R&S | 手把手搞推荐[2]：特征工程指南

原创 机智的叉烧 CS的陋室 2019-05-11

Shell Shocked

Wiz Khalifa - The Taylors Family Business

【R&S】

本栏目从原来的【RS】改为【R&S】，意为“recommend and search”，即推荐和搜索，结合本人近期的工作方向、最近上的七月在线的课、自己自学、而推出的特别栏目。当然，按照我往期的风格，更加倾向于去讨论一些网上其实讲得不够的东西，非常推荐大家能多看看并且讨论，欢迎大家给出宝贵意见，觉得不错请点击推文最后的好看，感谢各位的支持。

另外，【手把手搞推荐】是我近期开始的连载，结合自己所学，带上代码的手把手和大家分享一些模型和数据处理方式，欢迎关注。

往期回顾：

- R&S | 手把手搞推荐[0]：我的推荐入门小结
- R&S | 手把手搞推荐[1]：数据探索
- R&S | 爱奇艺搜索启发
- RS | 推荐系统整体设计
- RS | 论文阅读：用于YouTube推荐的深度神经网络

看了[1]的阅读量，有点失望，也不知道这次会不会更低，不知道是不是因为数据探索大家都没有重视，在我看来吧，其实非常重要，也没有想的那么简单，否则我也不会花几千字在上面(字数统计是3000+)，相信我，数据探索绝对是有学问在里面的，好的数据探索会为你后续的特征工程和建模带来巨大帮助，如果觉得我说的有道理且前面的没看，下面给你传送门。

R&S | 手把手搞推荐[1]：数据探索

数据探索之后，还没到模型呢(着急啥呢)，进入模型之前还需要对特征进行一系列的整理和计算，这样进入模型才会有比较好的效果，甚至能一次过，这样自信回头的样子超级酷不是么，所谓磨刀不误砍柴工，

相信我，直接怼模型换模型的成本绝对不会高于你在特征工程做的一系列努力，所以，我们开始吧。

懒人目录

特征工程的结构

特征工程一般出现在泛特征问题上，就是特征是大量、格式丰富、多样化的特征，和图像、NLP等问题非常不同，可以通过embedding等方式处理，推荐系统由于其特征的特点，对特征处理的要求很高。特征工程是对特征进行一系列处理的流程，把原始的数据信息转化为**有利于计算的形式**，这是一个非常巧妙的词，概念比较模糊的词，说白了特征工程主要的目标就是为了模型有更好的结果，怎么有，那就是见仁见智了，我能想到的，主要有下面的工作。

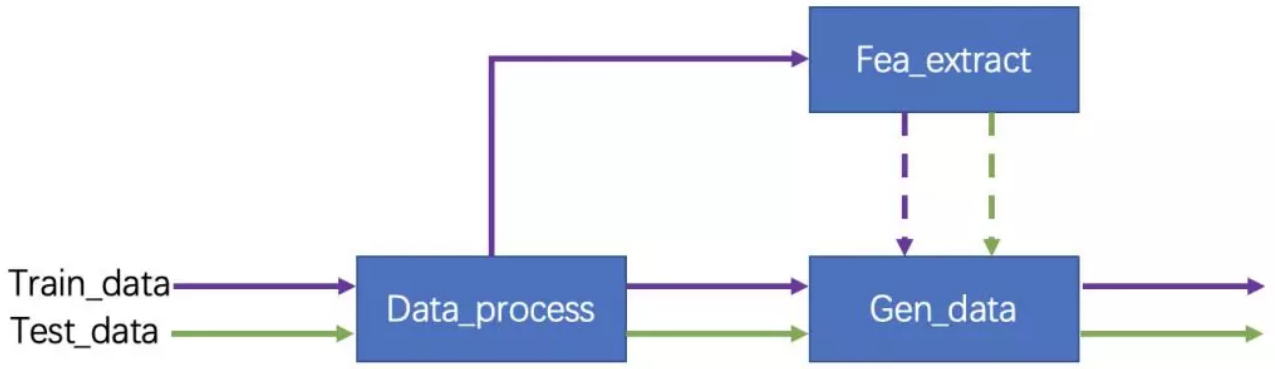
- 错误值、异常值的消除或转化。
- 特征格式转化，如分桶、one-hot、截断，甚至是函数映射等。
- 特征组合，必要时候进行特征的组合会有奇效。

