

6月3日，咱们 FATE 开源社区首届圆桌会完美落幕。这场圆桌会上，我们邀请了 40+位社区朋友，他们有的是踊跃帮助其他人解决技术问题的“热心群众”，有的已经把联邦学习应用于生产环境的 FATE 成为生产力工具的人。

在这场圆桌会上，首先由咱们 FATE 内部专家团队对 FATE 1.4 新版本的各模块功能点进行简介说明，并对大家遇到的一些问题进行交流解答，以下为专家分享主题&大纲（PPT 可添加 FATE 社区助手获取）：

FATE：1.4 更新简介

- 1、横向 SecureBoost 树
- 2、早停机制(Early Stop)
- 3、纵向广义线性模型逐步回归
- 4、联邦学习开源平台-FATE

——MingChao Tan

FATE v1.4 Release Notes-Federated ML

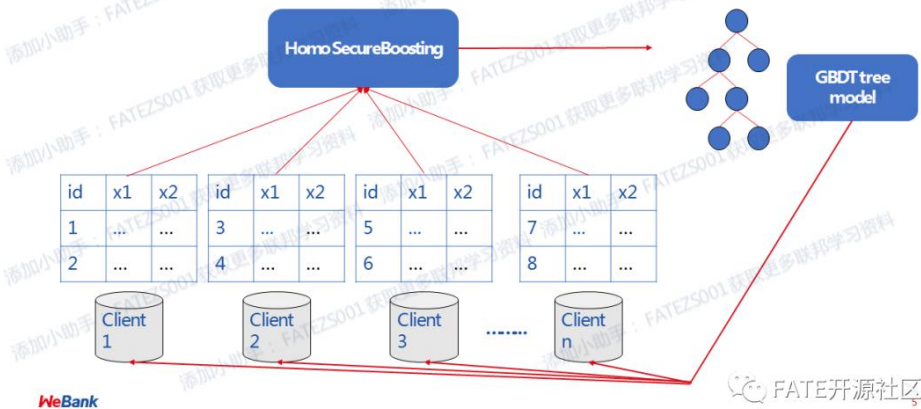
Federated ML

- ✓ 横向联邦支持SecureBoost树算法
- ✓ 纵向联邦算法支持Early Stop策略，允许训练时保存最优模型
- ✓ 纵向联邦广义线性模型系列全面支持基于AIC及BIC进行逐步回归模型选择
- ✓ 纵向联邦分箱支持最优分箱方法，支持iv\gini\chi-square\ks，同时支持非对称分箱方法
- ✓ AI生态互操作：横向联邦NN支持pytorch backend
- ✓ 横向联邦框架重构，显著提升横向联邦算法开发效率和降低接入门槛
- ✓ Local Baseline增加多分类任务支持
- ✓ 预测组件增加输入输出数据一致性校验
- ✓ 优化训练评估过程，3倍提升训练效率

WeBank

FATE开源社区

Homo secureboosting的场景



扫码添加小助手获取完整 PPT（微信号：FATEZS001）

Eggroll 2：新一代的计算引擎

- 1、Eggroll 2 的改进
- 2、全新的 session 与资源管理
- 3、参数配置及常见排错方法

——MAX

FATE v1.4 Release Notes - Eggroll

稳定性提升

- ✓ 全新的资源管理组件，全新的session机制。即使session出错，只需要一个简单函数调用，临时拉起的计算进程即可被清理
- ✓ 移除storage service，无需C++/native库的编译
- ✓ 支持联邦学习算法在28%的丢包率之下依然正常运行

性能提升

- ✓ 运行于Eggroll 2的联邦学习算法，性能显著提升。部分算法更达到了超过10倍的性能提升
- ✓ Join接口在联邦学习场景下，比pyspark快16倍

用户体验提升

- ✓ 快速部署。只需Maven编译、pip安装依赖、修改配置，即可运行
- ✓ 轻量依赖。查看我们的requirements.txt和pom.xml就知道了
- ✓ 易于调试。我们不仅提供了必要的运行上下文信息，还将调试的关键系统状态保存在日志文件及数据库中
- ✓ 少常驻进程。必要的常驻进程只是JVM应用

WeBank

FATE开源社区

常驻服务更少



WeBank

FATE开源社区

扫码添加小助手获取完整 PPT (微信号: FATEZS001)

