Federated Learning with Non-IID Data

问题背景

移动设备已成为全球数十亿用户的主要计算资源，这些设备会产生大量有价值的数据，使用这些数据训练的机器学习模型具有提高许多应用程序智能性的潜力。 但是在传统机器学习的设置中，在移动设备上启用这些功能需要设备在服务器上共享本地数据才能训练出令人满意的模型。但是这个从隐私，安全，法规或经济的角度来看，这是很难实现的。在这种背景下，联邦学习可以在保证数据隐私安全及合法合规的基础上，实现共同建模，提升AI模型的性能。

但是目前的联邦学习训练的算法，也就是联合平均算法(FebAvg)，在非独立同分布的数据集上训练出来的模型性能不能令人满意。如下图所示：

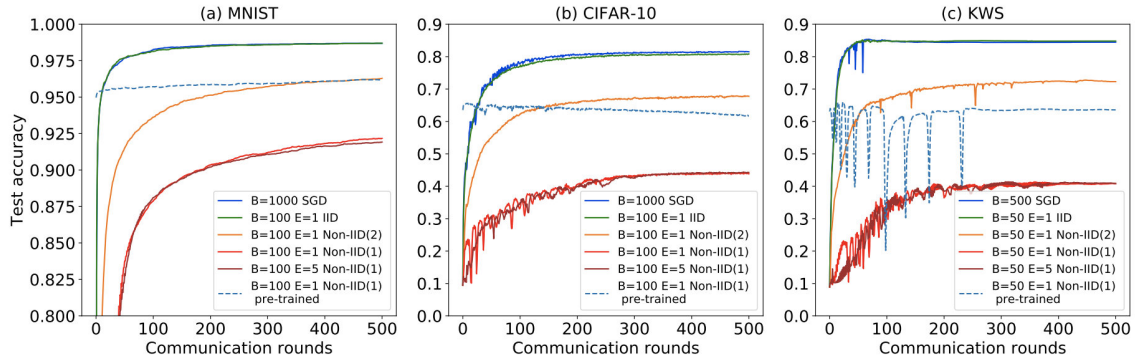

Figure 1: Test accuracy over communication rounds of $FedAvg$ compared to SGD with IID and non-IID data of (a) MNIST (b) CIFAR-10 and (c) KWS datasets. Non-IID(2) represents the 2-class non-IID and non-IID(1) represents the 1-class non-IID.

假设一共有10个设备参与联邦学习，使用同一个数据集，然后用不同的算法和不同的数据集划分方法来对模型进行训练，然后得到各自的准确率， B代表在单个设备上的训练集的大小，E表示在设备上训练的迭代次数：

- `B = 1000 SGD`：不对数据集进行划分，训练时使用`mini-batch`来选择训练的数据，不经过联邦学习，直接训练出模型
- `B = 100 E = 1 IID`：将数据集随机分为10个独立同分布的子集，随机分配到10个设备上进行训练，然后进行联合平均得到全局模型
- `B = 100 E = 1 Non-IID(2)`：将数据集分为20个子集，并且这些子集之间并不服从独立同分布，然后每个设备随机分配两个子集进行训练，最后进行联合平均得到全局模型
- `B = 100 E = 1 Non-IID(1)`：将数据集分为10个子集，并且这些子集之间并不服从独立同分布，然后每个设备随机分配一个子集进行训练，最后进行联合平均得到全局模型
- `B = 100 E = 5 Non-IID(1)`：同上，本地训练迭代次数改为5
- `B = 100 E = 1 Non-IID(1) pre-trained`：进行预训练

从上面几个图可以看出几个点：

- `FebAvg`算法在独立同分布的数据集上的性能与不经过联邦学习直接训练得到的模型性能相差无几
- `FebAvg`算法在非独立同分布的数据集上的性能要比以上两种情况要差很多
- 在训练集的划分上，`Non-IID(2)`要比`Non-IID(1)`要好，直观上理解，这是因为`Non-IID(2)`独

立同分布的“程度”要比`Non-IID(1)`要大
- 即便增加本地训练的迭代次数，也不能改善性能
- 预训练并不能显著提高最终得到的模型的性能

论文贡献

- 为什么`FebAvg`算法在非独立同分布的数据集上性能不好
- 一种改进的`FebAvg`算法

根源

因为模型的性能跟它的权重矩阵是直接相关的，可以通过寻找`FebAvg`算法训练出来的模型的权重矩阵和`SGD`算法训练出来的模型的权重矩阵的差异，来找到性能下降的原因

