异构性相匹配——它们是齐头并进的,因为边缘设备的特殊行为在非id数据和执行更新的非统一时间中都表现出来。鉴于此,只强制使用相等的模型而应用异步更新(或相反)实际上是一种半途而废的做法。*

FedBCD的直观变体

在FedBCD中，边缘设备只有在通信稳定的情况下才会更新本地模型，这可能导致本地模型更新缓慢。另一方面，我们可以合理地假设，边缘设备也可以离线训练低局部模型。例如，考虑一部手机充电后处于空闲状态。这是手机进行本地训练的好时机，即使它没有连接Wi-Fi。这样，我们就有Qn (t)倍于Q ~ (t)倍于Qn，其中Q ~ (t) n n为可用于本地培训但通信不良的边缘设备集合

Algorithm 2 Pseudocode of FedBCD-I for problem (2)

1: **for** $t = 0, 1, \ldots, T-1$ **do**
2: **for** $i \in \tilde{\mathcal{Q}}_n^{(t)}, n \in \mathcal{V}$ **do**
3: edge device updates $\boldsymbol{x}_i^{(t)}$ by $K_i^{(t)}$ epochs of ASPG on $g_i(\boldsymbol{x}_i)$ and obtain $\boldsymbol{x}_i^{(t+1)}$.
4: **for** $n \in \mathcal{V}$ **do**
5: cloud server sends $\boldsymbol{z}_n^{(t)}$ to activated edge device $i \in \mathcal{Q}_n^{(t)} \subseteq \tilde{\mathcal{Q}}_n^{(t)}$.
6: activated edge device updates $\boldsymbol{x}_i^{(t+1)}$ by $K_i^{(t)}$ epochs of $\boldsymbol{x}_i^{(t+1)} \leftarrow \Pi_{\mathcal{X}}(\boldsymbol{x}_i^{(t+1)} - \gamma_i(\boldsymbol{x}_i^{(t+1)} - \boldsymbol{z}_n^{(t)}))$.
7: cloud server receives the uploaded $\boldsymbol{x}_i^{(t+1)}$'s from activated edge devices.
8: cloud servers update their models $\boldsymbol{z}_n^{(t+1)}$'s by Sync-cloud/Async-cloud update with information only from $\mathcal{Q}_n^{(t)}$; i.e., replacing $\mathcal{Q}_n$ in (5) or (6) with $\mathcal{Q}_n^{(t)}$.

条件。然后，我们可以采用(Zhang, Choromanska, and LeCun 2015)中建议的直观方法，分别处理(2)中的成本函数和二次惩罚。简单地说，在第t轮，Q⊠$^{(t)}_n$中的边缘设备运行局部训练

使用成本函数g_i's。然后，Q_n中的边缘设备子集为全局模型(t)向云发送更新请求

调整他们的本地模式。云服务器更新保持不变。我们将生成的方案称为FedBCD- i，并在算法2中总结了其伪代码。FedBCD- i可以同时采用为FedBCD引入的同步云协议和异步云协议，只需稍加修改。唯一的区别是，这里的边缘设备只有在完成一轮本地训练后才会发送更新请求。如果边缘设备在预定的轮数内未能与云通信，它将停止本地训练，等待最早的机会与云联系。补充文档中提供了详细的算法和协议描述。

理论收敛分析

我们将在本节中研究FedBCD的行为。我们的结果适用于同步云协议和异步云协议设置。让我们从以下假设开始:

假设1*服务器/服务器和设备/服务器通信满足这一点*

*(i)每个边缘设备每*p*次迭代至少激活一次,其中*p <∞。

(ii)在异步云更新(6)中,*云服务器内部通信图is*$_S$*possibly时变且满足,对于某个*q <∞，(V, *t=1,*⋯， q E (t0 +t)对于所有t0都*是强连接*的。

假设2*在问题*(2)*中,满足以下条件:(i)可行集*X*是凸紧的。*

*(ii)对于所有*i∈Q_n,n∈V,*函数*g_i*在*X上是有界的,*并且在*X上是Li-Lipschitz*光滑的,其中*X≈= conv[0⩽ζ⩽1 {X + *ζ(X*− y) | | X, y∈*X}]是动量更新扩展的可行集。并且,它的梯度*∇g_i*在*X上有界。

