黑马程序员™ | 传智播客旗下
www.itheima.com
高端IT教育品牌

专题：聚类算法

黑马程序员
www.itheima.com
传智播客旗下高端IT教育品牌

目录 Contents

目标

TARGET

- 通过k-means算法了解聚类算法基本流程
- 学会评估及优化聚类算法的方法
- 深入了解其他常见聚类算法
- 完成实践项目，学会通过聚类进行数据分析，挖掘商业价值

黑马程序员
www.itheima.com
传智播客旗下高端IT教育品牌

目录 Contents

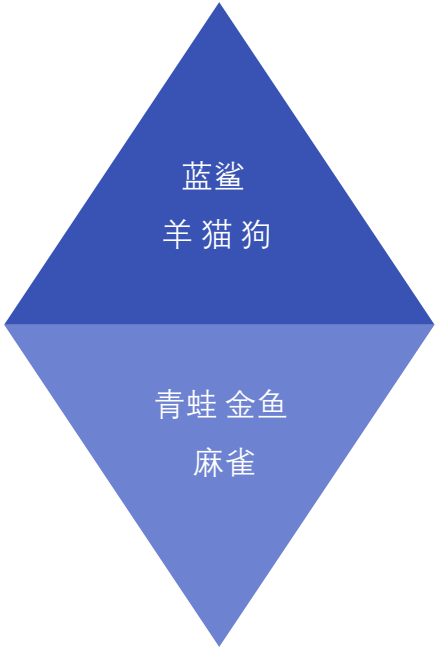
1 算法原理

1-1 课题导入

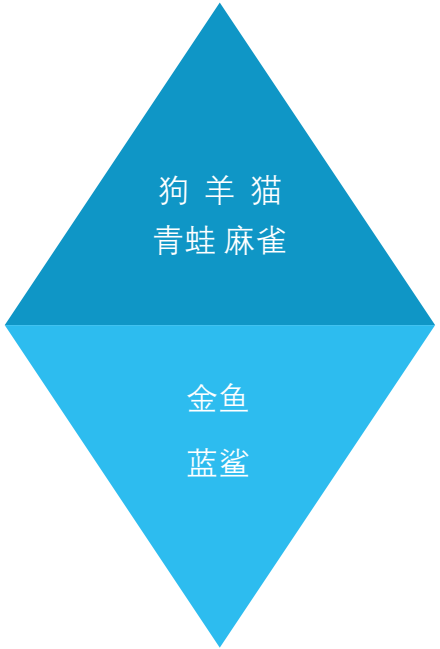
标签化

1 算法原理

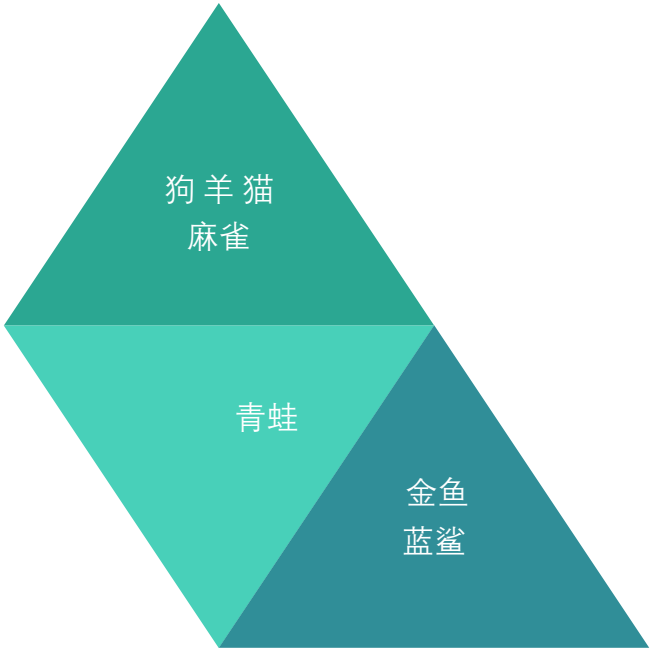
1-1 课题导入



繁衍后代方式



呼吸方式

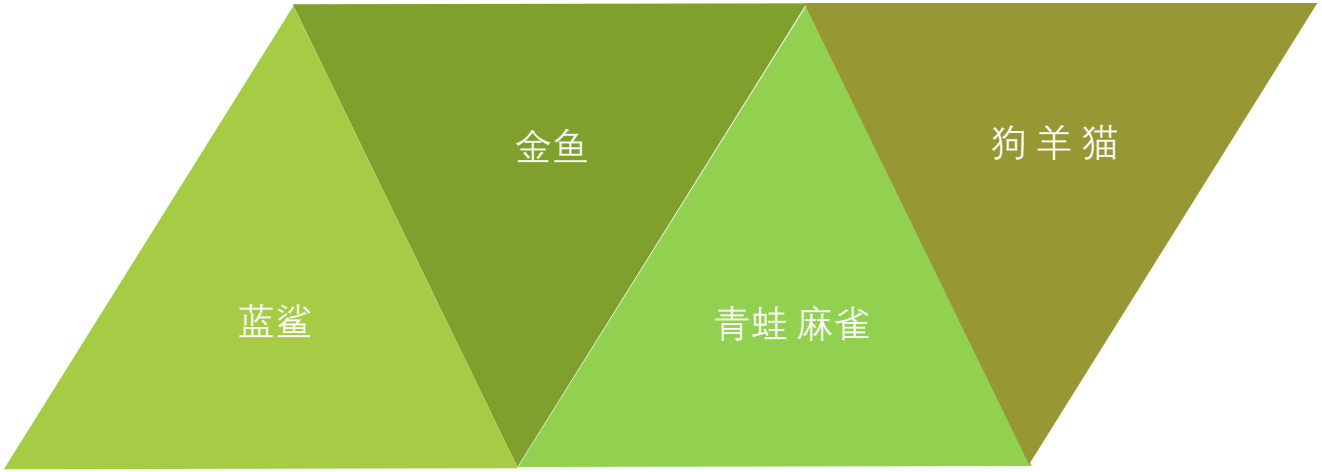


生活环境

1 算法原理

1-1 课题导入

使用不同的聚类准则，产生的聚类结果不同。



繁衍后代方式及呼吸方式(复合聚类)

1 算法原理

1-1 课题导入

基于位置信息的商业推送

新闻聚类，筛选排序

用户画像，广告推荐

Data Segmentation

搜索引擎的流量推荐

恶意流量识别

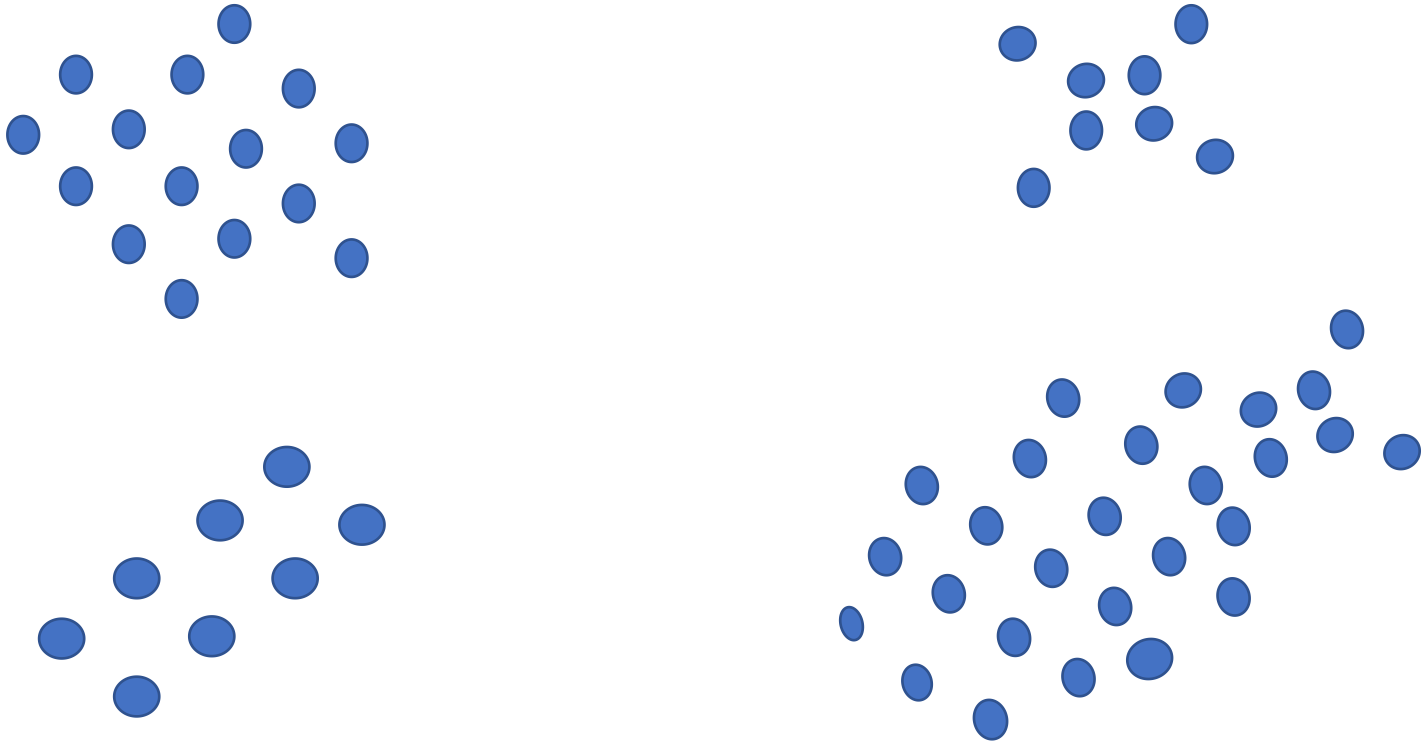
图像分割，降维，识别

离群点检测，信用卡异常消费

发掘相同功能的基因片段

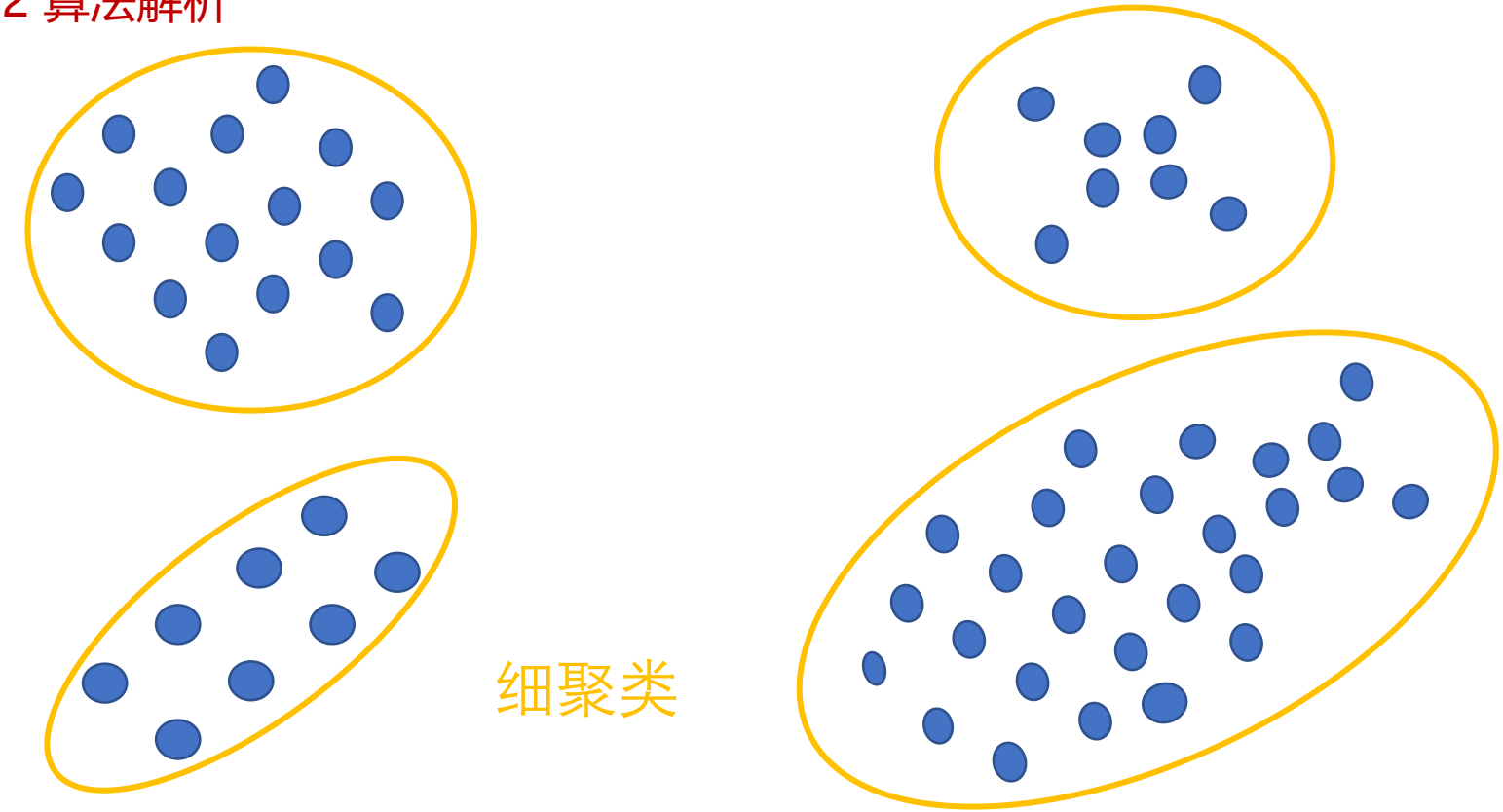
1 算法原理

1-2 算法解析



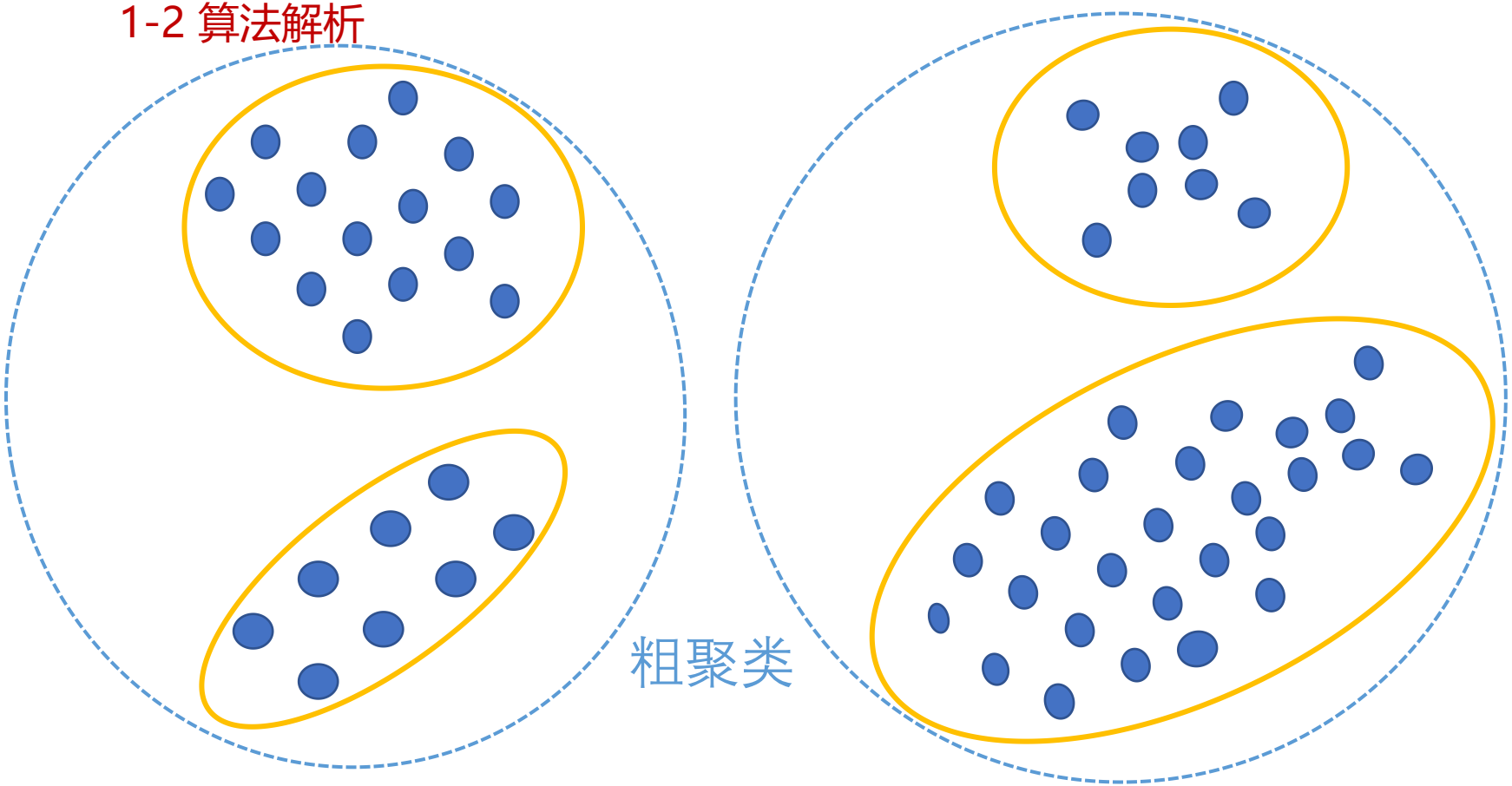
1 算法原理

1-2 算法解析



1 算法原理

1-2 算法解析



1 算法原理

黑马程序员
www.itheima.com
传智播客旗下高端IT教育品牌

1-2 算法解析

聚类算法：

一种典型的**无监督**学习算法，主要用于将相似的样本自动归到一个类别中。在聚类算法中根据样本之间的相似性，将样本划分到不同的类别中，对于不同的相似度计算方法，会得到不同的聚类结果，常用的相似度计算方法有欧式距离法。

