

如何解决推荐系统工程难题——深度学习推荐模型线上serving?



王喆

深度学习（Deep Learning）、机器学习 话题的优秀回答者

217 人赞同了该文章

这里是「王喆的机器学习笔记」的第二十三篇文章，这篇文章希望讨论的问题是深度推荐模型的线上serving问题。

对于推荐模型的离线训练，很多同学已经非常熟悉，无论是TensorFlow，PyTorch，还是传统一点的Spark MLlib都提供了比较成熟的离线并行训练环境。但推荐模型终究是要在线上环境进行inference的，如何将离线训练好的模型部署于线上的生产环境，进行线上实时的inference，其实一直是业界的一个难点。本篇文章希望跟大家讨论一下几种可行的推荐模型线上serving方法。

一、自研平台

无论是在五六年前深度学习刚兴起的时代，还是TensorFlow，PyTorch已经大行其道的今天，自研机器学习训练与上线的平台仍然是很多大中型公司的重要选项。

为什么放着灵活且成熟的TensorFlow不用，而要从头到尾进行模型和平台自研呢？重要的原因是由于TensorFlow等通用平台为了灵活性和通用性支持大量冗余的功能，导致平台过重，难以修改和定制。而自研平台的好处是可以根据公司业务和需求进行定制化的实现，并兼顾模型serving的效率。笔者在之前的工作中就曾经参与过FTRL和DNN的实现和线上serving平台的开发。由于不依赖于任何第三方工具，线上serving过程可以根据生产环境进行实现，比如采用Java Server作为线上服务器，那么上线FTRL的过程就是从参数服务器或内存数据库中得到模型参数，然后用Java实现模型inference的逻辑。

但自研平台的弊端也是显而易见的，由于实现模型的时间成本较高，自研一到两种模型是可行的，但往往无法做到数十种模型的实现、比较、和调优。而在模型结构层出不穷的今天，自研模型的迭代周期过长。因此自研平台和模型往往只在大公司采用，或者在已经确定模型结构的前提下，手动实现inference过程的时候采用。

二、预训练embedding+轻量级模型

完全采用自研模型存在工作量大和灵活性差的问题，在各类复杂模型演化迅速的今天，自研模型的弊端更加明显，那么有没有能够结合通用平台的灵活性、功能的多样性，和自研模型线上

inference高效性的方法呢？答案是肯定的。

现在业界的很多公司其实采用了“复杂网络离线训练，生成embedding存入内存数据库，线上实现LR或浅层NN等轻量级模型拟合优化目标”的上线方式。百度曾经成功应用的“双塔”模型是非常典型的例子（如图1）。

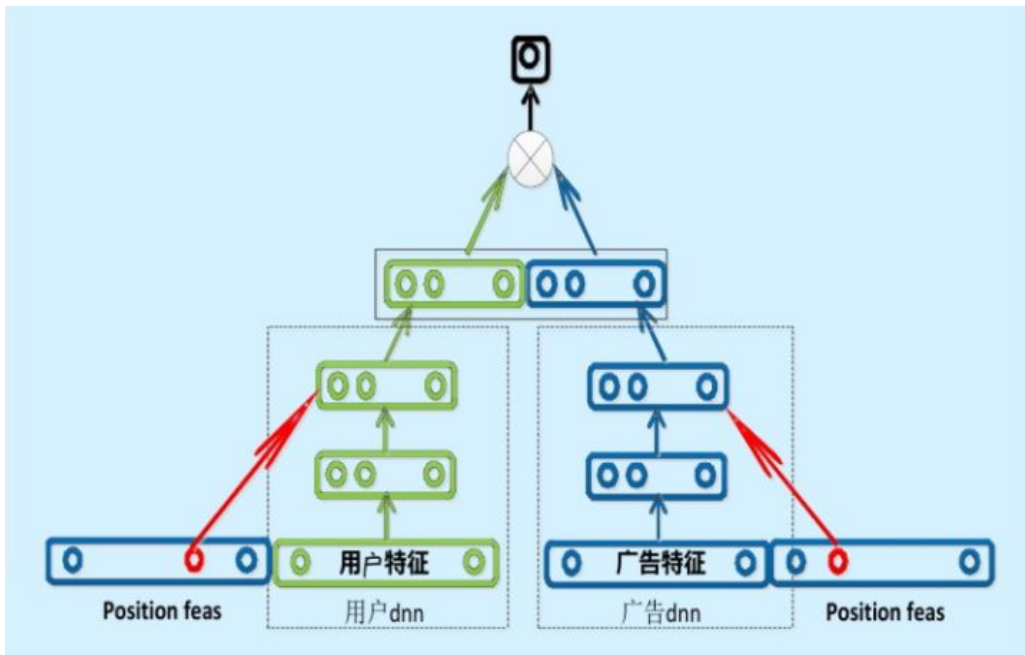


图1 百度的“双塔”模型

百度的双塔模型分别用复杂网络对“用户特征”和“广告特征”进行了embedding化，在最后的交叉层之前，用户特征和广告特征之间没有任何交互，这就形成了两个独立的“塔”，因此称为双塔模型。

