



05 | 特征处理：如何利用Spark解决特征处理问题？

2020-10-12 王喆

深度学习推荐系统实战

[进入课程 >](#)



讲述：王喆

时长 15:07 大小 13.85M



你好，我是王喆。

上节课，我们知道了推荐系统要使用的常用特征有哪些。但这些原始的特征是无法直接提供给推荐模型使用的，因为推荐模型本质上是一个函数，输入输出都是数字或数值型的向量。那么问题来了，像动作、喜剧、爱情、科幻这些电影风格，是怎么转换成数值供推荐模型使用的呢？用户的行为历史又是怎么转换成数值特征的呢？

而且，类似的特征处理过程在数据量变大之后还会变得更加复杂，因为工业界的数据集往往都是 TB 甚至 PB 规模的，这在单机上肯定是没法处理的。那业界又是怎样进行海量的特征处理呢？这节课，我就带你一起来解决这几个问题。

业界主流的大数据处理利器 -Spark

既然要处理海量数据，那选择哪个数据处理平台就是我们首先要解决的问题。如果我们随机采访几位推荐系统领域的程序员，问他们在公司用什么平台处理大数据，我想最少有一半以上会回答是 Spark。作为业界主流的大数据处理利器，Spark 的地位毋庸置疑。所以，今天我先带你了解一下 Spark 的特点，再一起来看怎么用 Spark 处理推荐系统的特征。

Spark 是一个分布式计算平台。所谓分布式，指的是计算节点之间不共享内存，需要通过网络通信的方式交换数据。Spark 最典型的应用方式就是建立在大量廉价的计算节点上，这些节点可以是廉价主机，也可以是虚拟的 docker container（Docker 容器）。

理解了 Spark 的基本概念，我们来看看它的架构。从下面 Spark 的架构图中我们可以看到，Spark 程序由 Manager node（管理节点）进行调度组织，由 Worker Node（工作节点）进行具体的计算任务执行，最终将结果返回给 Drive Program（驱动程序）。在物理的 worker node 上，数据还会分为不同的 partition（数据分片），可以说 partition 是 Spark 的基础数据单元。

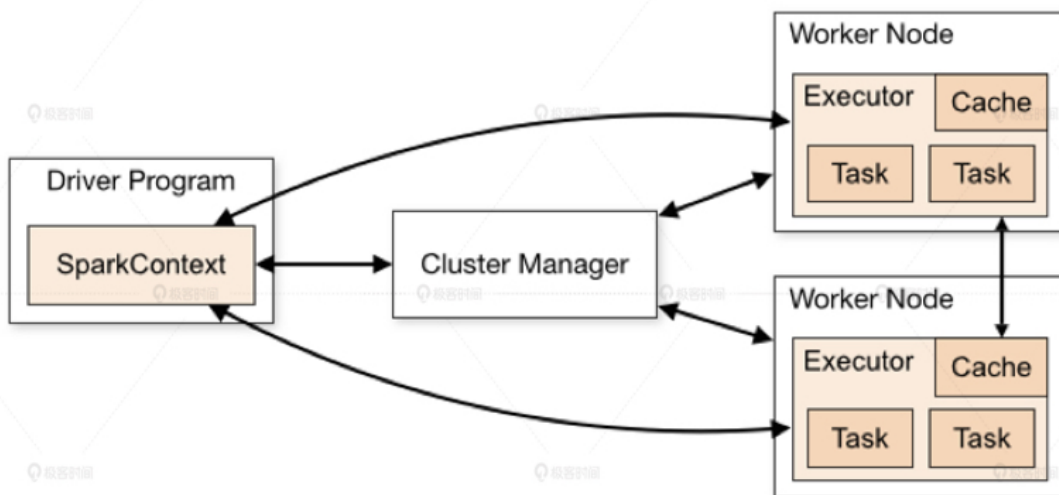


图1 Spark架构图

Spark 计算集群能够比传统的单机高性能服务器具备更强大的计算能力，就是由这些成百上千，甚至达到万以上规模的工作节点并行工作带来的。

那在执行一个具体任务的时候，**Spark 是怎么协同这么多的工作节点，通过并行计算得出最终的结果呢？** 这里我们用一个任务来解释一下 Spark 的工作过程。

这个任务并不复杂，我们需要先从本地硬盘读取文件 `textFile`，再从分布式文件系统 HDFS 读取文件 `hadoopFile`，然后分别对它们进行处理，再把两个文件按照 ID 都 `join` 起来得到最终的结果。

