

Methodology, Ethics and Practice of Data Privacy

实验二

Lan Zhang

*School of Computer Science and Technology
University of Science and Technology of China
Spring 2021*

Part 1

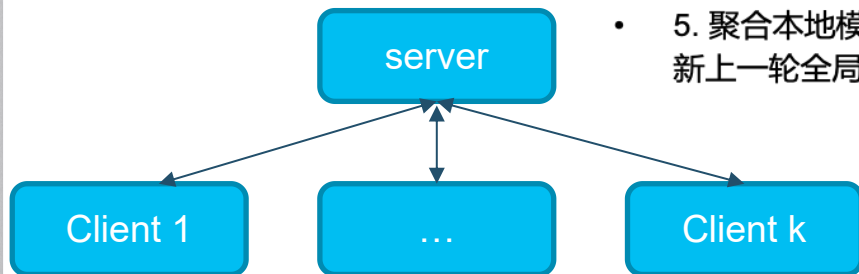
横向联邦学习简要介绍

横向联邦学习的一种实现

- 对于每一个epoch

Server端

- 4. 收到来自client的梯度 $[\Delta w_1, \Delta w_2, \dots, \Delta w_k]$
- 5. 聚合本地模型更新 $\Delta w_{avg} = \frac{\sum_{i=1}^k \Delta w_i}{k}$, 更新上一轮全局模型, 并发送给client



$$w_{t+1} = w_t + \frac{1}{k} \left(\sum_{i=1}^k \Delta w_i^t \right)$$

Client 端

- 1. 每个client 在自己的模型上训练local_epoch轮。
- 2. 求解local_epoch轮中参数的模型更新信息 Δw
- 3. 将 Δw 上传到server端
- 6. 获取更新后的模型作为自己的模型。

横向联邦学习中的隐私问题

模型更新会泄露用户数据集信息

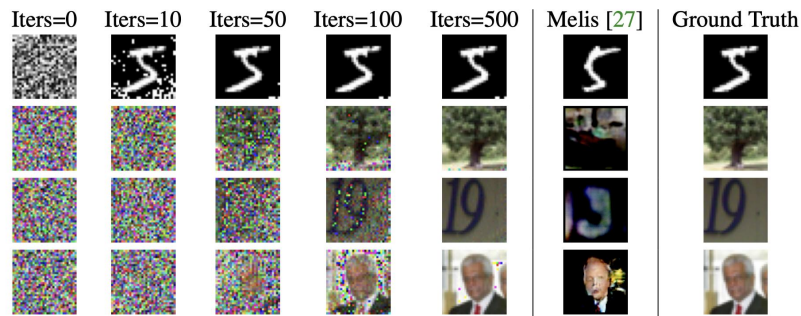


Figure 3: The visualization showing the deep leakage on images from MNIST [22], CIFAR-100 [21], SVHN [28] and LFW [14] respectively. Our algorithm fully recovers the four images while previous work only succeeds on simple images with clean backgrounds.

Part 2

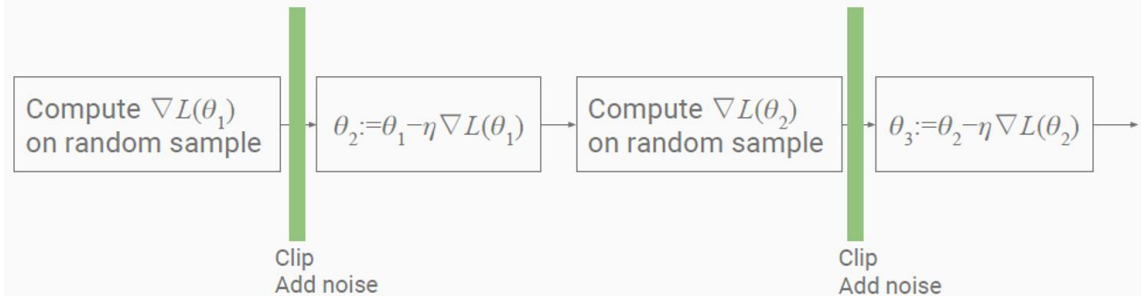
差分隐私机制在联邦学习中的应用

DP SGD in centralized model

- Stochastic gradient decent (SGD)



- Differentially Private SGD (DP SGD)



DP SGD in centralized model

» 算法描述

Algorithm 1 Differentially private SGD (Outline)

Input: Examples $\{x_1, \dots, x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N} \sum_i \mathcal{L}(\theta, x_i)$. Parameters: learning rate η_t , noise scale σ , group size L , gradient norm bound C .

