数据分析与数据挖掘第三次作业第一大题 – a问分析文档

数据分析与数据挖掘

DAM COURSE, SPRING 2018

BY

1552674 李　源

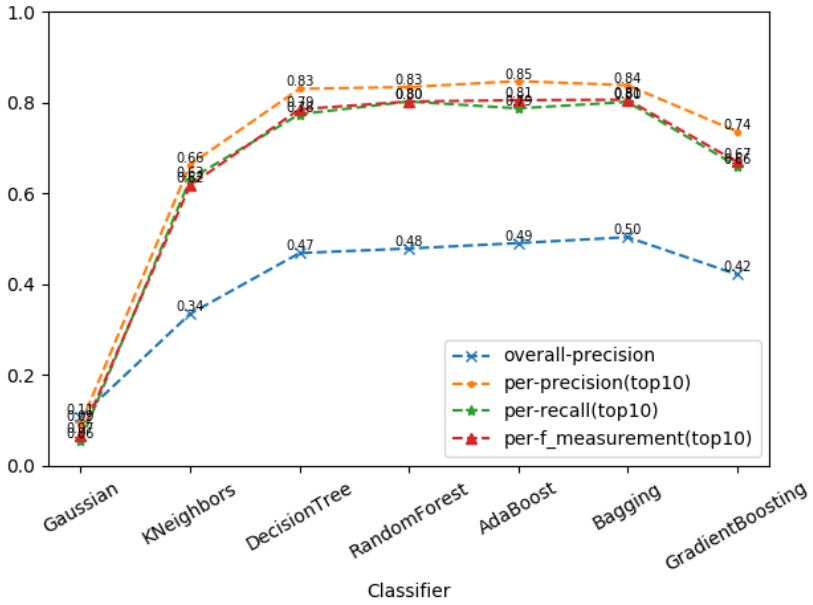
同濟大學
TONGJI UNIVERSITY

Tongji University
School of Software Engineering

1 代码运行结果

这里我参照了文档要求，选择了如下指标作为评判分类器预测结果，分别为：总体的 precision 值，格中点数前十的栅格的 precision 、 recall 以及 f-measurement 值，预测结果转为经纬度后的中位误差。

七种分类器的评判结果如下：



最终七种分类器作出的CDF图如下：

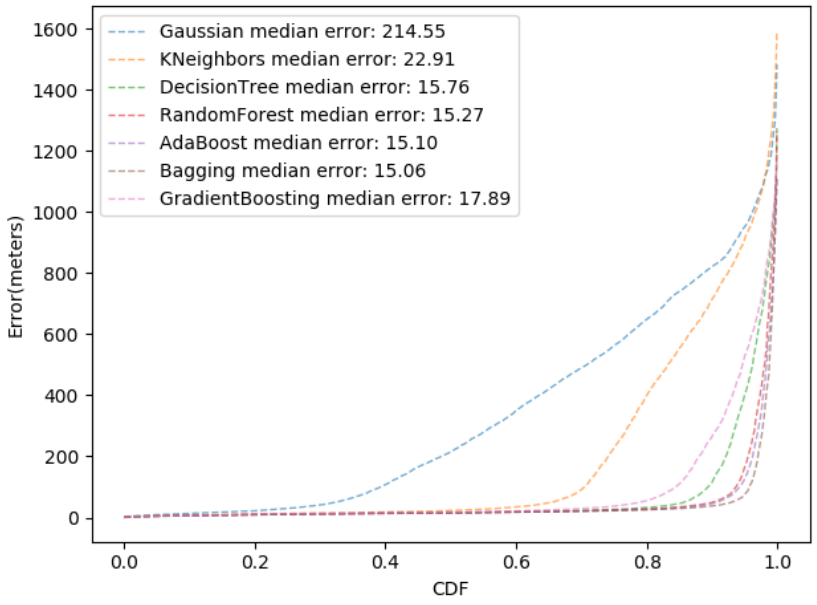


Figure 1.1: CDF图

七种分类器的 precision、recall、f-measurement 结果汇总如下（对于Top10的对应评价标准，我在这里取的平均值）：

分类器	评价标准			
	precision	per precision	per recall	per f-measurement
Gaussian	0.11	0.09	0.06	0.07
KNeighbors	0.34	0.66	0.63	0.62
DecisionTree	0.47	0.83	0.79	0.79
RandomForest	0.48	0.83	0.80	0.80
AdaBoost	0.49	0.85	0.79	0.81
Bagging	0.50	0.84	0.80	0.80
GradientBoosting	0.42	0.74	0.66	0.67

2 分析讨论

这里主要对原始数据的处理、Adaboost 和 GradientBoosting 的参数调节、预测结果这三个部分进行分析讨论。

2.1 原始数据的处理

针对最原始的 data_2g.csv 数据，其中基站的坐标信息是通过（RNCID，CellID）这样的键值对来表示的，首先我需要做的是，利用 2g_gongcan.csv 里面的数据将（RNCID，CellID）替换为对应的（Latitude，Longitude）键值对。

这里存在的一个问题是，2g_gongcan.csv 中的数据只能保证能够查到每一条记录的主基站的（RNCID，CellID）对应的经纬度。经过我的统计，还有 60 余个基站的经纬度信息是无法被找到的。对于这一类数据，我的做法是，将其经纬度设置为（0，-1），其依据是，我在原始数据中，能够发现一些（RNCID，CellID）为（0，-1）的基站信息，那么我可以把这一类不能找到对应其地理位置信息的基站都归为同一类型的数据，其表示方法也应该相同。