说到是工程，其实就要处理一个很繁杂的问题——**线上线下特征一致性**。模型的试验一般在线下，通过离线方式进行，数据集自己从数据库之类的渠道采集，自己构建数据集，然后开始跑，但是还要面临一个上线的问题，对于同一个服务，线上的数据是按照数据库的格式过来的，要做的处理在线上还要再做一遍，然后才能得到符合你的模型格式的数据格式，这个一致性就是一个非常难的问题，因此线上线下就需要统一一个严格的格式，说白了就是为了保证一种"重现性"。

另外，这种重现性不仅出现在线上线下，还有训练集和测试集上，需要保证的是训练集和测试集经历的是同一种操作，这个一种是一个非常高的要求，来举个栗子吧。

> 一个one-hot的encoder，为了保证训练集和测试集的分离，必须是只能用训练集来训练，训练以后用这个encoder要同时处理训练集和测试集的one-hot化。这个看起来简单，但是实际上，很多人会在one-hot encoder训练时同时用训练集和测试集，训练完之后再来划分数据集，这个并不合适。

说完了几个难点，我直接给出我现在用的解决方案吧~之前已经在其它问题上经过实测，还不错，这次我打算继续用，上图。



非常简单粗暴的一张图，但是足够我把他说清楚了哈哈哈。

Dataprocess是一套基本的数据操作，把一些内部的特征提取出来，以及一些异常数据剔除或者转化，做的工作主要就是一些特征上的基本操作，这些操作是**是否把训练集测试集分开都能做**的工作，例如，movielens里把moviename中的年份提取出来的工作，另外还有分桶等，都是可以在这块进行，也可以这么看，这一步是一些对单条数据即可执行的操作，有些并非如此的，例如求总体占比之类的，就不太好在这步进行，而是放在下部。

Fea_extract是特征提取工作，这套工作只需要训练集数据完成即可，目标是构建一些特征提取的算法，例如one-hot、求占比、求特征组合等，都在这块做，最终是在这部就是给每个特征单独构建一个可复用的处理函数。

Gendata是数据生成工作，根据Feaextract中对每个特征处理建立的函数，生成转化后的数据集，这块工作当然训练接集和测试集都需要进行。

代码时间

既然如此，我们就开始实践吧，在上一期，我们已经将数据整理成下面的格式：

> rating::movieId::movieName::genres::userId::gender::age::occupation

来个数据例子：

> 2::3124::Agnes Browne (1999)::Comedy|Drama::5493::M::35::12

由于我这里特征少，且需要在Dataprocess进行的步骤很少，所以不单独构建dataprocess模块了，可在Feaextract手动进行，单后在Gendata阶段再进行即可，时间不会多花很多。

这里先讲大部分流程，具体技术细节和要强调的点会另外开一节说~

Fea_extract阶段

特征工程，我想做的事情有这些：

- 把非数值型特征全部转为one-hot格式(有些用1，2，3表示的，其实也是非数值型，只是用数字表示罢了，这个要注意区分)
- 从movieName中提取年份特征
- genres特征下可能有多标签，例如上面的例子，电影是Comedy和Drama两种类型。

好了，目标明确，开始踩技术点。

- onehot，可用sklearn.preprocessing.OneHotEncoder进行，然后用sklearn.externals.joblib保存和后续加载
- movieName中年份的提取，可用正则从括号内提取年份
- 在不考虑组合特征的前提下，多标签对onehot encoder的建立本身并没有太大影响。