这里你没必要执着于任务的细节，只要清楚任务的大致流程就好。在 Spark 平台上处理这个任务的时候，它会将这个任务拆解成一个子任务 DAG（Directed Acyclic Graph，有向无环图），再根据 DAG 决定程序各步骤执行的方法。从图 2 中我们可以看到，这个 Spark 程序分别从 `textFile` 和 `hadoopFile` 读取文件，再经过一系列 `map`、`filter` 等操作后进行 `join`，最终得到了处理结果。

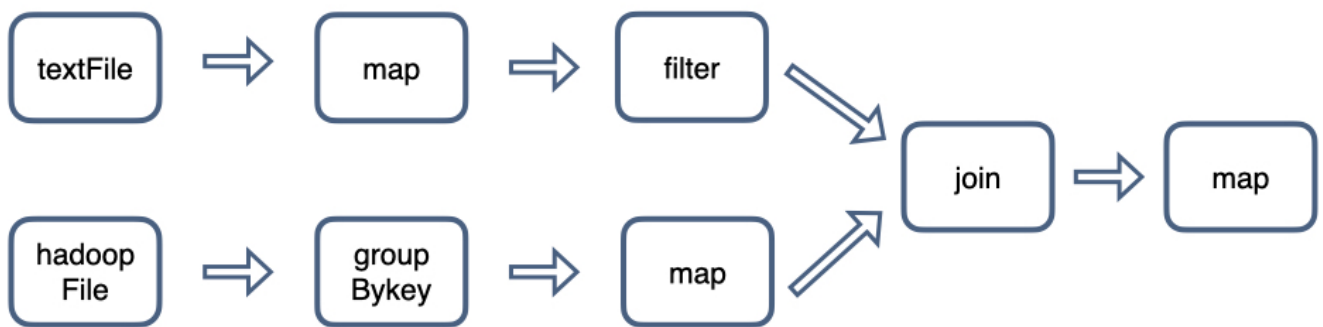


图2 某Spark程序的任务有向无环图

其中，最关键的过程是我们要理解哪些是可以纯并行处理的部分，哪些是必须 `shuffle`（混洗）和 `reduce` 的部分。

这里的 `shuffle` 指的是所有 `partition` 的数据必须进行洗牌后才能得到下一步的数据，最典型的操作就是图 2 中的 `groupByKey` 操作和 `join` 操作。以 `join` 操作为例，我们必须对 `textFile` 数据和 `hadoopFile` 数据做全量的匹配才可以得到 `join` 后的 `dataframe`（Spark

保存数据的结构)。而 `groupByKey` 操作则需要对数据中所有相同的 key 进行合并，也需要全局的 shuffle 才能完成。

与之相比，`map`、`filter` 等操作仅需要逐条地进行数据处理和转换，不需要进行数据间的操作，因此各 partition 之间可以完全并行处理。

此外，在得到最终的计算结果之前，程序需要进行 `reduce` 的操作，从各 partition 上汇总统计结果，随着 partition 的数量逐渐减小，`reduce` 操作的并行程度逐渐降低，直到将最终的计算结果汇总到 master 节点（主节点）上。可以说，shuffle 和 `reduce` 操作的触发决定了纯并行处理阶段的边界。

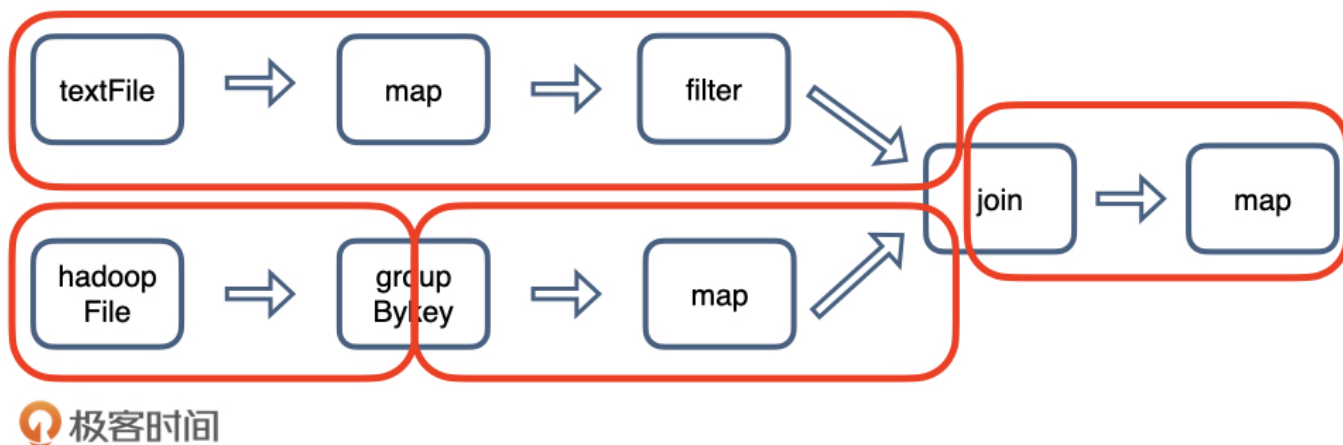


图3 被shuffle操作分割的DAG stages

除此之外，我还想强调的是，shuffle 操作需要在不同计算节点之间进行数据交换，非常消耗计算、通信及存储资源，因此 shuffle 操作是 spark 程序应该尽量避免的。

