Communication-Efficient Learning of Deep Networks from Decentralized Data

H. Brendan McMahan Eider Moore Daniel Ramage Seth Hampson Blaise Agüera y Arcas

摘要

现代移动设备可以访问大量适合学习模型的数据，这反过来可以极大地改善设备上的用户体验。例如，语言模型可以改进语音识别和文本输入，图像模型可以自动选择好的照片。

然而，这些丰富的数据通常是隐私敏感的、数量大的或两者兼有，这可能会妨碍使用传统方法登录到数据中心并在那里进行培训。在这篇论文中提倡一种替代方案，即让训练数据分布在移动设备上，并通过聚合本地计算的更新来学习共享模型。文中将这种去中心化的方法称为联邦学习。

关键词：Federated Learning；Privacy；FedAvg

1 引言

现代移动设备可以访问大量适合学习模型的数据，这反过来可以极大地改善设备上的用户体验。例如，语言模型可以改进语音识别和文本输入，图像模型可以自动选择好的照片。

然而，这些丰富的数据通常是隐私敏感的、数量大的或两者兼有，这可能会妨碍使用传统方法登录到数据中心并在那里进行培训。在这篇论文中提倡一种替代方案，即让训练数据分布在移动设备上，并通过聚合本地计算的更新来学习共享模型。文中将这种去中心化的方法称为联邦学习。

联邦学习：联邦学习的理想问题具有以下特性:1) 对来自移动设备的真实数据进行训练，比对数据中心中普遍可用的代理数据进行训练具有明显的优势。2) 该数据属于隐私敏感数据或数据量较大 (相对于模型的大小)，因此最好不要只是为了模型训练的目的而将其记录到数据中心 (服务于集中收集原则)。3) 对于监督任务，数据上的标签可以从用户交互中自然地推断出来。

许多智能模型都符合上述标准。作为两个例子，我们考虑图像分类，例如预测哪些照片最有可能在未来被多次浏览或分享; 还有语言模型，它可以通过改进解码、下一个单词预测，甚至预测整个回复，来改进语音识别和触摸屏键盘上的文本输入。这两项任务的潜在训练数据 (用户拍摄的所有照片和他们在移动键盘上输入的所有内容，包括密码、URL、消息等) 可能是隐私敏感的。

隐私性：与数据中心对持久性数据的训练相比，联合学习有明显的隐私优势。即使是持有” 匿名” 数据集，也会通过与其他数据的连接而使用户的隐私受到威胁。相比之下，为联合学习而传输的信息是改进特定模型所需的最小更新（当然，隐私利益的强度取决于更新的内容）。他们永远不会比原始训练数据包含更多的信息（根据数据处理的质量），而且通常会包含少得多的信息。此外，聚合算法不需要更新的来源，所以更新可以通过混合网络或通过受信任的第三方传输，而不需要识别元数据。

2 相关工作

McDonald 等人对感知器和 Povey 等人对语音识别 dnn 研究了通过迭代平均局部训练模型进行的分布式训练。Zhang 等人研究了一种具有“软”平均的异步方法。这些工作只考虑集群/数据中心设置，

而不考虑不平衡和非 iid 的数据集，这些属性对联邦学习设置至关重要。我们将这种算法风格应用于联邦设置，并执行适当的经验评估，这与数据中心设置中的相关问题不同，需要不同的方法。

使用与我们类似的动机，Neverova 等人也讨论了在设备上保存敏感用户数据的好处。Shokri 和 Shmatikov 的工作在几个方面有关联: 他们专注于训练深度网络，强调隐私的重要性，并通过在每一轮通信中只共享参数的子集来解决通信成本; 但是，他们也没有考虑不平衡和非 iid 数据，实证评价是有限的。

在凸设置中，分布式优化和估计问题得到了极大的关注，一些算法确实专门关注通信效率。除了假设凸性之外，这种现有的工作通常要求客户机的数量远远小于每个客户机的示例数量，数据以 IID 方式分布在客户机之间，每个节点有相同的数据点数量——所有这些假设都在联邦优化设置中被违反。SGD 的异步分布式形式也被应用于训练神经网络，例如 Dean 等人，但这些方法在联邦设置中需要大量的更新。分布式共识算法放松了 IID 假设，但仍然不能很好地适用于在非常多的客户机上进行通信约束优化。

