数据挖掘作业 3　　聚类 KMedoids

姓名：董永银

学号：2120171007

日期：2018.4.23

titanic 数据的聚类 KMedoids 分析报告

一.数据源

选取 https://www.kaggle.com/c/titanic/data 数据进行分类分析，共三个文件：gender_submission.csv、test.csv、train.csv，分别是提交格式、测试集和训练集。

二.数据分析

1、数据初探（一）数据概况

train.csv 文件里的内容，主要包含这么几列，可以简单地先判断一下那些数据比较有用：

PassengerId：只是个乘客序号；

Survived：最终是否存活；

Pclass：舱位，1 是头等舱，3 是最低等，从电影里看，这个影响还是挺大的；

Name：乘客姓名，除非是要算命，不然应该没啥影响；

Sex：性别，应该影响很大；

Age：年龄，有一部分数据缺失；

SibSp：一同上船的兄弟姐妹或配偶；

Parch：一同上船的父母或子女，目测这两项应该没啥影响吧，除非是要是一起死的那种；

Ticket：船票信息，比较乱，完全看不出有任何用处；

Fare：乘客票价，这个数据应该和 Pclass 有一定对应关系；

Cabin：客舱编号，应该不同的编号对应不同的位置，对逃生还是有一定影响的，然而这项数据缺失很多（204/891），所以我选择暂时忽略；

Embarked：上船地点，主要是 S（南安普顿）、C（瑟堡）、Q（皇后镇），这个应该也没啥影响，但不妨一试。

2、读取数据

data_train=pd.read_csv('train.csv')

	PassengerId	Survived	Pclass
0	1	0	3
1	2	1	1
2	3	1	3
3	4	1	1
4	5	0	3
5	6	0	3

	Name	Sex	Age	SibSp
0	Braund, Mr. Owen Harris	male	22.0	1
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1
2	Heikkinen, Miss. Laina	female	26.0	0
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1
4	Allen, Mr. William Henry	male	35.0	0
5	Moran, Mr. James	male	NaN	0

3、补充缺失值

把年龄的缺失值用现有数据的中位数来替代并赋值给原数据。

```
#把年龄的缺失值用现有数据的中位数来替代并赋值给原数据
data_train['Age'] =  data_train['Age'].fillna(data_train['Age'].median())
```

把男性置 0，女性置 1。

```
#把男性置0，女性置1
data_train.loc[data_train['Sex'] == 'male', 'Sex'] = 0
data_train.loc[data_train['Sex'] == 'female', 'Sex'] = 1
```

把登船口的缺失值用 S 替代（整体数据 S 偏多），并将所有登船口转换为数字。

```
#把登船口的缺失值用S替代（整体数据S偏多），并将所有登船口转换为数字
data_train['Embarked'] = data_train['Embarked'].fillna('S')
data_train.loc[data_train['Embarked'] == 'S', 'Embarked'] = 0
data_train.loc[data_train['Embarked'] == 'C', 'Embarked'] = 1
data_train.loc[data_train['Embarked'] == 'Q', 'Embarked'] = 2
```

去除乘客姓名、船票信息和客舱编号三个不打算使用的列，将数据转变为 int 型。

```
#去除乘客姓名、船票信息和客舱编号三个不打算使用的列
data_train=data_train.drop(['Name','Ticket','Cabin'],axis=1)

#数据转变为int型

data_train['Age']=np.array(data_train['Age'],dtype=np.int)

data_train['Age']=pd.DataFrame(data_train['Age'])
```

三. 聚类 KMedoids 分析

1、创建一个 900x3 的全零方阵 A

```
#先创建一个900x3的全零方阵A，并且数据的类型设置为float浮点型
A = zeros((900,3),dtype=float)
```

2、将 train.csv 数据的值存储到 trained.txt 中，比如：主要取数据中的 Survived 值、Pclass 值和 Age 值。

```
#训练集数据格式整理
def txtFileProduce1(f, data_train):
    with open(f, 'w') as file:
        for i in range(len(data_train)):

            #data.Survived和data['Survived']此处等价
            #python要求读写文件的数据是字符串
            file.write(str(data_train.Survived[i])
                       + ' ' + str(data_train.Pclass[i])
                       + ' ' + str(data_train.Age[i]) + '\n')
```