说了这么多，这里我们再用一句话总结 Spark 的计算过程：**Stage 内部数据高效并行计算，Stage 边界处进行消耗资源的 shuffle 操作或者最终的 reduce 操作。**

清楚了 Spark 的原理，相信你已经摩拳擦掌期待将 Spark 应用在推荐系统的特征处理上了。下面，我们就进入实战阶段，用 Spark 处理我们的 Sparrow Recsys 项目的数据集。

在开始学习之前，我希望你能带着这 2 个问题，边学边思考：经典的特征处理方法有什么？以及 Spark 是如何实现这些特征处理方法的？

如何利用 One-hot 编码处理类别型特征

广义上来讲，所有的特征都可以分为两大类。第一类是**类别、ID 型特征（以下简称类别型特征）**。拿电影推荐来说，电影的风格、ID、标签、导演演员等信息，用户看过的电影 ID、用户的性别、地理位置信息、当前的季节、时间（上午，下午，晚上）、天气等等，这些无法用数字表示的信息全都可以被看作是类别、ID 类特征。第二类是**数值型特征**。简单来说就是能用数字直接表示的特征就是数值型特征，典型的包括用户的年龄、收入、电影的播放时长、点击量、点击率等。

我们进行特征处理的目的，是把所有的特征全部转换成一个数值型的特征向量，对于数值型特征，这个过程非常简单，直接把这个数值放到特征向量上相应的维度上就可以了。但是对于类别、ID 类特征，我们应该怎么处理它们呢？

这里我们就要用到 One-hot 编码（也被称为独热编码），它是将类别、ID 型特征转换成数值向量的一种最典型的编码方式。它通过把所有其他维度置为 0，单独将当前类别或者 ID 对应的维度置为 1 的方式生成特征向量。这怎么理解呢？我们举例来说，假设某样本有三个特征，分别是星期、性别和城市，我们用 [Weekday=Tuesday, Gender=Male, City=London] 来表示，用 One-hot 编码对其进行数值化的结果。

[0,1,0,0,0,0,0]	[0,1]	[0,0,1,0,...,0,0]
Weekday=Tuesday	Gender=Male	City=London

图4 One-hot编码特征向量

从图 4 中我们可以看到，Weekday 这个特征域有 7 个维度，Tuesday 对应第 2 个维度，所以我把对应维度置为 1。而 Gender 分为 Male 和 Female，所以对应的 One-hot 编码就有两个维度，City 特征域同理。

除了这些类别型特征外，ID 型特征也经常使用 One-hot 编码。比如，在我们的 SparrowRecsys 中，用户 U 观看过电影 M，这个行为是一个非常重要的用户特征，那我们应该如何向量化这个行为呢？其实也是使用 One-hot 编码。假设，我们的电影库中一共有 1000 部电影，电影 M 的 ID 是 310（编号从 0 开始），那这个行为就可以用一个 1000 维的向量来表示，让第 310 维的元素为 1，其他元素都为 0。

下面，我们就看看 SparrowRecsys 是如何利用 Spark 完成这一过程的。这里，我们使用 Spark 的机器学习库 MLlib 来完成 One-hot 特征的处理。

其中，最主要的步骤是，我们先创建一个负责 One-hot 编码的转换器，OneHotEncoderEstimator，然后通过它的 fit 函数完成指定特征的预处理，并利用 transform 函数将原始特征转换成 One-hot 特征。实现思路大体上就是这样，具体的步骤你可以参考我下面给出的源码：

[复制代码](#)

```
1 def oneHotEncoderExample(samples:DataFrame): Unit = {
2   //samples样本集中的每一条数据代表一部电影的信息，其中movieId为电影id
3   val samplesWithIdNumber = samples.withColumn("movieIdNumber", col("movieId"))
4
5
6   //利用Spark的机器学习库Spark MLlib创建One-hot编码器
7   val oneHotEncoder = new OneHotEncoderEstimator()
8     .setInputCols(Array("movieIdNumber"))
9     .setOutputCols(Array("movieIdVector"))
10    .setDropLast(false)
11
12
13   //训练One-hot编码器，并完成从id特征到One-hot向量的转换
14   val oneHotEncoderSamples = oneHotEncoder.fit(samplesWithIdNumber).trans
15   //打印最终样本的数据结构
16   oneHotEncoderSamples.printSchema()
17   //打印10条样本查看结果
18   oneHotEncoderSamples.show(10)
19
20   _ (参考 com.wzhe.sparrowrecsys.offline.spark.featureeng.FeatureEngineering__中的
```

