**联邦学习系列|背景和基础算法**

**1 背景**

联邦学习（Federated Learning）一词最早来源于2015年的论文《Federated Optimization : Distributed Optimization Beyond the Datacenter》[[1]](https://zhuanlan.zhihu.com/p/656102264" \l "ref_1)，在2017年的AISTATS上谷歌正式发表的论文《Communication-Efficient Learning of Deep Networks from Decentralized Data》[[2]](https://zhuanlan.zhihu.com/p/656102264" \l "ref_2)被认为是开山之作。在此之前分布式机器学习已有很多工作，其主要解决的是使用多个计算资源对机器学习模型进行分布式训练等问题。比如Pytorch里的torch.distributed库，可以使用多张显卡训练一个模型。而联邦学习可以看成一类特殊的分布式机器学习。

**1.1 提出的动机**

上述提及的谷歌论文的作者在摘要中说

However, this rich data is often privacy sensitive, large in quantity, or both, which may preclude logging to the data center and training there using conventional approaches. We advocate an alternative that leaves the training data distributed on the mobile devices, and learns a shared model by aggregating locally-computed updates. We term this decentralized approach Federated Learning.

也就是说，提出联邦学习(Federated Learning)的主要目的是在使用移动设备进行分布式训练的基础上保护用户的数据隐私

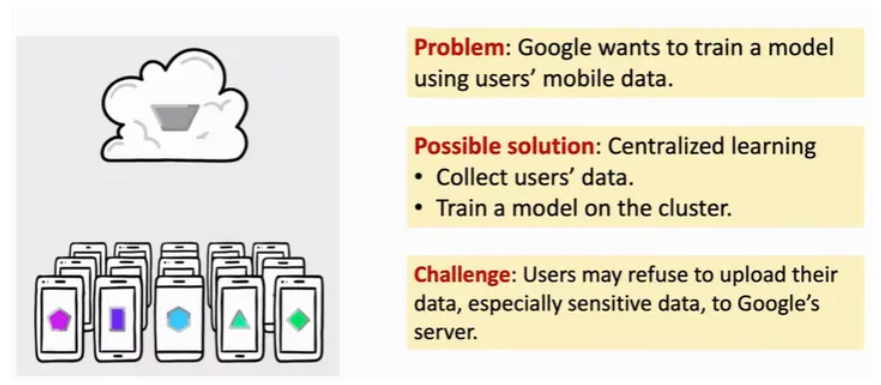


图1 联邦学习提出的动机1 https://github.com/wangshusen/DeepLearning/

一个应用场景是医院、银行等机构，这些机构每家机构都会有用户的数据，但是这些机构仅仅使用自家的用户数据并不能训练出一个很好的模型，将不同机构的数据以不泄露用户隐私的方式训练一个大的机器学习模型，这也是联邦学习提出的动机之一