异步云*更新*(6)*中的混合系数*$\{a^{(t)}_{n,m}\}_{t=0,1,\ldots}$*满足以下条件:*

*(i)存在一个标量*c > 0，使得如果(n, m)∈E (t)a$^{(t)}_{n,m}$≥c，否则为a$_{n,m}^{(t)}$=0。

*(ii)(双随机)*P $_{m\in V}$ a(t)$_{n,m}$ =P $_{n\in V}$ a(t)n,m = 1。

假设4*步长和动量权满足(i)* ηx≤min $_{\sqrt{\ }\ i\in Q_n,n\in V}${1/(Li + γ i)}。

(ii) ηz = d/ T≤1/(maxi∈Q$_{n}$,n∈V{γ $_{i}$·max $_{n\in V}$ {|Qn|}), *对于*T *的正常数*d*and* $_{\sqrt{\ }}$ *total通信(iii)*ζ≤mini∈Q$_{n}$,n∈V{ω/ 1 + ρ η x}*,对于某些常数*ω∈(0,1)*和*ρi*使得*ρi≤Li + γi。

假设5*对于任意*i*,设*∇h(x$_i$;ξ$_i$)*是*gi(xi)*的随机梯度,其中*ξi*是来自第*i*个边缘设备的随机样本[回忆一下,在*(1)*中我们表示*gi(xi):= $_{|S1i|}$ Pr∈S$_i$ h(xi;sr)]。*对于任意*xi和ξi, E[∇g$_i$(x$_i$)−∇*h*(*x*$_i$;ξ$_i$)])= 0*和*E[k∇g$_i$(x$_i$)−∇*h*(*x*$_i$;ξ$_i$)k^2])≤σ^2]*成立*。

让我们简单检查一下我们的假设:假设1确保没有边缘设备或云服务器被隔离;假设2-3在约束优化和平均共识算法中很常见;假设4要求步长和动量权重是有界的;假设5是典型的随机梯度下降。细心的读者可能会注意到，章节中的异步云协议违反了假设4，因为它考虑了时变的η$_{z_o}$。这个问题可以通过微小的修改来修复;然而，由此产生的版本在实现方面更为复杂，在补充文档中有详细说明。

我们刻画了FedBCD收敛到一个平稳点的特征。我们的收敛度量是(Assran et al. 2019)中的约束版本。

定理1*假设假设1-5成立,通信周期*T*足够大(见补充文件以获得明确定义),并且批大小*R*用于计算随机梯度(参见下文*(4)对随机梯度的详细讨论)。

(i) FedBCD生成的序列{xi $_{(t)}$}*,{zn (t)}满足*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\left\langle\nabla_{\boldsymbol{x}_i} f_i(\boldsymbol{x}_i^{(t+1)},\bar{\boldsymbol{z}}^{(t)}),\hat{\boldsymbol{x}}_i-\boldsymbol{x}_i^{(t+1)}\right\rangle\right]$$

$$\geq -\frac{A_1}{T^{1/4}}-\frac{A_2}{R^{1/4}},\ \forall\hat{\boldsymbol{x}}\in\mathcal{X},\ i\in\mathcal{Q}_n,n\in\mathcal{V},$$

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\left\|\sum_{n\in\mathcal{V}}\sum_{i\in\mathcal{Q}_n}\gamma_i(\boldsymbol{z}^{(t)}-\boldsymbol{x}_i^{(t+1)})\right\|^2\right]\leq\frac{B_1}{\sqrt{T}}+\frac{B_2}{\sqrt{R}},$$

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\bar{\boldsymbol{z}}^{(t)}-\boldsymbol{z}_n^{(t)}\|\leq\frac{C}{\sqrt{T}},\ \forall n\in\mathcal{V},$$

*其中*A1, A2, B1, B2, C*是正常数(由假设1-5中的参数决定),其精确形式在补充文档中,并且*z¯(t) = $_{|V|}$ 1pn∈V zn $_{(t)}$。*(ii)如果我们在更新期间使用精确梯度,则上述结果在删除所有与批大小*R*相关的项时成立。*

定理1的证明在补充文档中。我们看到，批大小R和通信轮T决定了收敛速度，这是基于投影的随机算法中的情况(Ghadimi, Lan, and Zhang 2016)。

请注意，在边缘设备上，数据大小并不大，因此精确梯度可能是可计算的。对于这种情况，我们的结果揭示了O(1/ T)的亚线性收敛速率，这与现有的分散非凸算法-算法一致(Li et al. 2018;Assran et al. 2019)。

