

K-means聚类的实现以及案例讲解

目录

1 k-means聚类步骤

2. 案例联系

3. K-means的api初步使用

3.1 api 介绍

4. 案列

4.1 流程分析

4.2 代码实现

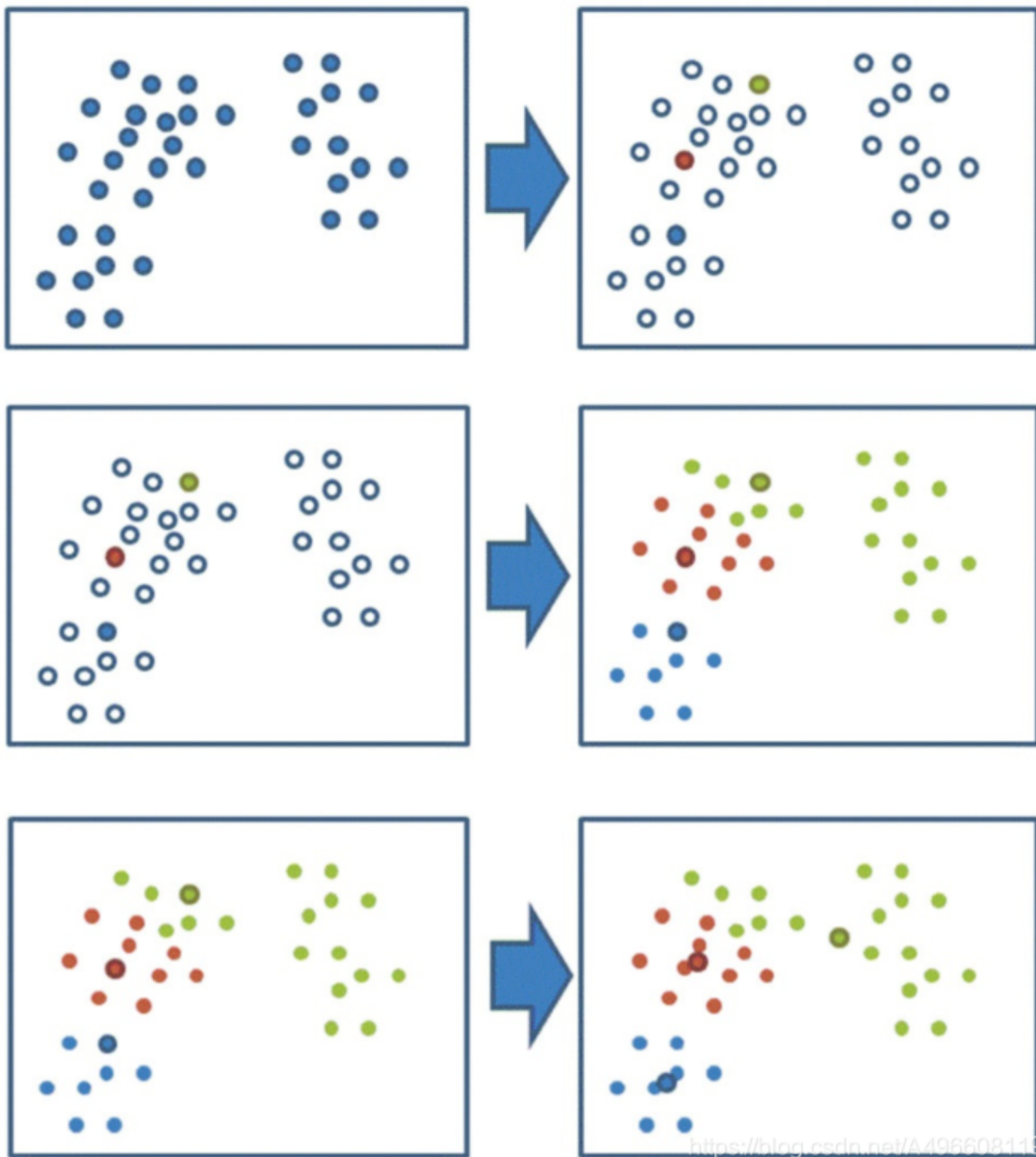
4.3 完整代码

4.4 实验结果

1 k-means聚类步骤

- 1、 随机设置K个特征空间内的点作为初始的聚类中心
- 2、 对于其他每个点计算到K个中心的距离， 未知的点选择最近的一个聚类中心点作为标记类别
- 3、 接着对着标记的聚类中心之后， 重新计算出每个聚类的新中心点（平均值）
- 4、 如果计算得出的新中心点与原中心点一样（质心不再移动） ， 那么结束， 否则重新进行第二步过程