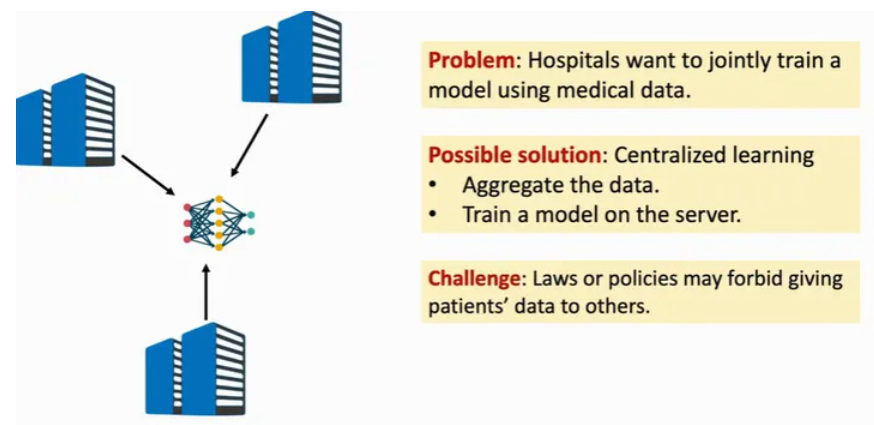


图2 联邦学习提出的动机2 https://github.com/wangshusen/DeepLearning/

**1.2 联邦学习基本流程**

开门见山地说一下联邦学习的基本流程，这里以横向联邦学习算法为例简要说明，后面再介绍联邦学习的分类

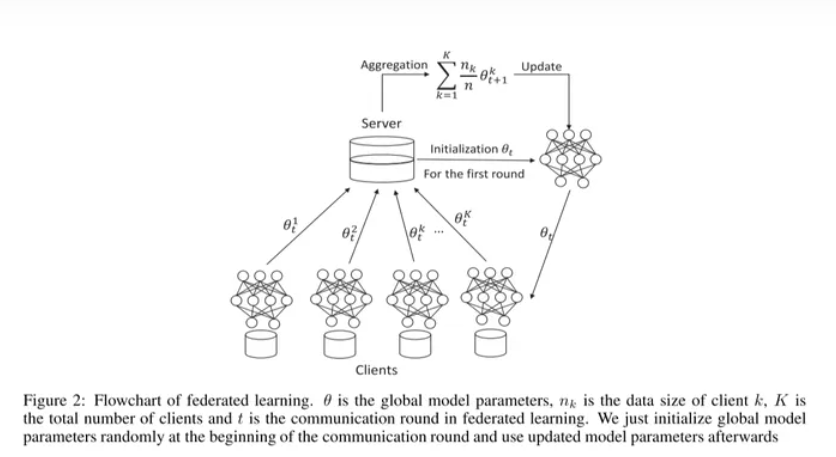


图3 联邦学习的基本原理 https://arxiv.org/abs/2009.05868

对于联邦学习，有一个中心的**服务器Server**，还有若干个移动设备**客户端Clients**

既然要保护用户数据隐私，训练联邦学习时，**客户端不会直接向服务器传输原始的用户数据，而是传输模型参数**（比如梯度）。服务器向移动用户客户端分发模型，用户的移动设备客户端下载模型并在本地进行训练，训练完向服务器传输模型参数，服务器对模型参数进行汇聚，更新服务器模型，然后再将更新后的模型分发给各客户端，客户端再重复训练，上传模型参数到服务器，服务器对模型参数进行汇聚，更新服务器模型，再将更新后的模型分发给各客户端，不断重复上述过程，直到模型训练完成，这就是联邦学习训练的基本流程。

Step 1 在服务器上初始化全局模型参数，并将模型分发给参与连接的各用户客户端  
Step 2 所有参与训练的客户端从服务器上下载模型，并使用本地数据进行训练多个epochs，训练完成后，将模型参数和梯度加密上传给服务器  
Step 3 服务器汇聚上传的客户端模型参数，并更新服务器全局模型参数  
Step 4 重复上面的Step2-3，直至模型训练完成[[3]](https://zhuanlan.zhihu.com/p/656102264" \l "ref_3)

**1.3 联邦学习优化问题**

考虑到训练时使用的是移动设备，比如说手机。这类设备有个特点，就是设备在训练时经常会掉线，比如你的手机一般会在充电，你睡觉和不计量的Wi-Fi连接时自愿参与优化训练模型，不会在你高强度使用时进行训练模型，而手机的使用时间是不可控的。移动设备的经常掉线也就会导致联邦学习训练时产生一些问题。同时不同用户使用不同设备产生不同数据也会导致一些问题。根据论文的总结，一般联邦学习有如下优化问题：

* 训练数据不满足独立同分布——Non-IID

联邦学习训练时，在给定的客户端上的训练数据取决于各种各样不同的移动设备用户，每个用户的设备里的数据类型不尽相同（比如甲手机里全是自拍，乙的手机里全是风景照片），数据集的选取等都是随机的，不满足独立同分布