*在我们的分析中,我们以统一的方式讨论了同步云和异步云设置的融合。如果我们只考虑同步云设置,FedBCD将成为集中计算方案的特殊情况(Wu, Wai, and Ma 2020)。通过修改假设,使FedBCD适合该计算方案,我们将能够获得*O(1/T)*次线性收敛速率,这是集中式非凸算法的标准。另一方面,应该指出的是,FedBCD的异步云设置在文献中没有见过,需要仔细而非琐碎的分析。*

数值实验

我们在PyTorch上训练了两个模型:用于MNIST数据集分类的三层神经网络(LeCun et al. 1998)和用于CIFAR-10数据集的更深层次的ResNet-20模型(He et al. 2016)。我们假设有10个云服务器，每个服务器连接到10个边缘设备。为了模拟数据的异质性，我们限制了训练数据的“多样性”;例如，“多样性”为3意味着每个边缘设备拥有三个标签的数据。我们还使用一个小的|Q$_n^{(t)}$|来模拟间歇性通信的发生。其他设置为:边缘设备学习率为η$_x$= 0.005，动量权ζ为0.9，边缘设备更新epoch每轮从[1,5]采样，X是一个元素[−2,2]框约束，批大小为R = 32;对于FedBCD/FedBCD- i，云服务器学习率为η$_z$= 0.5;对于FedBCD，惩罚参数为γi = 1;对于FedBCD- i，我们使用γi = 0.2，|Q ~ (t)n | = 8，边缘设备离线完成4轮后暂停局部训练;对于FedProx，边缘设备更新使用ASPG，惩罚参数为μ= 5。以上参数的选择是通过试错法得出的。

模型偏差vs.模型共识

我们将FedBCD/FedBCD- i(允许模型偏差)与fedag/FedProx(强制模型共识)进行比较。考虑了两种性能度量:个性化性能，其中边缘设备在与其训练数据具有相同多样性的测试数据上测试本地模型;以及全局性能，其中云在整个测试数据集上测试全局模型。为了公平起见，在本实验中，所有算法都使用同步云协议(参见fedag / FedProx的同步云协议的补充文档)，并且每轮记录两种类型的性能(即不考虑每轮延迟)。从图3可以看出，fedag /FedProx的个性化性能和全局性能几乎相同。这是有道理的，因为在这里到处都维护着相同的模型。相比之下，FedBCD/FedBCD- i通过允许局部模型偏离全局模型，提供更好的个性化性能。我们也看到了FedBCD

