机器学习与数据挖掘-HW4

—Clustering Techniques—

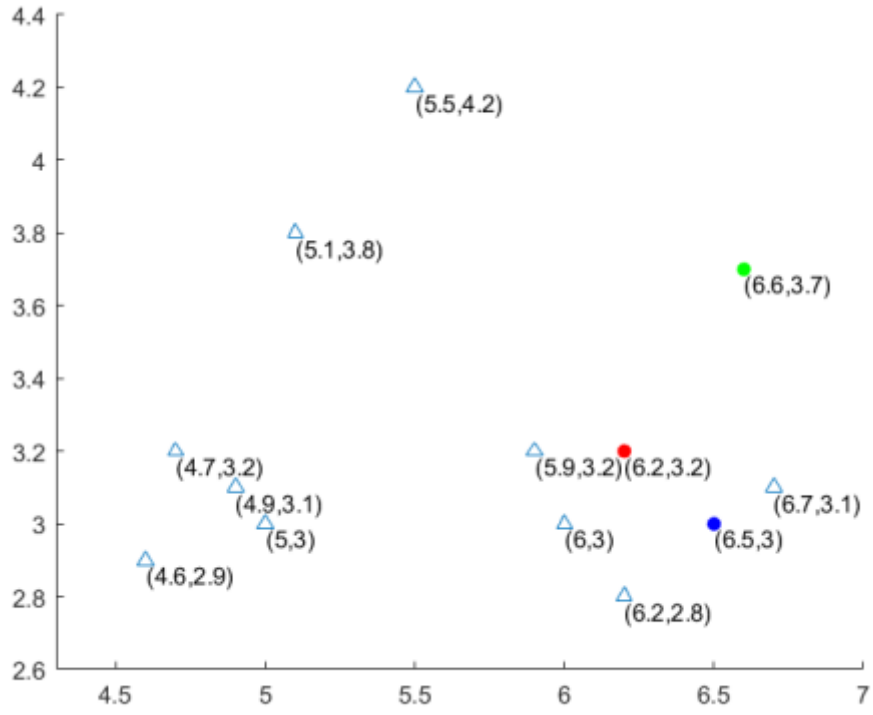
19335253 葉珺明

---

1 Ex1: Implement K-Means Manually

> K-Means:
>
> - 给定样本集：$D = x_1, x_2, x_3 \dots x_m$
> - 针对聚类划分：$C = C_1, C_2 C_3 \dots C_k$
> - 最小化平方误差：
>
> $$E = \sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu||_2^2$$
>
> 其中$\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$是簇$C_i$的均值向量