本文考虑的 (参数化) 算法族的一个端点是简单的一次性平均，其中每个客户端求解的模型最小化 (可能是正则化的) 其本地数据的损失，这些模型被平均生成最终的全局模型。这种方法在带有 IID 数据的凸情况下得到了广泛的研究，已知在最坏情况下，产生的全局模型并不比在单个客户端上训练模型更好。

3 本文方法

3.1 FedAvg 算法步骤

1. 在每一轮迭代的步骤 t，抽样子集中每个客户端上传更新后的本地参数 θi(t)
2. 服务端根据全局模型参数 θi(t) 计算出加权平均值 θt+1
3. 服务端发送当前全局模型参数 θt+1 给客户端
4. 非抽样子集中的客户端根据 θt+1，通过 SGD 更新本地模型

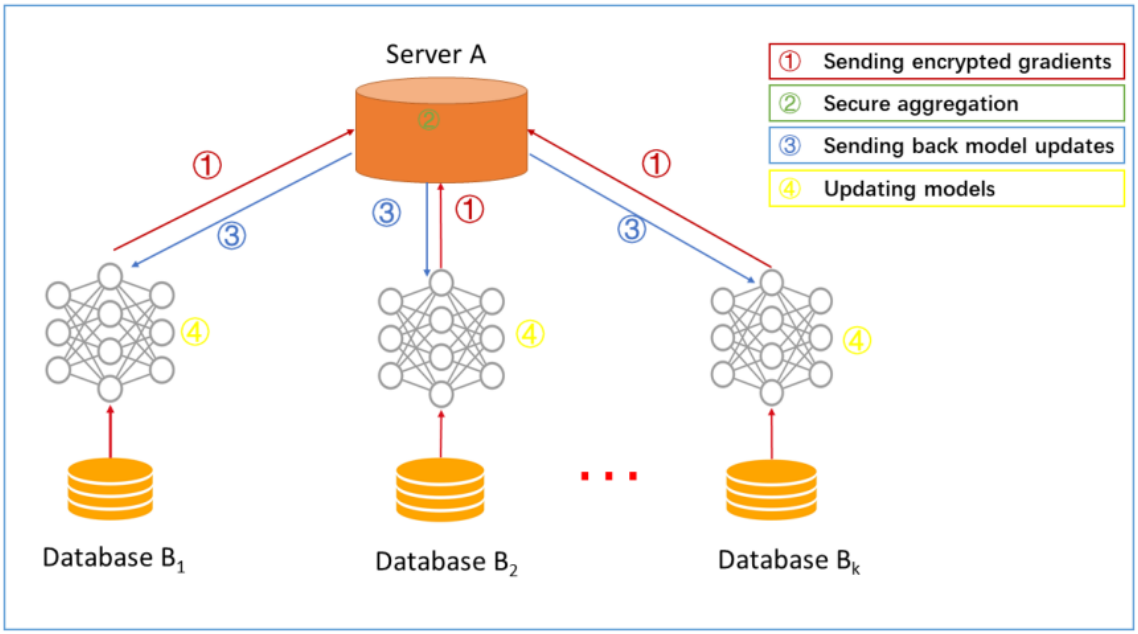

图 1: FedAvg 算法步骤示意图

3.2 损失函数定义

在 FedAvg 算法中，全局损失函数可被定义为

$$\min_{\omega \in \mathbb{R}^d} f(\omega) \quad where \quad f(\omega) \stackrel{def}{=} \frac{1}{n} \sum_{i=1}^{n} f_i(\omega).$$

对于机器学习问题，可以形式化定义 $f_i(\omega) = L(x_i, y_i; \omega)$, 其中 (x_i, y_i) 是输入向量和对应标签，ω 是模型参数。假设训练中有 K 个参与者，P_k 是参与的数据集，$n_k = |P_k|$。则全局损失函数可以重写为

$$f(\omega) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(\omega) \quad where \quad F_k(\omega) = \frac{1}{n_k} \sum_{i \in P_k} f_i(\omega).$$

