联邦学习

Heiko Ludwig, Nathalie Baracaldo

2023 年 3 月 23 日

目录

第一章　联邦学习介绍

1.0.1　摘要

联邦学习[**?**]（Federated Learning, FL）是一种机器学习的方法，其训练数据不是集中管理的。数据由参与联邦学习过程的数据方保留，不与任何其他实体共享。这使得联邦学习成为一种越来越流行的机器学习任务的解决方案。对于这些任务来说，无论是出于隐私、监管还是实际的原因，将数据集中到一个集中的存储库是有问题的。在本章中，我们介绍了联邦学习的基本概念，概述了它的应用案例，并从机器学习、分布式计算和隐私的角度讨论了它。我们还提供了一个介绍，以深入探讨后续章节中所涉及的事项。

1.0.2　概述

机器学习（Machine Learning, ML）已经成为开发认知和分析功能的关键技术，而这些功能在算法上很难得到有效的开发。随着深度神经网络（Deep Neural Networks, DNN）和能有效训练复杂网络的计算硬件的出现，计算机视觉、语音识别和自然语言理解方面的应用取得了飞跃性的进展。此外，经典的机器学习技术，如决策树、线性回归和支持向量模型（SVMs）也得到了更多的应用，特别是与结构化数据有关的应用。

机器学习的应用在很大程度上取决于高质量训练数据的可用性。但有时，隐私方面的考虑使训练数据无法被带到一个中央数据存储库中，为机器学习过程进行策划和管理。联邦学习（FL）是在[28]中首次以这个名字提出的一种方法，在不同地点的训练数据上训练ML模型，不需要集中收集数据。

这种不愿意使用中央数据存储库的一个重要驱动因素是不同司法管辖区的消费者隐私法规。欧盟的《一般数据保护条例》（GDPR）[50]、《健康

保险可携性和责任法案》（HIPAA）[53]和《加州消费者隐私法案》（CCPA）[48]是收集和使用消费者数据的监管框架的范例。此外，关于数据泄露的新闻报道提高了人们对存储敏感消费者数据所带来的责任的认识[9, 42, 43, 51]。联邦学习为使用数据提供了便利，而实际上不需要将其存储在一个中央存储库中，从而减轻了这种风险。监管也限制了数据在不同国家等管辖区之间的流动。这是因为考虑到其他国家的数据保护可能不足或与国家安全有关，要求关键数据保留在岸上[40]。国家和地区的法规对在不同市场拥有子公司但希望使用其所有数据来训练模型的国际公司构成了挑战。除了监管要求，从不同地点的数据中学习也可能只是实用。糟糕的通信连接和由传感器或电信设备收集的大量数据会使中央数据收集不可行。联邦学习也使不同的公司能够在不泄露其商业秘密的情况下，共同创建互利的模型。

那么联邦学习是如何工作的呢？在联邦学习方法中，一组控制着各自训练数据的不同各方，合作训练一个机器学习模型。他们这样做并不与其他各方或任何其他第三方实体分享他们的训练数据。合作的各方在文献中也被称为客户端或设备。当事人可以是各种各样的东西，包括消费者设备，如智能手机或汽车，但也包括不同供应商的云服务，在不同国家处理企业数据的数据中心，公司内部的应用仓，或嵌入式系统，如汽车厂的制造机器人。

虽然联邦学习协作可以以不同的方式进行，但其最常见的形式概述于图1.1。在这种方法中，一个聚合器，有时被称为服务器或协调器，促进了合作。各方根据他们的私人训练数据进行本地训练。当他们的本地训练完成后，他们将他们的模型参数作为模型更新发送到聚合器。模型更新的类型取决于要训练的机器学习模型的类型；例如，对于一个神经网络，模型更新可能是网络的权重。一旦聚合器收到各方的模型更新，它们就可以被合并到一个共同的模型中，这个过程我们称之为*模型融合*。在神经网络的例子中，这可以像FedAvg算法[38]中提出的那样，简单地对权重进行平均化。然后，合并后的模型将作为模型更新再次分发给各方，以形成下一轮学习的基础。这个过程可以重复多轮，直到训练过程收敛。聚合器的作用是协调各方的学习过程和信息交流，并执行融合算法，将各方的模型参数合并为一个共同的模型。融合过程的结果是一个基于各方训练数据的模型，而训练数据从不共享。

联邦学习方法似乎与集群上的分布式学习有关[15]，这是大型机器学习

任务的一种常见方法。分布式学习使用一个计算节点集群来分担机器学习的计算工作，从而加速学习过程。分布式学习通常使用一个参数服务器来汇总各节点的结果，这与联合模型中并无不同。然而，它在一些重要方面是不同的。在联邦学习中，数据的分布和数量不是集中控制的，如果所有的训练数据都是私有的，可能是未知的。我们不能对各方数据的独立同分布性（IID）做出假设。同样地，一些当事方可能比其他当事方拥有更多的数据，导致当事方之间数据集的不平衡。在分布式学习中，数据被集中管理，并以分片形式分布到不同的节点，中央实体了解数据的随机属性。在设计联邦学习训练算法时，必须考虑到各方数据的不平衡性和非独立同分布性。

相反，在联邦学习中，各方的数量可能会有所不同，这取决于用例。在一家跨国公司的不同数据中心的数据集上训练一个模型，可能有少于10个当事人。这通常被称为企业[35]或跨语境用例[26]。在一个移动电话应用的数据上进行训练，可能会有数以亿计的各方贡献。这通常被称为跨设备用例[26]。在企业用例中，一般来说，在每一轮中考虑所有或大多数当事方的模型更新是很重要的。在设备用例中，每一轮联邦学习只包括全部设备中的一个潜在的大子样本。在企业用例中，联邦学习过程考虑了相关各方的身份，并可以在训练和验证过程中使用这些。在跨设备的使用案例中，当事人的身份通常并不重要，而且一个当事人可能只参与一轮训练。

在设备使用案例中，比起企业场景，考虑到参与者的数量众多，可以假设一些设备的通信故障。手机可能关闭，或者设备可能处于网络覆盖不佳的地区。这可以通过对各方进行抽样调查和设置执行聚合的时间限制，或其他缓解技术来管理。在企业用例中，由于参与者人数较少，个人的贡献是相关的，所以必须仔细管理通信故障。

在本章的其余部分，我们将对联邦学习进行概述。我们在下一节中对所使用的主要概念进行了正式介绍。之后，我们从三个重要的角度讨论联邦学习，每个角度都有一个单独的章节。首先，我们从机器学习的角度讨论联邦学习；然后，我们通过概述威胁和缓解技术来涵盖安全和隐私的角度；最后，我们对联合学习的系统角度进行了概述。这将为本书的其余部分提供一个起点。

1.0.3 概念和术语

像任何机器学习任务一样，联邦学习在训练数据$\mathcal{D}$上训练一个代表预

测函数f的模型$\mathcal{M}$。$\mathcal{M}$可以有一个神经网络或任何其他非神经模型的结构。与集中式机器学习不同的是，$\mathcal{D}$被划分在n个当事方$P = \{P_1, P_2, \cdots, P_n\}$，其中每一方$P_k \in P$拥有一个私人训练数据集$\mathcal{D}_k$。一个联邦学习过程涉及一个聚合器$A$和一组当事人$P$。必须注意的是，$\mathcal{D}_k$只能由当事人$P_k$访问。换句话说，除了自己的数据集，没有任何一方知道其他的数据集，而A对任何数据集都没有了解。

图1.2显示了联邦学习过程是如何在这个抽象层面上进行的。为了训练一个全局机器学习模型$\mathcal{M}$，聚合器和各方执行一个联邦学习算法，该算法以分布式方式在聚合器和各方上执行。主要的算法组件是每一方的本地训练函数$\mathcal{L}$，它在数据集$\mathcal{D}_k$上进行本地训练，以及聚合器的融合函数$\mathcal{F}$，它将每一方的$\mathcal{L}$的结果结合成一个新的联合模型。可以有一组本地训练和融合的迭代，我们称之为轮次，使用索引t。算法的执行通过在各方和聚合器之间发送消息来协调。整个过程运行如下:

1. 这个过程从聚合器开始。为了训练模型，聚合器使用一个函数$\mathcal{Q}$，该函数将上一轮训练$\mathcal{M}_{t-1}$的模型作为输入，并为当前回合生成一个查询q_t。当这个过程开始时，$\mathcal{M}_0$可能是空的或只是随机的种子。另外，一些联邦学习算法可能包括$\mathcal{Q}$的额外输入，并可能为每一方定制查询，但为了讨论的简单性，在不损失一般性的情况下，我们使用这种更简单的方法。

2. 查询q_t被发送到各方，并要求提供关于他们各自的本地模型的信息或关于各方数据集的汇总信息。查询的例子包括对神经网络梯度或模型权重的请求，或对决策树计数的请求。

3. 当收到q_t时，本地训练过程执行本地训练函数$\mathcal{L}$，该函数将查询q_t和本地数据集$\mathcal{D}_k$作为输入，并输出模型更新$r_{k,t}$。通常情况下，查询q_t包含了一方可以用来初始化本地训练过程的信息。例如，这包括新的共同模型$\mathcal{M}_t$的模型权重，以初始化本地训练，或不同模型类型的其他信息。

4. 当$\mathcal{L}$完成后，$r_{k,t}$从p_k方发回给聚合器A，后者收集所有各方的$r_{k,t}$。

5. 当聚合器收到所有预期各方的模型更新$R_t = (r_{1,t}, r_{2,t}, \cdots, r_{n,t})$时，它们被应用融合函数$\mathcal{F}$进行处理，该函数将$R_t$作为输入并返回$\mathcal{M}_t$。

这个过程可以在多轮中执行，并持续到满足终止标准为止，例如，最大的训练轮数$t_{\max}$已经过去，最终形成一个全局模型$\mathcal{M} = \mathcal{M}_{t_{\max}}$。所需的训练轮数可以有很大的不同，从Naive Bayes方法的单一模型合并到典型的基于梯度的机器学习算法的多轮训练。

