R&S | 手把手搞推荐[1]：数据探索

原创 机智的叉烧 CS的陋室 2019-05-08

笑忘歌

五月天 - 后青春期的诗

快毕业了，换些有情调的，五月天的怎么样？

【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、而退出的特别栏目。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 | 爱奇艺搜索启发
- RS | 推荐系统整体设计
- RS | 深度讨论FM和FFM：不仅是推荐
- RS | 论文阅读：用于YouTube推荐的深度神经网络

要搞类似的算法项目，第一件事不是就整建模，而是了解数据，今天不多说别的复杂模型，就来谈谈，怎么对问题进行分析，对数据进行探索，从而为未来的建模和解决问题提供有力帮助。

懒人目录

- 数据集整理
- 再来一次数据探索
- 小结

目标确定

确认目标是一件事开始的第一步，道理大家都懂，但是能做到对问题有完整定义的却寥寥无几。

本系列，我要用Movielen提供的数据集作为基础，自己建立一个推荐系统。具体功能，就是根据给定用户，提供尽可能好的推荐内容，好的标准在于给他推荐的内容（测试集下）评分较高。

Movielens

首先来看看数据，Movielens是从官网上收集的电影评级数据，包括部分用户信息、电影信息和最终的评分结果。

此处，我们以“MovieLens 1M Dataset”作为数据，根据介绍显示，有4000部电影，6000名用户，100万评级。资源维度和用户维度数据其实并不是很多，但是由于评级量大，数据量其实足够建模，应该不会有什么大问题。

数据探索

数据探索的基本操作

有了数据就开始建模？肯定不是，相信我，数据探索绝对是值得你花时间的重要一步。

首先，要知道数据的内容，有什么数据，数据类型、字段是什么，每个字段下的数据类型是什么，离散的还是连续的等等，总结一下：

- 数据的格式（CSV，excel，dat等）以及其字段
- 字段数据类型，数字or文字，整数or分数，连续or离散，有限or无限等

然后，开始进行一些有关数据分布、相关性的分析。

- 部分数据是否具有周期性，如时间序列上的
- 数据具体的分布如何，尤其是和你目标直接相关的特征
- 是否存在严重不平衡的特征

非常推荐你打开数据看上几行（对于数据结构比较简单的我甚至会话1个小时去看看），看具体有什么特点。这些特别的样本可能就是你未来的暗坑，先知道可能会有问题，到时候排查就会简单很多。

- 有没有自己意想不到的情况，如性别为空，文字上存在多语言、标点符号等。

异常点检测，看看有没有很特别的样本，缺失值，数据记录明显有错（数据单位是万元，但是有一些几个亿的数据，基本可以判断是数据错误了），及时修正和更新，有时可能还要删除。

从上面的流程可以看到，数据探索是为了去了解数据内有什么特别地信息，让你更好地了解数据，这样才能让你后续建模思路更清晰，而不是在建模阶段才来翻看数据说明，这样效率会很低。

探索movielens-1m数据

说完上面的思路，下面就来看看在这里我是怎么做的。这里不包括所有数据操作，有些会在本文后续章节提到。

运气不错的是，他们给了我们一个非常完善的文档，里面有对数据的说明，所以数据的基本结构比较明朗（英文文档可以谷歌或者百度翻译哈~慢慢的要开始自己看英文文档）。

> - **ratings.dat**: UserID::MovieID::Rating::Timestamp
> - **users.dat**: UserID::Gender::Age::Occupation::Zip-code
> - **movies.dat**: MovieID::Title::Genres

具体每个字段的含义，可以在文档里面清晰看到，此处不赘述。

他说有4000部电影，6000个用户，到底是否真的如此，我们可以通过一个简单的命令看看，Linux或者mac shell环境下，windows下可以下载git bash或者在vscode下使用，部分命令甚至可以在powershell下尝试。

```
wc -l movies.dat
```

通过该语句可以看到movies具体有多少行。另外还非常建议大家看看数据具体是什么样的，例如我们就看前50行。

```
head -50 movies.dat
```

运行成功后你可以看到前50行的内容，通过仔细看看也是能获得一些比较重要的信息，这些信息在后续建模中是十分有用的。来举个例子吧。

> 12::Dracula: Dead and Loving It (1995)::Comedy|Horror
>
> 这是一条电影的数据，出乎意料的是，电影名上还带有电影上映的时间，这个时间可能可以在后续作为重要特征放入模型中。