4 复现细节

4.1 与已有开源代码对比

Procedure 1 FederatedAveraging Algorithm

Input: number of client *K,* local minibatch size *B,* number of local epochs *E,* learning rate η
Server executes:
initialize ω_0
for *each round t = 1, 2, ...* **do**
    $m \leftarrow max(C \cdot K, 1)$
    $S_t \leftarrow (random\ set\ of\ m\ clients)$
    for $each client k \in S_t$ **do**
        $\omega_{t+1}^k \in ClientUpdate(k, \omega_t)$
    end
    $\omega_{t+1}^k \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} \omega_{t+1}^k$
end
ClientUpdate(k, ω**):** // Run on client k
$\mathcal{B} \leftarrow (split$ P_k into batches of size B)
for *each local epoch i from 1 to E* **do**
    for $batch b \in \mathcal{B}$ **do**
        $\omega \leftarrow \omega - \eta \nabla \mathcal{L}(\omega; b))$
    end
    return ω to server
end

对于模型设置和训练代码参考自开源代码

```python
class MLP(nn.Module):
    def __init__(self, dim_in, dim_hidden, dim_out):
        super(MLP, self).__init__()
        self.layer_input = nn.Linear(dim_in, dim_hidden)
        self.relu = nn.ReLU()
        self.dropout = nn.Dropout()
        self.layer_hidden = nn.Linear(dim_hidden, dim_out)

    def forward(self, x):
        x = x.view(-1, x.shape[1]*x.shape[-2]*x.shape[-1])
        x = self.layer_input(x)
        x = self.dropout(x)
        x = self.relu(x)
        x = self.layer_hidden(x)
        return x
```

图 2: 模型设置示意图

```python
def train(self, net):
    net.train()
    # train and update
    optimizer = torch.optim.SGD(net.parameters(), lr=self.args.lr, momentum=self.args.momentum)

    epoch_loss = []
    for iter in range(self.args.local_ep):
        batch_loss = []
        for batch_idx, (images, labels) in enumerate(self.ldr_train):
            images, labels = images.to(self.args.device), labels.to(self.args.device)
            net.zero_grad()
            log_probs = net(images)
            loss = self.loss_func(log_probs, labels)
            loss.backward()
            optimizer.step()
            if self.args.verbose and batch_idx % 10 == 0:
                print('Update Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
                    iter, batch_idx * len(images), len(self.ldr_train.dataset),
                           100. * batch_idx / len(self.ldr_train), loss.item()))
            batch_loss.append(loss.item())
        epoch_loss.append(sum(batch_loss)/len(batch_loss))
    return net.state_dict(), sum(epoch_loss) / len(epoch_loss)
```

图 3: 训练过程示意图

4.2 数据集说明

MNIST 数据集来自美国国家标准与技术研究所, National Institute of Standards and Technology (NIST)。训练集 (training set) 由来自 250 个不同人手写的数字构成, 其中 50% 是高中学生, 50% 来自人口普查局 (the Census Bureau) 的工作人员。测试集 (test set) 也是同样比例的手写数字数据。