Initialize θ_0 randomly

for $t \in [T]$ **do**

 Take a random sample L_t with sampling probability L/N

Compute gradient

 For each $i \in L_t$, compute $\mathbf{g}_t(x_i) \leftarrow \nabla_{\theta} \mathcal{L}(\theta_t, x_i)$

Clip gradient

$\tilde{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i) / \max(1, \frac{\|\mathbf{g}_t(x_i)\|_2}{C})$

Add noise

$\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L} (\sum_i \tilde{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I}))$

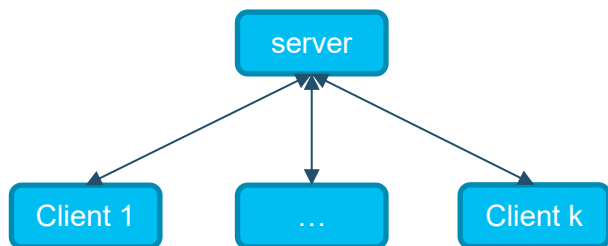
Descent

$\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{\mathbf{g}}_t$

Output θ_T and compute the overall privacy cost (ϵ, δ) using a privacy accounting method.

DP SGD in federated model

- 为了保护每个client的模型更新信息，对于每一个epoch



Server端

- 4. 收到来自client的梯度 $[\Delta w_1, \Delta w_2, \dots, \Delta w_k]$
- 5. 聚合本地模型更新 $\Delta w_{avg} = \frac{\sum_{i=1}^k \Delta w_i}{k}$ ，更新上一轮全局模型，并发送给client

$$w_{t+1} = w_t + \frac{1}{k} \left(\sum_{i=0}^k \Delta w^i \right) \xrightarrow{\text{添加高斯机制}} w_{t+1} = w_t + \frac{1}{k} \left(\sum_{i=0}^k \Delta w^i / \max(1, \frac{\|\Delta w^i\|_2}{C}) + N(0, \sigma^2 C^2 I) \right)$$

w_{t+1} 指更新后的模型参数， w_t 指原有的模型参数， k 是指client数量， Δw^i 是指用户 i 上传的模型更新， σ 是高斯机制中的参数， C 是模型更新的截断值， I 是单位矩阵

高斯机制中的 σ 与 (ϵ, δ) - DP的关系

$$w_{t+1} = w_t + \frac{1}{k} \left(\sum_{i=0}^k \Delta w^i \right) \xrightarrow{\text{添加高斯机制}} w_{t+1} = w_t + \frac{1}{k} \left(\sum_{i=0}^k \Delta w^i / \max(1, \frac{\|\Delta w^i\|_2}{C}) + N(0, \sigma^2 C^2 I) \right)$$

当 σ 满足以下定理时，添加高斯机制满足 (ϵ, δ) - DP

Theorem 3.22. Let $\epsilon \in (0, 1)$ be arbitrary. For $c^2 > 2 \ln(1.25/\delta)$, the Gaussian Mechanism with parameter $\sigma \geq c \Delta_2(f)/\epsilon$ is (ϵ, δ) -differentially private.

$$\sigma \geq \sqrt{2 \ln \left(\frac{1.25}{\delta} \right)} * \frac{C}{\epsilon} \quad (1)$$

TODO

» 基本内容

- 在原有代码框架上添加上述DP机制
- 代码正确，关键部分有注释
- 实验说明参数 σ, C 对模型准确度的影响，并计算对应的 ϵ 值 ($\delta = 10^{-3}$)
- 实验报告（关键代码截图以及说明）

» 附加内容

- 为了保证模型的可用性， σ 取值较小，此时由公式(1)可能计算出 $\epsilon \geq 1$ ，不满足定理3.22的条件，因此无法由此定理证明满足DP。
- 请调研当 $\epsilon \geq 1$ 时，如何证明此机制满足DP；或调研是否存在其他机制可解决此问题。

Part 3

同态加密算法在联邦学习中的应用

Paillier 同态加密简介

» 加法同态加密

- 加密情况下，仍然可以计算两个信息的和
- given encryption of m_1 and m_2 , one can compute the encryption of $m_1 + m_2$. $E(m_1) \oplus E(m_2) = E(m_1 + m_2)$

» Paillier: 一种基于公钥系统的加法同态和乘法半同态加密系统

- 密钥生成
随机并且独立选择两个**相同位数**大质数 p, q . 计算 $n = pq$
当 $p \neq q$ 时, 欧拉函数 $\lambda = \phi(n) = (p-1)(q-1)$.
当 $p = q$ 时, 欧拉函数 $\lambda = \phi(n) = q(q-1)$.
选择一个整数 $g = n + 1, \mu = \phi(n)^{-1} \bmod n$
(μ 有可能不存在, 需要重新生成).
公钥为 (n, g) , 私钥为 (λ, μ) .
- 加密
消息 $m (0 \leq m \leq n)$
选择随机数 r 满足 $0 < r < n$ 且 $r \in \mathbb{Z}_n^*$ (i.e., 保证 $\gcd(r, n) = 1$)
计算密文 $c = g^m \cdot r^n \bmod n^2$

Paillier 同态加密简介

» Paillier: 一种基于公钥系统的加法同态加密系统

- 解密

密文 $c \in \mathbb{Z}_{n^2}^*$

明文 $m = L(c^\lambda \bmod n^2) \cdot \mu \bmod n$, 其中 $L(x) = \lfloor \frac{x-1}{n} \rfloor$, 表示 n 除 $x-1$ 的商取下整

