智能物联网技术实验设计

联邦学习在物联网中的应用

任课教师：张盛
助教：杨丹妮
2023. 10. 24

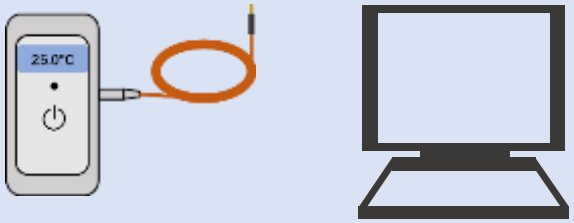
目录

CONTENTS

1 实验任务

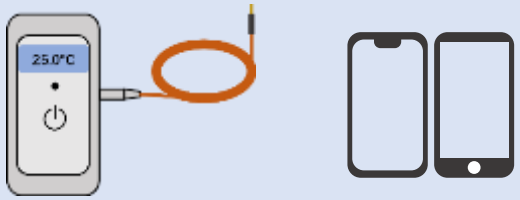
实验任务

边缘设备1



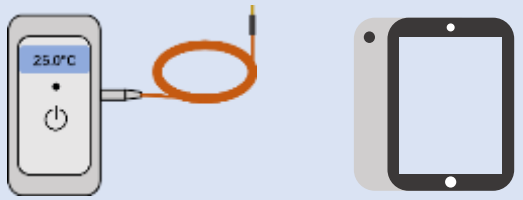
智能温度采集器　计算设备

边缘设备k

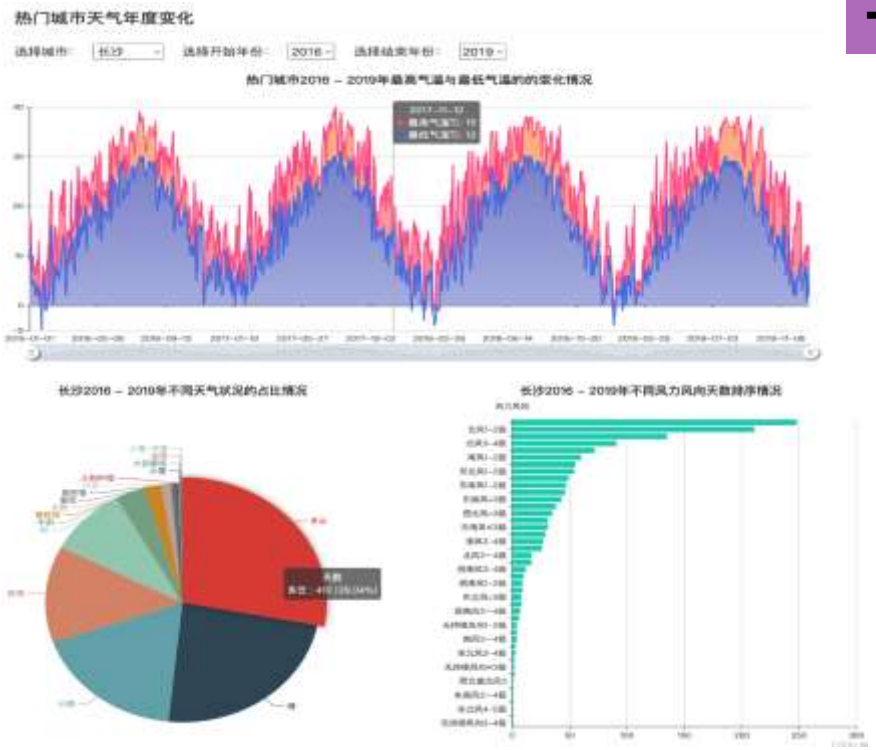


智能温度采集器　计算设备

边缘设备K



智能温度采集器　计算设备



一个简单的物联网系统

传统物联网系统：物联网中的数据搬移到一个云服务器计算。（数据量大，计算慢，隐私泄露）

联邦学习系统：数据不动，模型动原则

实验任务

将联邦学习应用到物联网系统中完成气温的分析

- 读取采集的气温数据、数据处理、数据可视化
- 搭建一个联邦学习框架用于气温数据分析
- 完成过去气温拟合和未来气温预测

2 实验内容

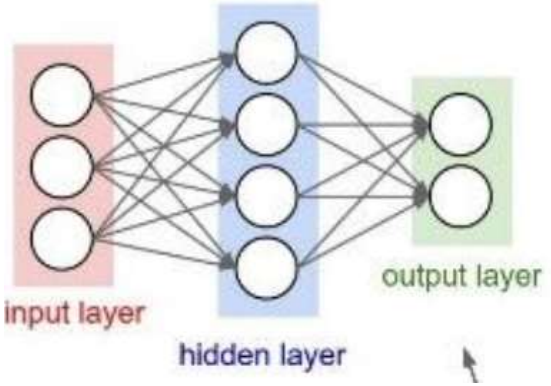
实验内容