* 训练数据负载不平衡——Unbalanced

由于不同用户在设备上的使用情况不同，有些用户在某些app上的数据多，有些用户的数据少

* 存在大量规模的分布式客户端——Massively distributed

与传统分布式机器学习相比，联邦学习分布式客户端数量更庞大，如抖音用户有上亿规模

* 存在通信限制问题——Limited communication

比如有很多手机客户端信号不是很好，网速较慢或者会经常掉线；同时，用户对自己的设备有绝对的控制权[[4]](https://zhuanlan.zhihu.com/p/656102264#ref_4)（就像联邦制度各个联邦有对自己的自治权），用户可以随时决定停止与服务器进行通信；联邦学习的通信代价很大，不同客户端与服务器远程连接，并不稳定

**1.4 FedSGD vs FedAVG**

前面提到联邦学习的几个特征，特别是由于存在通信限制的问题，联邦学习需要做的就是**减少通信次数**。比如你的手机经常断网，那么通信次数更少的算法在联邦学习上大概率比通信次数多的算法要好，当然通信次数少的代价可能是模型的计算量可能提高了。下面就比较一下传统的并行梯度下降FedSGD与谷歌使用的联邦平均算法FedAVG

* FedSGD

首先是联邦学习并行梯度下降的方法，存在服务器节点Server和客户端用户节点Worker

对于客户端第 i 个用户节点 i -th Worker，首先会从服务器接收模型参数 w ；然后使用模型参数w和本地数据计算出梯度 ; 最后将梯度  发送给服务器

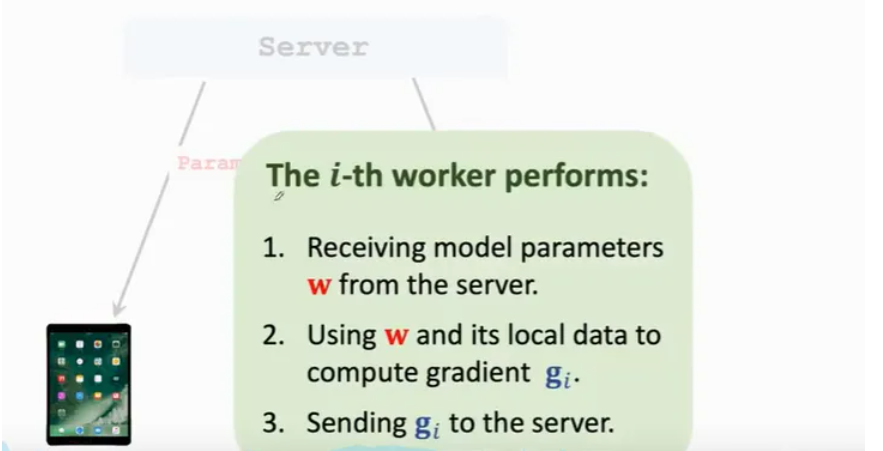


图4 并行梯度下降算法客户端操作 https://github.com/wangshusen/DeepLearning/

对于服务器节点Server，首先会接收所有来自Worker的梯度   到   ；然后将所有梯度加起来得到总的梯度 g ; 最后使用 g 更新模型参数 w

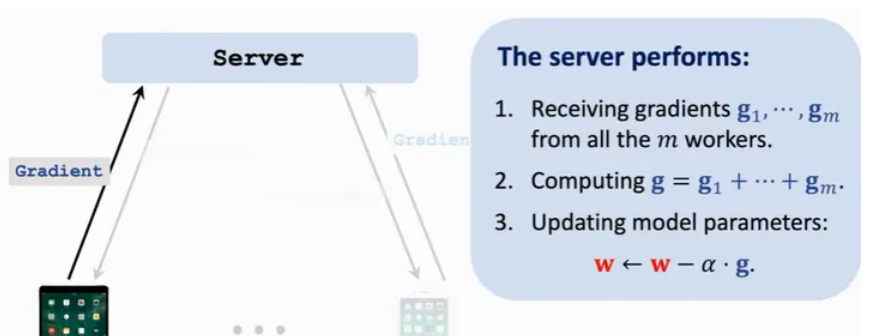


图5 并行梯度下降算法服务器操作 https://github.com/wangshusen/DeepLearning/

然后就是重复上述过程，直到模型收敛

* **FedAVG**

FedAVG即谷歌在论文《Communication-Efficient Learning of Deep Networks from Decentralized Data》[[2]](https://zhuanlan.zhihu.com/p/656102264" \l "ref_2)里提出的算法，实现如下

对于客户端第 � 个用户节点 � -th Worker，首先会从服务器接收模型参数 � ；然后会重复E次进行下面两步 ：

a）使用 � 和本地数据计算梯度 �

b) 使用本地计算得到的梯度 � 对 � 进行更新

经过多次更新后的参数 ��~ = � 再传递给服务器

一般而言重复次数E取5-20个左右，当E取1时和FedSGD算法相比，客户端所做的操作除了发送的参数不同其他是一样的