- 密文加法

$$Dec(Enc(m_1, r_1) \cdot Enc(m_2, r_2) \bmod n^2) = m_1 + m_2 \bmod n$$

- 与明文常数加法

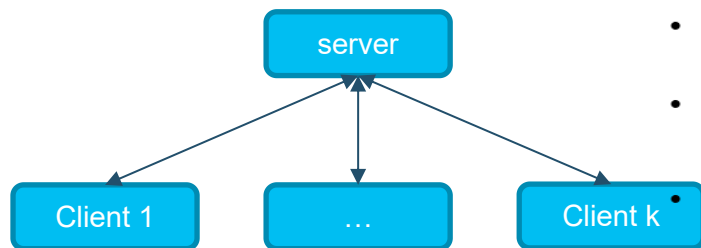
$$Dec(Enc(m_1, r_1) \cdot g^{m_2} \bmod n^2) = m_1 + m_2 \bmod n$$

- 与明文常数乘法

$$Dec(Enc(m_1, r_1)^k \bmod n^2) = km_1 \bmod n$$

Paillier 在横向联邦学习中的应用

- 为了保护client的模型更新信息，对于每一个epoch



Server端

- 4. 收到来自client的加密模型更新信息 $[enc(\Delta w_i)]$
- 5. 利用同态加密，聚合各个梯度
$$\Delta w_{avg} = \sum_{i=1}^k enc(\Delta w_i) \times \frac{1}{k}$$
- 6. 将 Δw_{avg} 发送给各个client

Client 端

- 1. 每个client 在自己的模型上训练local_epoch轮。
- 2. 求解local_epoch轮中参数的更新 Δw （默认client学习率公开且一致的情况下，可以用 $w_{new} - w_{old}$ 代替）
- 3. 利用同态加密，将加密之后的 $enc(\Delta w)$ 上传到server端
- 7. 利用收到的 Δw_{avg} ，更新自己的模型参数

Part2 实验内容

» 任务1: 了解paillier原理

- 使用gmpy2库填充paillier.py中enc, dec, enc_add, enc_add_const和enc_mul_const函数, 验证结果的正确性。
- 测试密钥长度为1024位(二进制)时, 长度10-1000bits整数加法, 验证其结果是否正确, 并合理测试运行时间。

» 任务2: paillier + federal learning

- 为了防止server端获得client端的梯度和模型参数, 使用paillier对于上传的模型更新进行加密和解密运算, 并且实现在加密情况下的梯度聚合操作。

» 实验要求:

- 任务1, 实现结果正确, 代码清晰
- 任务2, 实现结果正确, 说明实现方法。对比模型在minst上的训练时间。

推荐实验工具

» Python 3.7及以上

» gmpy2库

- <https://www.lfd.uci.edu/~gohlke/pythonlibs/#gmpy>提供了 gmpy2- 2.0.8- cp37- cp37m- win amd64.whl 以及支持其他python > 3.4版本的的gmpy2库
- 将whl文件下载到本地，并使用

Eg: (python -m) pip install

gmpy2- 2.0.8- cp37- cp37m- win_amd64.whl

进行安装

- <https://gmpy2.readthedocs.io/en/latest/mpfr.html> gmpy2 的documentation
- 有可能会使用到的函数: mpz, powmod, invert, is_prime, random_state, mpz_urandomb, rint_round, log2, gcd

推荐实验工具

» python-paillier库

- Python-paillier库的文档

<https://python-paillier.readthedocs.io/en/stable/index.html>

- pip install phe

» 说明:

- 对于任务1, 要求掌握paillier的原理, 因此不能直接使用phe库进行加解密。
- gmpy2库和phe库也不是必须的, 同学们可以自行选择语言和实现方法。
- Paillier如何应用在浮点数上, 实验不要求掌握, 可以直接使用现有的paillier的库进行实现

实验中用到的参数和数据

- 数据集 mnist, 使用torchvision.dataset 提供mnist数据集
- 每个client的网络结构

```
class CNNMnist(nn.Module):  
    def __init__(self, args):  
        super(CNNMnist, self).__init__()  
        self.conv1 = nn.Conv2d(args.num_channels, 10, kernel_size=5)  
        self.conv2 = nn.Conv2d(10, 20, kernel_size=5)  
        self.conv2_drop = nn.Dropout2d()  
        self.fc1 = nn.Linear(320, 50)  
        self.fc2 = nn.Linear(50, args.num_classes)  
  
    def forward(self, x):  
        x = F.relu(F.max_pool2d(self.conv1(x), 2))  
        x = F.relu(F.max_pool2d(self.conv2_drop(self.conv2(x)), 2))  
        x = x.view(-1, x.shape[1]*x.shape[2]*x.shape[3])  
        x = F.relu(self.fc1(x))  
        x = F.dropout(x, training=self.training)  
        x = self.fc2(x)  
        return x
```

- Client数量, 学习率等超参数: 在options.py中定义

实验分数占比和截止日期

» Part 1

- 基本内容 32'
- 附加内容 8'

» Part 2

- 任务1 30'
- 任务2 30'

» 截止日期

- 7.1号

THANKS!

Any questions?