通过下图解释实现流程：



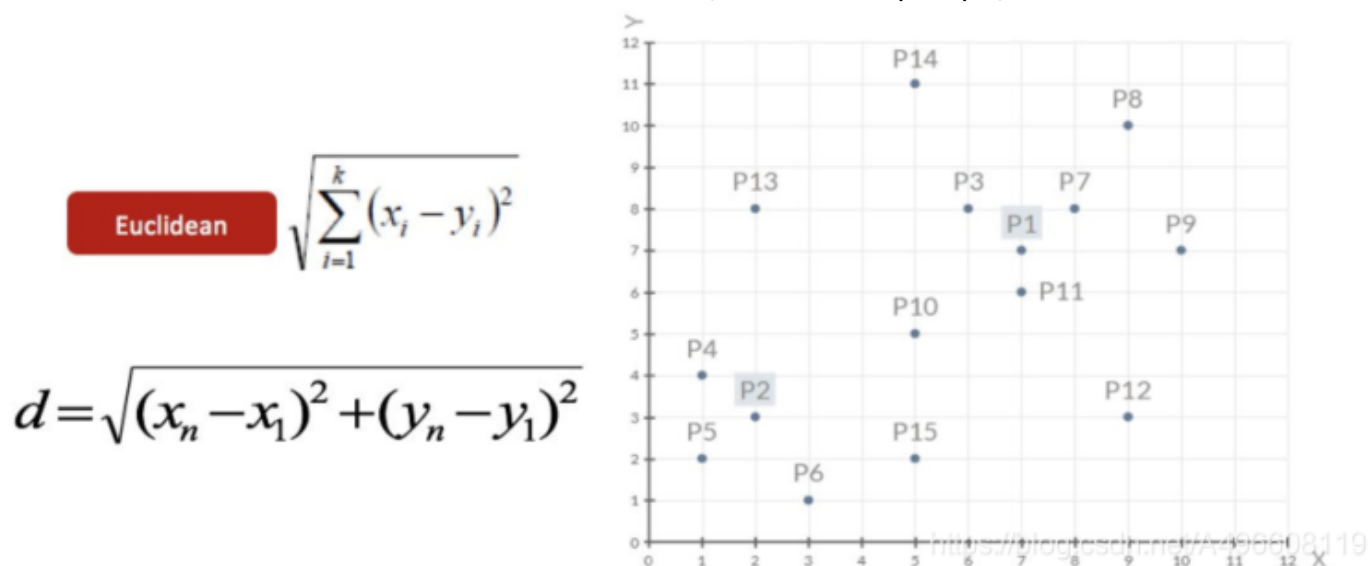
2. 案例联系

	X 值	Y 值
P1	7	7
P2	2	3
P3	6	8
P4	1	4
P5	1	2
P6	3	1
P7	8	8

	X 值	Y 值
P8	9	10
P9	10	7
P10	5	5
P11	7	6
P12	9	3
P13	2	8
P14	5	11
P15	5	2

<https://blog.csdn.net/A496608119>

- 1、随机设置K个特征空间内的点作为初始的聚类中心（本案例中设置p1和p2）



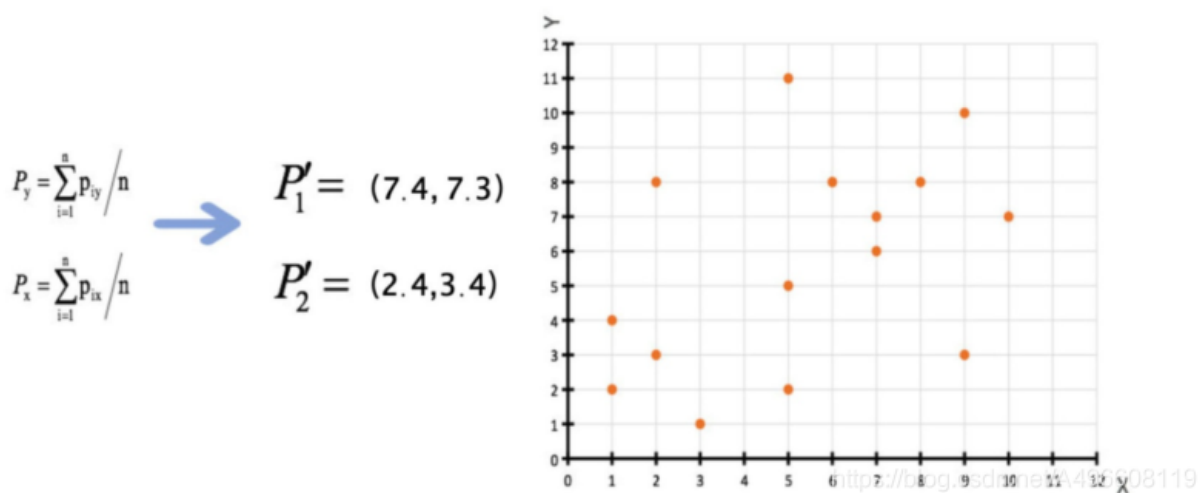
- 2、对于其他每个点计算到K个中心的距离，未知的点选择最近的一个聚类中心点作为标记类别

	P1 (7,7)	P2 (2,3)
P3	1.41	6.40
P4	6.71	1.41
P5	7.81	1.41
P6	7.21	2.24
P7	1.41	7.81
P8	3.61	9.90

	P1 (7,7)	P2 (2,3)
P9	3	8.94
P10	2.83	3.61
P11	1	5.83
P12	4.47	7.00
