实验 9_任务 1_聚类

问题：

任务描述：数据中给定议员对于 16 项不同议题的投票情况（属性表示为支持 Y、反对 N、弃权？），数据的第一列为议员所属的党派类别，请使用聚类方法对上述数据进行聚类分析：

1. 尝试定量的评价的聚类结果；
2. 尝试比较不同聚类方法或者不同初始情况对聚类结果的影响；

实验过程记录：

LAB9 TASK1

调库

```
import pandas as pd
import numpy as np
import math
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

导入数据并降维

```
filename = 'E:/大学课程/AI程序设计/实验部分/实验9 聚类-关联-异常/实验课聚类关联分析/house-votes-84.csv'
data_origin = pd.read_csv(filename, engine = 'python')
```

导入数据-*Data frame*格式

```
data_origin_matrix = data_origin.values
label_lst = []
for item in data_origin_matrix:
    if item[0]=='republican':
        label_lst.append(0)#共和党为0
    else:
        label_lst.append(1)#民主党为1
```

把原始的标签提取出来

```
data_product=data_origin.drop('A',axis=1)
data_product_matrix = data_product.values
pri_X = data_product_matrix

lst_extern = []

for item in pri_X:
    lst_essen = []
    for member in item:
        if member == 'n':
            lst_essen.append(-1)
        elif member == 'y':
            lst_essen.append(1)
        else:
            lst_essen.append(0)
    lst_extern.append(lst_essen)

X = np.mat(lst_extern)
```

把投票结果'*n*'，'*y*'，'？'，分别用−1，1，0代替

```
pca = PCA(n_components = 2)
reduced_X = pca.fit_transform(X)

X_lst = []
Y_lst = []

for item in reduced_X:
    X_lst.append(item[0])
    Y_lst.append(item[1])
```

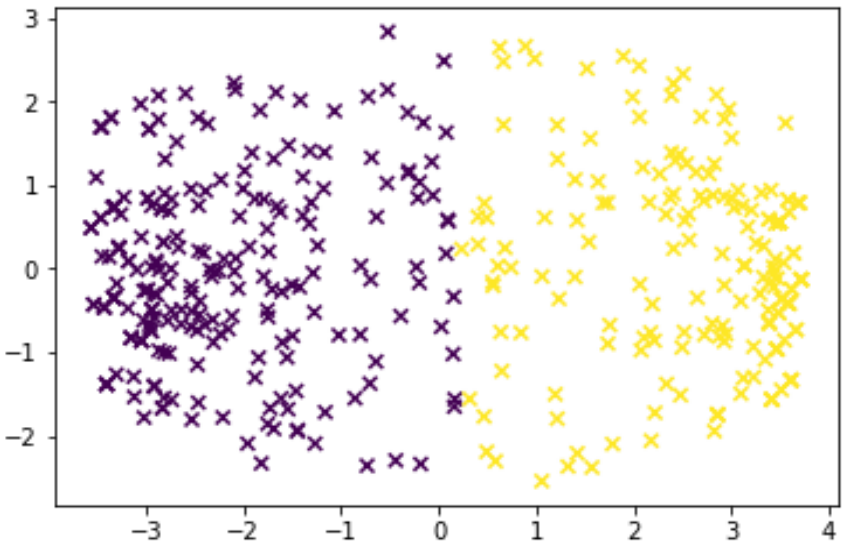
数据降维，降成3维

```
clf = KMeans(n_clusters=2)
y_pred = clf.fit_predict(reduced_X)
```

利用*Sklearn*中的*cluster*进行聚类

```
ax = plt.figure().add_subplot(111)
ax.scatter(X_lst,Y_lst, c=y_pred,marker='x')
```

```
<matplotlib.collections.PathCollection at 0xc3264e0>
```



利用*matplotlib*把聚类结果可视化

```
republic_x = []
republic_y = []

democrate_x = []
democrate_y = []

republic_x_pred = []
republic_y_pred = []

