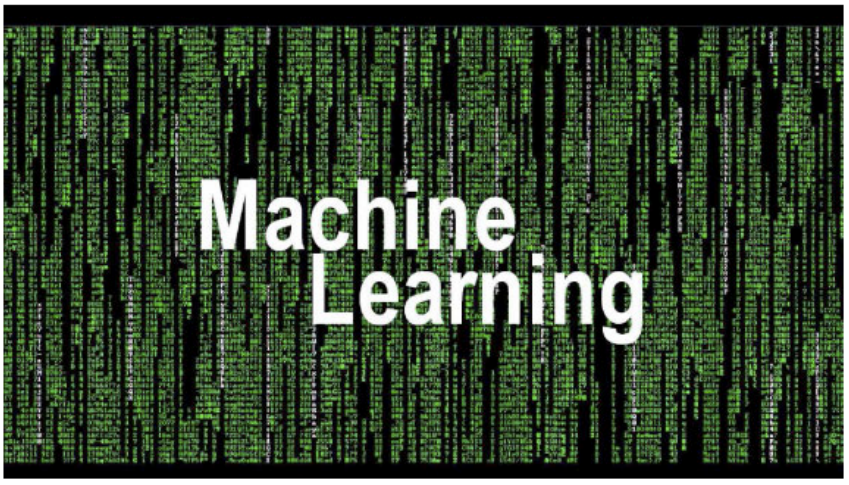
广告和推荐系统部署机器学习模型的两种架构

发表于2017年1月23日由lili

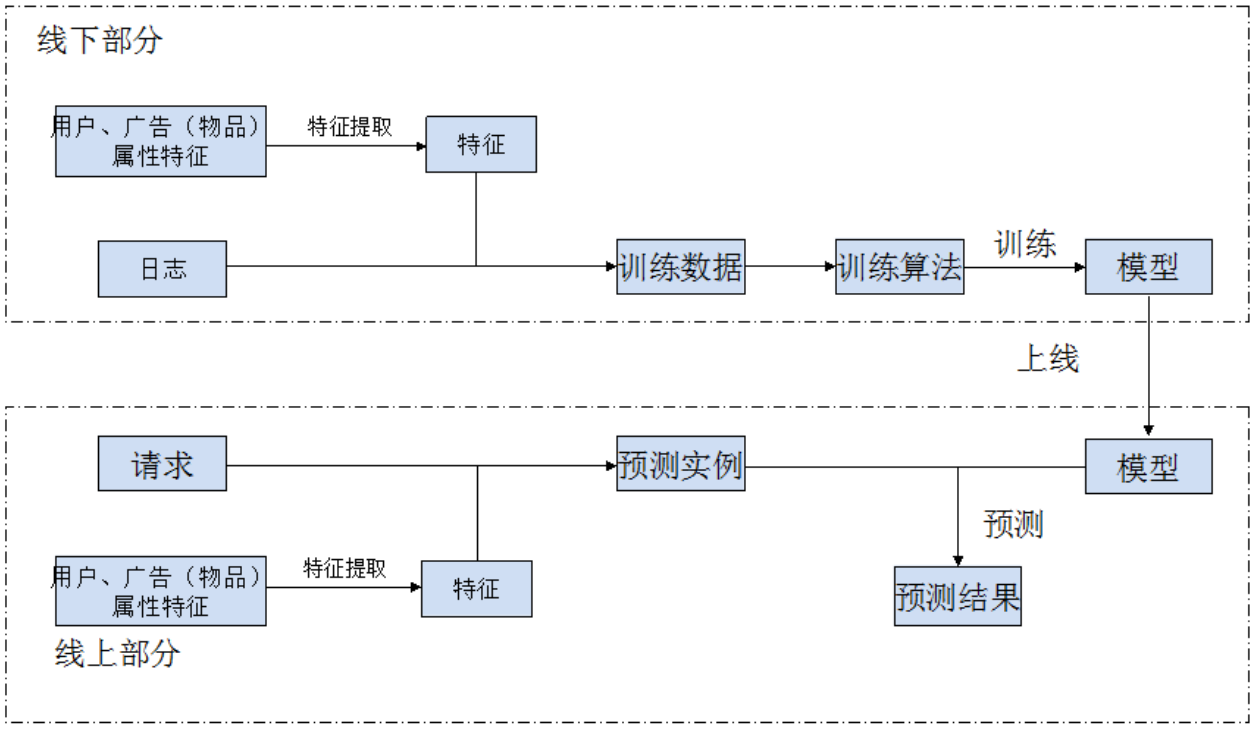
文章目录 [隐藏]