图6 FedAVG算法客户端操作 https://github.com/wangshusen/DeepLearning/

对于服务器节点Server，服务器先接收各个Worker节点的参数��~，然后将**所有**��~**参数加起来求平权值**，更新服务器的模型参数 �

图7 FedAVG算法服务器操作 https://github.com/wangshusen/DeepLearning/

对比FedSGD和FedAVG的通信效率，FedAVG收敛的更快，从而降低了达到收敛所需的通信次数

图8 FedAVG和FedSGD通信次数-Loss图

但是用训练轮次来衡量FedAVG的效率的话，FedAVG达到收敛所需epochs更多,也就是FedAVG用更大的计算量换取了更高的通信效率

值得注意的是，Li等人[[5]](https://zhuanlan.zhihu.com/p/656102264" \l "ref_5)证明了FedAVG算法可以收敛在**非独立同分布**的数据上，也有b站up主读相关论文

[丸一口投稿视频-丸一口视频分享-哔哩哔哩视频​space.bilibili.com/3461572290677609/video](https://link.zhihu.com/?target=https%3A//space.bilibili.com/3461572290677609/video)

总而言之，FedSGD通过可靠的网络连接让客户端频繁上传梯度，服务端进行梯度更新来保证模型使用更少的epochs收敛；而FedAVG通过使用更少的通信代价来进行模型更新，有一些的性能损失，并且由于客户端的不稳定性不能保证模型最终收敛[[6]](https://zhuanlan.zhihu.com/p/656102264#ref_6)

图9 FedAVG和FedSGD训练轮次-Loss图

**1.5 联邦学习的隐私保护与鲁棒性**

联邦学习说是保护用户数据隐私，不直接传输用户数据，但是真的保护隐私了吗？

图10 以最小二乘法为例表示梯度

可以看出，我们如果假设损失函数 �(��,��,��) 如图所示，那么对于给定的用户数据 ��,它的梯度就是对 �� 进行线性变换，其中（ ����−�� )为常数；所以传递了梯度信息，其实也就是**间接上泄露了用户数据，**有心之人就可以根据梯度来反推出用户数据。

所以，针对这一问题，比如**同态加密、差分隐私、模型聚合、多方安全计算**、标签翻转和一些密码学的方法被应用于联邦学习以解决联邦学习中的隐私保护。

联邦学习也存在拜占庭将军问题，也就是联邦学习的结点中出了一个叛徒的问题，也有一些文章去解决这些问题，这里暂时不关心这些内容。

图11 拜占庭将军问题

**1.6 联邦学习的分类**

联邦学习经过多年的研究也产生了很多类别，国内的学者一般会将联邦学习分为**横向联邦学习**（Horizontal Federated Learning)和**纵向联邦学习**（Vertical Federated Learning)，除此之外还有联邦迁移学习、联邦强化学习、**联邦学习大模型**等诸多分类，这些在我后续学习的过程中补充

