SCAFFOLD: Stochastic Controlled Averaging for Federated Learning

Sai Praneeth Karimireddy Satyen Kale Mehryar Mohri Sashank J. Reddi Sebastian U. Stich Ananda Theertha Suresh

摘要

联合平均算法（FedAvg）由于其简单性和低通信成本而成为联邦学习的首选算法。然而，尽管最近进行了相关研究，但其性能尚未完全了解。我们获得了 FedAvg 的紧收敛速度，并证明了当数据是异构的（non-iid）时，它会受到“客户端漂移”的影响，从而导致不稳定和缓慢的收敛。

作为一种解决方案，本文提出了一种新的算法（SCAFFOLD），它使用控制变量（方差减少）来校正其本地更新中的“客户端漂移”。本文证明，SCAFFOLD 需要的通信次数明显减少，并且不受数据异构性或客户端采样的影响。此外，本文还表明 SCAFFOLD 可以利用客户数据中的相似性，从而实现更快的收敛。

后者是第一个量化分布式优化中局部步骤有用性的结果。

1 引言

移动通信设备中有许多有用的数据，训练模型后可以提高用户体验。但是，这些数据通常敏感或很庞大，不能直接上传到数据中心，使用传统的方法训练模型。据此提出联邦学习，将训练数据分布在移动设备上，通过聚合本地计算的更新来学习共享模型。

当客户端数据是非独立同分布时，FedAvg 的收敛速度会受到所谓 client-drift 的影响。作为一种解决方案，本文作者提出了 SCAFFOLD，该算法使用控制变量来纠正其局部更新中的 client-drift。联邦优化存在以下关键挑战：

1. 服务器与客户端间网络连接相对较慢。
2. 在给定时间内只有一小部分客户端能够用于训练。
3. 不同客户端间数据分布的异质性。

FedAvg 虽然可以缓解通信压力，但它在异质数据上的表现不太好，如何改正 FedAvg 的这种缺陷也是联邦学习目前比较热门的一个研究方向。客户端间数据的异质性会在客户端的更新中引入一个 client-drift，这会导致收敛变缓。

作为一种解决方案，SCAFFOLD 试图纠正这种 client-drift。

2 相关工作

• 证明 FedAvg 的收敛速度比之前已知的具有客户端采样和异构数据的凸函数和非凸函数更高。

• 给出了匹配的下限，以证明即使没有客户端采样和全批次梯度，由于客户端漂移，FedAvg 也可能比 SGD 慢。

• 提出了一种新的随机控制平均算法（SCAFFOLD），用于纠正这种客户端漂移。证明了 SCAFFOLD 至少与 SGD 一样快，并且对于任意异构数据都收敛。

• 证明 SCAFFOLD 还可以利用客户之间的相似性，进一步减少所需的通信，首次证明了采取本地步骤优于大批量 SGD 的优势。

• 证明 SCAFFOLD 相对不受客户抽样获得方差降低率的影响，因此特别适合联合学习[1]。

3 本文方法

3.1 FedAvg 算法

FedAvg 更新方法：客户端抽样集合为 S，对每个被抽样的客户端，其本地模型 $y_i = x$ 将执行 K 次本地更新：

$$y_i \leftarrow y_i - \eta_l g_i(y_i) \tag{1}$$

η_l 为学习率。接着，客户端的更新 $y_i - x$（模型增量）将在服务器端进行聚合：

$$x \leftarrow x + \frac{\eta_g}{|S|} \sum_{i \in S} y_i - x \tag{2}$$

即对增量进行聚合。当然也可以直接对更新后的模型进行聚合：

$$x \leftarrow \frac{\eta_g}{|S|} \sum_{i \in S} y_i \tag{3}$$

3.2 SCAFFOLD 算法

与 FedAvg 不同的是，SCAFFOLD 为每个客户端（客户端控制变量 c_i）和服务器（服务器控制变量 c）设置了控制变量. 两种控制变量间的关系：

$$c = \frac{1}{N} \sum c_i \tag{4}$$

即服务器的控制变量为所有客户端控制变量的平均值，所有控制变量都需要进行初始化，最简单的情况是都初始化为 0。每一轮通信中，服务器端的参数 (x, c)（模型 + 控制变量）都被发送到被选中的客户端 S。每一个被选中的客户端都将其本地模型初始化为 $y_i - x$，然后进行本地更新：

$$y_i \leftarrow y_i - \eta_l (g_i(y_i) + c - c_i) \tag{5}$$

等到 K 次本地更新完毕后，局部控制变量 c_i 也需要进行更新：

$$c_i^+ \leftarrow \begin{cases} Option1. & g_i(x), or \\ Option2. & c_i - c + \frac{1}{K_{\eta_l}}(x - y_i) \end{cases} \tag{6}$$