One-hot 编码也可以自然衍生成 Multi-hot 编码（多热编码）。比如，对于历史行为序列类、标签特征等数据来说，用户往往会与多个物品产生交互行为，或者一个物品被打上多个标签，这时最常用的特征向量生成方式就是把其转换成 Multi-hot 编码。在 SparrowRecsys 中，因为每个电影都是有多个 Genre（风格）类别的，所以我们可以用

Multi-hot 编码完成标签到向量的转换。你可以自己尝试着用 Spark 实现该过程，也可以参考 SparrowRecsys 项目中 multiHotEncoderExample 的实现，我就不多说啦。

数值型特征的处理 - 归一化和分桶

下面，我们再好好聊一聊数值型特征的处理。你可能会问了，数值型特征本身不就是数字吗？直接放入特征向量不就好了，为什么还要处理呢？

实际上，我们主要讨论两方面问题，一是特征的尺度，二是特征的分布。

特征的尺度问题不难理解，比如在电影推荐中有两个特征，一个是电影的评价次数 fr，一个是电影的平均评分 fs。评价次数其实是一个数值无上限的特征，在 SparrowRecsys 所用的 movieLens 数据集上，fr 的范围一般在[0,10000]之间。对于电影的平均评分来说，因为我们采用了 5 分为满分的评分，所以特征 fs 的取值范围在[0,5]之间。

由于 fr 和 fs 两个特征的尺度差距太大，如果我们把特征的原始数值直接输入推荐模型，就会导致这两个特征对于模型的影响程度有显著的区别。如果模型中未做特殊处理的话，fr 这个特征由于波动范围高出 fs 几个量级，可能会完全掩盖 fs 作用，这当然是我们不愿意看到的。为此我们希望把两个特征的尺度拉平到一个区域内，通常是[0,1]范围，这就是所谓**归一化**。

归一化虽然能够解决特征取值范围不统一的问题，但无法改变特征值的分布。比如图 5 就显示了 Sparrow Recsys 中编号在前 1000 的电影平均评分分布。你可以很明显地看到，由于人们打分有“中庸偏上”的倾向，因此评分大量集中在 3.5 的附近，而且越靠近 3.5 的密度越大。这对于模型学习来说也不是一个好的现象，因为特征的区分度并不高。

