# PPML和SPU

ML和MPC关注的东西不一样

ML：网络、优化器

MPC：计算协议、密码学协议

SPU的编译器层：

复用了AI框架的部分优化，有MPC的独占优化（除换乘，减少比较次数）

# 浮点数转为定点数

浮点数表示：以float32为例

1符号 + 8指数 + 23小数

定点小数表示：以8bit环为例，

1符号 + 3整数 + 4小数

对比：

浮点数计算精度高，计算复杂，取值范围大

定点数计算精度低，计算量小，取值范围小

# 明文算法迁移流程

一、算法用jax api实现

考虑jax.numpy, jax.lax, jax.scipy

二、测试明文计算的精度

spu simulation (api: simulator)

(1)导入方式：

import spu.utils.simulation as spsim

import spu.spu\_pb2 as spu\_pb2

import spu

sim = spsim.Simulator.simple(2, spu\_pb2.ProtocolKind.ABY3, spu\_pb2.FieldType.FM64)

# 几方、什么协议、多大环

(2)定义运行函数

（python函数）

(3)运行密态程序

比较一下cheetah和aby3的结果差异

(4)分析问题

ML角度：算法本身有无改动？考虑增大batch\_size, 增大eps, 其他近似方式

MPC角度：cheetah算乘法会有误差。可以增大环、fxp的大小

三、测试密文计算精度

spu emulation (api: emulator)需要从spu源码编译

1. 定义emulator
2. 准备数据

需要把明文数据进行seal(emulator.seal(…))

1. 运行程序

emulator.run(…)

1. 优化性能：需要mpc的知识

# SPU常见问题

1、支持算子

可以通过emulation/simulation自己查

查看官方文档：np算子和xla算子

2、非线性算子大概误差：

两个方面，

系统设定误差（环、fxp的大小，以及truncation协议）

非线性算子拟合误差

一些数学算子的大概误差有文档提示

3、明文运行可以，密文跑不了

运行报错：jittable? SPU算子是否支持

误差过大：

溢出：数字太大/太小

SPU内部的浮点随机数生成器：spu对IEEE754支持不完善，暂时不要用

线性代数算子：矩阵分解、奇异值分解等（大概率下溢）

误差适中：增大环大小、提高fxp精度（如果就差了1-2个点）

4、emulation比simulation还快

数据没有seal

5、对密态算法优化的思路

(1) 减少耗时算子调用（计算公式重写、多项式近似…）

(2) 避免重复计算，空间换时间

(3) 并行化

在SPU并行优化的基础上进一步优化：

1. 算法层：for循环用高阶tensor代替，也可以考虑jax.vmap进行自动向量化
2. Runtime：开启更多并行，如experimental\_enable\_inter\_op\_par（实验性功能）