P13	5.10	5.00
P14	4.47	8.54
P15	5.39	3.16

P1	P3	P7	P8	P9	P10	P11	P12	P14
P2	P4	P5	P6	P13	P15			

3、接着对着标记的聚类中心之后，重新计算出每个聚类的新中心点（平均值）



4、如果计算得出的新中心点与原中心点一样（质心不再移动），那么结束，否则重新进行第二步过程【经过判断，需要重复上述步骤，开始新一轮迭代。

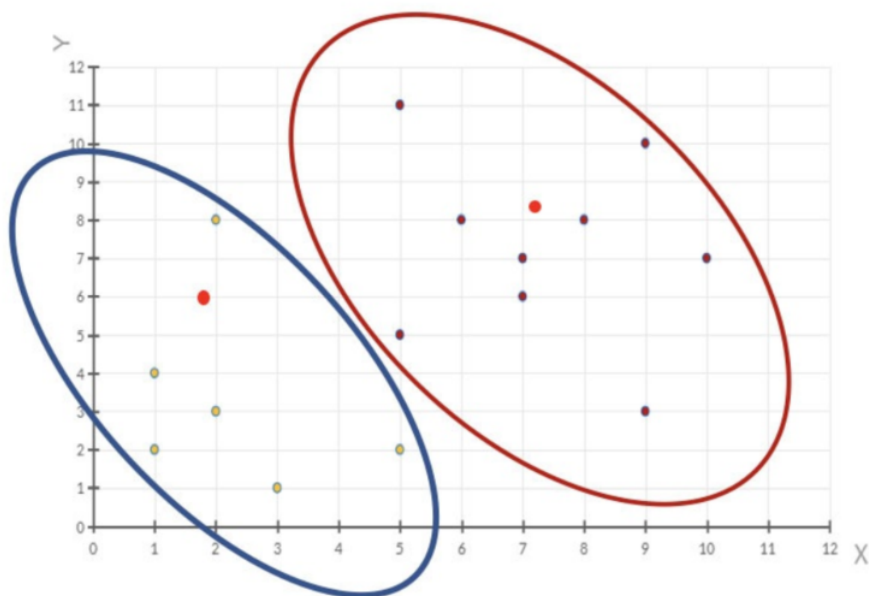
	$P'_1 (7.3, 7.2)$	$P'_2 (1.8, 4.6)$
P1	0.36	5.73
P2	6.75	1.61
P3	1.39	5.40
P4	7.02	1.00
P5	8.16	2.72
P6	7.57	3.79
P7	1.06	7.07

	$P'_1 (7.3, 7.2)$	$P'_2 (1.8, 4.6)$
P8	3.24	9.00
P9	2.82	8.54
P10	3.18	3.22
P11	1.32	5.39
P12	4.66	7.38
P13	5.25	3.41
P14	4.30	7.16
P15	5.25	3.41

