24 | 离线评估：常用的推荐系统离线评估方法有哪些？

你好，我是王喆。今天我们要进入一个全新的章节，模型评估篇。

在推荐系统这个行业，所有人都在谈效果。就像我们在学习推荐模型篇的时候，你肯定也有过这样的疑问：

1. DIEN 这个模型的效果到底怎么样啊？
2. 我们用深度学习来构建模型到底能让推荐系统效果提高多少啊？
3. DeepFM 的效果是不是会比 Wide&Deep 好呢？

那这个所谓的“效果”到底指的是什么呢？我们一般用什么方法来衡量这个“效果”呢？我们又应该如何根据效果评估的结果来更新模型呢？这就是模型评估篇要解决的问题。

在所有推荐系统的评估方法中，离线评估是最常用、最基本的。顾名思义，“离线评估”就是我们将模型部署于线上环境之前，在离线环境下进行的评估。由于不用部署到生产环境，“离线评估”没有线上部署的工程风险，也不会浪费宝贵的线上流量资源，而且具有测试时间短，可多组并行，以及能够利用丰富的线下计算资源等诸多优点。

因此，在模型上线之前，进行大量的离线评估是验证模型效果最高效的手段。这节课，我们就来讲讲离线评估的主要方法，以及怎么在 Spark 平台上实现离线评估。

离线评估的主要方法

离线评估的基本原理是在离线环境下，将数据集分为“训练集”和“测试集”两部分，“训练集”用来训练模型，“测试集”用于评估模型。但是如

何划分测试集和训练集，其实这里面有很多学问。我总结了一下，常用的离线评估方法主要有五种，分别是：Holdout 检验、交叉检验、自助法、时间切割、离线 Replay。接下来，我们一一来看。

Holdout 检验、交叉检验和自助法

首先，我们来看 Holdout 检验。 Holdout 检验是最基础，最常用的离线评估方法，它将原始的样本集合随机划分为训练集和测试集两部分，**所以 Holdout 检验的关键词就是“随机”**。举例来说，对于一个推荐模型，我们可以把样本按照 70%-30% 的比例随机分成两部分。其中，70% 的样本用于模型的训练，30% 的样本用于模型的评估。

虽然 Holdout 检验很简单实用，但它的缺点也很明显，就是评估的结果有一定随机性，因为训练集和验证集的划分是随机的，所以如果只进行少量的 Holdout 检验，得到的评估指标会存在一定的波动。那为了消除这种随机性，我们就要使用“交叉检验”的方法。

为了进行交叉检验，我们需要先将全部样本划分成 k 个大小相等的样本子集，然后依次遍历这 k 个子集，每次把当前遍历到的子集作为验证集，其余所有的子集作为训练集，这样依次进行 k 次模型的训练和评估。最后，我们再将所有 k 次评估指标的平均值作为最终的评估指标。在我们的实践中，k 经常取 10，也就是依次进行 10 次检验然后取指标均值。

不管是 Holdout 检验还是交叉检验，都是基于划分训练集和测试集的方法进行模型评估的。然而，当样本规模比较小时，将样本集进行划分会让训练集进一步减小，这往往会影响模型的训练效果。那有没有能维持训练集样本规模的验证方法呢？

“自助法”就可以在一定程度上解决这个问题。我这里所说的**自助法（Bootstrap）是基于自助采样的检验方法**，它的主要过程是：对于总数为 n 的样本集合，我们先进行 n 次有放回地随机抽样，得到大小为 n 的训练集。在 n 次采样过程中，有的样本会被重复采样，有的样本没有被抽出过，我们再将这些没有被抽出的样本作为验证集进行模型验证，这就是自助法的验证过程。

虽然自主法能够保持训练集的规模，但是它的缺点也很明显，它其实改变了原有数据的分布，有可能让模型产生一定程度的偏差。至于，到底是自助采样增加训练样本规模的收益大，还是数据分布被改变带来的损失大，这就需要我们在实践中进行验证了。

时间切割

说完了前三种方法，我们再来看时间切割法。在“模型实战准备（二）”那节课里，我们曾经讲过一个概念，叫“未来信息”。它是说，如果我们在 t 时刻进行模型预测，那么 t+1 时刻的信息就是未来信息。在构建特征工程的时候，我们要避免引入“未来信息”。

其实，在进行模型评估的时候，我们同样不应该在训练集中包含“未来”的样本。怎么理解这句话呢？比如，我们所有的样本数据分布在 t0到 tn这样的时间轴上，如果训练样本是通过随机采样得到的，那么训练数据也会分布在 t0到 tn上，同样，测试数据也会分布在 t0到 tn上。

如果你细想，这个事情其实是有点反常理的。因为训练模型的时候，我们已经使用了 tn这个时间窗口的数据，结果你却用它来预测 t0的事件，这不是很荒谬吗？这就相当于你有一个时光机，已经穿越到了明天，知道股票会涨，结果你又穿越回来，预测说明天股票会涨，这哪是预测呢？这就是“作弊”。