将神经网络作为基本模型进行气温预测，构建一个联邦学习系统，完成过去气温拟合和未来气温预测。

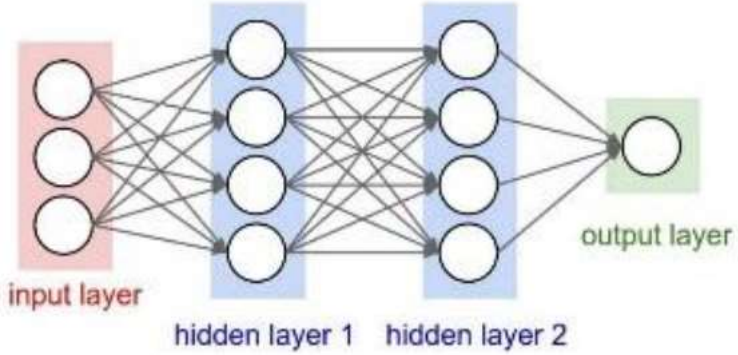
➢ 神经网络模型的气温预测

用pytorch/tensorflow框架，搭建一个简单的神经网络，作为联邦学习中的模型。可以完成输入特征向量、前向传播、反向传播、输出预测结果、根据损失完成梯度更新。

- 模型越简单越好，复杂的模型容易过拟合且不能充分体现联邦学习过程的作用，建议采用两个全连接层的结构，隐藏层节点数尽量少。
- 自定义优化器，可以选择Adam、SGD等等。
- 根据任务选择合适的损失函数，拟合任务->均方误差（MSE）
- 代码实现形式不做要求，都可以用封装好的API来实现。例如torch.nn.MSELoss



两个全连接层的网络

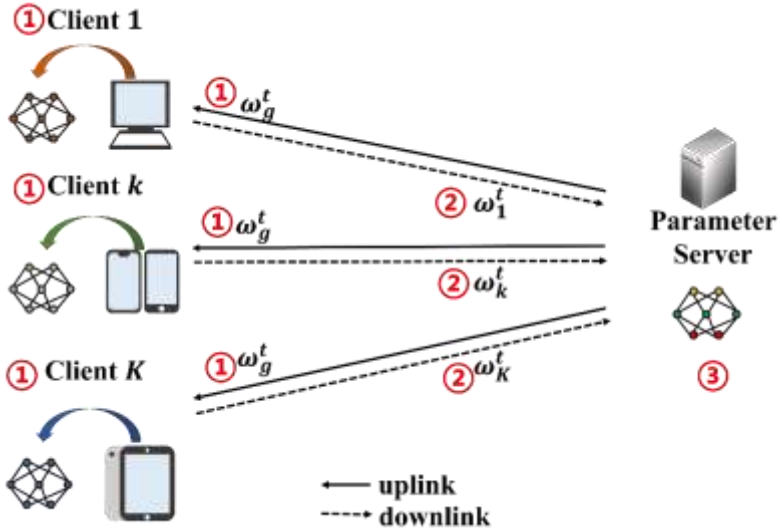


三个全连接层的网络

实验内容

➢ **一个联邦学习框架**

组成：在一个物联网系统中，包括一个参数服务器（PS）和一组K个节点的边缘设备，边缘设备从PS处获得一个共享的全局模型，在其本地数据集上开始训练。包含了用于联邦学习的d_k个数据样本。所有设备数据样本的总大小为$D = \sum_{k=1}^{K} d_k$。



