隐私计算线上慕课【第18讲】

隐语明密文混合编程实践

周爱辉 | 蚂蚁集团隐私计算技术部

目 录

Contents

隐语架构

01

隐语架构


用户界面
可视化操作界面
开放编程接口
AI & BI
隐私算法
多方安全计算
联邦学习
可信执行环境
明密文
混合调度
设备计算图
分布式调度引擎
明密文
计算设备与原语
密文计算设备
MPC设备
HE设备
TEE设备
TECC设备
明文计算设备
Python
解释器
SQL
执行环境
隐私保护原语
差分隐私
脱敏
资源管理
数据管理
计算管理
网络管理

安装和初始化

02

安装和初始化

安装要求

- Python == 3.8
- 操作系统：
    1. CentOS 7
    2. Ubuntu 18.04/20.04
    3. macOS 11.1+（intel芯片）
- 资源：≥ 8核16GB

安装和初始化

安装方式一：使用docker镜像

```
docker run -it secretflow/secretflow-anolis8:latest
```

安装和初始化

安装方式二：通过pypi

pip install -U secretflow

提示:

1. 要求pip >= 19.3
2. 建议使用conda管理python环境

安装和初始化

初始化

```
import secretflow as sf

# 定义alice、bob、carol三个参与方。
parties = ['alice', 'bob', 'carol']


sf.init(parties, num_cpus=16)
```

设备

03

设备

SECRET FLOW 隐语

PYU	SPU	HEU
明文设备	MPC密态设备	同态加密设备

设备

PYU - 明文设备

- 以python为底座，可以执行python代码
- 参与方本地计算

设备

构造PYU

```
import secretflow as sf

# 创建PYU实例
alice_pyu = sf.PYU('alice’)
bob_pyu = sf.PYU('bob')
carol_pyu = sf.PYU('carol')
```

执行代码

```
import pandas as pd

def load_data(file):
    return pd.read_csv(file).values

alice_arr: sf.PYUObject = alice_pyu(load_data)('alice.csv')
bob_arr: sf.PYUObject = bob_pyu(load_data)('bob.csv')
```

PYU数据对象

设备

PYU

明文设备

SPU

MPC密态设备

HEU

同态加密设备

设备

SPU - MPC密态设备

- 由多个物理设备构成的虚拟设备
- 使用MPC协议作为后端
- 使用XLA作为中间层表示
- 使用JAX作为前端语言

设备

构造SPU

```
import secretflow as sf
import spu

# 创建一个基于ABY3协议的SPU实例
cluster_def = {
        'nodes': [
                {‘party’: ‘alice’, ‘id’: ‘local:0’, ‘address’: ‘127.0.0.1:10001},
                {'party': 'bob', 'id': 'local:1', 'address': '127.0.0.1:10002’},
                {'party': 'carol', 'id': 'local:2', 'address': '127.0.0.1:10003'},
        ],
        'runtime_config': {'protocol': spu.spu_pb2.ABY3,
                        'field': spu.spu_pb2.FM64},
}
spu_device = sf.SPU(cluster_def=cluster_def)
```

执行jax

SPU数据对象

```
import jax.numpy as jnp

alice_arr_spu: sf.SPUObject = alice_arr.to(spu_device)
bob_arr_spu: sf.SPUObject = bob_arr.to(spu_device)
# jax.numpy.average(a, axis=None, weights=None, ...)
spu_object: sf.SPUObject = spu(jnp.average)(
   jnp.array([alice_arr_spu, bob_arr_spu]), axis=0
)
```

设备

PYU

明文设备

SPU

MPC密态设备

HEU

同态加密设备

设备

HEU - 同态加密设备

- 内置同态加密算法
- 提供numpy接口

设备

构造HEU

```
import secretflow as sf
import spu

# 创建HEU实例
heu_config = {
    'sk_keeper': {'party': 'alice'},
    'evaluators': [{'party': 'bob'}],
    'mode': 'PHEU',
    'he_parameters': {
        'schema': 'paillier',
        'key_pair': {'generate': {'bit_size': 2048}},
    },
}
heu_device = sf.HEU(heu_config, spu.spu_pb2.FM128)
```