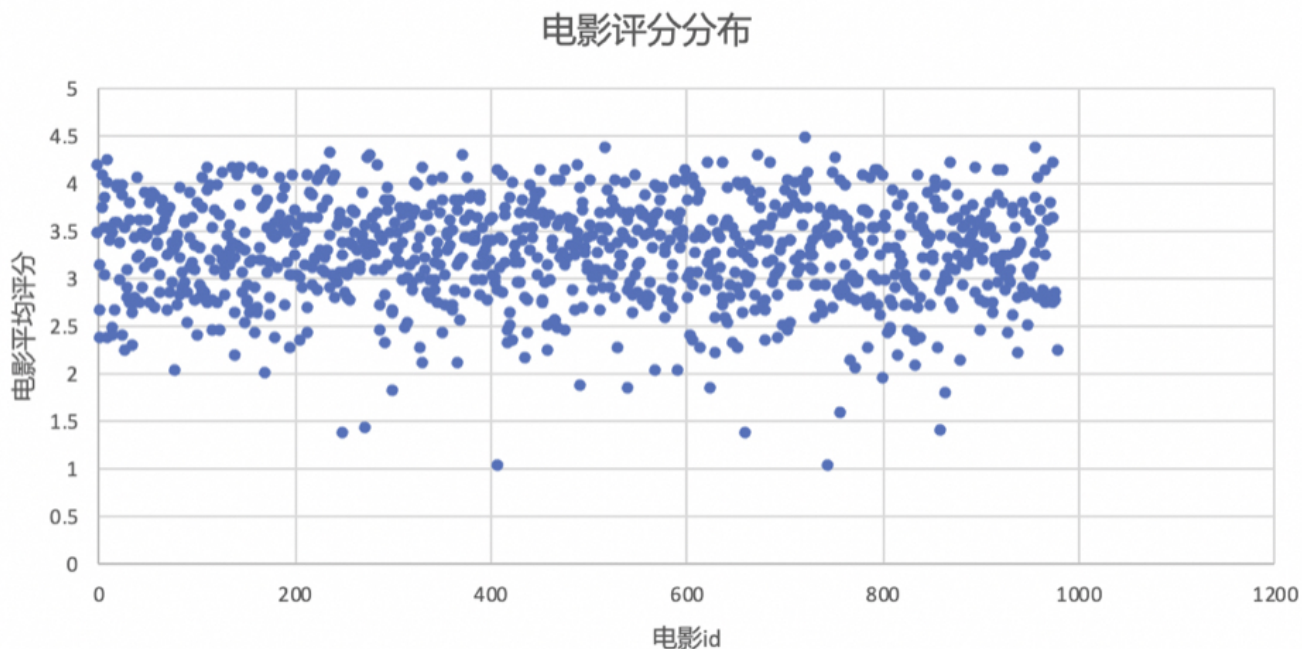


图5 电影的平均评分分布

这该怎么办呢？我们经常会用分桶的方式来解决特征值分布极不均匀的问题。所谓“分桶（Bucketing）”，就是将样本按照某特征的值从高到低排序，然后按照桶的数量找到分位数，将样本分到各自的桶中，再用桶 ID 作为特征值。

在 Spark MLlib 中，分别提供了两个转换器 MinMaxScaler 和 QuantileDiscretizer，来进行归一化和分桶的特征处理。它们的使用方法和之前介绍的 OneHotEncoderEstimator 一样，都是先用 fit 函数进行数据预处理，再用 transform 函数完成特征转换。下面的代码就是 SparrowRecSys 利用这两个转换器完成特征归一化和分桶的过程。

[复制代码](#)

```
1 def ratingFeatures(samples:DataFrame): Unit = {
2   samples.printSchema()
3   samples.show(10)
4
5
6   //利用打分表ratings计算电影的平均分、被打分次数等数值型特征
7   val movieFeatures = samples.groupBy(col("movieId"))
8     .agg(count(lit(1)).as("ratingCount"),
9         avg(col("rating")).as("avgRating"),
10        variance(col("rating")).as("ratingVar"))
11     .withColumn("avgRatingVec", double2vec(col("avgRating")))
12
13
14   movieFeatures.show(10)
15 }
```



```
16 //分桶处理, 创建QuantileDiscretizer进行分桶, 将打分次数这一特征分到100个桶中
17 val ratingCountDiscretizer = new QuantileDiscretizer()
18   .setInputCol("ratingCount")
19   .setOutputCol("ratingCountBucket")
20   .setNumBuckets(100)
21
22
23 //归一化处理, 创建MinMaxScaler进行归一化, 将平均得分进行归一化
24 val ratingScaler = new MinMaxScaler()
25   .setInputCol("avgRatingVec")
26   .setOutputCol("scaleAvgRating")
27
28
29 //创建一个pipeline, 依次执行两个特征处理过程
30 val pipelineStage: Array[PipelineStage] = Array(ratingCountDiscretizer, ratingScaler)
31 val featurePipeline = new Pipeline().setStages(pipelineStage)
32
33
34 val movieProcessedFeatures = featurePipeline.fit(movieFeatures).transform(movieFeatures)
35 //打印最终结果
36 movieProcessedFeatures.show()
37
38 _ (参考 com.wzhe.sparrowrecsys.offline.spark.featureeng.FeatureEngineering中的ratingCountDiscretizer和ratingScaler)
39
```

当然，对于数值型特征的处理方法还远不止于此，在经典的 YouTube 深度推荐模型中，我们就可以看到一些很有意思的处理方法。比如，在处理观看时间间隔（time since last watch）和视频曝光量（#previous impressions）这两个特征的时，Youtube 模型对它们进行归一化后，又将它们各自处理成了三个特征（图 6 中红框内的部分），分别是原特征值 x ，特征值的平方 x^2 ，以及特征值的开方，这又是为什么呢？

