北 京 邮 电 大 学

教师指导本科毕业设计（论文）记录表

学院	网络空间安全学院		专业	网络空间安全专业	
学生姓名	卢亭松	学号	2019212443	班级	2019211806
指导教师姓名	陆月明	职称	教授		

第 1－2 周记录：

1. 学习机器学习、隐私保护相关的知识，对研究背景进行调研

联邦学习的定义为：在进行机器学习的过程中，各参与方可借助其他方数据进行联合建模。各方无需共享数据资源，即数据不出本地的情况下，进行数据联合训练，建立共享的机器学习模型。

联邦学习系统需要保证：|联邦学习模型的效果-传统方法模型的效果|<有界正数。

联邦学习的价值机制：联邦学习技术基于“合作共赢”的价值机制，对于商业利益而言极具价值。在这样一个联邦机制下，各个参与者的身份和地位相同，而联邦系统帮助大家建立了“共同富裕”的策略，能够带动跨领域的企业级数据合作、催生基于联合建模的新业态和模式、降低技术提升成本和促进创新技术发展。

联邦学习在不同场景下的联邦方式：

1. 横向联邦学习：在两个数据集的用户特征重叠较多而用户重叠较少的情况下，将数据集按照横向（即用户维度）切分，并取出双方用户特征相同而用户不完全相同的那部分数据进行训练。这种方法叫做横向联邦学习。
2. 纵向联邦学习：在两个数据集的用户重叠较多而用户特征重叠较少的情况下，把数据集按照纵向（即特征维度）切分，并取出双方相同而用户特征不完全相同的那部分数据进行训练。这种方法叫做纵向联邦学习。
3. 联邦迁移学习：在两个数据集的用户与用户特征重叠都较少的情况下，不对数据进行切分，而可以利用迁移学习来克服数据或标签不足的情况。这种方法叫做联邦迁移学习。

现有的联邦学习开源实现：目前业界中主要的联邦学习框架有 FATE、TensorFlow Federated、PaddleFL、Pysyft 等。

2. 基于 FATE 搭建联邦学习集群实验环境

部署 CentOS7 主机 x2、Docker 20.10.21 环境

下载 1.8.0-a 版本 KubeFATE 并解压

修改配置文件 parties.conf

执行部署脚本，（bash ./generate_config.sh，bash ./docker_deploy.sh all）

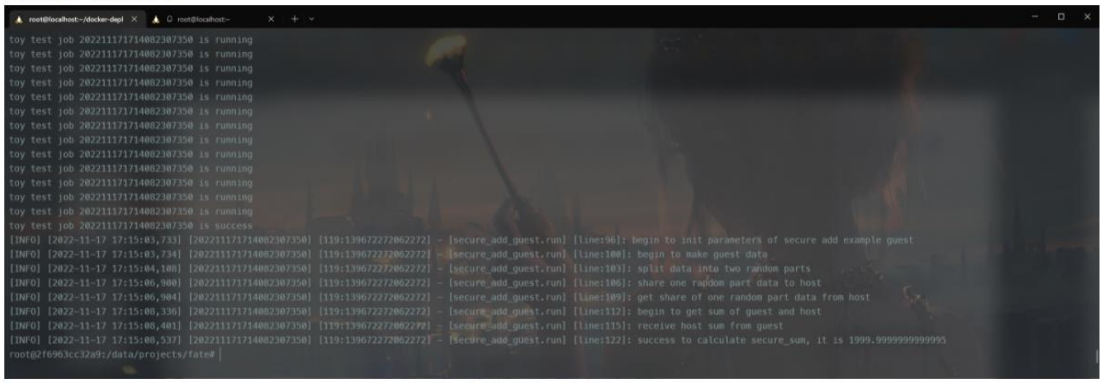
测试部署是否成功

$ docker exec -it confs-10000_client_1 bash

$ flow test toy --guest-party-id 10000 --host-party-id 9999

最终结果:

若成功，输出如下



3. 基于 DSL 配置任务

Host方上传数据

进入party10000 client容器

```
docker exec -it confs-10000_client_1 bash
```

修改examples/upload_host.json

```
cat > fateflow/examples/upload/upload_host.json <<EOF
{
  "file": "examples/data/breast_hetero_host.csv",
  "id_delimiter": ",",
  "head": 1,
  "partition": 10,
  "namespace": "experiment",
  "table_name": "breast_hetero_host"
}
EOF
```

1. file: 文件路径
2. table_name&namespace: 存储数据表的标识符号
3. head: 指定数据文件是否包含表头
4. partition: 指定用于存储数据的分区数