P'_1	P1	P3	P7	P8	P9	P10	P11	P12	P14
P'_2	P2	P4	P5	P6	P13	P15			

<https://blog.csdn.net/A496608119>

5、当每次迭代结果不变时，认为算法收敛，聚类完成，K-Means一定会停下，不可能陷入一直选质心的过程。



<https://blog.csdn.net/A496608119>

3. K-means的api初步使用

3.1 api 介绍

- `sklearn.cluster.KMeans(n_clusters=8)`
 - 参数:
 - `n_clusters`: 开始的聚类中心数量
 - 整型, 缺省值=8, 生成的聚类数, 即产生的质心 (centroids) 数。
 - 方法:
 - `estimator.fit(x)`
 - `estimator.predict(x)`
 - `estimator.fit_predict(x)`
 - 计算聚类中心并预测每个样本属于哪个类别, 相当于先调用`fit(x)`, 然后再调用`predict(x)`

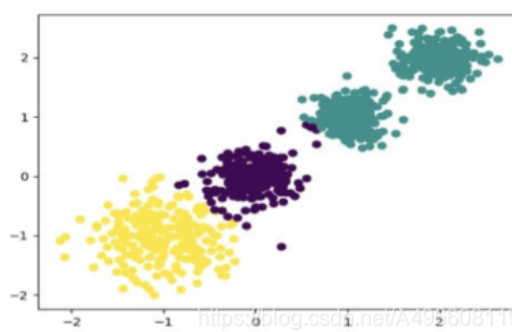
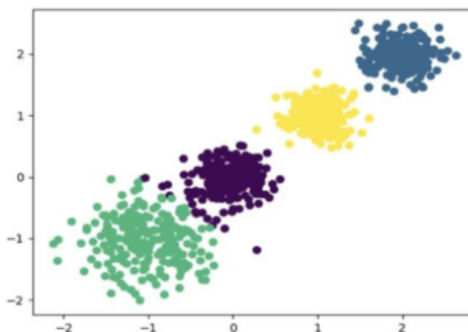
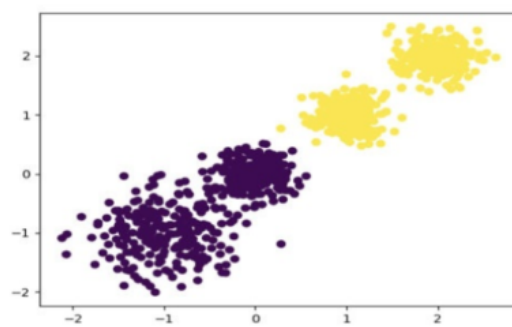
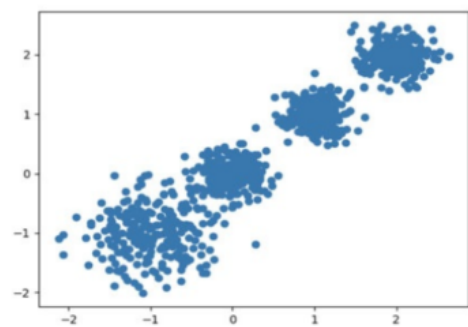
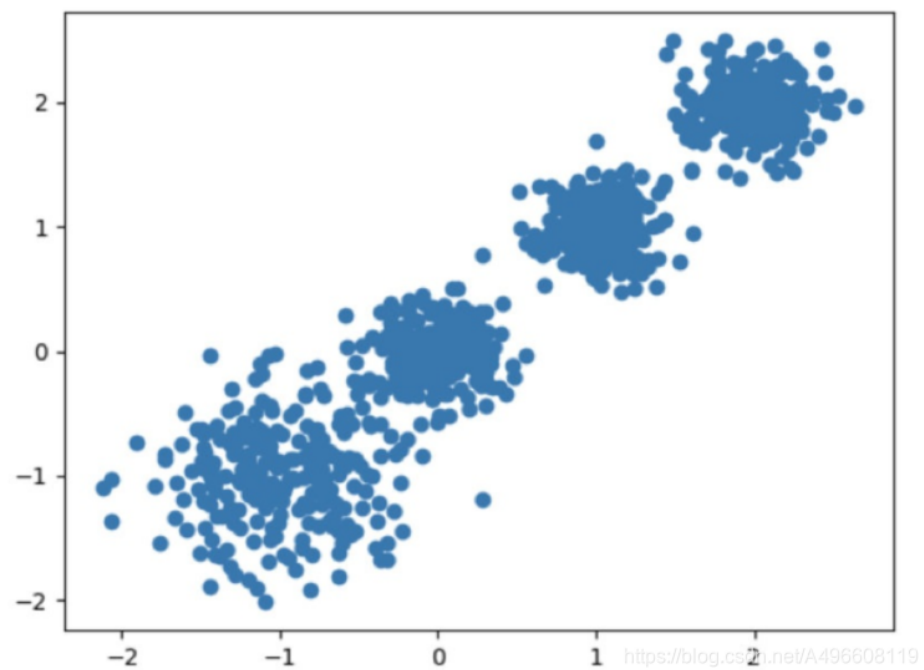
4. 案例

