KMeans算法与交通事故理赔审核预测

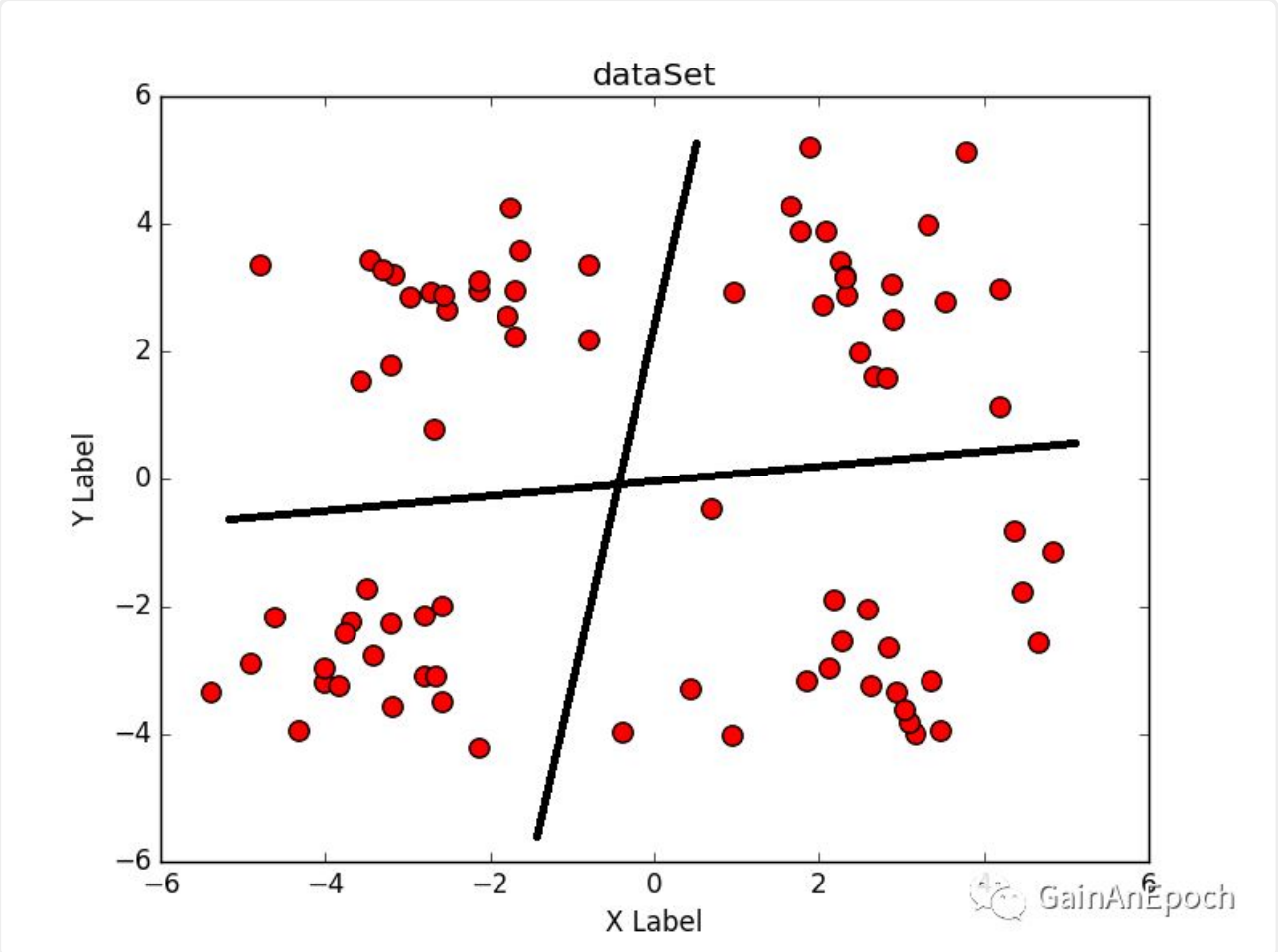
原创 李二 机器视觉与算法建模 2018-05-21

> 上一篇文章讲解了数据挖掘环境的配置，那这次就从一个小的实战开始吧。这次要学习的是KMeans算法，要挑战的是sofasofa上的一个竞赛（交通事故理赔审核预测）。现在开始吧