聚类算法与分类算法最大的区别：

聚类算法是无监督的学习算法，而分类算法属于监督的学习算法。

1 算法原理

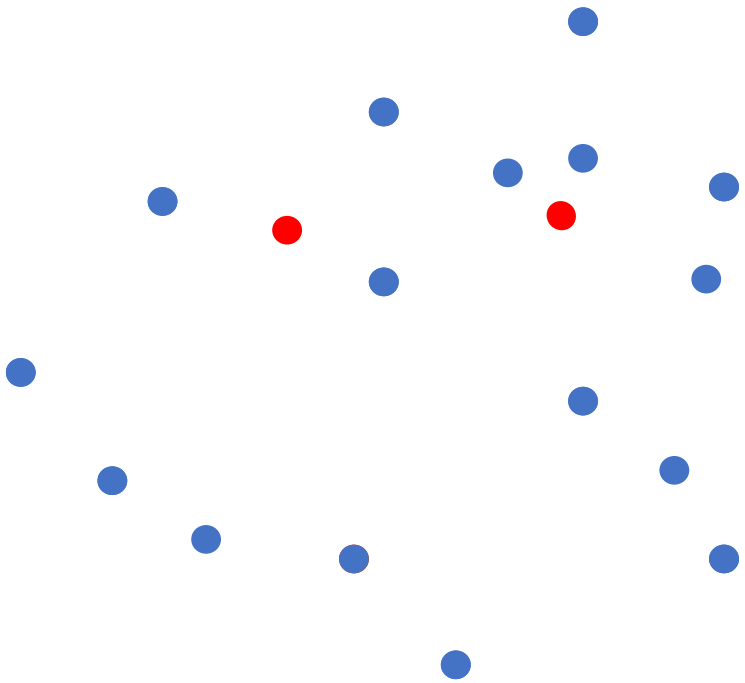
黑马程序员
www.itheima.com
传智播客旗下高端IT教育品牌

1-2 算法解析

K：初始中心点个数（计划聚类数）

K=2

means：求中心点到其他数据点距离的平均值



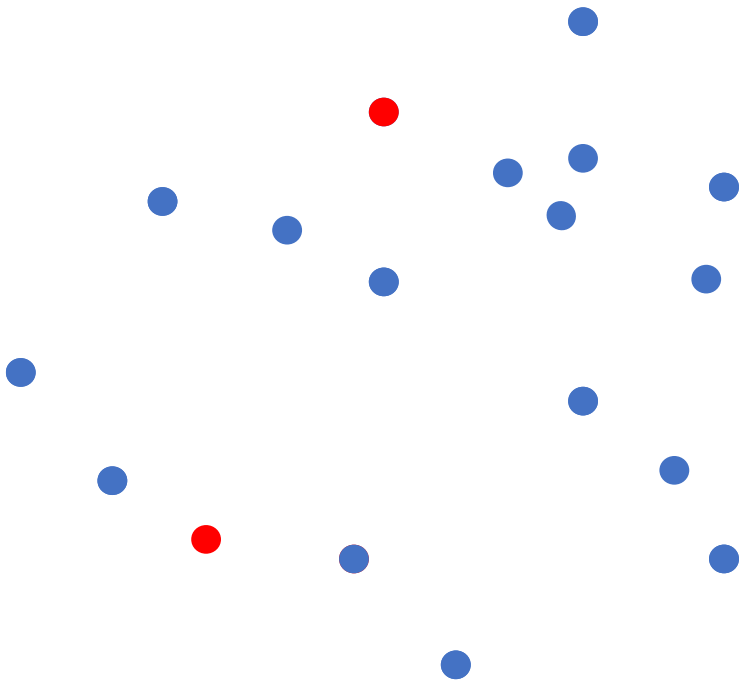
1 算法原理

1-2 算法解析

K：初始中心点个数（计划聚类数）

K=2

means：求中心点到其他数据点距离的平均值



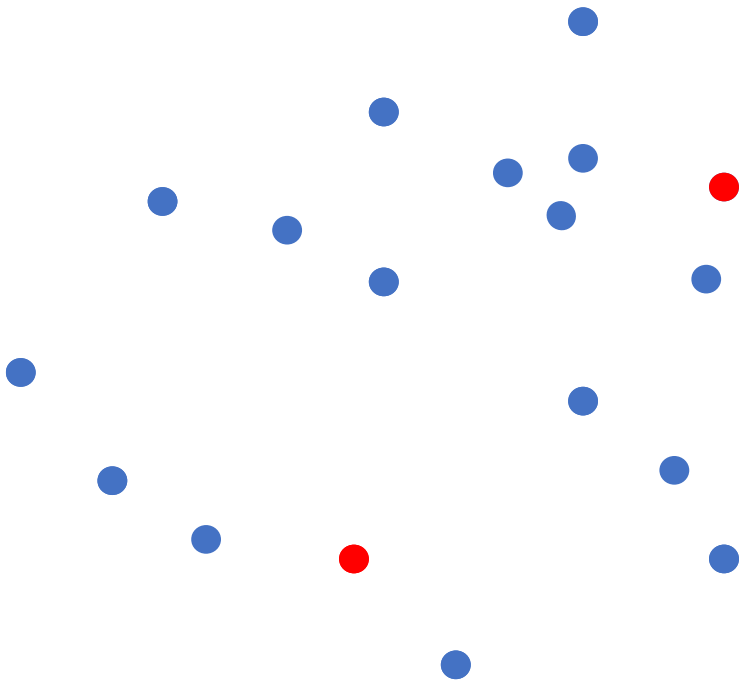
1 算法原理

1-2 算法解析

K：初始中心点个数（计划聚类数）

K=2

means：求中心点到其他数据点距离的平均值



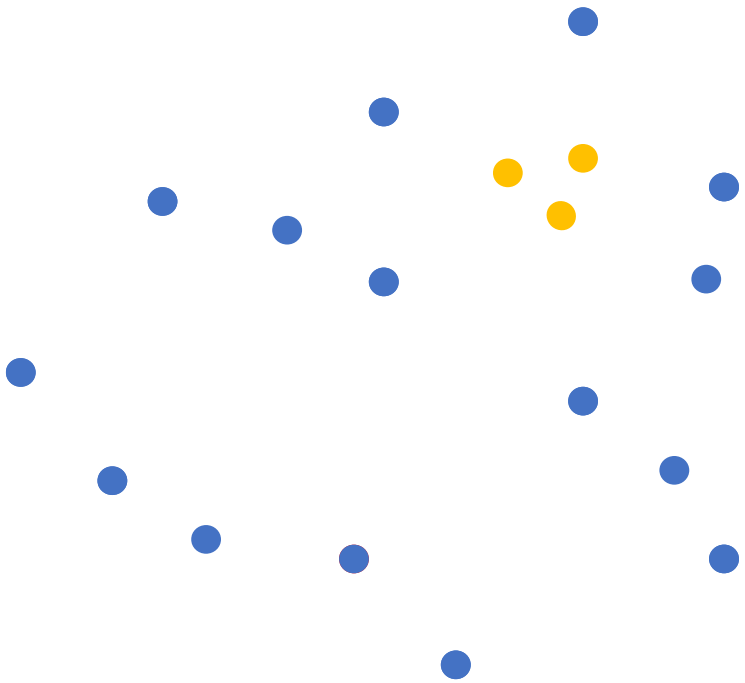
1 算法原理

1-2 算法解析

K : 初始中心点个数（计划聚类数）

K=3

means：求中心点到其他数据点距离的平均值



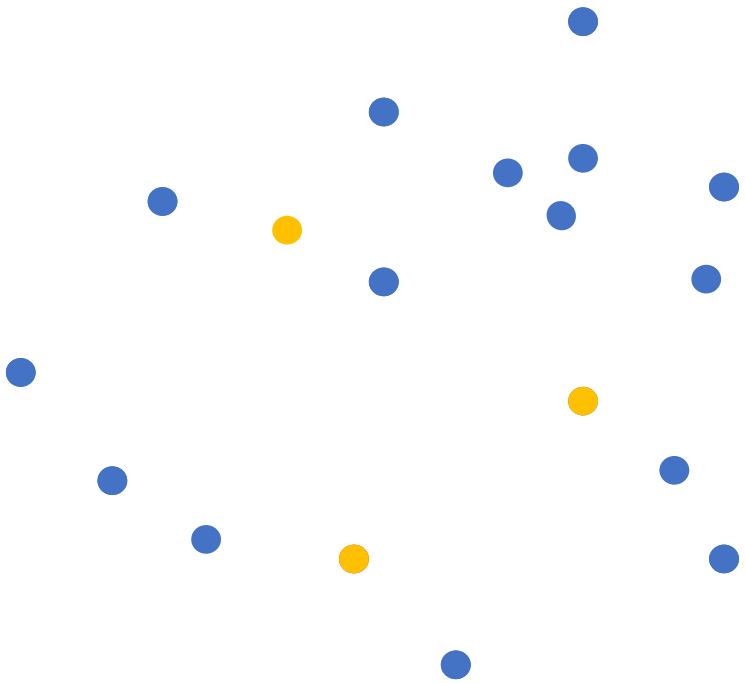
1 算法原理

1-2 算法解析

K：初始中心点个数（计划聚类数）

K=3

means：求中心点到其他数据点距离的平均值



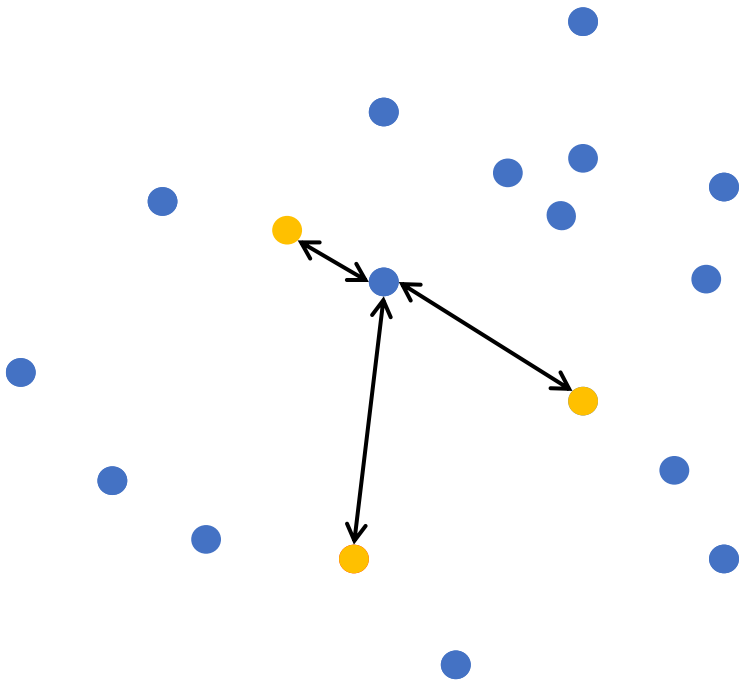
1 算法原理

1-2 算法解析

K：初始中心点个数（计划聚类数）

K=3

means：求中心点到其他数据点距离的平均值



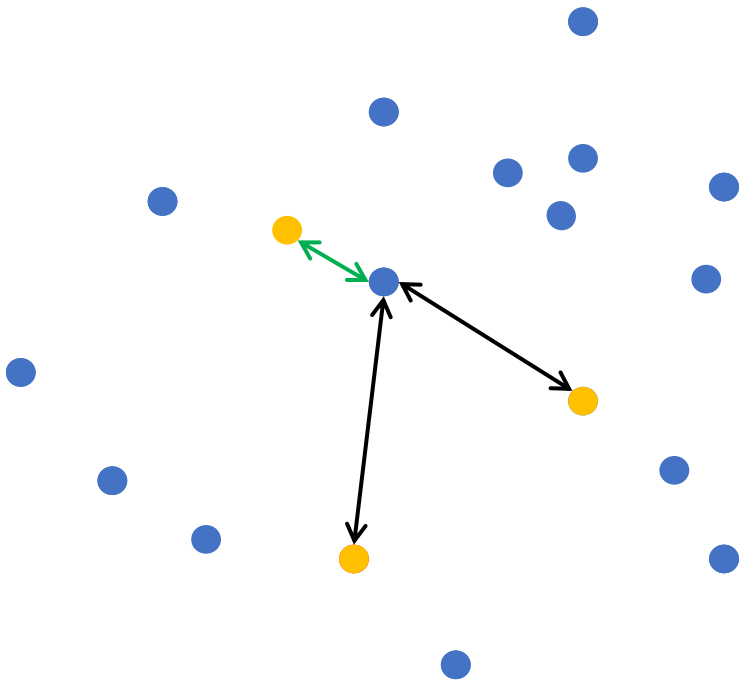
1 算法原理

1-2 算法解析

K：初始中心点个数（计划聚类数）

K=3

means：求中心点到其他数据点距离的平均值



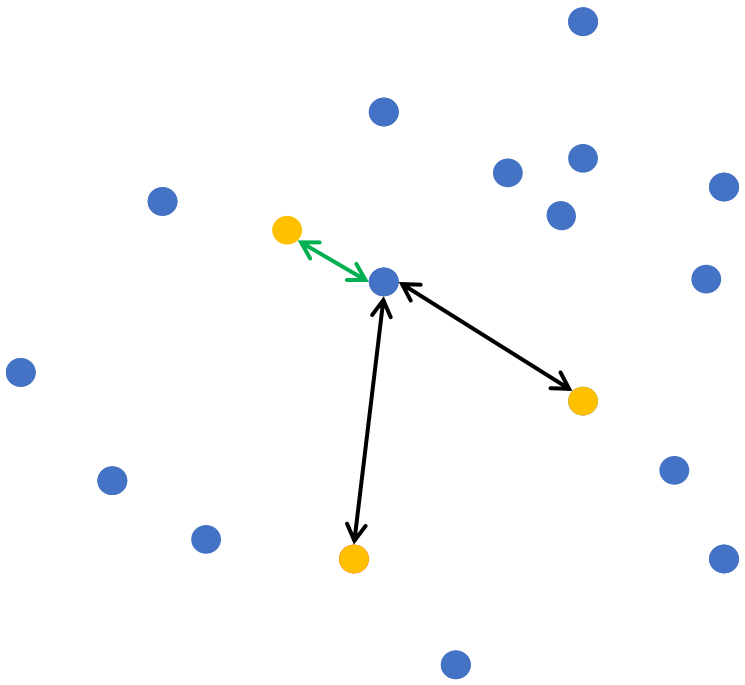
1 算法原理

1-2 算法解析

K：初始中心点个数（计划聚类数）

K=3

means：求中心点到其他数据点距离的平均值



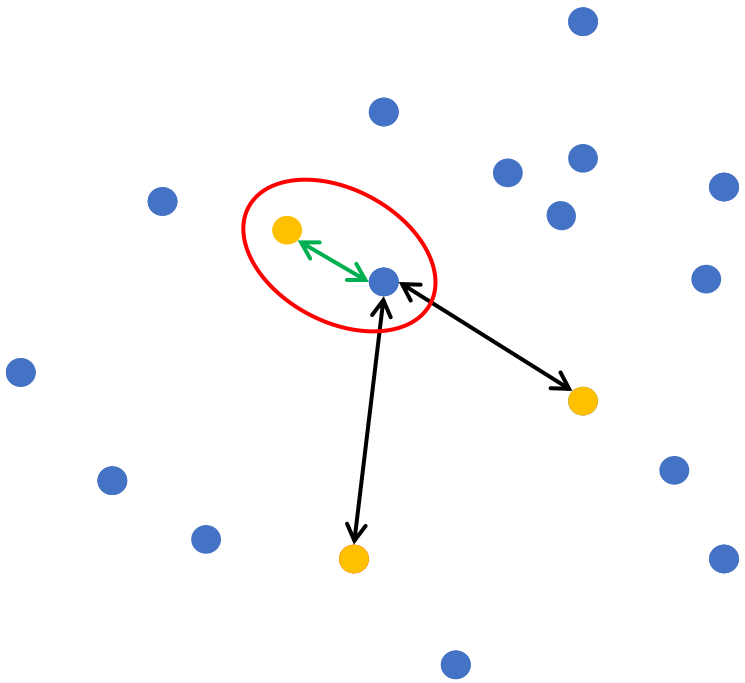
1 算法原理

1-2 算法解析

K：初始中心点个数（计划聚类数）

K=3

means：求中心点到其他数据点距离的平均值



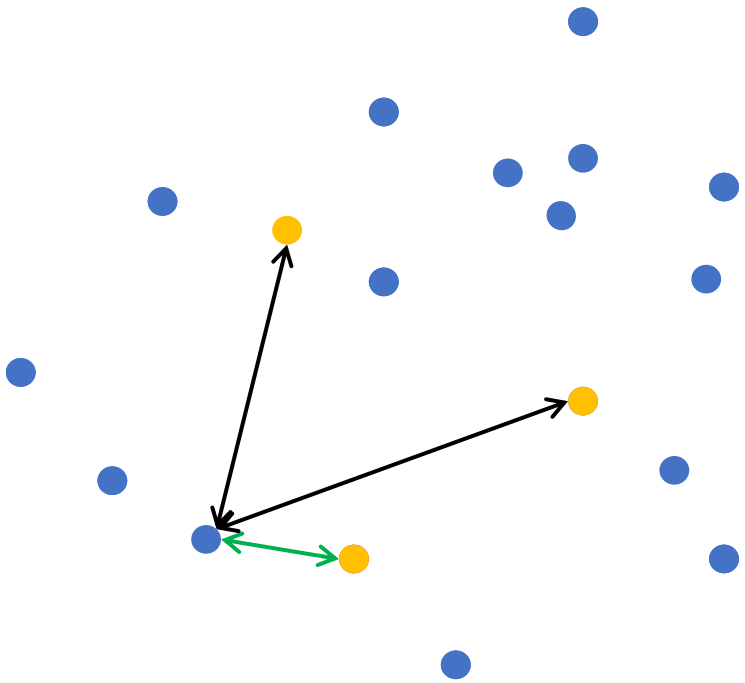
1 算法原理

1-2 算法解析

K：初始中心点个数（计划聚类数）

K=3

means：求中心点到其他数据点距离的平均值