上传host数据

```
flow data upload -c fateflow/examples/upload/upload_host.json
```

修改 examples/test_hetero_lr_job_dsl.json

配置算法组件间的输入输出流及接口，根据此拼接得到完整的工作模型

提交任务

```
flow job submit -d fateflow/examples/lr/test_hetero_lr_job_dsl.json -c
fateflow/examples/lr/test_hetero_lr_job_conf.json
```

成功如下

4. Pipeline 配置任务

安装Pipeline与基本配置

Pipeline与fate_clinet相绑定，在任意具有python、pip等基础环境的机器上，运行`pip install fate_client`，即可完成安装。

成功安装Fate_client后，用户需要为Pipeline配置服务器信息和日志目录。Pipeline提供了用于快速设置的命令行工具。运行以下命令以了解更多信息。(这里需要找一下安装的位置添加一下环境变量)

```
pipeline --help
```

pipeline需要配置关联的`FATE Flow Service`,假设在`192.168.160.139`存在`FATE Flow Service`,则执行以下命令进行配置:

```
pipeline init --ip 192.168.160.139 --port 9380
```

后续我们可以在python文件中使用pipeline相关库，自动化地向fate_flow服务上传数据，发布训练任务等。

```
from pipeline.backend.pipeline import PipeLine
```

使用Pipeline上传数据

在开始建模任务之前，应上传要使用的数据。通常，参与方通常是包含多个节点的群集。因此，当我们上传这些数据时，数据将被分配给这些节点。

生成Pipeline实例

```
from pipeline.backend.pipeline import PipeLine
pipeline_upload = PipeLine().set_initiator(role='guest',
party_id=9999).set_roles(guest=9999, host=10000)
```

定义数据存储分区

```
partition = 4
```

定义表名和命名空间，这将在 FATE 作业配置中使用

```
dense_data_guest = {"name": "breast_hetero_guest", "namespace": f"experiment"}
dense_data_host = {"name": "breast_hetero_host", "namespace": f"experiment"}
tag_data = {"name": "breast_hetero_host", "namespace": f"experiment"}
```

添加要上传的数据(注意：这里只会上传到guest，之后host读到的数据是之前DSL配置阶段上传的数据，理论上这里应该guest和host分别上传自己的数据，这里图方便就全上传到guest了；正常上传流程参考IMDB情感分类部分的上传)

```
import os
data_base = "workspace/FATE/" #根据工作目录灵活修改
pipeline_upload.add_upload_data(file=os.path.join(data_base,
"examples/data/breast_hetero_guest.csv"),
                                table_name=dense_data_guest["name"],             # table
name
                                namespace=dense_data_guest["namespace"],         #
namespace
                                head=1, partition=partition)               # data info

pipeline_upload.add_upload_data(file=os.path.join(data_base,
"examples/data/breast_hetero_host.csv"),
                                table_name=dense_data_host["name"],
                                namespace=dense_data_host["namespace"],
                                head=1, partition=partition)

pipeline_upload.add_upload_data(file=os.path.join(data_base,
"examples/data/breast_hetero_host.csv"),
                                table_name=tag_data["name"],
                                namespace=tag_data["namespace"],
                                head=1, partition=partition)
```

执行数据上传任务

```
pipeline_upload.upload(drop=1)
```

使用Pipeline完成secureboost训练与测试

完成数据上传后，使用Pipeline发布训练任务

引入Pipeline及相关组件库

```
from pipeline.backend.pipeline import PipeLine
from pipeline.component import Reader, DataTransform, Intersection, HeteroSecureBoost,
Evaluation
from pipeline.interface import Data
#创建实例
pipeline = PipeLine() \
        .set_initiator(role='guest', party_id=9999) \
        .set_roles(guest=9999, host=10000)
```

定义Reader组件加载数据

```
reader_0 = Reader(name="reader_0")
# set guest parameter
reader_0.get_party_instance(role='guest', party_id=9999).component_param(
    table={"name": "breast_hetero_guest", "namespace": "experiment"})
# set host parameter
reader_0.get_party_instance(role='host', party_id=10000).component_param(
    table={"name": "breast_hetero_host", "namespace": "experiment"})
```

添加 DataTransform 组件解析原始数据到数据实例中

添加组件到pipeline构建模型并编译

```
pipeline.add_component(reader_0)
pipeline.add_component(data_transform_0, data=Data(data=reader_0.output.data))
pipeline.add_component(intersect_0, data=Data(data=data_transform_0.output.data))
pipeline.add_component(hetero_secureboost_0,
data=Data(train_data=intersect_0.output.data))
pipeline.add_component(evaluation_0, data=Data(data=hetero_secureboost_0.output.data))
pipeline.compile();
```