技术点踩好了，动手吧。

```
data = {}
data["scores"] = []
data["movie_id"] = []
data["movie_year"] = []
data["movie_type"] = []
data["user_id"] = []
data["user_gentle"] = []
data["user_age"] = []
data["user_occupation"] = []

with open(SOURCE_DATA_PATH) as f:
    idx = 0
    for line in f:
        if idx == 0:
            idx = 1
            continue
        ll = line.strip().split("::")
        data["scores"].append([ll[0]])
        data["movie_id"].append([ll[1]])
        data["movie_year"].append([get_year(ll[2])])
        for item in ll[3].split("|"):
            data["movie_type"].append([item])
        data["user_id"].append([ll[4]])
        data["user_gentle"].append([ll[5]])
```

```
        data["user_age"].append([ll[6]])
        data["user_occupation"].append([ll[7]])
```

为了保证数据之间独立，所以我才独立进行处理。

特别地，从movieName中提取年份，用的正则表达式。

```
def get_year(movie):
    p1 = re.compile(r'[(](.*?)[)]', re.S)
    res = re.findall(p1, movie)
    if len(res) > 0:
        return int(res[-1].strip())
    else:
        return 0
```

另外这块代码。

```
for item in ll[3].split("|"):
    data["movie_type"].append([item])
```

因为可能存在多标签，此处单独弄one-hot，多面多标签相加即可。

然后可以开始进行one-hot了。

```
# 逐一one-hot化
oh_encoder = {}
for keys in data:
    oh_encoder[keys] = OneHotEncoder(sparse=False)
    oh_encoder[keys].fit(data[keys])
```

最后就可以保存了。

```
# 逐一保存模型
for keys in oh_encoder:
    joblib.dump(oh_encoder[keys], "%s/%s.model" % (SOURCE_OH_MODEL_PATH, keys))
```

Gen_data阶段

在数据生成阶段，主要就是上一阶段的模型基础上，对训练集和测试集使用，最终产生数据集。

首先就是加载one-hot模型，这个还是比较简单的。

```
# 加载所有one-hot模型
list_oh_path = os.listdir(SOURCE_OH_MODEL_PATH)
oh_encoder = {}
```

```
for i in list_oh_path:
    oh_encoder[i[:-6]] = joblib.load("%s/%s" % (SOURCE_OH_MODEL_PATH,i))
```

然后为了保证结果可查，还是希望记录有关onehot合并后的特征。

```
features = []
features = features + ["scores:"+str(item) for item in oh_encoder["scores"].categories_[
features = features + ["movie_id:"+str(item) for item in oh_encoder["movie_id"].categori
features = features + ["movie_year:"+str(item) for item in oh_encoder["movie_year"].cate
features = features + ["movie_type:"+item for item in oh_encoder["movie_type"].categorie
features = features + ["user_id:"+str(item) for item in oh_encoder["user_id"].categories
features = features + ["user_gentle:"+item for item in oh_encoder["user_gentle"].categor
features = features + ["user_age:"+str(item) for item in oh_encoder["user_age"].categori
features = features + ["user_occupation:"+str(item) for item in oh_encoder["user_occupat

# 特征工程配置文件
with open(GEN_DATA_CONFIG, "w") as f:
    for line in features:
        f.write("%s\n" % line)
```

记录在配置文件是这样的（节选）：

```
scores:1.0
scores:2.0
scores:3.0
scores:4.0
scores:5.0
movie_id:1.0
movie_id:2.0
```

