《Communication-Efficient Learning of Deep Networks from Decentralized Data》

關鍵:

1.Non-IID
2.Unbalanced
3.Massively distributed
4.limited communication

優化思路:

我们假设一个同步更新方案，在几轮通信中进行。有一组固定的 K 个客户，each with a fixed local dataset。

1.在每一轮开始时，随机选择一部分客户 。我们只选择一部分客户以提高效率，因为我们的实验表明，超过一定数量的客户，收益就会递减。

2.服务器向这些客户中的每一个发送当前的全局算法状态 current global algorithm state（例如，当前的模型参数）使各个 client 采用相同的参数模型进行训练，可以有效的减少训练集的损失。

3.然后，每个被选中的客户端根据全局状态和其本地数据集进行本地计算，并向服务器发送一个更新。

4.然后，服务器将这些更新应用于其全局状态，这个过程不断重复。

成本:

通訊成本占大头，所以使用额外的计算来减少训练一个模型所需的通信轮数。主要有 2 种方法来增加（额外）计算：

1）增加并行：在每一个通信轮中，使用更多个独立工作的客户端；

2）增加每一个客户端上的计算：在每一个通信轮次中，每个客户端不是执行一个简单的梯度计算，而是执行一个更复杂的计算。

FedAvg:

若全部用戶都參與的話，則每个客户端使用本地数据对当前模型本地地进行 SGD 演算法，然后服务器端再对结果模型做一个加权平均。

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:

initialize w_0

for each round $t = 1, 2, \dots$ **do**

$m \leftarrow \max(C \cdot K, 1)$

$S_t \leftarrow$ (random set of m clients)

for each client $k \in S_t$ **in parallel do**

$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$

$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w): *// Run on client k*

$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)

for each local epoch i from 1 to E **do**

for batch $b \in \mathcal{B}$ **do**

$w \leftarrow w - \eta \nabla \ell(w; b)$

return w to server

g_k: *当前模型w_t在客户端本地数据上的平均梯度*

w_{t+1}^k: 本地端的下一輪模型

η: *學習率*

n: 總*data的数量*

n_k: *client k上data的数量*