等到 K 次本地更新完毕后，局部控制变量 c_i 也需要进行更新：

$$x \leftarrow x + \frac{\eta_g}{|S|} \sum_{i \in S} y_i - x \tag{7}$$

$$c \leftarrow c + \frac{1}{N} \sum_{i \in S} (c_i^+ - c_i) \tag{8}$$

也可以直接对更新后的模型进行聚合：

$$x \leftarrow \frac{\eta_g}{|S|} \sum_{i \in S} y_i \tag{9}$$

$$c \leftarrow \frac{1}{N} \sum_{i \in S} c_i^+ \tag{10}$$

观察本地模型更新公式：

$$y_i \leftarrow y_i - \eta_l (g_i(y_i) + c - c_i) \tag{11}$$

如果局部控制变量 c_i 总是为 0，那么更新公式将变为：

$$y_i \leftarrow y_i - \eta_l g_i(y_i) \tag{12}$$

也就是说，SCAFFOLD 将退化为 FedAvg。

3.3 有效性分析

可以发现，SCAFFOLD 只是在 FedAvg 的基础上增加了一个修正项 $c - c_i$, 就可以有效缓解本地客户端的 client-drift. 如果通信成本不是问题，最理想的客户端更新机制应该为：

$$y_i = y_i + \frac{1}{N} \sum_j g_j(y_i) \tag{13}$$

该更新本质上是计算损失函数 f 的无偏梯度，相当于在 IID 情况下运行 FedAvg，但是这样的更新需要在每个更新步骤中与所有客户端进行通信。

与之对比，SCAFFOLD 使用了控制变量（选项 I）：

$$c_j \approx g_j(y_i) \quad and \quad c \approx \frac{1}{N} \sum_j g_j(y_i) \tag{14}$$

因为 SCAFFOLD 的本地更新方式为：

$$y_i \leftarrow y_i - \eta_l (g_i(y_i) + c - c_i) \tag{15}$$

又有：

$$(g_i(y_i) + c - c_i) \approx \sum_j g_j(y_i) \tag{16}$$

所以 SCAFFOLD 通过控制变量来近似模拟了理想状态下的更新。

因此，对于任意异质的客户端，SCAFFOLD 的本地更新保持同步和收敛。

观察控制变量的更新：

$$c_i^+ \leftarrow \begin{cases} Option1. & g_i(x), or \\ Option2. & c_i - c + \dfrac{1}{K_{\eta_l}}(x - y_i) \end{cases} \tag{17}$$

可以发现，控制变量中含有该客户端模型的更新方向（梯度）信息。

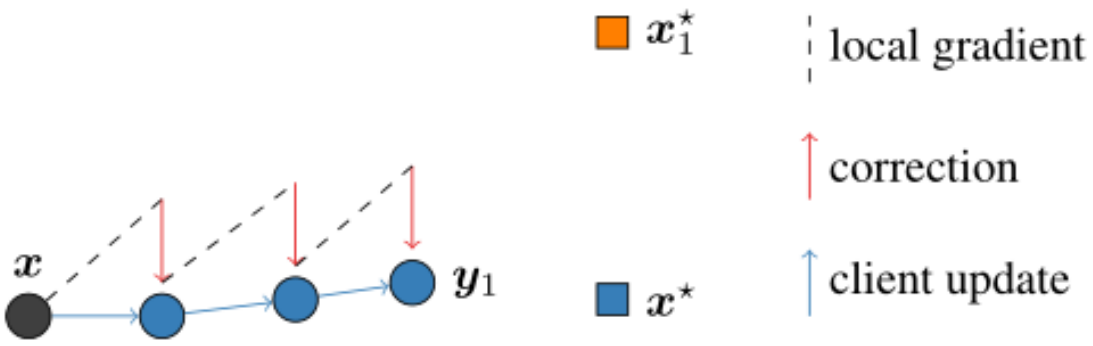
全局控制变量 c 是所有客户端本地控制变量的均值，即全局控制变量 c 中含有其他所有客户端的模型更新方向信息。

观察本地更新方式：

$$y_i \leftarrow y_i - \eta_l (g_i(y_i) + c - c_i) \tag{18}$$

$c - c_i$ 可以理解为全局模型相对于本地模型的 client-drift 值，也就是说我们在对本地模型进行更新时考虑了这种差异，这将有效克服 client-drift。

