特征工程｜数据清洗、特征生成、特征拼接

原创 Thinkgamer 搜索与推荐Wiki 2020-08-07

收录于话题

#Thinkgamer 11 #特征工程 92 #精品小系列内容 25

特征工程的完整流程是：特征设计 -> 特征获取 -> 特征处理 -> 特征存储 -> 特征监控。前边介绍了那么多，相当于是对特征设计、特征获取、特征存储进行了说明，而特征工程中最重要的环节则是特征处理。特征处理中还包括：数据清洗、特征生成、特征拼接、特征处理、特征转换、特征选择。本篇主要介绍数据清洗、特征生成、特征拼接。

> 文章较长，建议【先收藏再细品】，点击文末【阅读原文】查看更多精彩内容

数据清洗

从特征工程角度讲，数据清洗是特征工程的前置阶段（但是也会贯穿整个数据应用过程），其本义是对数据进行重新的审查和校验，目的在于删除重复信息、纠正存在的错误数据，并保证数据的一致性。数据清洗是整个数据分析过程中不可缺少的一个环节，其结果质量直接关系到模型效果和最终结论。

一个特征处理的完整流程可以表示为：

因此基础数据的准确性、完备性、一致性决定了后续特征数据的有效性。在我们日常使用的公开数据集中，很多都是已经被处理后的了，比如学术界中使用很广泛的MovieLens数据集，但是在真实的业务场景中，我们拿到的数据其实直接是没有办法使用的，可能包含了大量的缺失

值，可能包含大量的噪音，也可能因为人工录入错误导致有异常点存在，对我们挖据出有效信息造成了一定的困扰，所以我们需要通过一些方法，尽量提高数据的质量。

初期数据清洗更多的是针对单条样本数据的清洗和检测，包括：

- 数据表示一致性处理
- 逻辑错误值处理
- 缺失值处理
- 非必要性数据剔除

数据表示一致性处理

在实际的业务场景中，数据是由系统收集或用户填写而来，有很大可能性在格式和内容上存在一些问题，同样类型的数据在不同的团队上报过程中产出的内容或格式会不一致，同样不同数据源采集而来的数据内容和格式也会不一致。

数据格式的一致性。比如时间信息（以2020年6月18日，11点11分12秒为例），有的用毫秒时间戳表示（1592449872000），有的用秒时间戳表示（1592449872），有的用带横线的时间表示（2020-06-18 11:11:12），有的则用不带横线的时间表示（20200618 11:11:12）。

数据类型的一致性。比如不同的数据表中，同样表示用户ID的字段，有的可能是string类型，有的是int类型。如果用string类型表示，如果用户id缺失的话，正常可以留空，但是不同团队，不同人对于缺失的id处理方式也会不一致，比如有的用None，有的用Null，有的则用0表示，可谓是千奇百怪。小编在日常工作中也会经常遇到这种情况，被折磨的体无完肤。

所以在实际处理时，针对某一类型的数据约定特征的格式和规范，后续样本数据均采用该格式和规范进行处理，继而保证数据的一致性。特别是对于一些基础性的数据，这样可以减小数据在使用过程中的问题交流时间和避免出现一些不必要的麻烦。

逻辑错误值处理

逻辑错误值在实际的业务场景中出现的情况还是比较多的，在实际操作中要酌情处理。该步骤不仅在特征处理之前会进行，在整个特征工程或者数据分析的过程中都会存在，因为针对单条样本的处理，很难保证全局的准确性和一致性，我们能做的就是在每一步做好控制，尽量减少问题出现的可能性，使得产出的数据更准确。

数据重复。比如用户唯一表示，在登陆之前可能使用一个设备ID进行表示，登陆之后则会使用用户ID进行表示（登陆之后正常的数据操作是：设备ID产出的数据合并到用户ID上），但是如果操作不注意则数据中则会出现两条数据来表示一个用户，且因为用户登陆之后是以用户ID为准的，那么以设备ID表示的数据在一定程度上就失去了其时效性。

不合理值处理。比如用户的年龄，人的正常年龄应该是在0-100以内（这里不要进行道德绑架，只是正常的年龄范围，先忽略掉100岁以上的可敬可爱的爷爷奶奶们，并无歧视），但对于一

个具体的平台而言，年龄还可以进行更准确的划分，比如一个1岁的宝宝会去注册某个视频平台的账号吗，同样受科技发展的影响，一个100岁的老人还会不停的刷短视频吗（或许几十年后可能）！所以在处理用户的年龄数据时就是针对不合理的数据进行剔除或者修正。同样还有其他的许多例子，比如用户住址填写的是外太空，用户的籍贯写的是汉族，用户的收入写个200万等。这些都可以通过预先设置的经验阈值、或统计分析方法进行处理，只不过统计分析的话要涉及所有的用户数据，后边的特征处理会进行介绍。