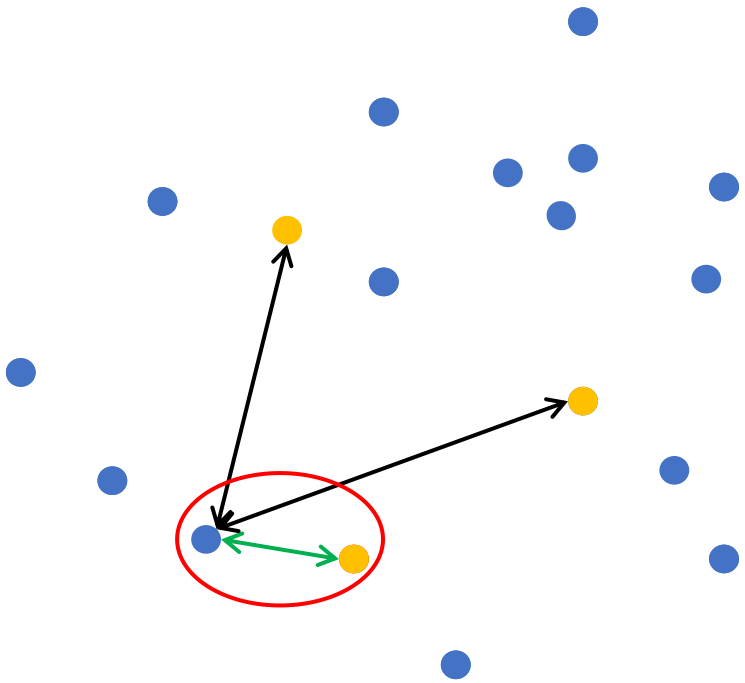
1 算法原理

1-2 算法解析

K：初始中心点个数（计划聚类数）

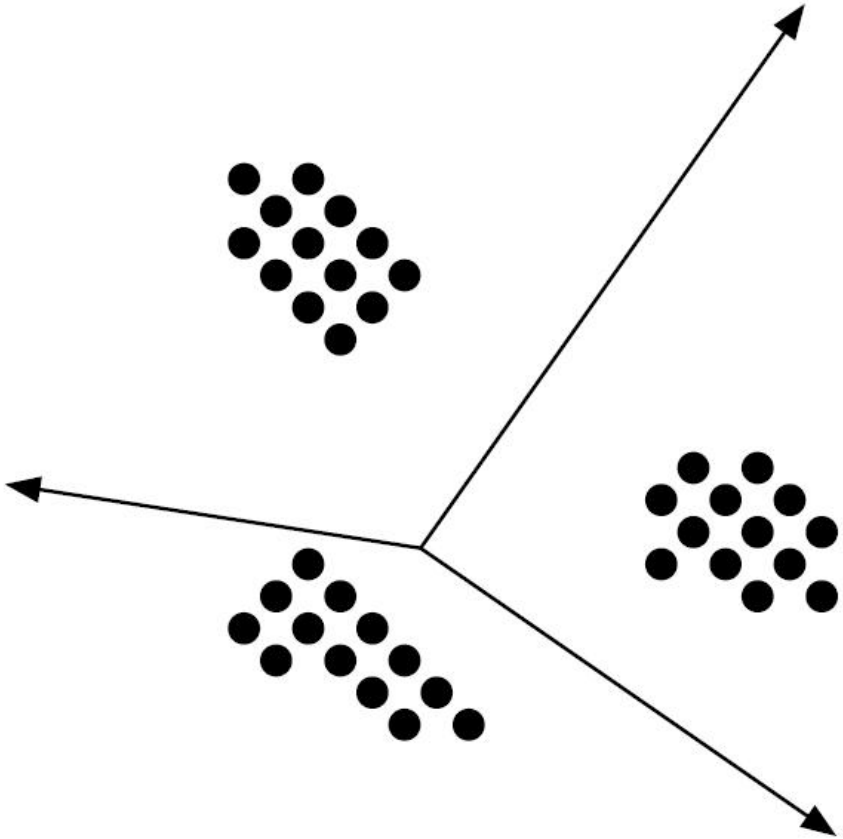
K=3

means：求中心点到其他数据点距离的平均值



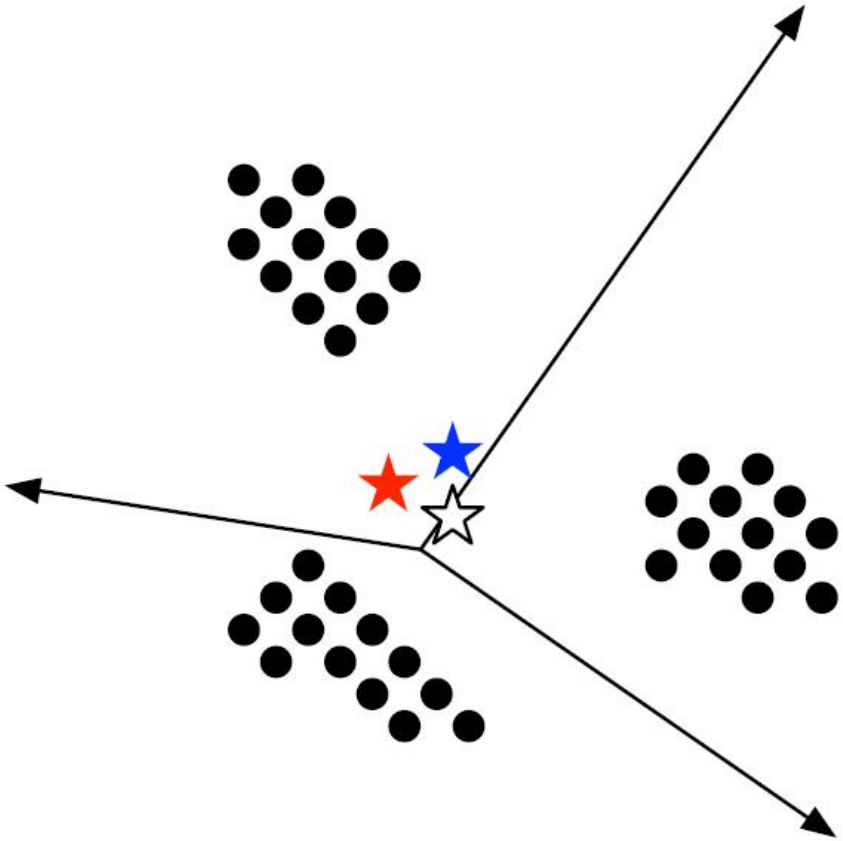
1 算法原理

1-2 算法解析



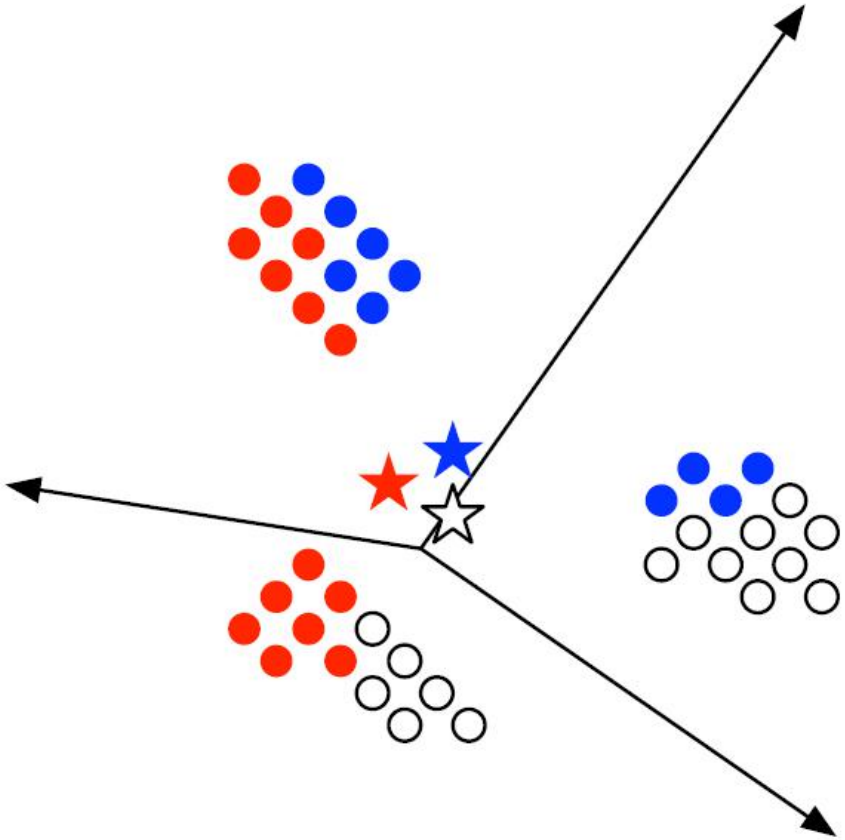
1 算法原理

1-2 算法解析



1 算法原理

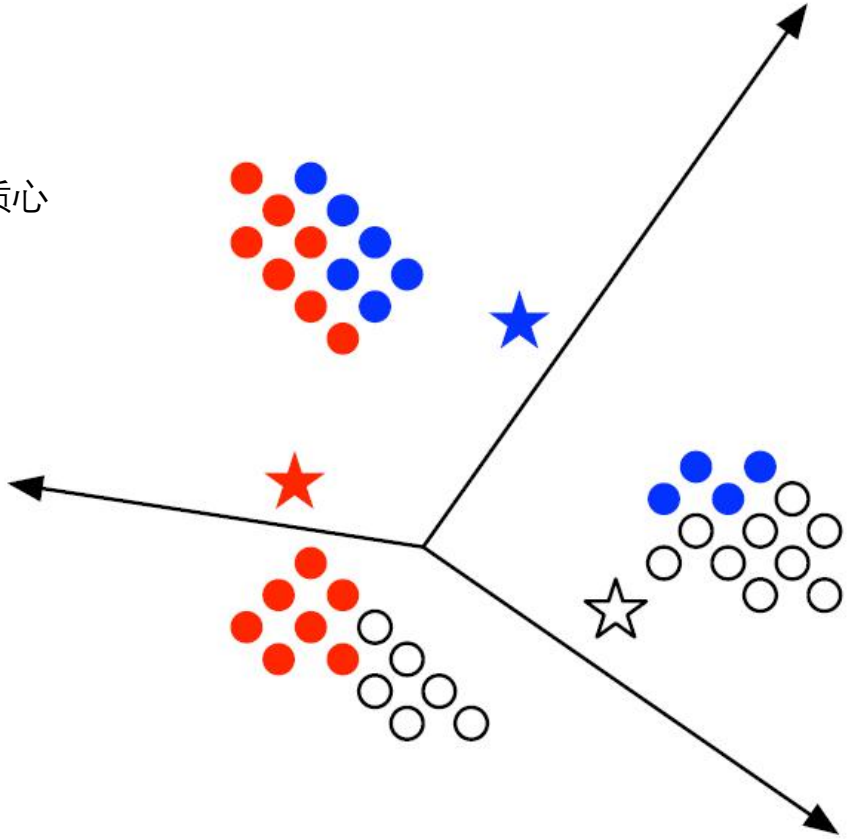
1-2 算法解析



1 算法原理

1-2 算法解析

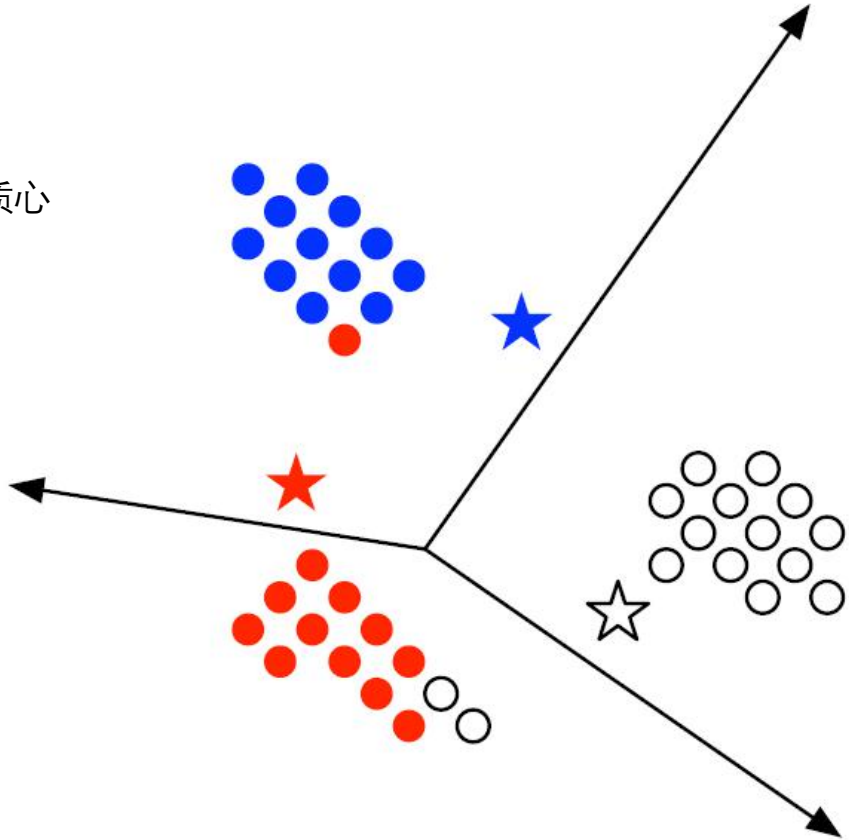
不断迭代，更新质心



1 算法原理

1-2 算法解析

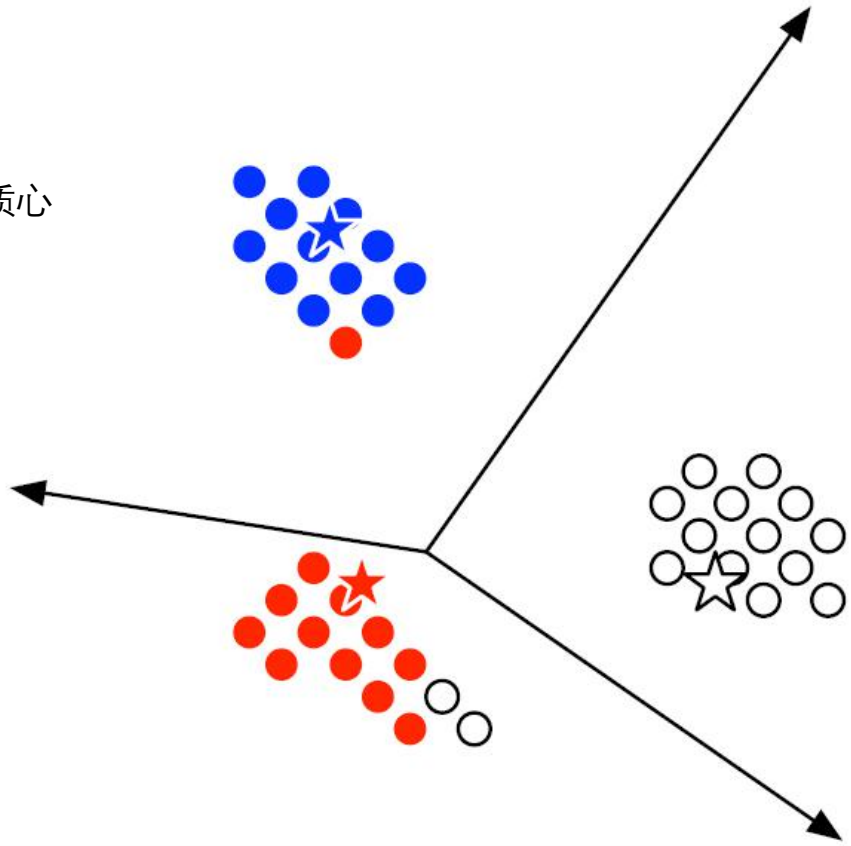
不断迭代，更新质心



1 算法原理

1-2 算法解析

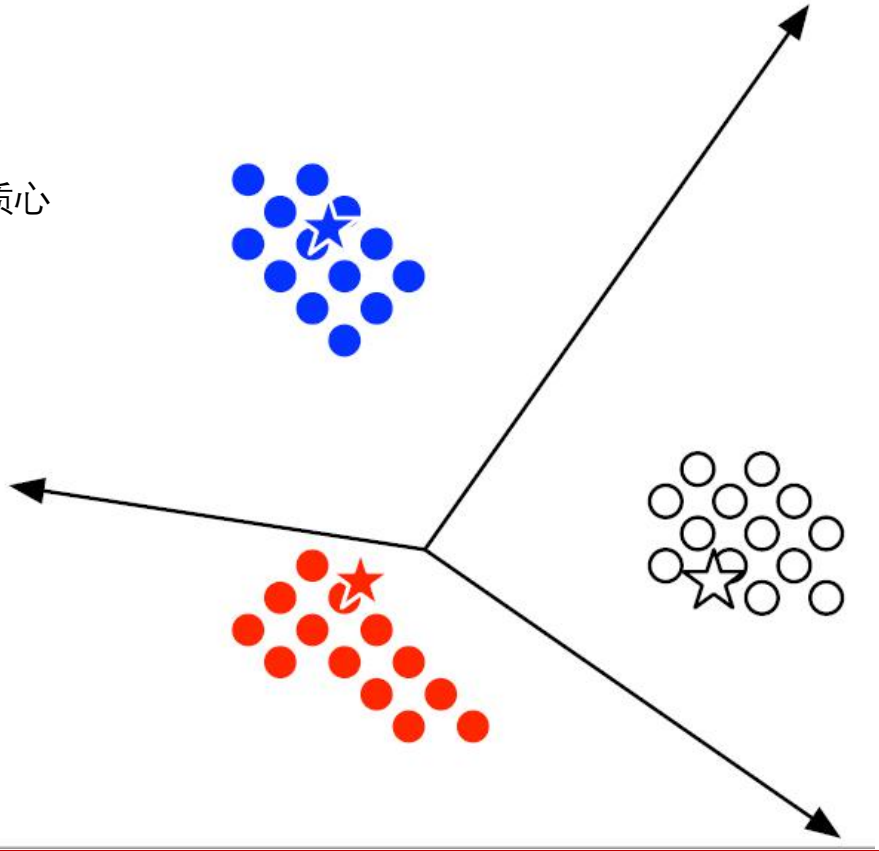
不断迭代，更新质心



1 算法原理

1-2 算法解析

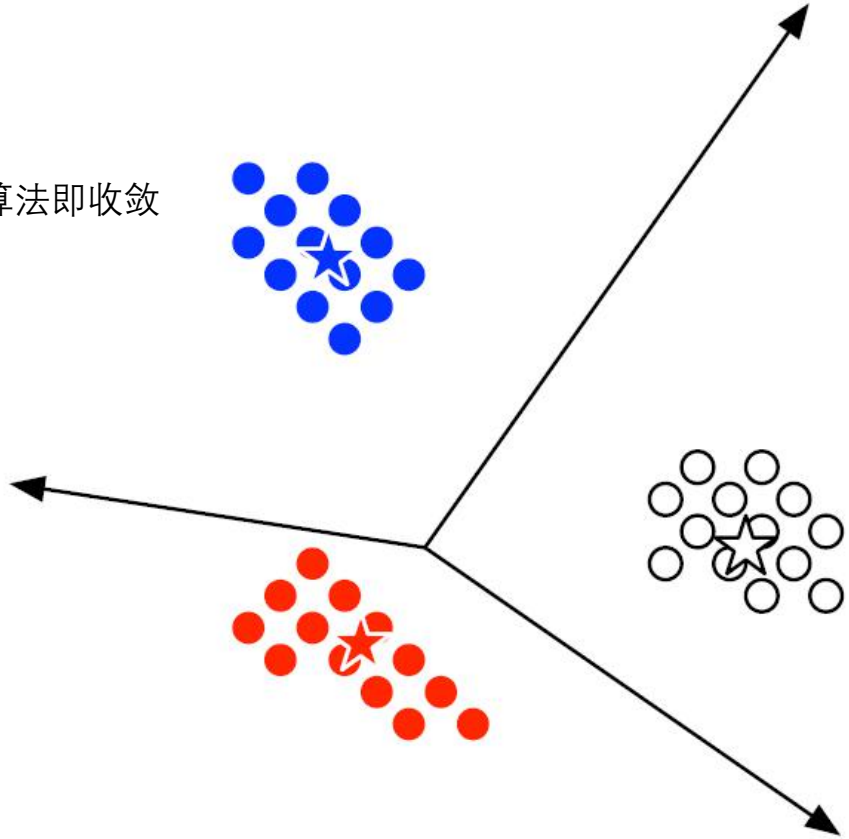
不断迭代，更新质心



1 算法原理

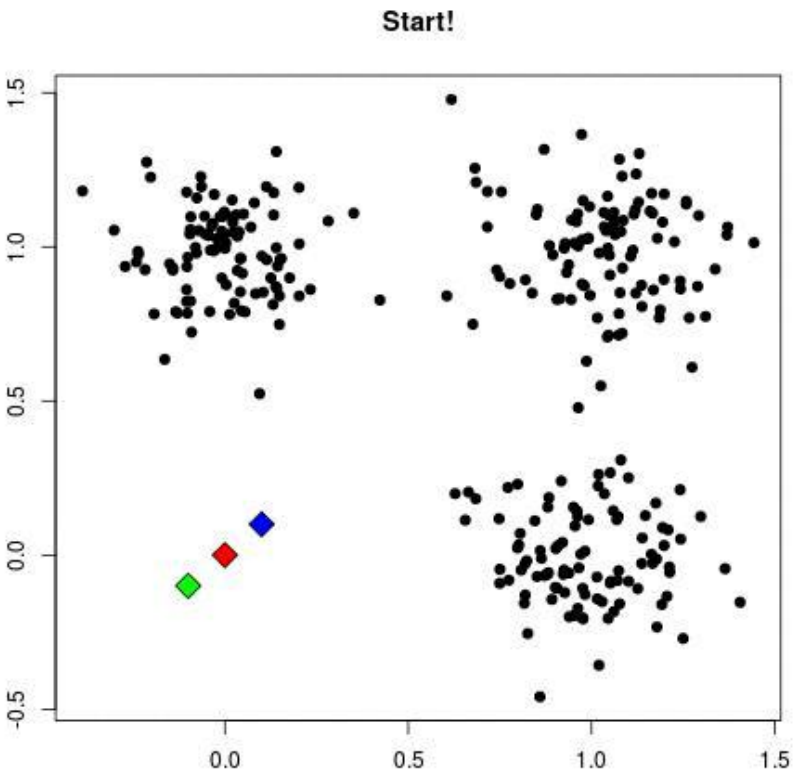
1-2 算法解析

质心不在移动，算法即收敛

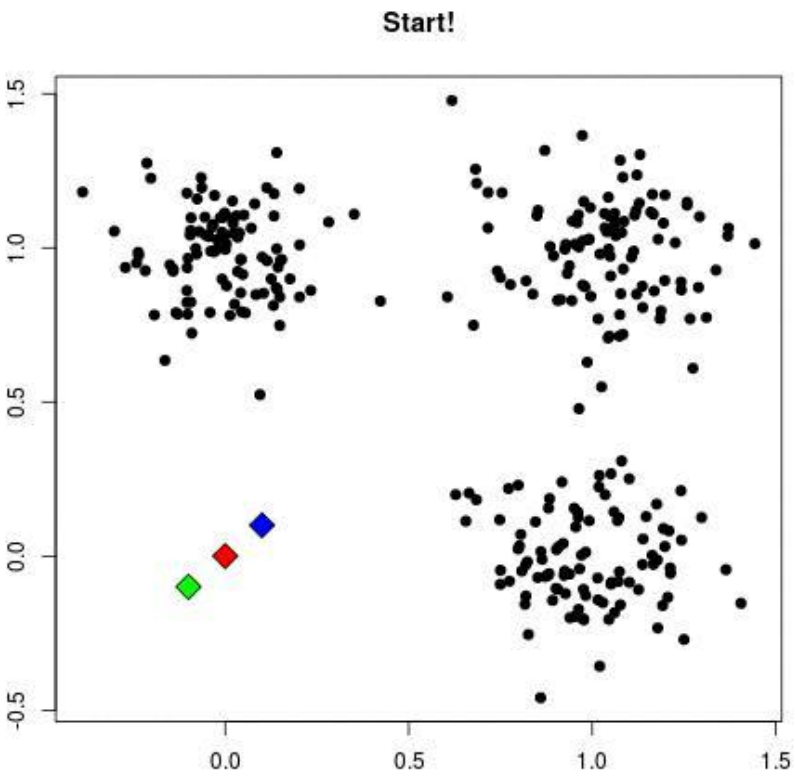


思考 请尝试着总结出k-means算法流程

思考



思考 请尝试着总结出k-means算法流程



1.选择聚类的个数k.

2.任意产生k个聚类，然后确定聚类中心，或者直接生成k个中心。

3.对每个点确定其聚类中心点。

4.再计算其聚类新中心。

5.重复以上步骤直到满足收敛要求。（通常就是确定的中心点不再改变。）

提问 将下列数据点用K-means方法进行聚类

提问

	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

1 算法原理

Euclidean

欧式距离

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Manhattan

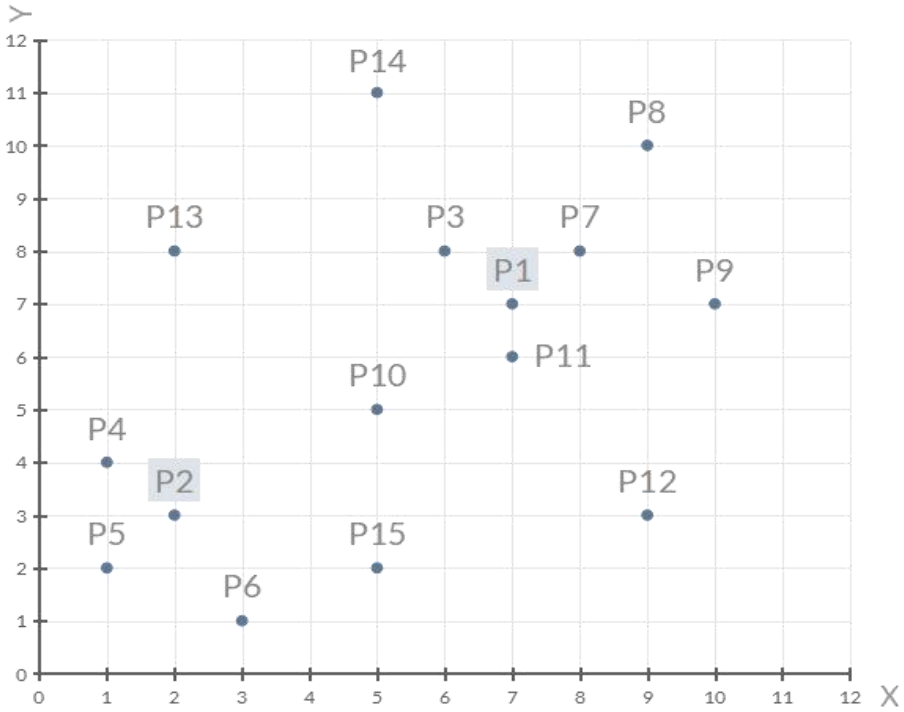
曼哈顿距离

$$\sum_{i=1}^{k}|x_i - y_i|$$