本地训练函数$\mathcal{L}$，融合函数$\mathcal{F}$，和查询生成函数$\mathcal{Q}$通常是一个互补的集

合，被设计为共同工作。$\mathcal{L}$与实际数据集交互，进行局部训练，生成模型更新$R_{k,t}$。R_t的内容是$\mathcal{F}$的输入，因此，必须由$\mathcal{F}$来解释，它根据这个输入创建下一个模型$\mathcal{M}_t$。如果需要另一个回合，$\mathcal{Q}$就会创建另一个查询。

在接下来的章节中，我们将详细描述这一过程在训练神经网络、决策树和梯度增强树的情况下是如何发生的。

我们可以为联邦学习的这一基本方法引入不同的变体。在跨设备联邦学习的情况下，各方的数量往往很大，达到数百万。并非所有各方都参与每一轮。在这种情况下，$\mathcal{Q}$不仅决定了查询，而且决定了哪些$P_s \subset P$的当事方要包括在下一轮的查询中。党派的选择可以是随机的，基于党派的特点，或基于先前贡献的优点。

另外，对每一方的查询可能是不同的，$\mathcal{F}$需要在创建一个新的模型$\mathcal{M}_t$时整合不同查询的结果。

虽然在大多数情况下，具有单一聚合器的方法是最常用和实用的，但也有人提出了其他替代的联邦学习架构。例如，每一方P_k可能有它自己的、相关的聚合器A_k，查询其他各方；各方的集合可能在聚合器之间被分割，并且可能发生一个分层的聚合过程。在介绍的其余部分中，我们重点讨论常见的单一聚合器配置。

1.0.4 机器学习的视角

在这一节中，我们从机器学习的角度来看待联邦学习。联邦学习系统方法的选择–比如在查询中发送什么信息–影响着机器学习行为。我们在下面的小节中针对不同的机器学习范式讨论这个问题。

1.0.4.1 深度神经网络

深度神经网络已经变得非常流行，并且可以轻松的迁移至联邦学习。它的基本方法是在每一方进行本地训练，并在聚合器处融合本地训练结果。本地训练$\mathcal{L}$通常相当于在P_k方对神经网络进行常规的集中训练，并在每一轮t中对其参数w_k进行优化。我们在每一方P_k进行优化，在该方的训练数据集$\mathcal{D}_k$上最小化参数w_k（神经网络的权重向量）的损失函数l。

$$w_k^* = arg\min_{w_k} \frac{1}{\mathcal{D}_k} \sum_{(x_i,y_i)\in\mathcal{D}_k} l(w_k; x_i, y_i)$$

如果使用梯度下降算法，在给定回合t的每个纪元τ中，w_k的更新方式如下:

$$w_k^{t,\tau} := w_k^{t,\tau-1} - \eta_k \nabla l(w_k^{t,\tau-1}, X_k, Y_k)\Theta$$

损失函数l是基于本地数据$\mathcal{D}_k = (X_k, Y_k$计算的，可以是任何合适的函数，如常用的平均平方误差（MSE）。这一轮的参数$w_k^{t,\tau}$是使用党派特定的学习率η_k更新的。每一轮本地训练都以来自聚合器的新模型更新为种子$w_k^{t,0}$，它为每一轮的本地训练提供了新的起点。

例如，在建立一个联邦学习系统或一个特定的联邦学习项目时，我们可以就党派-地方超参数做出选择。 - 我们应该为党的本地梯度下降算法选择哪种批量大小：一个，即原始的随机梯度下降（SGD）；整个集合；或一个合适的小批量大小？ - 在向聚合器发送模型更新Rk,t之前要运行多少个本地历时？所有各方在每一轮都应该使用相同数量的历时？在每一方中只训练一个历时，可以防止本地模型wk与对方有很大的差异，但会导致更多的网络流量和频繁的聚合活动。运行多个历时，甚至在不同的一方使用不同数量的历时，会造成更大的差异，但可以用来适应各方计算能力的差异和训练数据集的大小。 - 我们为每一方选择多大的学习率ηk？各方数据分布的差异会使不同的学习率变得有利。 - 其他优化算法可能使用不同的局部超参数，如动量或衰变率[27]。深度神经网络已经变得非常流行，它的基本方法是在每一方进行本地训练，并在聚合器处融合本地训练结果，以相对简单的方式借给联邦学习。本地训练$\mathcal{L}$通常相当于在P_k方对神经网络进行常规的集中训练，并在每一轮t中对其参数w_k进行优化。

让我们考虑一个简单的联邦SGD的情况，如[38]所述，在这种情况下，与集中式SGD一样，每个新样本都会导致模型变动。聚合器将选择一方P_k，并向被选择的一方发送一个查询$q_{t,k} < w_t >$。P_k选择下一个训练样本$(x_i, y_i) \in \mathcal{D}_k$，并执行其本地训练$\mathcal{L}$，计算该样本的损失梯度$\nabla l(w_t, x_i, y_i)$。我们将把某一方$P_k$在特定回合$t$中的梯度称为$\mathcal{D}_k$中训练样本的平均梯度。

$$g_{k,t} := \frac{1}{|\mathcal{D}_k|} \sum_{(x_i,y_i)\in\mathcal{D}_k} \nabla l(w_k, x_i, y_i)$$

P_k将其作为回复$r_{k,t} < g_{t,k} >$返回给聚合器。然后，聚合器根据P_k的回复和聚合器的学习率，用模型权重计算新的查询内容：

$$w_{t+1} := w_t - \eta_a g_{k,t}\Theta$$

然后，下一轮开始，由聚合器选择另一方来贡献。在这种简单的方式下，它是相当没效率的，因为它引入了通信开销，并且没有利用并发训练

的优势。为了使联邦SGD更加有效率，我们可以在每一方进行小批量的训练，每一轮增加每一方的计算量。我们也可以在所有的或$P_s \subset P$的子集上同时进行训练，在计算新的模型权重时对各方的回复梯度进行平均化：

$$w_{t+1} := w_t - \eta_a \frac{1}{|K|} \sum_K g_{k,t} \Theta$$

虽然这比天真的方法更有效，但它仍然涉及到与聚合器的大量通信和潜在的协调延迟，当批次大小是完整的$\mathcal{D}_k$时，每一个纪元至少有一次协调延迟，当我们使用迷你批次时多次协调。

FedAvg，如[28]中提出的，通过利用每一方的独立处理，更加有效。每一方在回复前都会运行多个历时。与其用梯度回复，各方可以在每一方P_k直接计算一组新的权重$w_{t,k}$，使用一个共同的学习率η，并以$r_{k,t} < w_{k,t}, n_k >$、他们的模型和样本数进行回复。聚合器的融合算法F对每一方的参数进行平均，以每一方的样本数加权，用于下一轮：

$$w_{t+1} := \sum_{k \in K} \frac{n_k}{n} w_{k,t} \Theta$$

实验表明，这种方法对不同的模型类型表现良好[38]。FedAvg使用了方程（1.2）中列出的大部分变量，但我们可以想象引入其他参数，如梯度下降算法的局部或可变学习率。

进一步的方法可以在这些基本的FL融合和局部训练算法上进行扩展，以适应数据分布、客户选择和隐私要求的不同属性。[32]中的论文提出了一种基于动量的FL方法来加速收敛，其灵感来自集中式ML优化，如[27]。有状态的优化算法，如ADMM一般只适用于合作中的所有各方每次都参与，保留一方的状态[7]。不同的方法，包括[18]和[17]，使ADMM适应实际的FL设置。FedProx[31]引入了一个近似正则化项，以解决非IID用例中各方的数据异质性。其他方法，如[36]，超越了梯度下降法的优化。

对于每个处理数据异质性、模型结构和各方的具体方面的FL方法，我们需要定义一个算法，该算法由$\mathcal{L}$、$\mathcal{F}$以及各方和聚合器之间的交互协议组成，即q_k和r_k的格式。在本书的其余部分，我们发现有不同的最先进的方法来处理数据和模型的异质性方面。

1.0.4.2 经典的机器学习模型

经典的机器学习技术也可以应用于联合学习的场景。其中一些技术可

以与DNN非常相似地进行处理。其他技术则必须为分散的训练而完全重新考虑。

线性模型，包括回归和分类，可以通过类似于调整神经网络训练过程的方式调整训练过程，在联邦学习中进行训练。具有特征向量$x_i = (x_i^1, x_i^2, \cdots, x_i^m)$的训练数据可用于训练线性回归的预测器，例如，其形状为

$$y_i = w_1 x_i^1 + w_2 x_i^2 + \cdots + w_m x_i^m + b\Theta$$

$$wk, t+1 := wt` k\nabla l(wk, t+1, Xk, Yk)$$

在权重的梯度上应用针对各方的学习率ηk。所有各方将他们的模型权重发送到聚合器，在那里权重被平均化，由（wt，bt）定义的新模型M被重新分配给各方。我们也可以应用其他的融合方法，如Federated SGD或上一小节中讨论的任何高级方法。由于w较小，这往往比DNN的情况下收敛得更快。其他经典的线性模型，如逻辑回归或线性支持向量机（SVM）[20]，也可以用类似的方法转化为联合学习方法。

决策树和更高级的基于树的模型需要一个不同的方法来进行联合学习，而不是像我们讨论到这里的那些具有静态参数结构的模型类型。决策树是一种成熟的分类模型类型，通常用于分类问题[46]。在决策的可解释性对社会很重要的领域，如医疗、金融和其他监管要求展示决策所依据的标准的领域，它尤为重要。虽然DNN和线性模型可以在本地训练，并且本地参数可以在聚合器处合并，但是还没有提出好的融合算法来将独立训练的树模型合并成一棵决策树。