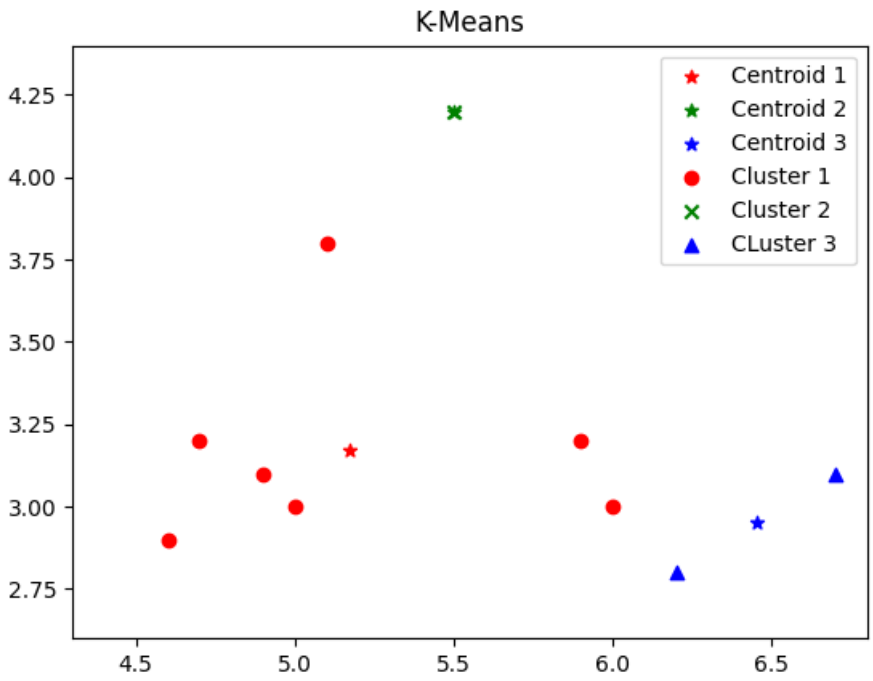
初始散点分布：



1.a The Center of Cluster Red after one iteration

经过一次迭代后，μ_1的均值向量为：$(5.1714, 3.1714)$

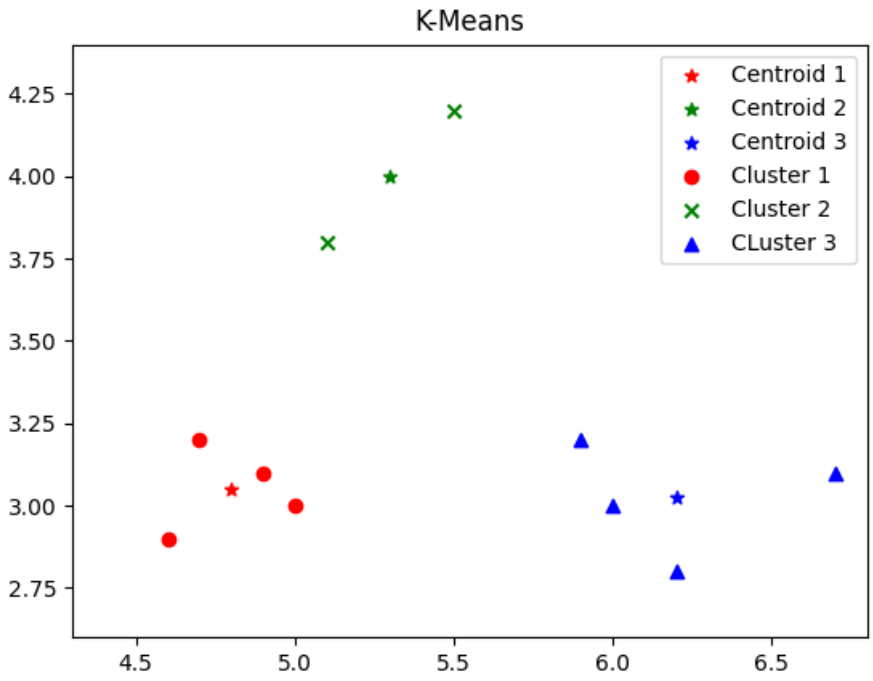
此时散点分布：



1.b The Center of Cluster Green after one iteration

经过两次迭代后，μ_2的均值向量为：$(5.3, 4.0)$

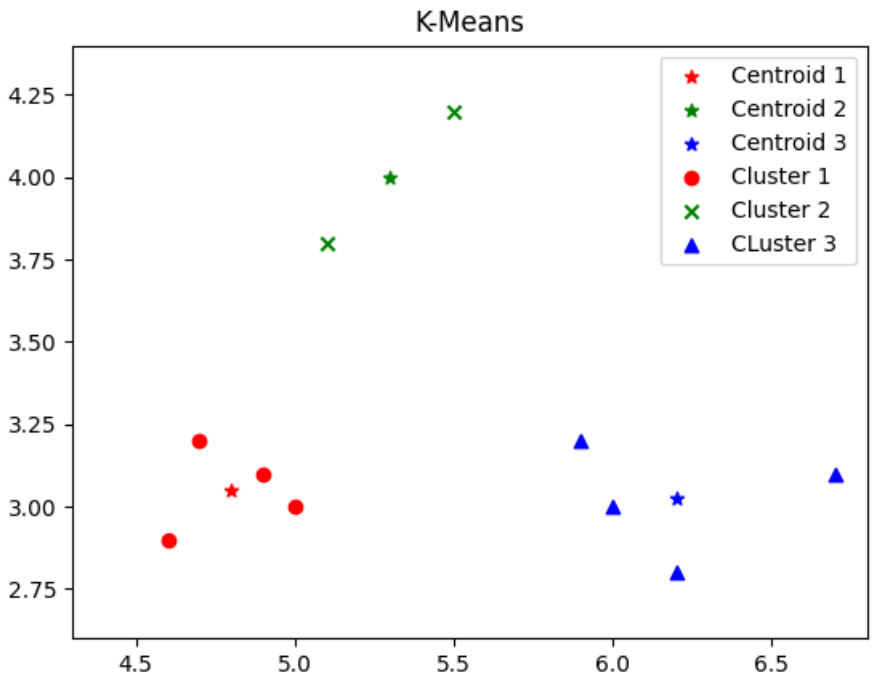
此时散点分布：



1.c The Center of Cluster Blue when clustering converges

收敛迭代后，μ_3的均值向量为：$[6.2，3.025]$

此时散点分布：



1.d The number of iterations

由1.b和1.c散点分布比较发现，第二次结果与收敛后的结果相同，故要使三个均值向量收敛的迭代次数为2，第三次的均值向量计算结果与第二次计算所得一致。

