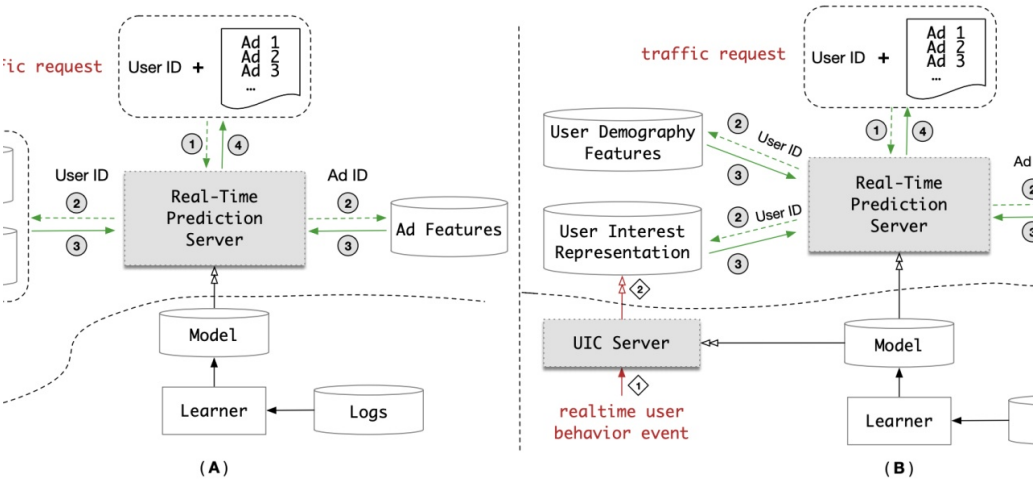


从阿里的User Interest Center看模型线上实时serving方法

王喆
深度学习（Deep Learning）、机器学习 话题的优秀回答者

123 人赞同了该文章

这里是**「王喆的机器学习笔记」**的第二十九篇文章。

最近写书的时候在总结一些深度学习模型线上serving的主流方法。之前的专栏文章也有所涉及（如何解决推荐系统工程难题——深度学习推荐模型线上serving）。

但是看到阿里妈妈去年KDD上的文章 *Practice on Long Sequential User Behavior Modeling for Click-Through Rate Prediction* 之后，还是有种相见恨晚的感觉，因为文中提供的model serving的解决方案是自己想实现还未来得及实现的。。所以赶快跟大家分享一下，我想应该能解决不少同学实践中的问题。

什么是Model Serving?

一句话解释就是：

Model Serving解决的是模型离线训练好之后，如何进行线上实时推断的问题。

在每个服务器节点动辄几千上万QPS的压力下，必然不可能在tensorflow，spark mllib等训练环境中进行实时推断。必须有一个模型服务器来承载模型相关的参数或者数据，进行几十毫秒级别的实时推断，这就是model serving面临的主要挑战。

Model Serving的主要方法

在之前的专栏文章如何解决推荐系统工程难题——深度学习推荐模型线上serving 中提到了几种主流的方法，分别是：

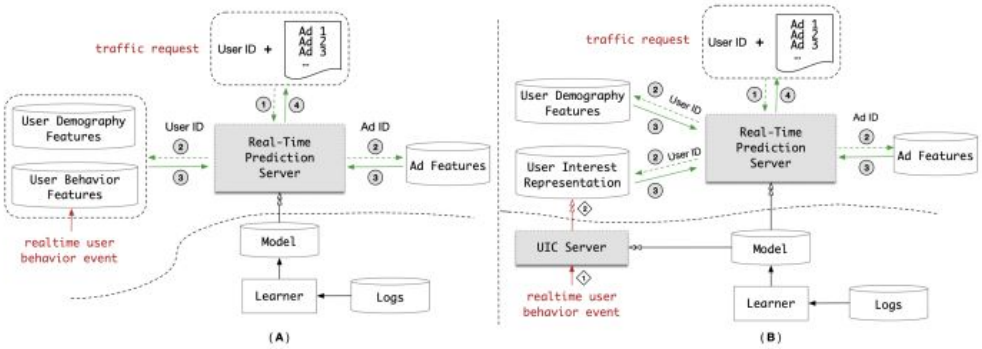
1. **自研平台**
2. **预训练embedding+轻量级模型**
3. **PMML等模型序列化和解析工具**

4. **TensorFlow serving等平台原生model serving工具**

这里就不再详细介绍，感兴趣的同学可以回顾一下之前的文章。今天主要想跟大家探讨的还是阿里妈妈提出的Model Serving方法 **User Interest Center**。

什么是User Interest Center?