可视化解释如下：



对于单个客户端，接收服务端的模型 x 然后进行更新，如果采用 FedAvg 的更新机制，那么最优解将向着 x_1^* 移动，但加入一个修正项 $c - c_i$ 后，将对模型的更新方向产生一个修正，使其朝着真正的最优解 x^* 移动。

3.4 收敛性分析

4 复现细节

4.1 与已有开源代码对比

没有参考任何相关源代码。

SCAFFOLD 算法的伪代码如下：

Algorithm 1 SCAFFOLD: Stochastic Controlled Averaging for federated learning

server input: initial $\boldsymbol{x}$ and $\boldsymbol{c}$, and global step-size η_g
client i's input: $\boldsymbol{c}_i$, and local step-size η_l
for each round $r = 1, \ldots, R$ **do**
  sample clients $\mathcal{S} \subseteq \{1, \ldots, N\}$
  communicate $(\boldsymbol{x}, \boldsymbol{c})$ to all clients $i \in \mathcal{S}$
  on client $i \in \mathcal{S}$ **in parallel do**
    initialize local model $\boldsymbol{y}_i \leftarrow \boldsymbol{x}$
    for $k = 1, \ldots, K$ **do**
      compute mini-batch gradient $g_i(\boldsymbol{y}_i)$
      $\boldsymbol{y}_i \leftarrow \boldsymbol{y}_i - \eta_l \left(g_i(\boldsymbol{y}_i) - \boldsymbol{c}_i + \boldsymbol{c}\right)$
    end for
    $\boldsymbol{c}_i^+ \leftarrow$ (i) $g_i(\boldsymbol{x})$, or (ii) $\boldsymbol{c}_i - \boldsymbol{c} + \frac{1}{K\eta_l}(\boldsymbol{x} - \boldsymbol{y}_i)$
    communicate $(\Delta \boldsymbol{y}_i, \Delta \boldsymbol{c}_i) \leftarrow (\boldsymbol{y}_i - \boldsymbol{x}, \boldsymbol{c}_i^+ - \boldsymbol{c}_i)$
    $\boldsymbol{c}_i \leftarrow \boldsymbol{c}_i^+$
  end on client
  $(\Delta \boldsymbol{x}, \Delta \boldsymbol{c}) \leftarrow \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} (\Delta \boldsymbol{y}_i, \Delta \boldsymbol{c}_i)$
  $\boldsymbol{x} \leftarrow \boldsymbol{x} + \eta_g \Delta \boldsymbol{x}$ and $\boldsymbol{c} \leftarrow \boldsymbol{c} + \frac{|\mathcal{S}|}{N} \Delta \boldsymbol{c}$
end for