然后是开始对训练集进行特征工程计算，这块也是比较简单粗暴，直接transform即可。

```
# 加载数据准备计算
data = []
with open(TRAIN_DATA_PATH) as f:
    fout = open(GEN_DATA_TRAIN_PATH, "w")
    idx = 0
    for line in f:
        if idx == 0:
            idx = 1
            continue
        data_item = []
        ll = line.strip().split("::")
        data_item = []
        data_item = data_item + oh_encoder["scores"].transform([[ll[0]]])[0].tolist()
        data_item = data_item + oh_encoder["movie_id"].transform([[ll[1]]])[0].tolist()
        data_item = data_item + oh_encoder["movie_year"].transform([[get_year(ll[2])]])[
        data_item = data_item + get_movie_type_oh(ll[3],oh_encoder["movie_type"]).tolist
        data_item = data_item + oh_encoder["user_id"].transform([[ll[4]]])[0].tolist()
```

```
        data_item = data_item + oh_encoder["user_gentle"].transform([[ll[5]]])[0].tolist
        data_item = data_item + oh_encoder["user_age"].transform([[ll[6]]])[0].tolist()
        data_item = data_item + oh_encoder["user_occupation"].transform([[ll[7]]])[0].to
        str_write = ",".join([str(get_item) for get_item in data_item])
        fout.write("%s\n" % (str_write))
        idx = idx + 1
        if idx % 10000 == 0:
            print("train gen %s items" % idx)
    fout.close()
print("train data gen done")
```

里面有一个对电影类型movie_type的计算，为了保证代码整洁，我单独写成了函数，因为对一部电影可能有多种类型合并，所以分别onehot后合并起来计算。

```
def get_movie_type_oh(info,oher):
    music_type_list = info.split("|")
    music_type_oh = [oher.transform([[item]])[0] for item in music_type_list]
    res = [0 for i in range(len(music_type_oh[0]))]
    for item in music_type_oh:
        item_list = item.tolist()
        for idx in range(len(item_list)):
            res[idx] = res[idx] + item_list[idx]
    return np.array(res)
```

测试集也是类似的操作，此处就不重复啦~

细节提炼

技术上其实这块非常好理解，此处提炼一些细节详细讨论一下。

- onehotEncoder其实可以合并起来多特征做特征工程实现简单粗暴的onehot，但此处有针对movie_type的多特征，且保证每个特征的独立性，后续方便进行函数化和修改，所以分别进行计算。
- 注意onehotEncoder.fit()和transform()内的数据形式。
- np.array()和list两种类型非常接近，但是API不同，各有特点，注意转化计算灵活运用。
- IO流能按照点或者是按照行读取数据，能省很多内存，每行读然后每行写。

未来其他尝试

特征工程的重要性对很多做工程的同学，包括我在内，重要性可能比模型本身还要高，原因在于改进成本低，而且效果可能很明显，相比之下模型体量大，修改成本高，有时候收益不明显，因此此处我整理了一些自己后续可能会添加的处理。

- ONEHOT基本功能
  - ~~datapreprocess 数据预处理，由于此处预处理工作较少，所以就不需要了~~
  - ~~feaextract 特征提取~~
  - ~~gen_data 数据集构建~~
  - 必要的工作函数化
  - 避免出现不存在于onehot词典的现象
- default特征引入
  - default特征定义
- embedding信息加入
  - 哪些特征可以embedding
- 其他特征
  - 历史观看信息参照
  - 相似电影
  - 相似人
  - 构造协同过滤特征

会在带入模型后，上面内容会逐步完善~

所以下一篇，就带个模型试试吧，基线，不求最好吧，重在方法实现嘛对吧。

小结

今天主要是给大家讲解了特征工程的主要内容以及我个人在用的架构，如果觉得合适欢迎使用。

下一篇就是带着模型走一遍哈哈，TF还是sklearn好呢，另外这个只是线下阶段，线上阶段涉及工程，在比较远的未来，我会尝试加上代码，计划是c++写服务吧，GRPC？可以期待一下。

我是叉烧，欢迎关注我！

叉烧，机器学习算法实习生，北京科技大学数理学院统计学研二硕士（保研），本科北京科技大学信息与计算科学、金融工程双学位毕业，硕士期间发表论文5篇，学生一作3篇，1项国家自然科学基金面上项目学生第2参与人，参与国家级及以上学术会议4次，其中，1次优秀论文，国家奖学金。曾任去哪儿网大住宿事业部产品数据，美团点评出行事业部算法工程师。

微信 zgr950123
邮箱 chashaozgr@163.com

微信个人公众号
CS的陋室

文章已于2019-05-11修改