

第六讲：神经网络的分布式学习

南京大学人工智能学院

申富饶

目录

CONTENTS

- 01.** 分布式学习简介
- 02.** 常用的划分方法
- 03.** 常用的通讯机制
- 04.** 常用的模型聚合方法
- 05.** 扩展：联邦学习

01

分布式学习简介

什么是分布式学习

- 随着模型规模、训练数据量等不断增长，单个计算机已经无法完成机器学习训练过程中一些的计算、存储任务。
- 分布式机器学习尝试用若干廉价的、普通的机器取代单个高性能计算机，来完成和加速机器学习的训练过程，即利用更多的机器，处理更多的数据。
- 神经网络的分布式学习主要研究如何使用计算机集群来加速大规模神经网络模型的训练。

为什么需要分布式学习

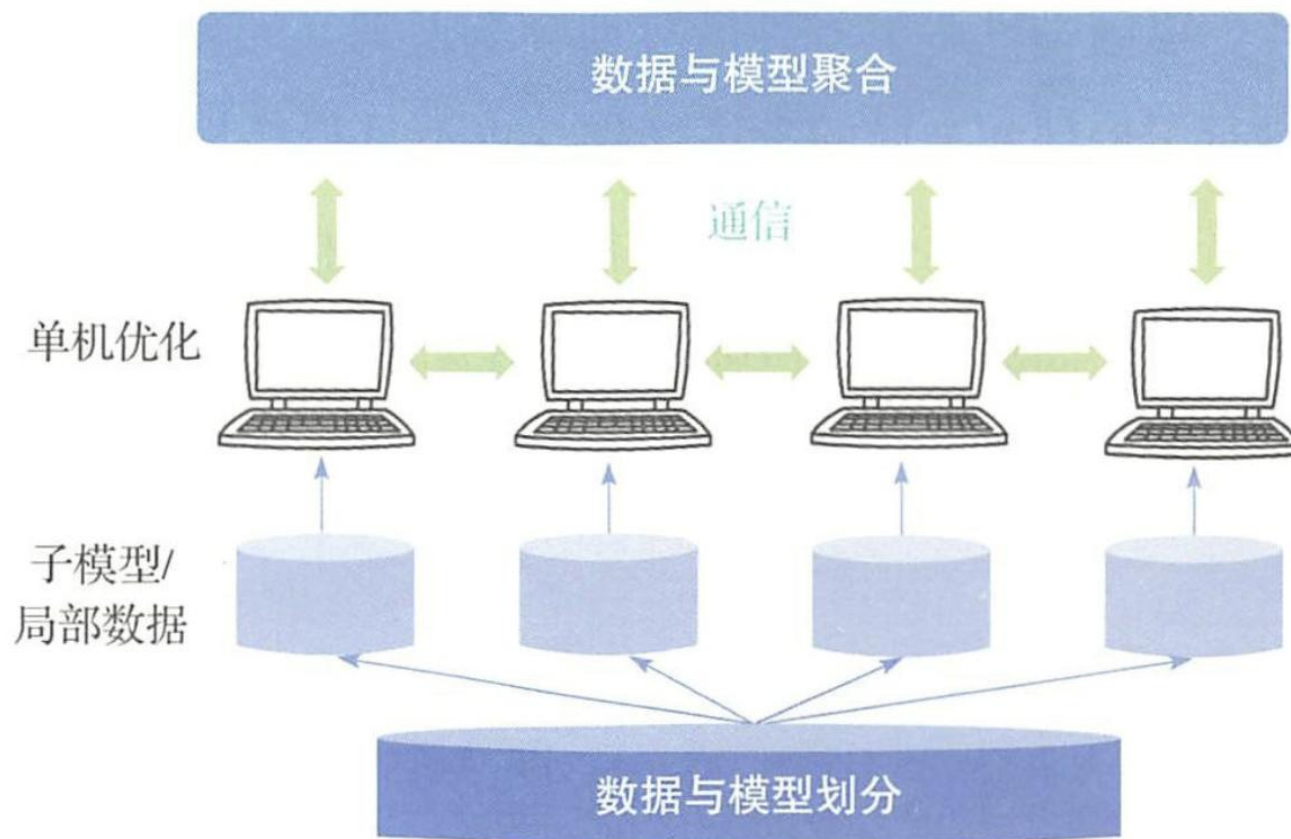
- 神经网络已经为人工智能领域带来了巨大的发展和进步，但训练神经网络模型需要大量的计算。
- 有研究表明，在具有一个现代 GPU 的机器上完成一次 ImageNet 等基准数据集的训练可能要耗费**多达一周**的时间！
- 而借助分布式学习可以通过上千个 GPU 集群将 ImageNet 的训练时间**降低至几分钟**。

为什么需要分布式学习

- 在实际生活中运用神经网络技术解决生产问题时，海量训练数据、问题复杂程度高等诸多挑战是难以避免的，因此必须使用更复杂的神经网络模型来解决问题，这些问题单机训练是无法解决的。
- 因此当面临以下三种情况时，一般会优先考虑使用分布式学习的方法来训练神经网络模型：
 - 第一，计算量太大；
 - 第二，训练数据太多；
 - 第三，模型规模太大。

分布式学习的基本流程

- 分布式学习的基本流程可以划分为以下四个主要部分：**数据与模型划分单元**、**单机优化单元**、**通信模块单元**以及**数据与模型聚合单元**。



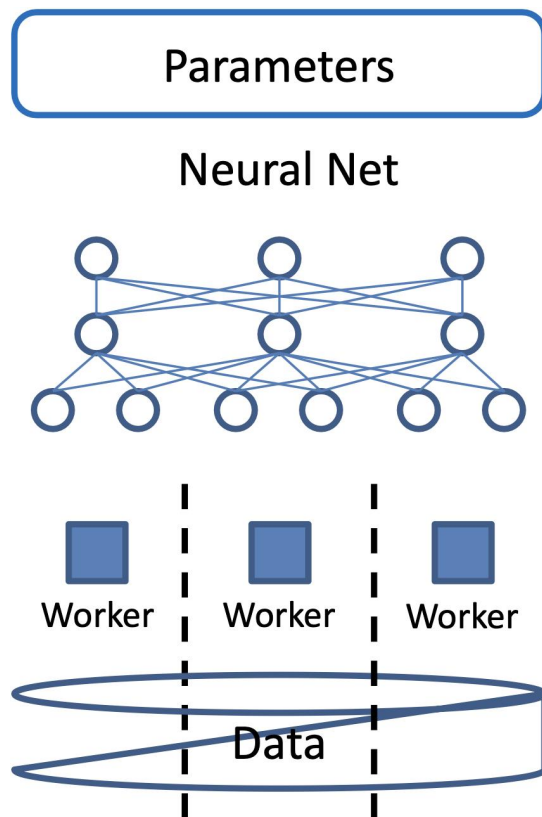
02

常用的划分方法

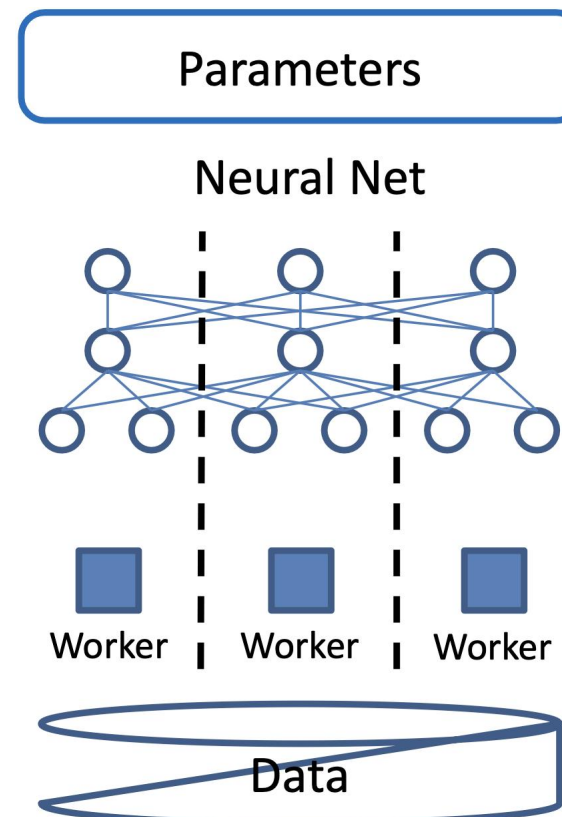
常用的划分方法

- 当我们拥有大量训练数据或大规模神经网络模型时，由于存储容量和计算性能受限等原因，通常无法在单个工作节点上完成模型学习，需要先将任务进行划分并分配给各个工作节点。
- 常用的划分方法：
 - 数据划分
 - 模型划分

常用的划分方法示例



数据并行



模型并行

基于数据的划分

- **数据划分**：当数据规模过大，无法单独存储在一个工作节点上，数据将被分割并分发到每个工作节点。然后，每个工作节点使用分配得到的数据对神经网络模型进行本地训练。
- **要求**：尽可能让每台机器上的局部训练数据与原训练数据独立同分布。
- 常用的划分方式有：**随机采样、置乱切分**

基于数据的划分

➤ 随机采样：

- 把训练数据作为采样的数据源，通过有放回的方式进行随机采样，然后按照每个工作节点的容量为其分配相应数目的训练样本。
- **优点：** 随机采样方法可以保证每台机器上的局部训练数据与原始训练数据是独立同分布的，因此在训练的效果上有理论保证。
- **缺点：** 因为训练数据较大，实现全局采样的计算复杂度比较高；其次，如果随机采样的次数小于数据样本的数目，可能有些训练样本会一直未被选出，导致辛苦标注的训练样本并没有得到充分的利用。

基于数据的划分

➤ 置乱切分：

- 将训练数据进行乱序排列，然后按照工作节点的个数将打乱后的数据顺序划分成相应的小份，随后将这些小份数据分配到各个工作节点上。每个工作节点在进行模型训练的过程中，只利用分配给自己的局部数据。一定阶段后，再次进行数据打乱和重新分配。
- **比较：**置乱切分方法相比于随机采样方法，虽然数据的分布与原始数据分布略有偏差，但是其计算复杂度比全局随机采样要小很多，而且置乱切分能保留每一个样本，直观上对样本的利用更充分。

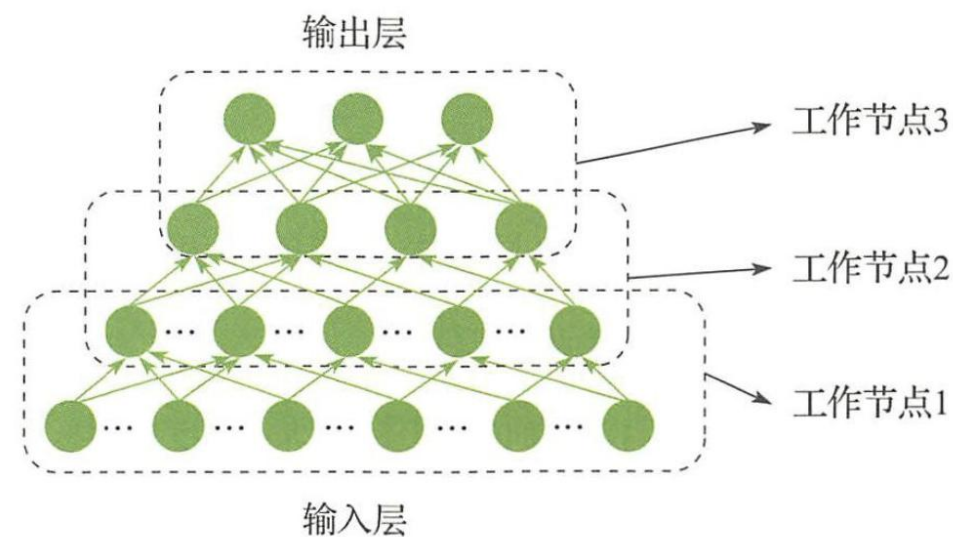
基于模型的划分

- **模型划分**：对于规模大、参数多、无法在单个工作节点存储的神经网络模型，有必要将神经网络模型的结构进行划分，并分配给各个工作节点。
- 与简单的线性模型不同，神经网络是高度非线性的，各个工作节点不能相对独立地完成对自己负责的参数的训练和更新，必须依赖与其他工作节点的协作。
- 常用的划分方式有：**横向按层划分、纵向跨层划分、模型随机划分。**

基于模型的划分

➤ 横向按层划分：

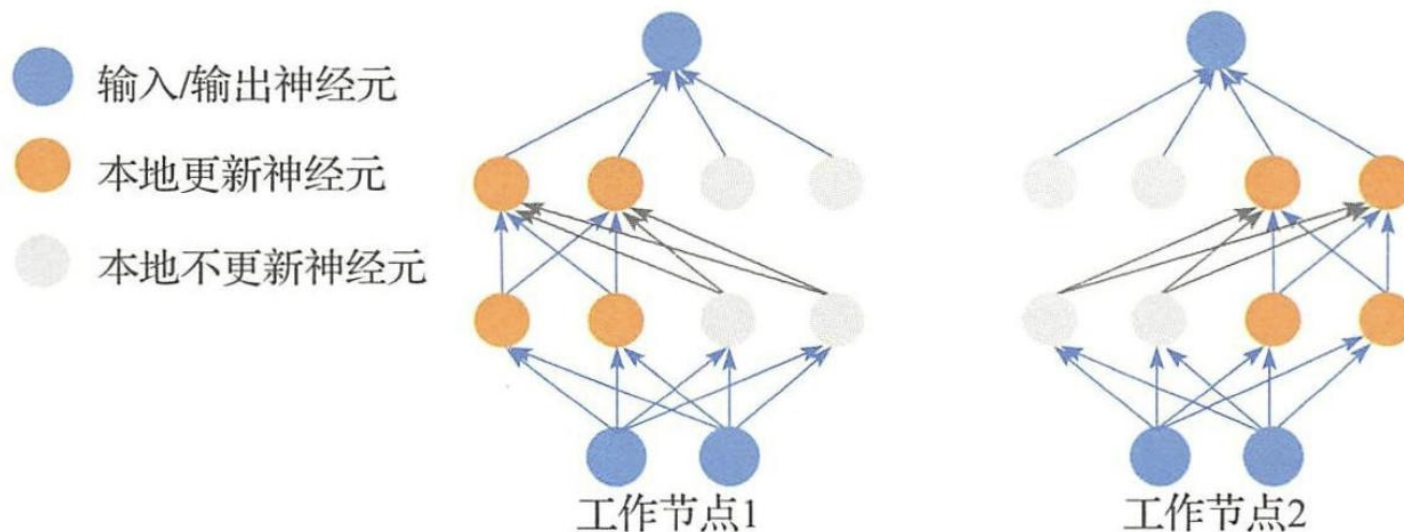
- 将神经网络每两层间的参数、激活函数值和误差传播值存储于一个工作节点。
- 当前传过程需要前一层的激活函数值时，向对应的工作节点发出请求并接收相应的数据，然后再利用这些数据计算本层的前向激活函数值；
- 当后传过程需要后一层的误差传播值时，同样，向对应的工作节点发出请求并接收相应的数据，然后利用这些数据计算本层的后向误差传播值。



基于模型的划分

➤ 纵向跨层划分：

- 将每一层的参数均等地划分成若干份，每一个工作节点存储由所有层的一部分参数构成的子网络模型。
- 在前传和后传过程中，如果需要子模型以外的激活函数值和误差传播值，则向对应的工作节点发出请求并接收相应的数据。