白皮书[35]描述了ID3算法[46]的联合方法，其中树的形成发生在聚合器上，各方的作用是根据他们的本地训练数据，对提议的类别分割做出计数响应。它适用于数字和分类数据。在其集中的原始版本中，ID3决策树计算每个特征的信息增益，将训练数据集分成给定的类别。它选择具有最大信息增益的特征，并计算该特征的值，使其对D进行最佳分割。一个属性通常不能充分地分割D。对于刚刚创建的树的每个分支，我们递归地应用同样的方法。我们通过计算每个子树数据集相对于其余特征的信息增益，询问哪一个下一个特征能够最好地分割每个子树中的数据子集。该算法继续递归地完善分类，直到当一个树节点的所有成员具有相同的类别标签或满足最大深度时停止。

在联合版本中，聚合器的融合函数F计算信息增益并选择下一个特征来增长树。为了获得计算信息增益的输入，聚合器用提议的特征和分割值查

询所有各方。各方计算每个提议的子树的成员及其标签，作为其本地训练函数F，并将这些计数作为回复返回给聚合器。聚合器将所有各方提出的每个特征的计数相加，然后继续计算这些汇总计数的信息增益。与集中式版本一样，选择下一个最佳特征，并再次分割子树，如此循环。

在这种方法中，聚合器发挥了突出的作用，并进行了大部分的计算，而各方主要提供与特征和分割值有关的计数。与其他联合学习方法一样，训练数据从未离开任何一方。根据训练数据集的数量和类成员的数量，这可能需要进一步的隐私保护措施，以确保在这种简单的方法中不会有太多的信息被披露。尽管如此，这是一个很好的例子，说明联合学习的方式与DNN和线性模型的方式不同。

决策树集合方法通常比单个决策树提供更好的模型性能。随机森林[8]，特别是梯度提升树，如流行的XGBoost[13]被成功地用于不同的应用，也被用于Kaggle比赛，提供了更好的预测精度。联合随机森林算法可以追求与决策树类似的方法，在聚合器中生长单个树，然后使用各方的数据收集。每次添加时都会随机选择一个特征子集，创建集合体的下一棵树，然后再次从各方查询。对于并非所有各方都拥有每个相关数据记录的相同特征集的情况，提出了更复杂的算法，例如，[34]和[20]。这种情况被称为垂直联合学习（更多内容见下一小节），需要用加密技术将每一方的记录与同一实体相匹配。

梯度增强树在预测效果不佳的决策空间区域加入合集，而在随机森林中则是随机加入。为了确定从合集的下一棵树开始，必须对Dk中的所有训练数据样本计算损失函数，这些样本位于各方。与其他基于树的算法一样，树的生长和对集合体的决策是在聚合器中进行的。然而，除此之外，各方还需要在给聚合器的回复中包括梯度和Hessians，以便对合集的下一棵树做出选择。聚合器的融合函数也需要一个量化的近似值，例如，潜在类别中训练数据样本的直方图。联合梯度增强树，就像其集中训练的对应树一样，通常有很好的准确性，并且可能比其他基于树的学习算法的过拟合更少。Ong等人[45]提出的方法使用党派适应性的量化草图来减少信息泄露。其他联合XGBoost的方法使用加密方法和安全的多方计算方法进行交互和损失计算[14, 33]。这需要在相当的模型性能下有更高的训练时间，适合于需要非常严格的隐私的企业场景。[63]中的概述提供了关于联合梯度提升中隐私权衡的有趣讨论，也可以应用于更简单的基于树的模型。

第二章更详细地介绍了训练树状模型的多种算法，包括梯度增强树。

通过这次简短的参观，我们对最流行经典机器学习和神经网络方法进行了概述。我们看到普通机器学习算法的联邦版本可以通过仔细考虑哪些计算在聚合器进行，哪些在各方进行，以及各方和聚合器之间需要什么样的互动来创建。

1.3.3 横向、纵向的联合学习和分离学习

到目前为止，在讨论数据在各方之间的分布时，我们一般假设所有各方的训练数据包括每个样本的相同特征，各方拥有与不同样本有关的数据。例如，医院A有一些病人的健康记录和图像；第二家医院B有其他病人的记录，如图1.3所示。

在神经网络的情况下，我们假设每一方都有同等大小和内容的样本。

然而，在某些情况下，各方可能有不同的特征，指的是同一个实体。再以卫生保健为例，初级保健医生可能有与病人长期就诊有关的电子健康记录，而放射科医生则有与病人疾病有关的图像。一个骨科医生可能有病人的手术记录。在寻找骨科手术健康结果的预测因素时，根据所有三方（初级保健、放射科医生和骨科医生）的数据进行预测可能是有益的。在这种情况下，只有一方，即骨科医生，可能有实际的标签：手术的结果。我们称这个数据集为垂直分割的。

图1.4说明了垂直分区，特征在身份密钥中重叠，以匹配双方的记录，例如，政府标识符。由于并非所有的相关特征都存在于任何一方，所以学习不能在每一方独立进行。此外，身份密钥必须被匹配，以了解每一方的特征如何相互补充。为了保护每一方的数据隐私，我们需要一种加密的方法来匹配数据和执行学习过程。Hardy等人提出了一种基于部分同态加密的开创性的早期方法[24]，其他人如Xu等人[62]提出了更有效的变体，减少了通信和计算要求，以至于它在实际企业实践中变得可行。垂直FL将在第18章中详细介绍。在本章后面，我们将更深入地讨论联合学习的安全和隐私问题。

Vepakomma等人[55]以及其他一些人[49]提出了与垂直联合学习有点关系的分裂学习。在分割学习中，DNN在客户和服务器之间进行分割，客户保持DNN的 ”上层 ”部分，直到分割层，服务器拥有分割层和下面的部分。在其基本形式中，客户端拥有输入数据，服务器拥有标签。当使用SGD作为训练算法时，前向传递从客户端的输入开始，在分割层传播到服务器。反向传播是通过分割层从服务器到客户端进行的。通过这种方法，一方的数据也可以保持隐私，而另一方则拥有模型结构的一部分。分割式学习可

以变化为客户端也有标签，最后一个完全连接的层通过第二个分割层在客户端，或者多个客户端有垂直分割的数据，并使用分割层的分区与服务器进行通信。后一种情况可以被看作是垂直联合学习的概括。第19章更深入地讨论了分割学习。

1.3.4 模型个性化

模型的个性化是指根据参与FL过程的特定各方的数据分布，对（联邦训练的）全球模型进行调整。虽然参与FL过程使所有各方都能从大量的训练数据中受益，但有时对最终模型进行个性化处理以确保其反映特定各方所拥有的数据是有益的。如果各方对应于个人用户或组织，这一点尤其重要。在一个天真的情况下，个别当事方可以在本地数据上运行额外的本地训练纪元，以结束FL过程。Wang等人提出了一种方法来评估每一方的个性化的好处[56]。

Mansour等人[37]分析了三种不同的个性化方法：用户聚类，在插值数据（全局和局部之间）上进行训练，以及模型插值。第一种方法需要放宽隐私要求或先进的隐私技术，以基于训练数据对用户进行聚类。数据插值是基于创建一个全局数据集。虽然所有方法都有效，但从隐私的角度来看，模型插值的适用性最广。Grimberg等人提出了一种方法，通过确定优化的权重来优化全局模型和局部模型，以达到个性化的目的，并对之前讨论的方法进行了扩展[22]。

虽然个性化的方法仍在不断发展，但这是对FL过程的一个重要补充。第4章和第5章深入讨论了模型的个性化。

安全和隐私

通过将数据留在原地，FL在一开始就提供了一个固有的隐私水平。然而，仍然存在着侵犯数据隐私的可能性。重要的是要了解在FL应用过程中可能出现的不同威胁模式，以确保用正确的防御措施适当地减轻相关风险。在这一节中，我们将概述FL的脆弱性，并勾勒出相应的缓解技术。

图1.5介绍了FL的潜在威胁以及潜在对手的特征。

本章提供了一个介绍联合学习。我们讨论的主要动机训练数据,而不是将所有数据放在一起,因为它是在集中的毫升。需要遵守隐私规定,保密的数据,和务实考虑如网络质量是主要的驱动程序。我们介绍了政党的主要概念和聚合器,然后通过我们需要考虑的主要观点FL:机器学习的角度来看,安全和隐私的角度来看,然后是系统的视角。所有这些观点手拉手去设计一个FL系统适合它的任务。

让我们首先了解对手可能利用的潜在攻击面来分析风险。一个设置良好的FL系统利用安全和认证的渠道来确保各方和聚合者之间交换的所有信息不会被其他实体截获，同时防止冒充。因此，我们可以假设，聚合器和各方是唯一能够访问他们之间交换的信息和训练过程中产生的工件的两个实体。考虑到这一点，我们可以将潜在的对手分为内部人和外部人。内部对手是参与训练过程的实体，他们可以接触到训练期间产生的工件和针对他们的信息。所有其他的潜在对手被认为是局外人。在这种分类中，收到FL训练过程中产生的最终模型的实体被认为是外部人员。

我们可以把对FL的威胁分为操纵和推理威胁，其中操纵威胁是指内部人员的目标是通过操纵她在训练过程中可以接触到的任何工件来影响模型，使之对他们有利；而推理威胁是指内部人员或外部人员试图提取有关训练数据的私人信息。在下文中，我们将更详细地解释这些攻击中的一些。

1.4.1 操纵攻击

有多种类型的操纵攻击，内部对手的主要目标是操纵FL训练期间产生的模型，使其对自己有利。在某些情况下，对手可能想造成有针对性的错误分类，而在其他情况下，她可能想降低模型性能，使其无法使用。后门攻击[2, 23]和拜占庭攻击[29]分别是有目标和无目标攻击的两个例子。后门攻击会产生有针对性的错误分类，而拜占庭攻击会导致模型性能变差。拜占庭攻击可能由单方或多方串通进行，可能简单如注入随机噪声[57]，也可能复杂如运行优化以规避现有防御[4, 60]。标签翻转攻击，即一个或多个恶意方翻转一些标签，是另一种降低模型性能的流行方式[19]。