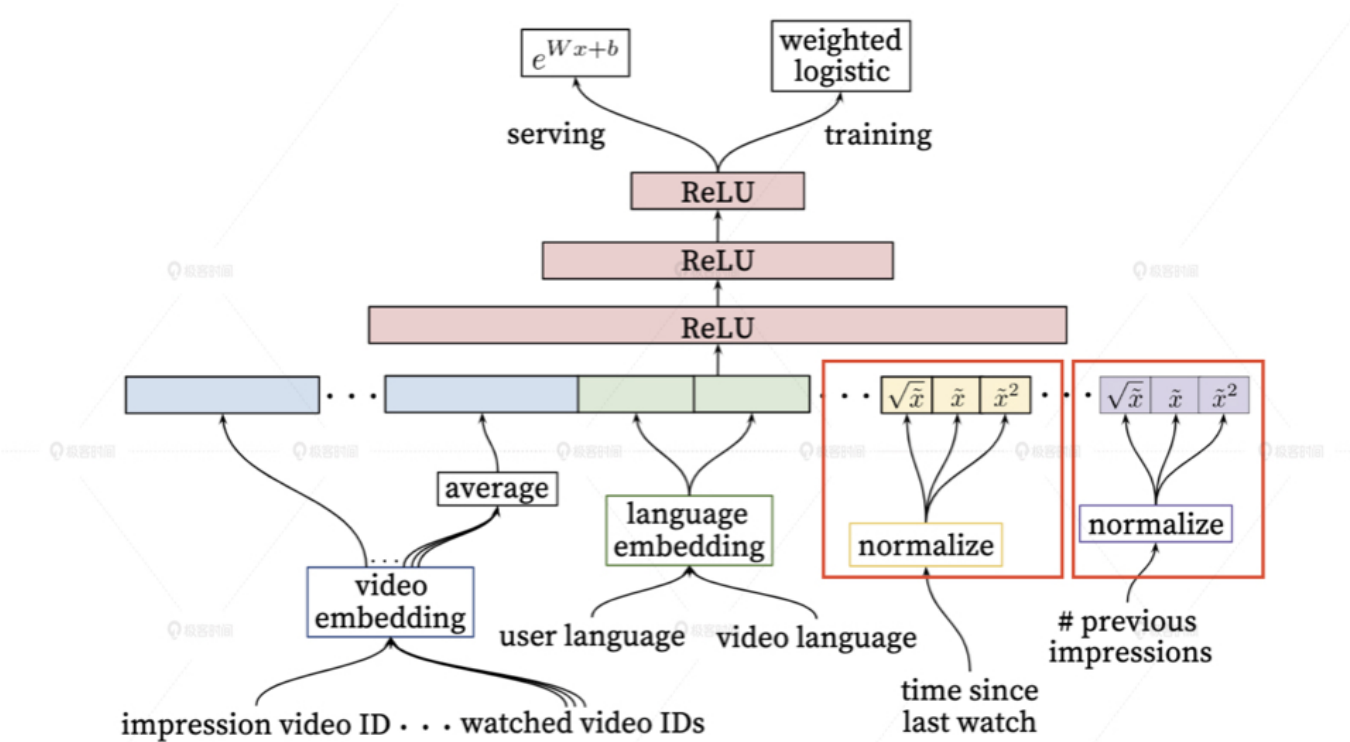


图6 YouTube推荐模型（来源：Deep Neural Networks for YouTube Recommendations）

其实，无论是平方还是开方操作，改变的还是这个特征值的分布，这些操作与分桶操作一样，都是希望通过改变特征的分布，让模型能够更好地学习到特征内包含的有价值信息。但由于我们没法通过人工的经验判断哪种特征处理方式更好，所以索性把它们都输入模型，让模型来做选择。

这里其实自然而然地引出了我们进行特征处理的一个原则，就是**特征处理并没有标准答案**，不存在一种特征处理方式是一定好于另一种的。在实践中，我们需要多进行一些尝试，找到那个最能够提升模型效果的一种或一组处理方式。

小结

这节课我们介绍了推荐系统中特征处理的主要方式，并利用 Spark 实践了类别型特征和数值型特征的主要处理方法，最后我们还总结出了特征处理的原则，“特征处理没有标准答案，需要根据模型效果实践出真知”。

针对特征处理的方法，深度学习和传统机器学习的区别并不大，TensorFlow、Pytorch 等深度学习平台也提供了类似的特征处理函数。在今后的推荐模型章节我们会进一步用到这些方法。

最后，我把这节课的主要知识点总结成了一张表格，你可以利用它巩固今天的重点知识。

知识点	要点总结
Spark的运行原理	Spark是业界主流的分布式计算平台，通过创建任务DAG图的方式并行执行数据处理任务。
类别型特征处理	One-hot编码、Multi-hot编码
数值型特征处理	归一化、分桶，及其他改变特征值分布的处理方式
特征处理的原则	特征处理没有标准答案，需要根据模型效果实践出真知



这节课是我们的第一堂实战课，对于还未进入到工业界的同学，相信通过这节课的实践，也能够一窥业界的大数据处理方法，增强自己的工程经验，让我们一起由此迈入工业级推荐系统的大门吧！

课后思考

1. 请你查阅一下 Spark MLlib 的编程手册，找出 Normalizer、StandardScaler、RobustScaler、MinMaxScaler 这几个特征处理方法有什么不同。
2. 你能试着运行一下 SparrowRecSys 中的 FeatureEngineering 类，从输出的结果中找出，到底哪一列是我们处理好的 One-hot 特征和 Multi-hot 特征吗？以及这两个特征是用 Spark 中的什么数据结构来表示的呢？

这就是我们这节课的全部内容了，你掌握得怎么样？欢迎你把这节课转发出去。下节课我们将讲解一种更高阶的特征处理方法，它同时也是深度学习知识体系中一个非常重要的部分，我们到时候见！