执行计算

HEU数据对象

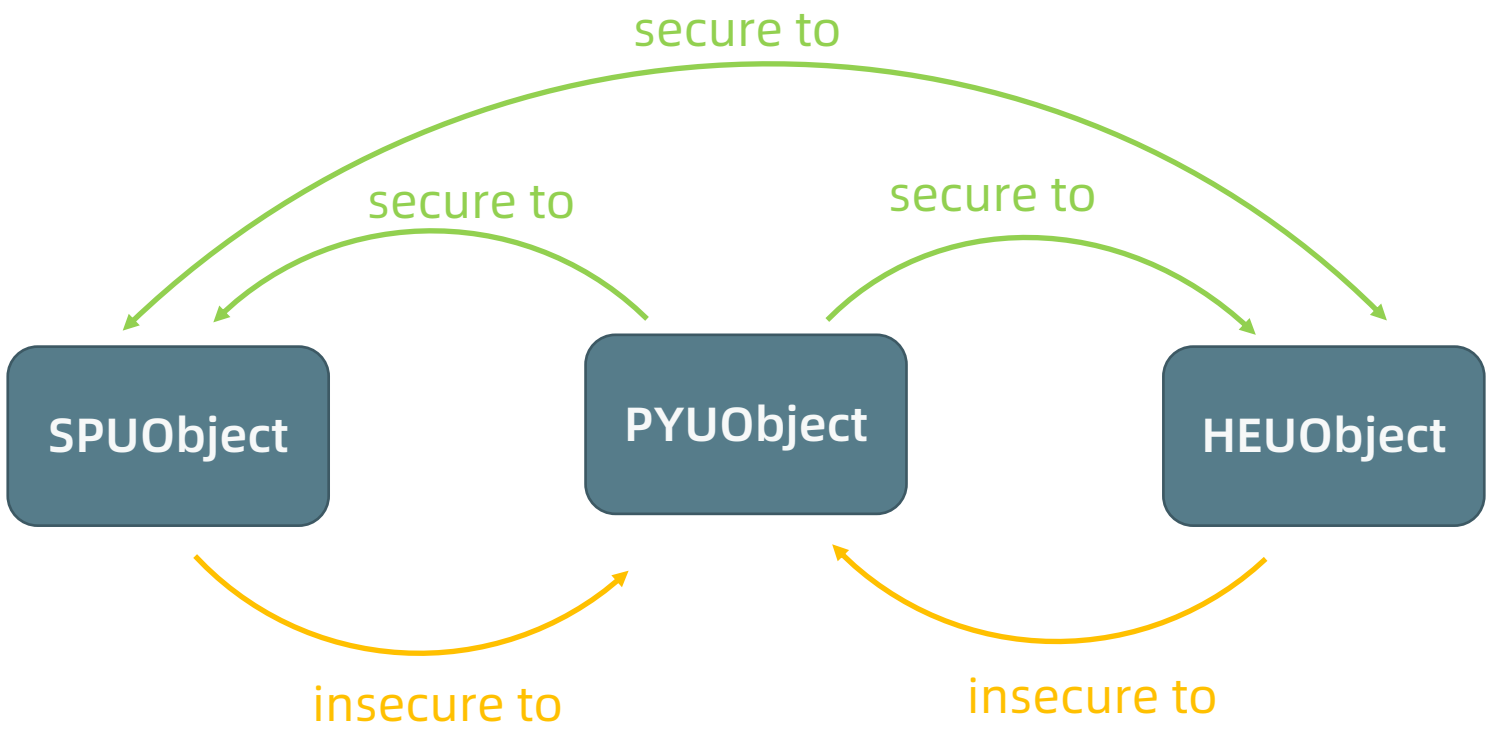
```
import numpy as np


x = alice_pyu(np.random.rand)(3, 4)
y = bob_pyu(np.random.rand)(3, 4)
x_, y_ = x.to(heu_device), y.to(heu_device)

z: sf.HEUObject = x_ + y_
```