Minkowski

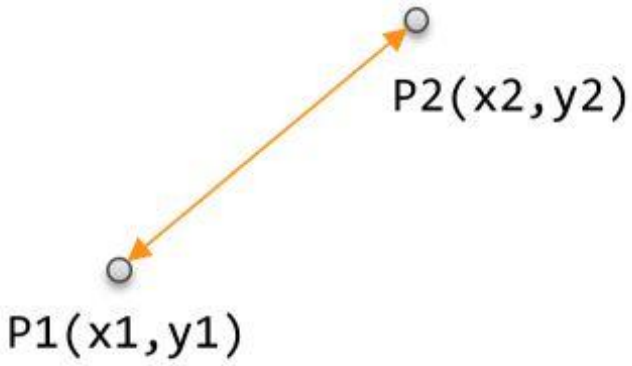
闵氏距离

$$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$$

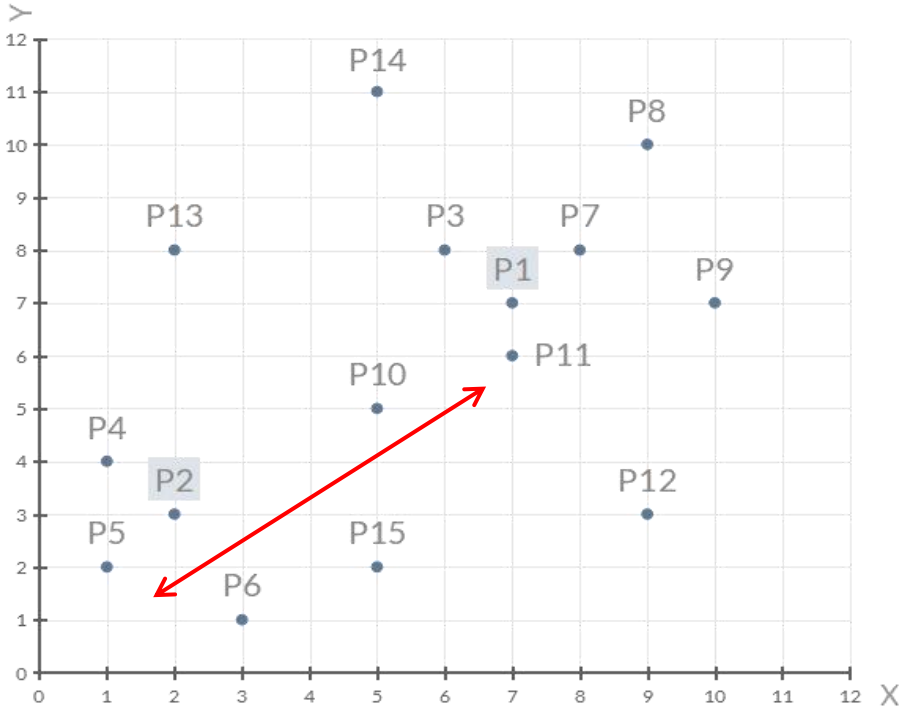


1 算法原理

1-3 算法流程



$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$



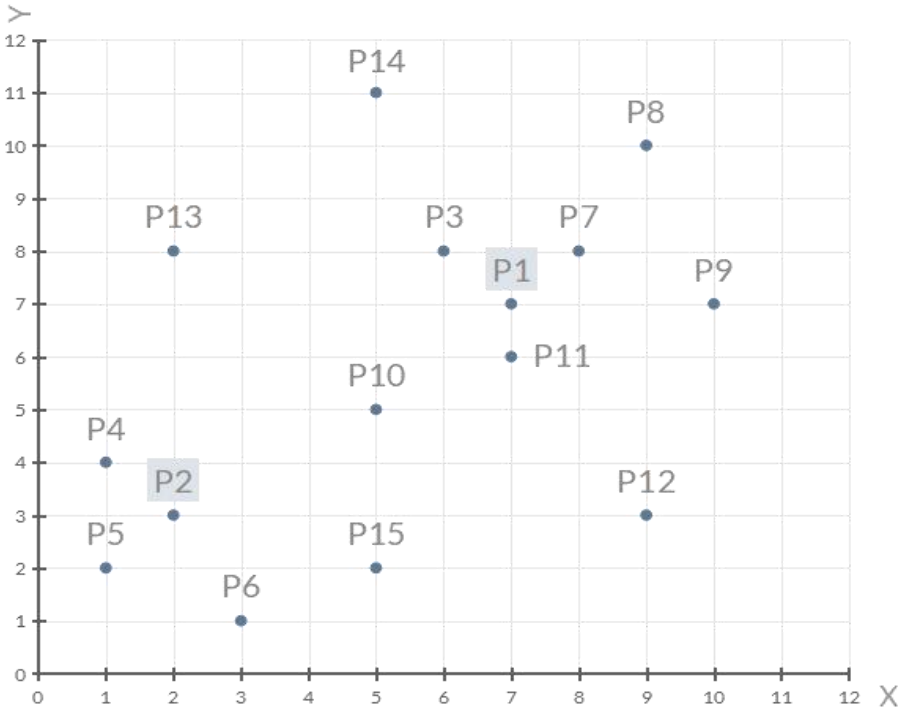
1 算法原理

1-3 算法流程

Euclidean

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

$$d = \sqrt{(x_n - x_1)^2 + (y_n - y_1)^2}$$



1 算法原理

1-3 算法流程

（1）通过距离公式将分别计算各点到P1,P2数据点距离：

	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

1 算法原理

1-3 算法流程

（2）选取距离较近的点整理进入相应队列：

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

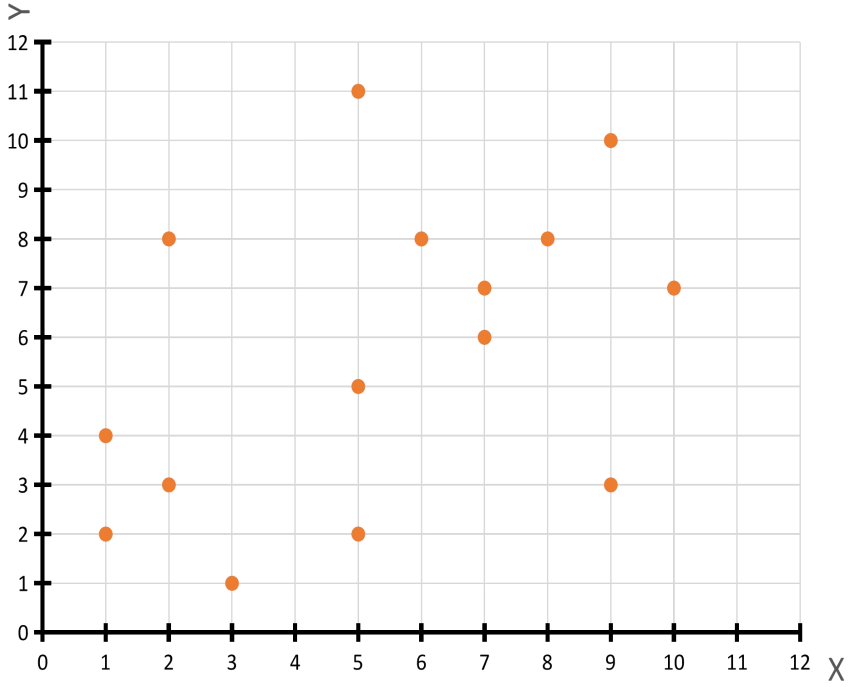
1 算法原理

1-3 算法流程

（3）计算出新一轮的队列中心：

$$P_y = \sum_{i=1}^{n} p_{iy} \Big/ n$$

$$P_x = \sum_{i=1}^{n} p_{ix} \Big/ n$$



1 算法原理

1-3 算法流程

（4）重复上述步骤，开始新一轮迭代，算距离，取最近：

	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

1 算法原理

1-3 算法流程

（5）重复上述步骤，开始新一轮迭代，算距离，取最近：

	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

1 算法原理

1-3 算法流程

（6）再次选取距离较近的点整理进入相应队列：

发现点的队列位置和上一轮并无差别：

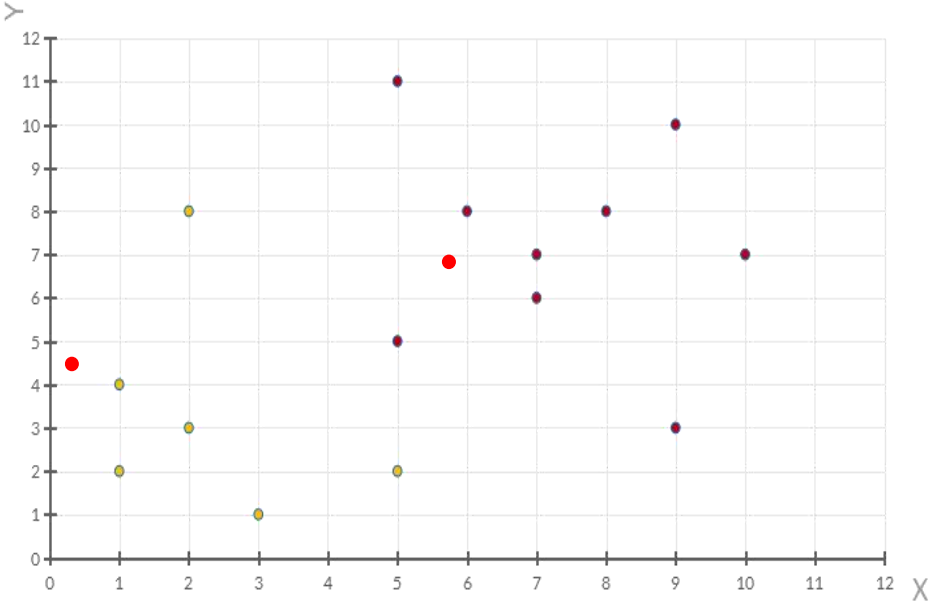
P'_1	P1	P3	P7	P8	P9	P10	P11	P12	P14
P'_2	P2	P4	P5	P6	P13	P15			