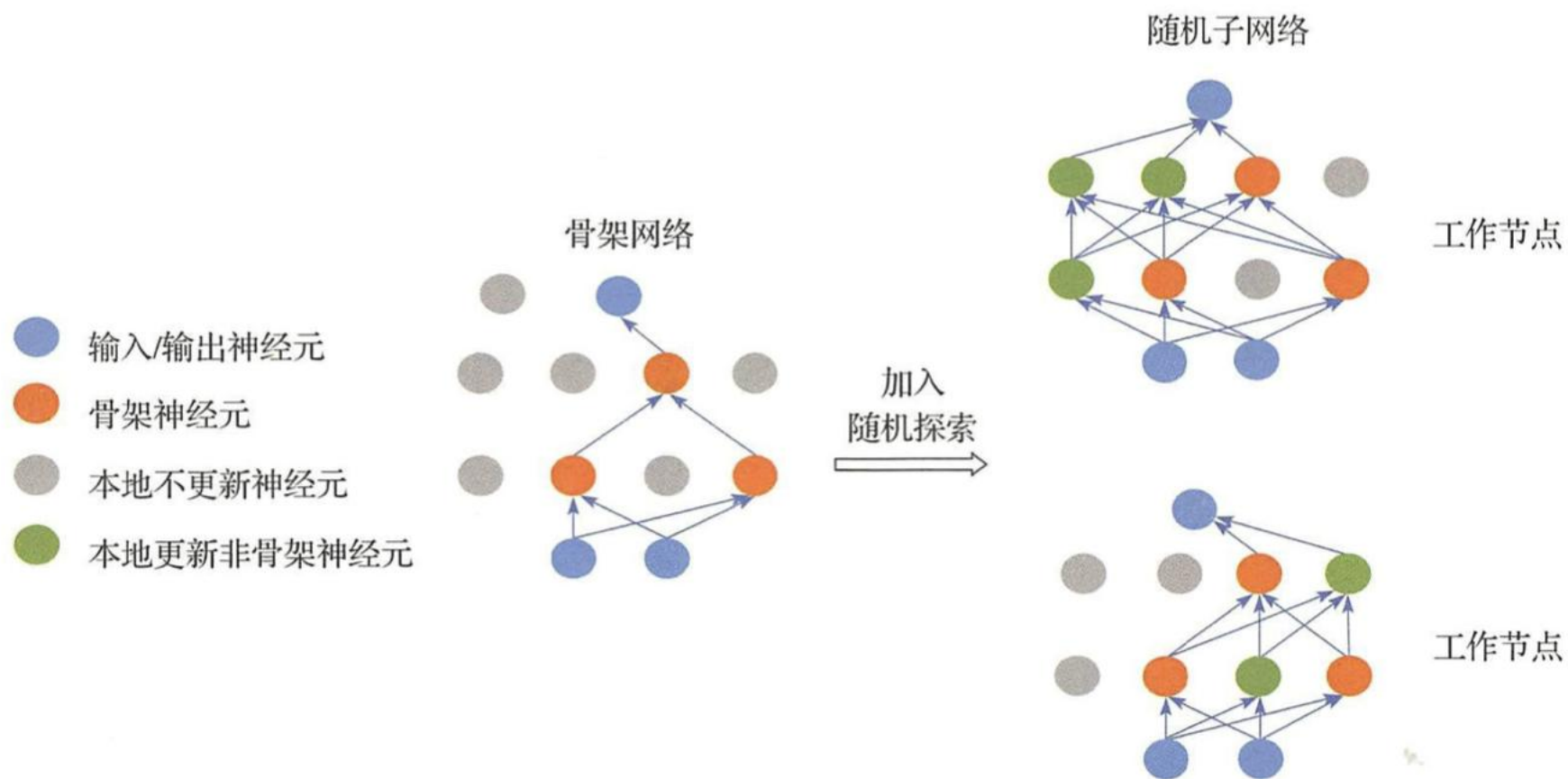
基于模型的划分

➤ 模型随机划分：

- 人们发现神经网络具有一定的冗余性，可以找到一个规模小很多的子网络（称为**骨架网络**），其效果与原网络差不多。
- 可以首先按照某种准则，在原网络中选出骨架网络，作为公用子网络存储于每个工作节点。除骨架网络之外，每个工作节点还会随机选取一些其他节点存储，以探索骨架网络之外的信息。骨架网络周期性地依据新的网络重新选取，而用于探索的节点也会每次随机选取。

基于模型的划分

➤ 模型随机划分示例：



03

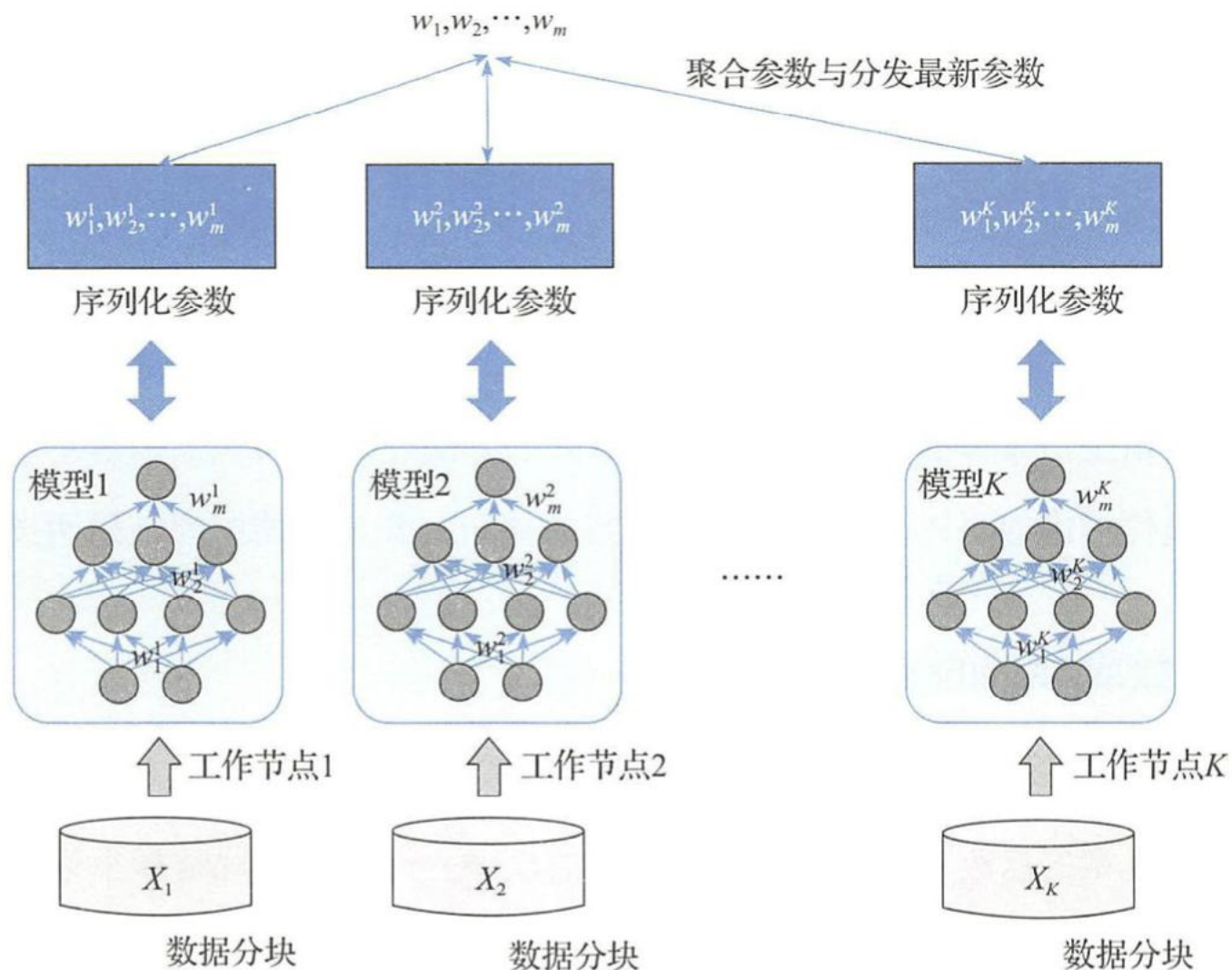
常用的通信机制

通信的内容

- 通信的内容与划分方法有关：
- 基于数据划分的方法：工作节点各自完成本地的学习任务，然后互相交流各自对模型的修改。因此，在此情形下通信的内容是**模型的参数或者参数的更新**。
- 基于模型划分的方法：各个工作节点利用同一份数据对模型的不同部分进行训练，每个节点要依赖其他节点的中间计算结果。因此，在此情形下通信的内容是**计算的中间结果**。