除此之外，还有部分的数据，其被记录下来的基站信息是小于7个的，那么就会存在部分空值。对于空值的处理我是全部补 0，这样能够正常训练同时也符合实际意义。

在完成经纬度转换后，还需要做的进一步操作是，利用所有MR数据的GPS labels，确定一个整体的位置范围，并将该范围转换成一个大的矩形，再将该矩形划分成若干个等大的小正方形栅格。最后利用正方形栅格的 ID 作为数据的 label。

我的方法是，首先参考网上代码，实现了计算两点（用经纬度表示）之间的距离的函数 haversine（可参见 a/util.py）。再结合左下角、右上角的经纬度信息，将整个范围划分为 82*65 的栅格。其 ID 是从左到右、从下到上进行的标注，且初始值为 1。

对于数据集特征的构成，我选择了数据的七个基站的地理位置信息（即转换后的经纬度）和 AsuLevel、SignalLevel 两个信号强度特征共同构成。对于数据集的标签，我是使用之前转换好的正方形栅格的 ID。

2.2 Adaboost 和 GradientBoosting 的参数调节

sklearn 中的 sklearn.model_selection 包提供了 GridSearchCV 方法来尝试找出最优的参数。因为这是一个多分类问题，所以我使用 f1_micro 来作为不同参数的评判标准，并且进行 5-fold 交叉验证。

针对 Adaboost 分类器，我考虑了4个参数，分别是弱分类器的种类、弱分类器的个数、步长以及其内部的算法。其待选择的参数如下：

```
param_test1 = {'n_estimators': range(30, 101, 10),
               'learning_rate': np.arange(0.01, 0.1, 10),
               'base_estimator': [GaussianNB(), KNeighborsClassifier(), DecisionTreeClassifier()],
               'algorithm': ['SAMME', 'SAMME.R']}
```

Figure 2.1: Adaboost 待选择参数

针对 GradientBoosting 分类器，我参考了这篇博客：http://www.cnblogs.com/pinard/p/6143927.html，考虑到了弱分类器的个数、步长、决策树最大深度、最大特征数以及子采样的比例。其待选择的参数如下：

```
param_test1 = {'n_estimators': range(10, 61, 10),
               'learning_rate': np.arange(0.01, 0.1, 10)}
param_test2 = {'max_depth': range(3, 14, 2)}
param_test3 = {'max_features': range(7, 20, 2),
               'subsample': [0.6, 0.7, 0.75, 0.8, 0.85, 0.9]}
```

Figure 2.2: GradientBoosting 待选择参数

最后两种分类器选择出的结果如下：

参数	结果
n_estimators	30
learning_rate	0.01
algorithm	SAMME.R
base_estimator	DecisionTreeClassifier(max_depth=20)

Table 1: Adaboost 参数

参数	结果
n_estimators	60
learning_rate	0.01
max_depth	3
max_features	19
sub_sample	0.75

Table 2: GradientBoosting 参数

2.3 预测结果分析

七种分类器的预测结果可以分为三种情况。第一类是 Gaussian 分类器，其结果在所有的评判标准上均取得的效果较差，可以说 Gaussian 分类器是不适合本次研究的分类问题的。第二类是 KNeighbors 和 GradientBoosting 分类器，这两个分类器最终的结果要略逊一筹。剩下的四个分类器 DecisionTree、RadomForest、Adaboost 和 Bagging 在评判标准上都取得了相对较好的成绩，且都比较接近。

这里我想将 RadomForest 和 GradientBoosting 进行一个比较。这两种分类器都是以**集成学习 + 决策树**为基础，利用了集成的思想来提升了单颗决策树的分类性能。额，也可以理解为“三个臭皮匠，顶个诸葛亮”。

RadomForest，它的随机体现在随机选择样本和随机选择特征进行训练，这样做是为了让随机森林不容易陷入过拟合，并且对缺省值不敏感，即抗噪能力强。这一点我觉得是在本次问题中其效果比 GradientBoosting 好的关键之一，我先简单说一下 GradientBoosting 的原理，再对这个做阐述。

GradientBoosting 与 RadomForest 的不同就在于，它的决策树是迭代生成的。每一棵树学的是之前所有树结论和的残差，这个残差就是一个加预测值后能得真实值的累加量。那么，这样会存在的一个情况就是，其对异常值和缺省值比较敏感。

好的，现在再来看这次训练用到的数据，每条记录对应的基站理论上有 7 个，然而事实上存在着很多基站的信息为空的情况，这里我做了一个简单的统计：

有效的基站信息个数	数据条数
2	114
3	81
4	249
5	886
6	2216
7	2550

Table 3: 基站信息情况统计

存在缺省值的数据实际上挺多的，达到了 51%。因此在一定程度上导致了对缺省值比较敏感的 GradientBoosting 效果略差。

还有一个问题是，由于 GradientBoosting 的特性，其决策树的生成是串行的而非并行的，所以耗时也是明显的要大很多，我跑一次分类大概需要 1 个多小时，这也增大了我调参的难度。

3 性能分析

七种分类器的耗时如下：

分类器	耗时（循环10次结果）
Gaussian	0:00:20.24
KNeighbors	0:00:13.12
DecisionTree	0:00:16.76
RandomForest	0:00:26.09
AdaBoost	0:06:21.42
Bagging	0:01:28.11
GradientBoosting	11:14:17.90

Table 4: 分类器耗时

GradientBoosting 的耗时及其“喜人”，跑完 10 次需要将近半天时间。我在前面有提到，其的决策树生成是串行的，所以耗时较久。