提交训练任务

```
pipeline.fit()
```

指导教师签字		日期	年　月　日

第 3—4 周记录：

1. 基于 Pipeline 实现横向联邦 LSTM 完成 IMDB 文本情感分类

本样例是在FATE框架上，使用keras风格搭建**LSTM长短期神经网络**模型，完成**IMDB数据集**上的文本情感分类任务。任务类型为**文本二分类**任务。主要包含以下流程：

1. 文本数据预处理
2. 基于Pipeline完成IMDB数据上传
3. 基于Pipeline构建LSTM模型并发布训练任务
4. 模型训练效果查看与评估

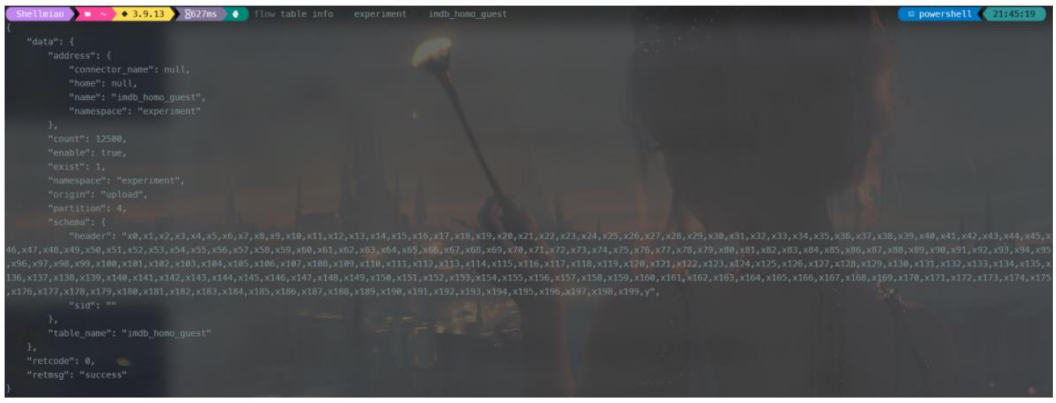
[LSTM字符粒度下文本序列生成学习与实践](https://md.shellmiao.com/s/QCKFa9hFl)

1. IMDB 文本数据预处理
    1. 添加需要用到的库文件(Tensorflow-gpu 2.10.1)
    2. 读入训练数据和测试数据
    3. 使用 tokenizer 将数据中的每个单词转化为整数值的唯一映射
    4. 将数据集按联邦节点划分并输出到 csv 文件
2. 基于 Pipeline 上传 IMDB 数据
3. 使用 Fate-cli 查看状态（一种 Debug 方式）

    [Fate-cli使用手册](https://fate.readthedocs.io/en/develop-1.5/_build_temp/python/fate_client/flow_client/README.html)

    ```
    flow init --ip 192.168.160.139 --port 9380
    ```

    ```
    flow table info -n experiment -t imdb_homo_guest
    ```

    上传后的数据表名为：

    `imdb_homo_guest,imdb_homo_host,imdb_homo_test_guest,imdb_homo_test_host`

    可分别对Guest、Host验证

4. 基于 Pipeline 构建 LSTM 模型并发布训练任务

基础配置

```
from pipeline.backend.pipeline import PipeLine
from pipeline.component import DataTransform
from pipeline.component import Reader
from pipeline.component import HomoNN
from pipeline.interface import Data
from pipeline.component import Evaluation
import os
os.environ["CUDA_VISIBLE_DEVICES"]="-1"

pipeline = PipeLine() \
        .set_initiator(role='guest', party_id=9999) \
        .set_roles(guest=9999, host=[10000], arbiter=10000)
# 别忘了把pipeline切换回guest
```

数据读入组件

```
reader_0 = Reader(name="reader_0")
reader_0.get_party_instance(role='guest', party_id=9999).component_param(
    table={"name": "imdb_homo_guest", "namespace": "experiment"})
reader_0.get_party_instance(role='host', party_id=10000).component_param(
    table={"name": "imdb_homo_host", "namespace": "experiment"})

data_transform_0 = DataTransform(name="data_transform_0", with_label=True)
data_transform_0.get_party_instance(role='guest', party_id=9999).component_param(
    with_label=True,label_name="y")
data_transform_0.get_party_instance(role='host', party_id=[10000]).component_param(
    with_label=True,label_name="y")
```