通信的内容

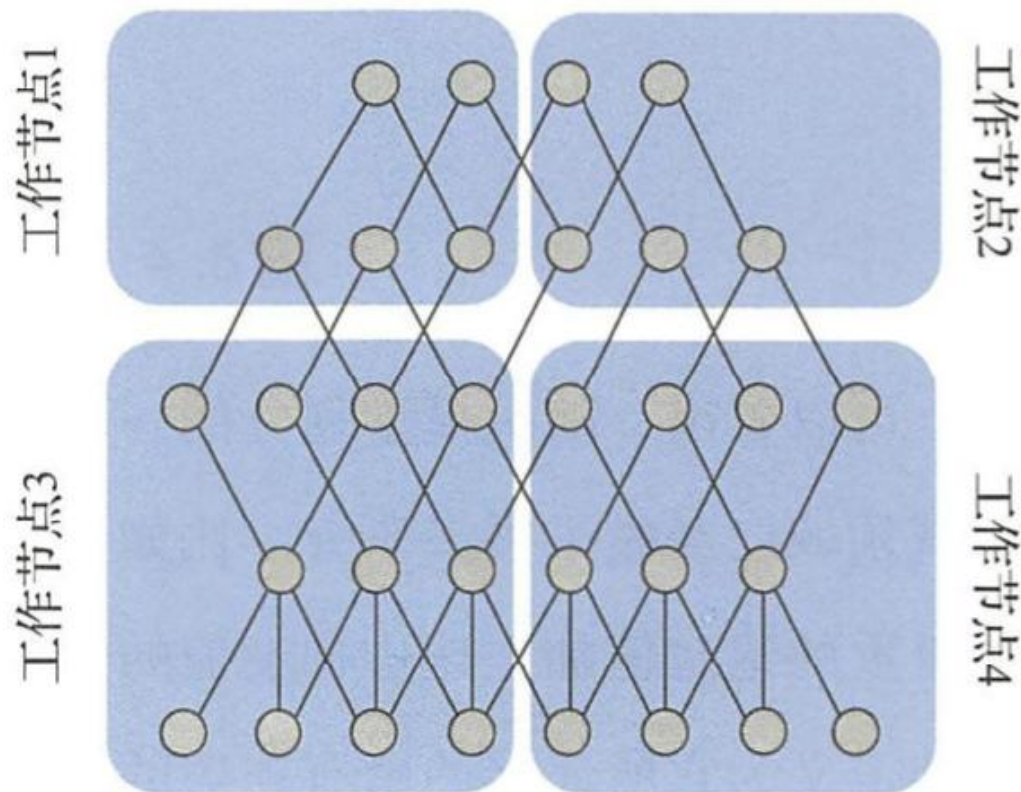
- 参数（或参数的更新）示例



通信的内容

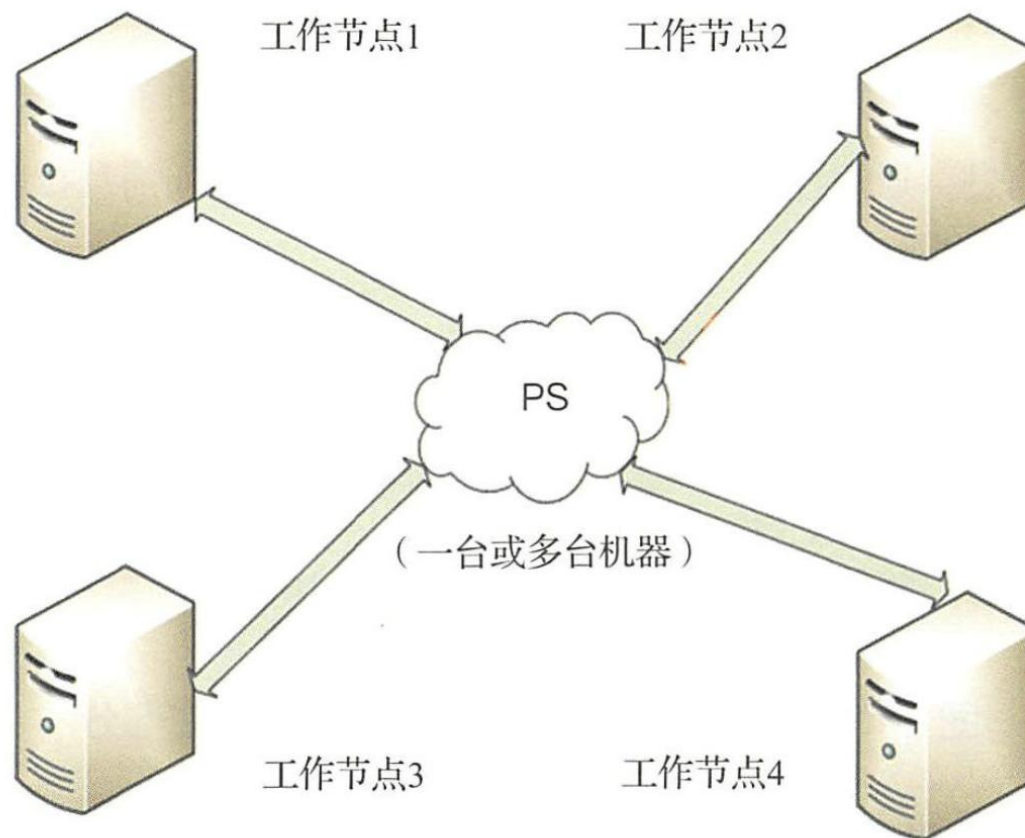
● 计算的中间结果示例

- 激活函数值
- 误差信息
- 梯度更新
- 等等



通信的拓扑结构

- 在神经网络的分布式学习中，常采用**基于参数服务器的通信拓扑**



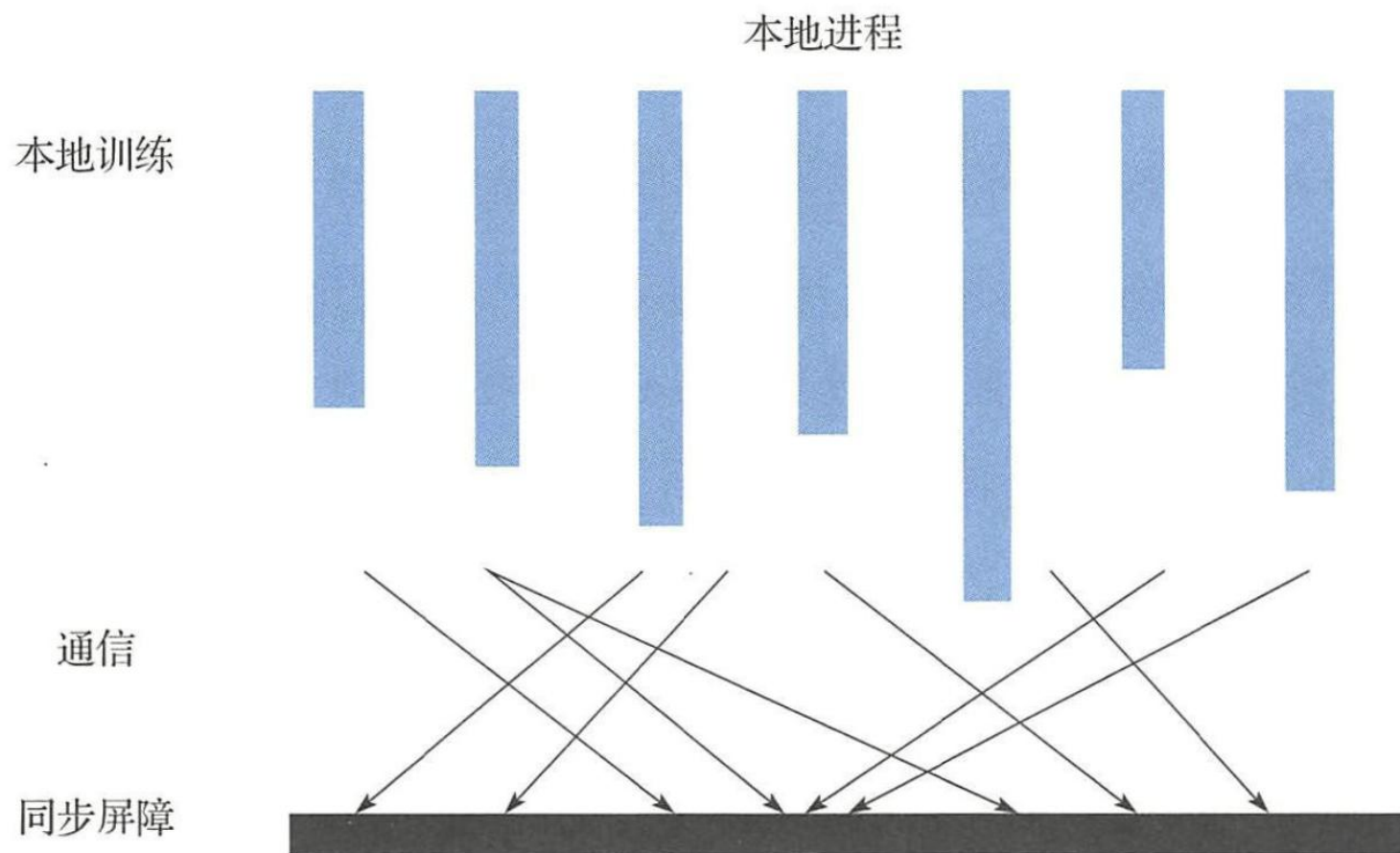
通信的步调

- 参数服务器这种通信拓扑将各个工作节点之间的计算相互隔离，取而代之的是工作节点与参数服务器之间的交互。利用参数服务器提供的参数存取服务，各个工作节点可以独立于彼此工作。
- 工作节点对全局参数的访问请求通常分为获取参数（PULL）和更新参数（PUSH）两类。服务器节点响应工作节点的请求，对参数进行存储和更新。
- 有了参数服务器，通信的步调既可以是**同步**的，也可以是**异步**的，甚至是**同步和异步混合**的。

通信的步调

➤ 同步通信

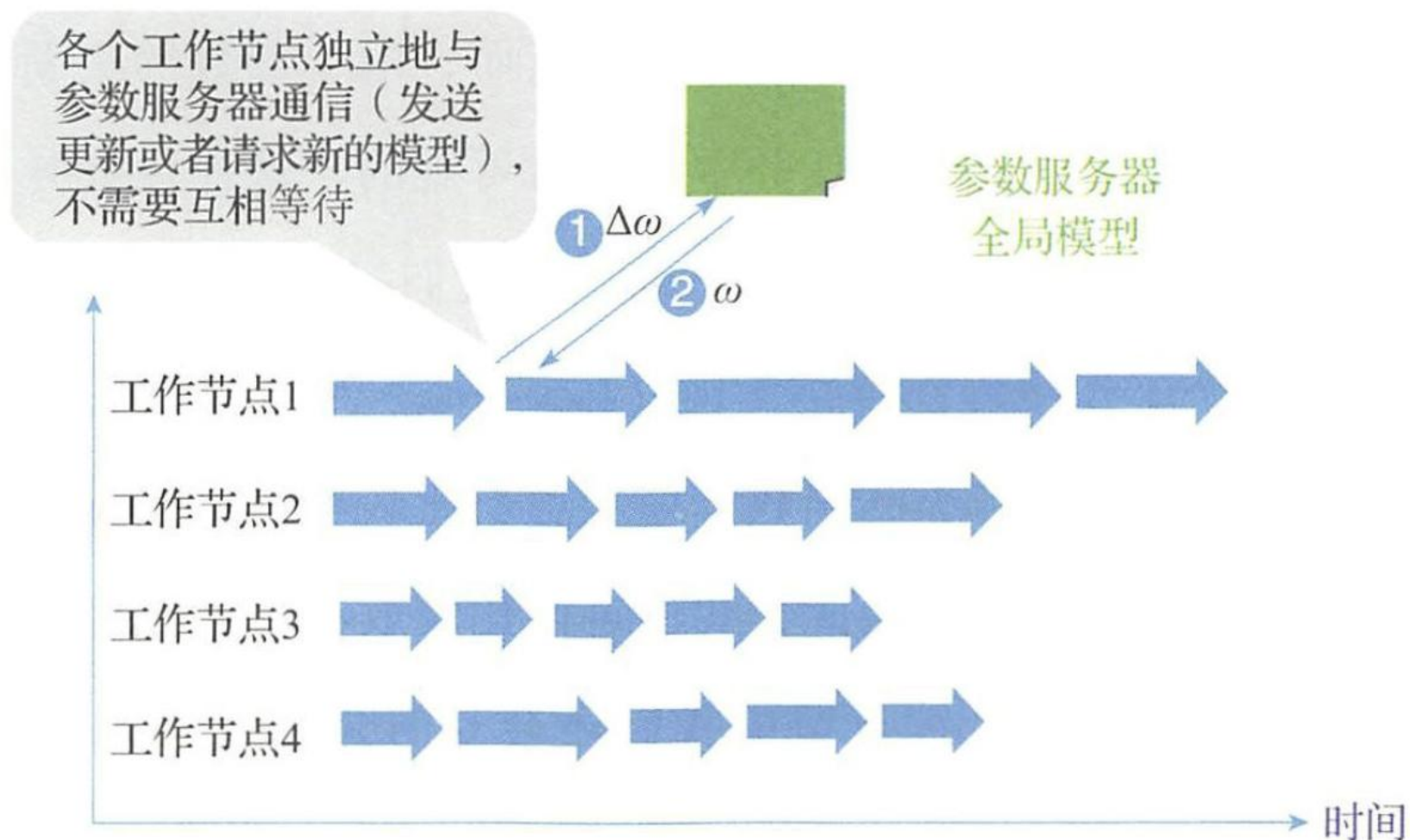
当集群中的一个工作节点完成本轮迭代后，需要等待集群中的其他工作节点都完成各自的任务，才能共同进行下一轮迭代。



通信的步调

➤ 异步通信

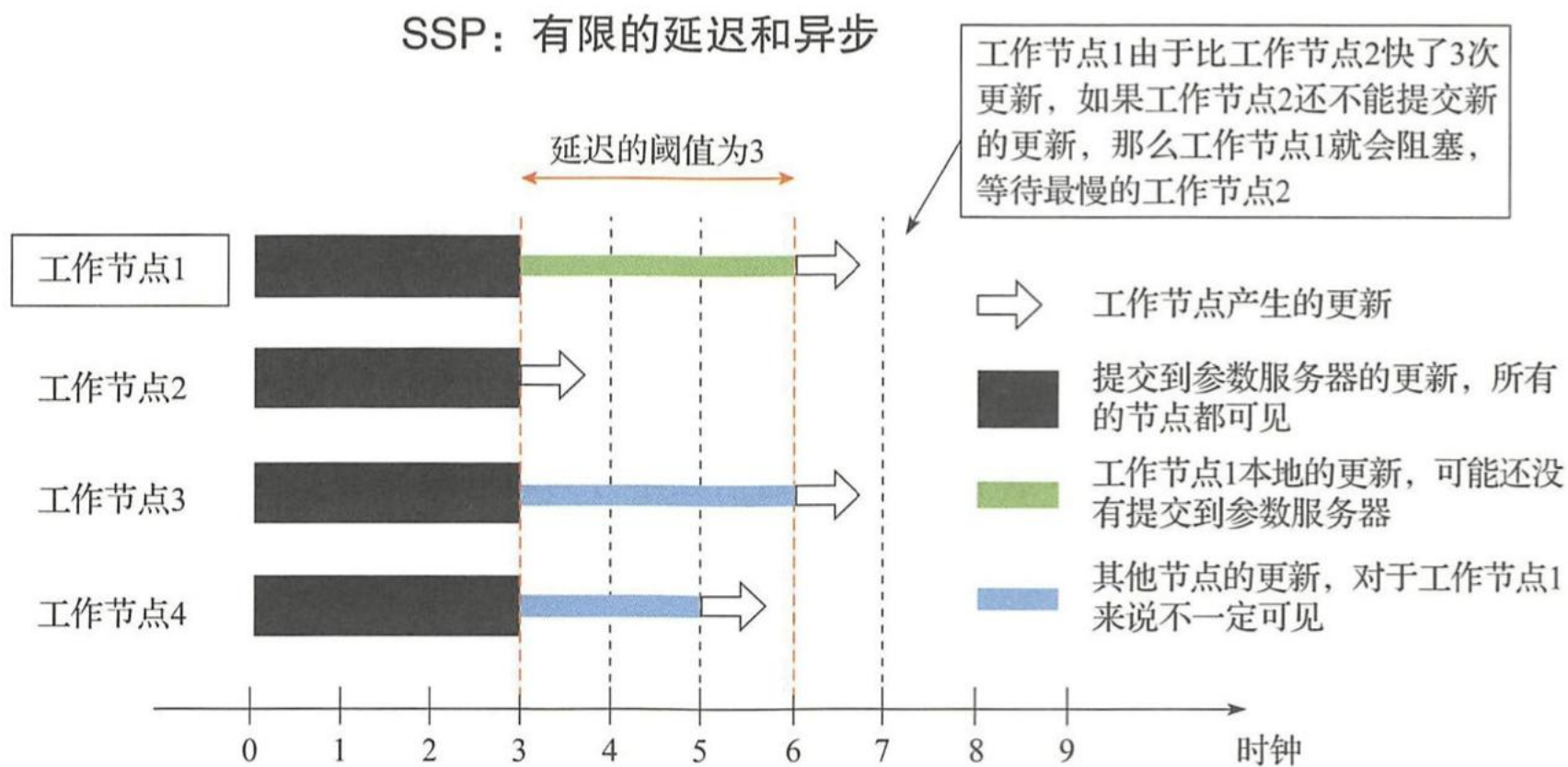
当集群中的一个工作节点完成本轮迭代后，无须等待集群中的其他工作节点，就可以继续进行后续训练，因此系统效率可以大大提高。



通信的步调

➤ 同步和异步的平衡

延时同步并行 SSP：控制最快和最慢节点之间相差的迭代次数不超过预设的阈值



04

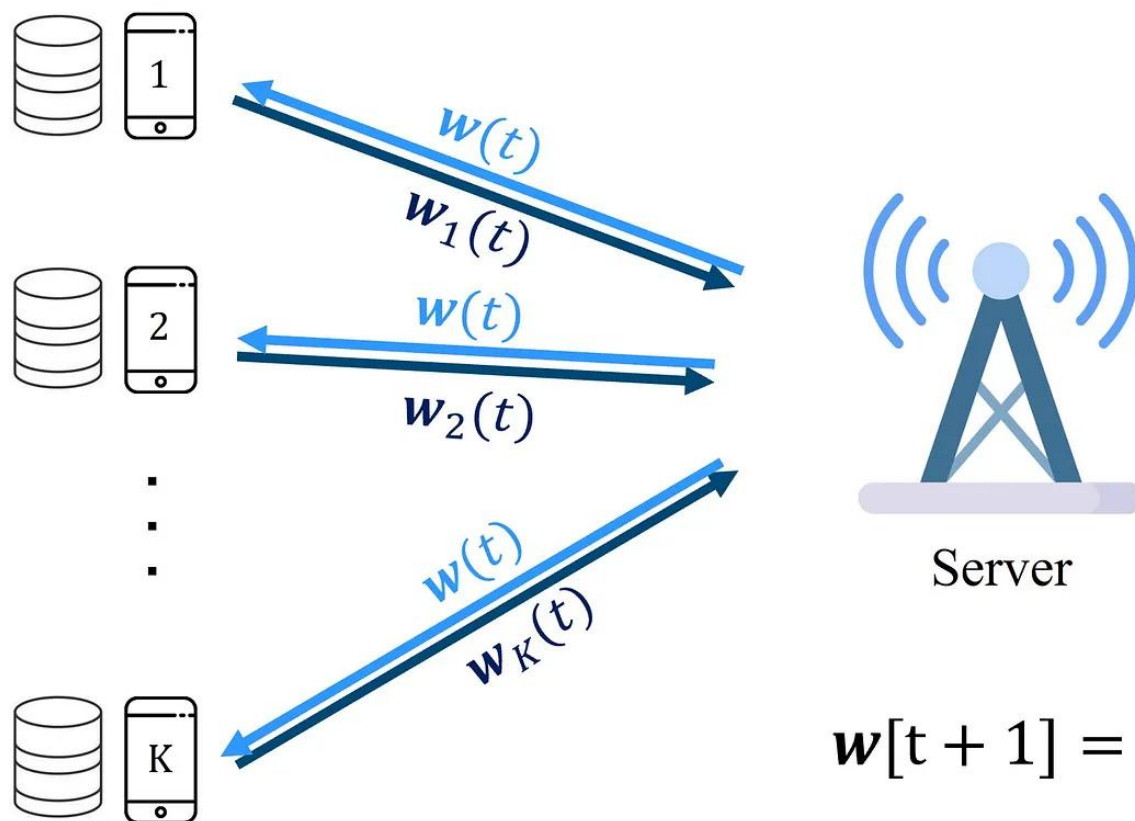
常用的模型聚合方法

常用的模型聚合方法

- 聚合是分布式学习特有的逻辑，当划分的数据使用分布式训练后，需要对多个设备的本地模型进行聚合。
- 有效的聚合往往会带来更好的加速效果。通常我们对聚合方法有以下需求：
 - 聚合本身的时间代价比较少，这样给整个学习流程带来的额外负担较小；
 - 聚合算法合理、有效，整体的收敛性仍然能保持与单机算法大体一致。
- 常用的模型聚合方法有：**基于模型加和的聚合、基于模型集成的聚合**