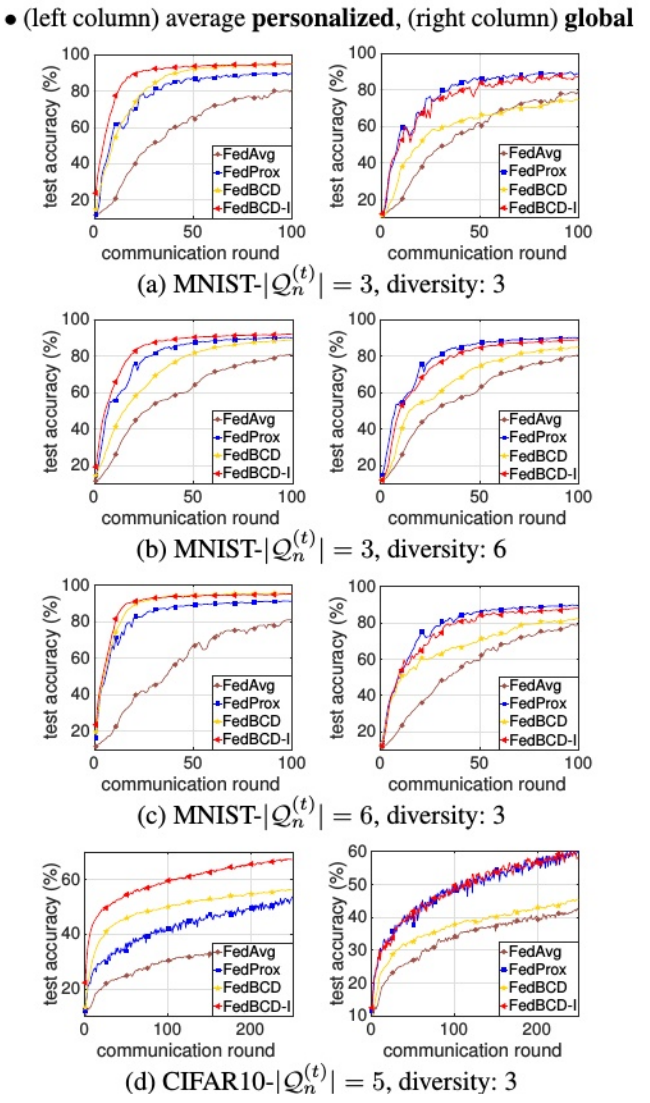

图3:不同算法在不同多样性和激活下对模型的测试精度。

当$|Q_n^{(t)}|$较低且仅比fedag快时，更新进度缓慢(这并不奇怪，请参阅第1节)。相比之下，它的直观变体FedBCD- i更具竞争力;特别是，我们看到在更具挑战性的CIFAR-10任务中，FedBCD- i具有与FedProx相当的全局性能，同时提供更好的个性化性能。

同步云与async云

在第一部分的延迟分析之后，我们将演示异步云更新可能带来的效率增益。为了模拟联邦学习的分布式场景，我们在Python中使用TCP套接字模拟每个边缘设备/云服务器和协调器进行进程间通信，并在四个60核服务器上进行测试，每个服务器具有Intel Xeon E7-4870 v2 cpu和256 GB内存。在实验中，我们

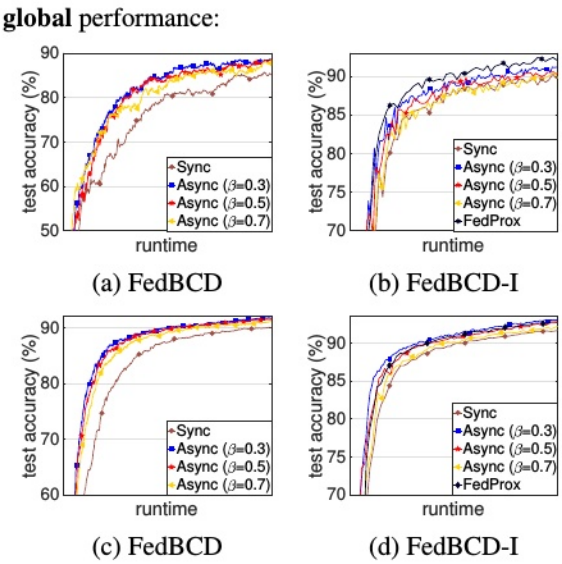

图4:算法在MNIST条件下的效率，$|Q_n^{(t)}|$= 3，“diversity”(a-b) 3，(c-d) 6

。

为“响应”生成数字随机延迟$\tau_n^{(t)}$

和更新”阶段(图2中的绿色阶段)，并假设服务器聚合计算(图2中的非绿色阶段)可以忽略不计，即总体运行时由$\tau_n^{(t)}$主导。我们模拟中使用的详细设置和统计数据在补充文档中给出。结果如图4所示。我们首先注意到，更高的“多样性”导致全局性能的更平滑的提升。这是合理的，因为一个更“偏倚”的局部模型预计会给全局模型训练带来更多的“噪音”。此外，我们可以看到Async-cloud的更新显然更有效率。如前所述，基于β的大小，在每轮的持续时间和更新进度之间存在权衡。在这个实验中，我们看到β= 0.3或0.5似乎取得了很好的平衡，并带来了吸引人的性能。我们还比较了FedBCD-I和FedProx。请注意，FedProx专注于提高全局性能，并且通常比我们的全局性能方法更快。然而，通过使用异步更新，我们的方案甚至可以在运行时方面优于FedProx的全局性能;如图4(d)所示。

结论

这项工作有两个主要贡献:首先，我们认识到边缘设备上的机器学习模型反映了用户习惯，因此可以有所不同;第二，我们强调了云是一个功能强大的服务器集群，它们之间的内部通信可以被用来进行更高效的更新。我们研究了这两个方面，并提供了一个在理论和经验上都被证明很有前途的解决方案。

致谢

这项工作得到了美国陆军研究办公室的支持，项目编号为ARO #W911NF-20-1-0153。吴瑞元和马永健的研究获香港研究资助局(研资局)普通研究基金资助，项目编号为CUHK 14208819。

参考文献

Arivazhagan, M. G.; Aggarwal, V.; Singh, A. K.; and Choudhary, S. 2019. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*.