可以将以上两种权重矩阵的差异定义为：

$$weight\ divergence = \frac{||w^{FebAvg} - w^{SGD}||}{||w^{SGD}||}$$

然后可以得到上面这个实验中训练出来的模型各层的$weight\ divergence$值：

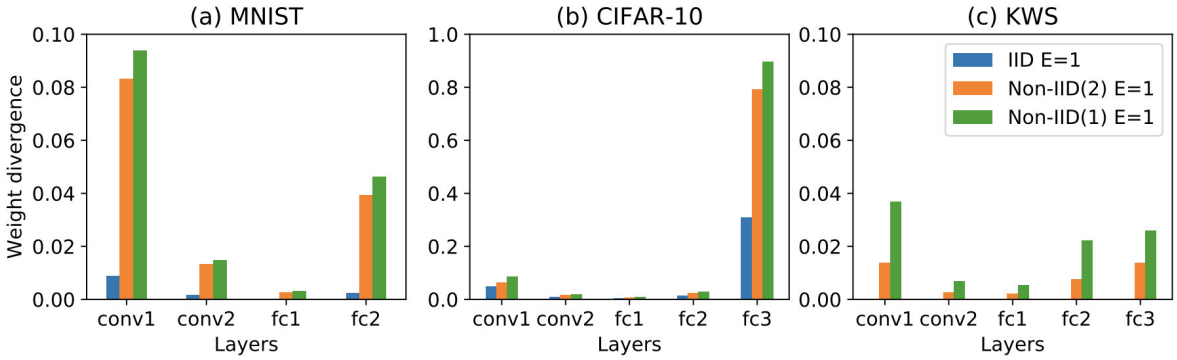

Figure 2: Weight divergence of CNN layers for IID, 2-class non-IID and 1-class non-IID.

可以发现：

- 在独立同分布的数据集中训练的模型参数跟使用`SGD`直接训练得到的模型差异不大
- 而在非独立同分布的数据集中训练的模型参数跟使用`SGD`直接训练得到的模型差异很大
- `Non-IID(2)`要比`Non-IID(1)`差异要小

因此问题从`FebAvg`在两种不同的数据集设置上的性能差异转变为模型参数的差异

假设我们正在做的是分类问题，那么有总的数据集C个分类，设$y = [C]$为分类的标签，$[C] = 1, \ldots, C$。x是特征向量，并且x, y在$X \times Y$上符合概率分布p，将模型看成一个映射$f : X -> S$，S是一个概率单纯形，满足$S = \{z | \sum_{i=1}^{C} z_i = 1, z_i >= 0, \forall i \in [C]\}$，并且$f_i(x, w)$表示对于输入的样本$x$，模型将样本分类到第$i$类的概率。然后我们可以用交叉熵来定义损失函数：

$$\ell(\boldsymbol{w}) = \mathbb{E}_{\boldsymbol{x}, y \sim p}[\sum_{i=1}^{C} \mathbb{1}_{y=i} \log f_i(\boldsymbol{x}, \boldsymbol{w})] = \sum_{i=1}^{C} p(y = i) \mathbb{E}_{\boldsymbol{x}|y=i}[\log f_i(\boldsymbol{x}, \boldsymbol{w})].$$

然后模型学习的目标就变为：

$$\min_{\boldsymbol{w}} \sum_{i=1}^{C} p(y=i)\mathbb{E}_{\boldsymbol{x}|y=i}[\log f_i(\boldsymbol{x}, \boldsymbol{w})].$$

使用SGD更新的方式为：

$$\boldsymbol{w}_t^{(c)} = \boldsymbol{w}_{t-1}^{(c)} - \eta \nabla_{\boldsymbol{w}} \ell(\boldsymbol{w}_{t-1}^{(c)}) = \boldsymbol{w}_{t-1}^{(c)} - \eta \sum_{i=1}^{C} p(y=i) \nabla_{\boldsymbol{w}} \mathbb{E}_{\boldsymbol{x}|y=i}[\log f_i(\boldsymbol{x}, \boldsymbol{w}_{t-1}^{(c)})].$$

然后第k个设备也会在本地训练模型，更新方法也是一样的：

$$\boldsymbol{w}_t^{(k)} = \boldsymbol{w}_{t-1}^{(k)} - \eta \sum_{i=1}^{C} p^{(k)}(y=i) \nabla_{\boldsymbol{w}} \mathbb{E}_{\boldsymbol{x}|y=i}[\log f_i(\boldsymbol{x}, \boldsymbol{w}_{t-1}^{(k)})].$$

假设每T个轮次进行一次全局模型的更新，那么在第m次更新之后得到的模型为：

$$\boldsymbol{w}_{mT}^{(f)} = \sum_{k=1}^{K} \frac{n^{(k)}}{\sum_{k=1}^{K} n^{(k)}} \boldsymbol{w}_{mT}^{(k)}.$$

从这里可以看出，差异主要发生在$w_t^{(k)}$和$w_t^{(c))}$之间，因为在各个设备上的数据不是独立同分布的，那么用这些数据训练出来的模型会产生差异

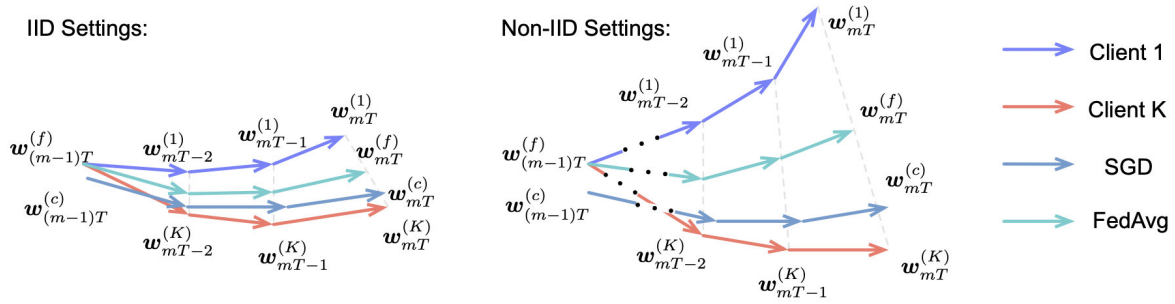

Figure 3: Illustration of the weight divergence for federated learning with IID and non-IID data.