设备

to：设备之间数据流转



算法

04

算法

SECRET FLOW 隐语

PSI	SS LR	联邦NN
隐私求交	秘密分享逻辑回归	水平联邦深度学习

算法 - PSI

1. 准备数据集
2. 求交集

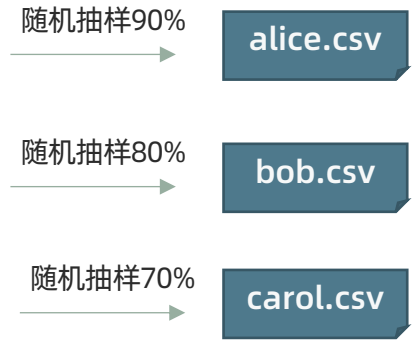
```
import numpy as np
from sklearn.datasets import load_iris

data, target = load_iris(return_X_y=True, as_frame=True)
# 添加uid列用于求交
data['uid'] = np.arange(len(data)).astype('str')

# 随机抽样模拟产生三份随机未对齐的数据
da, db, dc = data.sample(frac=0.9), data.sample(frac=0.8), data.sample(frac=0.7)

da.to_csv('alice.csv', index=False)
db.to_csv('bob.csv', index=False)
dc.to_csv('carol.csv', index=False)
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	uid
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	1
2	4.7	3.2	1.3	0.2	2
3	4.6	3.1	1.5	0.2	3
4	5.0	3.6	1.4	0.2	4
...	...	...	...	...	...



算法 - PSI

1. 准备数据集
2. 求交集

```
input_path = {alice: 'alice.csv', bob: 'bob.csv', carol: 'carol.csv'}
output_path = {
        alice: 'alice_psi.csv',
        bob: 'bob_psi.csv',
        carol: 'carol_psi.csv’,
}

spu_device.psi_csv(
        key='uid',
        input_path=input_path,
        output_path=output_path,
        receiver='alice',
        protocol='ECDH_PSI_3PC',
)
```

https://secretflow.readthedocs.io/zh_CN/latest/components/psi.html#

算法

PSI

隐私求交

SS LR

秘密分享逻辑回归

联邦NN

水平联邦深度学习

算法 - SS LR

基于梯度下降的逻辑回归

预测 $$y^{'} = \frac{1}{1 + e^{-(w^T x + b)}}$$

损失函数 $$J(\theta) = \sum -ylog(y^{'}) - (1-y)log(1-y^{'})$$

参数更新 $$\theta = \theta - \eta \frac{\partial J(\theta)}{\partial \theta}$$

算法 - SS LR

1. 准备数据
2. 定义相关方法
3. 执行训练
4. 查看训练效果

以经典的breast cancer为例

```
import jax.numpy as jnp
import numpy as np
from sklearn.datasets import load_breast_cancer
from sklearn.preprocessing import Normalizer

def breast_cancer(with_y = False):
    scaler = Normalizer(norm='max')
    x, y = load_breast_cancer(return_X_y=True)
    x = scaler.fit_transform(x)
    if with_y:
        return x[:, :15], y
    else:
        return x[:, 15:], _

x_a, y = alice_pyu(breast_cancer, num_returns=2)(with_y=True)
x_b, _ = bob_pyu(breast_cancer, num_returns=2)(with_y=False)
W = sf.to(spu_device, jnp.zeros((30,)))
b = sf.to(spu_device, 0.0)
x_a_, x_b_, y_ = (
    x_a.to(spu_device),
    x_b.to(spu_device),
    y.to(spu_device),
)
```

算法 - SS LR

1. 准备数据
2. 定义相关方法
3. 执行训练
4. 查看训练效果

```
import jax.numpy as jnp
from jax import value_and_grad


def sigmoid(x):
    return 1 / (1 + jnp.exp(-x))


def predict(W, b, inputs):
    return sigmoid(jnp.dot(inputs, W) + b)


def loss(W, b, inputs, targets):
    preds = predict(W, b, inputs)
    label_probs = preds * targets + (1 - preds) * (1 - targets)
    return -jnp.mean(jnp.log(label_probs))


def train_step(W, b, x1, x2, y, learning_rate):
    x = jnp.concatenate([x1, x2], axis=1)
    loss_value, Wb_grad = value_and_grad(loss, (0, 1))(W, b, x, y)
    W -= learning_rate * Wb_grad[0]
    b -= learning_rate * Wb_grad[1]
    return loss_value, W, b
```

算法 - SS LR

1. 准备数据
2. 定义相关方法
3. 执行训练
4. 查看训练效果