2 Ex2: Application of K-Means

2.a dataset A: A2

2.b dataset B: B2

2.c dataset C: C2

2.d dataset D: D1

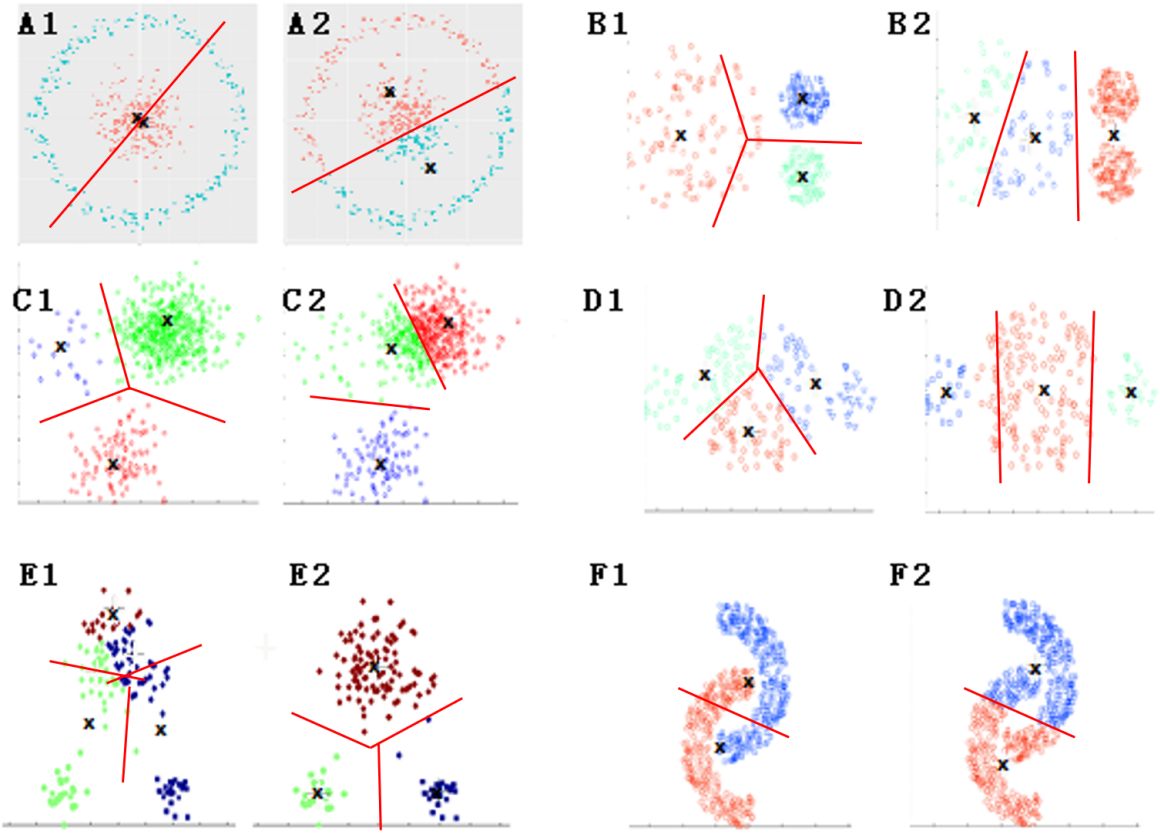
2.e dataset E: E2

2.f dataset F: F2

2.g Reason to Q(a)~Q(f)

对图中聚类中心作两两的垂直平分线，当K-Means算法收敛时，垂直平分线即为聚类的分界

有：



故有第一题答案，$A2$，$B2$，$C3$，$D1$，$E2$，$F2$

2.h Other Clustering Algorithms

对于数据集F，K-Means的表现不好，在数据集F中，聚类的结果应为两个弧形的数据集

适用于数据集F的聚类算法：

- DBSCAN（有代表性的基于密度的聚类算法）
- Spectral Clustring（谱聚类）
- Agglomerative Clustering（凝聚法层次聚类，e.g. Ward）