1 算法原理

1-3 算法流程

（6）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。

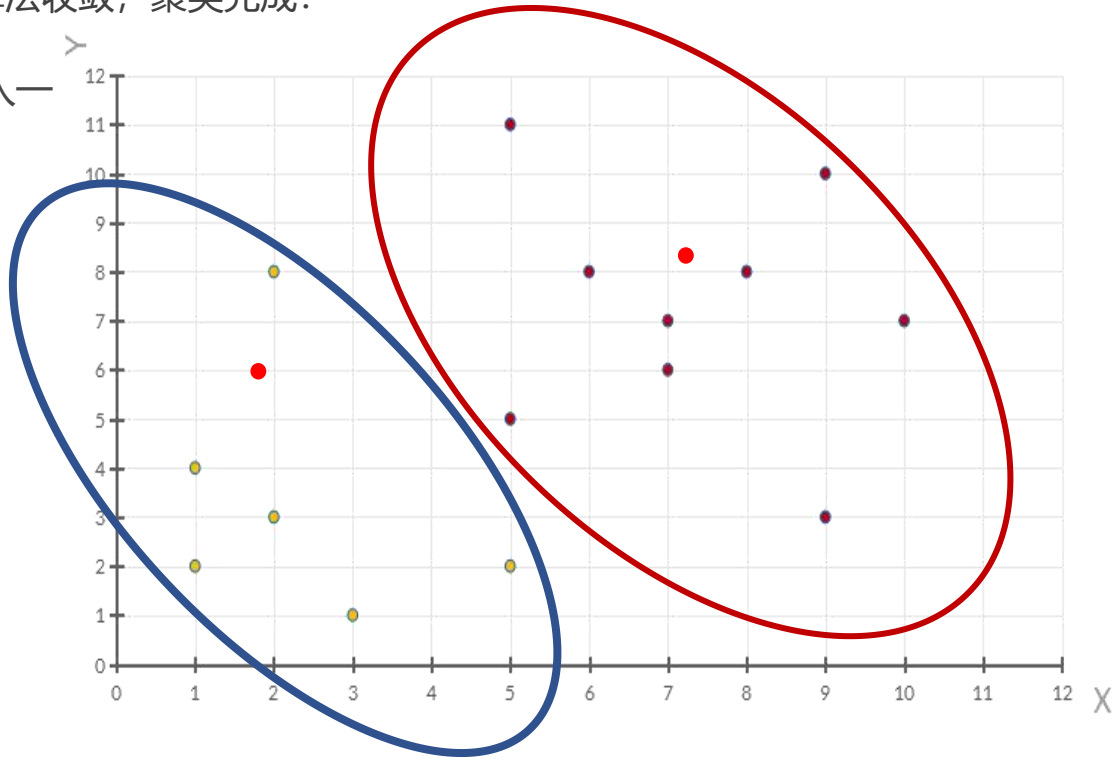


1 算法原理

1-3 算法流程

（6）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。



提问 请按之前学习的方法对下列数据集运用Kmeans进行聚类

黑马程序员
www.itheima.com
传智播客旗下高端IT教育品牌

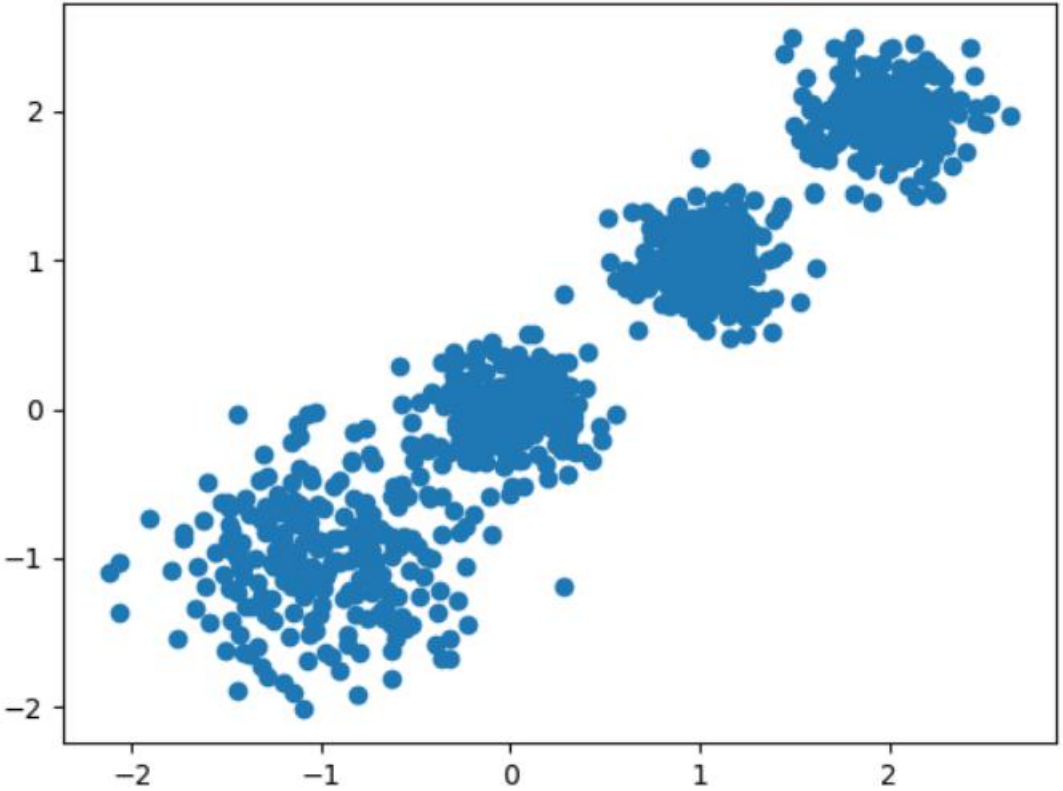
提问

	X 值	Y 值
P1	0	0
P2	1	2
P3	3	1
P4	8	8
P5	9	10
P6	10	7

1 算法原理

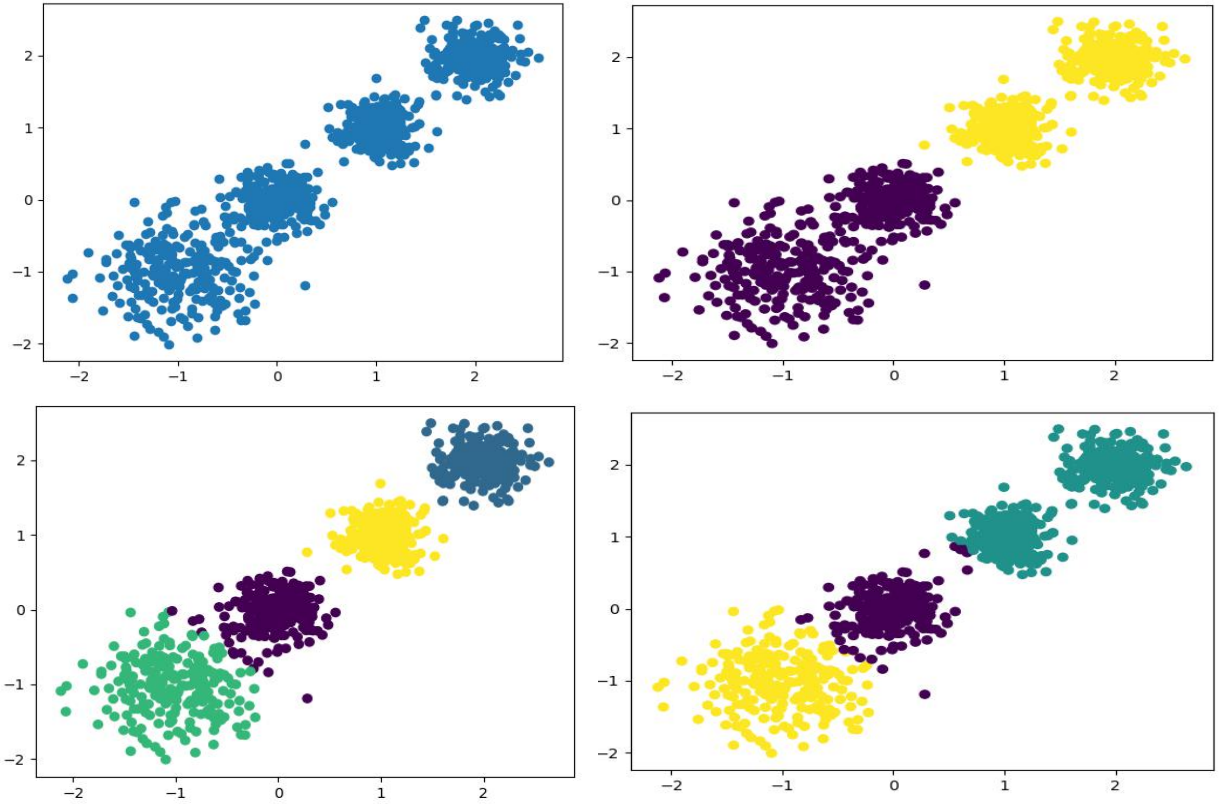
案例1：不同数据集的k-means聚类 -- 介绍

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：



1 算法原理

案例1：不同数据集的k-means聚类 -- 分析



1 算法原理

案例1：不同数据集的k-means聚类 -- 分析

导入工具库

numpy,matplotlib...

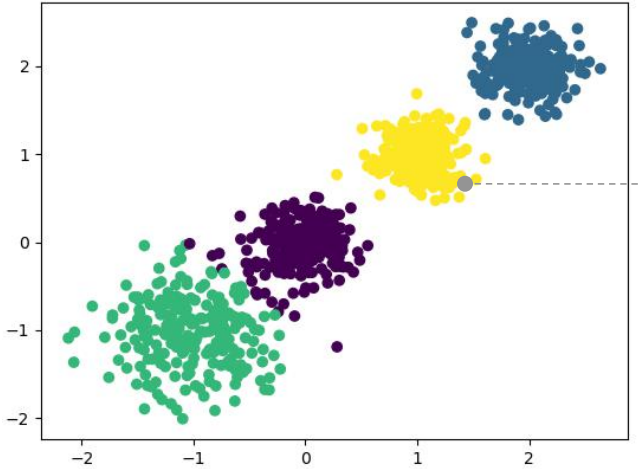
1

创建数据集并显示

2

3

应用KMeans



1 算法原理

案例1：不同数据集的k-means聚类 -- 分析

导入工具库

numpy,matplotlib...

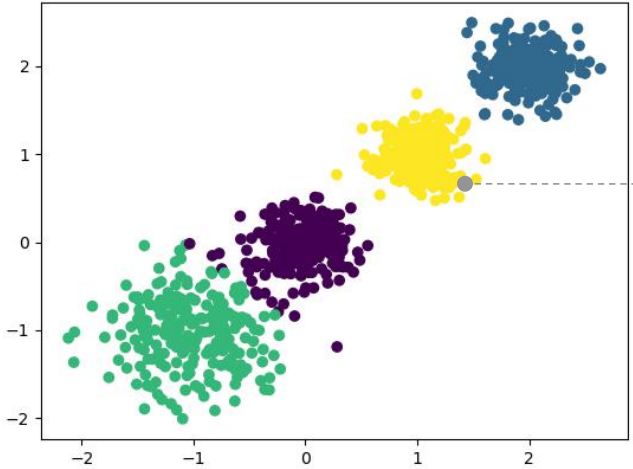
1

创建数据集并显示

2

3

应用KMeans



1 算法原理

案例1：不同数据集的k-means聚类 -- 分析

SKlearn的K-means API参数简介：

```
from sklearn.cluster import k_means
k_means()
```

```
sklearn.cluster.k_means_.k_means k_means(X, n_clusters,
sample_weight=None, init='k-means++',
    precompute_distances='auto', n_init=10, max_iter=300,
    verbose=False, tol=1e-4, random_state=None, copy_x=True,
    n_jobs=None, algorithm="auto", return_n_iter=False)
```

KMeans中的默认参数

案例1：不同数据集的k-means聚类 -- 分析

参数	解释
n_clusters	整型，缺省值=8 生成的聚类数，即产生的质心（centroids）数。
max_iter	整型，缺省值=300 执行一次k-means算法所进行的最大迭代数。
n_init	整型，缺省值=10 用不同的质心初始化值运行算法的次数，最终解是再inertia意义下选出的最有结果。
init	有三个可选值：‘k-means++’ ‘random’或这传递一个ndarray向量。 此参数指定初始化方法，默认值为‘k-means++’。 ‘random’随机从训练数据中选取初始质心。 如果传递的是一个ndarray，则应该形如（n_clusters, n_features）并给出初始质心。

案例1：不同数据集的k-means聚类 -- 分析

参数	解释
n_clusters	整型，缺省值=8 生成的聚类数，即产生的质心（centroids）数。
max_iter	整型，缺省值=300 执行一次k-means算法所进行的最大迭代数。
n_init	整型，缺省值=10 用不同的质心初始化值运行算法的次数，最终解是再inertia意义下选出的最有结果。
init	有三个可选值：‘k-means++’ ‘random’或这传递一个ndarray向量。 此参数指定初始化方法，默认值为‘k-means++’。 ‘random’随机从训练数据中选取初始质心。 如果传递的是一个ndarray，则应该形如（n_clusters, n_features）并给出初始质心。

参数1

案例1：不同数据集的k-means聚类 -- 分析

属性	解释
cluster_centers_	向量[n_clusters,n_features] Coordinates of cluster centers(每个簇中心的坐标)
Labels_	每个点的分类
inertia_	Float型，每个点到其簇的质心的距离之和

参数2

1 算法原理

案例1：不同数据集的k-means聚类 -- 实现

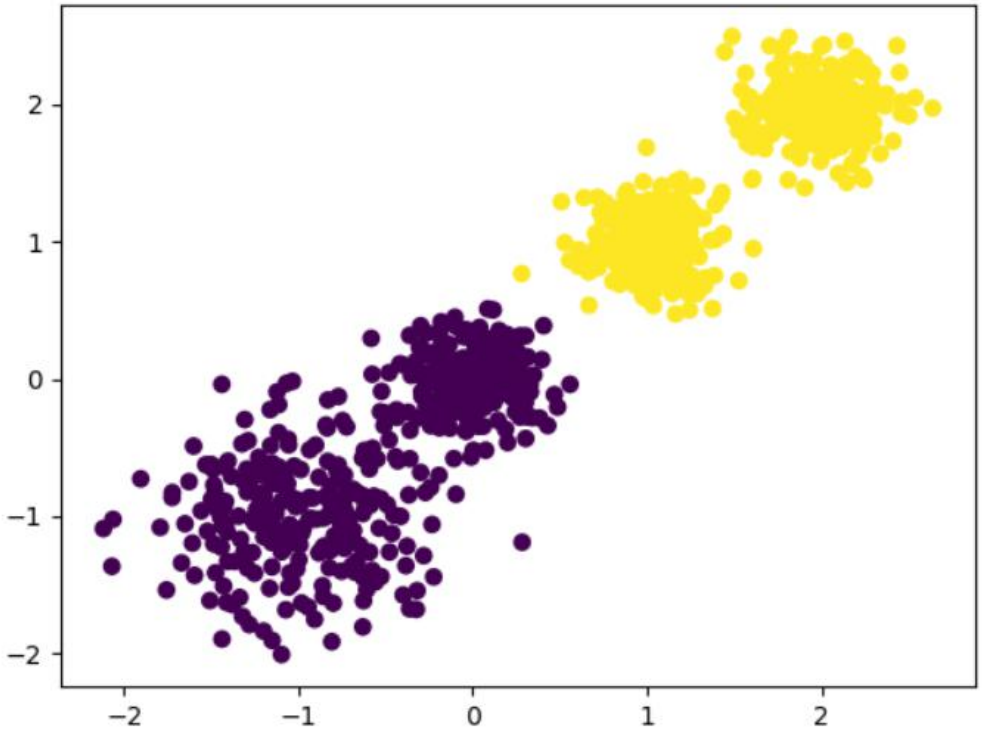
（1）创建数据集：

```
X, y = make_blobs(n_samples=1000, n_features=2,
centers=[[-1,-1], [0,0], [1,1], [2,2]],
cluster_std=[0.4, 0.2, 0.2, 0.2],
random_state =9)
plt.scatter(X[:, 0], X[:, 1], marker='o')
plt.show()
```

1 算法原理

案例1：不同数据集的k-means聚类 -- 实现

（2） K=2的聚类效果图：



案例1：不同数据集的k-means聚类 -- 实现

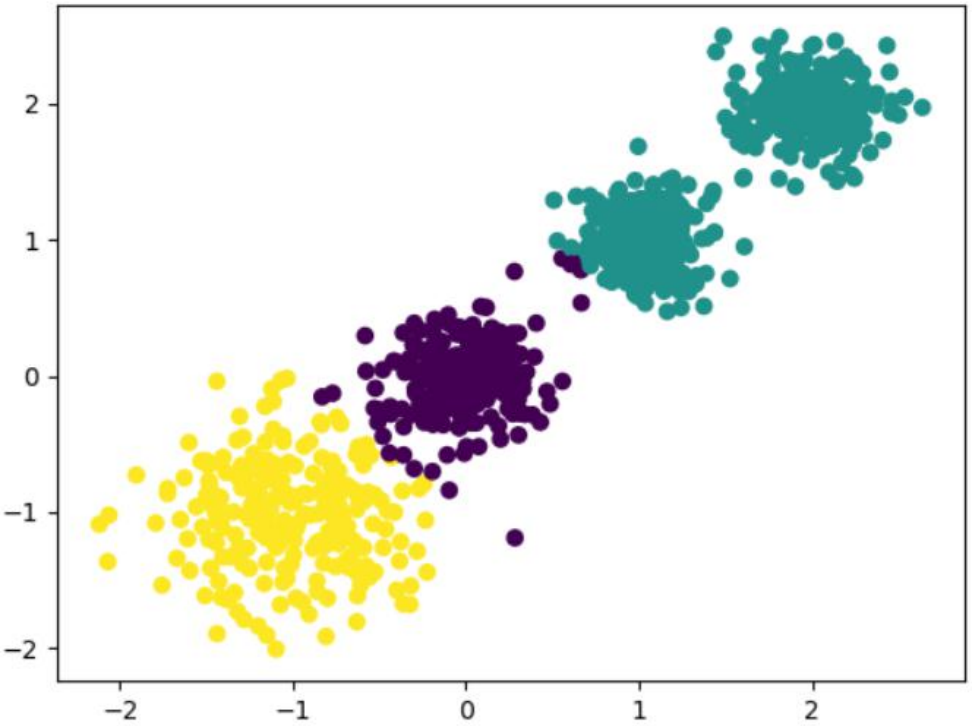
（3） K=2的聚类实现代码：

```
from sklearn.cluster import KMeans
y_pred = KMeans(n_clusters=2, random_state=9).fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=y_pred)
plt.show()
```