S1-S3重复T个全局轮，得到最终的全局模型

S1：本地模型计算 ①

在第t个全局轮，PS下发全局模型。边缘设备根据这个全局模型和它的本地数据集，进行本地模型计算。这个过程就是以搭建好的神经网络模型为基础的计算过程。w_g^t是t个全局轮下发的全局模型，边缘端根据实验中设定的本地模型训练次数完成后，得到w_k^t是t个全局轮边缘设备最终更新的神经网络。

S2：本地模型通信（仿真实验中省略）②

所有边缘设备将它更新好的本地模型上传到PS端。

S3：本地模型聚合 ③

$$w_g^{t+1} = \sum_k \alpha_k^t w_k^t$$

PS对收到的本地模型参数进行聚合，采用加权平均聚合策略，聚合权重α_k^t采用样本数量加权。即$\alpha_k = \frac{d_k}{D}$。因此得到新的全局模型参数w_g^{t+1}。实验中需要设定的参数：训练的全局轮T，边缘端在本地训练的轮数e，客户端数量K

3 实验步骤

实验步骤

1.边缘端获取并处理实验数据。

实验已经准备了采集到2013-2022年某地区的最高气温数据。共有3480条。已经将其划分为训练集（3000条）和测试集（480条）。数据结构如右表。

year,month,day,week	具体的时间（年、月、日、周）
day_2	前天的最高温度值
day_1	昨天的最高温度值
average	历史中，每年这一天的平均最高温度值
actual	当天的真实最高温度，即拟合的标签值

	A	B	C	D	E	F	G	H
1	year	month	day	week	day_2	day_1	average	actual
2	2013	1	1	Fri	45	45	45.6	45
3	2013	1	2	Sat	44	45	45.7	44
4	2013	1	3	Sun	45	44	45.8	41
5	2013	1	4	Mon	44	41	45.9	40
6	2013	1	5	Tues	41	40	46	44
7	2013	1	6	Wed	40	44	46.1	51
8	2013	1	7	Thurs	44	51	46.2	45
9	2013	1	8	Fri	51	45	46.3	48
10	2013	1	9	Sat	45	48	46.4	50
11	2013	1	10	Sun	48	50	46.5	52
12	2013	1	11	Mon	50	52	46.7	45
13	2013	1	12	Tues	52	45	46.8	49
14	2013	1	13	Wed	45	49	46.9	55
15	2013	1	14	Thurs	49	55	47	49
16	2013	1	15	Fri	55	49	47.1	48
17	2013	1	16	Sat	49	48	47.3	54
18	2013	1	17	Sun	48	54	47.4	50
19	2013	1	18	Mon	54	50	47.5	54
20	2013	1	19	Tues	50	54	47.6	48
21	2013	1	20	Wed	54	48	47.7	52
22	2013	1	21	Thurs	48	52	47.8	52
23	2013	1	22	Fri	52	52	47.9	57
24	2013	1	23	Sat	52	57	48	48
25	2013	1	24	Sun	57	48	48.1	51

S1：划分样本的特征和标签

实验中actual一列为标签值，是需要拟合的数据。在实验数据中将其分割出来，得到剩余部分作为样本特征。

S2：实验数据预处理

- **数据编码。week这一列无法在训练中进行数值计算，需要把week进行编码。（独热编码）00001-10000**
- **数据标准化。sklearn.preprocessing包提供了几个常用的实用函数和转换器类。数据标准化为网络输入数据。**