先上架构图

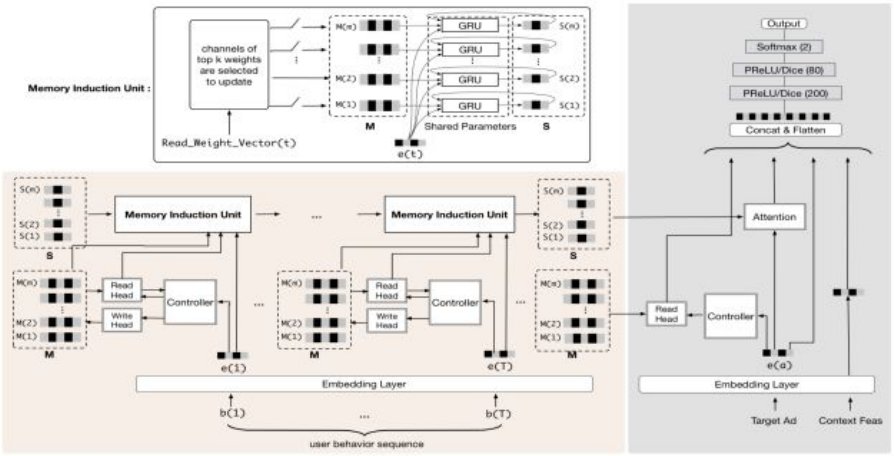


UIC server的架构图

上图的(A)和(B)分别代表了两种不同的模型服务架构，两图中部横向的虚线代表了**在线环境**和**离线环境**的分隔。

那么A架构就代表着一个非常经典的解决方案：离线部分做模型训练，在线部分根据用户各类特征（User Demography Features 和 User Behavior Features）以及广告特征（Ad Features），在模型服务器（Real-Time Prediction Server）中进行预估。

这个过程是直观的，也是看上去天经地义的。但“盛世之下潜藏着危机“啊，问题的关键就在于**在模型越来越复杂之后，特别是像DIEN或者MIMN这类模型加入序列结构之后，这些序列结构的推断时间实在是太长了**。结构已经复杂到模型服务器在几十毫秒的时间内根本没有可能推断完的地步。



阿里最新的MIMN（Multi-channel user Interest Memory Network）模型

怎么办?

阿里的解决方案是图中的B架构，包括两大部分：

1.用户兴趣“表达”模块

B架构将A架构的“用户行为特征(User Behavior Features)在线数据库”替换成了“**用户兴趣表达(User Interest Representation)在线数据库**”。

这一变化对模型推断过程非常重要。无论是 DIEN 还是 MIMN，它们表达用户兴趣的最终形式都是兴趣 Embedding 向量。如果在线获取的是用户行为特征序列，那么对实时预估服务器(Real-time Prediction Server)来说，还需要运行复杂的序列模型推断过程生成用户兴趣向量。

而如果在线获取的是用户兴趣向量，那么实时预估服务器就可以跳过序列模型阶段，直接开始 MLP 阶段的运算。MLP 的层数相较序列模型大大减少，而且便于并行计算，因此整个实时预估的延迟可以大幅减少。

当然，虽然“用户兴趣表达模块”这个名字很fancy，但本质上应该是以类似redis的内存数据库为主实现的。

2. 用户兴趣“中心”模块

B架构增加了一个服务模块——**用户兴趣中心(User Interest Center，UIC)**。 UIC 用于根据用户行为序列生成用户兴趣向量，对 DIEN 和 MIMN 来说，UIC 运行着生成用户兴趣向量的部分模型。

与此同时，实时用户行为事件(realtime user behavior event)的更新方式也发生着变化，对A架构来说，一个新的用户行为事件产生时，该事件会被插入用户行为特征数据库中，而对B架构来说，**新的用户行为事件会触发 UIC 的更新逻辑**，UIC 会利用该事件更新对应用户的兴趣Embedding向量。

这个解决方案让我觉得优雅的地方在于：

在大幅降低了模型在线推断复杂度的同时，它居然是准实时的。因为UIC的更新逻辑是用户的行为发生改变，也就是说用户的embedding会在行为发生改变时通过模型离线推断完成更新。某种意义上说，这也可以算是一种模型online learning的方法之一了。

由于embedding的更新过程是离线进行的，与线上实时预估服务器是异步的，因此就不会拖累线上预估的速度。

而由于embedding异步更新的触发条件是用户行为的变化，所以embedding的更新也是准实时的。而不像有些embedding一样，一旦生成，除了下次模型训练，就不再更新。

采用UIC后Model Serving延迟大幅减少

我们可以根据UIC架构图中从1到4的圆圈编号再捋一遍Model Serving的过程：

1. 流量请求(traffic request)到来，其中携带了用户 ID(User ID)和待排序的候选商品 ID(Ad ID)。
2. 实时预估服务器根据用户 ID 和候选商品 ID 获取用户和商品特征(Ad Features)，用户特征具体包括用户的人口属性特征(User Demography Features) 和用户行为特征(a 架构)或用户兴趣表达向量(b 架构)。
3. 实时预估服务器利用用户和商品特征进行预估和排序，返回最终排序结果给请求方。b架构对最耗时的序列模型部分进行了拆解，因此大幅降低了模型服务的总延迟。
4. 返回模型预估结果。

根据阿里巴巴公开的数据，每个服务节点在 500 QPS(Queries Per Second， 每秒查询次数)的压力下，DIEN 模型的预估时间从 200 毫秒降至 19 毫秒。这无疑是从工程角度优化模型服务过程