* **横向联邦学习**

也称为**按样本划分**的联邦学习，其主要特点是**数据集特征和标签信息相同，但样本ID不同。**例如某地有两家银行，这两家银行的用户不同（样本不同），但是两家银行的业务是一样的，并且有相同类型的用户数据（比如两家银行都有用户年龄和存款数额的数据），将两家银行的数据不离开本地在云服务器的协调下训练全局模型的过程就是横向联邦学习的过程。

上面介绍联邦学习的基本流程时就是以横向联邦学习为例介绍的，每台机器下都是**相同且完整的模型**，且机器之间不直接传输用户数据，在预测时每台机器也可以**独立预测**，可以把这个过程看作成基于样本的分布式模型训练。[[7]](https://zhuanlan.zhihu.com/p/656102264#ref_7)

* **纵向联邦学习**

纵向联邦学习的特点是**各数据集特征和标签信息不同，但样本 ID 信息相同。**例如现有一家电商和一家银行，出于某些目的，电商和银行想要共同训练一个模型，电商和银行的用户是相同的（样本相同），但是电商和银行的业务不同，掌握的数据是不同的，例如银行掌握用户存款，电商掌握消费记录，双方不直接传输用户数据来训练一个根据存款推荐商品的模型，这就是一个纵向联邦学习的过程。

纵向联邦学习的训练流程和横向有一些区别，主要运用了一些加密技术，下面纵向联邦学习的流程参考[[6]](https://zhuanlan.zhihu.com/p/656102264#ref_6)

**Part1 加密实体对齐**  
�1 公司、 �2 公司和 �3 公司的用户群体不同，系统使用一种基于加密的**用户ID对齐**技术来确保 �1,�2 和 �3 不需要暴露各自的原始数据便可以对齐用户。  
**Part2 加密模型训练**  
--Step1 协调者 �3 创建密钥对，并将公共密钥发送给�1,�2 和 �3公司  
--Step2 �1,�2 和 �3对中间结果进行加密和交换，中间结果用于帮助计算梯度和损失值  
--Step3 �1,�2 和 �3计算加密梯度并附加掩码，�1,�2 和 �3将加密结果发送给 �3  
--Step4 �3 对梯度和损失信息进行解密，并将结果发送给�1,�2 和 �3，�1,�2 和 �3解除梯度信息掩码，并根据梯度信息来更新各自的模型参数

图12 纵向联邦学习图示 https://arxiv.org/abs/2211.12814

关于纵向联邦学习也可参考

[14.3. 纵向联邦学习 — 机器学习系统：设计和实现 1.0.0 documentation (openmlsys.github.io)​openmlsys.github.io/chapter\_federated\_learning/vertical\_fl.html](https://link.zhihu.com/?target=https%3A//openmlsys.github.io/chapter_federated_learning/vertical_fl.html)

当然，我在阅读论文时发现，其实更多的论文会将联邦学习划分为**跨机构(Cross-silo)**联邦学习和**跨设备(Cross-device)**的联邦学习，它们的特点如图所示，详细区别可以参考[《A Field Guide to Federated Optimization》](https://link.zhihu.com/?target=https%3A//arxiv.org/abs/2107.06917" \t "_blank)