广告和推荐系统是机器学习是最成熟的应用领域。那么广告和推荐系统是怎么在线上部署机器学习模型的呢？



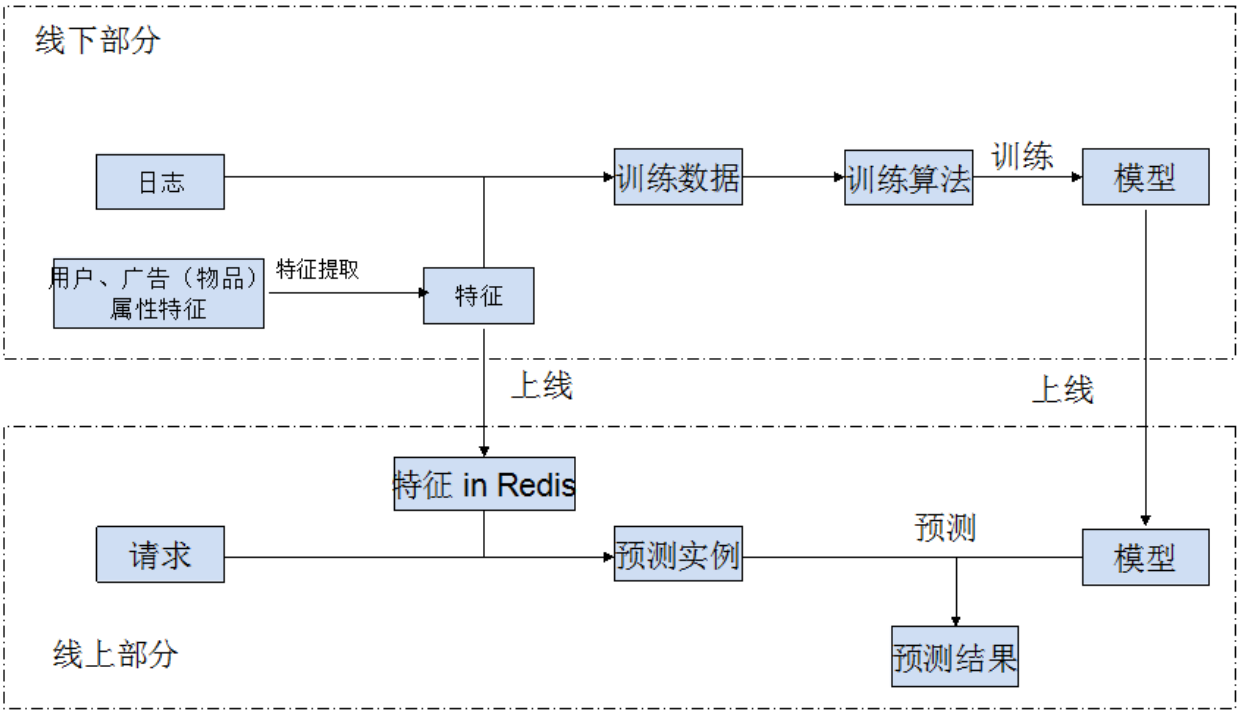
1.预测函数上线

刚刚学习机器学习时候，我认为广告和推荐系统过程如下图所示：1）线下部分，从用户和广告（物品）属性抽取用户和物品特征，将抽取的特征合并进日志生成训练数据，训练机器学习模型；2）线上部分，来了一个请求，从用户和广告（物品）属性抽取请求中的用户和物品的特征，将这些特征合并请求生成预测实例，用线上模型得到预测结果。



但是这个架构有两个问题：1）从用户和广告（物品）属性抽取特征的程序有线上线下两套，这两套程序必须保持完全一致。但由于调参的原因，特征抽取是机器学习系统中最经常发生变化的模块。经常变化的模块需要保持一致，这很困难。那么我们能不能强行地用一套程序呢？比如，我们把特征抽取和特征处理模块写成 .so 文件。这样也有问题：线下要求快速变化以方便工程师调特征，可能会使用一些训练框架（比如 Spark）；线上要求程序快速实时，要求工程师编码严谨。写成严谨的 .so 文件，能够保证线上的需求，但无法快速变化，也不能在 Spark 上使用。2）线上特征抽取要求非常快速，特别在线上吞吐量很大的情况。但有些重度特征不可能在短时间内抽取出来，比如广告的历史点击率（生成这个特征需要遍历一段时间的点击日志）。