4.2 实验环境搭建

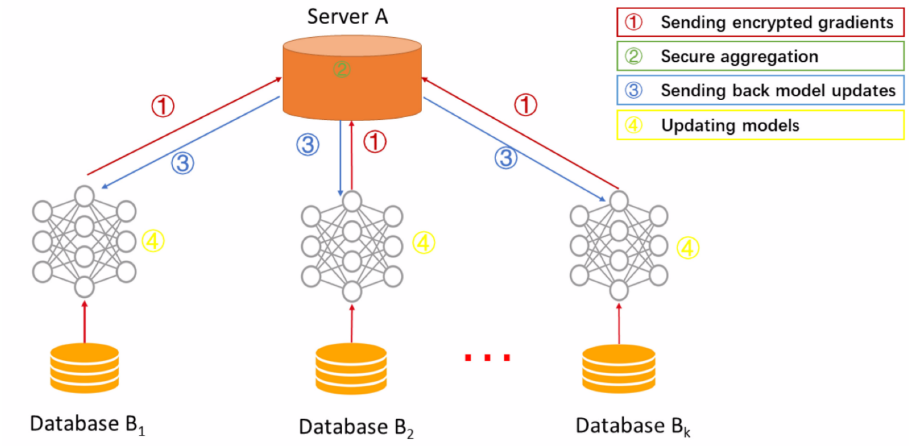
1.anaconda、python3.7、PyTorch

2.GPU 安装 CUDA、cuDNN

数据集：CIFAR10

模型：ResNet-18

环境角色: 中心服务器、多个客户端

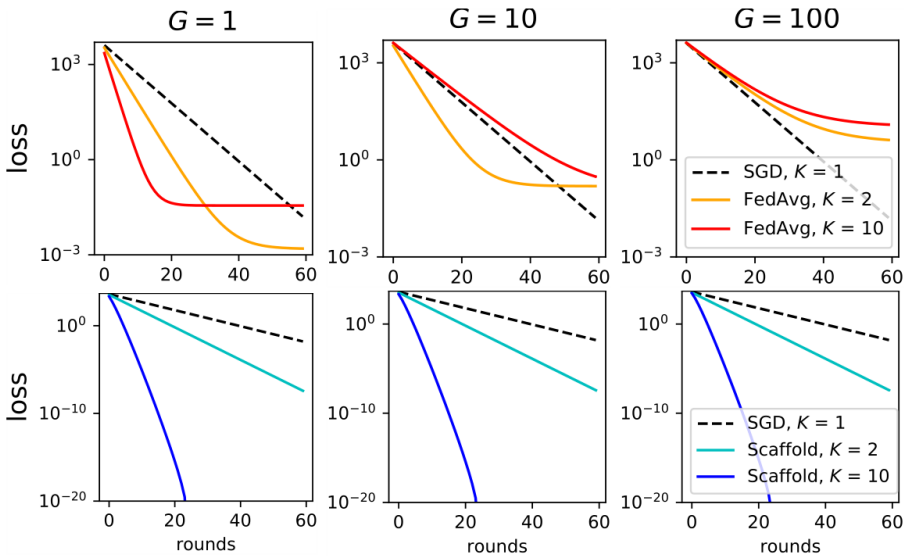


基本的流程：

1. 服务器按配置生成初始化模型，客户端按照顺序将数据集横向不重叠切割
2. 服务器将全局模型发送给客户端
3. 客户端接收全局模型（来自服务器）通过本地多次迭代计算本地参数差值返回给服务器
4. 服务器聚合各个客户端差值更新模型，再评估当前模型性能
5. 如果性能未达标，则重复 2 过程，否则结束

5 实验结果分析

5.1 模拟数据集



将梯度异质性 $(A1)$ 变为 $G \in [1, 10, 100]$。可以发现，对于所有的 G 值，随着局部更新次数的增加，FedAvg 收敛变慢。这是因为随着本地更新次数的增加，客户端漂移会增加，从而阻碍全局收敛。此外，当我们增加 G 时（增加异质性），FedAvg 的收敛继续减缓。当异质性很小 $(G = \beta = 1)$ 时，FedAvg 可以与 SGD 竞争。

5.2 EMNIST 数据集

	Epochs	0% similarity (sorted) Num. of rounds	Speedup	10% similarity Num. of rounds	Speedup	100% similarity (i.i.d.) Num. of rounds	Speedup
SGD	1	317	(1×)	365	(1×)	416	(1×)
SCAFFOLD	1	77	(4.1×)	62	(5.9×)	60	(6.9×)
	5	152	(2.1×)	20	(18.2×)	10	(41.6×)
	10	286	(1.1×)	16	(22.8×)	7	(59.4×)
	20	266	(1.2×)	11	(33.2×)	4	(104×)
FedAvg	1	258	(1.2×)	74	(4.9×)	83	(5×)
	5	428	(0.7×)	34	(10.7×)	10	(41.6×)
	10	711	(0.4×)	25	(14.6×)	6	(69.3×)
	20	1k+	(< 0.3×)	18	(20.3×)	4	(104×)
FedProx	1	1k+	(< 0.3×)	979	(0.4×)	459	(0.9×)
	5	1k+	(< 0.3×)	794	(0.5×)	351	(1.2×)
	10	1k+	(< 0.3×)	894	(0.4×)	308	(1.4×)
	20	1k+	(< 0.3×)	916	(0.4×)	351	(1.2×)

可以发现：

1.SCAFFOLD 在所有的相似度中表现都是最好的。

2. 当数据间相似度为 0 时，FedAvg 随着本地更新次数的增加，其通信轮数也在增加，并且最后始终无法达到预定精度。

3. 随着相似度增加，FedAvg 和 SCAFFOLD 都始终优于 SGD。

4.FedProx 的效果始终是最差的。

6 总结与展望

本文研究了异质性对联邦学习优化算法性能的影响。理论分析表明，FedAvg 会受到梯度差异的严重阻碍，甚至比 SGD 还要慢。鉴于此，本文提出了一种新的联邦优化算法 SCAFFOLD，SCAFFOLD 引入了服务器控制变量 c 和客户端控制变量 c_i，控制变量中含有模型的更新方向信息，通过在本地模型的更新公式中添加一个修正项 $c-c_i$，SCAFFOLD 克服了梯度差异，有效缓解了 client-drift。

参考文献

[1] KARIMIREDDY S P, KALE S, MOHRI M, et al. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning[C]//: vol. 119. 2020: 5132-5143.