基于模型加和的聚合

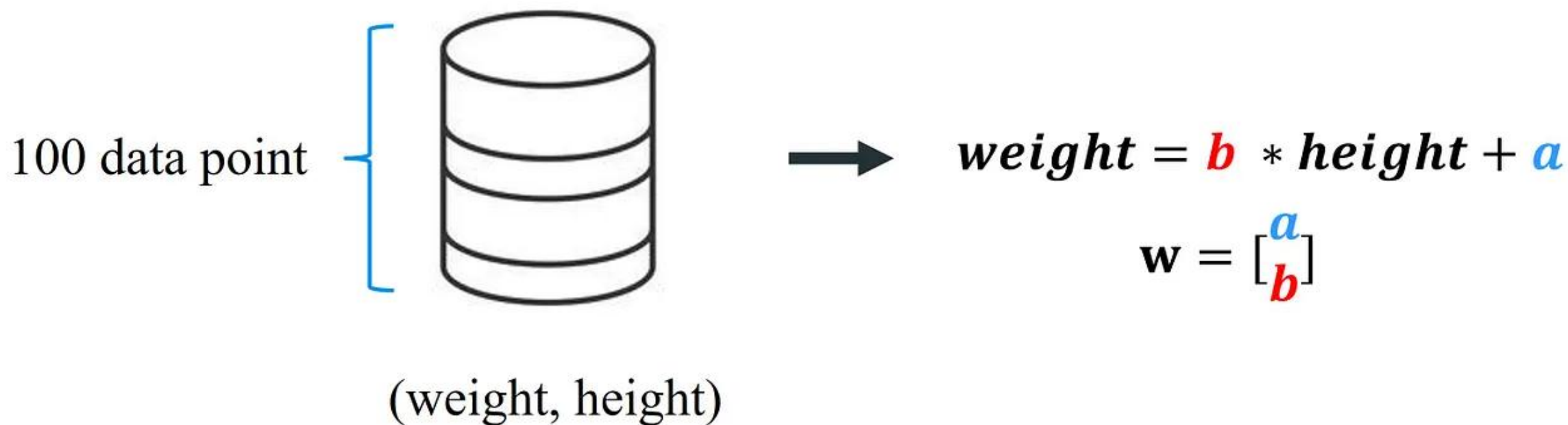
基于模型加和的聚合方法是指在参数服务器上将来自不同工作节点的模型或者模型更新进行加权求和。



$$\mathbf{w}[t + 1] = \frac{1}{K} \sum_{i=1}^K \mathbf{w}[t]$$

示例

- 假如我们想学习人的身高和体重之间的线性关系，并且我们拥有100个人的体重和身高数据，那么我们可以用这些数据来训练一个模型，如下图所示， a 和 b 是需要学习的参数



示例

- 假设使用梯度下降的方法进行学习

➤ 首先，假设设置 $a = 0$ 和 $b = 2$ ，并计算每个数据点的模型如下：

$$\text{weight} = b * \text{height} + a$$
$$a = 0, b = 2$$

100 data point	{	Person 1	$72 = 2 * 1.70 + 0$
		Person 2	$75 = 2 * 1.83 + 0$
		\vdots	
		Person 100	$65 = 2 * 1.69 + 0$

示例

➤ 然后，针对所有数据点计算该模型的误差 f ：

$$weight = \textcolor{red}{b} * height + \textcolor{blue}{a}$$

Person 1	$72 = \textcolor{red}{2} * 1.70 + \textcolor{blue}{0} \rightarrow error_1 = 72 - \textcolor{red}{2} * 1.70 + \textcolor{blue}{0}$
Person 2	$75 = \textcolor{red}{2} * 1.83 + \textcolor{blue}{0} \rightarrow error_2 = 75 - \textcolor{red}{2} * 1.83 + \textcolor{blue}{0}$
	\vdots
Person 100	$65 = \textcolor{red}{2} * 1.69 + \textcolor{blue}{0} \rightarrow error_{100} = 65 - \textcolor{red}{2} * 1.69 + \textcolor{blue}{0}$

$$\min f = \frac{\textcolor{green}{1}}{\textcolor{green}{100}} \sum_{i=1}^{100} error_i^2$$

示例

➤ 接下来，以 b 为例，计算误差 f 相对于参数 b 的梯度

$$f = \frac{1}{100} \sum_{i=1}^{100} error_i^2 = \frac{1}{100} \sum_{i=1}^{100} (weight_i - \mathbf{b} * height_i + \mathbf{a})^2$$

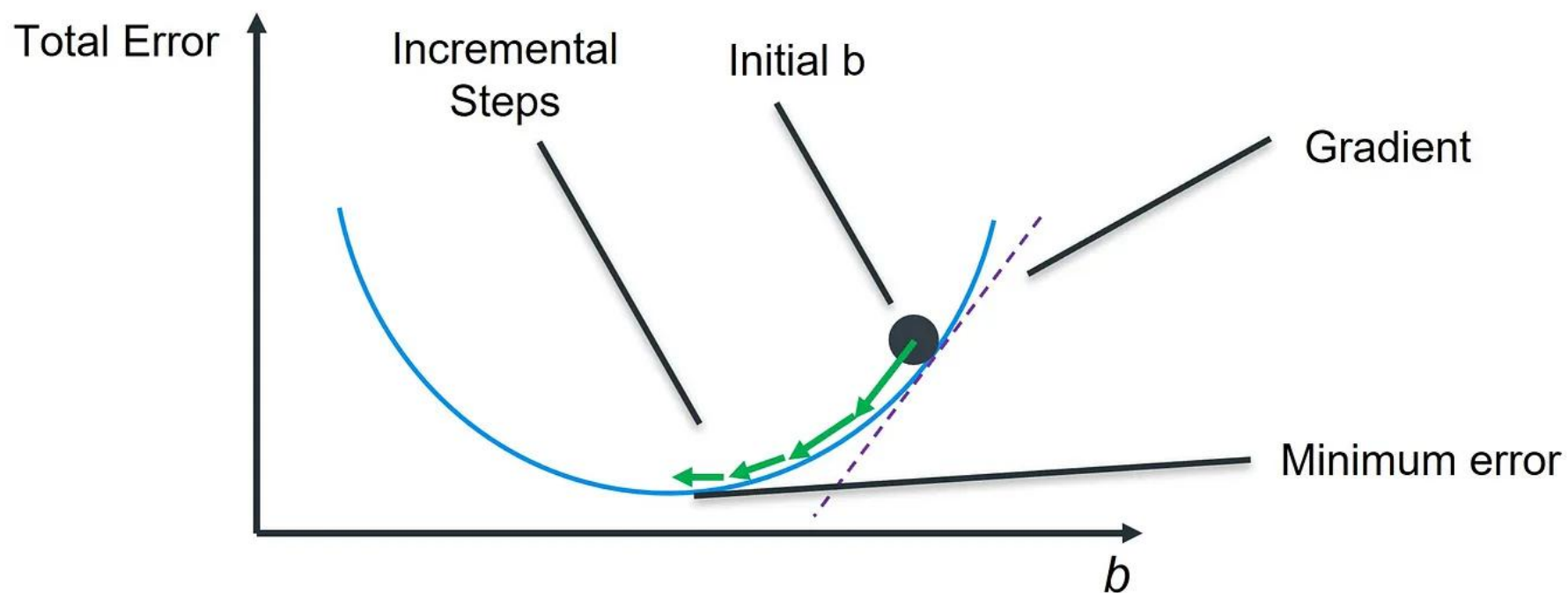
$$\nabla f_b = \frac{1}{100} \sum_{i=1}^{100} 2 * (weight_i - \mathbf{b} * height_i + \mathbf{a}) * height_i$$

$$\nabla f_b \text{ of person } i = \nabla f_b^i$$

示例

- 最后，使用梯度下降法找到 b 的最优值：

$$b_{new} = b_{previous} - \lambda * \nabla f_b (b_{previous})$$



示例-分布式

- 通常来说，如果采用集中学习的方式，我们可以使用下式来计算梯度：

$$\nabla f_b \approx \frac{1}{100} \sum_{i=1}^{100} \nabla f_b^i$$

- 但也可以考虑将上式分解成两部分求和，这意味着我们不需要将所有 100 个数据点放在一个地方：

$$\nabla f_b = \frac{1}{100} \sum_{i=1}^{100} \nabla f_b^i = \frac{1}{2} \left(\frac{1}{50} \sum_{i=1}^{50} \nabla f_b^i + \frac{1}{50} \sum_{i=50}^{100} \nabla f_b^i \right)$$

Averaging

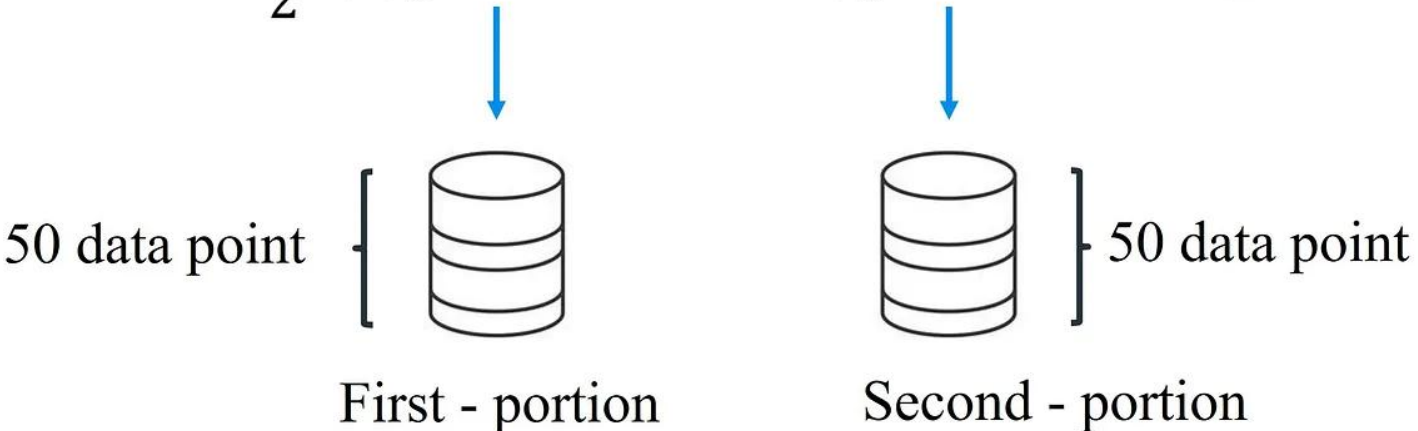
$\nabla f_b^{first-portion}$

$\nabla f_b^{second-portion}$

示例-分布式

- 即我们可以将数据分成两部分，放在两个客户端上分布式地计算每个部分的梯度：

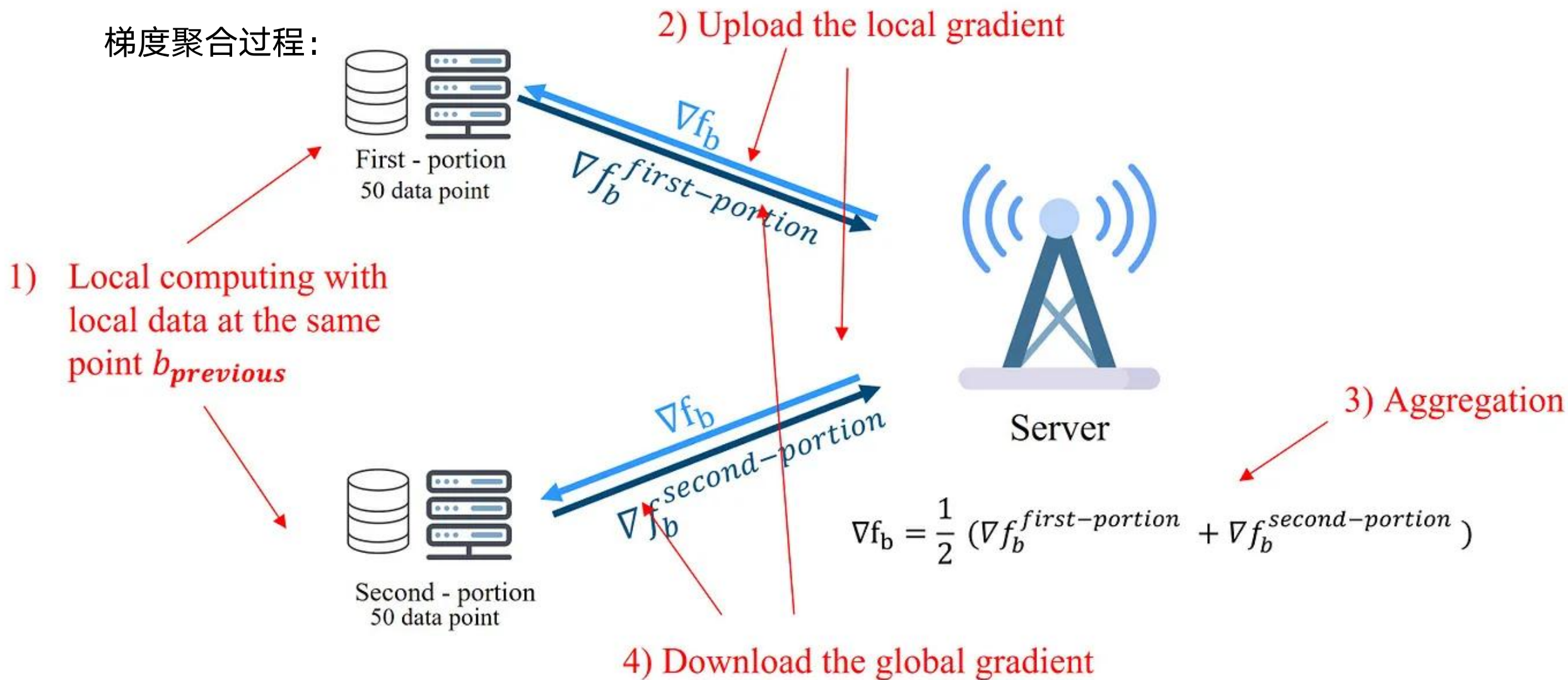
$$\begin{aligned}\nabla f_b &= \frac{1}{100} \sum_{i=1}^{100} \nabla f_b^i = \frac{1}{2} \left(\frac{1}{50} \sum_{i=1}^{50} \nabla f_b^i + \frac{1}{50} \sum_{i=50}^{100} \nabla f_b^i \right) \\ &= \frac{1}{2} \left(\nabla f_b^{first-portion} + \nabla f_b^{second-portion} \right)\end{aligned}$$



First - portion Second - portion

示例-分布式

梯度聚合过程：



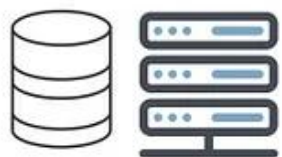
示例-分布式

- 此外，我们还可以先让每个客户端使用局部梯度来计算局部模型的参数，服务器在收到所有客户端的模型之后，将这些模型的参数进行平均，从而得到新的全局模型参数。



First - portion
50 data point

$$b_{new}^1 = b_{previous} - \lambda * \nabla f_b^{\text{first-portion}}(b_{previous})$$



Second - portion
50 data point

$$b_{new}^2 = b_{previous} - \lambda * \nabla f_b^{\text{second-portion}}(b_{previous})$$

+

$$\begin{aligned} \frac{1}{2}(b_{new}^1 + b_{new}^2) &= b_{previous} - \lambda * \frac{1}{2}(\nabla f_b^{\text{first-portion}}(b_{previous}) + \nabla f_b^{\text{second-portion}}(b_{previous})) \\ &= b_{previous} - \lambda * \nabla f_b(b_{previous}) \\ &= b_{new} \end{aligned}$$

示例-分布式

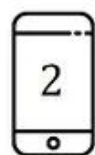
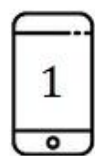
参数聚合过程:

1) Local computing with local data

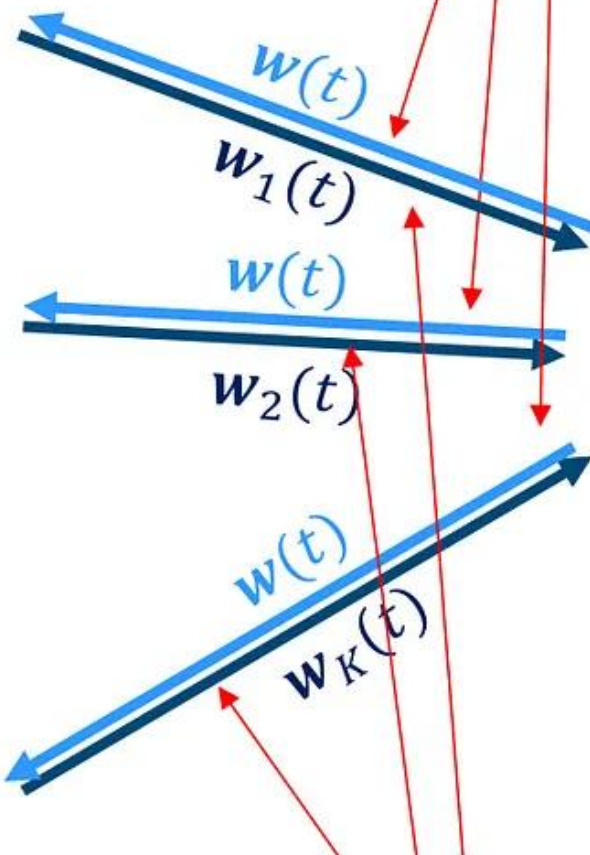
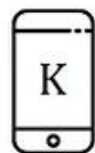
2) Upload the local model

3) Aggregation

4) Download the global model



...