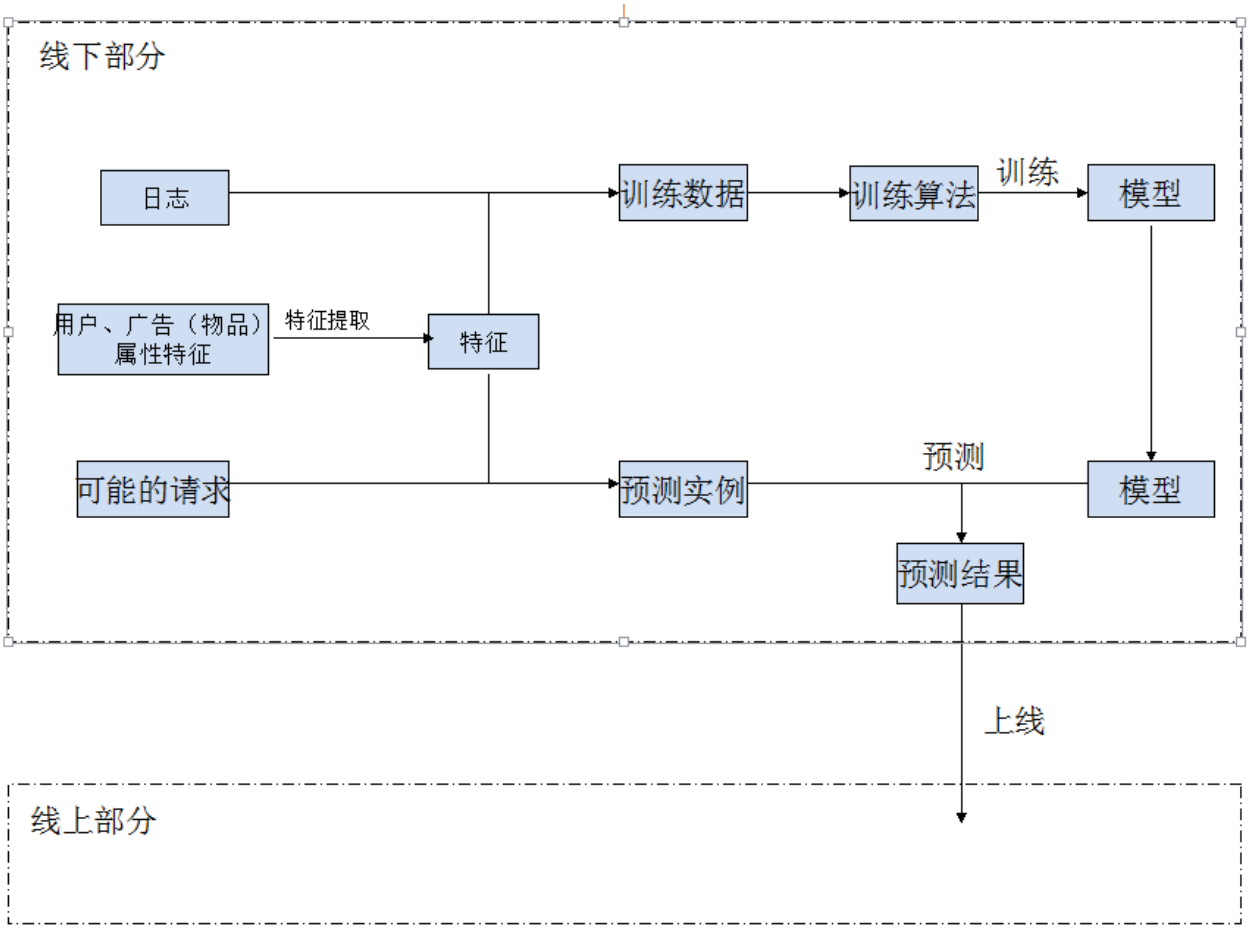
在读书期间，这两个问题困扰了我很久, 直到 2014 年我知道了神器 Redis。Redis 是一个开源内存数据库，支持集群模式、持久化和 Key-Value 数据结构。在使用时，我们可以将 Redis 看成一个巨大的哈希表。Redis 在后台开发中经常用作 cache 服务器, 后来被工程师们用于广告和推荐系统中的特征服务器。工程师将用户和广告（物品）的 ID 作为 Key，将用户和广告（物品）的特征作为 Value 存入 Redis，这样线上程序只需要用户和广告（物品）的 ID 就能知道特征。引入 Redis 之后，广告和推荐系统过程如下所示：1）线下部分，从用户和广告（物品）属性抽取用户和广告（物品）特征，把抽取的特征合并进日志生成训练数据用于训练机，并把抽取的特征上载到线上 Redis 服务器；2）线上部分，来了一个请求，从 Redis 服务器取出用户和广告（物品）特征，将特征合并进请求生成预测实例，用线上模型得到预测结果。



