带秘密分享验证的基于联邦学习



除去秘密共享部分之外联邦学习

横向联邦学习的服务端的主要功能是将被选择的客户端上传的本地模型进行模型聚合。对于一个功能完善的联邦学习框架，服务端的功能要复杂得多，比如服务端需要对各个客户端节点进行网络监控、对失败节点发出重连信号等。由于本工程是在本地模拟的，不涉及网络通信细节和失败故障等处理，因此不讨论这些功能细节，仅涉及模型聚合功能。

首先定义一个服务端类Server，类中的主要函数包括以下几个。

* 定义构造函数。在构造函数中，服务端的工作包括：第一，将配置信息拷贝到服务端中；第二，按照配置中的模型信息获取模型，这里我们使用torchvision 的models模块内置的ResNet-18/50/densenet121或者其他适合的模型。
* 具体代码在：**server.py**

```python

class Server(object):

def \_\_init\_\_(self, conf, eval\_dataset):

self.conf = conf

self.global\_model = models.get\_model(self.conf["model\_name"])

self.eval\_loader = torch.utils.data.DataLoader(eval\_dataset,

batch\_size=self.conf["batch\_size"], shuffle=True)

```

- 定义模型聚合函数。前面我们提到服务端的主要功能是进行模型的聚合，因此定义构造函数后，我们需要在类中定义模型聚合函数，通过接收客户端上传的模型，使用聚合函数更新全局模型。聚合方案有很多种，本节我们采用经典的FedAvg 算法。

```python

def model\_aggregate(self, weight\_accumulator):

for name, data in self.global\_model.state\_dict().items():

update\_per\_layer = weight\_accumulator[name] \* self.conf["lambda"]

if data.type() != update\_per\_layer.type():

data.add\_(update\_per\_layer.to(torch.int64))

else:

data.add\_(update\_per\_layer)

```

- 定义模型评估函数。对当前的全局模型，利用评估数据评估当前的全局模型性能。通常情况下，服务端的评估函数主要对当前聚合后的全局模型进行分析，用于判断当前的模型训练是需要进行下一轮迭代、还是提前终止，或者模型是否出现发散退化的现象。根据不同的结果，服务端可以采取不同的措施策略。

```python

def model\_eval(self):

self.global\_model.eval()

total\_loss = 0.0

correct = 0

dataset\_size = 0

for batch\_id, batch in enumerate(self.eval\_loader):

data, target = batch

dataset\_size += data.size()[0]

if torch.cuda.is\_available():

data = data.cuda()

target = target.cuda()

output = self.global\_model(data)

total\_loss += torch.nn.functional.cross\_entropy(output, target,

reduction='sum').item() # sum up batch loss

pred = output.data.max(1)[1] # get the index of the max log-probability

correct += pred.eq(target.data.view\_as(pred)).cpu().sum().item()

acc = 100.0 \* (float(correct) / float(dataset\_size))

total\_l = total\_loss / dataset\_size

return acc, total\_l

```

## 3.3 客户端

横向联邦学习的客户端主要功能是接收服务端的下发指令和全局模型，利用本地数据进行局部模型训练。与前一节一样，对于一个功能完善的联邦学习框架，客户端的功能同样相当复杂，比如需要考虑本地的资源（CPU、内存等）是否满足训练需要、当前的网络中断、当前的训练由于受到外界因素影响而中断等。目前偶看到流行的联邦学习框架源代码和文档，比如FATE，获取更多的实现细节。我们简化版：仅考虑客户端本地的模型训练细节。我们首先定义客户端类Client，类中的主要函数包括以下两种。

* 定义构造函数。在客户端构造函数中，客户端的主要工作包括：首先，将配置信息拷贝到客户端中；然后，按照配置中的模型信息获取模型，通常由服务端将模型参数传递给客户端，客户端将该全局模型覆盖掉本地模型；最后，配置本地训练数据，在本案例中，我们通过torchvision 的datasets 模块获取cifar10 数据集后按客户端ID切分，不同的客户端拥有不同的子数据集，相互之间没有交集。
* 主要代码在**client.py:**

```python

class Client(object):

def \_\_init\_\_(self, conf, model, train\_dataset, id = -1):

self.conf = conf

self.local\_model = models.get\_model(self.conf["model\_name"])

self.client\_id = id

self.train\_dataset = train\_dataset

all\_range = list(range(len(self.train\_dataset)))

data\_len = int(len(self.train\_dataset) / self.conf['no\_models'])

train\_indices = all\_range[id \* data\_len: (id + 1) \* data\_len]

self.train\_loader = torch.utils.data.DataLoader(self.train\_dataset,

batch\_size=conf["batch\_size"], sampler=torch.utils.data.sampler.SubsetRandomSampler(train\_indices))

```

秘密共享部分

使用了椭圆曲线加密生成了一个公钥和一个私钥，然后进行了证明和验证阶段。在证明阶段，随机选择一个比特b，如果b为0，则发送私钥k的值给验证方，如果b为1，则发送公钥pk的值给验证方。在验证阶段，随机选择一个比特b，如果b为0，则接受私钥k，并验证它是否可以用于签名，如果b为1，则接受公钥pk，并验证它是否可以用于验证签名。如果验证成功，输出"Verification succeeded"，否则输出"Verification failed”。

同时配合flask api：Flask框架实现了一个接收POST请求的API，路由为"/verify"。在请求中，会获取一个JSON格式的数据，并解析出其中的"b"和"signature"字段。如果"b"为0，则使用之前读取的公钥pk对象验证"signature"字段是否是对"Hello, world!"消息的签名；如果"b"为1，则同样使用pk对象验证"signature"字段。如果验证成功，返回一个JSON格式的响应，其中"result"字段为"Verification succeeded"；否则，返回一个400错误的响应，其中"result"字段为"Verification failed"。该API会在本地的5000端口监听请求