改进的FebAvg算法

在进行联邦学习的训练之前，预设置一个共享的数据集，这个数据集可以有各设备按照数据集的大小比率预先上传一小部分的数据组成，然后之后各个设备进行训练的时候会用这个共享的数据集和本地的数据集一起训练，如下图所示：

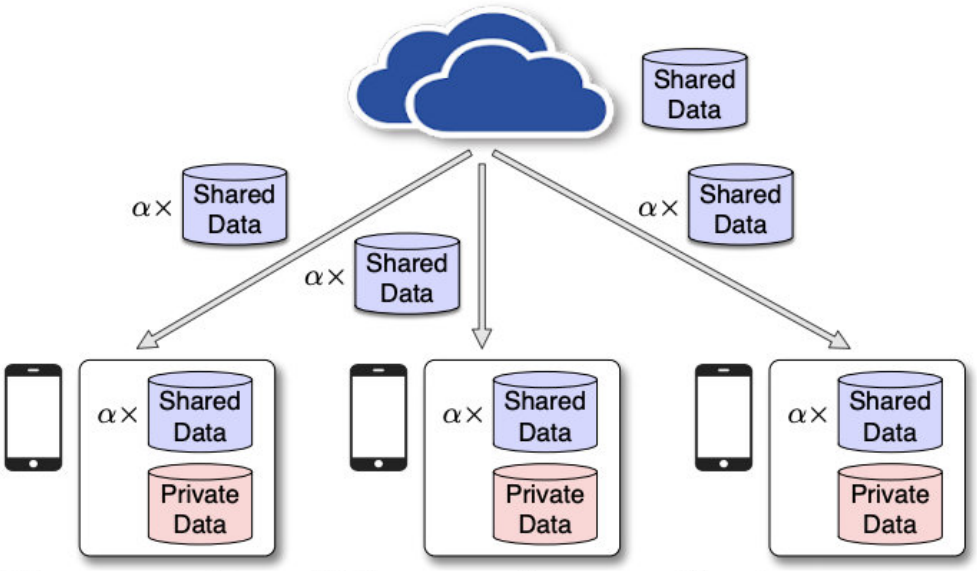

Figure 6: Illustration of the data-sharing strategy.

通过这种方法，确实可以提高训练出来的模型的性能

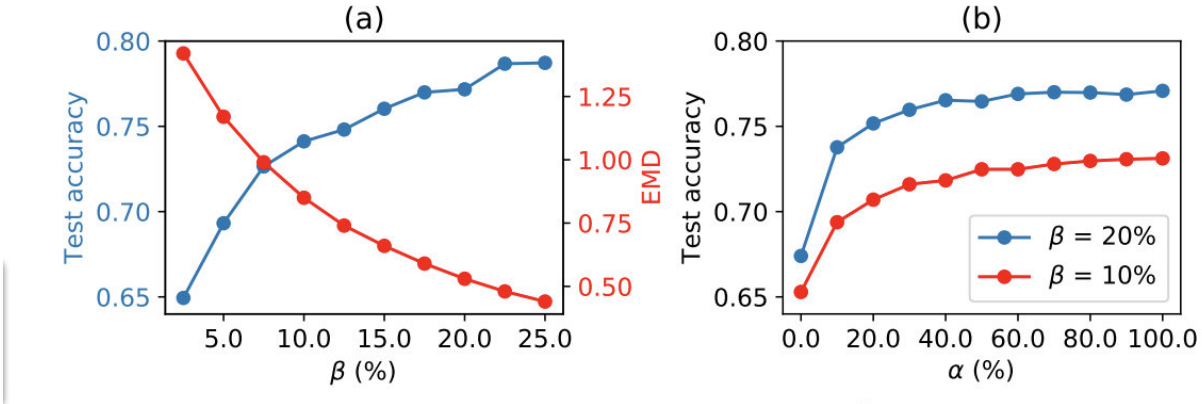

Figure 7: (a) Test accuracy and EMD vs. β (b) Test accuracy vs. the distributed fraction α

其中$\beta = \frac{||G||}{||D||} \times 100\%$，表示的是共享数据集大小与总的数据集大小的一个比例，α是分布分数

该改进算法的优缺点

优点：

- 不需要很大的额外开销，只需要在训练之前各设备上传数据，然后再下载共享数据集到本地就好
- 可以通过牺牲一小部分的设备隐私，来获得性能的较大提升

缺点：

- 很难获得接近`SGD`的准确率

- 进一步增加共享数据集的大小很难得到显著的性能提升