democrate_x_pred = []
democrate_y_pred = []
```

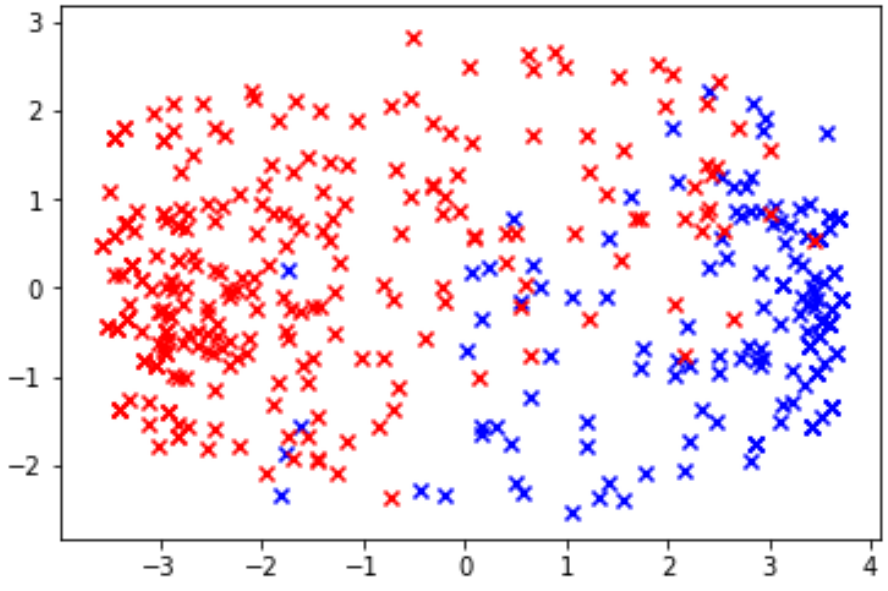
```
for i in range(435):
    if label_lst[i] == 0:
        republic_x.append(reduced_X[i][0])
        republic_y.append(reduced_X[i][1])
    else:
        democrate_x.append(reduced_X[i][0])
        democrate_y.append(reduced_X[i][1])

for i in range(435):
    if y_pred[i] == 0:
        republic_x_pred.append(reduced_X[i][0])
        republic_y_pred.append(reduced_X[i][1])
    else:
        democrate_x_pred.append(reduced_X[i][0])
        democrate_y_pred.append(reduced_X[i][1])
```

绘制原来真实的类别图

```
ax = plt.figure().add_subplot(111)
ax.scatter(republic_x,republic_y,c='blue',marker='x')
ax.scatter(democrate_x,democrate_y,c='red',marker='x')
```

```
<matplotlib.collections.PathCollection at 0xc3d6908>
```



下面尝试对该聚类结果进行评估

首先找到两个聚类的中心

```
centroids=clf.cluster_centers_
```

定义平面上两点求距离的函数

```
def dist(M,N):
    distance = math.sqrt((pow(M[0]-N[0],2)+pow(M[1]-N[1],2)))
    return distance
```

下面寻求$avg(C)$

```
C_republic = len(republic_x_pred)#199
C_democrate = len(democrate_x_pred)#236
```

```
republic_coordinate_set = []
for i in range(C_republic):
    single_coordinate = []
    single_coordinate.append(republic_x_pred[i])
    single_coordinate.append(republic_y_pred[i])
    republic_coordinate_set.append(single_coordinate)

democrate_coordinate_set = []
for i in range(C_democrate):
    single_coordinate = []
    single_coordinate.append(democrate_x_pred[i])
    single_coordinate.append(democrate_y_pred[i])
    democrate_coordinate_set.append(single_coordinate)

sum_distance_republic = 0
for i in range(C_republic):
    for j in range(C_republic):
        if j>i:
            sum_distance_republic=sum_distance_republic+dist(republic_coordinate_set[i],republic_coordinate_set[j])
avg_republic_C = 2/((C_republic-1)*C_republic)*sum_distance_republic

sum_distance_democrate = 0
for i in range(C_democrate):
    for j in range(C_democrate):
        if j>i:
            sum_distance_democrate=sum_distance_democrate+dist(democrate_coordinate_set[i],democrate_coordinate_set[j])
avg_democrate_C = 2/((C_democrate-1)*C_democrate)*sum_distance_democrate
```

聚类得到的共和党派的$avg(c)$为1.8949841881585363

聚类得到的民主党派的$avg(c)$为1.9125389919760962

下面求$diam(C)$

```
republic_distance_max = 0
for i in range(C_republic):
    for j in range(C_republic):
        if j>i:
            if dist(republic_coordinate_set[i],republic_coordinate_set[j]) > republic_distance_max:
                republic_distance_max = dist(republic_coordinate_set[i],republic_coordinate_set[j])

democrate_distance_max = 0
for i in range(C_democrate):
    for j in range(C_democrate):
        if j>i:
            if dist(democrate_coordinate_set[i],democrate_coordinate_set[j]) > democrate_distance_max:
                democrate_distance_max = dist(democrate_coordinate_set[i],democrate_coordinate_set[j])
```

聚类得到的共和党派的$diam(c)$为5.198664619227401

聚类得到的民主党派的$diam(c)$为5.315512164657216

下面求$d_{min}(C_i,C_j)$

