Federating Learning & Pysyft Introduction

paper : Communication-Efficient Learning of Deep Networks from Decentralized Data

https://github.com/OpenMined/PySyft

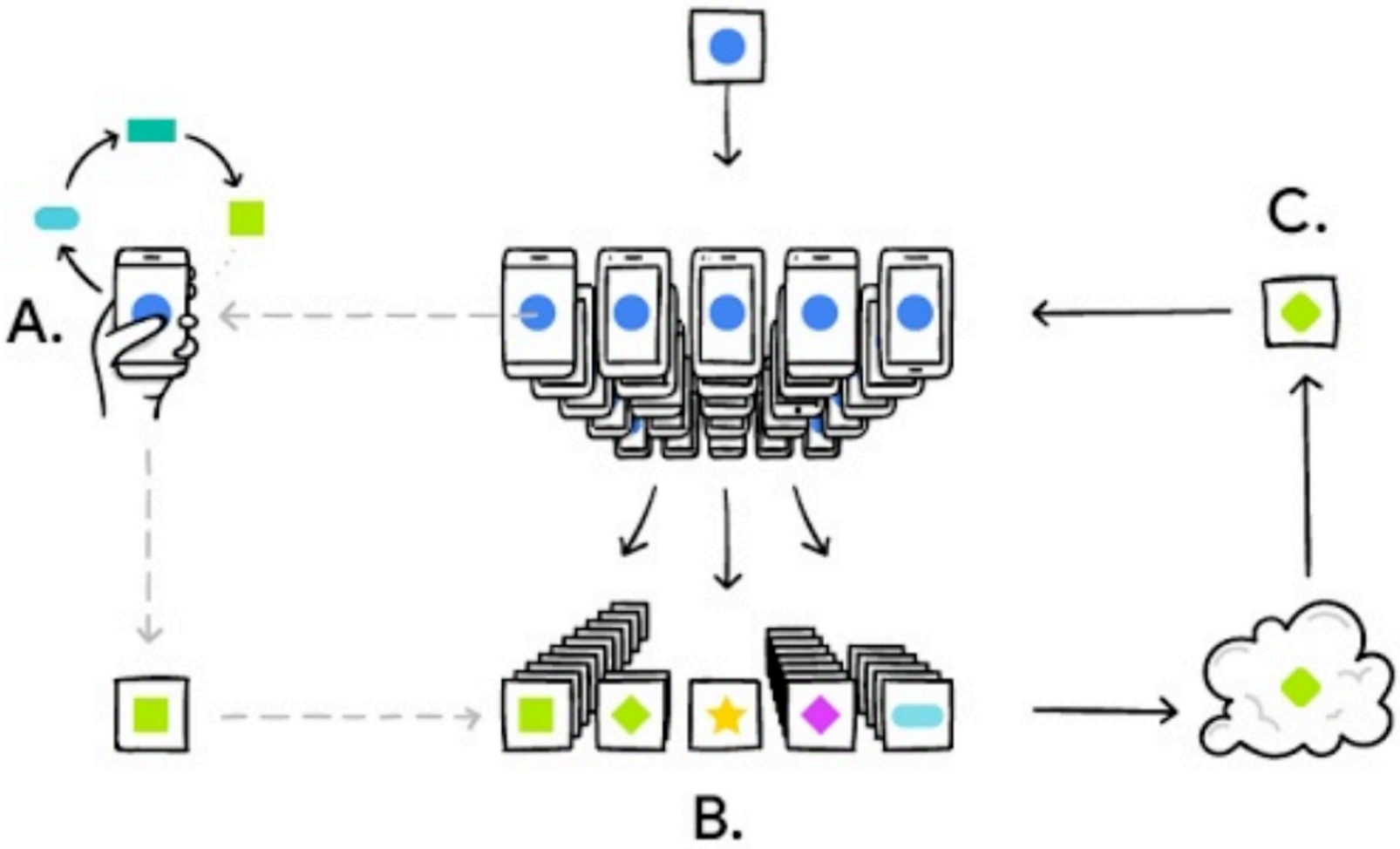
刘丽锋

Federating Learning

概述

The FederatedAveraging Algorithm

训练数据分布在移动设备上，并通过本地计算的更新模型 =>聚合得共享模型


A.
C.
B.

1. Introduction

终端设备计算力的发展

终端持有的数据量

数据的隐私性？？

1. Introduction

联合学习的理想问题具有以下特性：

1.非平衡

2. 非IID数据分布

优化 1，2，4

3. 终端数量多

4. 通信成本

2.Federated Learning

随机梯度下降（SGD）的变体进行优化：
通过使用简单的基于梯度的方法使模型的结构（以及因此损失函数）更适合于优化

=> **从SGD开始构建用于联合优化的算法**

FederatedSGD

Client

$$g_k = \nabla F(w_t);$$

Server

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

通信?