的功劳。

当然也可以看到，阿里的UIC架构还是遵循了“Embedding+轻量级线上模型”的部署方案，只不过利用UIC对Embedding部分的生成、存储、更新进行了近乎完美的管理。可以说是“**Embedding+轻量级线上模型**”这种Model Serving方法的best practice。

总结

上部分和线下部分，模型复杂的序列结构在线下运行，利用UIC生成和更新Embedding，结果存储在“用户兴趣表达模块”；线上实现模型较为轻量级的MLP部分，使模型能够利用更多的特征进行实时预估。

可以说这是一次机器学习理论和机器学习工程系统完美结合的方案，推荐受困于model serving效率和实时性的团队尝试。

照例给大家提出两个问题讨论：

1. UIC中用户Embedding更新的触发方式到底是应该是什么？是Real-time prediction server接收到的一次用户请求，还是通过flink处理的一个window内部的用户行为？如果你是工程师，你会采用哪种方法？两种方法的优缺点是什么？
2. 为什么一定要把序列模型放到离线进行处理，序列模型线上inference的速度就真的没有方法提高吗？你有什么经验？

最后欢迎大家关注我的**微信公众号：王喆的机器学习笔记（wangzhenotes）**，跟踪计算广告、推荐系统等机器学习领域前沿。

想进一步交流的同学也可以**通过公众号加我的微信**一同探讨技术问题。另外很多同学通过知乎和微信询问我书的事情，出版社刚刚通知我下周就应该能够上市了，希望到时候大家多支持，谢谢！

发布于昨天 12:35

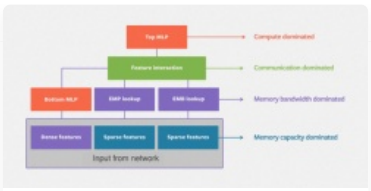
深度学习（Deep Learning） 机器学习 人工智能

文章被以下专栏收录

推荐阅读

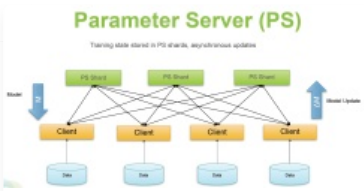
阿里DIN源码之如何建模用户序列（1）：base方案

厚德载物丶 发表于推荐系统打...



透着浓浓工业风的Facebook深度学习推荐系统论文

王喆 发表于王喆的机器...



一文读懂「Parameter Server」的分布式机器学习训

王喆 发表于王喆的机器...

《Graph Neural Ne 分享

关于《Graph Neural N Review of Methods ar Applications》分享引 篇中我提到，其作者所 还有一篇相关的综述， 与基金撰写还有一段时

Lemon... 发表于

11 条评论 切换为时间排序

写下你的评论...

Ghost Fan 1 天前

对于问题1 我估计还是会采用window的方式，实时处理负载太高，而且我估计在用户的下一次刷新中整个更新流程是不能做完的，还是有一定延后，虽然实时理论上能最快的捕捉用户兴趣，但是既然在下一次刷新前 流程做不完，意义也不大了

1

张明锋 回复 **Ghost Fan** 1 天前

我也是认为应该以windows作为作业切割。因为兴趣这东西仅仅针对用户当前行为关联，对用户的下一个行为没有多少推导作用

1

王喆 (作者) 回复 **Ghost Fan** 1 天前

想法完全一致 基于flink的准实时完全够用了 基于request的更新压力太大 但收益不见得显著 技术上的投资回报比太低了

3

张相於 1 天前

严格来说B方案里用户兴趣的更新应该叫作近线而不是离线吧？前几年我做电商的时候也实现过类似方案，本质上是一种缓存，即用提前的、可能冗余的计算和存储来换取实时计算的压力。

赞

王喆 (作者) 回复 **张相於** 1 天前

是的 near real-time

赞

付鹏 1 天前

这篇文章让人摸不到头脑，这么说的话，双塔模型不就是这种思维的吗

1

王喆 (作者) 回复 **付鹏** 1 天前

广义上也算 但这篇的关键点在于怎么做embedding更新

1

知乎用户 23 小时前

额 冒昧问下UIC 怎样利用该事件更新对应到用户的兴趣Embedding向量？每个用户的兴趣Embedding是唯一的么？推荐小白的问题。

1

王喆 (作者) 回复 知乎用户 16 小时前

因为用户embedding是基于用户行为序列生成的 所以当行为序列更新更新时 再跑一边模型inference过程就可以更新emb。

在阿里的mimn模型中 兴趣向量是多个 但其实大部分模型的都是一个 根据你的模型结构而定

赞

龚禹pangolulu 16 小时前

在追求推荐模型的实时性上，我们可以更加极致：developer.aliyun.com/ar...

赞

王喆 (作者) 回复 **龚禹pangolulu** 16 小时前

边缘计算 嵌入式模型 大赞 最近也在做相关研究 感觉必然是下一波推荐系统的趋势了

赞