为了防止这类“信息穿越”导致的模型作弊现象发生，我们一般会使用时间切割的方案去划分训练集和测试集，它的做法很简单。比如，你一共处理了 30 天的样本，从第 25 天末开始切割，前 25 天的样本作为训练集，后 5 天的样本作为测试集，这样我们就从根源上切断了引入“未来信息”的可能。当然切割的比例到底如何，也需要根据你的实践来定，一般来说我们控制训练集跟测试集的比例在 3:1 到 10:1 之间，比例太小训练样本不够，比例太大测试结果不够稳定。

离线 Replay

时间切割的方法虽然能避免“信息穿越”，但也不是没有缺点的。它的缺点就在于整个评估过程是静态的，模型不会随着评估的进行而更新，这显然是不符合事实的。就拿我们刚才举的例子来说，用前 25 天的数据做训练集，用后 5 天的数据做测试集。如果在生产环境中，模型是日更新的，那后 5 天的评测过程就不准确，因为在离线测试中，我们并没有在后 5 天的评测过程中做到日更模型。

那怎么解决这个问题呢？我们也可以在离线状态下对线上更新过程进行仿真，让整个评估过程“动”起来。**业界把这样离线仿真式的评估方式叫做离线 Replay。**

下图就是动态的 Replay 评估法与静态的时间分割评估法的对比示意图。我们可以看到，“Replay 评估方法”先根据产生时间对测试样本，由早到晚地进行排序，再让模型根据样本时间的先后进行预测。在模型更新的时间点上，模型需要增量学习更新时间点前的测试样本，更新模型后，再继续评估更新点之后的样本。

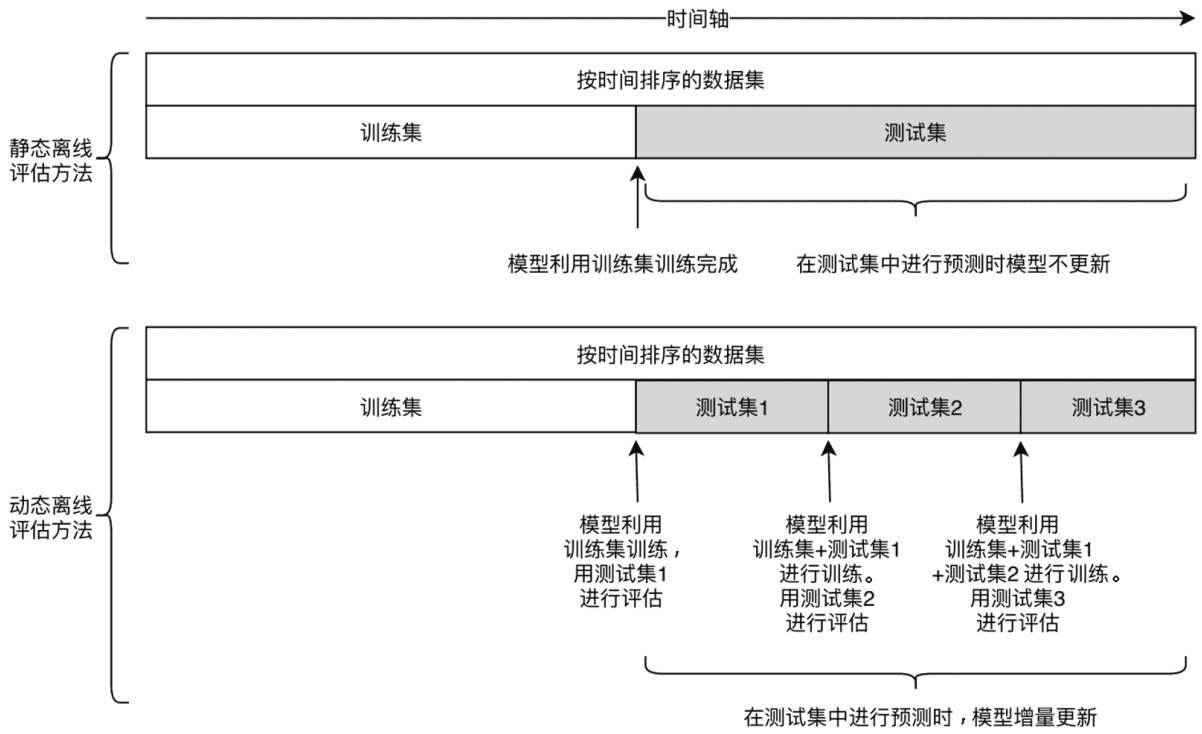

图1 静态时间分割评估与动态Replay评估（出自《深度学习推荐系统》）

你应该也发现了，Replay 评估的过程更接近于真实的线上环境，因为它在线下还原了模型在线上的更新、预估过程。这也让 Replay 方法的评估结果更加权威可信，毕竟，我们最终的目标是让模型在线上产生更好的效果。

当然，Replay 评估方法也有弊端，因为它需要在评估过程中不断更新模型，这让评估过程的工程实现难度加大，因为包含了模型训练的时间，所以整个评估过程的总时长也会加长，影响评估和调参的效率。到底是要评估的准确性，还是要评估的效率，这又是一个需要权衡的问题，我们需要根据自己工程上的侧重点进行选择。