Server

$$\mathbf{w} = \begin{bmatrix} a \\ b \end{bmatrix}$$

$$\mathbf{w}[t + 1] = \frac{1}{K} \sum_{i=1}^K \mathbf{w}[t]$$

基于模型加和的聚合

- 但是，并非所有情况下这类基于模型参数加和的聚合方法都是可取的。事实上，只有在凸优化问题中这种简单加和的手段才能保证训练性能。
- 具体地，假设损失函数关于模型参数是凸函数，于是以下不等式成立：

$$l(g((\bar{w}; x)), y) = l\left(g\left(\frac{1}{K} \sum_{k=1}^K w^k; x\right), y\right) \leq \frac{1}{K} \sum_{k=1}^K l(g(w^k; x), y)$$

- 其中，左端是参数平均后的模型 \bar{w} 对应的损失函数取值，右端是各个局部模型的损失函数值的平均值。这说明，在凸优化问题中，平均模型的性能不会低于原有各个模型性能的平均值。

基于模型集成的聚合

- 但是，对于非凸的神经网络，由于损失函数关于模型参数非凸。所以，上述不等式将不再成立，模型的性能在参数平均后也不再具有保证。
- 为了解决这个问题，人们提出了**基于模型集成的聚合方法**。

基于模型集成的聚合

- 虽然神经网络的损失函数关于模型参数是非凸的，但是它关于模型的输出一般是凸的。这时利用损失函数的凸性可以得到如下不等式：

$$l\left(\frac{1}{K}\sum_{k=1}^K g(w^k; x), y\right) \leq \frac{1}{K}\sum_{k=1}^K l(g(w^k; x), y)$$

- 其中，左端是对不同局部模型的输出进行平均后对应的损失函数取值，右端是局部模型的损失函数值的平均值。所以，如果**对局部模型的输出进行加和平均**，所得到的预测结果要好于局部模型预测结果的平均值。
- 这种对模型输出进行加和平均的方法称为**模型集成(ensemble)**。通过集成多个模型的预测结果，可以取得比单个模型更好的性能。

基于模型集成的聚合

- 虽然模型集成在模型精度上有更好的保障，但是模型集成后产生的新模型参数量是局部模型 K 倍。考虑到模型聚合会在分布式学习的迭代算法中多次发生，这很可能导致**模型规模爆炸**的问题。
- 人们提出了一种用压缩方法来降低集成模型规模的算法，称为“**集成一压缩**”的模型聚合方法。利用模型压缩技术（比如知识蒸馏），获得与局部模型大小相同的新模型作为最终的聚合结果。
- 这种集成一压缩的聚合方法，既可以通过集成获得性能提升，又可以在整个机器学习的迭代过程中保持全局模型的大小基本不变。

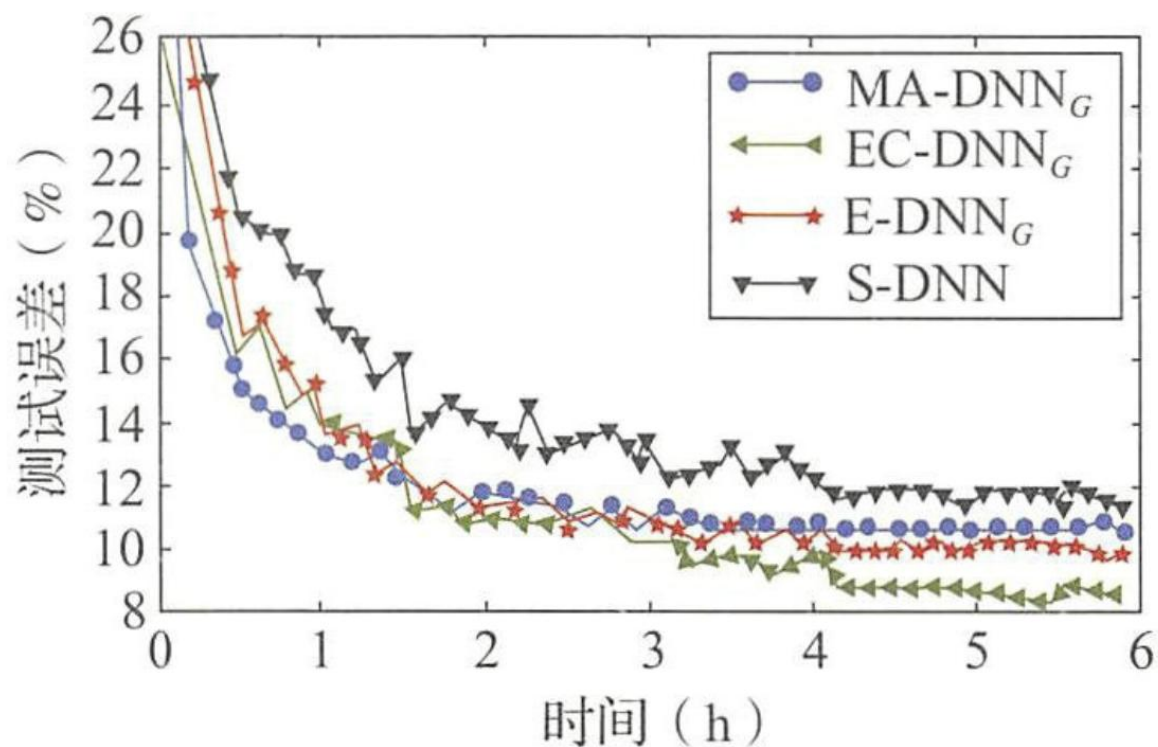
基于模型集成的聚合

一种集成-压缩算法流程：

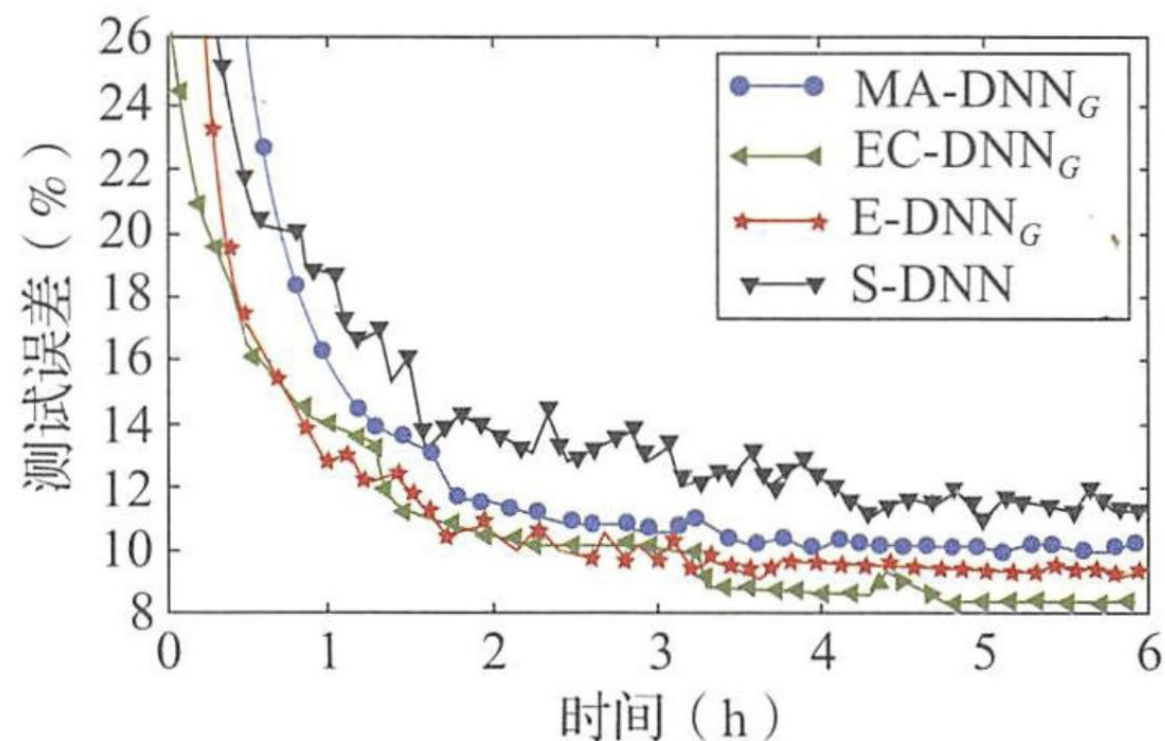
- (1) 各工作节点在本地进行模型更新后，将新模型发送到参数服务器
- (2) 参数服务器获取到 K 个机器上的局部模型 ω_t^k 后，**集成**： $\tilde{w}_t \leftarrow \{w_t^1, w_t^2, \dots, w_t^k\}$
- (3) 参数服务器**压缩**集成模型，使其仍然和局部模型一样大： $w_{t+1} = \text{compress}(\tilde{w}_t)$
- (4) 参数服务器将压缩后的模型 w_{t+1} 分发给各个工作节点

在CIFAR数据集上对比不同聚合方式

- S-DNN: 表示模型仅在一个 GPU 上进行训练
- E-DNN: 表示基于模型集成的聚合方法
- MA-DNN: 表示基于模型加和平均的聚合方法
- EC-DNN: 表示基于模型集成-压缩的聚合方法