Assran, M.; Loizou, N.; Ballas, N.; and Rabbat, M. 2019. Stochastic gradient push for distributed deep learning. In *Proc. Int. Conf. Mach. Learn.*, volume 97, 344−353. PMLR. Beck, A.; and Teboulle, M. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.* 2(1): 183−202.

Bertsekas, D. P. 1997. Nonlinear programming. *J. Oper. Res. Soc.* 48(3): 334−334.

Bonawitz, K.; Eichner, H.; Grieskamp, W.; Huba, D.; Ingerman, A.; Ivanov, V.; Kiddon, C.; Konecny, J.; Mazzocchi, S.; McMahan, H. B.; et al. 2019. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*.

Chen, Y.; Ning, Y.; Slawski, M.; and Rangwala, H. 2019. Asynchronous online federated learning for edge devices with non-IID data. *arXiv preprint arXiv:1911.02134*. Fallah, A.; Mokhtari, A.; and Ozdaglar, A. 2020. Personal-ized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*.

Ghadimi, S.; Lan, G.; and Zhang, H. 2016. Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. *Math. Program.* 155(1-2): 267−305. He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 770−778.

Hu, R.; Guo, Y.; Li, H.; Pei, Q.; and Gong, Y. 2020. Personalized federated learning with differential privacy. *IEEE Internet Things J.* 7(10): 9530−9539.

Jiang, Y.; Kone$^{\vee}$cn`y, J.; Rush, K.; and Kannan, S. 2019. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceed. IEEE* 86(11): 2278−2324.

Li, T.; Sahu, A. K.; Talwalkar, A.; and Smith, V. 2020. Federated learning: Challenges, methods, and future directions. *IEEE Signal Process. Mag.* 37(3): 50−60.

Li, T.; Sahu, A. K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; and Smith, V. 2018. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*.

Lu, Y.; Huang, X.; Dai, Y.; Maharjan, S.; and Zhang, Y. 2019. Differentially private asynchronous federated learning for mobile edge computing in urban informatics. *IEEE Trans. Industr. Inform.* 16(3): 2134−2143.

McMahan, H. B.; Moore, E.; Ramage, D.; Hampson, S.; et al. 2016. Communication-efficient learning of deep networks from decentralized data. In *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*.

Mohri, M.; Sivek, G.; and Suresh, A. T. 2019. Agnostic federated learning. *arXiv preprint arXiv:1902.00146*. Mosteller, F. 2006. On some useful inefficient statistics. In *Selected Papers of Frederick Mosteller*, 69−100. Springer. Nedi´c, A.; Olshevsky, A.; and Rabbat, M. G. 2018. Net-work topology and communication-computation tradeoffs in decentralized optimization. *Proc. IEEE* 106(5): 953−976. Patarasuk, P.; and Yuan, X. 2009. Bandwidth optimal all-reduce algorithms for clusters of workstations. *J. Parallel Distrib. Comput.* 69(2): 117−124.

Ram, S. S.; Nedi´c, A.; and Veeravalli, V. V. 2010. Distributed stochastic subgradient projection algorithms for convex optimization. *J. Optim. Theory Appl.* 147(3): 516−545. Smith, V.; Chiang, C.-K.; Sanjabi, M.; and Talwalkar, A. S.

2017. Federated multi-task learning. *Adv. Neural. Inf. Pro-cess. Syst.* 30: 4424−4434.

Wu, Q.; He, K.; and Chen, X. 2020. Personalized federated learning for intelligent IoT applications: A cloud-edge based framework. *IEEE Comput. Graph Appl.* .

Wu, R.; Wai, H.-T.; and Ma, W.-K. 2020. Hybrid inexact BCD for coupled structured matrix factorization in hyper-spectral super-resolution. *IEEE Trans. Signal Process.* 68: 1728−1743.

Yang, Q.; Liu, Y.; Chen, T.; and Tong, Y. 2019. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.* 10(2): 1−19.

Yang, Z.; Gang, A.; and Bajwa, W. U. 2020. Adversary-resilient distributed and decentralized statistical inference and machine learning: An overview of recent advances under the Byzantine threat model. *IEEE Signal Process. Mag.* 37(3): 146−159.

Zhang, S.; Choromanska, A. E.; and LeCun, Y. 2015. Deep learning with elastic averaging SGD. In *Adv. Neural Inf. Process. Syst.*, 685−693.