在FL文献中，进行操纵攻击的内部人员通常被认为是恶意的一方[57, 59]。然而，一个恶意的聚合者也可以进行这种类型的攻击。这就要求聚集者用中毒的样本对聚集的模型进行几个历时的训练，然后将新的操纵模型发送给各方。还有一种攻击是，多个串通的各方同意操纵模型的更新，因此最终的模型会导致有针对性的分类[65]或不良行为。

不幸的是，FL中的操纵攻击并不容易被发现。首先，并不是所有的数据都可以让潜在的防御者运行在集中式环境中经常应用的防御措施。1其次，数据的异质性已被证明会影响FL的稳健性[65]，使得它很难区分不应该被包括在内的恶意模型更新和包括在内将有利于最终模型的良性模型更新。第三，一个成功的攻击不需要长时间的操作；通过正确把握攻击时机，有可能获得高的攻击成功率[65]。最后，随着防御措施的发展，攻击也在不断发展；自适应攻击被设计用来规避一些提议的防御措施[4, 60]，在攻击者和

防御者之间形成了熟悉的竞争。

大多数防御方法假定聚合器是防御者，并可能过滤掉恶意的模型更新。聚合者收到的模型更新被检查，以确定差异过大的更新。这一类的防御方法使用多种距离度量，有些假设一定数量的当事人总是恶意的[5, 12, 64]。然而，大幅不同的模型更新不一定是攻击；它可能是由一方有机地产生的，其数据相对于其他各方表现出大量的非IID性。要知道不寻常的更新是良性的还是恶意的，这一困难显然因聚合者不能访问训练数据的事实而加剧。为了克服这个困难，已经开发了一些方法，假设聚合者可以获得一个与各方持有的数据集相似的分布[58]。但是，对于某些用例来说，这可能很难获得。另一种方法[54]在训练神经网络时并不完全摒弃异常更新，而是适应神经网络的一些层以防止过度拟合。在[3]中提出了一个本质上不同的方法，其中问责制被用来阻止攻击。这种方法存储了整个训练过程的不可抵赖的记录。通过确保所有各方对他们的模型更新以及训练过程负责来提供透明度，而聚合器也对其融合的方式负责。

第16章将介绍操纵攻击和防御的概况，第17章主要介绍训练神经网络时对拜占庭攻击和防御的理解。

1.4.2 推理攻击

不共享数据的训练是应用FL的驱动力之一，也是最重要的优势。回顾一下，模型更新是与聚合器共享的唯一数据，而私人训练数据永远不会被泄露。这种设计确保私人信息的简单暴露在FL系统中不是一个问题。因此，隐私泄露只能通过推理发生。

推理攻击利用FL过程中或之后产生的人工制品，试图推断出私人信息。推理威胁对机器学习来说并不新鲜。事实上，大量的工作已经记录了敌方的方式，敌方只需访问一个ML模型，就可以推断出其训练数据的私人信息。这种黑箱设置中的攻击包括。

- 成员推理攻击，即对手可以确定某个特定的样本是否被用来训练模型。例如，当模型使用来自某些社会群体的数据时，这是一种隐私侵犯，例如政治或性取向或疾病。 - 模型反转攻击，对手想找到每个类别的代表。在人脸识别系统中，例如，这可能会暴露出一个人的脸。 - 提取攻击，对手的目标是获得训练过程中使用的所有样本。 - 属性推理攻击，其中独立于训练任务的属性可能被揭示。

在FL设置中，外人可以获得最终的ML模型，而内人可以获得中间模型；因此，内人和外人都可以进行上述的攻击。

此外，有趣的是，交换的模型更新乍一看似乎是无害的，也可以被内部人员用来推断私人信息。使用模型更新的攻击包括[21, 25, 39, 41, 67, 68]，在某些情况下，表现出比使用模型进行的攻击更快更高的成功率。基于模型更新的攻击可以由好奇的各方或恶意的聚合器。这些攻击可以是被动的，即对手唯一地检查所产生的工件，也可以是主动的，即它采取行动加快推理的速度。

鉴于这些隐私暴露的风险，已经提出了几种保护FL过程的技术。它们包括使用差分隐私(DP)[16]，安全的多方计算技术[6, 44, 61, 66]，两者的结合[52]，以及使用可信的执行环境[11, 30]，等等。

DP是一个严格的数学框架，当且仅当训练数据集中包含的单个实例仅对算法的输出造成统计学上不明显的变化时，算法可被描述为差异性私有。该方法通过DP机制增加噪音，该机制是为数据集和将用数据回答的查询而定制的。DP提供了一个可证明的数学保证；但是，它可能会大大降低模型的准确性。另一种流行的防止推理攻击的技术是使用安全的多方计算，好奇的聚合者不能得到从各方收到的单个模型更新，但仍然可以获得最终的融合结果（以明文或密码文本）。这些技术包括屏蔽[6]、Paillier[44, 66]、Threshold Paillier[52]和Functional encryption[61]，仅举几例。所有这些技术都有略微不同的威胁模型，因此，适用于不同的场景。

现有的防御措施提供不同的保护，并针对不同的推理攻击。重要的是，要确保根据手头的使用情况选择防御措施，以确保实现正确的保护水平。在完全信任的情况下，可能不需要额外的保护，例如，当一家公司用来自多个数据中心的数据训练一个模型时。然而，在一个竞争者的联盟中，推断的风险可能太高，导致使用一个或多个保护机制。

本书的多个章节都涉及推理攻击的威胁和防御措施。章分析了FL系统的推理风险。它介绍了现有的攻击和防御措施，证明每一种防御措施所提供的保护水平适用于稍微不同的情况。该章还提出了一个分析，以帮助确定如何将不同的场景和信任假设匹配到合适的防御措施中。第14章对基于信任执行环境的防御措施进行了更深入的审查，第15章详细介绍了基于梯度的数据提取攻击的机制。

1.5 联邦学习系统

一个FL进程最终是在一个分布式系统上执行的，各方和聚合器在该系统上运行。这个系统的各个部分必须满足各方、聚合器的计算、内存和网络要求，以及它们之间的通信。由于本地模型的训练是在数据所在的地方

进行的，我们必须密切关注各方训练时的可用资源。聚合器大多在数据中心环境下运行，至少在常用的单一聚合器架构。不过，在处理大量的当事人时，他们还是需要有合适的资源和能力来扩展。最后，网络连接和带宽要求可能会根据模型大小、模型更新内容、频率和加密协议的使用而有所不同，这一点在上一节已经讨论过了。因此，联合学习的系统要求与集中学习的方法有很大不同。

当事人客户。与集中式ML系统最明显的区别在于，一方可能不在我们通常选择的ML平台的系统上。虽然当各方是位于不同司法管辖区的数据中心时，这可能没有那么大的问题，但在嵌入式系统、边缘计算和移动电话中问题更大。三种不同类型的功能可能会占用资源。 - 如果模型很大，本地机器学习过程可能需要大量的计算和内存。特别是对于大型的DNN，例如大型的语言模型，就是这种情况。它可能需要GPU的支持，而在嵌入式系统中，甚至在远程数据中心或软件即服务相关的数据存储中，可能都没有GPU。然而，经典技术甚至在小型设备（如Rasberry Pis®）上也是可行的，还有一些小尺寸的软件包，如Tensorflow Lite®，需要较少的随行存储和内存。 - 联合学习方的客户端驱动本地机器学习模型，并与聚合器进行通信。不过在大多数情况下，它的占地面积很小，即使在小型边缘设备中也能容纳。 - 然而，如果使用加密协议，例如，基于阈值Paillier密码系统的安全多方计算（SMC）实现，它可能会使一方客户端的计算成本增加几个数量级。大多数加密和解密技术可以并行，因此可以由GPU或专用硬件支持。

聚合器服务器。聚合器通常位于数据中心环境中，可以获得充足的资源。然而，扩展到大量的当事方会带来一些挑战。

为了与大量的各方进行通信，聚合器必须能够维持大量的连接。池化连接是一种成熟的方法，适用于各种系统，可以在这里以类似方式使用。

在聚合器上执行融合算法往往会产生适度的编译成本。简单的融合算法，如1.3节中讨论的FedAvg，执行相当简单的平均化操作。其他融合算法可能更复杂，但通常比一方的本地训练有更低的计算要求。然而，在大型DNN和大量当事方的情况下，从当事方收到的权重向量集的大小可能非常大。一方的权重向量可以达到几十兆字节。处理成千上万的当事人，在一个计算节点的内存中进行平均计算可能太多。

已经提出了不同的方法来解决聚合器计算上的扩展。可以使权重持久化，融合算法可以用并行计算的方式进行，例如使用Hadoop或Spark。其

他方法使用加法的换元特性，将各方分成组。这些组被分配给一个聚合器，每个聚合器计算这个组的平均数。然后，一个主要的聚合器将本地聚合的结果聚合起来，按每个聚合器的各方数量加权。So等人[47]提出了一个这样的方法，还有不同的变体，包括多级聚合。对于非常大的当事人集合，在每一轮中经常使用当事人的子抽样，可以补充其他方法。

基于树的FL算法通常对聚合者提出更多的计算要求，而对各方提出的要求较少。

沟通。在FL设计中必须考虑聚合者和各方之间的通信数量和质量。在数据中心和云环境中，我们通常可以假设带宽是足够的，连接是可靠的。FL过程可能需要相当长的时间。因此，通信协议需要对偶尔的断线有良好的适应性。在企业背景下，一个重要的实际考虑是连接方向。企业的IT部门有严格控制的流程来打开网络端口。选择一个不需要各方打开端口的网络协议，而是让他们初始化与聚合器的连接，将加速FL系统的实施。