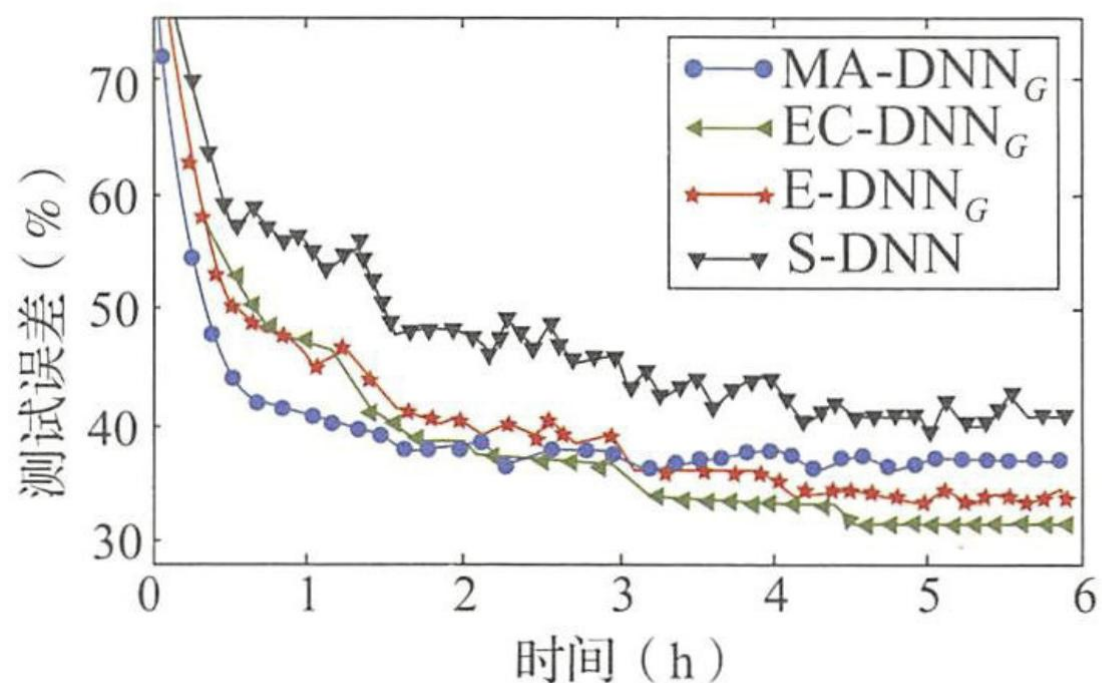


a) $K=4$, CIFAR-10

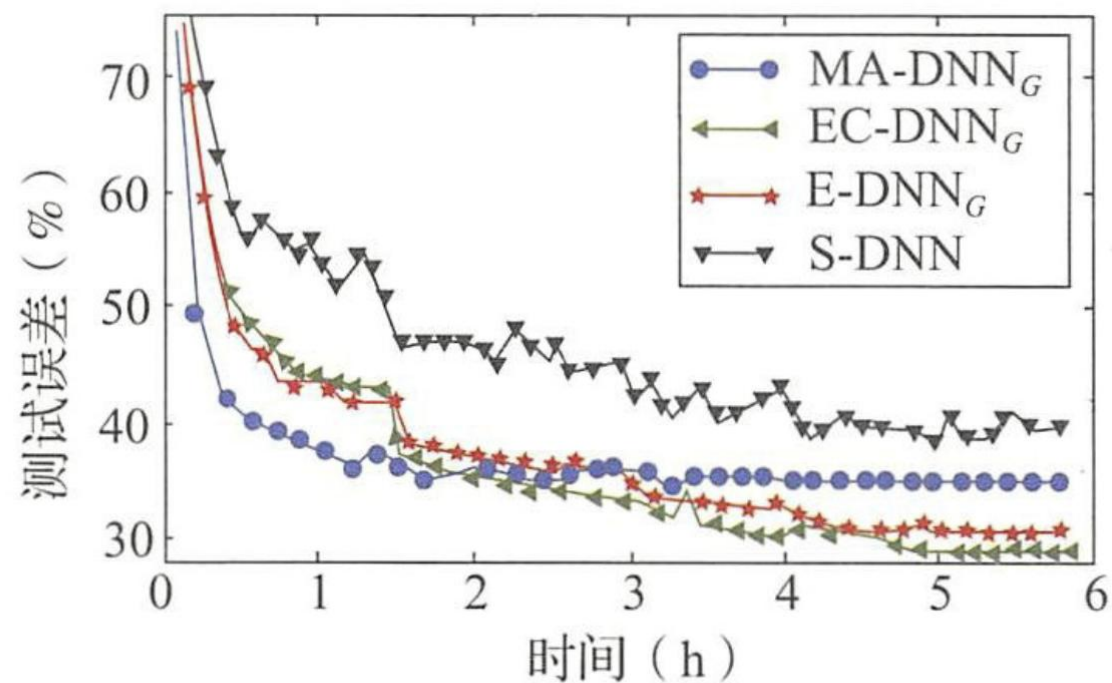


b) $K=8$, CIFAR-10

在CIFAR数据集上对比不同聚合方式



c) $K=4$, CIFAR-100



d) $K=8$, CIFAR-100

05

扩展：联邦学习

一种特殊的分布式学习

场景一

- 需求：

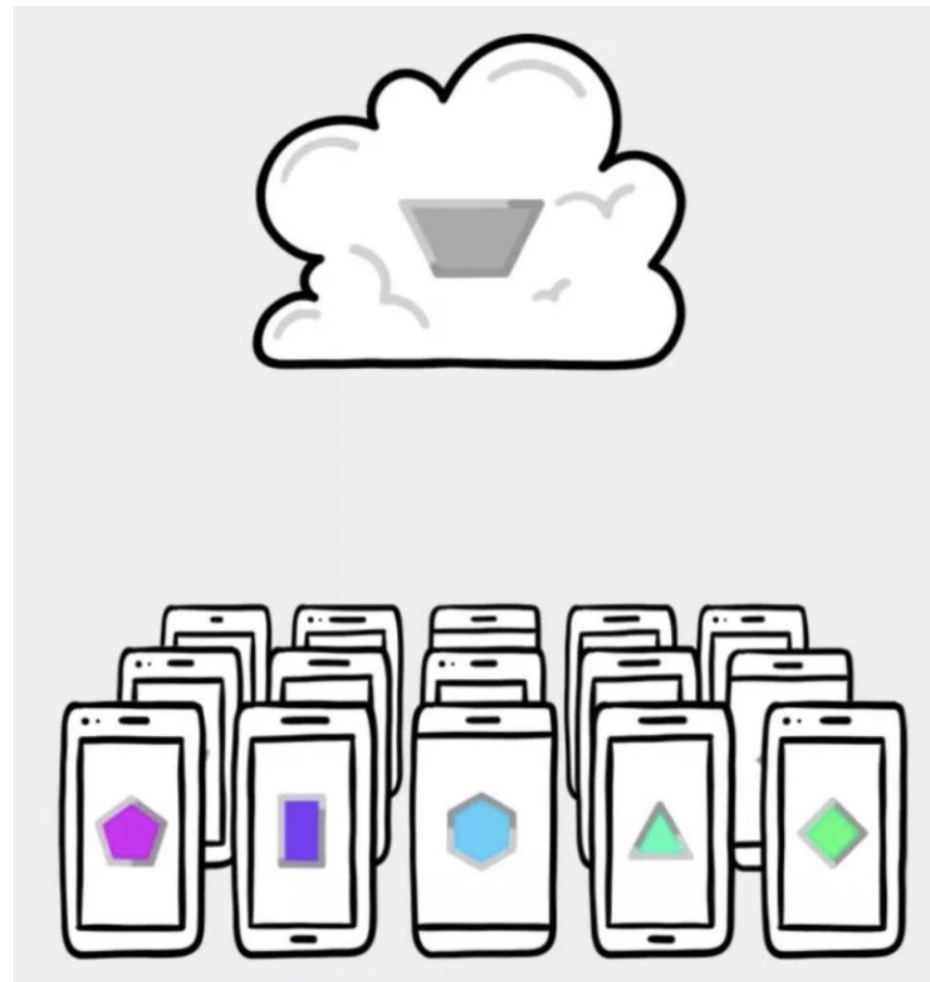
某公司想要使用各用户手机数据训练一个模型

- 传统方法：

1. 收集用户数据
2. 在云服务器上训练模型

- 问题：

用户不愿意上传数据（特别是敏感信息）给公司



场景二

- 需求：

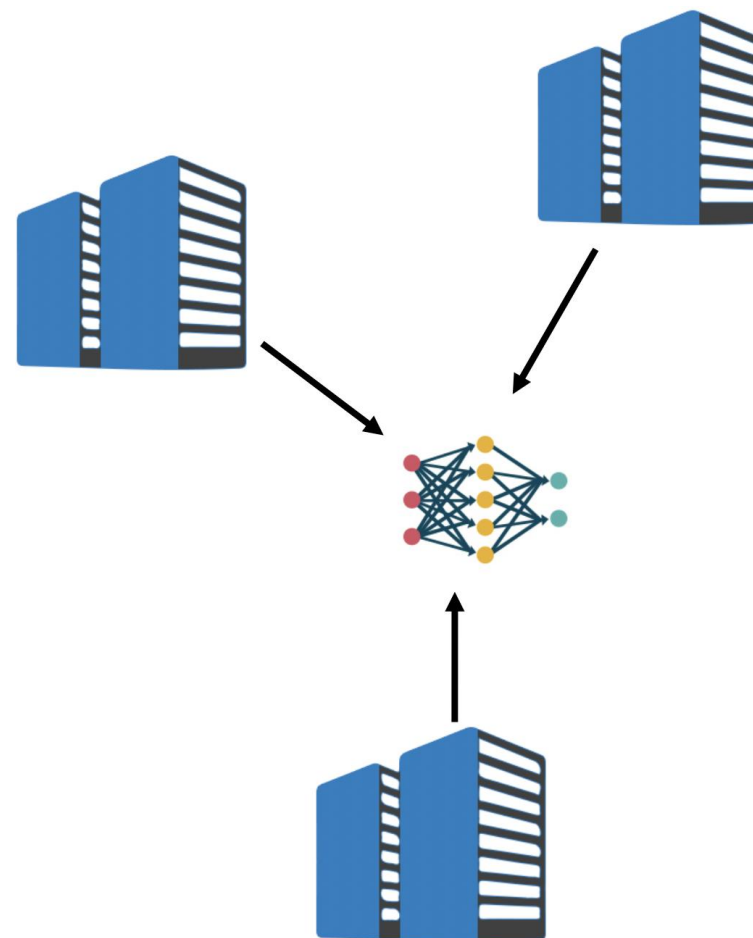
各医院希望使用病人医疗数据联合训练一个模型

- 传统方法：

1. 聚合各医院的病人医疗数据
2. 在云服务器上训练模型

- 问题：

法律法规不允许传播病人数据



面临的挑战

● 隐私保护

- 2017 年 6 月 1 日开始实施的《中华人民共和国网络安全法》中要求网络运营者对用户数据的收集必须公开和透明，对收集的用户信息应当严格保密。
- 2018 年 5 月，欧盟通过《通用数据保护条例》（General Data Protection Regulation, GDPR）法案，指出对数据的使用行为必须获取用户的明确授权。

面临的挑战

- 隐私保护



这些法律法规的建立在不同程度上对人工智能传统的数据处理模式提出了新的挑战。

面临的挑战

● 数据孤岛

各个机构都拥有各自的数据，这些数据互有关系却又独立保存在不同位置。

出于安全性、隐私性等方面考虑，这些小规模的数据之间存在着难以打破的壁垒，无法很好的互通和协作，导致潜在的数据价值没有得到充分的挖掘和应用。

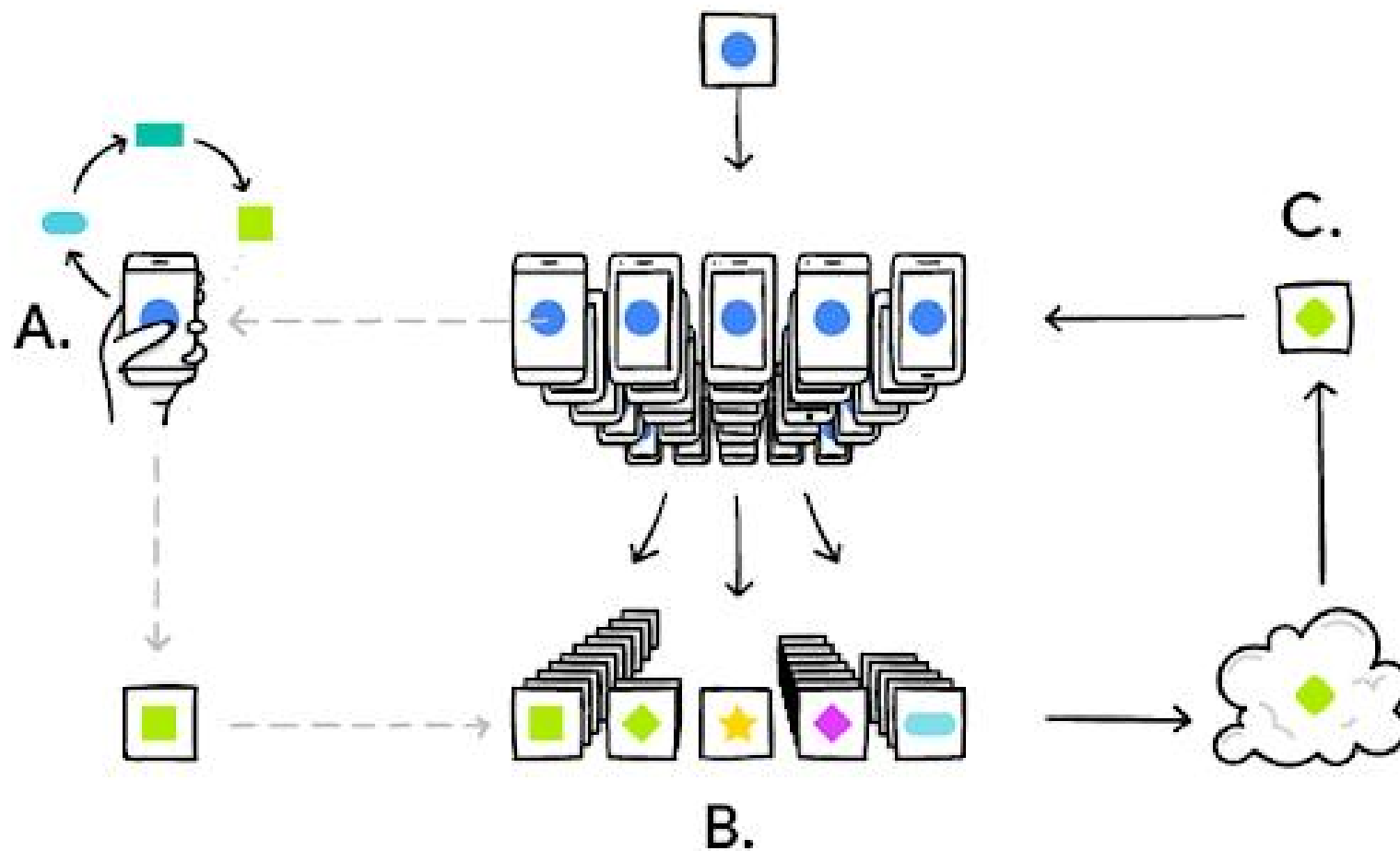
这就好像在信息技术这片大海之中，数据各自存储、各自定义，形成了海上的一座座孤岛。



联邦学习

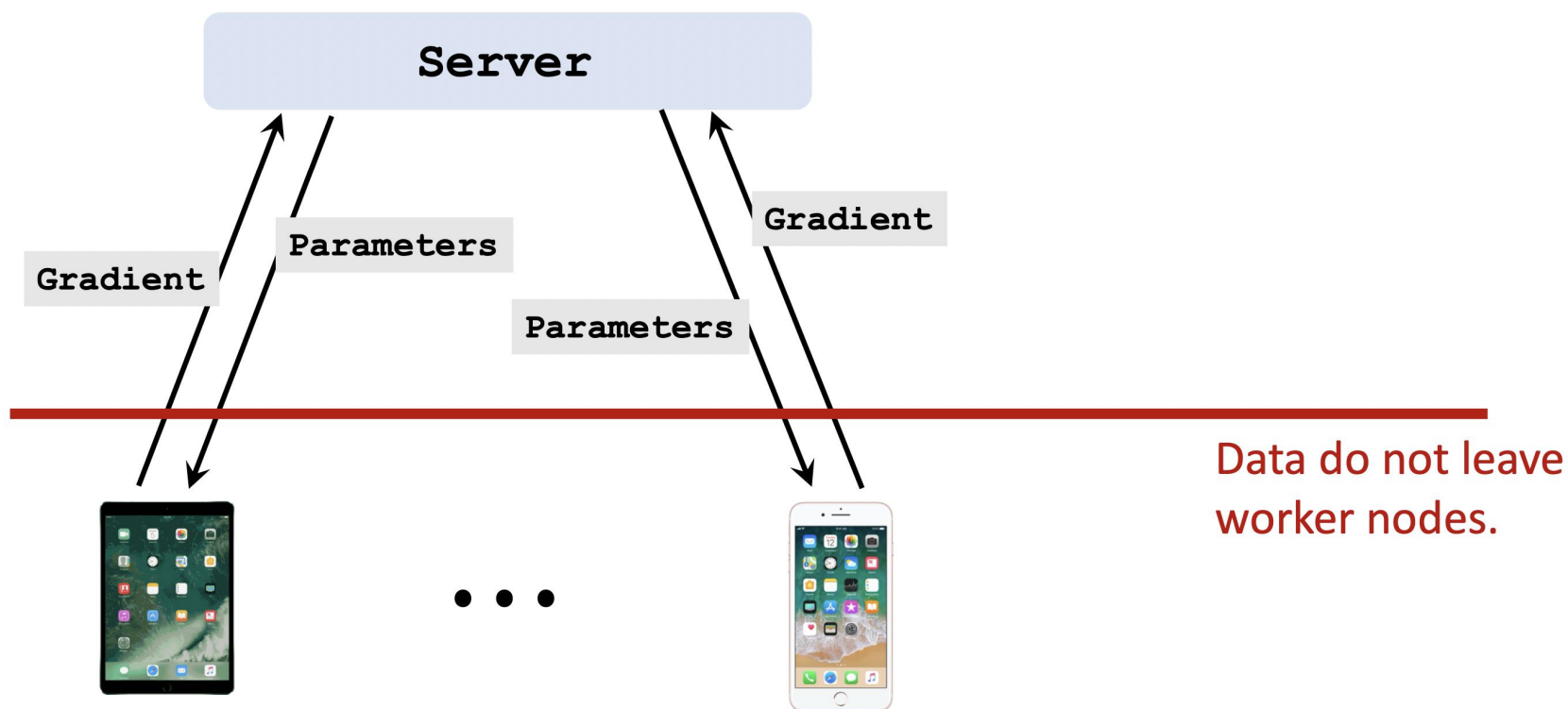
- 要面对上述这些挑战，仅仅靠传统的方法已经出现瓶颈。
- 如何在满足数据隐私、安全和监管要求的前提下，设计一个分布式机器学习框架，让人工智能系统能够更加高效、准确的共同使用各自的数据，解决数据孤岛问题？
- 于是人们提出了一个可行的解决方案，叫做**联邦学习**。

联邦学习



联邦学习的基本思想

- **基本思想**：在进行机器学习的过程中，各参与方可借助其他方数据进行联合建模。各方无需共享数据资源，即**数据不出本地**的情况下，进行数据分布式训练，建立共享的机器学习模型。