矛盾内容修正。比如一个金融平台，在收集用户年龄和身份证号时是可以进行相互验证的。这个时候需要判断数据来源的准确性，继而推断出用户年龄的准确值。

缺失值处理

缺失值是特征工程中比较常见的一个问题，其主要包含两个方面的内容：

- 确定缺失值范围
- 缺失值处理

1、确定缺失值范围

结构化数据分为：

- 离散型数据：一个有限的数据集合，比如性别。
- 连续型数据：一个无限的数据集合，数据可以有上下限，也可以没有上下限，比如身高、人口数量等

不同类型的数据缺失值范围是不同的，比如离散型数据，是有一个数据集合，像性别分男女，主流的手机系统分Android、iOS。但对于连续型数据我们并不能得到一个数据集合，但是对于一些限定条件的数据是可以确定数据的上下限的，比如一个商品一天内的UV，其下限为0，上限则是平台总的用户数。

确定数据的缺失值范围之后，还要对数据的缺失比例和重要性做一个评估，以此来判断不同缺失值的处理方式。下图为一个简单的象限图，可以作为一个参考：

2、缺失值处理

在上文的图中也说明了数据重要性和缺失率不同组合情况的对应处理办法。在具体是应用中，要灵活运用和处理，更多关于缺失值处理的方法，参考后文的特征处理部分。

非必要性数据剔除

非必要性数据剔除是指的对于一些准确率不高的字段值或者对于最终的模型训练没有太大帮助的字段值进行删除。比如某平台用户属性信息中的生日字段，假设其中超过半数使用的都是系统的默认值，那么这个字段数据其实就是无效的，因为我们并不能保证没有使用默认值的生日数据也是准确的。

在删除数据的时候需要注意的是：删除操作不要在原始数据上进行操作（这应该是算法、数据从业者的基本素养了），最好是抽取数据之后形成新的数据文件，继而进行后续的分析和建模。删除操作简单但要慎用，不过一般效果不错，删除一些丢失率高以及重要性低的数据可以降低模型的训练复杂度同时又不会降低模型效果。

特征生成

对基础数据进行清理之后需要做的就是生成我们需要的特征，在特征设计部分提到特征主要分为四大维度，根据小编的经验特征又可以根据其值的属性划分为：

- 类别特征：即特征的属性值是一个有限的集合，比如用户性别、事物的类别、事物的ID编码类特征等
- 连续特征：即用户行为、类别、组合特征之类的统计值，比如用户观看的视频部数、某类别下事物的个数等
- 时间序列特征：即和时间相关的特征，比如用户来访时间、用户停留时长、当前时间等。
- 组合特征：即多种类别的组合特征，比如用户在某个类别下的行为统计特征、当天内事物被访问次数特征等
- 文本特征：即和文本相关的特征，比如评论数据、商品描述、新闻内容等。
- Embedding特征：即一些基础特征的高层次表示，比如用户ID编码的Embedding表示、事物ID编码的Embedding表示、用户访问事物序列的Embedding编码等。

当然不同种类特征之间可能会有交叉，比如组合特征可能是类别特征和连续特征的组合，时间序列特征也会和连续特征有交叉，这里不必咬文嚼字，只需明白我们在日常工作中经常使用的特征类型和处理方式即可。

基于基础数据加工特征的时候，可以根据以上几个维度进行相关的特征生成，当然并不局限于以上几种，而且不同的业务场景中也有所不同，上文所提到的只是绝大多数情况下建模需要构造的特征种类。比如一些特定场景的特征（比如图像、视觉等领域），上述的特征划分就没有那么适用了。

特征拼接

这里的特征拼接表示的是将几种类型的特征（用户维度特征、事物维度、特征类别维度、特征组合属性特征）拼接在一起，构造出初步的特征数据，继而进行后续的特征处理、特征转换和

特征选择等操作。

比如电商CTR场景中预测用户对某商品的点击率。可以将用户维度的属性特征、该商品维度的属性特征、该商品所属类别（店铺、品牌等）维度的属性特征、该商品相关组合属性特征（用户x该商品所属类别、用户x该商品所属品牌等）拼接在一起，构造成<features, label>形式的数据，这里的label如果用户点击了该商品则为1，否则为0。

The end

分享，点赞，在看，安排一下？

收录于话题 #特征工程·92个

上一篇

特征工程｜类别特征的常见处理方式（含代码）

下一篇

特征工程｜特征获取、特征规范和特征存储

阅读原文