随机创建不同二维数据集作为训练集, 并结合k-means算法将其聚类, 你可以尝试分别聚类不同数量的簇, 并观察聚类

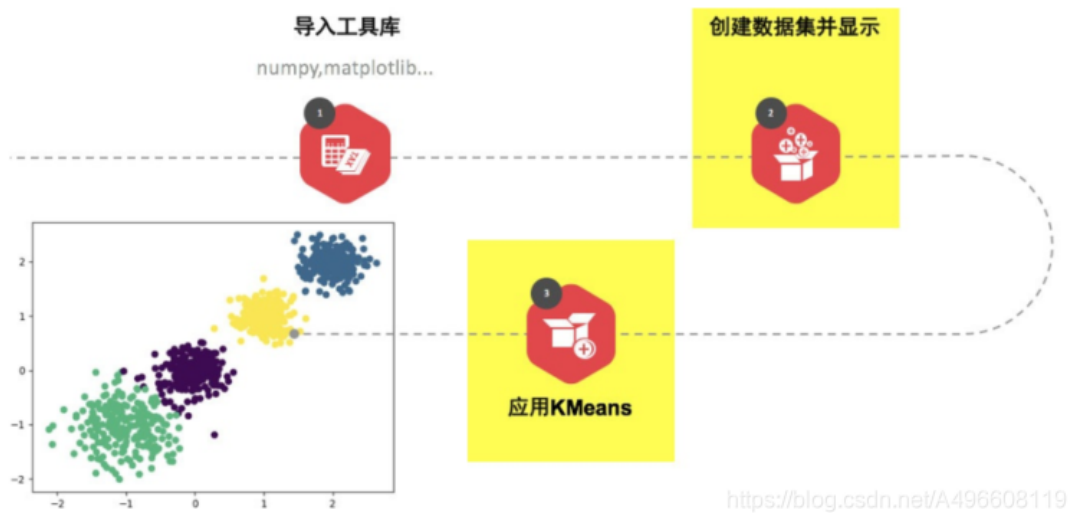
效果:

聚类参数`n_cluster`传值不同, 得

到的聚类结果不同



4.1 流程分析



4.2 代码实现

1. 创建数据集

```
1. import matplotlib.pyplot as plt

2. from sklearn.datasets.samples_generator import make_blobs

3. from sklearn.cluster import KMeans

4. from sklearn.metrics import calinski_harabaz_score

5. # 创建数据集

6. # x为样本特征， y为样本簇类别， 共1000个样本， 每个样本2个特征， 共4个簇，

7. # 簇中心在[-1,-1], [0,0], [1,1], [2,2]， 簇方差分别为[0.4, 0.2, 0.2, 0.2]

8. X, y = make_blobs(n_samples=1000, n_features=2, centers=[[-1, -1], [0,
    0], [1, 1], [2, 2]],

9. cluster_std=[0.4, 0.2, 0.2, 0.2],

10. random_state=9)

11. # 数据集可视化

12. plt.scatter(X[:, 0], X[:, 1], marker='o')

13. plt.show()
```

2. 使用K-means进行聚类，并使用CH 方法进行评估

CH 系数 (Calinski-Harabasz Index)

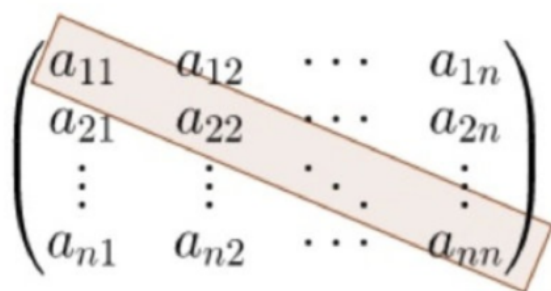
类别内部数据的协方差越小越好，类别之间的协方差越大越好（换句话说：类别内部数据的距离平方和越小越好，类别之间的距离平方和越大越好），这样的Calinski-Harabasz分数s会高，分数s高则聚类效果越好。