还有一些其他分类，比如按是否有中心客户端分的：

**2 联邦学习简单优化算法**

由于是刚学联邦学习，只例举一些简单点的算法，有些算法对数学的要求比较高（wo kan bu dong)就暂时不例举了

**2.1 FedAVG**

FedAVG作为联邦学习最基本的算法，前面已经介绍过，用公式表达如下：

* K个用户client在本地先聚合**E**次：（E一般取5-20，小写的k表示下标）

��+1�←��−���

* 然后传递给server进行对参数 � 最终聚合， �� 代表每个客户端数据长度， �� 代表所有客户端数据长度，即聚合是对 ��+1� 做**加权平均**

��+1←∑�=1�������+1�

然后就是重复上述过程，直到训练结束

用论文里的算法框架表示的话是这样的

图13 FedAVG算法框架

具体来讲，对于**ClientUpdate（k,w)**，k可以看作第k个客户端， � 是模型参数

我们先将客户端的数据集 �� 进行分块，以**batchsize=B**为分块大小  
**外层for循环**表示客户端的训练轮次从第1轮训练到**第E轮**  
**内层for循环**表示对每个小batch进行梯度下降，对参数 � 的更新，直到所有batch更新完  
循环结束后返回参数 � 给服务器

对于**Server executes**

首先初始化服务器的参数 �0  
**外层for循环**是指定服务器的训练轮次，未达到训练轮次就执行下面的语句  
每次取m=max(C\*K,1)个客户端，其中**C代表所选取客户端的比例**，K代表总的客户端数目  
�� 代表这随机选取的m个客户端  
**内层for循环**是对这m个客户端里的每个客户端进行并行的操作  
具体而言是进行客户端的本地梯度下降得到参数 ��+1�  
然后对所有客户端的参数 ��+1� 求加权平均得到参数 ��+1 (其中 �� 代表每个客户端所拥有的数据长度， �� 代表所有客户端的数据长度）

代码框架也可参考

[TsingZ0/PFL-Non-IID: Personalized federated learning simulation platform with non-IID and unbalanced dataset (github.com)​github.com/TsingZ0/PFL-Non-IID](https://link.zhihu.com/?target=https%3A//github.com/TsingZ0/PFL-Non-IID)

**2.2 FedAVGM**

FedAVGM（Federated Averaging with **Server Momentum**）是论文《Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification》[[8]](https://zhuanlan.zhihu.com/p/656102264" \l "ref_8)的工作

单纯的比较算法方面与FedAVG的区别，只是在服务器端**引入了动量来更新权重**�**,提升了FedAVG算法对非独立同分布数据的训练效果**

首先，我们需要用一个新的变量 � 来表示动量，新更新的动量 � 等于引入的可调参数 � 乘以原有的动量 � ，再加上 △� ；其中 △� 等于服务器原有的全局参数 � -从客户端新上传的参数 ����

服务器的全局参数 � 则等于服务器原有的参数 � -动量 �

�←��+△�

�←�−�

用[FedNTD](https://link.zhihu.com/?target=https%3A//github.com/Lee-Gihun/FedNTD" \t "_blank)里代码表示如下

**def** **update\_momentum\_weights**(m\_flat, dg\_flat, new\_flat, beta**=**0.9):

m\_flat\_new **=** beta **\*** m\_flat **+** (dg\_flat **-** new\_flat)

ag\_flat **=** dg\_flat **-** m\_flat\_new *# new\_flat -> dg\_flat*

**return** m\_flat\_new, ag\_flat

除了添加了动量，FedAVGM这篇论文里还基于迪利克雷分布，提出了一种FL中**Non-IID数据的生成方法**，并做了不少实验

参考[论文原文](https://link.zhihu.com/?target=https%3A//arxiv.org/abs/1909.06335" \t "_blank)

**2.3 FedProx**

FedProx算法发表于2020年MLSys的论文《Federated Optimization in Heterogeneous Networks》[[9]](https://zhuanlan.zhihu.com/p/656102264" \l "ref_9)

论文的摘要里提到了联邦学习的两大挑战

(1) significant variability in terms of the systems characteristics on each device in the network (**systems heterogeneity**) ——不同设备之间存在通信和计算能力的差异  
(2) non-identically distributed data across the network (**statistical heterogeneity**)——不同用户的数据不满足独立同分布