在完成双塔模型的训练后，可以把最终的用户embedding和广告embedding存入内存数据库。而在线上inference时，也不用复现复杂网络，只需要实现最后一层的逻辑，在从内存数据库中取出用户embedding和广告embedding之后，通过简单计算即可得到最终的预估结果。

同样，在graph embedding技术已经非常强大的今天，利用embedding离线训练的方法已经可以融入大量user和item信息。那么利用预训练的embedding就可以大大降低线上预估模型的复杂度，从而使得手动实现深度学习网络的inference逻辑成为可能。

知乎



首发于
王喆的机器学习笔记

关注

Embedding+线上简单模型的方法是实用却高效的。但无论如何还是把模型进行了割裂。不完全是End2End。训练+End2End部署这种最“完美”的形式。有没有能够在离线训练完模型之后，直接部署的方式呢？本小节介绍一种脱离于平台的通用的模型部署方式PMML。

赞同 217

PMML的全称是“预测模型标记语言”(Predictive Model Markup Language, PMML)。是一种通用的以XML的形式表示不同模型结构参数的标记语言。在模型上线的过程中，PMML经常作为中间媒介连接离线训练平台和线上预测平台。

这里以Spark mllib模型的训练和上线过程为例解释PMML在整个机器学习模型训练及上线流程中扮演的角色（如图2）。

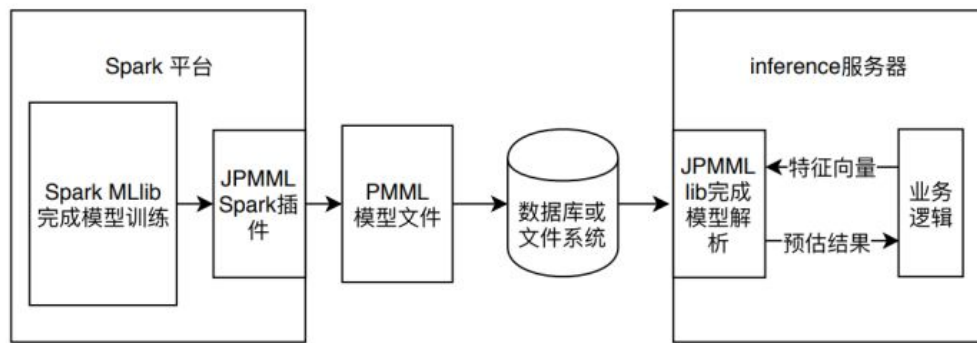


图2 Spark模型利用PMML的上线过程

图2中的例子使用了JPMML作为序列化和解析PMML文件的library。JPMML项目分为Spark和Java Server两部分。Spark部分的library完成Spark MLlib模型的序列化，生成PMML文件并保存到线上服务器能够触达的数据库或文件系统中；Java Server部分则完成PMML模型的解析，并生成预估模型，完成和业务逻辑的整合。

由于JPMML在Java Server部分只进行inference，不用考虑模型训练、分布式部署等一系列问题，因此library比较轻，能够高效的完成预估过程。与JPMML相似的开源项目还有MLeap，同样采用了PMML作为模型转换和上线的媒介。

事实上，JPMML和MLeap也具备sk-learn，TensorFlow简单模型的转换和上线能力。但针对TensorFlow的复杂模型，PMML语言的表达能力是不够的，因此上线TensorFlow模型就需要TensorFlow的原生支持——TensorFlow Serving。

四、TensorFlow Serving等原生serving平台

TensorFlow Serving 是TensorFlow推出的原生的模型serving服务器。本质上讲TensorFlow Serving的工作流程和PMML类的工具的流程是一致的。不同之处在于TensorFlow定义了自己的模型序列化标准。利用TensorFlow自带的模型序列化函数可将训练好的模型参数和结构保存至某文件路径。