S3：边缘端数据划分

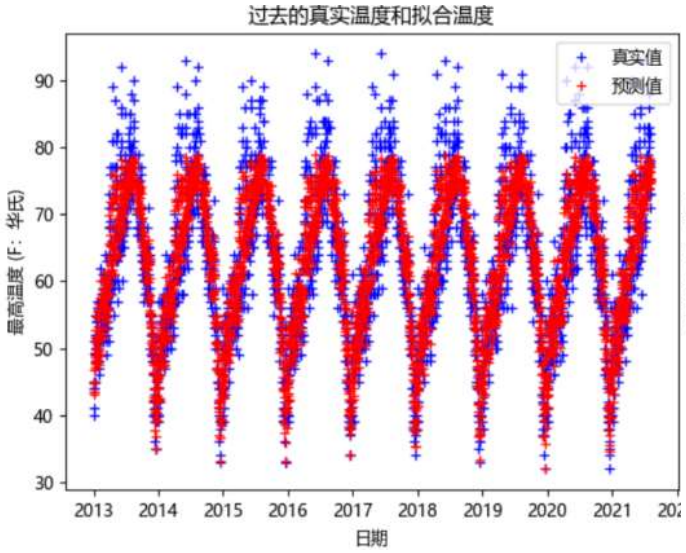
构建K个边缘端，将训练集中的样本打乱顺序**平均分配给**他们。（**仿真实验必须**，真实的FL场景中不进行数据划分。）

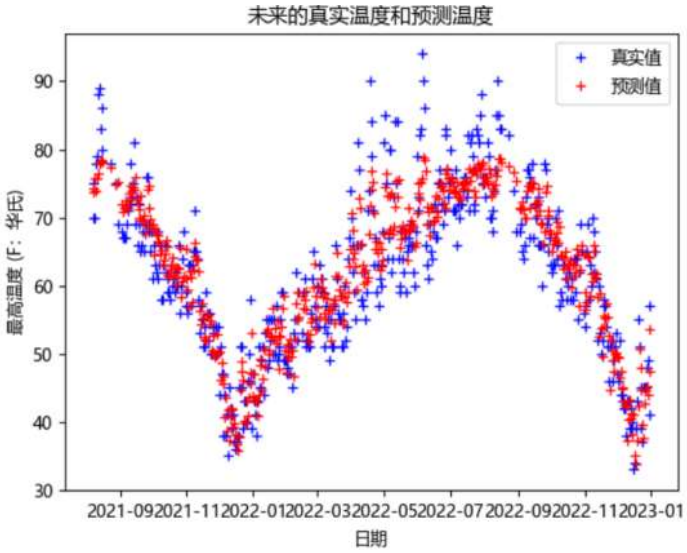
```
原始数据维度：(3480, 8)，数据：
    year  month  day  week  temp_2  temp_1  average  actual
0   2013      1    1   Fri      45      45     45.6      45
1   2013      1    2   Sat      44      45     45.7      44
2   2013      1    3   Sun      45      44     45.8      41
3   2013      1    4   Mon      44      41     45.9      40
4   2013      1    5  Tues      41      40     46.0      44

标准化原始数据，维度：(3000, 13) 具体数据：
[[-1.5666989  -1.5678393  -1.65682171 ... -0.40482045 -0.41913682
  -0.40482045]
 [-1.5666989  -1.5678393  -1.54267126 ... -0.40482045 -0.41913682
  -0.40482045]
 [-1.5666989  -1.5678393  -1.4285208  ... -0.40482045 -0.41913682
  -0.40482045]
 ...
 [ 1.21854359  0.43596791 -1.31437034 ...  2.47023092 -0.41913682
  -0.40482045]
 [ 1.21854359  0.43596791 -1.20021989 ... -0.40482045 -0.41913682
  -0.40482045]
 [ 1.21854359  0.43596791 -1.08606943 ... -0.40482045 -0.41913682
```

