滴滴算法大赛解题思路

队伍：五岳剑派

2016-6-18

中国科学院计算技术研究所 • 网络数据实验室
北京市海淀区科学院南路 6 号

说明：

本工程由 Github 私有仓库托管（https://github.com/HouJP/di-tech-16），其完整记录了第一赛季过程中的版本更新过程。由于该项目目前为私有仓库，如需审查，请联系 15201442067（侯建鹏同学）开通权限。

在邮件的附件中，提供了线上最优成绩所对应的训练及预测的数据文件(LIBSVM 格式)，以及训练模型的脚本程序，执行该脚本程序，即可生成线上最优成绩的结果文件。

一 工程框架

图 1 工程框架图

在本项目中，我们队伍所使用的工程框架如图(1)所示。各文件夹及文件详细说明如下：

- bin/：该目录下存放可执行的 shell 及 python 脚本工具，例如 xgboost 训练预测工具等；
- conf/：该目录下存放 shell 及 python 脚本的配置参数，例如工程路径、xgboost 模型参数等；
- di-tech-16-on-spark/：该目录为通过 IntelliJ IDEA 构建的支持 spark 分布式计算框架

的代码工程，编程语言为 scala，需要运行时可通过编译器生成可执行 jar 包，用来完成数据预处理、特征抽取及线下评分等功能；

- data/raw/season_2/：该目录为第一赛季更换数据后的数据目录，存放全部数据文件；

下面，详细介绍 data/raw/season_2/目录的构成，如下：

- data/raw/season_2/fs/：该目录为特征池，以日期及特征名构建子目录，存放全部特征文件；
- data/raw/season_2/label/：该目录为标签池，以日期构建子目录，存放全部标签文件；
- data/raw/season_2/features：该文件实质上为配置文件，内容为当前模型选择使用的特征名，在训练文件生成阶段，程序会根据该文件的内容进入特征池选取指定特征来生成 lisvm 训练文件；
- data/raw/season_2/dates：该文件实质上为配置文件，记录原始数据集中包含的日期范围，在数据预处理及特征生成时，将会根据该文件的内容按日期处理数据或生成特征；
- data/raw/season_2/training_set/：该文件夹对应线下训练/预测过程的数据文件夹，里面包含完成线下训练所需的数据文件(train_libsvm/train_key)、线下预测所需的数据文件(test_libsvm/test_key)、线下生成的答案目录(ans/)以及线下训练及预测所采用的时间片区间(train_time_slices/test_time_slices)；
- data/raw/season_2/test_set_2/：该文件夹对应第一赛季更换数据后线上训练/预测过程的数据文件夹，里面包含完成线上训练所需的数据文件(train_libsvm/train_key)、线上预测所需的数据文件(test_libsvm/test_key)、线上生成的答案目录(ans/)以及线上训练及预测所采用的时间片区间(train_time_slices/test_time_slices)。

二 数据预处理

由于原始数据文件 order_data/较大，为了提升数据处理及特征抽取的速度，我们对原始数据文件做了如下处理：

- 删除 order_id，driver_id，passenger_id；
- 将 start_district_id，dest_district_id 替换为 cluster_map 中的 district_id，若其不包含某区域 hash 值，则对应 id 置为-1；.
- 解析 Time 字段，增加年、月、日、时、分、秒以及时间片 id。

三 特征工程

特征名	Key	描述	维度
Week	Date	样本日期所对应周几的 one-hot 编码，以及最后一维是否是假日	1-8
TID	Time_ID	时间片 ID	9
FPOI	District	区域的 POI 特征	10-185
FineArriveSelf	District,	样本时间片前半个小时中，按照 5 分钟间隔	186-