3 Ex3: Applications of Clustering Techniques in IR and DM

- 聚类在Information Retrieval 的应用：
  - 搜索结果的聚类（对搜索出来的结果聚类再选择性地向用户展示）
    - 提供面向用户的更有效的展示
  - 基于文档聚类的检索（先对文档聚类，信息检索时返回某个最相关聚类）
    - 加快了搜索的速度

        - 提高了搜索召回率
- 聚类在data mining 的应用：
    - 商品划分
        - 在交易数据库中, 顾客一次购买的商品(数据项)构成了一条交易, 将经常同时购买的数据项聚类到一起有利于改善商品的布置, 提高销售利润 。
    - 顾客划分
        - 将具有相似的购买模式的顾客聚类到一起, 分析每一类顾客的特征, 有利于对特定的顾客群进行特定商品的宣传和销售。
    - 模式识别
        - 在医疗分析中, 通过对一组新型疾病聚类, 得到每类疾病的特征描述, 就可以对这些疾病进行识别, 提高治疗的功效。
    - 趋势分析
        - 在天文学上, 研究人员利用聚类分析宇宙仿真系统得到的数据, 更好地理解黑洞形成和进化的物理过程。
        - 金融股票的预测分析，通过对不同时间段和不同支股票聚类，预测股票未来的趋势。

核心代码

- Ex1：

```
import numpy as np
from pylab import *

er = 1e-3

D = [[6.2, 3.2], [6.6, 3.7], [6.5, 3.0]]
D = np.array(D)
X = [[5.9, 3.2], [4.6, 2.9], [6.2, 2.8], [4.7, 3.2], [5.5, 4.2],
    [5.0, 3.0], [4.9, 3.1], [6.7, 3.1], [5.1, 3.8], [6.0, 3.0]]
X = np.array(X)

def draw_pic(C0, C1, C2, D):

    plt.scatter(D[0][0], D[0][1], color='r', marker='*', label='Centroid 1')
    plt.scatter(D[1][0], D[1][1], color='g', marker='*', label='Centroid 2')
    plt.scatter(D[2][0], D[2][1], color='b', marker='*', label='Centroid 3')

    x0 = []
    y0 = []
    for i in range (len(C0)):
        x0.append(C0[i][0])
        y0.append(C0[i][1])

    plt.scatter(x0, y0, color='r', marker='o', label='Cluster 1')

    x1 = []
    y1 = []
    for i in range (len(C1)):
        x1.append(C1[i][0])
        y1.append(C1[i][1])

```

```
    plt.scatter(x1, y1, color='g', marker='x', label='Cluster 2')

    x2 = []
    y2 = []
    for i in range (len(C2)):
        x2.append(C2[i][0])
        y2.append(C2[i][1])

    plt.scatter(x2, y2, color='b', marker='^', label='CLuster 3')
    plt.legend(loc="upper right")
    plt.title('K-Means')

    plt.xlim((4.3, 6.8))
    plt.ylim((2.6, 4.4))
    plt.show()


def update_C(prev_D):
    new_D = np.mean(prev_D, 0)
    return new_D.tolist()

n = 0
while(1):
    r = []
    g = []
    b = []
    for i in range (len(X)):
        d0 = math.sqrt(np.sum((X[i]-D[0])**2))
        d1 = math.sqrt(np.sum((X[i]-D[1])**2))
        d2 = math.sqrt(np.sum((X[i]-D[2])**2))

        if d0 >= d1:
            if d1 >= d2:
                b.append(X[i])
            else:
                g.append(X[i])
        elif d0 >= d2:
            if d2 >= d1:
                g.append(X[i])
            else:
                b.append(X[i])
        else:
            r.append(X[i])

    new_D0 = update_C(r)
    new_D1 = update_C(g)
    new_D2 = update_C(b)
    if math.fabs(math.sqrt(np.sum(new_D0-D[0])**2)) >= er \
        or math.fabs(math.sqrt(np.sum(new_D1-D[1])**2)) >= er\
            or math.fabs(math.sqrt(np.sum(new_D2-D[2])**2)) >= er:

        n += 1
        D[0] = new_D0
        D[1] = new_D1
        D[2] = new_D2
        draw_pic(r, g, b, D)
        print(D)
    else:
```

```
            print(D)
            break

    print(n)
```