1 算法原理

案例1：不同数据集的k-means聚类 -- 实现

（3）K=3的聚类效果图：



1 算法原理

案例1：不同数据集的k-means聚类 -- 实现

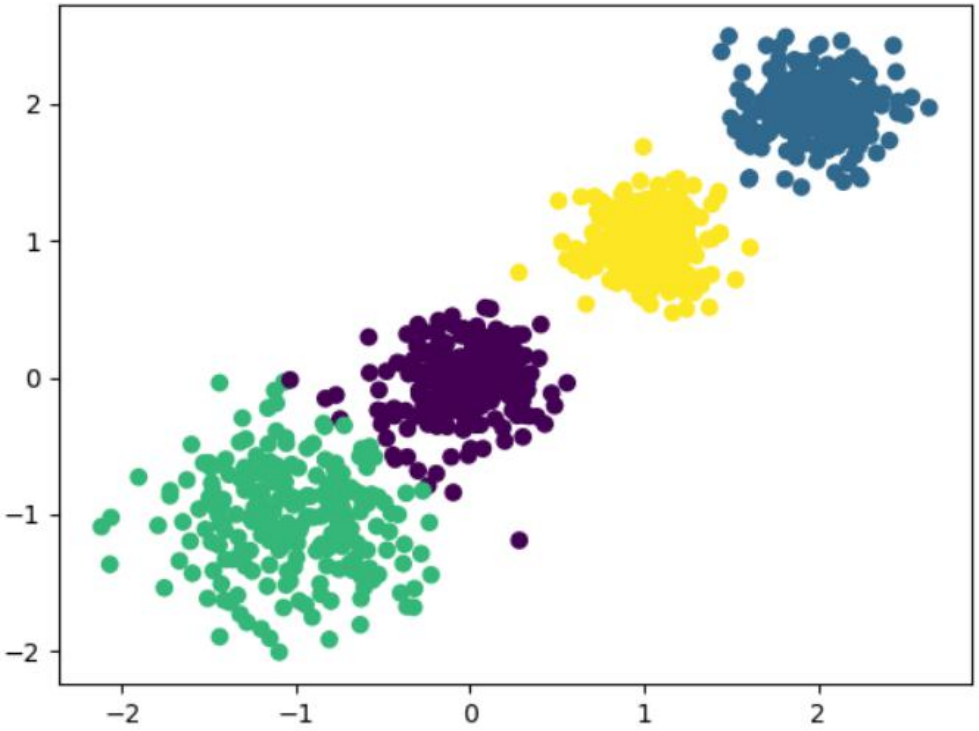
（3）K=3的聚类实现代码：

```
from sklearn.cluster import KMeans
y_pred = KMeans(n_clusters=3, random_state=9).fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=y_pred)
plt.show()
```

1 算法原理

案例1：不同数据集的k-means聚类 -- 实现

（4） K=4的聚类效果图：



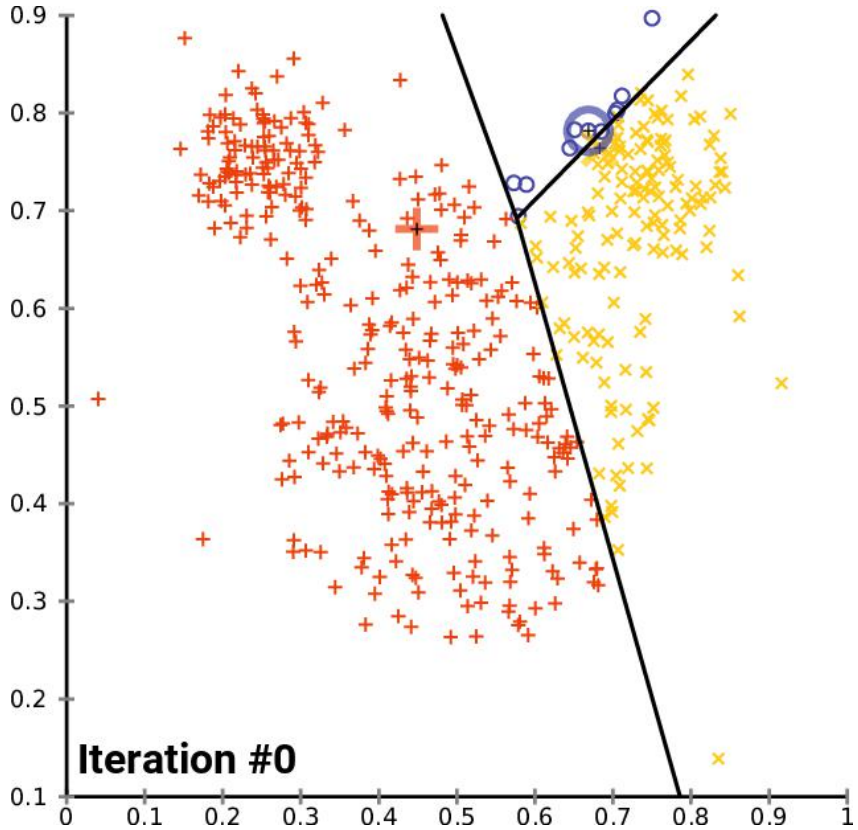
1 算法原理

案例1：不同数据集的k-means聚类 -- 实现

（4） K=4的聚类实现代码：

```
from sklearn.cluster import KMeans
y_pred = KMeans(n_clusters=4, random_state=9).fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=y_pred)
plt.show()
```

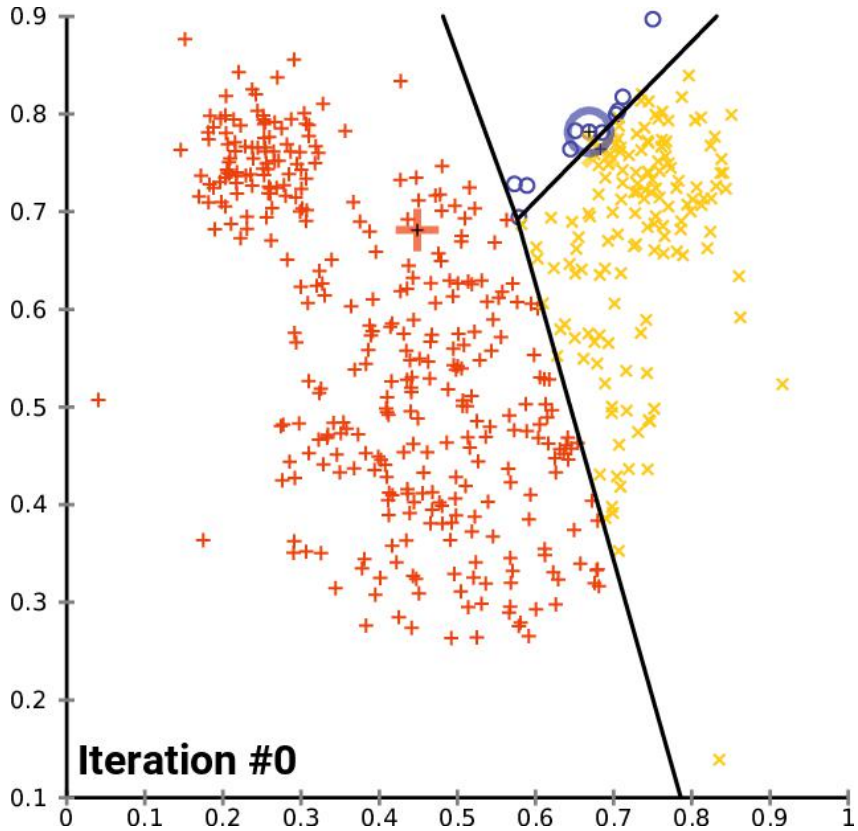
总结



K-means:

事先确定常数K，常数K意味着最终的聚类类别数，首先随机选定初始点为质心，并通过计算每一个样本与质心之间的相似度(这里为欧式距离)，将样本点归到最相似的类中，接着，重新计算每个类的质心(即为类中心)，重复这样的过程，直到质心不再改变，最终就确定了每个样本所属的类别以及每个类的质心。由于每次都要计算所有的样本与每一个质心之间的相似度，故在大规模的数据集上，K-Means算法的收敛速度比较慢。

总结



K-means：

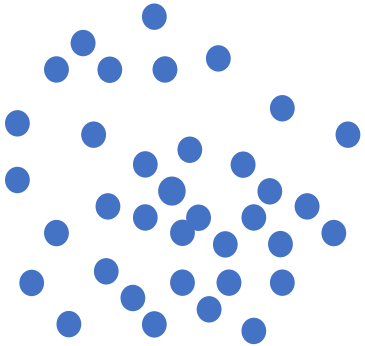
事先**确定常数K**，常数K意味着最终的聚类类别数，首先随机**选定初始点为质心**，并通过计算每一个样本与质心之间的相似度(这里为欧式距离)，将样本点归到最相似的类中，接着，**重新计算**每个类的质心(即为类中心)，重复这样的过程，直到**质心不再改变**，最终就确定了每个样本所属的类别以及每个类的质心。由于每次都要计算所有的样本与每一个质心之间的相似度，故在大规模的数据集上，K-Means算法的收敛速度比较慢。