K-means

介绍

K-Means是基于划分的聚类方法，他是数据挖掘十大算法之一。基于划分的方法是将样本集组成的矢量空间划分为多个区域，每个区域都存在一个样本中心，通过建立映射关系，可以将所有样本分类到其相应的中心。



假设有样本集合D={Xj}，KMeans算法的目标是将数据划分为K类：S={S1,S2,...Sk}，并

且使划分后的K个子集合满足类内误差平方和最小。

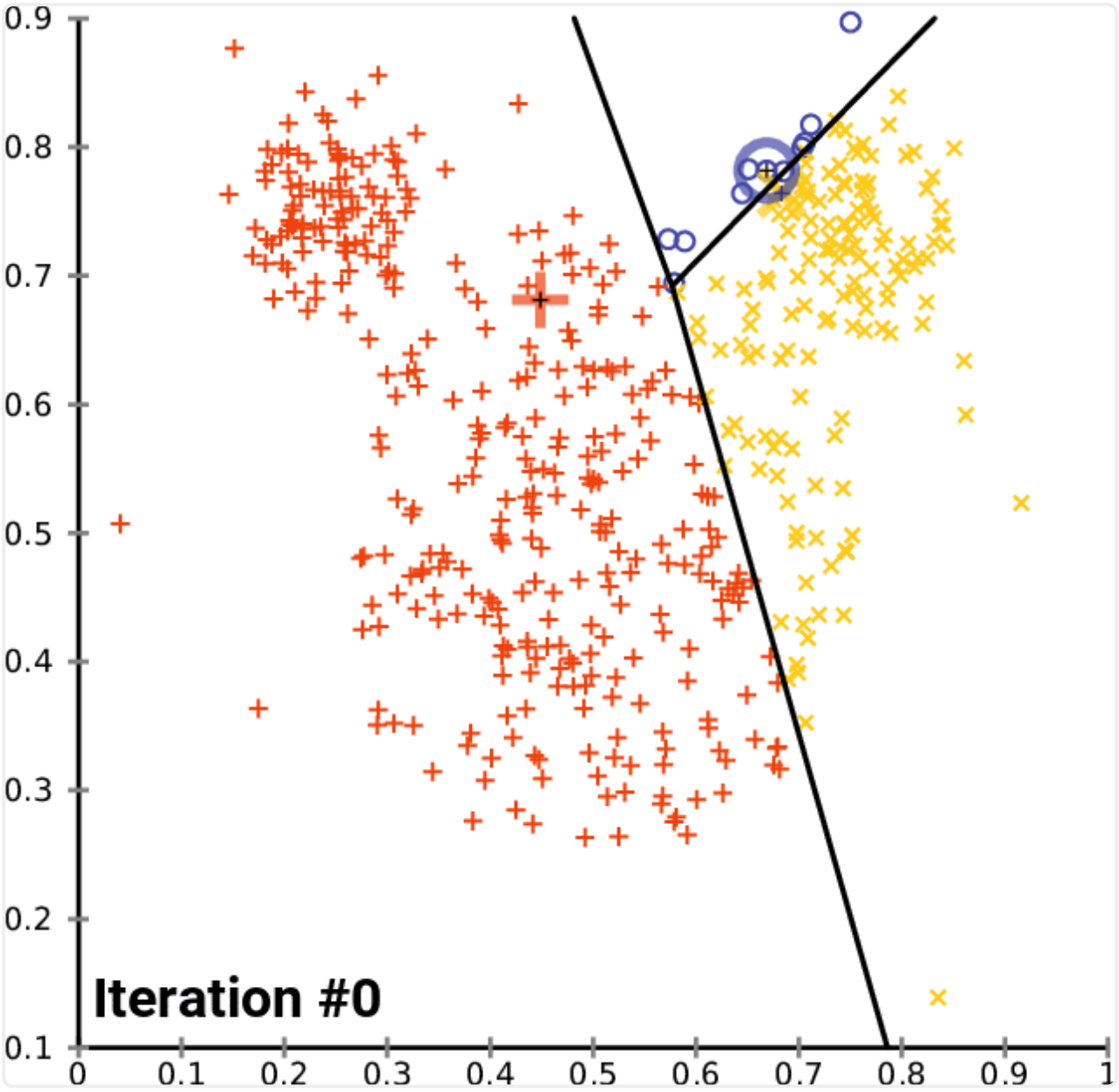
目标函数：$$\ell_{k-means}(S)=\arg\min_{S=\{S_i\}_{i=1}^{k}}\sum_{i=1}^{k}\sum_{x\in S_i}\|x-c_i\|_2^2$$ 其中