```
def fit(W, b, x1, x2, y, epochs=1, learning_rate=1e-2):
    losses = jnp.array([])
    for _ in range(epochs):
        l, W, b = train_step(W, b, x1, x2, y, learning_rate=learning_rate)
        losses = jnp.append(losses, l)
    return losses, W, b


losses, W_, b_ = spu_device(
    fit,
    static_argnames=['epochs'],
    num_returns_policy=sf.device.SPUCompilerNumReturnsPolicy.FROM_USER,
    user_specified_num_returns=3,
)(W, b, x_a_, x_b_, y_, epochs=10, learning_rate=1e-2)
```

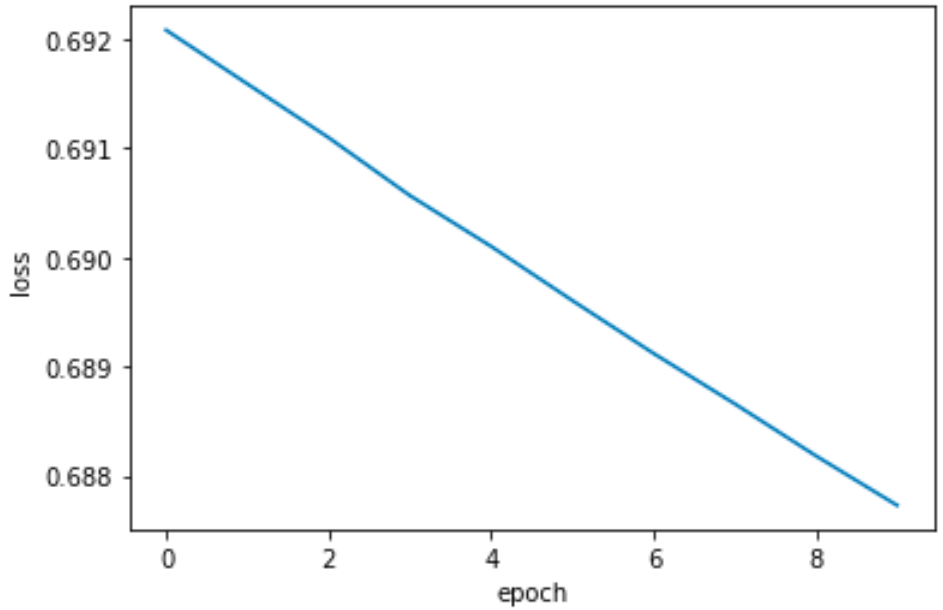
算法 - SS LR

1. 准备数据
2. 定义相关方法
3. 执行训练
4. 查看训练效果

```
import matplotlib.pyplot as plt


def plot_losses(losses):
plt.plot(np.arange(len(losses)), losses)
plt.xlabel('epoch')
plt.ylabel('loss’)

losses = sf.reveal(losses)
plot_losses(losses)
```



算法 - SS LR

SecretFlow内置SS LR算法

```
from secretflow.data.vertical import VDataFrame
from secretflow.ml.linear.ss_sgd import SSRegression

x = VDataFrame({alice_pyu: x_a, bob_pyu: x_b})
y = VDataFrame({alice_pyu: y})