TensorFlow Serving最普遍也是最便捷的serving方式是使用Docker建立模型Serving API。在准备好Docker环境后，仅需要pull image即可完成TensorFlow Serving环境的安装和准备：

```
docker pull tensorflow/serving
```

在启动该docker container后，也仅需一行命令就可启动模型的serving api：

```
tensorflow_model_server --port=8500 --rest_api_port=8501 \
  --model_name=${MODEL_NAME} --model_base_path=${MODEL_BASE_PATH}/${MODEL_NAME}
```

这里仅需注意之前保存模型的路径即可。

当然，要搭建一套完整的TensorFlow Serving服务并不是一件容易的事情，因为其中涉及到模型更新，整个docker container集群的维护和按需扩展等一系列工程问题；此外，TensorFlow Serving的性能问题也仍被业界诟病。但Tensorflow Serving的易用性和对复杂模型的支持仍使其是上线TensorFlow模型的第一选择。

除了TensorFlow Serving之外，Amazon的Sagemaker，H2O.ai的H2O平台都是类似的专业用于模型serving的服务。平台的易用性和效率都有保证，但都需要与离线训练平台进行绑定，无法做到跨平台的模型迁移部署。

总结

深度学习推荐模型的线上serving问题是非常复杂的工程问题，因为其与公司的线上服务器环境、硬件环境、离线训练环境、数据库/存储系统都有非常紧密的联系。正因为这样，各家采取的方式也都各不相同。可以说在这个问题上，即使本文已经列出了4种主要的上线方法，但也无法囊括所有业界的推荐模型上线方式。甚至于在一个公司内部，针对不同的业务场景，模型的上线方式也都不尽相同。

因此，作为一名算法工程师，除了应对主流的模式部署方式有所了解之外，还应该针对公司客观的工程环境进行综合权衡后，给出最适合的解决方案。

按惯例提出两个讨论题，欢迎大家积极分享自己的见解：

1、在应用TensorFlow Serving的过程中，你有哪些实践经验？需要把所有流量都发给TensorFlow Serving进行inference吗？有哪些减轻TensorFlow Serving负担从而加快inference速度的经验吗？

2、作为一个工程性极强的工程问题，你是在模型serving这个问题上进行取舍的？结合多种serving方式，改进现有平台，还是自研serving过程？

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，谢谢。

编辑于 2019-08-12

[人工智能](#) [深度学习 \(Deep Learning\)](#) [机器学习](#)

文章被以下专栏收录

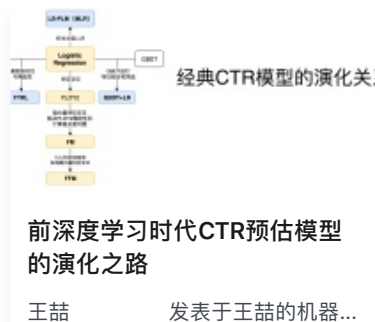


王喆的机器学习笔记

我是一名硅谷的高级机器学习工程师，开设这个专栏主要是为了记录、追踪机器学习...

进入专栏

推荐阅读



经典CTR模型的演化关系

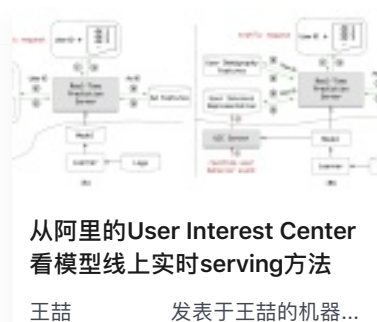
前深度学习时代CTR预估模型的演化之路

王喆 发表于王喆的机器...



神经网络不应视为模型，推理过程当为机器学习问题——等公

论智 发表于论智



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

王喆 发表于王喆的机器...



孙剑首个涉雨：3年看

旷视科技

46 条评论

切换为时间排序

写下你的评论...



知乎用户

7 个月前

以前用tf-serving，后来把tf-serving代码拆出来直接嵌入到我们c++算法服务里面用了

3



王喆 (作者) 回复 知乎用户

7 个月前

没研究过tf-serving底层，是用C++写的是吧

赞



知乎用户 回复 王喆 (作者)