这种架构还有一个变种：在线下抽取特征之后不生成训练数据而是直接送到 Redis，在线上用 Storm 实时拼接训练数据。但我对这个变种的前因后果不太了解，就不展开讨论了。这种架构将预测函数（也就是训练出来的模型）部署在线上。为了和下面的架构区分开来，我们将这种架构称为预测函数上线架构。

2.预测结果上线

了解预测函数上线架构之后，我将之作为广告和推荐系统线上部署模型的“正统”。 因此当 2014 年我接触到另一种架构时，我内心是拒绝的。这种架构的要点在于把预测结果上线，具体过程如下所示：1）在线上，从用户和广告（物品）属性抽取用户和物品特征，将抽取的特征合并进日志生成训练数据，训练机器学习模型；将几乎所有可能的请求合并特征，进而生成预测实例，用模型得到预测结果；2）线上就很简单了，接入线下传过来的预测结果。这里稍微难理解的是“穷尽几乎所有可能的请求”，疑惑那么多可能的请求怎么可能穷尽呢？微博广告系统（虚构的）所有可能的请求貌似很多，但每个用户只需要匹配若干个广告就行了。因此微博广告系统的预测结果“userid,adid1,adid2...,adidn”上载到线上，一旦线上传一个 userid 请求展示广告，线上模块就按照一定的逻辑返回预测结果中这个用户对应的广告。这种架构是将预测结果部署到线上，我们将之称为预测结果上线架构。



慢慢地我也开始明白预测结果上线的好处了。预测结果上线架构将机器学习全过程和绝大部分控制逻辑都搬到线下，规避了线上的各种隐患。这样不那么厉害的工程师用不那么厉害的机器也能搞定线上模块了，毕竟线上模块只需要实现少量的控制逻辑和展示。这大大降低了建立一个广告系统或者推荐系统的难度。

我正式工作之后，组里支持运营活动的推荐系统采用了预测结果上线的架构。我发现有不少时间浪费在重跑数据上，原因在于有时需要临时增加或者删除物品。一旦增加或者删除物品，预测结果上线的推荐系统就需要重新生成预测数据（因此之前跑的数据要么没有要加的物品，要么有要删的数据）。另外一个问题就是预测结果上线架构有延时性：今天线上展示的是昨天准备的预测结果，今天准备的预测结果要等明天才能展示，这会导致节奏慢一些。最后还有一个问题，预测结果上线架构只适用于几乎所有可能的请求能够穷尽的场景。比如，预测结果上线架构不适用于搜索广告系统，因为搜索广告系统不能穷尽所有可能的请求。

3.总结

预测函数上线架构能够覆盖预测结果上线架构的适用场景，但是预测结果上线架构不能够覆盖预测函数上线架构的适用场景。同时预测函数上线架构更具灵活性。预测函数上线架构不愧为部署机器学习模型的“堂堂正正”之法。

预测结果上线架构的好处就是难度比较低。预测结果上线架构将机器学习全过程和绝大部分控制逻辑，规避了线上的各种隐患。在机器、时间和人力等各种条件不充足的情况，预测结果上线架构不失为一个好的选择。预测结果上线架构是“剑走偏锋”的机器学习模型部署之法。兵法有云：以正合以奇胜，选择哪一种架构还是需要仔细的分析和权衡。

最后欢迎关注我的公众号，每两周的更新就会有提醒哦~

欢迎关注

公众号讲述**机器学习**和**系统研发**的轶事，希望讲得有趣，**每周日更新**~

扫描二维码即可关注。您，不关注下么？

此条目发表在算法荟萃, 编程开发分类目录。将固定链接加入收藏夹。

《广告和推荐系统部署机器学习模型的两种架构》有 **2** 条评论

Pingback引用通告： *Build a recommender system with Spark: Logistic Regression – I failed the Turing Test*

Pingback引用通告： 一个特殊场景的 *LR* 预测优化 *Trick | AlgorithmDog*