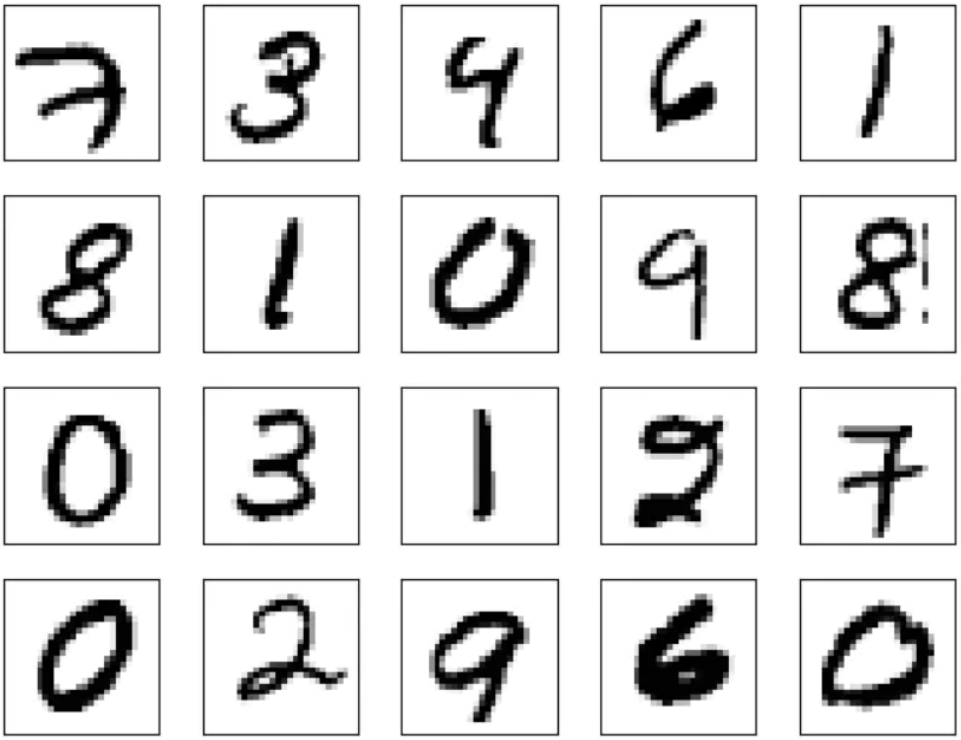

图 4: MNIST 手写数据集示意图

4.3 实验环境搭建

实验设置：参与的客户端总数 K=100，训练轮次 epoch=20，批次 batch=10，学习率 $\eta = 0.01$ 。

实验环境：系统 Windows 10 专业版，CPU Intel(R) Core(TM) i5-8250U，GPU NVIDIA GeForce MX130，RAM 32GB，Python=3.8，pytorch=1.13.0，CUDA=11.0.208

5 实验结果分析

实验实现以联邦学习 FedAvg 架构实现 100 个客户端联合训练识别 MNIST 数据集的 MLP 模型，并记录实验结果。训练过程中，loss 变化趋势如图 5 所示，联邦学习中多个客户端的联合模型精度随训练 epoch，loss 逐渐下降收敛。最终 epoch=20 训练结束时，模型的训练正确率为 90.50%，测试准确率为 90.73%

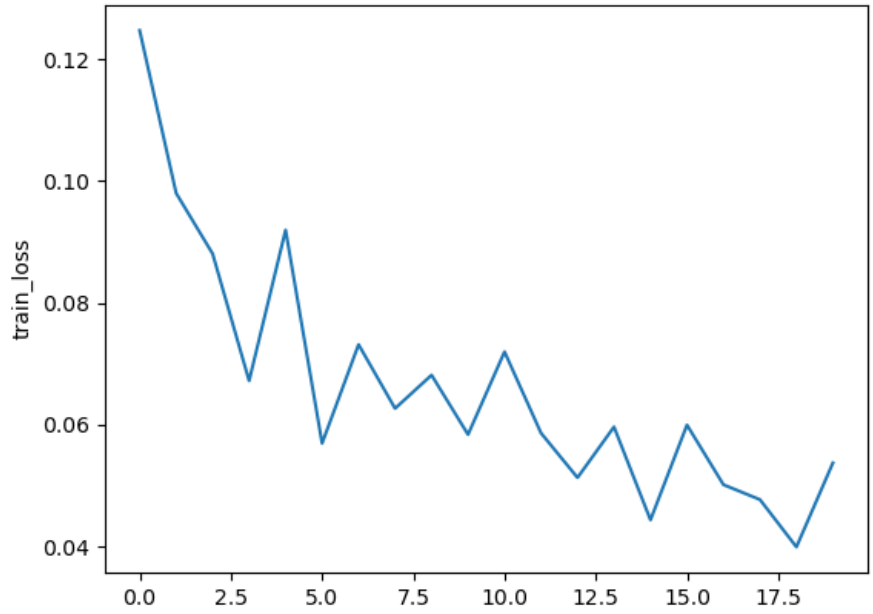

图 5: 训练损失示意图

6 总结与展望

在本实验报告中，介绍了如今数据驱动模型面临的隐私性担忧和联邦学习的基本理念和基本方法。在论文的实验部分，复现了“Communication-Efficient Learning of Deep Networks from Decentralized Data”一文中的 FedAvg 方法，通过 FedAvg 模型训练多个客户端联合学习，实现对 MNIST 数据集的识别。

自 FedAvg 后，围绕联邦学习提出了许多考虑通信效率、模型性能、非 IID 问题、和安全隐私的改进方法，后续会继续进行对这一方向的学习和实践。