提建议

更多学习推荐

机器学习训练营

成为能落地的实干型机器学习工程师

王然 众微科技 AI Lab 负责人

前100名秒杀 ¥3649  加赠书籍

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 04 | 特征工程：推荐系统有哪些可供利用的特征？

下一篇 06 | Embedding：所有人都在谈的Embedding技术到底是什么？

精选留言 (11)

 写留言**JustBuyIt**

2020-10-12

Normalizer、StandardScaler、RobustScaler、MinMaxScaler 都是用让数据无量纲化
Normalizer: 正则化；（和Python的sklearn一样是按行处理，而不是按列[每一列是一个特征]处理，原因是：Normalization主要思想是对每个样本计算其p-范数，然后对该样本中每个元素除以该范数，这样处理的结果是使得每个处理后样本的p-范数(l1-norm, l2-norm)等于1。）针对每行样本向量：l1: 每个元素/样本中每个元素绝对值的和，l2: 每个元...
展开 ∨

作者回复：非常棒，推荐其他同学参考。

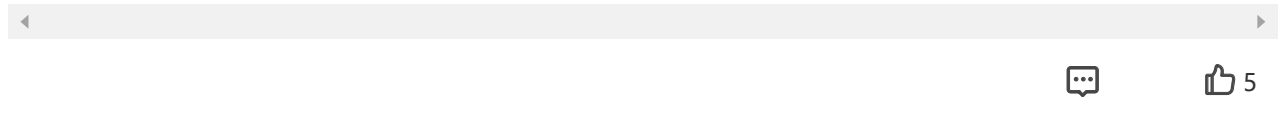
 3 10**JustBuyIt**

2020-10-12

Multiple编码

顾名思义，Multiple编码特征将多个属性同时编码到一个特征中。在推荐场景中，单个用户对哪些物品感兴趣的特征就是一种Multiple编码特征，如，表示某用户对产品1、产品2、产品3、产品4是否感兴趣，则这个特征可能有多个取值，如用户A对产品1和产品2感兴趣，用户B对产品1和产品4感兴趣，用户C对产品1、产品3和产品4感兴趣，则用户兴...
展开 ✓

作者回复: 不错的文章，也推荐大家学习。



fsc2016

2020-10-16

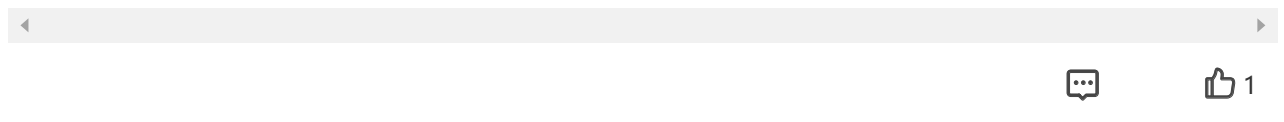
第二个问题：

One-hot特征是调用OneHotEncoderEstimator对movieId转换，生成了特征movieIdVector

Multi-hot 特征是调用Vectors.sparse方法，对处理后的genreIndexes转换，生成vector。 ...

展开 ✓

作者回复: 非常好。比之前的回答更准确一些，推荐大家参考。



杨佳亦

2020-10-16

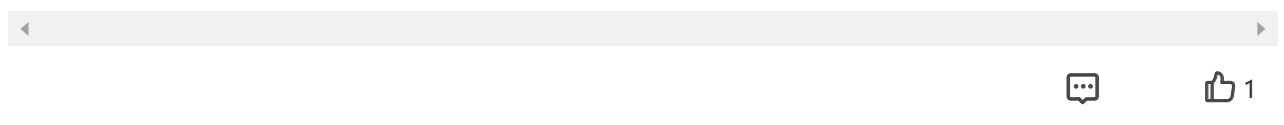
MinMaxScaler:

记录数据整体的最大/最小值为max/min, 对于Feature E, 若 $E_{max} \neq E_{min}$, 则rescale后值 $= (E_i - E_{min}) / (E_{max} - E_{min}) * (max - min) + min$. 若 $E_{max} == E_{min}$, 则rescale后值 $= 0.5 * (min + max)$.

...