2.Federated Learning

The FederatedAveraging Algorithm

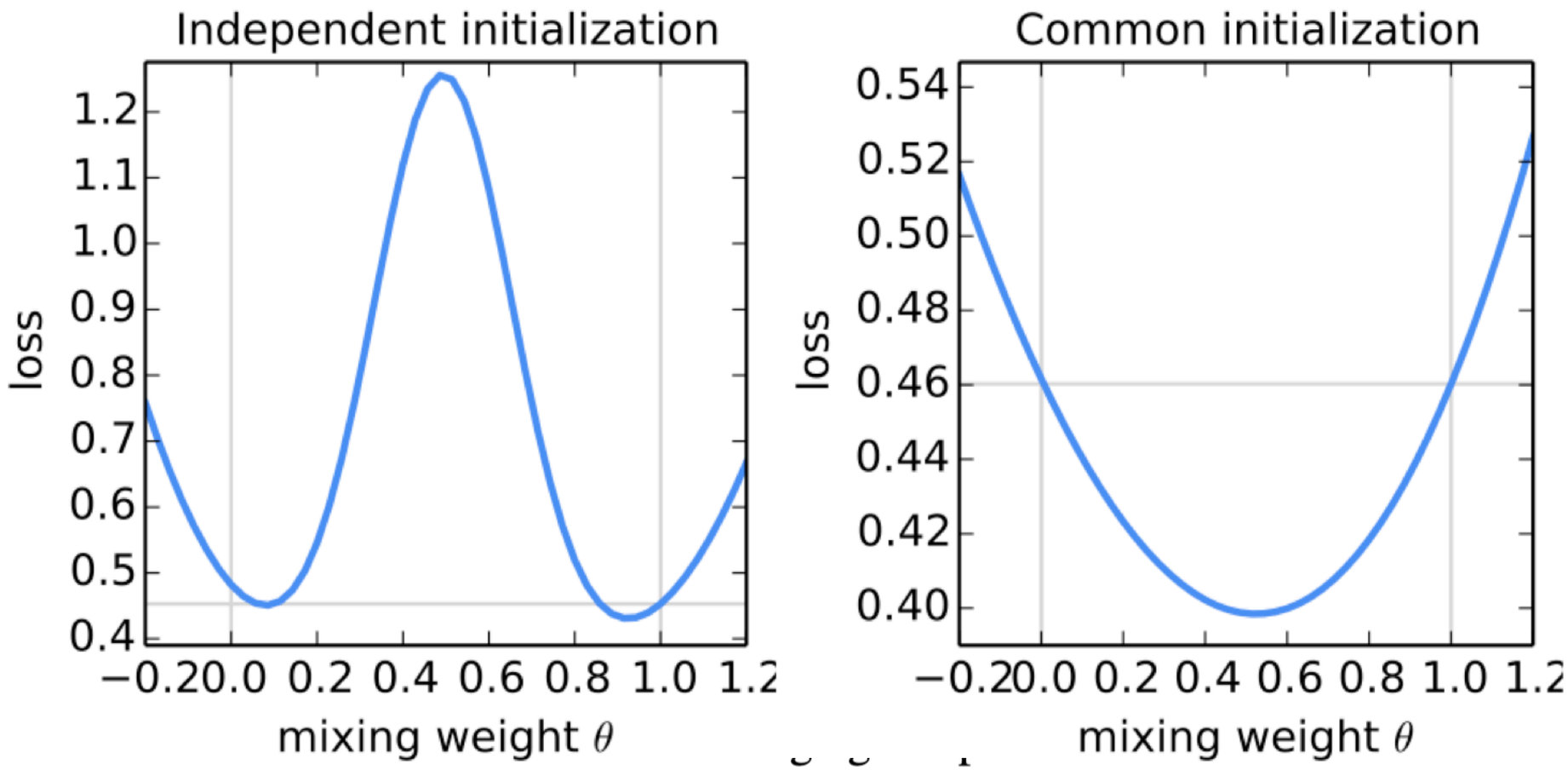
每一轮中选择一小部分C客户，并计算这些客户持有的所有数据的损失梯度。

每个客户端使用其本地数据在当前模型上本地采用梯度下降的一步

迭代本地更新为每个客户端添加迭代E次； B batch_size

然后**服务器对所得模型进行加权平均**

2.Federated Learning



$$\theta w + (1-\theta) w'$$

与分布式算法不一样

不考虑**不平衡和非IID数据**

2.Federated Learning

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)

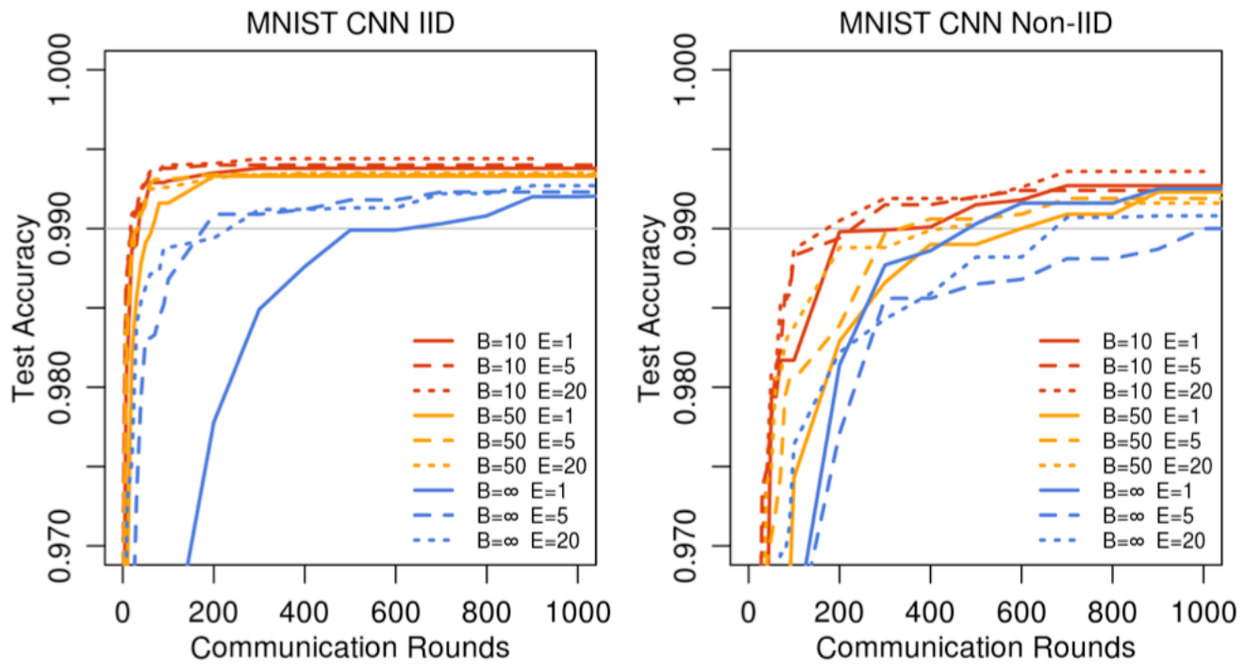
for each local epoch i from 1 to E **do**

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

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

return w to server

3.Result


MNIST CNN IID
MNIST CNN Non-IID
Test Accuracy
1.000
0.990
0.980
0.970
0
200
400
600
800
1000
Communication Rounds
B=10 E=1
B=10 E=5
B=10 E=20
B=50 E=1
B=50 E=5
B=50 E=20
B=∞ E=1
B=∞ E=5
B=∞ E=20

4. Conclusions and Future Work

隐私保护？

多方安全计算

差分隐私 dp

同态加密

秘钥分享

Pysyft Introduction

A library for encrypted, privacy preserving deep learning

Demo

```
In [1]: import syft as sy
        hook = sy.TorchHook()
        sy.FloatTensor([1,2,3,4,5])

Out[1]:
         1
         2
         3
         4
         5
        [syft.core.frameworks.torch.tensor.FloatTensor of size 5]

In [2]: import torch
        x = torch.FloatTensor([1,2,3,4,5])
        y = x + x
        print(x)
        print(y)
        print(y[0])

         1
         2
         3
         4
         5
        [syft.core.frameworks.torch.tensor.FloatTensor of size 5]
```