联邦学习



典型的联邦学习框架：一个中心服务器(Server)+多台设备(Clients)

Step1. 选择联机的Clients：Server从一组满足联机要求的Clients中抽样。例如，只抽取正接入无线网络且处于空闲状态的设备登陆到服务器。

Step2. 广播：每个被选中的Client从Server处下载当前的模型参数和一个训练程序。

Step3. 计算：每个Client利用本机上的数据，执行训练程序进行局部计算。例如利用本机数据和当前参数进行随机梯度下降，得到梯度估计量，并将计算得到的梯度或参数信息（而非原始数据）发送到Server。

Step4. 聚合更新：Server收集Clients的信息（例如梯度）进行汇总平均，并更新模型参数。

联邦学习

联邦学习可以看作一种特殊的分布式机器学习。

区别	传统分布式学习	联邦学习
工作状态	服务器对各客户端具有绝对控制权	用户控制自己的设备和数据
数据分布	数据可以任意打乱、平衡地分配给所有客户端	数据不互通，数量和分布都可能存在差异
节点状态	所有节点稳定运行	节点可能不在线，连接不可靠
主要瓶颈	计算代价	通信代价
考虑角度	提升计算效率	隐私保护

应用案例：基于联邦学习的目标检测应用

这里以微众银行的目标检测应用为例

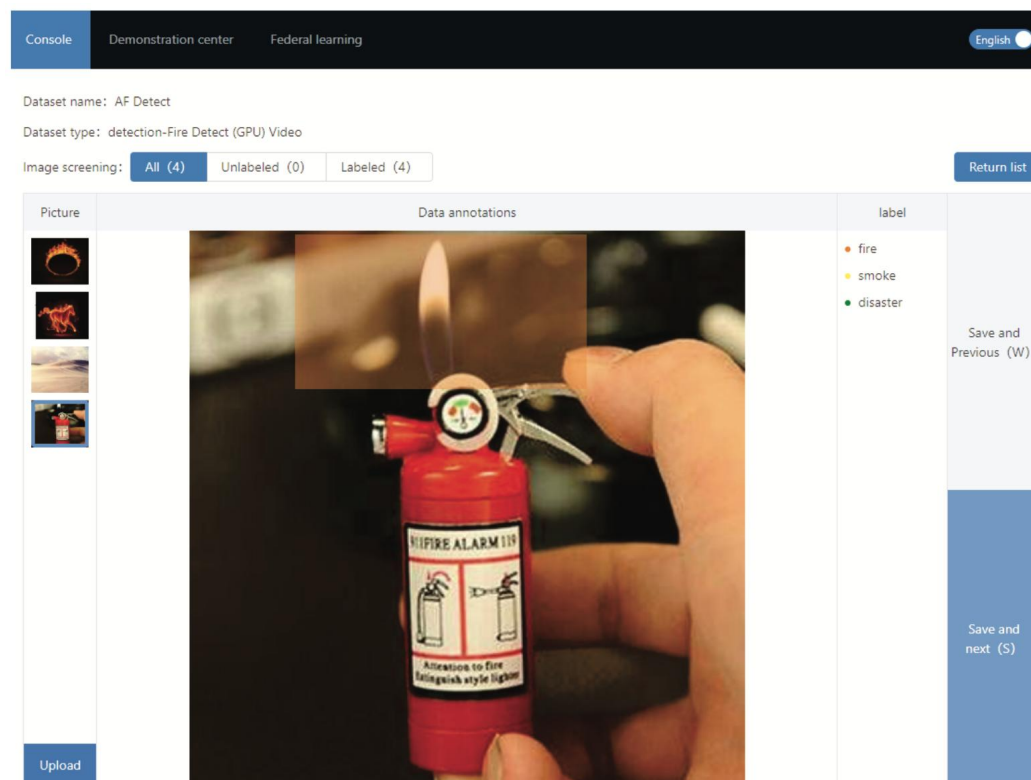
●传统方式：



应用案例：基于联邦学习的目标检测应用

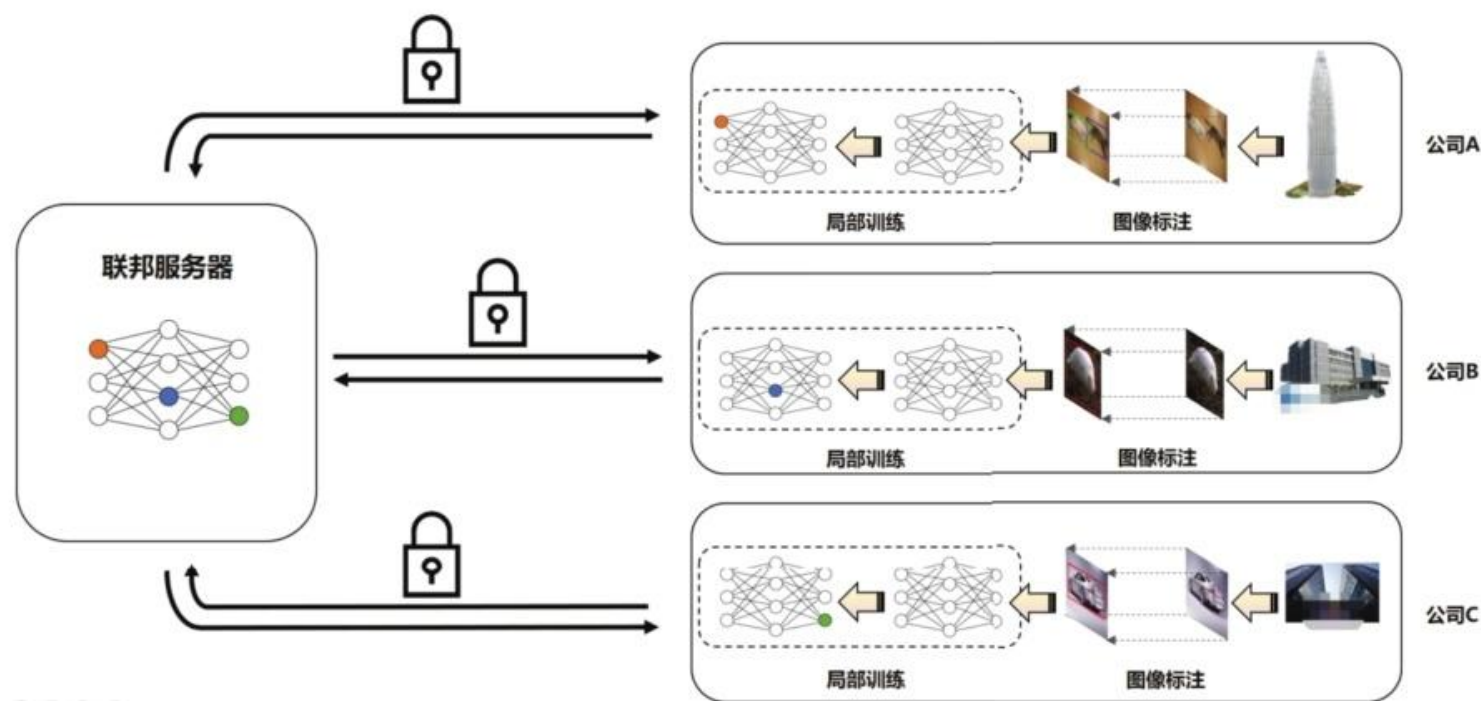
- 联邦学习方式：

在各个公司使用统一的图像标注工具进行标注，得到相同格式的标签数据



应用案例：基于联邦学习的目标检测应用

在各个客户端进行对本地图像数据集进行局部训练，然后上传加密参数，在服务器端进行聚合，完成模型的训练。



联邦学习面临的挑战

- 异构问题
- 通讯问题
- 隐私问题
- 安全问题

问题一：异构问题

- 系统（设备）异构

- 通信能力：节点间在通讯带宽和连接质量上存在差异

- 计算能力：节点间算力不同

难以协同，一些计算能力差、连接不稳定的设备会拖慢训练时间，影响整体算法的训练效果。



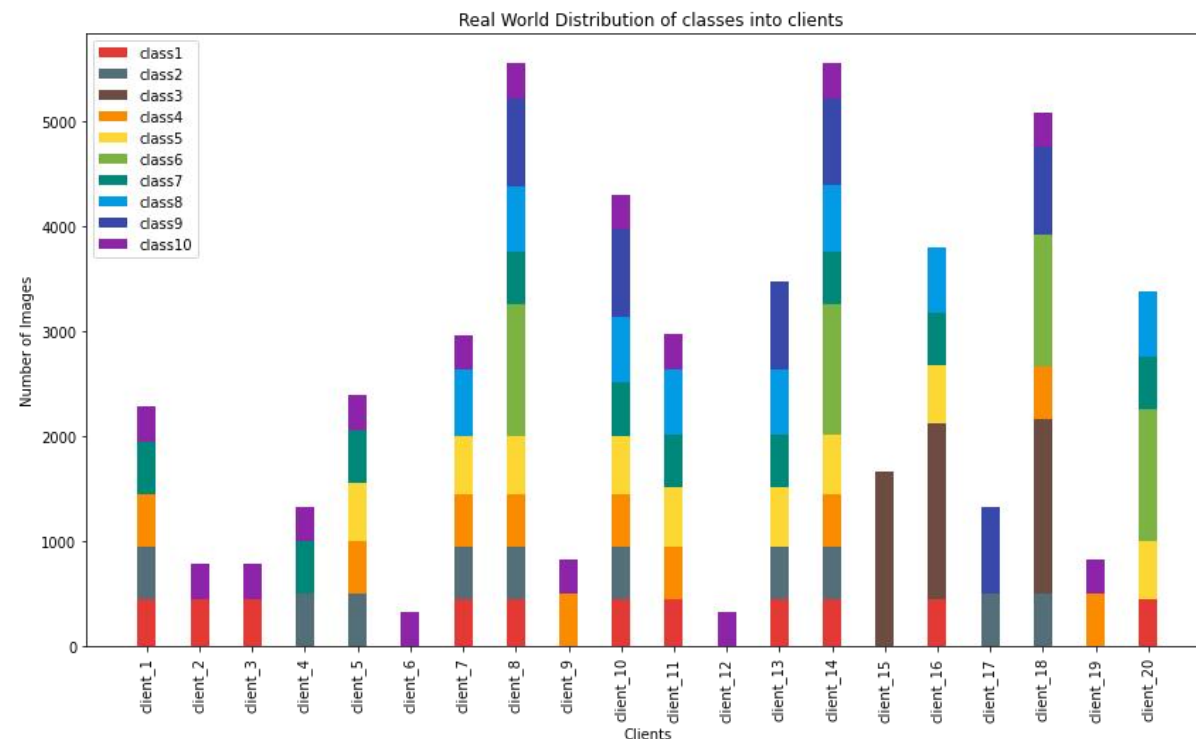
问题一：异构问题

●统计（数据）异构

➤ Non-IID：不同节点间的数据分布严重不均匀

➤ 数据失衡：不同节点间的数据量差距很大

全局模型难以训练，影响训练时间和训练精度。



问题二：通讯问题

- 采用联邦学习的协同训练方式严重依赖于数据传输，当大量设备同时与服务器连接时会存在网络缓慢、传输延迟甚至数据丢包等现象。

- 影响通信代价的几个因素：

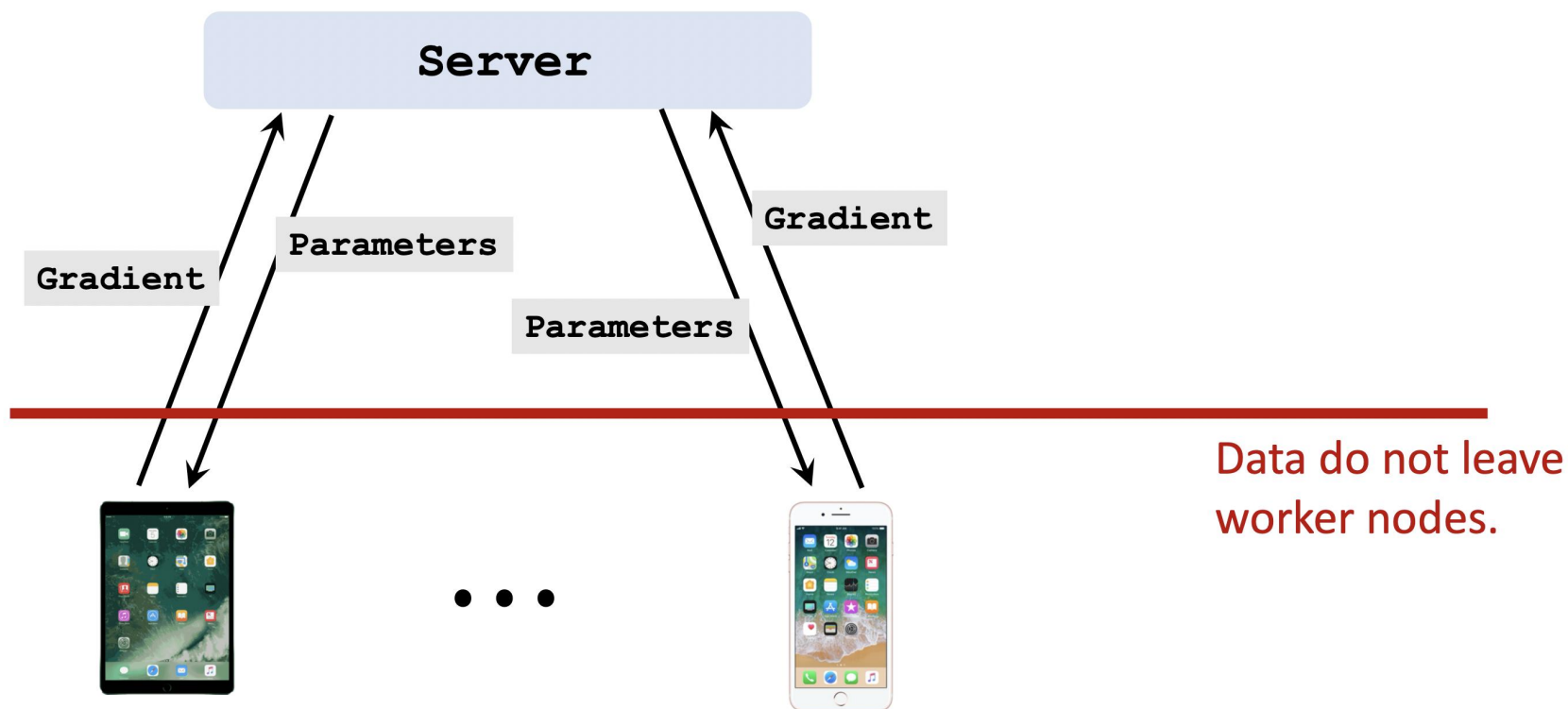
- 所需传递参数的数量
- 每次通信的客户端个数
- 网络延迟
- 带宽

- 减少通信代价的两种方式：

总通信数据量 = 通信次数 * 单次传输的数据量

- 提高全局模型的收敛速度（减少通信次数）
- 对参数或梯度进行数据压缩（减少单次传输代价）

问题三：隐私问题



基于梯度、参数等传输数据仍有可能反推原始数据

问题三：隐私问题

- 梯度携带原始数据相关信息

$$\min_{\mathbf{w}} \sum_{i=1}^n l(\mathbf{w}, \mathbf{x}_i, y_i), \quad \text{where } l(\mathbf{w}, \mathbf{x}_i, y_i) = \frac{1}{2} (\mathbf{x}_i^T \mathbf{w} - y_i)^2.$$

$$\mathbf{g}_i = \frac{\partial l(\mathbf{w}, \mathbf{x}_i, y_i)}{\partial \mathbf{w}} = (\mathbf{x}_i^T \mathbf{w} - y_i) \mathbf{x}_i.$$