那么这两个问题会造成什么影响呢？

* **systems heterogeneity**

对于systems heterogeneity，也就是不同设备之间存在通信和计算能力的差异。在FedAvg中，被选中的客户端在本地都训练相同的epoch，虽然作者指出提升epoch可以有效减小通信成本，但较大的epoch下，可能会有很多设备**无法按时完成训练**，而无法完成训练的这些客户端**会被drop掉**。无论是直接drop掉这部分客户端的模型还是直接利用这部分未完成的模型来进行聚合，都将对最终模型的收敛造成不好的影响，对模型的参数产生偏移。

* **statistical heterogeneity**

对于statistical heterogeneity，也就是不同用户的数据不满足独立同分布。本地客户端在训练模型时的参数不仅会与其他客户端训练的参数有较大偏差，也会和服务器全局模型的参数有偏差，这就要使得本地客户端模型的更新不能太偏离服务器全局模型。

论文解决statistical heterogeneity的方法是引入了所谓的**Proximal term**

在联邦学习中我们所优化的目标是最小化损失函数 �(�) ，其中 �� 代表每个客户端数据所占的权重(概率），即FedAVG算法里的 ���� (其中 �� 代表每个客户端所拥有的数据长度， �� 代表所有客户端的数据长度）； �� ≥ 0, and ∑�=1���=1

现在，我们在这个损失函数后面加上Proximal term，**对偏离服务器模型的参数进行正则化**，用于减少Non-IID数据对模型的影响，提高了模型在面对Non-IID数据时的效果

�  ��� ℎ�(�;��)=�(�)+�2‖�−��‖2

其中 �2‖�−��‖2 中的 �−�� 即全局模型的参数与客户端模型的参数之差，如果是Non-IID分布的数据， �−�� 的 �2 范数可能会很大，最小化新的目标函数 ℎ�(�;��) 也就是最小化这种Non-IID数据造成的模型参数偏移的影响； � 是可调的参数，我们调小 � 即降低约束效果；系数有1/2是因为两边求梯度时可以消掉

图14 定义1

用[FedNTD](https://link.zhihu.com/?target=https%3A//github.com/Lee-Gihun/FedNTD" \t "_blank)里代码表示如下

**def** **\_proximal\_term**(self, dg\_model, model, mu):

"""Proximal regularizer of FedProx"""

vec **=** []

**for** \_, ((name1, param1), (name2, param2)) **in** enumerate(

zip(model**.**named\_parameters(), dg\_model**.**named\_parameters())

):

**if** name1 **!=** name2:

**raise** **RuntimeError**

**else**:

vec**.**append((param1 **-** param2)**.**view(**-**1, 1))

all\_vec **=** torch**.**cat(vec)

square\_term **=** torch**.**square(all\_vec)**.**sum()

proximal\_loss **=** 0.5 **\*** mu **\*** square\_term

**return** proximal\_loss

除了引入Proximal term来解决statistical heterogeneity，作者还引入了所谓���**-inexactness**来解决systems heterogeneity，其中如果 �∗ 满足下式则称为 min�ℎ�(�;��) 的 ��� -**inexact solution**. �∈[0,1]