嵌入式系统、边缘设备和移动系统构成了一个更大的挑战。一些方系统可能是间歇性的连接，例如在车辆中，或者带宽很差，是低成本的设备。这对FL进程来说可能是个问题。如果各方没有及时回应下一轮，我们需要一个策略来管理这些辍学的情况。我们需要建立一个法定人数，这个法定人数可能是针对特定用例的。当各方重新加入时，我们也需要一种方法。虽然quora是简单的辍学管理手段，但其他方法，如TIFL，提出了积极的落伍者管理，按响应时间对各方进行分组，并减少查询频率[10]。响应时间的系统差异甚至会导致模型的偏差[1]。

间歇性或低带宽的通信也可以通过算法来解决，例如，通过减少回合数、压缩模型和融合更多分歧的模型。第6章和第7章对此有更详细的讨论。

使用安全计算方法，如SMC，可能会增加消息的大小和数量，并可能对连接不畅的设备构成问题。此外，一些用于垂直联合学习的SMC协议可能需要各方之间的点对点通信，这在两个方面存在问题。它要求各方将端口暴露给他们的同行，这在企业中是一个实施障碍。如果通过将所有流量通过聚合器或另一个中介机构进行路由来缓解，这又会使网络流量翻倍。因此，虽然SMC通常是一个非常好的保护隐私的方法，但它有很大的资源要求。

设计选择和权衡。在实现FL系统时，我们经常需要用合适的算法方法来交换可用资源。如果我们能够对一方可用的硬件进行选择，我们就可以选择一个适合的我们选择的ML方法。我们可以在车辆或制造机器人上

添加一个带有强大GPU的嵌入式系统，或者在我们想让其参与联盟的数据中心上添加GPU。这并不总是可能的。在各方的计算平台是给定的情况下，我们可以使用适合我们资源的ML方法。虽然DNN在一方是资源密集型的，但基于树的模型，如联合的XGBoost，则要求不高。而且，算法可以适应系统的限制。

1.6 摘要和结论

本章对联合学习进行了介绍。我们讨论了将训练带到数据上的主要动机，而不是像集中式ML那样将所有的数据集中起来。遵守隐私法规的需要，数据的保密性，以及网络质量等务实的考虑，是主要的驱动力。我们介绍了各方和聚合者的主要概念，然后通过我们需要考虑的FL的主要观点：机器学习的观点，安全和隐私的观点，然后是系统的观点。所有这些角度都是携手并进的，以设计一个适合其任务的FL系统。

我们从一个特殊的角度来看待实施FL的企业的需求。这包括需要同时支持神经网络和经典方法，各方数据和系统的异质性，以及当不同类别的数据保存在不同系统中时，需要垂直FL。这与FL在移动设备中的应用有些不同，移动设备大多比较同质化，但带来的规模问题也不同。

本书的其余部分将更深入地讨论所有这些方面： - 第一部分从机器学习的角度看FL，讨论了基于树的模型、效率、个性化和公平性。 - 第二部分更深入地讨论了系统的观点。 - 第三部分包括五章，涉及隐私和安全。详细描述了推理和操纵攻击，并提供了更多关于防御措施的信息，何时应用这些措施。 - 第四部分包含的章节详细介绍了垂直FL以及分裂学习。 - 第五部分展示了FL的应用工作和重要应用领域的要求，如医疗和金融。

本书的这一范围为寻求深入背景的研究人员和从业人员提供了企业中FL的最先进的概述。

第一部分

联邦学习视为一个机器学习问题

第二章　通信高效的分布式优化算法

Gauri Joshi and Shiqiang Wang

摘要

在联邦学习中，连接边缘参与者和中央聚合器的通信链路有时是有带宽限制的，而且会有很高的网络延迟。因此，设计和部署具有通信高效的分布式训练算法是急需的。在本章中，我们将回顾两种不同的具有通信高效的分布式随机梯度下降（SGD）方法：（1）本地更新随机梯度下降（SGD），客户端进行多次本地模型更新，并周期性的进行聚合；（2）梯度压缩和稀疏化方法，以减少每次更新传输的比特数。在这两种方法中，误差收敛与迭代次数和通信效率之间存在着权衡关系。

2.1　引言

ML训练中的随机梯度下降。大多数监督学习问题都是使用经验风险最小化框架来解决的[1, 2]，其目标是最小化经验风险目标函数$F(\boldsymbol{x}) = \sum_{j=1}^{n} f(\boldsymbol{x}, \xi)/n$。其中，$n$是训练数据集的大小，$\xi_j$是第$j$个已标注的训练样本，$f(\boldsymbol{x}, \xi_n)$是（通常是非凸的）损失函数。一种普遍优化$F(\boldsymbol{x})$的算法是随机梯度下降(SGD)。在这种算法中，我们计算$f(\boldsymbol{x}, \xi_n)$在小的、随机选择的子集$\mathcal{B}$（称为迷你批次）上的梯度，每个子集有b个样本[3, 4, 5, 6, 7, 8]，并根据$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \eta \sum_{i \in \mathcal{B}} \nabla f(\boldsymbol{x}_k; \xi_i)/b$ 更新$\boldsymbol{x}$，其中η被称为学习率或步长。虽然算法是为凸目标设计的，但由于小型批量SGD有能力摆脱鞍点和局部最小值[9,

10, 11, 12]，因此即使在非凸损失表面也有良好的表现。因此，它是最先进的机器学习中的主导训练算法。

对于像Imagenet[13]这样的大规模数据集，在单个节点上运行小批量的SGD可能会非常慢。进行梯度计算并行化的一个标准方法是参数服务器（PS）框架[14]，由一个中央服务器和多个工作节点组成。拖拽的工人节点和通信延迟会成为将这个框架扩展到大量工人节点的瓶颈。一些方法，如异步[15, 16, 17, 18]和周期性梯度聚合[19, 20, 21]已被提出，以提高基于数据中心的ML训练的可扩展性。

联邦学习的动机。尽管算法和系统的进步提高了效率和可扩展性，但基于数据中心的训练仍有一个主要局限。它要求将训练数据集集中在参数服务器上，由它在工作节点上进行打乱和拆分。手机、物联网传感器和具有设备上计算能力的相机等边缘方的迅速扩散，导致了这种数据分区模式的重大转变。边缘方从它们的环境中收集丰富的信息，这些信息可用于数据驱动的决策。由于有限的通信能力以及隐私问题，这些数据不能直接发送到云端进行集中处理或与其他节点共享。联邦学习框架建议将数据留在边缘方，而将模型训练放在边缘。在联邦学习中，数据被保存在边缘方，模型以分布式的方式被训练。只有梯度或模型更新在边缘方和聚合器之间进行交换。

系统模型和符号。如图2.1所示，一个典型的联邦学习设置包括一个连接到K个边缘方的中央聚合器，其中K可能是数千甚至数百万的量级。每一方i都有一个由n_i个样本组成的本地数据集$\mathcal{D}_i$，它不能被传输到中央聚合器或与其他边缘方共享。我们用$p_i = n_i/n$来表示第i方所占数据比例，其中$n = \sum_{i=1}^{K} n_i$。聚合器试图用本地数据集的并集$\mathcal{D} = \cup_{i=1}^{K} \mathcal{D}_i$来训练一个机器学习模型$\boldsymbol{x} \in \mathbb{R}^d$。模型向量$x$包含模型的参数，例如，神经网络的权重和偏差。为了训练模型x，聚合器试图最小化以下经验风险目标函数：

$$F(\boldsymbol{x}) := \sum_{i=1}^{K} p_i F_i(\boldsymbol{x}) \tag{2.1}$$

其中$F_i(\boldsymbol{x}) = \frac{1}{n_i} \sum_{\xi \in \mathcal{D}_i} f(\boldsymbol{x}; \xi)$是第$i$方的本地目标函数。$f$是由模型$\boldsymbol{x}$定义的损失函数（可能是非凸的），$\xi$代表本地数据集$\mathcal{D}_i$的一个数据样本。请注意，我们分配的权重与第$i$方的数据比例成正比。这是因为我们想模拟一个集中式的训练场景，将所有的训练数据传输到一个中央参数服务器。因此，拥有更多数据的一方将在全局目标函数中获得更高的权重。

由于边缘方的资源限制和大量参与方，联邦训练算法必须在严格的通

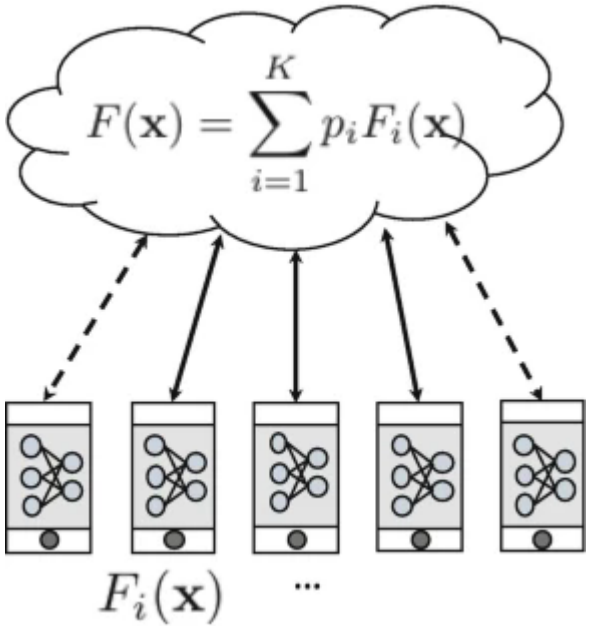

图 2.1: 在联邦优化中，聚合器的目标是最小化边缘方局部目标函数$F_i(\boldsymbol{x})$的加权平均值

信限制下运行，并应对数据和计算的异质性。例如，连接每个边缘方和中央聚合器的无线通信链路可能有带宽限制，并且有很高的网络延时。另外，由于网络连接有限和电池的限制，边缘方可能只是间歇性地可用。因此，在给定的时间内，只有K个边缘方中只有m个子集可以参与训练模型x。为了在这些通信约束条件下运行，联邦学习框架需要新的分布式训练算法，超越在数据中心环境中使用的算法。在第6.2节中，我们回顾了local-update SGD算法及其变体，这些算法减少了边缘各方与聚合器的通信频率。在第6.3节中，我们回顾了压缩和量化的分布式训练算法，这些算法减少了边缘各方发送给聚合器的每次更新的比特数量。