这样的Calinski-Harabasz分数s会高，分数s高则聚类效果越好。

$$s(k) = \frac{tr(B_k)}{tr(W_k)} \frac{m-k}{k-1}$$

tr为矩阵的迹, B_k 为类别之间的协方差矩阵, W_k 为类别内部数据的协方差矩阵;

m为训练集样本数, k为类别数。



$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

迹，定义为： $a_{11} + a_{22} + \cdots + a_{nn}$

使用矩阵的迹进行求解的理解：矩阵的对角线可以表示一个物体的相似性

在机器学习里，主要为了获取数据的特征值，那么就是说，在任何一个矩阵计算出来之后，都可以简单化，只要获取矩阵的迹，就可以表示这一块数据的最重要的特征了，这样就可以把很多无关紧要的数据删除掉，达到简化数据，提高处理速度。

CH需要达到的目的：

用尽量少的类别聚类尽量多的样本，同时获得较好的聚类效果。

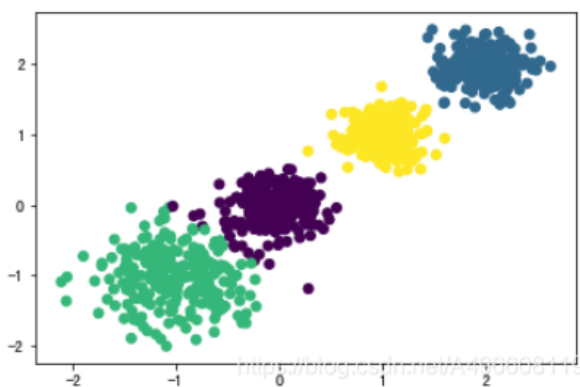
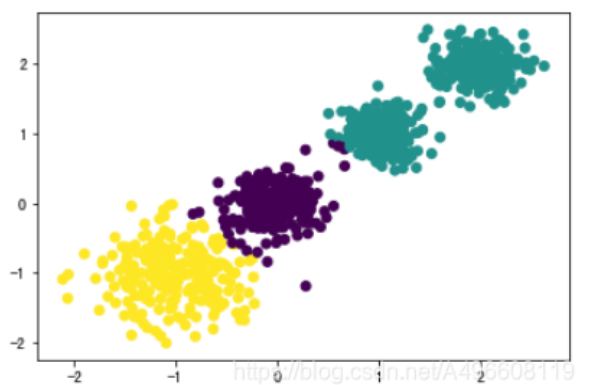
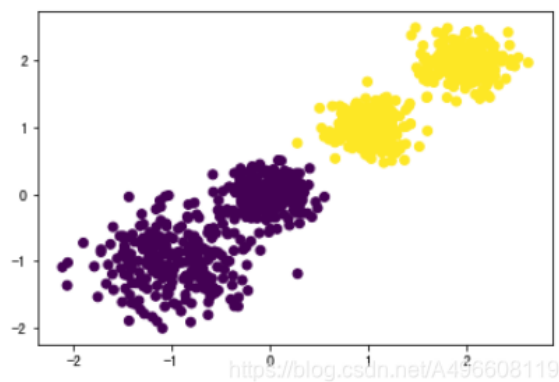
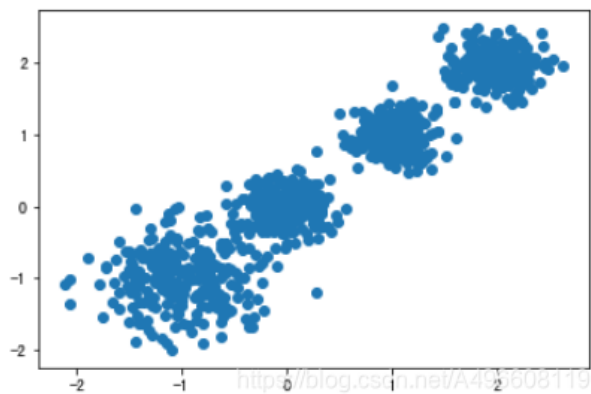
4.4 实验结果

原始数据

K=2

K = 3

K = 4



CH:3116.1706763322227

CH:2931.625030199556

CH: 5924.050613480169