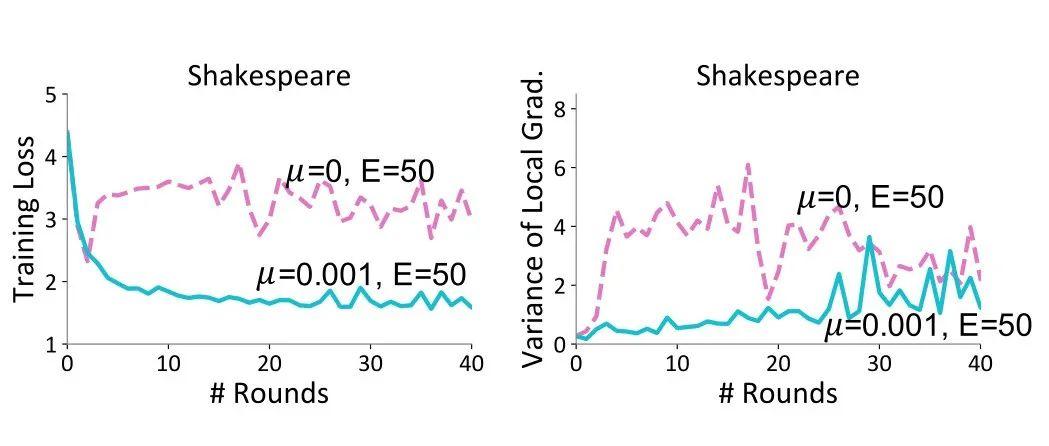
||∇ℎ�(�∗;��)||≤||���∇ℎ�(��;��)||∇ℎ�(�;��)=∇��(�)+�(�−��)

图15 定义2

作者在论文里说， ��� 通过测量本地客户端 � 第 � 轮的计算量这种方式来**动态调整本地迭代次数**， �∗是满足上面式子的**不准确解**，怎么解的**我没搞明白**；客户端在向服务器传递参数时不再是传递 �而是传递 �∗ ， 从而可以解决不同设备计算能力的差异导致的systems heterogeneity，不过这点我找到的FedProx的代码里并没有体现，应该是作者为了写论文提出的理论可行的东西

图16 FedProx算法框架

参考[论文原文](https://link.zhihu.com/?target=https%3A//arxiv.org/abs/1812.06127)以及

[联邦学习算法FedProx的PyTorch实现​mp.weixin.qq.com/s/sLc61t8BpS6yBOyGhavm2w](https://link.zhihu.com/?target=https%3A//mp.weixin.qq.com/s/sLc61t8BpS6yBOyGhavm2w)

**参考**

1. [**^**](https://zhuanlan.zhihu.com/p/656102264#ref_1_0)Federated Optimization: Distributed Optimization Beyond the Datacenter <https://arxiv.org/pdf/1511.03575.pdf>
2. ^[**a**](https://zhuanlan.zhihu.com/p/656102264#ref_2_0)[**b**](https://zhuanlan.zhihu.com/p/656102264#ref_2_1)Communication-Efficient Learning of Deep Networks from Decentralized Data <https://arxiv.org/abs/1602.05629>
3. [**^**](https://zhuanlan.zhihu.com/p/656102264#ref_3_0)FROM FEDERATED LEARNING TO FEDERATED NEURAL ARCHITECTURE SEARCH: A SURVEY <https://arxiv.org/abs/2009.05868>
4. [**^**](https://zhuanlan.zhihu.com/p/656102264#ref_4_0)王树森联邦学习 <https://www.bilibili.com/video/BV1YK4y1G7jw?p=7&vd_source=c6447bef06fe76ccb22a54330bab6703>
5. [**^**](https://zhuanlan.zhihu.com/p/656102264#ref_5_0)On the Convergence of FedAvg on Non-IID Data <https://arxiv.org/abs/1907.02189>
6. ^[**a**](https://zhuanlan.zhihu.com/p/656102264#ref_6_0)[**b**](https://zhuanlan.zhihu.com/p/656102264#ref_6_1)《联邦学习》杨强等 电子工业出版社 <https://book.douban.com/subject/35062813/>
7. [**^**](https://zhuanlan.zhihu.com/p/656102264#ref_7_0)详解联邦学习 <https://zhuanlan.zhihu.com/p/79284686>
8. [**^**](https://zhuanlan.zhihu.com/p/656102264#ref_8_0)Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification <https://arxiv.org/abs/1909.06335>
9. [**^**](https://zhuanlan.zhihu.com/p/656102264#ref_9_0)Federated Optimization in Heterogeneous Networks <https://arxiv.org/abs/1812.06127>