2.2 Local-Update SGD和FedAvg

在本节中，我们首先讨论local-update SGD及其变体。FedAvg算法是联邦学习的核心，它是local-update SGD的扩展。我们将讨论FedAvg如何建立在local-update SGD之上，以及在联邦学习中用来处理数据和计算异质

性的各种策略。

2.2.1 Local-Update SGD及其变体

同步分布式SGD。在数据中心的设置中，训练数据集$\mathcal{D}$被打乱并平均分配到m个工作者节点中。训练机器学习模型的标准方法是使用同步分布式SGD，其中梯度由工作者节点计算，然后由中央参数服务器汇总。在同步SGD的第t次迭代中，工作者从参数服务器中拉取模型的最新版本x_t。每个工作者i使用从本地数据集$\mathcal{D}_i$中抽取的B个样本的小批量数据$\mathcal{B}$来计算一个小批量随机梯度$g_i(x) = \sum_{\xi \in \mathcal{B}} f(x;\xi)$。然后，参数服务器收集来自所有工作者的梯度，并按以下方式更新模型参数

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \frac{\eta}{m} \sum_{i=1}^{m} g_i(\boldsymbol{x}) \tag{2.2}$$

随着工作者数量m的增加，同步SGD的误差与迭代收敛性得到改善。然而，由于工作者的局部梯度计算时间存在差异，等待所有工作者完成梯度计算的时间也会增加。为了提高工作者数量的可扩展性，中提出了同步SGD的¡mark¿拖拽者弹性变体¡/mark¿，进行异步梯度聚合。

Local-Update SGD。尽管异步聚合方法对提高分布式SGD的可扩展性很有效，但在许多分布式系统中，工作者和参数服务器之间交换梯度和模型更新的通信时间会主导本地梯度计算时间的变化。因此，每次迭代后的节点间的持续通信可能是过于昂贵和缓慢的。Local-Update SGD是一种通信效率高的分布式SGD算法，它通过让工作者节点执行多次本地SGD更新而不是仅仅计算一个小批量梯度来克服这个问题。

如图6.2所示，Local-Update SGD将训练分为几轮通信。在一个通信轮中，每个工作者使用SGD对其目标函数$F_i(x)$进行局部优化。每个工作者i从当前的全局模型开始，用x_t表示，并执行τ次SGD迭代，以获得模型$x_{t+\tau}^{(i)}$。然后，所得到的模型由m个工作者发送到参数服务器，服务器对其进行平均，以更新全局模型，如下所示:

$$x_{t+\tau} = \frac{1}{m} \sum_{i=1}^{m} x_{t+\tau}^{(i)} \Theta$$

Local-Update SGD每一次迭代的运行时间。通过在与参数服务器通信之前在每个工作器上执行 τ 个本地更新，本地更新SGD减少了每次迭代的预期运行时间。让我们通过考虑以下延迟模型来量化这种运行时间的

节省。第i个工作者在第k个局部步骤计算小批梯度的时间被建模为随机变量Yi,k FY，假定在工作者和小批之间是独立和相同的分布（i.i.d）。通信延迟用一个常数D表示，它包括将本地模型发送到参数服务器和从参数服务器接收平均的全局模型所需的时间。由于每个工人i进行了 τ 次本地更新，其平均本地计算时间（完成图6.3中3个蓝色箭头的一个序列所需的时间）由以下公式给出

$$\bar{Y} = \frac{Y_{i,1} + Y_{i,2} + \cdots + Y_{i,\tau}}{\tau}$$

如果 τ =1，在这种情况下，本地更新SGD会简化为同步SGD，那么随机变量Y与Y是相同的。

$$\mathbb{E}[T_{Local-update}] = \mathbb{E}[max(\bar{Y}_1, \bar{Y}_1, \cdots, \bar{Y}_m] + \frac{D}{\tau} = \mathbb{E}[\bar{Y}_{m:m}] + \frac{D}{\tau}$$

术语$Y_{m:m}$表示具有概率分布$Y \sim F_Y$的m个i.i.d.随机变量的最大顺序统计。从（6.6）中我们可以看出，执行更多的局部更新可以通过两种方式减少每次迭代的运行时间。首先，通信延迟在 τ 次迭代中得到摊销，并减少了一个系数 τ 。其次，执行局部更新也提供了一个减少散兵游勇的好处，因为$\bar{Y}_{m:m}$的尾部比$Y_{m:m}$轻，因此（6.6）中的第一个项随着τ而减少。

本地更新SGD的错误收敛。正如我们在上面看到的，将工作者和参数服务器之间的通信频率降低到在 τ 迭代中只有一次，可以使每次迭代的运行时间大大减少。然而，设定一个大的τ值，局部更新的数量会导致较差的误差收敛。这是因为，随着τ的增加，工作节点的模型$x_{t+\tau}^{(i)}$会相互偏离。论文给出了局部更新SGD在局部更新数量τ方面的错误收敛分析。假设目标函数$F(x)$是L-Lipschitz光滑的，学习率η满足$\eta L + \eta^2 L^2 \tau(\tau - 1)d1$。随机梯度$g(x;\xi)$是$\nabla F(x)$的无偏估计，即$\mathbb{E}_\xi[g(x;\xi)] = \nabla F(x)$。随机梯度$g(x;\xi)$被假定为有边界的方差，即$Var(g(x;\xi))d\sigma^2$。如果起点是$x_1$，那么经过局部更新SGD的$T$次迭代后，$F(x_T)$被约束为

$$\mathbb{E}[\frac{1}{T}\sum_{t=1}^{T} \|\nabla F(x_t)\|^2] \leq \frac{2[F(x_1 - F_{\inf})]}{\eta t} + \frac{\eta L \sigma^2}{m} + \eta^2 L^2 \sigma^2 (\tau - 1)$$

其中x_t表示第t次迭代时的平均模型。设置$\tau = 1$使得本地更新SGD及其误差收敛边界与同步分布式SGD相同。随着τ的增加，约束的最后一项会增加，从而增加收敛时的误差底线。

适应性沟通策略。从上面的运行时间和误差分析中，我们可以看到，当我们改变 τ 时，每个迭代的误差和通信延迟之间存在着权衡。较大

的τ可以减少预期的通信延迟，但是产生更差的误差收敛。为了获得快速收敛和低误差底线，提出了一个在训练过程中适应τ的策略。对于一个固定的学习率n，中的以下策略会逐渐减少τ:

$$\tau_\ell = \left\lceil \sqrt{\frac{F(x_{t=\ell T_0})}{F(x_{t=0})}} \tau_0 \right\rceil$$

其中，τ_ℓ是训练中T_0秒的第ℓ个区间内的局部更新次数。这个更新规则也可以被修改，以考虑到基本的可变学习率时间表（图6.4）。

弹性平均法和重叠SGD。在本地更新的SGD中，在下一组 τ 更新开始之前，需要将更新的全局模型传达给各节点。此外，在m个节点中最慢的节点完成其 τ 个本地更新之前，全局模型不能被更新。这种通信障碍会成为全局模型更新的瓶颈，并增加每轮训练的预期运行时间。由于这种通信障碍是由算法而不是系统实现强加的，我们需要一种算法方法来消除它，并允许通信与本地计算重叠。诸如等作品使用异步梯度聚合来消除同步障碍。然而，异步聚合会导致模型僵化，也就是说，慢速节点会有任意过时的全局模型版本。最近的一些工作提出了本地更新SGD的变种，允许通信和计算的重叠。在这些算法中，工作节点从一个锚模型开始他们的本地更新，该模型甚至在最慢的节点完成上一轮本地更新之前就可以使用。这种方法受到提出的弹性平均SGD（EASGD）算法的启发，该算法在目标函数中增加了一个近似项。近似方法，如，虽然不是为此目的而设计的，但自然允许通信和计算的重叠。

6.2.2 联合平均法（FedAvg）算法及其变体

FedAvg算法。由于在联合学习中，边缘伙伴的通信能力有限，本地更新的SGD特别适合于联合学习的132G。Joshi和S. Wang的学习框架，在这里它被称为FedAvg算法。其主要区别如下。首先，作为云中服务器的工作节点被移动和物联网设备等边缘方所取代。由于边缘方的间歇性可用性，与数据中心的设置不同，每轮训练中只有K方中的m个子集参与。其次，数据集Dican的大小和组成在边缘方之间都是高度异质的，不像数据中心的设置，数据集D被洗牌并均匀地划分到工人节点。

联合平均算法（FedAvg）也将训练分为通信轮。在一个通信轮中，聚合器从可用的各方中均匀地随机选择m个边缘方。每个边缘方使用类似于局部更新SGD的SGD对其目标函数Fi(x)进行局部优化。与基本的本地更新SGD不同的是，每个工作者执行相同数量的本地更新 τ ，在FedAvg中，本地更新的数量 τ 可能在不同的边缘方和通信回合中有所不同。一个常

见的实施做法是，各方运行的本地纪元E是相同的。因此， τ i= Eni其中B是小批量的大小。另外，如果每个通信轮次在壁钟时间上有一个固定的长度，那么 τ irep代表i方在时间窗口内完成的局部迭代，并且可能在不同的客户（取决于他们的计算速度和可用性）和不同的通信轮次中变化。在第r轮通信中，边缘各方从全局模型xr,0开始，各自进行 τ i个局部更新。假设他们得到的模型用x(i) r, τ i表示。共享的全局模型xris的更新方式如下。其中pi=—Di——D—，第i个边缘方的数据部分。