2.搭建联邦学习框架

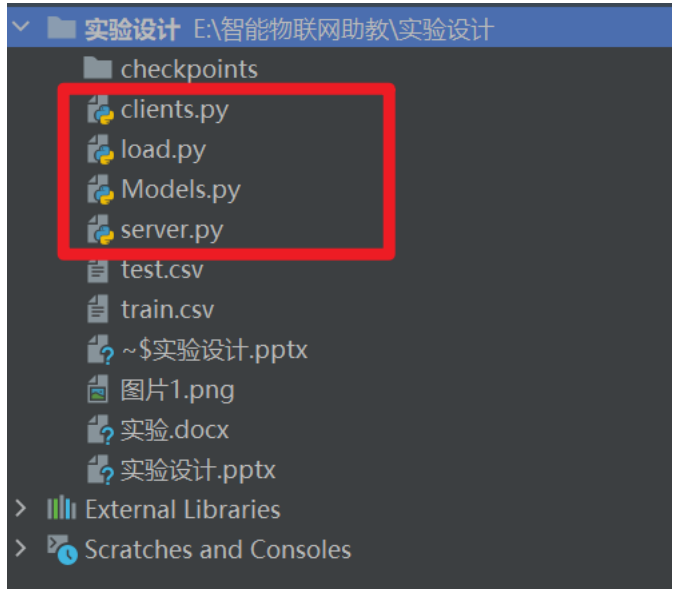
在电脑上仿真一个前述的联邦学习框架，完成全局模型下发、本地模型上传、本地模型聚合的过程，经过T个全局轮后得到最终的全局模型，即联邦学习的输出，用框架得到过去最高气温（训练数据）的拟合结果和未来最高气温的预测结果（测试数据）。绘制实验结果图如下：





4 代码参考

代码参考



4个.py文件

- client 客户端
- server 参数服务器
- Model 神经网络
- load 数据处理及分配

2个.csv文件

- train 训练集
- test 测试集

调用关系

- server->load、client、model
- client->load

model.py

- 构建神经网络

```
import torch.nn as nn
import torch.nn.functional as F

class tiangi_2NN(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(13, [redacted])
        self.fc2 = nn.Linear([redacted], 1)

    def forward(self, inputs):
        tensor = F.sigmoid(self.fc1(inputs))
        tensor = self.fc2(tensor)
        return tensor
```

具体功能

load.py

- 读取train.csv, test.csv
- 划分样本的特征和标签
- 数据编码
- 数据标准化
- 边缘端数据划分

```
# 标签    也就要预测的温度的真实值
all_labels_train = np.array(features_train['actual'])
```

```
# 在特征中去掉标签
features_train = features_train.drop('actual', axis=1)
```

```
# 独热编码    将week中的Fri、Sun等编码而不是String格式
features_train = pd.get_dummies(features_train)
```

```
all_features_train = preprocessing.StandardScaler().fit_transform(features_train)
```

```
def getdata(class_num):
```

返回划分给客户端的数据索引

可能需要的库

```
import numpy as np
import pandas as pd
from sklearn import preprocessing
import warnings
```

代码参考

client.py

- 构造边缘端集合类

```
class ClientsGroup(object):
    def __init__(self, dev, class_num):
        self.dev = dev
        self.clients_set = {}
        self.class_num = class_num
        self.dataSetBalanceAllocation()
```

- 初始化集合的内容

```
def dataSetBalanceAllocation(self):
    index_class = getdata(self.class_num)
```

- 边缘端数据划分

```
for i, idcs in enumerate(index_class):
    local_label, local_data = np.vstack(train_labels_shuffle[idcs]), np.vstack(train_features_shuffle[idcs])
    num_example = len(local_label)
    someone_1 = client(TensorDataset(torch.tensor(local_data, dtype=torch.float, requires_grad=True), torch.te

    self.clients_set['client{}'.format(i)] = someone_1
```