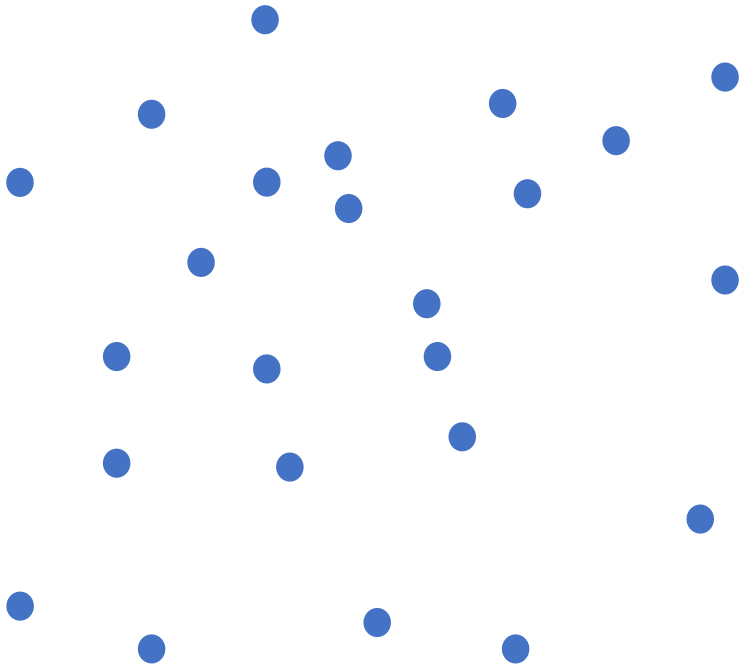
黑马程序员
www.itheima.com
传智播客旗下高端IT教育品牌

目录 Contents

2 算法效果衡量标准

2-1 SSE：误差平方和

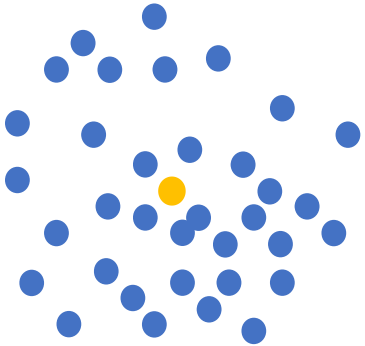


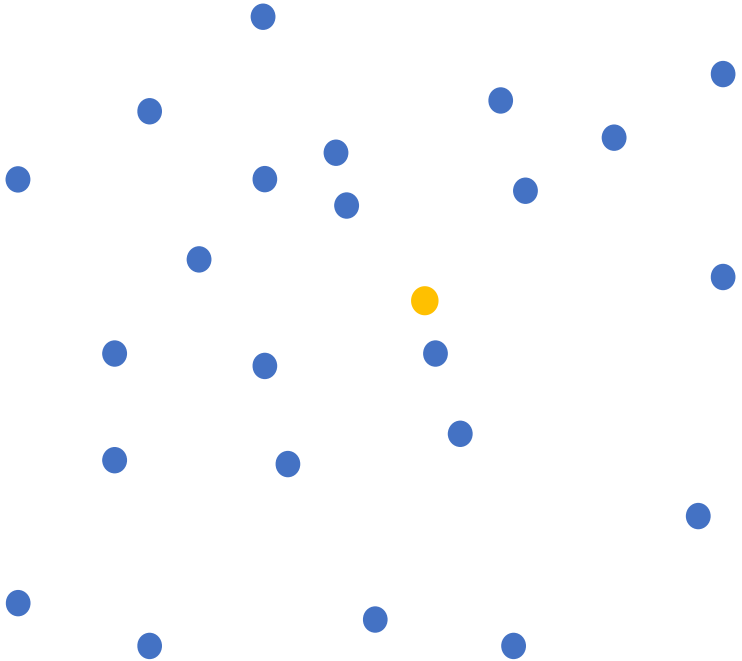


$$SSE = \sum_{i=1}^{k} \sum_{p \in C_i} \left| p - m_i \right|^2$$

2 算法效果衡量标准

2-1 SSE：误差平方和

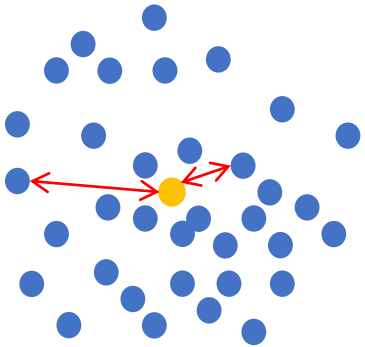


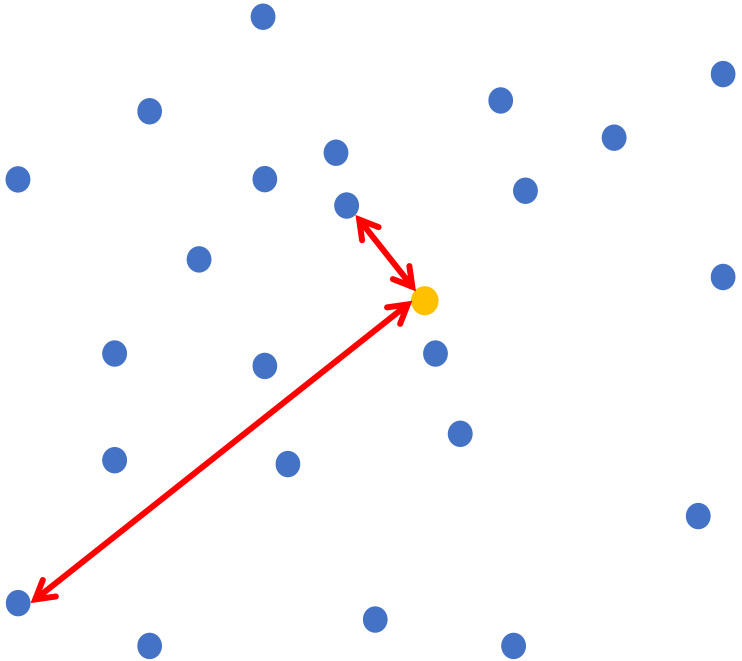


$$SSE = \sum_{i=1}^{k} \sum_{p \in C_i} \left| p - m_i \right|^2$$

2 算法效果衡量标准

2-1 SSE：误差平方和

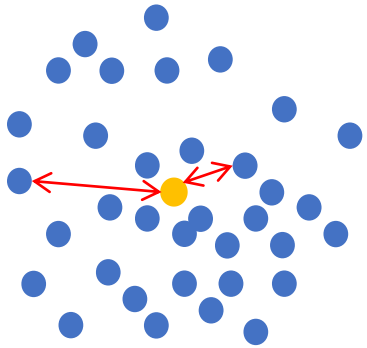


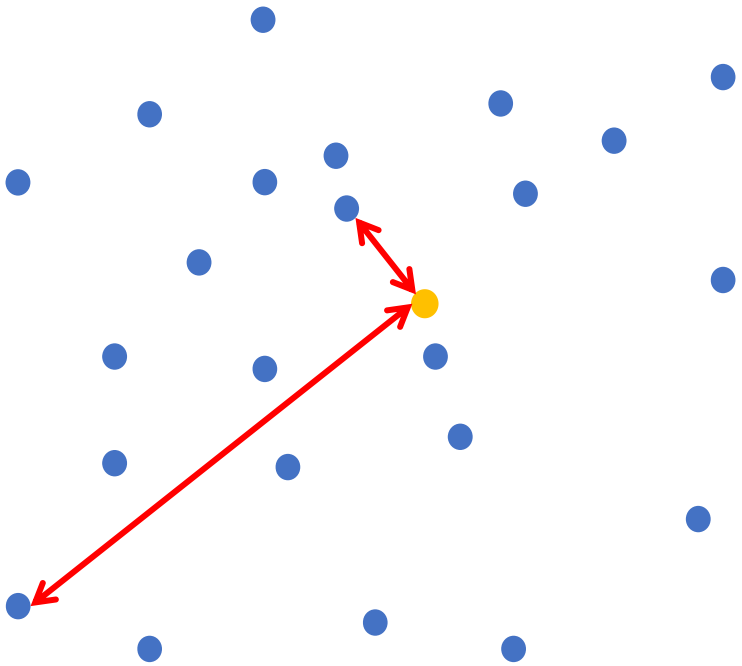


$$SSE = \sum_{i=1}^{k} \sum_{p \in C_i} \left| p - m_i \right|^2$$

2 算法效果衡量标准

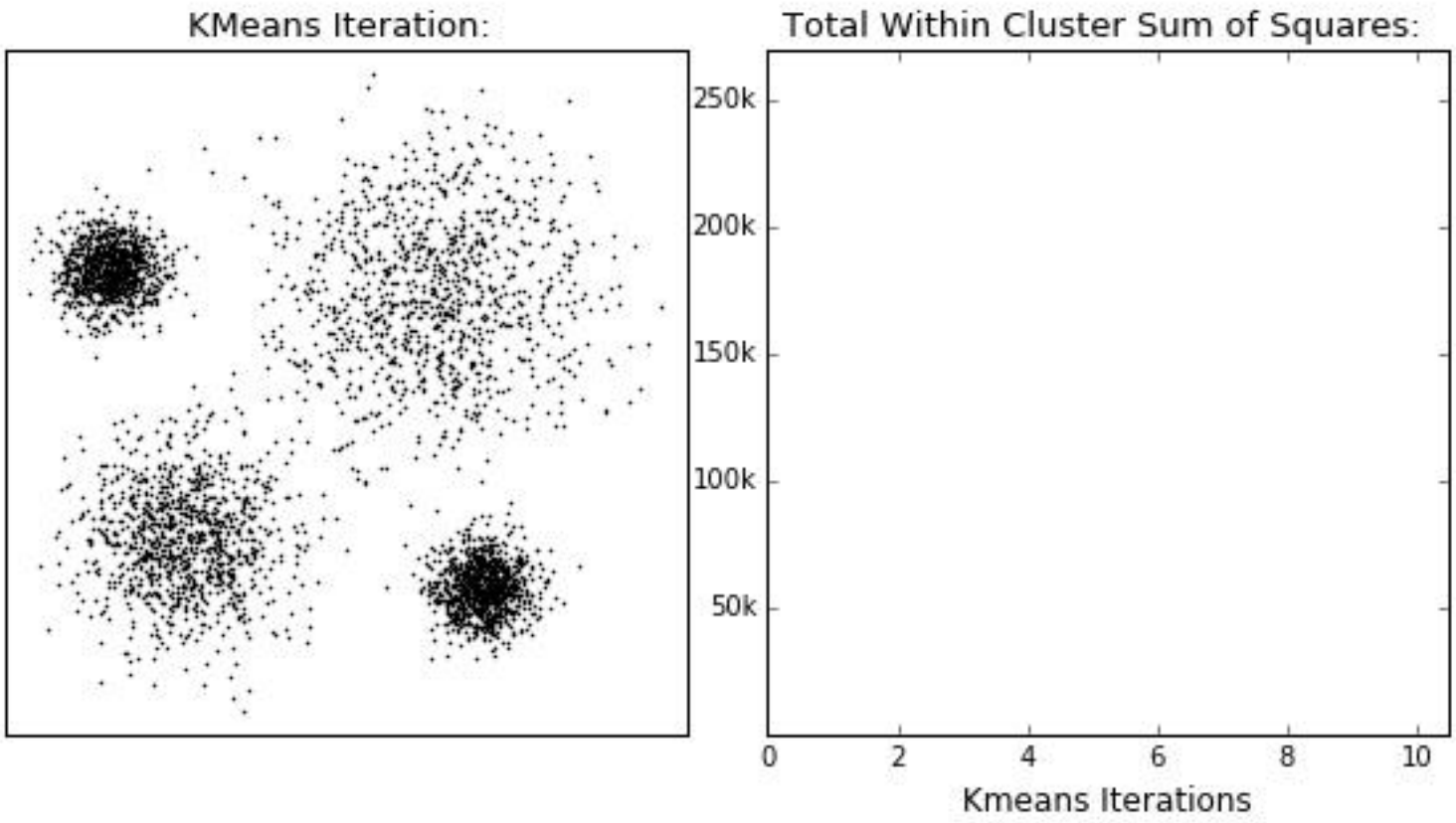
2-1 SSE：误差平方和



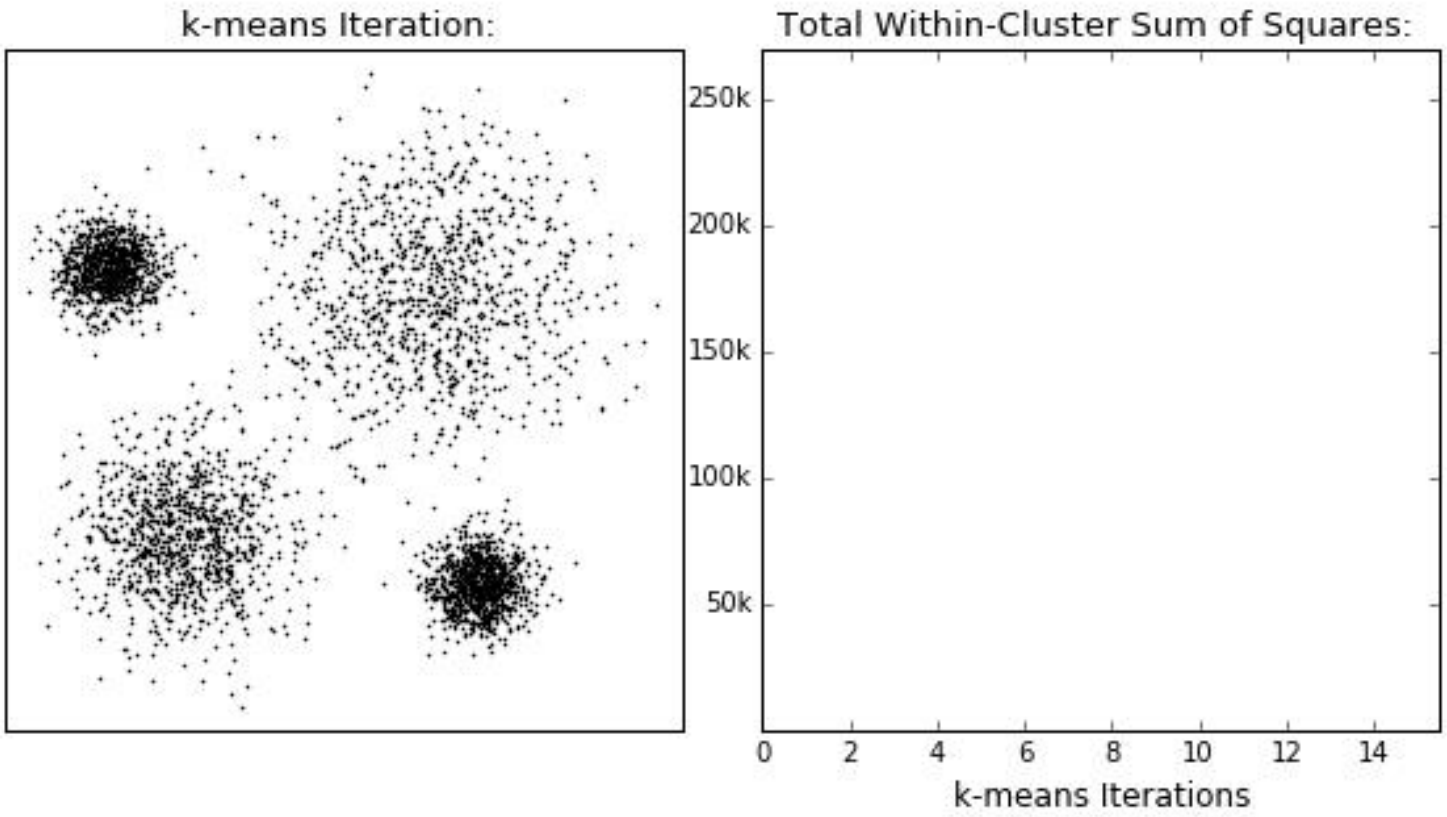


SSE(左图)<SSE(右图)

2 算法效果衡量标准


KMeans Iteration:
Total Within Cluster Sum of Squares:
250k
200k
150k
100k
50k
0
2
4
6
8
10
Kmeans Iterations

2 算法效果衡量标准


k-means Iteration:
Total Within-Cluster Sum of Squares:
250k
200k
150k
100k
50k
0
2
4
6
8
10
12
14
k-means Iterations

2 算法效果衡量标准

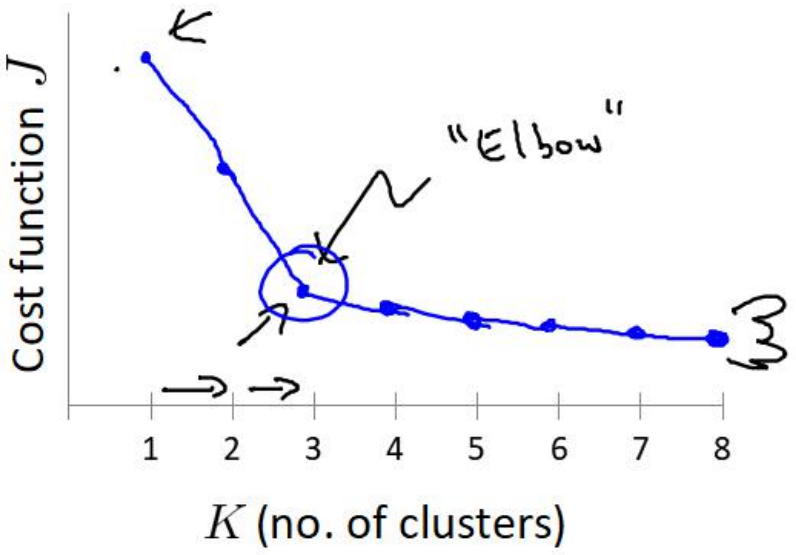
2-2　K值确定

Elbow method就是“肘”方法：

（1）对于n个点的数据集，迭代计算k from 1 to n，每次聚类完成后计算每个点到其所属的簇中心的距离的平方和；

（2）平方和是会逐渐变小的，直到k==n时平方和为0，因为每个点都是它所在的簇中心本身。

（3）在这个平方和变化过程中，会出现一个拐点也即“肘”点，**下降率突然变缓时即认为是最佳的k值**。



2 算法效果衡量标准

2-3 轮廓系数法（Silhouette Coefficient）

结合了聚类的凝聚度（Cohesion）和分离度（Separation），用于评估聚类的效果。

$$S = \frac{(b-a)}{\max(a,b)} \qquad S \in [-1,1]$$

a是Xi与同簇的其他样本的平均距离，称为凝聚度；

b是Xi与最近簇中所有样本的平均距离，称为分离度。

2 算法效果衡量标准

2-3 轮廓系数法（Silhouette Coefficient）

最近簇的定义：

$$C_j = \arg\min_{C_k} \frac{1}{n} \sum_{p \in C_k} \left| p - X_i \right|^2$$

p是某个簇Ck中的样本。即，用Xi到某个簇所有样本平均距离作为衡量该点到该簇的距离后，选择离Xi最近的一个簇作为最近簇。

求出所有样本的轮廓系数后再求平均值就得到了平均轮廓系数。

平均轮廓系数的取值范围为[-1,1]，系数越大，聚类效果越好。

簇内样本的距离越近，簇间样本距离越远

2 算法效果衡量标准

2-4 Calinski-Harabasz Index

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

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

m为训练集样本数，k为类别数，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。

2 算法效果衡量标准

2-4 Calinski-Harabasz Index

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

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

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。

m为训练集样本数，k为类别数，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。

2 算法效果衡量标准

2-4 Calinski-Harabasz Index

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

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

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。

m为训练集样本数，k为类别数，tr为矩阵的迹。

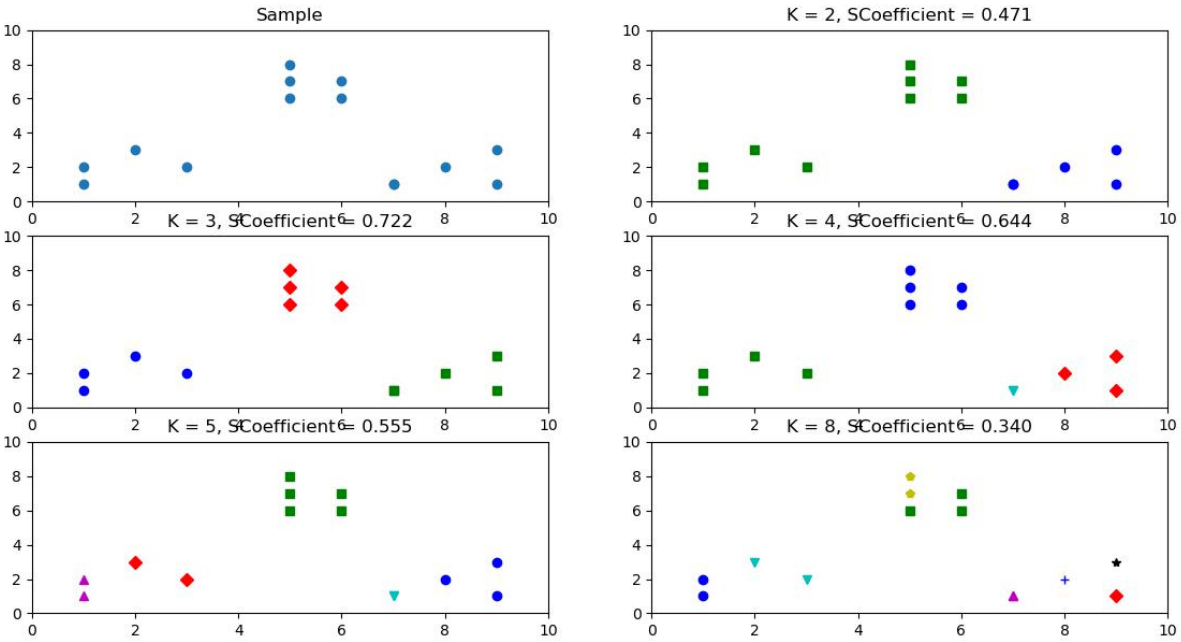
B_k为类别之间的协方差矩阵，W_k为类别内部数据的协方差矩阵。

2 算法效果衡量标准

案例2： k-means聚类效果评估

（1）介绍

我们已经学过SC系数等关于簇聚类效果的评估方法，请在案例1生成的数据集及簇聚类结果基础之上，进一步分析其在不同K值下的聚类效果，实现如右图示例。



2 算法效果衡量标准

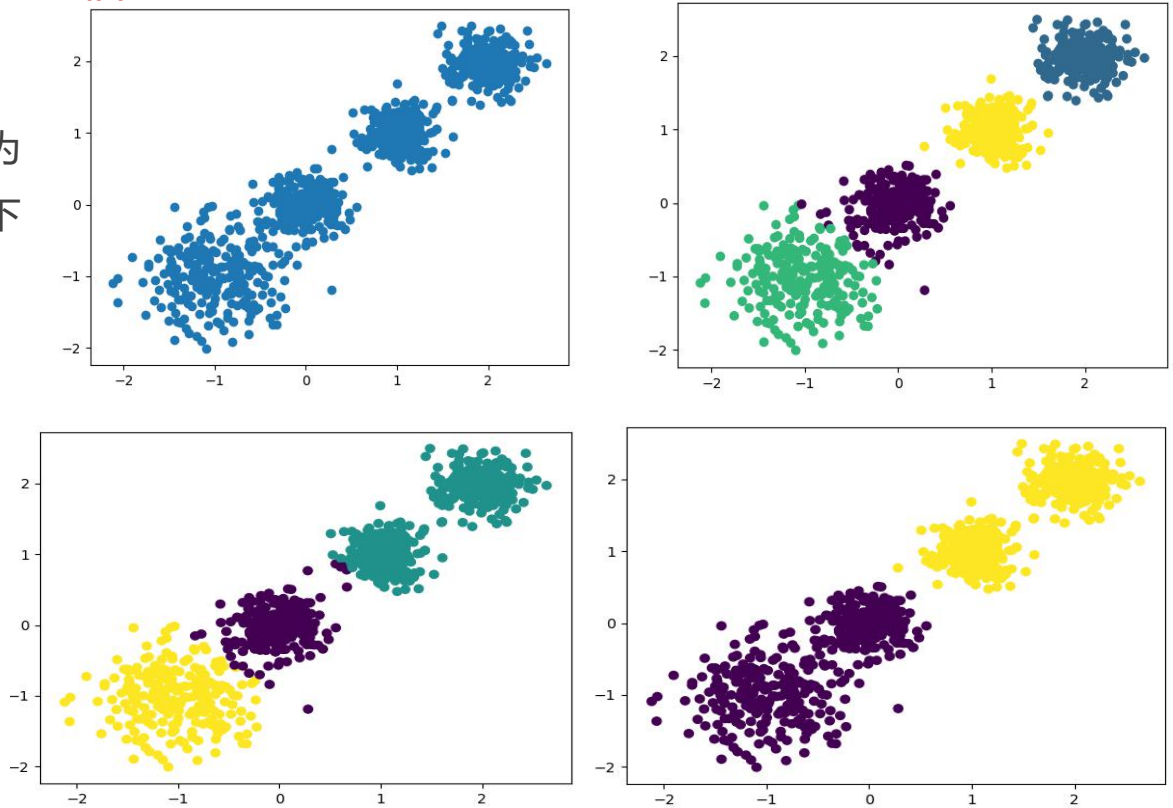
案例2： k-means聚类效果评估

（2）分析

随机创建一些二维数据作为训练集，观察在不同的k值下Calinski-Harabasz分数。

（3）实现

如右图

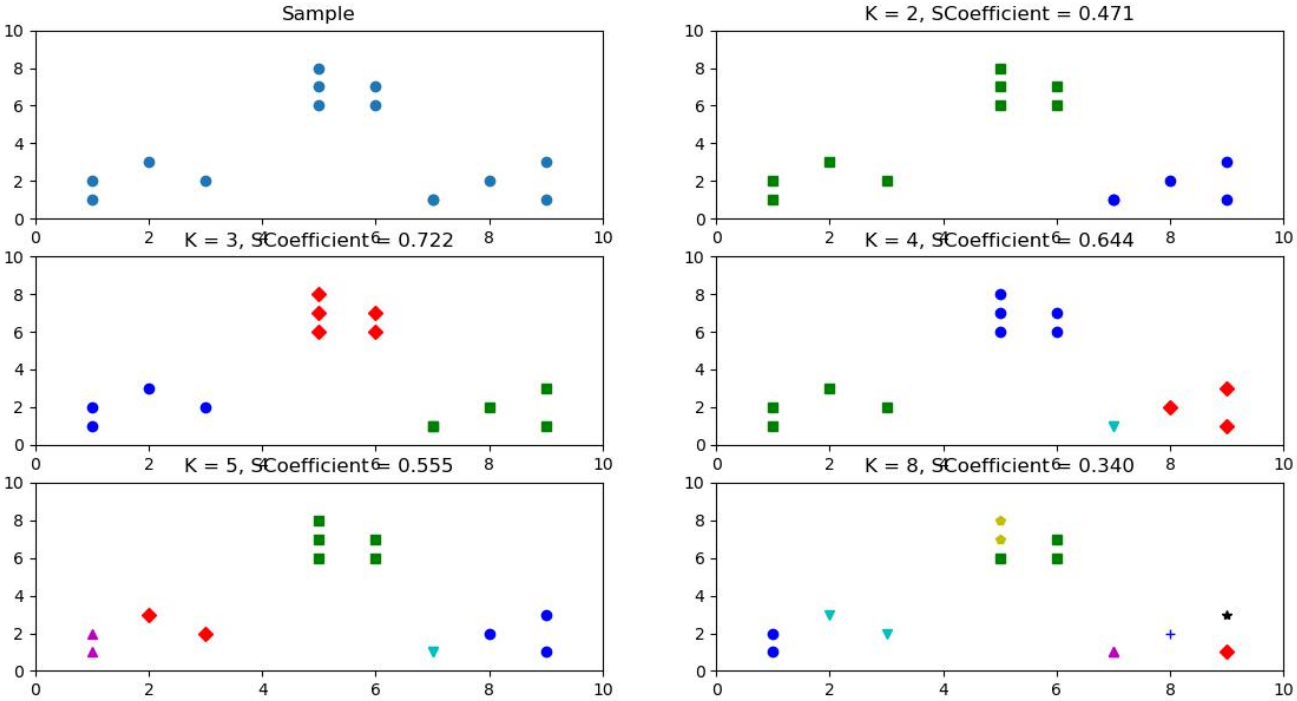


2 算法效果衡量标准

案例2： k-means聚类效果评估

（4）小结

轮廓系数取值为[-1, 1]，其值越大越好，且当值为负时，表明ai<bi，样本被分配到错误的簇中，聚类结果不可接受。对于接近0的结果，则表明聚类结果有重叠的情况。