$$c_i=\frac{1}{|S_i|}\sum_{x\in S_i}x$$ Ci即划分后的子集合的中心。

求解步骤

求解目标函数是一个NP-hard问题，无法保证得到的就是全局最优解。在经典K-Means聚类算法中采取迭代优化策略，一般包含以下四个步骤

- 1.初始化聚类中心
- 2.分配个样本xj到相近的聚类集合，依据是（p!=j）
- $$S_i^{(t)}=\{x_j\mid\|x_j-c_i^{(t)}\|_2^2\leqslant\|x_j-c_p^{(t)}\|_2^2\}$$
- 3.根据步骤二结果，更新聚类中心。
- $$c_i^{(t+1)}=\frac{1}{|S_i^{(t)}|}\sum_{x_j\in S_i^{(t)}}x_j$$
- 4.若达到最大迭代步数或两次迭代差小于设定的阈值则算法结束，否则重复步骤2。



算法改进

经典的K-means算法在初始化聚类中心时采用的是随机采样的方式，不能保证得到期望的聚类结果，可以选择重复训练多个模型，选取其中表现最好的。但有没有更好的方法呢？David Arthur提出的K-means++算法能够有效的产生初始化的聚类中心。

首先随机初始化一个聚类中心C1，然后通过迭代计算最大概率值X*，将其加入到中心点中，重复该过程，直到选择k个中心。

$$x^* = \arg\max_x \frac{d(x,C)}{\sum_{j=1,\cdots,n} d(x_j,C)}$$

交通事故理赔审核预测

> SofaSofa是专门为数据挖掘新人准备练手比赛的地方，这的比赛都会提供几个标杆模型的代码给你参考，新手想要快速入门可以多去这个网站上看看。

赛题

这个比赛的链接：http://sofasofa.io/competition.php?id=2

- 任务类型：二元分类
- 背景介绍：在交通摩擦（事故）发生后，理赔员会前往现场勘察、采集信息，这些信息往往影响着车主是否能够得到保险公司的理赔。训练集数据包括理赔人员在现场对该事故方采集的36条信息，信息已经被编码，以及该事故方最终是否获得理赔。我们的任务是根据这36条信息预测该事故方没有被理赔的概率。
- 数据介绍：

训练集中共有200000条样本，预测集中有80000条样本。

变量说明：

变量名	解释
CaseId	案例编号，没有实际意义
Q1	理赔员现场勘察采集的信息，Q1代表第一个问题的信息。信息被编码成数字，数字的大小不代表真实的关系。
Qk	同上，Qk代表第k个问题的信息。一共36个问题。
Evaluation	**表示最终审核结果。0表示授予理赔，1表示未通过理赔审核。在test.csv中，这是需要被预测的标签。**

- 评价方法：Precision-Recall AUC

代码

在官方下载好数据集，在本地解压。打开jupyter notebook开始动手。

首先导入必要的包

```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
%matplotlib inline
```

读入数据集

```
homePath = "data"
trainPath = os.path.join(homePath, "train.csv")
testPath = os.path.join(homePath, "test.csv")
submitPath = os.path.join(homePath, "sample_submit.csv")
trainData = pd.read_csv(trainPath)
testData = pd.read_csv(testPath)
submitData = pd.read_csv(submitPath)
```

参照数据说明，CaseID这列是没有意义的编号，因此这里将他丢弃。

- ~drop()函数： axis指沿着哪个轴，0为行，1为列； inplace指是否在原数据上直接操作

```
# 去掉没有意义的一列
trainData.drop("CaseId", axis=1, inplace=True)
testData.drop("CaseId", axis=1, inplace=True)
```