基于 Spark 的离线评估方法实践

熟悉了离线环节的主要模型评估方法，就又到了实践的环节。其实，无论是基于 Python 的 TensorFlow 还是基于 Scala 的 Spark，都有很多支持离线评估的库，这里我们选择了 Spark 进行实践，主要是因为在业界数据集很大的情况下，Spark 在分布式环境下划分训练集和测试集的效率是最高的。

下面，我就来看一下如何使用 Spark 实现 Holdout 检验、交叉检验和时间切割评估法。至于另外两种方法，由于自助法不太常用，离线 Replay 又涉及过多的附加模块，我们暂时就不在项目里实现。

实现 Holdout 检验的时候，我们要清楚如何利用 Spark 随机划分测试集和训练集。它的关键代码只有下面这一行，就是利用 randomSplit 函数把全量样本 samples 按比例分割成 trainingSamples 和 testSamples。在 Spark 的后端，这个 randomSplit 函数会在各个节点分布式执行，所以整个执行效率是非常高的。源代码你可以参考 com.wzhe.sparrowrecsys.offline.spark.featureeng.FeatureEngForRecModel 中的 splitAndSaveTrainingTestSamples 函数。

```
val Array(trainingSamples, testSamples) = samples.
```

实现交叉检验的过程相对比较复杂，好在，Spark 已经提供了交叉检验的接口可以直接使用，我们直接看一下这部分的关键代码。

```
val cv = new CrossValidator()
  .setEstimator(modelPipeline)
  .setEvaluator(new BinaryClassificationEvaluator)
  .setEstimatorParamMaps(paramGrid)
  .setNumFolds(10)  // Use 3+ in practice
val cvModel = cv.fit(training)
```

这段代码中有三个关键参数，一是 setEstimator，这是我们要评估的对象，它需要把我们构建的模型 pipeline 设置进去；二是 setEvaluator，它用来设置评估所用的方法和指标；三是 setNumFolds，它设置的是交叉检验中 k 的值，也就是把样本分成多少份用于交叉检验。本质上 Spark 的 CrossValidator 其实是通过交叉检验来选择模型的最优参数，但也可以通过模型中 cvModel.avgMetrics 参数查看模型的评估指标。

接下来，我们来实现时间切割方法。既然是要按时间划分，如果你知道样本的时间跨度，直接用 where 语句就可以把训练集和测试集划分开了，这也是我最推荐的方法，因为它最高效，不用专门判断切割点。

如果你不知道样本的时间跨度，就要按照时间求取样本的分位数。具体来说就是，通过 Spark 的 approxQuantile 函数，我们可以找到划分样本集为 8:2 的训练集和测试集的时间戳的值。那么接下来我们根据这个值通过 where 语句划分就可以了。我把这个过程的关键代码贴到了下面，供你参考。完整的源代码，你可以参考 com.wzhe.sparrowrecsys.offline.spark.featureeng.FeatureEngForRecModel 中的 splitAndSaveTrainingTestSamplesByTimeStamp 函数。

```
//找到时间切割点
val quantile = smallSamples.stat.approxQuantile("t
val splitTimestamp = quantile.apply(0)
//切割样本为训练集和测试集
val training = smallSamples.where(col("timestampLo
val test = smallSamples.where(col("timestampLong")
```

小结

这节课，我们学习了五种主流的推荐模型离线评估方法，它们分别是

其中，Holdout 检验最简单常用，它通过随机划分的方式把样本集划分成训练集和测试集。而交叉检验的评估效果更加稳定准确，它通过划分样本集为 k 份，再进行 k 次评估取平均的方式得到最终的评估指标。

自助法是为了解决样本量过少而提出的，它可以通过有放回采样的方式扩充训练集，但有改变数据本身分布的风险。而时间切割法在某个时间点上把样本分成前后两份，分别用于模型训练和评估，避免引入未来信息。最后是离线 Replay，它通过仿真线上模型更新过程来进行评估，是最接近线上环境的离线评估方法，但实现起来比较复杂。

总之，各种评估方法都有优有劣，你需要根据实践中的侧重点选择使用，我把它们的优缺点也总结在了文稿的表格里，方便你进行对比。

评估方法	优点	缺点
Holdout检验	简单直接	单次评估结果不稳定，会引入未来信息
交叉检验	多次评估，结果稳定准确	无法解决引入未来信息的问题，实现较复杂
自助法	解决训练样本量少的问题	会改变原数据分布
时间切割法	避免了引入未来信息的问题	与线上真实模型更新过程不符
Replay	最接近线上真实环境， 评估结果更加可信	实现非常复杂

这节课我们讲了评估模型效果的方法之一，离线评估。但我们并没有具体来讲“效果”的衡量指标到底是什么。别着急，下节课我们就来学

习推荐系统主要使用的效果评估指标，也会利用这节课学习到的评估方法来生成这些指标。

课后思考

你觉得离线 Replay 这个方法，跟我们之前讲过的增强学习有什么相似之处吗？你知道它们两个还有什么更深层次的关系吗？

期待在留言区看到你的发现和思考，我们下节课见！