FATE v1.4 主要更新-系统工程部分

- 1、FATE v1.4 整体架构优化
- 2、FATE v1.4 部署介绍

——Jarvis Zeng

FATE v1.4 Release Note - Engineering

FATE-Flow

1. 重构模型管理，本机文件目录方式存储，存储结构更加灵活，信息更丰富。
2. 支持模型文件形式导出导入、模型存储到可靠分布式系统及恢复(目前支持Redis)
3. 使用MySQL代替Redis实现作业队列，降低系统复杂度
4. 支持上传客户端所在机器的本地文件
5. 自动检测数据表是否已经存在，并提供销毁删除选项
6. 系统、算法、调度命令日志分离，调度命令日志独立可审计

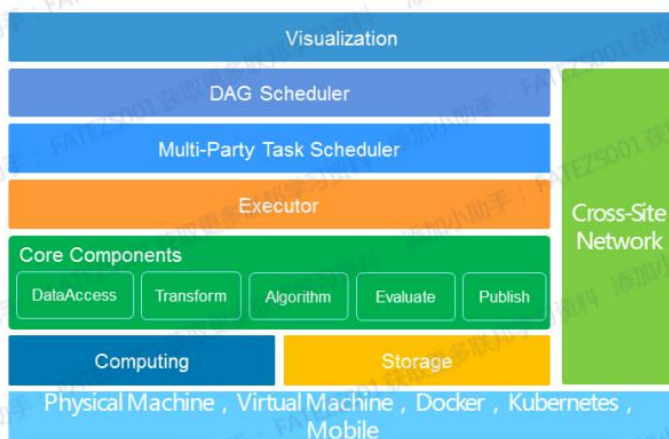
FATEBoard

1. pearson相关性：支持根据role或特征的筛选来绘制矩阵图；支持对相关性值的区间筛选来绘制矩阵图；支持按照相关性大小进行排序；
2. 横向Secureboost：新增homo secureboost模型的可视化，包括树模型、特征重要性、交叉验证等；
3. GLM-stepwise：新增对纵向LR、纵向线性回归、纵向poisson回归的stepwise的可视化，包括每步的模拟拟合统计、特征分析、最大似然分析、待进入特征分析，以及总结输出；
4. 纵向LR、localbaseline：增加对one_vs_rest的输出可视化；
5. 模型输出中标明最佳迭代模型版本

WeBank

FATE开源社区

FATE离线平台基本架构



Keywords :

1. DAG
2. Component-based Pipeline
3. Multi-Party Scheduling
4. Cross-Site Transmission
5. Distributed Computing
6. Distributed Storage
7. Visualization

WeBank

FATE开源社区

扫码添加小助手获取完整 PPT (微信号: FATEZS001)

在专题分享后，我们共同探讨了 FATE 存在的一些问题并提出了解决方案与思路，并对 FATE 的后续版本能力的可能性进行论述，一起对 FATE 未来的应用畅想探讨，以下为本次圆桌会上的互动环节的问答内容（节选）：

Q1、FATE 的计算引擎能否直接用已经比较成熟的引擎，比如 Spark？

当前版本（1.4）已经支持 Spark。另外，其实 EggRoll 吸取了现有引擎的一些经验，尤其在联邦学习场景下非常有优势，所以其会是 FATE 主推的引擎。当然为了生态的多样性、打通不同框架的互操作性，FATE 同时也在持续优化支持更多的引擎。

Q2、可否再详细列举下 Spark 和 EggRoll 在设计上的不同呢？

其中一个比较大的差异是，Spark 和 EggRoll 对于存储这一点的理解，设计上是有不同的。Spark 可能更多地使用 hdfs，它在调度的时候，需要把数据从 hdfs 拉到执行计算的节点，或者拉到比较接近计算的那个节点去执行的。如果用一种流行说法来说，spark 采取的做法叫：存储和计算分离。

EggRoll 跟 Spark 这一方面对比，走了另外一条路线。EggRoll 是把存储和计算绑定在一起的，就是不传输数据，而是把计算逻辑传输到数据所在的地方去执行。由于计算逻辑较小，所以不需要进行大规模的数据移动。

另外一点不同，EggRoll 的每个节点是有状态的。前面提到 EggRoll 是要把存储和计算绑在一起，所以在使用集群的时候，每个节点是有状态的。因为每一个节点上存储的数据是属于不同分片的。所以这两点可能是其中两个与 spark 比较大的区别。

Q3、 FATE1.4 后续是否提供 helm 部署方式(在 k8s 上)?