展开 ✓

作者回复: 非常详细了，也推荐在spark中自己进行实践。



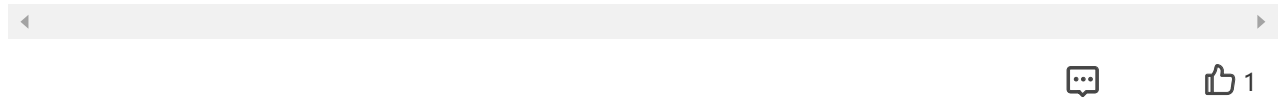
张弛 Conor

2020-10-14

请教老师，像在电影评分这样的离散数值(且比较稀疏)例子中，如果需要取得分桶数较多，而导致分位数附近均是同一数值的情况下，如何使用分桶的方法呢？
比如按照分桶法首先排序得到评分为5,5,4,4,4,4,4,4,3,3,3,3,2,2,1(共15个)。取桶数为3时，第一个桶内有前两个5，而后面的6个4中应该选择哪3个来分到第一个桶呢？

展开 ∨

作者回复: 其实不建议这种离散数值，取值数量有比较少的特征进行分桶操作。把相同分值强制分到两个桶里，不仅没有意义，而且引入噪声。



smjccj

2020-10-12

Normalizer: 转化为方差为1，不影响数据分布

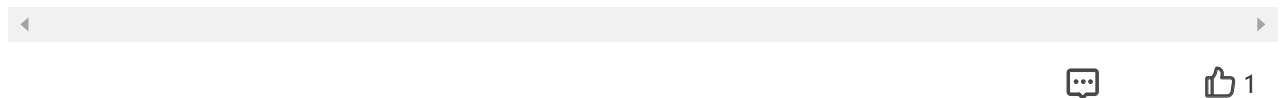
StandardScaler: 将数据转化为 variance= 1, mean =0(unit variance and/or zero mean)

RobustScaler: 中位数或四分位数进行缩放，处理异常值带来的影响

MinMaxScaler: 将特征缩放至0-1之间

展开 ∨

作者回复: 赞



阳光

2020-10-16

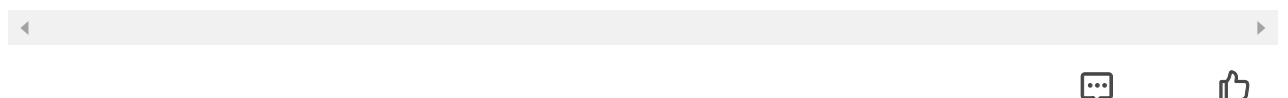
我运行FeatureEngineering 类出错，显示 “错误: 找不到或无法加载主类 com.wzhe.sparrowrecsys.offline.spark.featureeng.FeatureEngineering

原因: java.lang.ClassNotFoundException: com.wzhe.sparrowrecsys.offline.spark.featureeng.FeatureEngineering”

这是怎么回事？

展开 ∨

作者回复: 按照之前类似的问题，应该是scala版本的问题，请查一下是不是使用了scala2.11的版本。



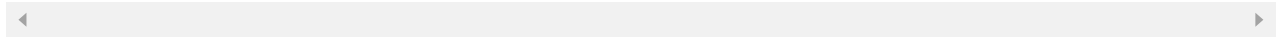
李@君

2020-10-16

对训练数据进行平方或者开方，是为了改变训练数据的分布。训练数据的分布被改变后，训练出来的模型岂不是不能正确拟合训练数据了。

展开 ∨

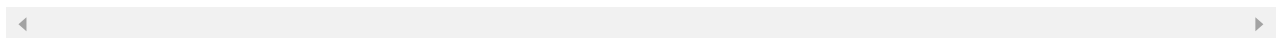
作者回复: 这是个好问题。但不应该这样理解，本质上是改变了特征的分布，特征的分布和训练数据的分布没有本质的联系。只要你不改变训练数据label的分布，最终预测出的结果都应该是符合数据本身分布的。因为你要预测的是label，并不是特征本身。

**浣熊当家**

2020-10-16

老师我们是不是一般用到spark相关的包，就会一定要用scala？可不可以也用java呢？或者比如线上部分的代码现在用的是java，可不可以用scala呢？我们有哪些选择语言的标准和规律吗？

作者回复: scala和java都可以，pyspark也可以，看自己的喜好。因为spark原生支持scala，属于最官方的语言，所以本门课程使用scala

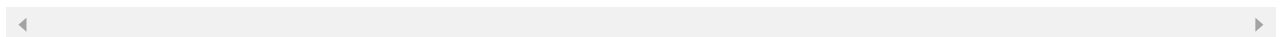
**Geek_63ee39**

2020-10-12

思考2: movieIdVector是处理好的One-hot 特征；vector是处理好的Multi-hot 特征；这两个特征是使用Vector表示的

展开 ∨

作者回复: 是这样，希望大家都能运行程序查看一下数据细节。

**吴波**

2020-10-12

想问下为什么右键运行FeatureEngineering会报错，提示 错误: 找不到或无法加载主类 com.wzhe.sparrowrecsys.offline.spark.featureeng.FeatureEngineering

展开 ∨

作者回复: 应该是环境问题造成的个例。建议自己debug一下。