```
distance_between_clusters = 200
for i in range(C_republic):
    for j in range(C_democrate):
        if dist(republic_coordinate_set[i],democrate_coordinate_set[j]) < distance_between_clusters:
            distance_between_clusters = dist(republic_coordinate_set[i],democrate_coordinate_set[j])
```

两个聚类的簇内最近的两个样本的距离为：0.15248405503877316

下面求$d_{cen}(C_i,C_j)$

```
distance_between_two_centers = dist(centroids[0],centroids[1])
```

两个簇的中心点的距离为4.851762143635815

下面对真实的分类做检测

下面寻求真实分布$avg(C)$

```
C_republic_real = len(republic_x)#168
C_democrate_real = len(democrate_x)#267
```

```
republic_coordinate_real_set = []
for i in range(C_republic_real):
    single_coordinate = []
    single_coordinate.append(republic_x[i])
    single_coordinate.append(republic_y[i])
    republic_coordinate_real_set.append(single_coordinate)

democrate_coordinate_real_set = []
for i in range(C_democrate_real):
    single_coordinate = []
    single_coordinate.append(democrate_x[i])
    single_coordinate.append(democrate_y[i])
    democrate_coordinate_real_set.append(single_coordinate)

sum_distance_republic_real = 0
for i in range(C_republic_real):
    for j in range(C_republic_real):
        if j>i:
            sum_distance_republic_real=sum_distance_republic_real+dist(republic_coordinate_real_set[i],republic_coo
avg_republic_C_real = 2/((C_republic_real-1)*C_republic_real)*sum_distance_republic_real

sum_distance_democrate_real = 0
for i in range(C_democrate_real):
    for j in range(C_democrate_real):
        if j>i:
            sum_distance_democrate_real=sum_distance_democrate_real+dist(democrate_coordinate_real_set[i],democrate_
avg_democrate_C_real = 2/((C_democrate_real-1)*C_democrate_real)*sum_distance_democrate_real
```

真实的共和党派的$avg(c)$为1.9279370693855515

真实的民主党派的$avg(c)$为2.5163726676860785

```
republic_distance_max_real = 0
for i in range(C_republic_real):
    for j in range(C_republic_real):
        if j>i:
            if dist(republic_coordinate_real_set[i],republic_coordinate_real_set[j]) > republic_distance_max_real:
                republic_distance_max_real = dist(republic_coordinate_real_set[i],republic_coordinate_real_set[j])

democrate_distance_max_real = 0
for i in range(C_democrate_real):
    for j in range(C_democrate_real):
        if j>i:
            if dist(democrate_coordinate_real_set[i],democrate_coordinate_real_set[j]) > democrate_distance_max_rea
                democrate_distance_max_real = dist(democrate_coordinate_real_set[i],democrate_coordinate_real_set[j
```

真实的共和党派的$diam(c)$为6.734401503840723

真实的民主党派的$diam(c)$为7.111757611276915

下面求真实$d_{min}(C_i, C_j)$

```
distance_between_clusters_real = 200
for i in range(C_republic_real):
    for j in range(C_democrate_real):
        if dist(republic_coordinate_real_set[i],democrate_coordinate_real_set[j]) < distance_between_clusters_real:
            distance_between_clusters_real = dist(republic_coordinate_real_set[i],democrate_coordinate_real_set[j])
print(distance_between_clusters_real)
```