已经支持，可以参考子项目 KubeFATE：

<https://github.com/FederatedAI/KubeFATE>

Q4、任务运行时间比较久的话，会有大量的僵尸进程占据内存，如何 kill 掉之前任务的僵尸进程？

需要僵尸进程具体的名称进行具体分析。作为一个已大规模在生产使用的工业级框架，FATE 做了大量且严格的性能测试与稳定性测试。尤其在 1.4 版本 EggRoll 引擎大升级的情况下，过去这段时间实际生产运行的观察也符合预期。

Q5、关于模型导入和导出的问题，模型导出后还是分布式模型，但无法在本地环境中还原成中心化的模型来做预测吗？

此前的 1.3 版本，横向的模型通过 component_model 的接口，导出后是个 zip 的文件包，解开后可以还原成 keras 的模型。

纵向的模型的话，本来就是分布式的，所以导出后也无法在本地使用？

这是两个不同的问题，模型的导入导出，是指模型的拥有方可以在其不同的运行环境下复用模型。纵向联邦的模型，本身就无法支持一方单独使用，这个是其原理所决定的。

Q6、关于加密的问题，在对比了官方提供的三个横向的模型，发现存在多种加密方式 paillier 加密、Secure Aggregation。为什么不用统一一点的方式呢？因为横向模型差异较少的情况下，为什么有的选择用 Paillier，有的用 secure aggregation？

在横向中,Paillier 加密和 Secure Aggregation 的作用是不一样的。paillier 只会用在 lr 里的 paillier 加密，是为了不希望 Host 方拿到这个模型，所以我们对它加密，在 LR 当中也可以不用 Paillier 加密，只使用 Secure Aggregation 也是可以的。

Secure Aggregation 的作用就是不希望 Arbiter 会拿到每一个单独某一方的这个模型，他只能拿到这个所有方聚合后的模型，所以它作用是不太一样的。

Q7、 FATE 为什么只用 paillier 做加密？

FATE 不会聚焦于特定的一种产品算法，比如 Paillier。现在它已经支持了 SecureSharing，以及一些仿射变换的同态加密、对称加密等多种算法。我们会根据你算法特点、实际情况去选择合适的加密算法来满足特定需求。

Q8、 请问是否会考虑，通过直连外部各种异构数据源将数据直接将数据写入 Imdb，进行后续计算。不用将数据写成离线文件，在 p 再上传到 Imdb？

这个能力正在开发当中，也是 FATE 正在逐步去支持当前现有的大数据生态和一些计划，如后期我们会支持直接从 hdfs 和 hive 去直接导入数据到 fate 的集群里面。

从 EggRoll 角度分析，其实外部的多种异构数据源，在 EggRoll 2 里面是比较容易去支持的。在我们的另外一个业务里面，已经实现了直接读写 hdfs，这个后续内部会讨论具体哪种方式去比较好。因为在刚才介绍架构的图的最底层，看到 EggRoll 已经支持不同的存储介质，然后这里面我们已经做了抽象，所以已经可以只读写 htfs 这种了，按性能的话肯定不如 lmdb 好。

Q9、问下升级后的版本对资源要求如何？

Eggroll 方面而言，因常驻进程的减少，所以它的常驻占用的内存是少了的，在具体去执行的时候，计算引擎的资源要求跟以前应该是差不多的。总的来说常驻的一部分是少了，而且计算引擎执行完以后会释放。

Q10、请问老师能不能省去第三方，第三方能不能是合作的某一方？

我们的纵向算法不是所有的算法都一定有第三方，比如纵向的 secure boost 布置就没有第三方，纵向的 nn 也没有第三方，未来的话 lr 也是希望往消除第三方的思路方式发展。有第三方确实是比较麻烦，如果说目前你必须在 lr 里面是使用第三方的情况下呢，但只有两方又要使用 lr 算法的情况下，我们会推荐把 Arbiter 放在 Host 这边，相对而言，arbiter 和 host 的串通，若想要破解 guest 方的数据，是比较困难的。但这仅是权宜之计，从安全考量还是尽量加入第三方。

原文链接：<https://mp.weixin.qq.com/s/6yDd58N8KVs78bPD-cC3Rg>