问题三：隐私问题

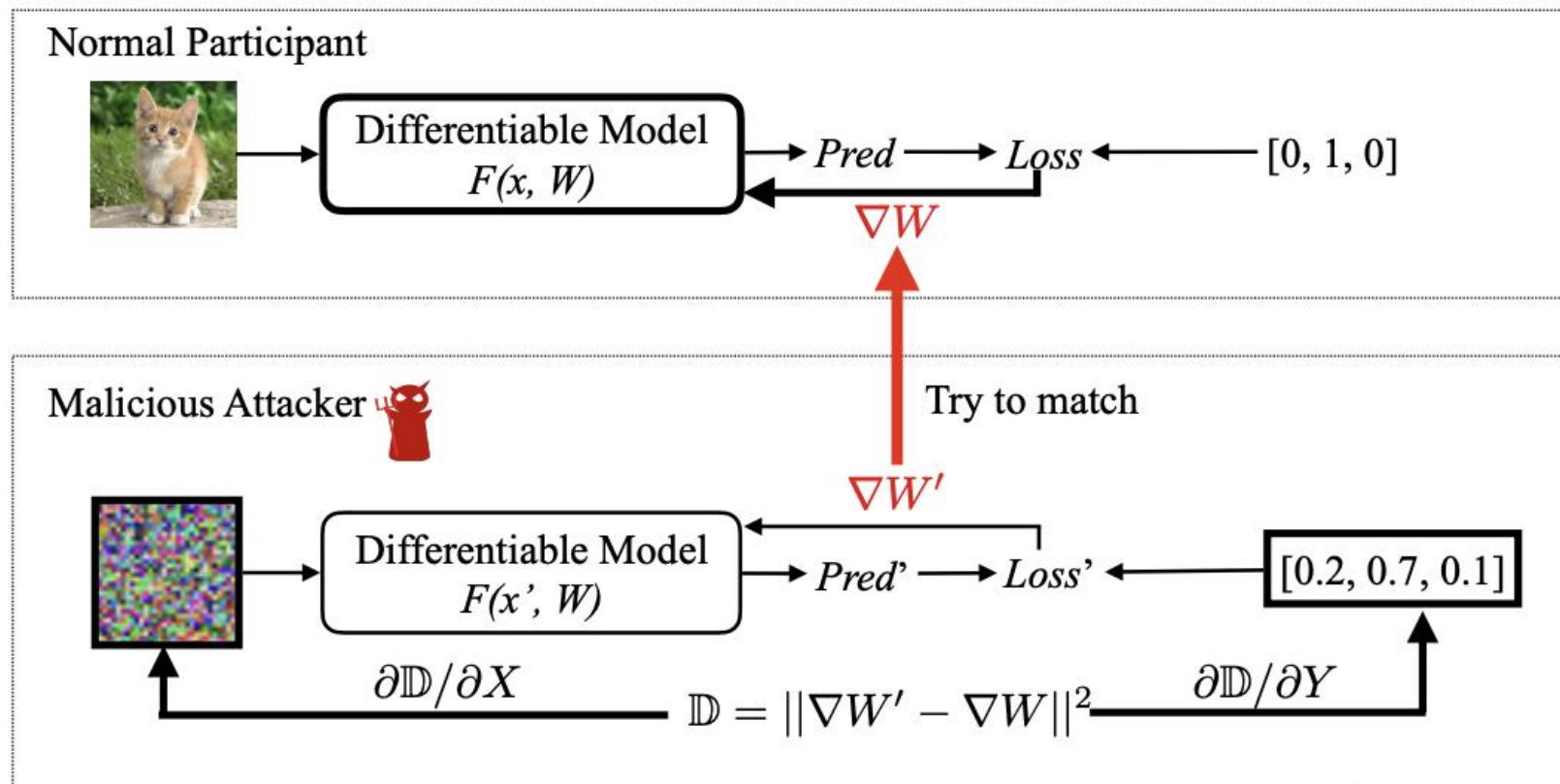
- 攻击方式1：



把梯度作为输入数据，用多个分类器分别来判别性别、人种、年龄等，从而反推出原始数据的某些信息。

问题三：隐私问题

● 攻击方式2：

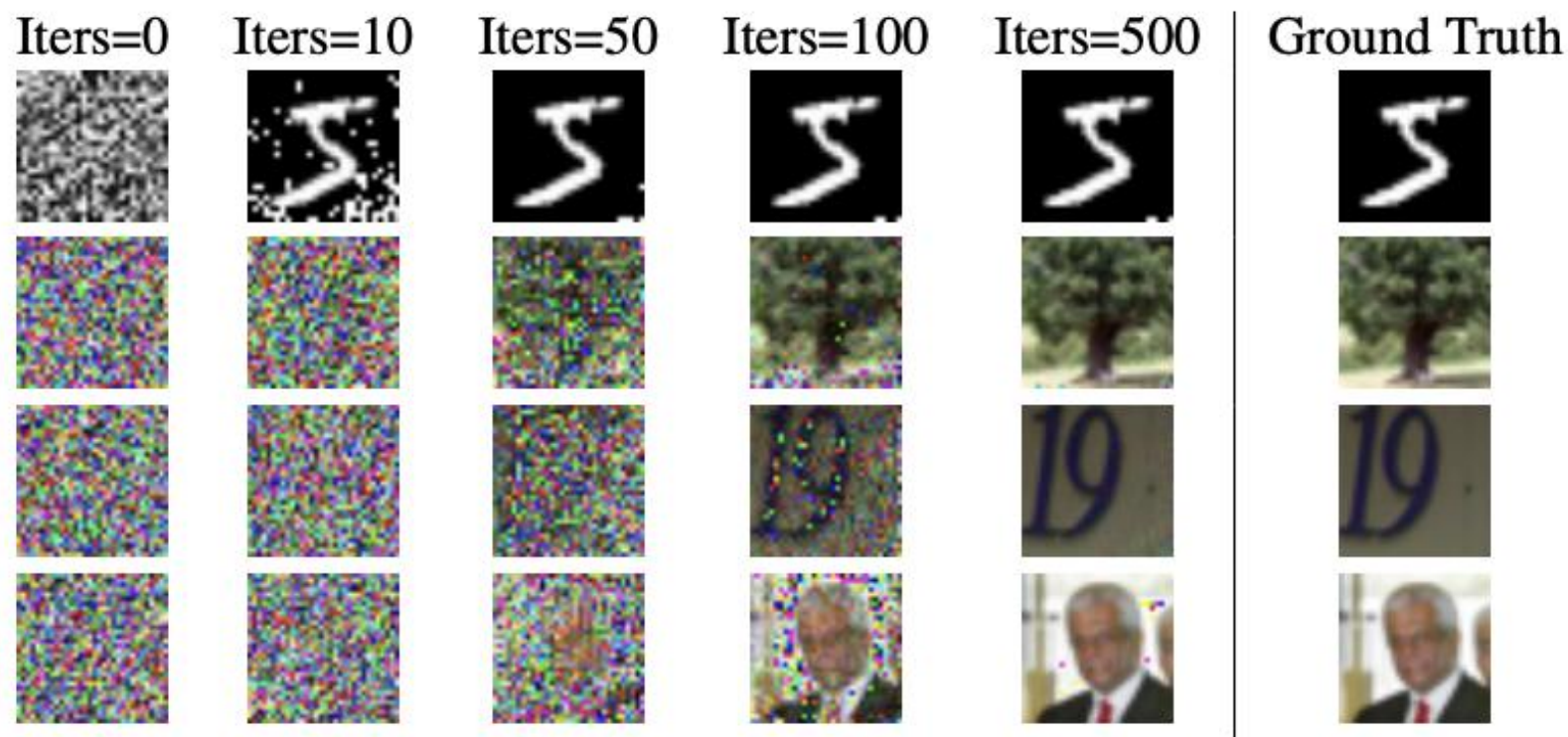


整个过程不需要训练数据集的任何额外信息，就可以完全恢复训练数据！

问题三：隐私问题

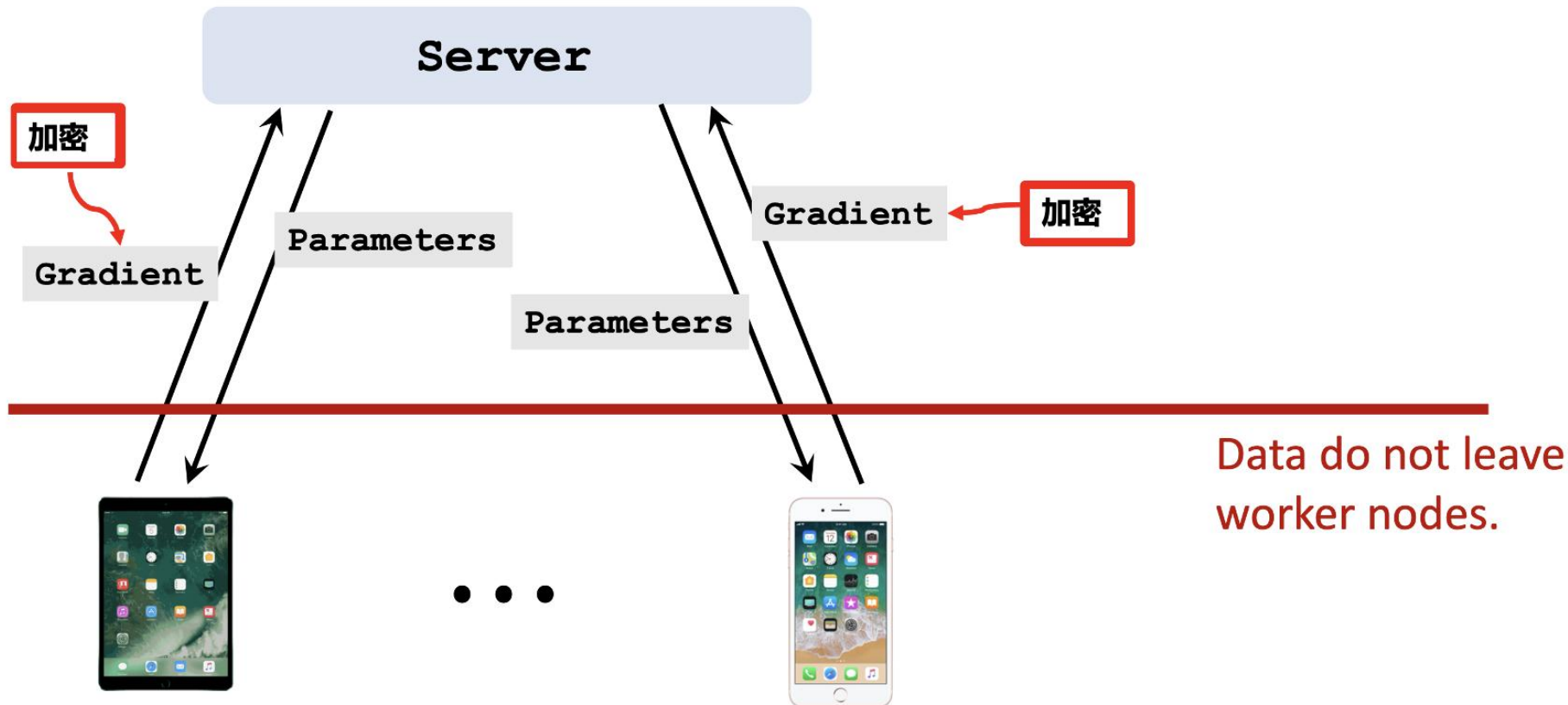
- 攻击方式2：

在常用的图片分类任务上测试该攻击方法的效果。



问题三：隐私问题

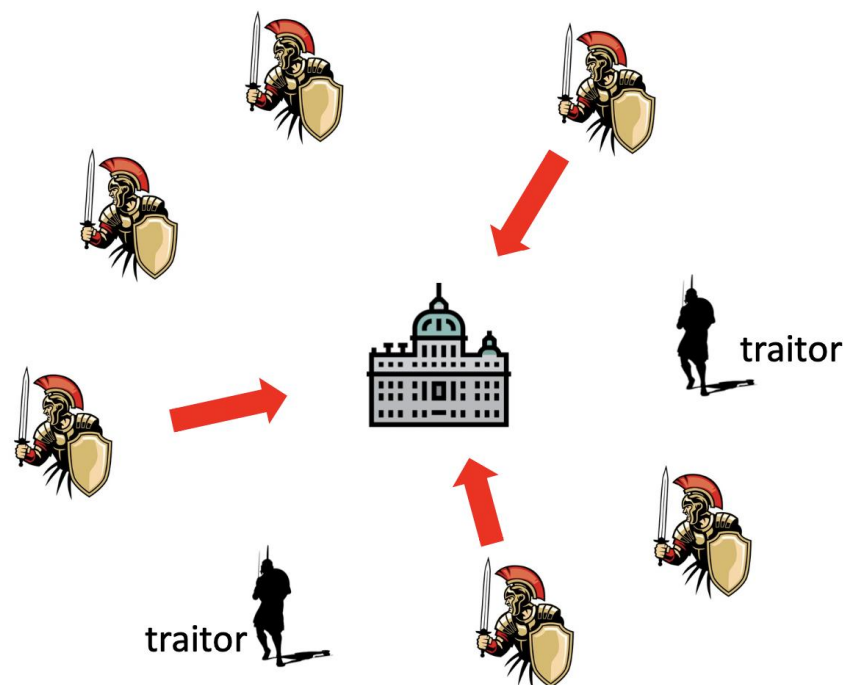
一种可行的解决方案：从同态加密、差分隐私及安全多方计算等方法出发，设计高效可靠的信息传输加密算法。



问题四：安全问题

- 参与方恶意攻击

传统分布式学习方法并没有假设输入模型的数据可能有误，甚至有设备会故意搅乱数据的分布



问题四：安全问题

●攻击手段

➤投毒攻击

主要是指在训练或再训练过程中，恶意参与者通过攻击训练数据集或模型，例如添加错误的标签或有偏差的数据来降低数据质量，或是发送错误的参数或损坏的模型来破坏全局模型的学习过程，来降低机器学习模型的训练质量。

➤对抗攻击

主要是指恶意构造输入样本，导致模型以高置信度输出错误结果。

问题四：安全问题

●防御手段

➤投毒攻击防御

针对数据投毒，防御方法包括利用身份验证机制在各参与方进行数据交互之前来保证参与节点的可靠性；利用上下文信息进行训练集中有毒样本点的检测。

针对模型投毒，防御的重点在于对恶意参与方的识别以及对错误更新参数的检测，包括模型准确度检测和参数更新统计差异检测等。

问题四：安全问题

●防御手段

➤对抗攻击防御

对抗训练方法：将真实的样本和对抗样本一起作为训练集，来训练出最后的模型。对抗训练适用于多种监督问题，它可以使得模型在训练过程中就学习到对抗样本的特征，提高了模型的健壮性。

梯度正则化方法：在训练模型的目标函数上对输入与输出的变化进行惩罚，从而限制了输入的扰动对于预测结果的影响。

Q&A

Questions and Answers