将 trained.txt 数据的值读入矩阵 A 中

```
f = open('trained.txt')          #打开数据文件文件
lines = f.readlines()            #把全部数据文件读到一个列表lines中
A_row = 0                        #表示矩阵的行，从0行开始
for line in lines:               #把lines中的数据逐行读取出来
    list = line.strip('\n').split(' ')      #处理逐行数据：strip表示把头尾的'\n'去掉，split表示以空格来分割行数据，
                                             #然后把处理后的行数据返回到list列表中
    A[A_row:] = list[0:4]                    #把处理后的数据放到方阵A中。list[0:3]表示列表的0,1,2列数据放到矩阵A中的A_row行
    A_row+=1                                 #然后方阵A的下一行接着读

dataSet=A
```

3、采用 kmedoids 聚类方法对数据进行挖掘

Kmedoids 算法的基本思想是：与 K-means 算法类似，区别在于中心点的选取，K-means 中选取的中心点为当前类中所有点的重心，而 K-medoids 法选取的中心点为当前 cluster 中存在的一点，准则函数是当前 cluster 中所有其他点到该中心点的距离之和最小，这就在一定程度上削弱了异常值的影响，但缺点是计算较为复杂，耗费的计算机时间比 K-means 多。

根据当前的 medoidids 中心和欧几里得距离来计算每个数据点的归属。

```
#clustering()函数 分配每一点的归属 根据当前的情况.
def clustering(data, medoids):
    '''
根据当前的medoidids中心和欧几里得距离来计算每个数据点的归属。
'''

    # pdb.set_trace()
    med_idx = medoids[-1]
    med = data[med_idx]
    k = len(med_idx)


    dis = cdist(data, med)
    best_med_it_belongs_to = dis.argmin(axis = 1)
    for i in range(k):
        medoids[i] =where(best_med_it_belongs_to == i)
```

根据当前设置计算总成本。

```
'''data是一个array, 每一行是一个数据点.下面这个函数计算total cost 根据当前的情况:'''
def total_cost(data, medoids):
    '''
根据当前设置计算总成本。
'''
    med_idx = medoids[-1];
    k = len(med_idx);
    cost = 0.0;

    med = data[ med_idx]
    dis = cdist( data, med, 'euclidean')
    cost = dis.min(axis = 1).sum()

    medoids[-2] = [cost]
```

Kmedoids 算法的过程：

```
#kmedoids() 函数
def kmedoids( data, k):

    N = len(data)
    cur_medoids = {}
    cur_medoids[-1] = list(range(k))
    clustering(data, cur_medoids)
    total_cost(data, cur_medoids)
    old_medoids = {}
    old_medoids[-1] = []

    iter_counter = 1
    # stop if not improvement.
    while not set(old_medoids[-1]) == set(cur_medoids[-1]):
        print('iteration couter:' , iter_counter)
        iter_counter = iter_counter + 1
        best_medoids = copy.deepcopy(cur_medoids)
        old_medoids = copy.deepcopy(cur_medoids)
        # pdb.set_trace()
        # iterate over all medoids and non-medoids
        i=0
        j=0
        for i in range(N):
            for j in range(k):
                if not i ==j :
                    # swap only a pair
                    tmp_medoids = copy.deepcopy(cur_medoids)
                    tmp_medoids[-1][j] = i

                    clustering(data, tmp_medoids)
                    total_cost(data, tmp_medoids)
                    # pick out the best configuration.
                    if( best_medoids[-2] > tmp_medoids[-2]):
                        best_medoids = copy.deepcopy(tmp_medoids)
        cur_medoids = copy.deepcopy(best_medoids)
        print('current total cost is ', cur_medoids[-2])
    return cur_medoids
```

4、将聚类值进行可视化

```
# plot different clusters with different colors.
scatter( data[medoids[0], 0] ,data[medoids[0], 1], c = 'r')
scatter( data[medoids[1], 0] ,data[medoids[1], 1], c = 'g')
scatter( data[medoids[2], 0] ,data[medoids[2], 1], c = 'y')
scatter( data[medoids[-1], 0],data[medoids[-1], 1] , marker = 'x' , s = 500)
show()
```

结论图如下所示：

```
iteration couter: 1
current total cost is  [4697.5670727327015]
iteration couter: 2
current total cost is  [4257.479273209171]
iteration couter: 3
current total cost is  [3932.7603893625455]
iteration couter: 4
current total cost is  [3932.7603893625455]
```