7 个月前

嗯，c++的，所以可以抠出来直接拿到我们c++服务里面用，就可以跟python代码一样run指定的op

赞

查看全部 11 条回复



Ghost Fan

7 个月前

我们组 好像是用go重写模型的。。。

赞



王喆 (作者) 回复 Ghost Fan

7 个月前

看来为了上线都没少折腾

赞

jjsmz 回复 Ghost Fan

4 个月前

我擦，dnn这些也自己重新实现？

赞



知乎用户

7 个月前

双塔模型部分的用户侧embedding通常是动态的，比如包含用户query信息，是需要在线计算的

1



王喆 (作者) 回复 知乎用户

7 个月前

可以是静态的，不用在线计算的吗？

 赞

知乎用户 回复 王喆 (作者)

7 个月前

取决于用什么原始特征吧，如果只用静态特征当然可以。但用户侧塔的原始特征通常比较动态，一个是通常包含context特征，比如请求时间、当前地理位置之类的，这些都是没法离线准备好embedding的；二是对新用户embedding没办法离线提前准备好，但如果通过用户塔在线inference就可以从原始特征直接计算得到embedding。

 3

展开其他 3 条回复



大护法

7 个月前

双塔只是放粗选~粗选主要是为了性能考虑~

 5

老昶信

7 个月前

我们是把模型通过freeze graph的方式生成一个常量的graph，线上通过C++的tf lib直接load graph，通过session run来feed input fetch output，比tf serving快好多，需要词表这种依赖也是没问题的，会init表。

 3

龙红星 回复 老昶信

7 个月前

这个也是把tf serving单独抽出来了作为一个lib?

 赞

LostFrequency 回复 老昶信

7 个月前

feed dict的方式居然比tf serving快吗

 赞

展开其他 3 条回复



杜泽宇

7 个月前

我们的玩法是，session run外面包了一层自研的server，比grpc更可控，性能调优还是更多的在模型本身

 赞

浮生若梦

6 个月前

我们线上用的java服务 将inference过程用java重写一遍

 赞

九老师

6 个月前

问一个问题，一般推荐领域预测，是给一个user和一个batch的item，但是训练的时候，图组织的形式并不是这样，user和item是一对一在train的，那么预测的时候用户的输入要复制到item相同数量输入吗？这样显然很慢，那是在用户侧转为某个emb之后再复制吗？那样可能要手动去串代码，对train的代码做限制？所以，想知道有什么优雅的办法

 赞

知乎用户 回复 九老师

6 个月前

图里面要么只有item，或者只有user

 赞

cq563365 回复 知乎用户

3 个月前

分别训练user和item吗？不是一起训练， $y=F(\text{user}, \text{item})$ ，然后分别提取user隐含向量和item隐含向量来做推荐吗？

👍 赞



pory

5 个月前

对于TFtrain出来的超大模型，线上serving有什么建议呢？现在有两种思路，一是把模型分裂成embedding和nn网络，弊端就是分裂模型了，不方便；另一种思路就是把TF里的PS给改造线上可用，这种思路优点很明显，更通用易扩展，但缺陷就是改造难度不好估量，不知是否行得通

👍 1



王喆 (作者) 回复 pory

5 个月前

tensorflow serving可以先试一下 但确实性能堪忧 知道有简化ts serving的案例，应该比抽离ps的工作量小一些

👍 赞



pory 回复 王喆 (作者)

5 个月前

tf-serving不支持分布式，只有单机的[捂脸]

👍 1

查看全部 6 条回复



不知道吃啥

5 个月前

我自己用c++实现了各类常用模型结构的inference，时间性能上使用了一些矩阵计算加速的库结合一些代码优化来做的。。。

👍 1



du111

4 个月前

大厂不用flask或者django写服务，是因为性能吗？

👍 赞



Sanders

3 个月前

在线Serving的挑战在于低延时，高吞吐，大批量

1. 延时即E2E完成一个请求预测的时间，我认为可以通过缓存技术，计算加速 (OpenBLAS/MKL/CUDA)，还有Approximate技术(没有实践过)来实现；
2. 在保证单机TPS的前提下，吞吐量交给k8s来做；
3. 我认为这几个问题中大批量是最难的，当请求的batch size从上百到成千上万，E2E延时就是个问题了，比较挫的方法是客户端加并行，不知道还有没有更好的办法

👍 赞



高级电吹风

3 个月前

TensorFlow Serving 线上部署的时候是如何控制GPU大小的，部署都是独占，太浪费了，导致不能多起几个docker

👍 1



夏峥 回复 高级电吹风

1 个月前

可以在model file里控制占用gpu大小，或者启动时指定

👍 赞



夏峥

1 个月前

之前在公司里用tf-serving做人脸识别，并发性能确实堪忧，优化办法就是减小图片传输次

数，比如设置等待时间，做batch传输进行推导

 赞