model = SSRegression(spu=spu_device)
model.fit(
    x=x,
    y=y,
    epochs=5,
    learning_rate=1e-2,
    batch_size=32
)
```

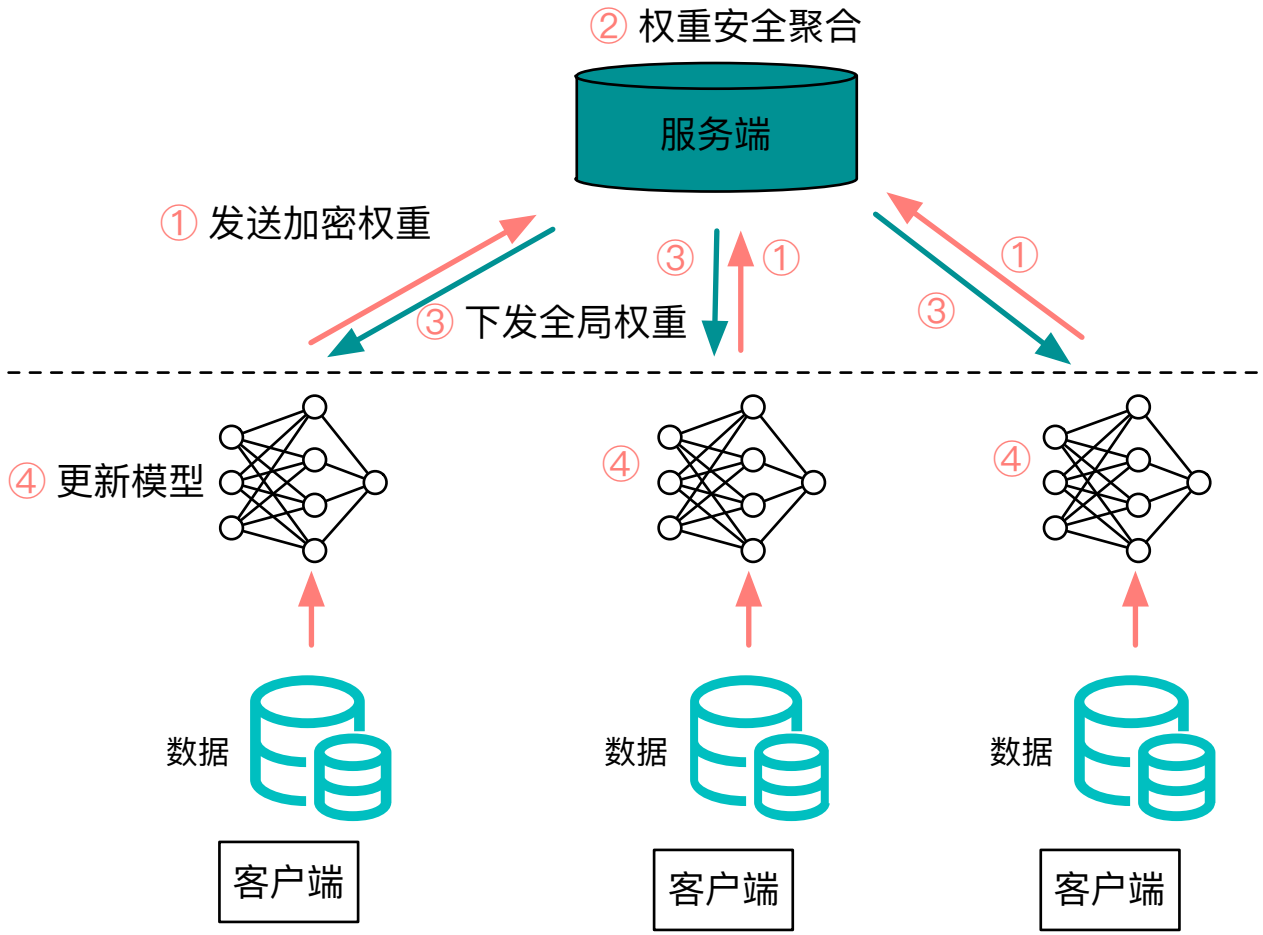
只需要简单几行

算法

PSI	SS LR	联邦NN
隐私求交	秘密分享逻辑回归	水平联邦深度学习

算法 - 联邦NN

水平联邦原理



算法 - 联邦NN

1. 准备数据
2. 定义模型
3. 定义客户端
4. 安全聚合
5. 执行训练

以经典的图片数据集mnist为例

```
from sklearn.preprocessing import OneHotEncoder
from tensorflow import keras


def dataset(batch_size, epochs):
    (x_train, y_train), (_, _) = keras.datasets.mnist.load_data()
    x_train = x_train / 255

    encoder = OneHotEncoder(sparse=False)
    y_train = encoder.fit_transform(y_train.reshape(-1, 1))
    return iter(
        tf.data.Dataset.from_tensor_slices((x_train, y_train))
        .batch(batch_size)
        .repeat(epochs)
    )
```

算法 - 联邦NN

1. 准备数据
2. 定义模型
3. 定义客户端
4. 安全聚合
5. 执行训练

```
def create_model(input_shape, num_classes):
        from tensorflow import keras
        from tensorflow.keras import layers
        model = keras.Sequential(
                [
                        keras.Input(shape=input_shape),
                        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
                        layers.MaxPooling2D(pool_size=(2, 2)),
                        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
                        layers.MaxPooling2D(pool_size=(2, 2)),
                        layers.Flatten(),
                        layers.Dropout(0.5),
                        layers.Dense(num_classes, activation="softmax"),
                ]
        )
        model.compile(loss='categorical_crossentropy‘,
                      optimizer='adam‘,
                      metrics=["accuracy"])
        return model
```

算法 - 联邦NN

1. 准备数据
2. 定义模型
3. 定义客户端
4. 安全聚合
5. 执行训练