处理数据异质性的策略。由于数据集在各节点间高度异质化，边缘方的本地训练模型可能彼此有很大的不同。而且随着本地更新数量的增加，模型可能会变得对本地数据集过度拟合。因此，FedAvg算法可能会收敛到一个不正确的点，而这个点不是全局目标函数F（x）的静止点。例如，假设每个边缘方执行了大量的局部更新，第i方的局部模型收敛到x(i) = min Fi(x)。那么这些局部模型的加权平均将收敛于x= Ki=1pix(i) ，这可能与真正的全局最小值x =min F(x)有任意的不同。为了减少这种由数据异质性引起的求解偏差，一个解决方案是选择一个小的或衰减的学习率 η 和或保持小的局部更新数量 τ 。其他用于克服解决方案偏差的技术包括近似的局部更新方法，如，该方法为全局目标添加了一个正则化项，以及旨在最小化跨方模型的方法。通过交换控制变量来实现漂移。在高层次上，这些技术阻止了边缘方的模型偏离全局模型的情况。

处理计算异质性的策略数据异质性的影响会因边缘各方的计算异质性而加剧。即使边缘各方进行不同数量的局部更新 τ i，标准的FedAvg算法建议将所产生的模型按照数据比例pi进行简单的聚合。然而，这可能会导致一个不一致的解决方案，与预期的全局目标不匹配，如所示，并在图6.5中说明。最终的解决方案变得偏向于局部最优x(i) = min Fi(x)，而且它可能离全局最小x(i) = min F(x)有任意的距离。论文通过将累积的局部更新（x(i)r, τ i- x(i)r,0）按局部更新的数量 τ i进行归一化，然后再将其发送到中央聚合器，从而解决了这种不一致的情况。这种被称为FedNova的规范化联合平均算法的结果是一致的解决方案，同时保留了快速收敛率。

除了局部更新数量 τ i的变化，由于边缘方使用局部动量、自适应局部优化器（如AdaGrad）或不同的学习率计划，也可能出现计算异质性和解决方案不一致的情况。在这些情况下，需要一个通用的FedNova版本来解决不一致的问题。

处理边缘方间歇性可用性的策略 在一个联合学习设置中，边缘方

的总数可以达到数千甚至数百万台设备的数量。由于本地计算资源的限制和带宽的限制，边缘方只能间歇性地参与训练。例如，目前手机只有在插电充电时才会被用于联合训练，以节省电池。因此，在每一轮通信中，只有一小部分边缘方参与到FedAvg算法中。大多数关于设计和分析联合学习算法的工作都假设边缘方的子集是从整个边缘方集合中均匀地随机选择的。这种部分和间歇性的参与通过给误差增加一个方差项而放大了数据异质性的不利影响。最近一些134G. Joshi和S. Wang的作品提出了应对这种异质性并提高收敛速度的客户端选择方法。这些策略将更高的选择概率分配给具有较高局部损失的边缘方，并表明它可以加速全局模型的进展。然而，这种加速是以较高的非消失偏差为代价的，这种偏差随着数据异质性程度的增加而增加。论文提出了一种自适应策略，逐渐减少选择倾斜，以实现收敛速度和误差底限之间的最佳权衡。

2.3 模型压缩

除了执行多次本地更新外，模型在通信和计算过程中也可以被压缩。一种方法是使用标准的无损压缩技术，然而这只能在有限的程度上减少模型的大小，并且需要在接收方进行解压。在本节中，我们将讨论一类特殊的有损压缩技术，该技术通常用于提高联邦学习和分布式SGD中的通信效率。这些技术不需要在接收方进行解压，并且可以保证训练收敛。我们在第6.3.1和6.3.2节中重点介绍了提高通信效率的方法，在6.3.3节中重点介绍了提高通信和计算效率的方法。

2.3.1 有压缩更新的SGD

一个广泛使用的方法是压缩各参与方和聚合器之间传输的模型更新[22, 23]。特别是，我们定义了一个压缩器$\mathcal{C}(\boldsymbol{z})$，它产生任意向量$\boldsymbol{z}$的压缩版本。流行的压缩器包括那些实现量化[22]和稀疏化[24]的压缩器。根据它们的特点，压缩器可以分为无偏和一般（即可能有偏）。我们在下文中讨论这两种压缩器的变体，其中我们考虑一种用于一般压缩器的误差反馈技术，这种技术对于避免方差爆炸和保证收敛是很有用的。请注意，我们在本节中的偏差概念是在概率建模的背景下，无偏的压缩器意味着压缩向量的期望值（从该压缩器得到）等于原始向量。

2.3.1.1 没有误差反馈的无偏压制器

一个无偏的压缩器$\mathcal{C}(\boldsymbol{z})$满足以下两个特征:

$$\mathbb{E}[\mathcal{C}(\boldsymbol{z})|\boldsymbol{z}] = \boldsymbol{z} \tag{2.3}$$

$$\mathbb{E}[\|\mathcal{C}(\boldsymbol{z}) - \boldsymbol{z}\|^2|\boldsymbol{z}] \leq q\|z\|^2 \tag{2.4}$$

其中，$q \geq 0$是一个常数，用于捕获压缩器实现的相对近似间隙。直观地说，相对近似间隙指的是压缩后的向量与原始向量相比的相对误差。我们很容易看到，$q = 0$是$\mathcal{C}(\boldsymbol{z}) = \boldsymbol{z}$（即不压缩）的必要条件，$q = 1$是$\mathcal{C}(\boldsymbol{z}) = 0$（即不传输）的必要条件。一般来说，较大的$q$对应于由$\mathcal{C}(\boldsymbol{z})$产生的更压缩的向量。正如我们在接下来介绍的“随机-k”例子中所看到的，在某些情况下，我们可能会放大压缩结果以保证无偏性，这可能会产生一个大于1的q值。

*样例:*无偏压缩器的一个例子是一个随机量化器:

$$[\mathcal{C}(\boldsymbol{z})]_i = \begin{cases} \lfloor z_i \rfloor, \text{以 } \lceil z_i \rceil - z_i \text{ 的概率} \\ \lceil z_i \rceil, \text{以 } \lfloor z_i \rfloor - z_i \text{ 的概率} \end{cases} \tag{2.5}$$

为该向量的第i个分量，其中$\lfloor \cdot \rfloor$和$\lceil \cdot \rceil$分别表示下界（向下舍入为整数）和上界（向上舍入为整数）运算符。我们注意到，在浮点表示法的情况下，这里的整数可以是基数。可以很容易地看出，这种量化操作满足无偏性属性(2.3)。注意到量化操作给出$q = \max_{y \in [0,1]}(1-y)^2 y + y^2(1-y)$，我们有$q = 1/4$。

另一个例子是，从原始向量$\boldsymbol{z}$中随机选择k个分量，其概率同为k/d，并将结果放大d/k。即，

$$[\mathcal{C}(\boldsymbol{z})]_i = \begin{cases} \frac{d}{k} z_i, \text{以 } \frac{k}{d} \text{ 的概率} \\ 0, \text{以 } 1 - \frac{k}{d} \text{ 的概率} \end{cases} \tag{2.6}$$

为向量的第i个分量。这通常被称为随机-k稀疏化技术。显然，这种操作也是无偏的。(2.4)的左边是所有分量$[(d/k-1)^2 \cdot k/d + 1 \cdot (1-k/d)]z_i^2$的总和。因此，我们得到$q = d/k - 1$。

带有压缩更新的本地更新SGD。当使用压缩和本地更新SGD时，每一方都像往常一样计算其本地更新。这些更新在发送到聚合器之前被压缩，然后聚合器对压缩的更新进行平均，以获得下一个全局模型参数。假设一个有 τ 个迭代的回合从迭代t开始，这就给出了以下递推关系。

在不同的实现中，可以在服务器上应用另一种压缩操作，以保持相同的压缩水平（例如，量化精度或要传输的组件数量）。这样就可以得到

(6.14)和(6.15)中的操作是相似的，可能是整体近似间隙q不同。

收敛的边界。在适当选择学习率的情况下，使用（6.14）进行T次迭代后的最优性（以梯度的平方准则表示）可以被约束为。其中xt:=1m mi=1x(i)t对于所有t，即使没有压缩

方差放大。从（6.16）中，我们可以看到，当T足够大时，误差由第一项O 1+q √T 主导。当q很大时，我们需要将迭代次数T增加q2倍才能消除q的影响并达到相同的误差，这是有问题的，因为压缩的优势会被增加的计算量所抵消，特别是对于随机-k这样的压缩器，1+q与k成反比，正如我们前面讨论的那样。由于（6.16）的第一项也与随机梯度的方差成正比，为了简单起见，我们将其吸收到O(-)的符号中，这种现象在文献中也被称为方差吹胀。

接下来，我们将看到，误差反馈可以通过在本地积累压缩参数向量和实际参数向量之间的差异来解决方差吹大的问题，这样就可以在未来的通信回合中传输。

2.3.1.2 带有误差反馈的普通压缩机

我们首先介绍一个一般的（可能是有偏见的）压缩器。一般的压缩器C(z)满足以下属性。

其中，α 是一个常数，0≤α ¡1，表示压缩机实现的相对近似差距。与(6.10)和(6.11)中无偏压缩器的特性相比，关键的区别是一般压缩器不保证无偏性。当我们让α =q时，方程（6.11）和（6.17）基本上是相同的，只是为了收敛分析的目的，我们要求α ¡1。保持α 与q不同的另一个原因是为了区分两种类型的压缩机。满足（6.17）的压缩机也被称为α -收缩性压缩机。还有一个更严格的(6.17)版本，其中不等式在没有期望的情况下成立。