- 构造每个边缘端类

```
class client(object):
    def __init__(self, trainDataSet, dev, num_example):
        self.train_ds = trainDataSet
        self.dev = dev
        self.train_dl = None
        self.num_example = num_example
        self.state = {}
```

- 本地计算函数

```
def localUpdate(self, localBatchSize, localepoch, Net, lossFun, opti, global_parameters):
    Net.load_state_dict(global_parameters, strict=True)
    self.train_dl = DataLoader(self.train_ds, batch_size=localBatchSize, shuffle=True)
    for epoch in range(localepoch):
```

可能需要的库

```
import numpy as np
import torch
from torch.utils.data import TensorDataset
from torch.utils.data import DataLoader
from load import *
```

代码参考

server.py

- 设定全局的参数---客户端数量、全局轮数、每个边缘端训练轮数localepoch、batchsize、learning rate

```
net = tianqi_2NN()
if torch.cuda.device_count() > 1:
    print("Let's use", torch.cuda.device_count(), "GPUs!")
    net = torch.nn.DataParallel(net)
net = net.to(dev)
```

- 实例化神经网络

- 确定损失函数、优化器

```
loss_func = torch.nn.MSELoss(reduction='mean')
opti = optim.SGD(net.parameters(), lr=args['learning_rate'])
```

- 实例化边缘端集合对象

```
myClients = ClientsGroup(dev, args['num_of_clients'])
```

- 边缘端计算

```
for i in range(1, args['num_comm']+1):
    print('-----------------------------------------------fedavg-----------------------------------------------')
    print('-----------------------------------------------第', i, '轮训练-----------------------------------------------')
    sum_parameters = None
    for client in tqdm(clients_in_comm_100):
        print(client)
        local_parameters = myClients.clients_set[client].localUpdate(args['batchsize'], args['epoch'], net,
                                                                     loss_func, opti, global_parameters)
```

- 本地模型聚合

```
        for var in sum_parameters:
            sum_parameters[var] = sum_parameters[var] + local_parameters[var] * example
for var in global_parameters:
    global_parameters[var] = sum_parameters[var] / sum_example
```

- 验证拟合和预测结果

```
x = torch.tensor(all_features_test, dtype=torch.float)
x = x.to(dev)
# y = torch.tensor(test_labels, dtype=torch.float)
predict_1 = net(x)
predict = predict_1.cpu().detach().numpy()
```

可能需要的库

```
import os
import argparse
from tqdm import tqdm
from torch import optim
from clients import ClientsGroup
import sys
from Models import tianqi_2NN
from load import *
```

5 实验形式及要求

形式及要求

2人一组完成实验，一组一份实验报告、PPT，在网络学堂提交代码和报告。

实验报告中说明参数设定：训练的全局轮T（建议不小于10），边缘端在本地训练的轮数e（建议不大于5），客户端数量（建议不小于10，为10的倍数），每个客户端的样本数量，神经网络模型结构（网络大小，batchsize，学习率等）。

实验要求：对实验过程和结果进行形象的展示，包括流程的图形化和数据的图形化（说得越清楚分越高）

参考资料

神经网络、pytorch、tensorflow	https://zhuanlan.zhihu.com/p/65472471
	https://www.bilibili.com/video/BV1hE411t7RN?p=21&vd_source=e4fec414faa886cd34b97db4421035e9
	https://www.bilibili.com/video/BV1Wv411h7kN/?spm_id_from=333.337.search-card.all.click&vd_source=e4fec414faa886cd34b97db4421035e9
联邦学习	https://zhuanlan.zhihu.com/p/619721874
	https://blog.csdn.net/qq_36018871/article/details/121361027
	https://blog.csdn.net/weixin_47754029/article/details/128803590
数据处理	https://zhuanlan.zhihu.com/p/248452034
	https://zhuanlan.zhihu.com/p/134495345

实验软件：pycharm、anaconda

实验环境：python、pytorch、tensorflow

以及前述库环境