数据集整理

在对数据有非常初步的认识后，可以尝试通过集成数据后在进行进一步的分析和讨论。

为了更快接近目标，可以把数据合并起来，整理成与未来进行分类相似的形式，这里非常推荐使用python中的pandas。

这里我想啰嗦一下，平时其实自己并不喜欢用pandas，直接使用数组进行操作是我的常态，加上numpy已经是极限了，这里喜欢用pandas的原因是他对数据合并具有很强的效果，我甚至觉得他有类似sql的功能，在进行表级别查询和筛选时我就会选择用pandas。

完整代码

下面是一套完整代码，处理了一套合并后的完整数据集，后续也分为了训练集和测试集，代码比较稚嫩，欢迎各位大佬提出意见。**觉得看大块代码太痛苦的继续往下翻，我会有分解动作。**

```
# 目标是构建一个可供训练和测试的数据集

import os
import pandas as pd
from sklearn.model_selection import train_test_split

MOVIE_PATH = "../../data/ml-1m/movies.dat"
RATING_PATH = "../../data/ml-1m/ratings.dat"
USERS_PATH = "../../data/ml-1m/users.dat"
SET_PATH = "../../data/ml-1m_20190508"
MOVIE_RATING_PATH = "%s/rating_combine_20190508.csv" % SET_PATH
TRAIN_RATING_PATH = "%s/TRAIN_20190508.csv" % SET_PATH
TEST_RATING_PATH = "%s/TEST_20190508.csv" % SET_PATH

if not os.path.exists(SET_PATH):
    os.makedirs(SET_PATH)

# 读取数据
movies = pd.read_csv(MOVIE_PATH, sep="::", header=None, names=[
                     "movieId", "movieName", "genres"], engine='python')
users = pd.read_csv(USERS_PATH, sep="::", header=None, names=[
                    "userId", "gender", "age", "occupation", "zipcode"], engine='python'
rating = pd.read_csv(RATING_PATH, sep="::", header=None, names=[
                     "userId", "movieId", "rating", "timestamp"], engine='python')

# 数据合并
data = pd.merge(movies, rating, on="movieId")
data = pd.merge(data, users, on="userId")

# 信息组合
data = data[["rating", "movieId", "movieName", "genres", "userId", "gender",
```

```
            "age", "occupation"]]
data.to_csv(MOVIE_RATING_PATH, index=False, sep="@")

# 训练集和测试集组合
X_train, X_test, y_train, y_test = train_test_split(data[["rating"]], data[[
                                                    "movieId", "movieName", "genres", "u
train_set = y_train.join(X_train)[
    ["rating", "movieId", "movieName", "genres", "userId", "gender", "age", "occupation'
test_set = y_test.join(X_test)[
    ["rating", "movieId", "movieName", "genres", "userId", "gender", "age", "occupation'
train_set.to_csv(TRAIN_RATING_PATH, index=False, sep="@")
test_set.to_csv(TEST_RATING_PATH, index=False, sep="@")
```

分解动作

数据集存储文档命名与初始化

大概对应代码的下面这段：

```
MOVIE_PATH = "../../data/ml-1m/movies.dat"
RATING_PATH = "../../data/ml-1m/ratings.dat"
USERS_PATH = "../../data/ml-1m/users.dat"
SET_PATH = "../../data/ml-1m_20190508"
MOVIE_RATING_PATH = "%s/rating_combine_20190508.csv" % SET_PATH
TRAIN_RATING_PATH = "%s/TRAIN_20190508.csv" % SET_PATH
TEST_RATING_PATH = "%s/TEST_20190508.csv" % SET_PATH

if not os.path.exists(SET_PATH):
    os.makedirs(SET_PATH)
```

代码估计比较简单，只要对基本语法熟悉就能看懂。所以我简单把几个要点说下。

- 我个人喜欢文件名之类的固定值提前定义好，甚至是一些训练的参数放在前面，**大写字母表示**。
- 按照日期或者日期+号码的方式命名后缀，可以记录自己的各种更新和改变，甚至可以在文件夹内部加上README记录必要的细节和区别。

数据读取

大概对应代码的下面这段：

```
# 读取数据
movies = pd.read_csv(MOVIE_PATH, sep="::", header=None, names=[
                     "movieId", "movieName", "genres"], engine='python')
users = pd.read_csv(USERS_PATH, sep="::", header=None, names=[
                    "userId", "gender", "age", "occupation", "zipcode"], engine='python'
rating = pd.read_csv(RATING_PATH, sep="::", header=None, names=[
                     "userId", "movieId", "rating", "timestamp"], engine='python')
```