例子一般压缩器的一个典型例子是top-k稀疏化技术，它选择幅度最大的k个分量。这可以表示为：。

为矢量的第i个分量。由于这个操作对给定的z来说是确定的，所以它是有偏差的。我们可以得到α =1-kdb，因为z中其余分量的平方不能大于幅度最大的k个分量。

具有压缩更新和错误反馈的本地更新SGD。当使用错误反馈时，除了在客户端和服务器之间交换压缩的更新外，未被传达的部分（这里称为 ”错误”）将在本地累积。在下一轮中，累积的误差将被添加到该轮的最新更新中，这个和向量将被压缩器用来计算压缩向量。每一方i保留一个误

差向量e(i)，初始化为e(i)0=0。在每一轮r中，执行以下步骤。

1. 对于每一方i∈1, 2, . . . a. 从全局参数xr开始，计算局部梯度下降的τ步，以获得x(i) r, τ 。 b. 将累积误差与当前更新相加：z(i)r:= e(i)r+x(i)r, τ -xr。 c. 计算压缩结果 (i)r:= C z(i)r (这就是将被发送给聚合器的结果)。 d. 减去压缩结果，得到下一轮的剩余误差 e(i) r+1= z(i)r- (i)r。

2. 聚合器根据从各方收到的压缩更新来更新下一轮的全局参数，即138G. Joshi和S. Wang xr+1= xr+ 1 m m i=1 (i)r= xr+ 1 m m i=1 C z(i)r 。(6.19) 我们可以看到（6.14）和（6.19）的唯一区别是，我们现在对z(i)r进行压缩，这包括前几轮的累积误差。注意，为了方便起见，我们在这里使用r轮索引，而不是（6.14）中的迭代索引t。与(6.15)类似，上述程序也可以扩展到压缩和累积双方和聚合器的误差。

收敛的边界。与（6.16）类似，我们提出错误反馈机制的最优性约束。在适当选择学习率的情况下，我们有我们注意到，尽管（6.16）和（6.20）的左手边略有不同，但它们的物理含义是相同的。这种微小的差异是由于在推导这些界限时使用了不同的技术。与（6.16）相比，我们看到由于压缩而产生的近似差距，由α捕获，现在在（6.20）的第二项中。当T足够大时，我们现在的收敛率为O 1 √T ，这就避免了方差爆炸的问题。

请注意，由于我们要求0⩽α¡1，我们这里的分析对（6.13）中的随机-k压缩器不成立。(6.13)中的随机-k压缩器，但我们可以修改(6.13)，去掉放大系数d/k，因为我们不再要求无偏性了。所得的得到的压缩器满足α=1-k/d，这与top-k的压缩器相同。然而，在实践中，top-k通常比随机-k更有效，因为它的实际逼近的差距通常比1 -k/d的上界小得多。

这些结果表明，错误反馈机制通常比非错误反馈机制表现更好。然而，最近有工作表明，通过以系统的方式将有偏见的压缩器转化为无偏见的压缩器，我们实际上可能获得更好的性能。这是一个活跃的研究领域，从业者可能需要试验不同的压缩技术，以了解哪种技术对手头的问题效果最好。

自适应压缩率

压缩更新的SGD中的一个问题是如何确定压缩率（即（6.11）和（6.17）中的量q和α），以最小化达到目标函数的某个目标值的训练时间。在这种情况下，最佳压缩率取决于每个迭代中的计算和每个回合中的通信所产生的物理时间。这个问题类似于第6.2.1节中讨论的确定6Communication-Efficient Distributed Optimization Algorithms139的最佳局部更新数τ，但

这里的控制变量是压缩率。可以采用类似的方法来解决这个问题，即压缩率适应方法来自收敛边界，如第6.2.1节所述。为了克服在收敛边界中估计或消除未知参数的困难，也可以使用无模型的方法，如基于在线学习的方法。实质上，基于在线学习的方法采用探索-利用的方式，在最初几轮探索不同的压缩率选择，并逐渐切换到利用那些之前已经有利的压缩率。一个挑战是，探索需要有最小的开销，因为否则，即使与没有优化的情况相比，它也会延长训练时间。

为了促进有效的探索，可以制定一个问题来寻找最佳的压缩率，使训练时间最小化，以减少单位数量的经验风险。这个问题的确切目标是未知的，因为很难预测在使用不同的压缩率时训练将如何进展。然而，经验证据表明，对于一个给定的（当前）经验风险，我们可以假设之前使用的压缩率与未来经验风险的进展无关。再加上其他一些假设，我们可以把这个问题放在一个在线凸优化（OCO）框架中，它可以用在线梯度下降法解决，梯度是单位风险降低的训练时间相对于压缩率的导数。注意，这里的这个梯度与学习问题的梯度不同。然后，在线梯度下降程序是在每一轮的训练时间目标上使用梯度下降来更新压缩率，不同的轮次可以有不同的目标，这些目标是事先未知的。理论上可以证明，尽管我们只对每一轮的目标进行梯度下降，但累积的最优性差距（称为遗憾）在时间上呈亚线性增长，因此，当时间变为无穷大时，时间平均的遗憾会归于零。然而，这种方法需要一个梯度神谕，以每轮选择的压缩率给出准确的导数，这在实践中是很难得到的。

为了克服这个问题，中使用了一种基于符号的在线梯度下降方法，它只根据导数的符号而不是实际值来更新压缩率。估计导数的符号相对容易，只要估计正确符号的概率高于估计错误符号的概率，就可以保证有类似的亚线性遗憾。经验结果表明，这种算法能迅速收敛到一个接近最佳的压缩率，并比选择一个任意固定的压缩率提高性能。

2.4 模型剪枝

除了压缩参数更新外，模型本身也可以通过修剪（去除）神经网络中一些不重要的权重来压缩。这既加快了计算和通信的速度，又保持了最终模型的类似精度。图6.6显示了修剪的一个例子。一个著名的修剪方法是迭代训练和修剪模型，在包括多次SGD迭代的时间间隔内，删除一定比例的

小幅度权重。 当把剪枝和联合学习结合起来时，可以使用一个两阶段的程

图 2.2: 模型剪枝示意图

序，即在第一阶段对单方进行模型训练和剪枝，然后在涉及多方的常规联合学习过程中进一步剪枝。最初的修剪阶段允许联合学习从一个小的模型开始，与从完整的模型开始相比，可以节省计算和通信，同时随着模型及其权重在进一步修剪阶段的调整，仍然收敛到全局最优。为了确定哪些权重应该被修剪（或在第二阶段加回），可以制定一个目标，使修剪后的模型接近于原始模型，并在未来几轮中保持 "可训练性"。为了接近原始模型，可以采用标准的基于幅度的修剪，并适当选择修剪率，这样只有那些幅度足够小的权重可以被修剪掉。当从修剪后的模型中执行一步SGD时，可以使用经验风险降低的一阶近似值来捕获可训练性。基于这个近似值，我们可以求出应该修剪的权重集（如果之前已经修剪过，则可以加回来）以保持可训练性。总的来说，这种方法随着时间的推移调整模型的大小，以（近似）最大化训练效率。

2.5 讨论

在这一章中，我们回顾了在联邦学习中使用的具有通信高效的分布式优化算法，特别是减少通信频率的本地更新SGD算法和减少通信比特数的压缩方法。这些方法可以与其他算法相结合，提高联邦学习的收敛速度和效率。例如，边缘方可以使用加速[**?**]、方差缩减[**? ?**]或自适应优化方法，而不是使用经典的SGD作为本地求解器。

参考文献

[1] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[2] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[3] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM review*, 60(2):223–311, 2018.

[4] Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13(1), 2012.

[5] Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J Smola. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 661–670, 2014.

[6] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[7] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[8] Dong Yin, Ashwin Pananjady, Max Lam, Dimitris Papailiopoulos, Kannan Ramchandran, and Peter Bartlett. Gradient diversity: a key ingredient for scalable distributed learning. In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 1998–2007. PMLR, 09–11 Apr 2018. URL `https://proceedings.mlr.press/v84/yin18a.html`.

[9] Pratik Chaudhari and Stefano Soatto. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks, 2018.

[10] Behnam Neyshabur, Ryota Tomioka, Ruslan Salakhutdinov, and Nathan Srebro. Geometry of optimization and implicit regularization in deep learning, 2017.

[11] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information, 2017.

[12] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[13] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[14] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, et al. Large scale distributed deep networks. *Advances in neural information processing systems*, 25, 2012.

[15] Henggang Cui, James Cipar, Qirong Ho, Jin Kyu Kim, Seunghak Lee, Abhimanu Kumar, Jinliang Wei, Wei Dai, Gregory R Ganger, Phillip B Gibbons, et al. Exploiting bounded staleness to speed up big data analytics. In *2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14)*, pages 37–48, 2014.

[16] Sanghamitra Dutta, Gauri Joshi, Soumyadip Ghosh, Parijat Dube, and Priya Nagpurkar. Slow and stale gradients can win the race: Error-runtime trade-offs in distributed sgd, 2018.

[17] Suyog Gupta, Wei Zhang, and Fei Wang. Model accuracy and runtime tradeoff in distributed deep learning: A systematic study. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 171–180. IEEE, 2016.

[18] Jian Zhang and Ioannis Mitliagkas. Yellowfin and the art of momentum tuning. *arXiv preprint arXiv:1706.03471*, 2017.

[19] Sebastian U Stich. Local sgd converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018.

[20] Jianyu Wang and Gauri Joshi. Cooperative sgd: A unified framework for the design and analysis of communication-efficient sgd algorithms, 2019.

[21] Hao Yu, Sen Yang, and Shenghuo Zhu. Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5693–5700, 2019.

[22] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. *Advances in neural information processing systems*, 30, 2017.

[23] Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes signsgd and other gradient compression schemes. In *International Conference on Machine Learning*, pages 3252–3261. PMLR, 2019.

[24] Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified sgd with memory. *Advances in Neural Information Processing Systems*, 31, 2018.