	Date, Time_ID	滑动大小为 10 分钟的窗口，统计从本地到本地有司机接单订单数	190
FineArrive	District, Date, Time_ID	样本时间片前半个小时中，按照 5 分钟间隔滑动大小为 10 分钟的窗口，统计到本地有司机接单的订单数	191-195
FineGap	District, Date, Time_ID	样本时间片前半个小时中，按照 5 分钟间隔滑动大小为 10 分钟的窗口，统计从本地出发没有司机接单的订单数	196-200
FineDemand	District, Date, Time_ID	样本时间片前半个小时中，按照 5 分钟间隔滑动大小为 10 分钟的窗口，统计从本地的总订单数	201-205
FWeatherOHE	Date	对天气进行 OHE 映射，温度，PM2.5	206-214
FDTGap	District, Time_ID	前 21 天，每天的 gap 值，求出的均值，中位数，标准差，最小值，最大值	215-219
FDTDemand	District, Time_ID	前 21 天，每天的 Demand 值，求出的均值，中位数，标准差，最小值，最大值	220-224
FDTGapByHoliday	District, Time_ID	前 21 天，每天的 Gap 值按照是否是假日分开求出的均值，中位数，标准差，最小值，最大值	225-234
FDTSupply	District, Time_ID	前 21 天，每天的供应值按照是否是假日分开求出的均值，中位数，标准差，最小值，最大值	235-239
FDateDistrict	District	前 21 天， 每天每个区域总 gap 数，求均值，中位数，标准差，最小值，最大值	240-244
FDateTimeGap	Time_ID	前 21 天，每天每时间片所有区域总 gap 数，求均值，中位数，标准差，最小值，最大值	245-249
FHolidayTimeGap	Time_ID	前 21 天，每天每个时间片所有区域 gap 数，按照是否是节假日分别求均值，中位数，标准差，最小值，最大值	250-259
FWeekTimeGap	WeekDay, Time_ID	前 21 天，每天每个时间片总 gap 数，按照是周几分别求均值，中位数，标准差，最小值，最大值	260-264
FHolidayDistrictGap	Time_ID	前 21 天，每天每个区域总 gap 数，按照是否是节假日分别求均值，中位数，标准差，最小值，最大值	265-274
FWeekDistrictGap	WeekDay, Time_ID	前 21 天，每天每个区域总 gap 数，按照是周几分别求均值，中位数，标准差，最小值，最大值	275-279
FPrefixGap	District, Date, Time_ID	当前时间片往前 2,4,6,8,10,…,30 分钟累计的 gap 数	280-294
FineTimeGap	Date, Time_ID	样本时间片前半个小时中，按照 5 分钟间隔滑动大小为 10 分钟的窗口，统计出所有地	295-299

		区 gap 总数	
FineGapStat	Date, Time_ID	对 FineTimeGap 的每个时间片所对应的值求均值，中位数，标准差，最小值，最大值	300-304
FTrafficTotal		每个区域路况信息中四个 level 数的总数	307

四 模型选择

在本项目中，我们采用的模型为 GBRT（Gradient Boosting Regression Trees），采用的工具为 xgboost（https://github.com/dmlc/xgboost）。在该模型中，我们使用了自定义的损失函数，如下所示：

$$\text{loss} = \left[abs\left(\frac{pred - gap}{gap}\right)\right]^{\frac{3}{2}}$$

模型的参数如下所示：

```
{
    "objective": "reg:linear",
    "nthread" : 15,
    "eta" : 0.05,
    "max_depth":5,
    "min_child_weigth": 10,
    "gamma" : 0,
    "subsample" : 0.8,
    "colsample_bytree" : 0.9,
    "n_round" :3000
}
```

五 模型融合

在上述参数的基础之上，随机化 max_depth, eta, subsample, colsample_bytree, seed 等参数。代码如下所示：

```
params['max_depth'] = params['max_depth'] + random.randint(-1, 1)
params['eta'] = params['eta'] +    (random.random() - 0.5) / 50
params['subsample'] = params['subsample'] +    (random.random() - 0.5) / 5
params['colsample_bytree'] = params['colsample_bytree'] +    (random.random() - 0.5) / 5
params['seed'] = int(time.time())
```

由此训练出 24 个 xgboost 模型，最后预测结果取这 24 个模型输出结果的平均值。