快速了解数据

- ~head()：默认显示前5行数据，可指定显示多行，例如.head(50)显示前50行

```
trainData.head()
```

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	...	Q28	Q29	Q30	Q31	Q32	Q33	Q34	Q35	Q36	Evaluation
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	1	1	1	1	0	0	0	0	0
2	0	0	0	0	0	0	0	1	0	0	...	1	2	2	2	1	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	1	3	2	3	1	0	0	1	1	0
4	0	0	0	0	0	0	0	0	0	0	...	1	4	2	4	1	0	0	1	1	0

5 rows × 37 columns

显示数据简略信息，可以每列有多少非空的值，以及每列数据对应的数据类型。

```
trainData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 37 columns):
Q1              200000 non-null int64
Q2              200000 non-null int64
Q3              200000 non-null int64
Q4              200000 non-null int64
Q5              200000 non-null int64
Q6              200000 non-null int64
Q7              200000 non-null int64
Q8              200000 non-null int64
```


以图的形式，快速了解数据

- ~hist():绘制直方图，参数 `figsize` 可指定输出图片的尺寸。
- 关于绘图可参考我之前的一篇文章，一文教会你使用Matplotlib绘图

```
trainData.hist(figsize=(20, 20))
```

想要了解特征之间的相关性，可计算相关系数矩阵。然后可对某个特征来排序。

```
corr_matrix = trainData.corr()
corr_matrix["Evaluation"].sort_values(ascending=False) # ascending=False 降序排列
```

```
Evaluation    1.000000
Q28           0.410700
Q30           0.324421
Q36           0.302709
Q35           0.224996
Q34           0.152743
Q32           0.049397
Q21           0.034897
Q33           0.032248
Q12           0.022602
```

GainAnEpoch

从训练集中分离标签

```
y = trainData['Evaluation']
```

```
trainData.drop("Evaluation", axis=1, inplace=True)
```

使用K-Means训练模型

- KMeans()： n_clusters指要预测的有几个类； init指初始化中心的方法，默认使用的是 k-means++方法，而非经典的K-means方法的随机采样初始化，当然你可以设置为 random使用随机初始化； n_jobs指定使用CPU核心数，-1为使用全部CPU。

```
from sklearn.cluster import KMeans
est = KMeans(n_clusters=2, init="k-means++", n_jobs=-1)
est.fit(trainData, y)
y_pred = est.predict(testData)
```

保存预测的结果

```
submitData['Evaluation'] = y_pred
submitData.to_csv("submit_data.csv", index=False)
```

现在你可以在运行目录找到这个文件，在比赛网站上可提交查看实际分数。

标杆模型：随机森林

使用K-means可能得到的结果没那么理想。在官网上，举办方给出了两个标杆模型，效果最好的是随机森林。以下是代码，读者可以自己测试。

```
# -*- coding: utf-8 -*-
import pandas as pd
from sklearn.ensemble import RandomForestClassifier

# 读取数据
train = pd.read_csv("data/train.csv")
test = pd.read_csv("data/test.csv")
submit = pd.read_csv("data/sample_submit.csv")

# 删除id
train.drop('CaseId', axis=1, inplace=True)
test.drop('CaseId', axis=1, inplace=True)

# 取出训练集的y
y_train = train.pop('Evaluation')

# 建立随机森林模型
clf = RandomForestClassifier(n_estimators=100, random_state=0)
clf.fit(train, y_train)
y_pred = clf.predict_proba(test)[:, 1]

# 输出预测结果至my_RF_prediction.csv
submit['Evaluation'] = y_pred
submit.to_csv('my_RF_prediction.csv', index=False)
```

总结

K-means算法是数据挖掘的十大经典算法之一，但实际中如果想要得到满意的效果，还是非常难的，以后会讲到集成学习，使弱学习器进阶为强学习器。

> 关于数据挖掘的更多内容，我将持续更新在该项目，欢迎感兴趣的朋友赏个star：
> https://github.com/wmpscc/DataMiningNotesAndPractice