- 根据探索的数据，按照一定的格式读取，因为我后需要用到pandas，所以这里建议大家用pandas的方式读取乘dataframe格式，方便后续使用，如果没这个需求，那用csv设置IO流的方式读取，其实都没关系，处理好即可。

数据合并

大概对应代码的下面这段：

```
# 数据合并
data = pd.merge(movies, rating, on="movieId")
data = pd.merge(data, users, on="userId")

# 信息组合
data = data[["rating", "movieId", "movieName", "genres", "userId", "gender",
             "age", "occupation"]]
data.to_csv(MOVIE_RATING_PATH, index=False, sep="@")
```

- 用 pd.merge进行合并，非常简单方便，还有一些更复杂的操作可以去看pandas的API
- 信息组合下的这一行 其实就体现了pandas类似SQL的功能，这里这么整的意思是按照一定顺序来读取列，保证列是按照我们的需求排列的
- to_csv是pandas下dataframe的函数，具体含义自己去查哈。

训练集测试集划分

大概对应代码的下面这段：

```
# 训练集和测试集组合
X_train, X_test, y_train, y_test = train_test_split(data[["rating"]], data[[
                                                    "movieId", "movieName", "genres", "u
train_set = y_train.join(X_train)[
    ["rating", "movieId", "movieName", "genres", "userId", "gender", "age", "occupation'
test_set = y_test.join(X_test)[
    ["rating", "movieId", "movieName", "genres", "userId", "gender", "age", "occupation'
train_set.to_csv(TRAIN_RATING_PATH, index=False, sep="@")
test_set.to_csv(TEST_RATING_PATH, index=False, sep="@")
```

- train_test_split是非常好的数据集划分工具。
- dataframe.join是一个合并dataframe的优良工具，与 str.join是两个函数，这里注意。

通过上述步骤，数据集就构建完成了。

再来一次数据探索

什么？？为啥还要一次，其实是因为有些分析合并前不好做，所以集成之后，和最终预测的数据结构类似，非常利于进行分析和计算，在python层我做简单这些分析，还不完善。

```
import pandas as pd

MOVIE_RATING_PATH = "../../data/rating_combine_20190506.csv"

combine_data = pd.read_csv(MOVIE_RATING_PATH, sep="::",engine='python')
pd.set_option('display.max_rows', 1000)

# 随便找个人看看打分的分布
print(combine_data[["userId", "movieId", "rating"]][combine_data["userId"]==9].groupby(b
print("----------------------------------------------------------")

# 随便找个电影看看打分的分布
print(combine_data[["userId", "movieId", "rating"]][combine_data["movieId"]==20].groupby
print("----------------------------------------------------------")

# 统计每个用户评论电影的数量
print(combine_data[["userId", "movieId", "rating"]].groupby(by='userId').count())
print("----------------------------------------------------------")

# 统计每个电影被评论的数量
print(combine_data[["userId", "movieId", "rating"]].groupby(by='movieId').count())
print("----------------------------------------------------------")

# 统计给电影打分次数的分布
print(combine_data[["userId", "movieId", "rating"]].groupby(by='userId').count().groupby
print("----------------------------------------------------------")
```

看看具体的案例，从一个人的角度，一部电影的角度等，去进行分析，分析的目标有这几个：

- 有没有比较特别的案例
- 是否存在数据不平衡的问题
- 单特征样本量是否会不足

小结

本文主要给大家谈到了数据探索的原因和方法，也给大家提供了代码甚至是分解动作，在这里简单总结一下。

- 数据探索是一个对数据有深入了解的过程，对数据都不了解谈不上解决问题，做饭总得知道冰箱里有啥菜，够不够吃，要不要再去买，一个道理。
- 所谓得了了解数据，除了知道有什么，还要知道很多细节信息，数据类型，数据平衡问题，缺失问题等。
- 有些工作通过shell的角度可以快速解决，python有时候会太拖沓。
- pandas在进行数据查询之类操作十分高效，欢迎尝试。
- 提醒一个暗坑，pandas似乎有点吃内存。

好了，现在对数据有基本的了解了，下一篇开始我就开始弄第一个基线模型，尝试用LR来进行用户打分预估，敬请期待。

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。

微信个人公众号
CS的陋室

微信 zgr950123
邮箱 chashaozgr@163.com