构建LSTM模型

词嵌入层设置128个神经元，LSTM层包含64个神经元；最大迭代次数为100，batch长度为32，设置early_stop机制。使用Adam进行梯度更新，学习率为1e-5，损失函数为二元交叉熵

```
max_features = 10000
max_len = 200
embedding_neurons = 128
lstm_neurons = 64

homo_nn_0 = HomoNN(
        name="homo_nn_0",
#         encode_label=True,
        max_iter=100,
        batch_size=32,
        early_stop={"early_stop": "diff", "eps": 0.0001},)

from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import Embedding
from tensorflow.keras.layers import Reshape
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import LSTM
```

```
homo_nn_0.add(Embedding(max_features,embedding_neurons,input_length=max_len))
homo_nn_0.add(BatchNormalization())
homo_nn_0.add(LSTM(units=lstm_neurons, dropout=0.2 ,recurrent_dropout=0.2))
homo_nn_0.add(Dropout(0.5))
homo_nn_0.add(Dense(units=1,activation='sigmoid'))

from tensorflow.keras import optimizers
homo_nn_0.compile(
    optimizer=optimizers.Adam(learning_rate=0.00001),
    metrics=["accuracy", "AUC"],
    loss="binary_crossentropy")
```

完成组件搭建和任务发布

```
evaluation_0 = Evaluation(name="evaluation_0", eval_type="binary")

pipeline.add_component(reader_0)
pipeline.add_component(data_transform_0, data=Data(data=reader_0.output.data))
pipeline.add_component(homo_nn_0, data=Data(train_data=data_transform_0.output.data))
pipeline.add_component(evaluation_0,data=Data(homo_nn_0.output.data))
pipeline.compile();

pipeline.fit()
```

5. 遇到的问题解决方案

no job could be found问题

问题：使用本地pipeline发布任务，运行时报错 `{'data': [], 'retcode': 0, 'retmsg': 'no job could be found'}`

解决：pipeline init的ip和代码init的ip不一样

tensorflow组件缺失

在执行神经网络，使用tensorflow相关组件构建算法模型，报错 `No module named tensorflow` ,是因为fate自身部署的python环境缺乏tensorflow，在每个节点使用如下命令进入python容器。

```
docker exec -it confs-xxxx_python_1 bash
```

使用pip安装适合版本的tensorflow即可，这里使用1.15.0版本

```
pip install tensorflow==1.15.0 -i http://mirrors.aliyun.com/pypi/simple/ --trusted-host mirrors.aliyun.com
```

2. 基于 Pipeline 实现横向联邦 CNN 完成中文文本主题分类

正常的 CNN 完成中文文本主题分类 学习记录可以参考如下这篇博客：

[[NLP]基于 CNN 对 THUCNews 数据集进行文本分类](https://md.shellmiao.com/s/aLxE1btif)

1. 数据预处理
2. 分别上传至 guest、host 主机
3. 训练模型

```
from pipeline.backend.pipeline import PipeLine
from pipeline.component import DataTransform
from pipeline.component import Reader
from pipeline.component import HomoNN
from pipeline.interface import Data
from pipeline.component import Evaluation
import os
os.environ["CUDA_VISIBLE_DEVICES"]="-1"

pipeline = PipeLine() \
        .set_initiator(role='guest', party_id=9999) \
        .set_roles(guest=9999, host=[10000], arbiter=10000)
```

定义reader、data_transform

```
reader_0 = Reader(name="reader_0")
reader_0.get_party_instance(role='guest', party_id=9999).component_param(
    table={"name": "thucnews_guest", "namespace": "experiment"})
reader_0.get_party_instance(role='host', party_id=10000).component_param(
    table={"name": "thucnews_host", "namespace": "experiment"})

data_transform_0 = DataTransform(name="data_transform_0", with_label=True)
data_transform_0.get_party_instance(role='guest', party_id=9999).component_param(
    with_label=True,label_name="y")
data_transform_0.get_party_instance(role='host', party_id=[10000]).component_param(
    with_label=True,label_name="y")
```

4. 遇到的问题与解决方法

```
TypeError: ('Keyword argument not understood:', 'groups')
```

这是tensorflow和keras版本低了导致的，需要fate的python docker里面的包升级了一下

```
# 进入docker
docker exec -it confs-xxxx_client_1 bash
# 更新包
pip install --upgrade tensorflow -i https://pypi.mirrors.ustc.edu.cn/simple/
```

指导教师签字		日期	年　月　日

注：每 2 周指导内容记录在一个表格中，双面打印。