```python
import tensorflow as tf

@sf.proxy(sf.PYUObject)
class Client:
    def __init__(self, model_builder, dataset_builder):
        self.model = model_builder()
        self.dataset = dataset_builder()

    def train_step(self):
        x, y = next(self.dataset)
        with tf.GradientTape() as tape:
            y_pred = self.model(x, training=True)
            loss = self.model.compiled_loss(
                y, y_pred, regularization_losses=self.model.losses,
            )
            trainable_vars = self.model.trainable_variables
            gradients = tape.gradient(loss, trainable_vars)
            self.model.optimizer.apply_gradients(zip(gradients, trainable_vars))

    def get_weights(self):
        return self.model.get_weights()

    def set_weights(self, weights):
        self.model.set_weights(weights)

    def metrics(self):
        return self.model.metrics
```

算法 - 联邦NN

1. 准备数据
2. 定义模型
3. 定义客户端
4. 安全聚合
5. 执行训练

```python
import jax.numpy as jnp


def aggregate(spu_device, weights_list):
    weights_list_spu = [ws.to(spu_device) for ws in weights_list]

    def _average(*data):
        return [jnp.average(jnp.array(elements), axis=0) for elements in zip(*data)]

    return spu_device(_average)(*weights_list)
```

使用SPU
完成安全聚合

算法 - 联邦NN

1. 准备数据
2. 定义模型
3. 定义客户端
4. 安全聚合
5. 执行训练

```
from functools import partial

epochs = 2
batch_size = 128
model_builder=partial(create_model, input_shape = (28, 28, 1), num_classes=10)
dataset_builder=partial(dataset, batch_size, epochs)
alice_c = Client(model_builder, dataset_builder, device=alice_pyu)
bob_c = Client(model_builder, dataset_builder, device=bob_pyu)
steps_per_epoch = int(60000 / batch_size)

for epoch in range(epochs):
  for step in range(steps_per_epoch):
    alice_c.train_step()
    bob_c.train_step()

    w_a = alice_c.get_weights()
    w_b = bob_c.get_weights()
    w_avg = aggregate(spu_device, [w_a, w_b])
    alice_c.set_weights(w_avg.to(alice_pyu))
    bob_c.set_weights(w_avg.to(bob_pyu))

    print(
      f'Epoch {epoch} step {step} loss:',
      sf.reveal(alice_c.metrics())[0].result().numpy()
    )
```

```
Epoch 0 step 0 loss: 2.320026
Epoch 0 step 1 loss: 2.3099656
Epoch 0 step 2 loss: 2.3020344
Epoch 0 step 3 loss: 2.297265
Epoch 0 step 4 loss: 2.294231
...
```

算法 - 联邦NN

SecretFlow内置了水平联邦NN

```
from functools import partial
from secretflow.security.aggregation import SecureAggregator
from secretflow.ml.nn import FLModel
from secretflow.utils.simulation.datasets import load_mnist


(x_train, y_train), (x_test, y_test) = load_mnist(
    parts=[alice_pyu, bob_pyu], normalized_x=True, categorical_y=True
)
model_builder=partial(create_model, input_shape=(28, 28, 1), num_classes=10)
aggregator = SecureAggregator(alice_pyu, [alice_pyu, bob_pyu])
fed_model = FLModel(
    server=alice_pyu,
    device_list=[alice_pyu, bob_pyu],
    model=model_builder,
    aggregator=aggregator,
    strategy="fed_avg_w",
    backend="tensorflow",
)
fed_model.fit(
    x=x_train,
    y=y_train,
    validation_data=(x_test, y_test),
    epochs=2,
    batch_size=128,
    aggregate_freq=1,
)
```

调用fit
即可搞定

算法

更多算法，访问我们的文档

https://secretflow.readthedocs.io/

- 水平XGBoost
- 拆分学习
- 混合切分逻辑回归
- HESS-LR
- SS-XGB
- ...

课程总结

设备

- SPU - 明文计算设备
- SPU - MPC密态设备
- HEU - 同态加密设备

to - 设备之间数据流转

算法

- PSI（隐私求交）
- SS LR（秘密分享逻辑回归）
- 联邦NN（水平联邦深度学习）

课程总结

灵活	PYU、SPU、HEU各种设备自由组合
易用	Python作为前端语言，与现有ML生态完全兼容
丰富	内置各种常见算子

THANKS