Demo

```
bob = sy.VirtualWorker(id="bob")
```

For all intents and purposes, Bob's machine is on another planet - perhaps on Mars! But, at the m
create some data so that we can send it to Bob and learn about pointers!

```
x = torch.FloatTensor([1,2,3,4,5])
y = torch.FloatTensor([1,1,1,1,1])
```

And now - let's send our tensors to Bob!!

```
x_ptr = x.send(bob)
y_ptr = y.send(bob)
```

BOOM! Now Bob has two tensors! Don't believe me? Have a look for youself!

```
bob._objects
```

```
{26311085192: [_LocalTensor - id:26311085192 owner:bob],
 81620284840: [_LocalTensor - id:81620284840 owner:bob]}
```

结构

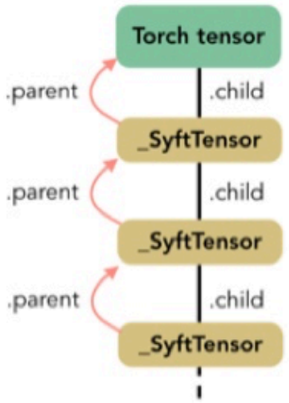


Figure 1: General structure of a tensor chain

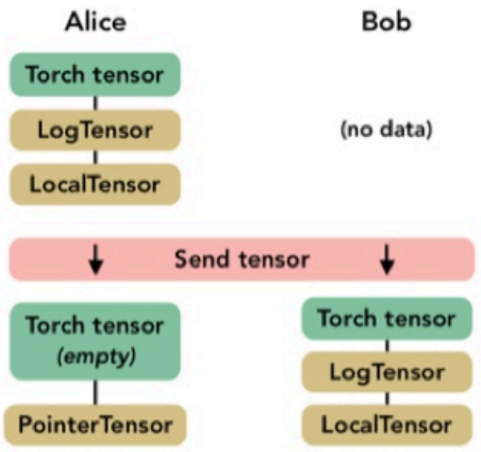


Figure 2: Impact of sending a tensor on the local and remote chains

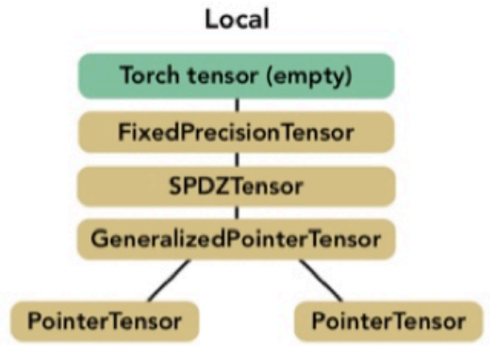


Figure 3: Chain structure of a SPDZ tensor

the chain of commands

a tensor chain

Hook

完成重载 pytorch的 variables 和 Tensors.

> *SyftTensors表示数据的状态或者转换(不存放真实数据)，相互之间可以链接在一一起*
> *Wrapper (FloatTensor) -> LocalTensor (our class) -> FloatTensor (actually holds the data)*

Note：Wrapper就是包装器，被pytorch其余使用，一起工作，统一接口; operation and transaformation 就是 LocalTensor; FloatTensor就是真实存放数据的FloatTensor

关键点：
1. Wrapper (FloatTensor) 与LocalTensor (our class)工作
2. LocalTensor(our class)与FloatTensor (actually holds the data)工作

改进:
LocalTensor.child -> Wrapper 提高了节省内存，提高效率

worker

worker类型分为Virtual Worker; Websocket Worker; socket Worker三类:

1. Virtual Worker 用于本地开发测试，
2. Websocket Worker用于本地浏览器中不同标签页开发测试，不涉及网络
3. Socket Worker真实环境的开发部署使用

使用**work机制**来实现不同client间进行识别连接;发送 the chain of command 来控制对应的worker

SyftTensor

_LocalTensor 存储数据，子类存储数据

_TorchTensor 对象发送和交流