```
0.042264160918674486
```

真实的两个簇内最近的两个样本的距离为：0.042264160918674486

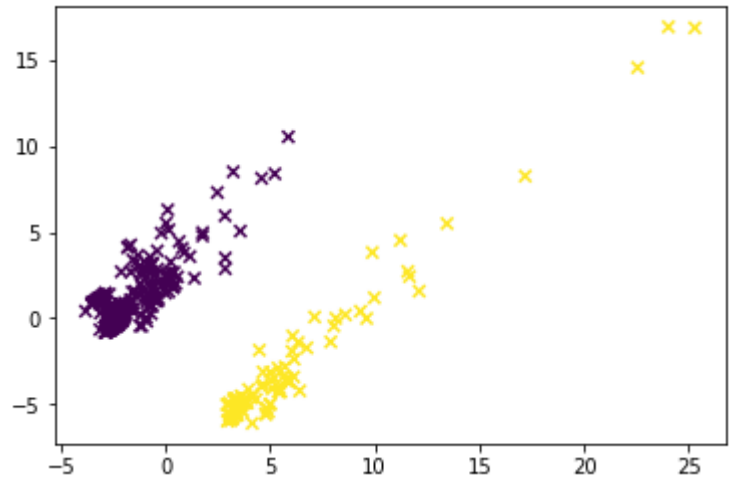
问题 1：

I．数据处理主要包括以下步骤：

1） 用 1，-1，0 分别代替，‘y’，‘n’，‘？’。

2） 把多维数据降维为 2 维（使用上述数据替代方法的原因也是为了在降维中尽量减少对数据的影响---保持‘y’，‘n’的同等效果，减少‘？’的影响）

注：尝试过用 9 来代替‘？’，结果产生了如下的聚类结果：



3） 在对聚类做评价时，采用了内部评价指标，即

$$\mathrm{avg}(C) = \frac{2}{|C|(|C|-1)} \sum_{1 \leqslant i < j \leqslant |C|} \mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

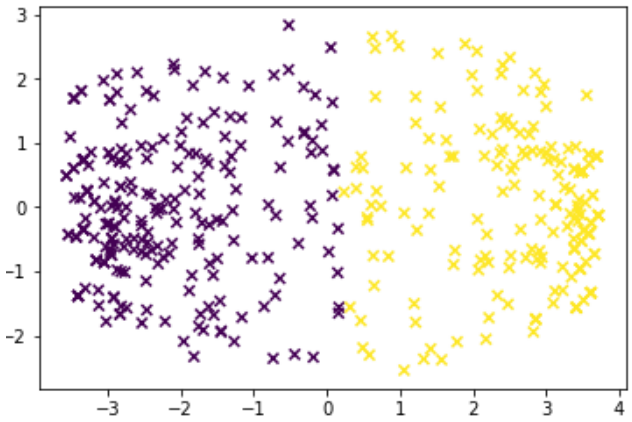
$$\mathrm{diam}(C) = \max_{1 \leqslant i < j \leqslant |C|} \mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

$$d_{\min}(C_i, C_j) = \min_{\boldsymbol{x}_i \in C_i, \boldsymbol{x}_j \in C_j} \mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

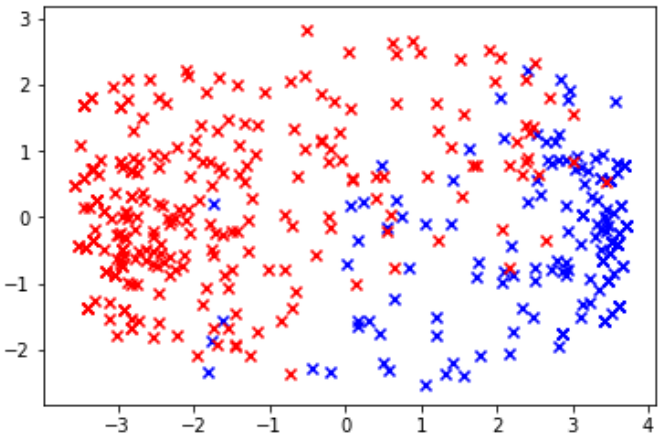
$$d_{\mathrm{cen}}(C_i, C_j) = \mathrm{dist}(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j)$$

II.聚类成果

1） 可视化：



----→此图为聚类结果的可视化图



----→此图是按照真实的党派标签的结果

III． 聚类评价

聚类得到的各个指标：

1）簇内平均距离：

聚类得到的共和党派的 avg(c)为 1.8949841881585363

聚类得到的民主党派的 avg(c)为 1.9125389919760962

2） 簇内最大距离：

聚类得到的共和党派的 diam(c)为 5.198664619227401

聚类得到的民主党派的 diam(c)为 5.315512164657216

3） 簇间最小距离：

两个聚类的簇最近的两个样本的距离为：0.15248405503877316

4） 簇中心距离：

两个簇的中心点的距离为 4.851762143635815

真实分类的各个指标：

1） 簇内平均距离：

真实的共和党派的 avg(c)为 1.9279370693855515

真实的民主党派的 avg(c)为 2.5163726676860785

2） 簇内最大距离：

真实的共和党派的 diam(c)为 6.734401503840723

真实的民主党派的 diam(c)为 7.111757611276915

3） 簇间最小距离：

真实的两个的簇最近的两个样本的距离为：0.042264160918674486

问题 2：

由于在 KMeans 聚类算法中，每次的中心点是随机选择的，也就是说，每次运行聚类库函数的代码，都会随机选择不同的中心。所以说，即便每次运行了聚类函数，产生的聚类结果并无明显差异，而且所得评价指标的值均未发生改变，所以，我认为，选取不同的初始位点，对于聚类的最终结果没有影响。