总结

总结

1. 肘部法

下降率突然变缓时即认为是最佳的k值

2. SC系数

取值为[-1, 1]，其值越大越好

3. CH系数

分数s高则聚类效果越好

2 算法效果衡量标准

案例3： 聚类算法的图片压缩实战应用

（1）介绍

运用Kmeans算法实现图像压缩，并观察压缩后图像的变化

（2）分析

将K分别设置成8，32，64等，通过实现颜色聚类来实现

（3）实现

pycharm

jupyter notebook

黑马程序员
www.itheima.com
传智播客旗下高端IT教育品牌

目录 Contents

3 算法优化

3-1 算法优点

1

原理简单（靠近中心点），实现容易

2

聚类效果中上（依赖K的选择）

3

空间复杂度o(N)
时间复杂度o(I*K*N)

N为样本点个数，K为中心点个数，I为迭代次数

3 算法优化

3-1 算法缺点

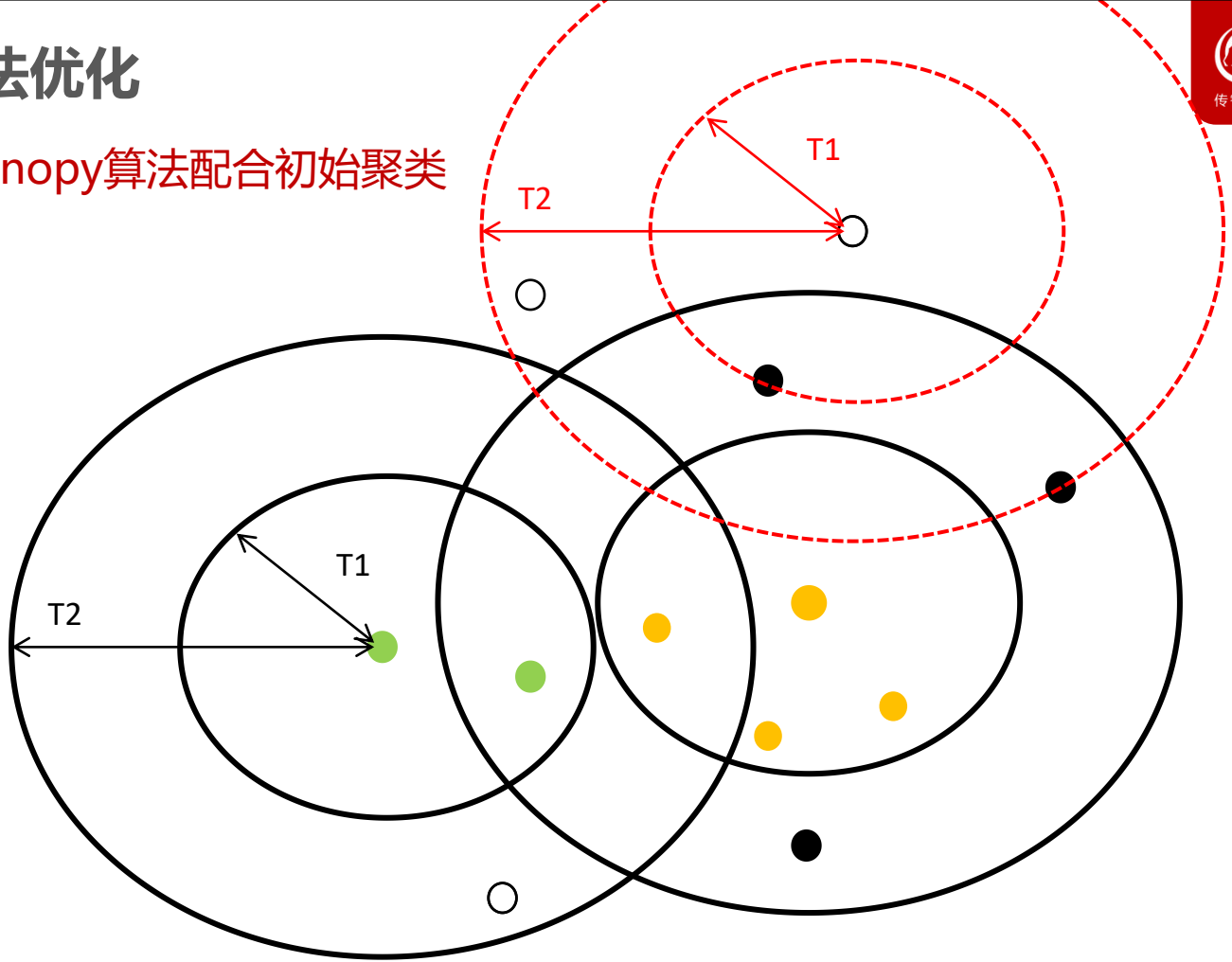
1. 对离群点，噪声敏感 （中心点易偏移）

2. 很难发现大小差别很大的簇及进行增量计算

3. 结果不一定是全局最优，只能保证局部最优（与K的个数及初值选取有关）

3 算法优化

3-2 Canopy算法配合初始聚类



3 算法优化

黑马程序员
www.itheima.com
传智播客旗下高端IT教育品牌

3-2 Canopy算法配合初始聚类

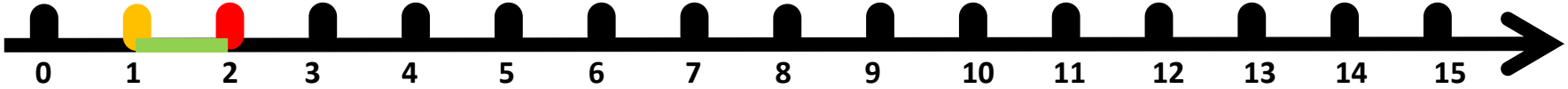
1. Kmeans对噪声抗干扰较弱，通过Canopy对比，将较小的NumPoint的Cluster直接去掉有利于抗干扰。
2. Canopy选择出来的每个Canopy的centerPoint作为K会更精确。
3. 只是针对每个Canopy的内做Kmeans聚类，减少相似计算的数量。

1. 算法中 T1、T2的确定问题

3 算法优化

3-3 K-means++

$$P = \frac{D(x)^2}{\sum_{x \in X} D(x)^2}$$



$$P(2\_1) = \frac{1}{15}$$

3 算法优化

3-3 K-means++

$$P = \frac{D(x)^2}{\sum_{x \in X} D(x)^2}$$

$$P(2\_1) = \frac{6}{15}$$

3 算法优化

3-3 K-means++

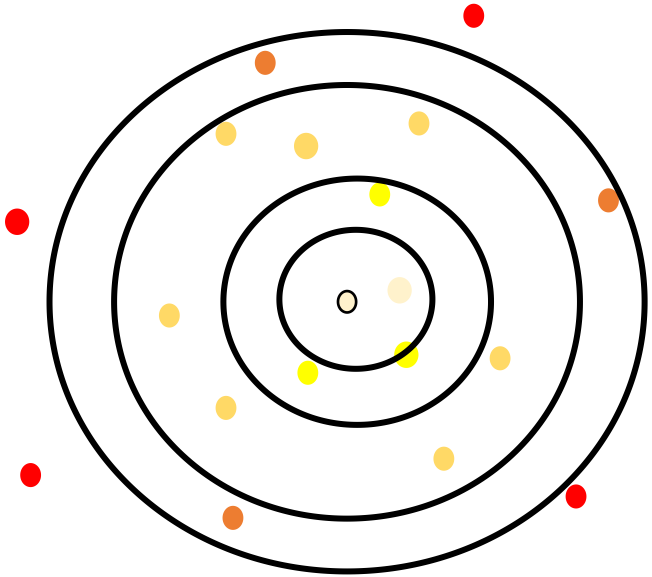
$$P = \frac{D(x)^2}{\sum_{x \in X} D(x)^2}$$

$$P(2\_1) = \frac{13}{15}$$

3 算法优化

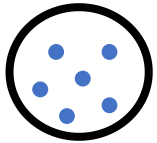
3-3 K-means++

P(A)>P(B)>P(C)>P(D)>P(E)



3 算法优化

3-3 K-means++

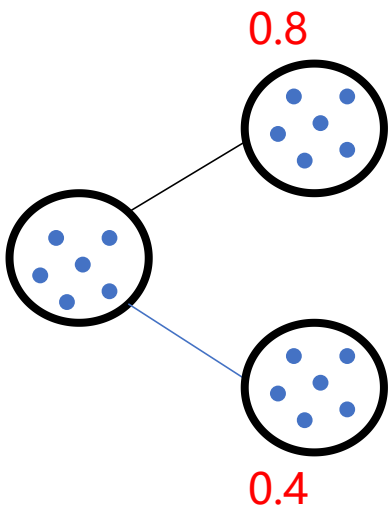


1. 所有点作为一个簇
2. 将该簇一分为二
3. 选择能最大限度降低聚类代价函数（也就是误差平方和）的簇划分为两个簇。

.4 以此进行下去，直到簇的数目等于用户给定的数目k为止。

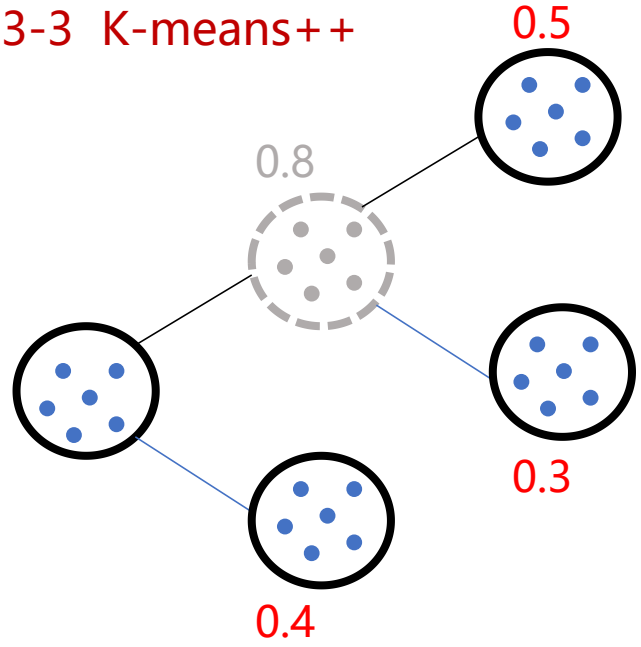
3 算法优化

3-3 K-means++



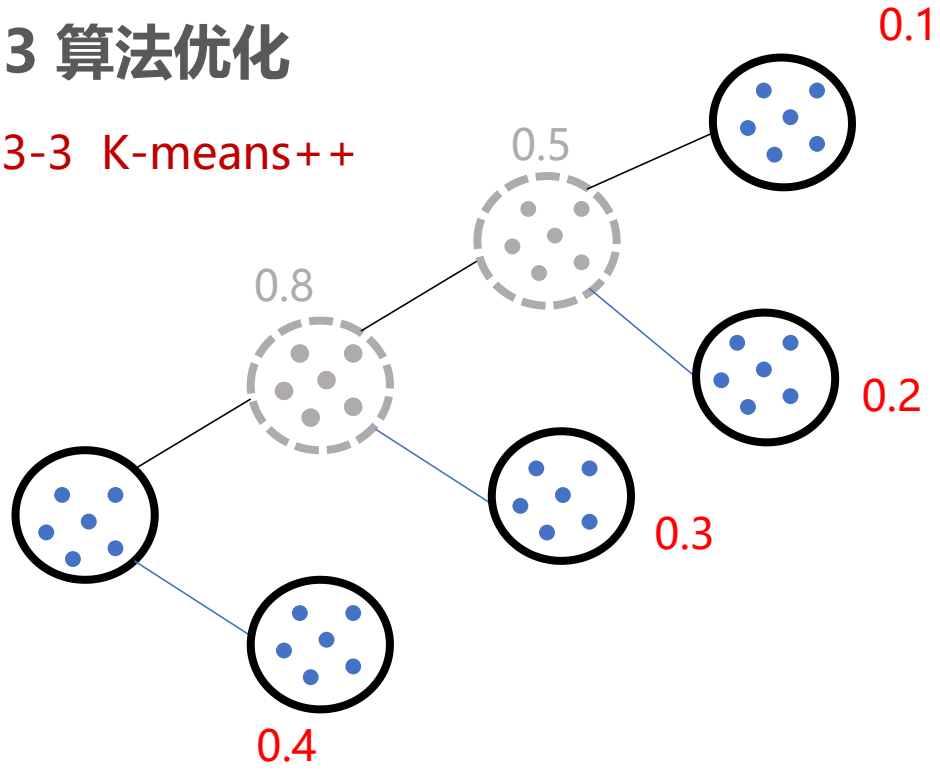
3 算法优化

3-3 K-means++



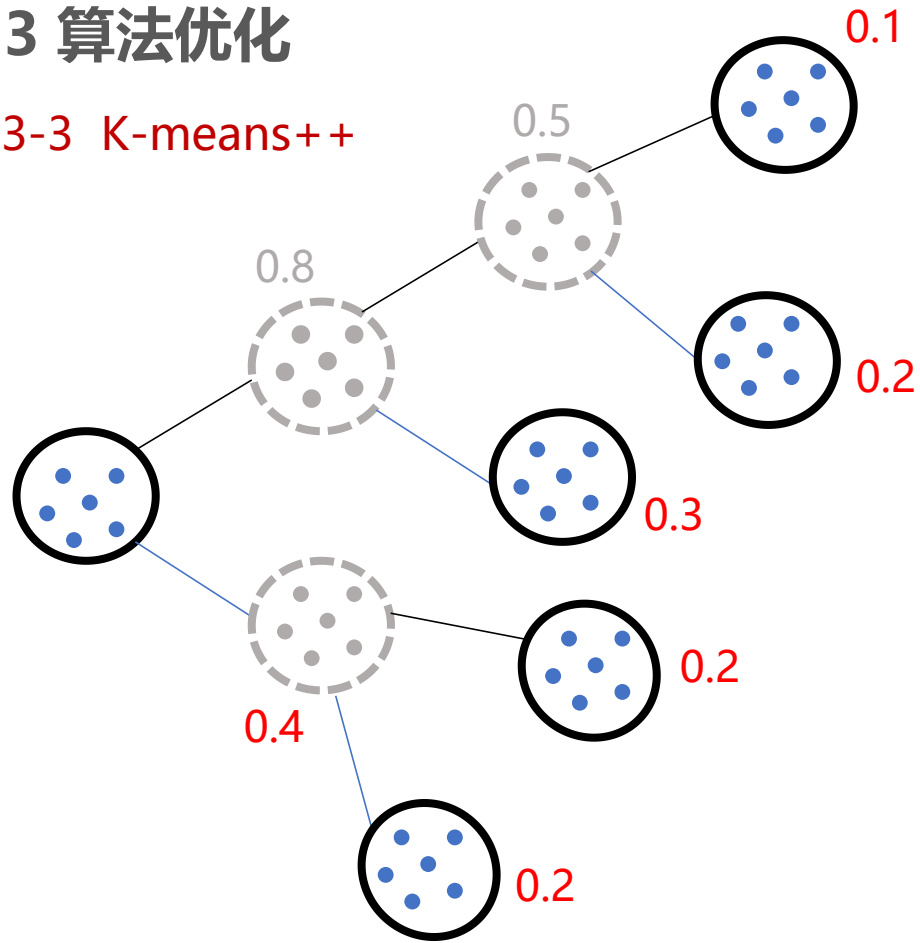
3 算法优化

3-3 K-means++



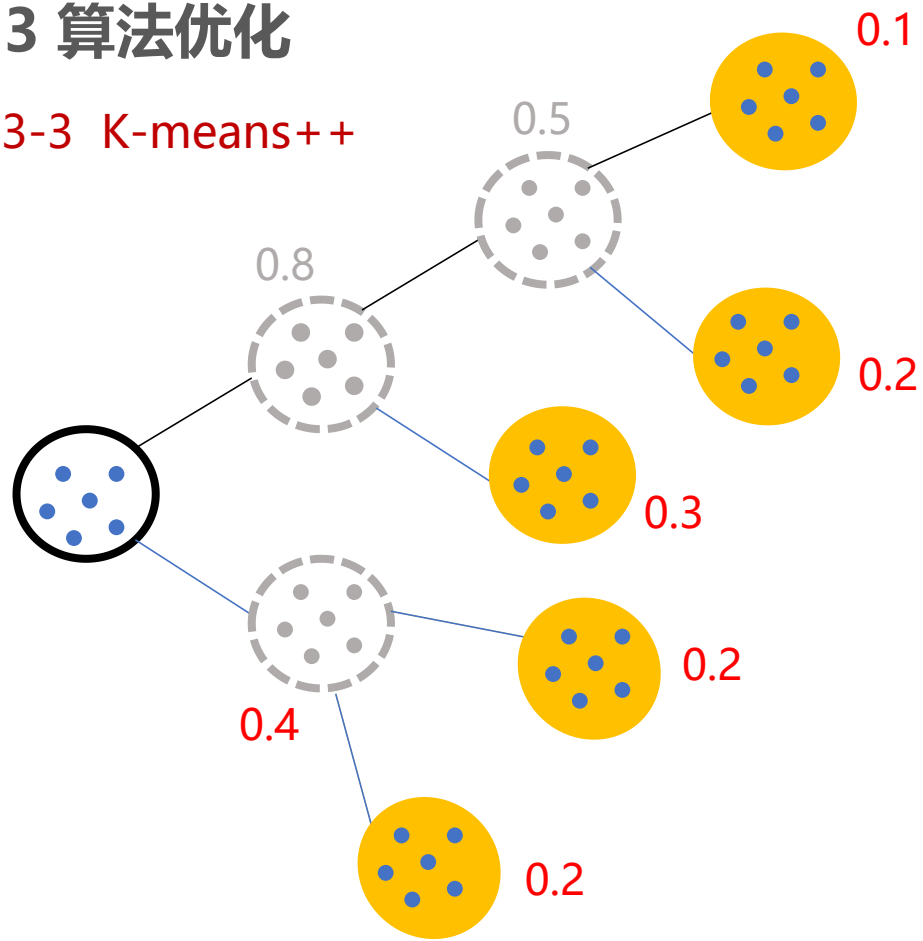
3 算法优化

3-3 K-means++



3 算法优化

3-3 K-means++

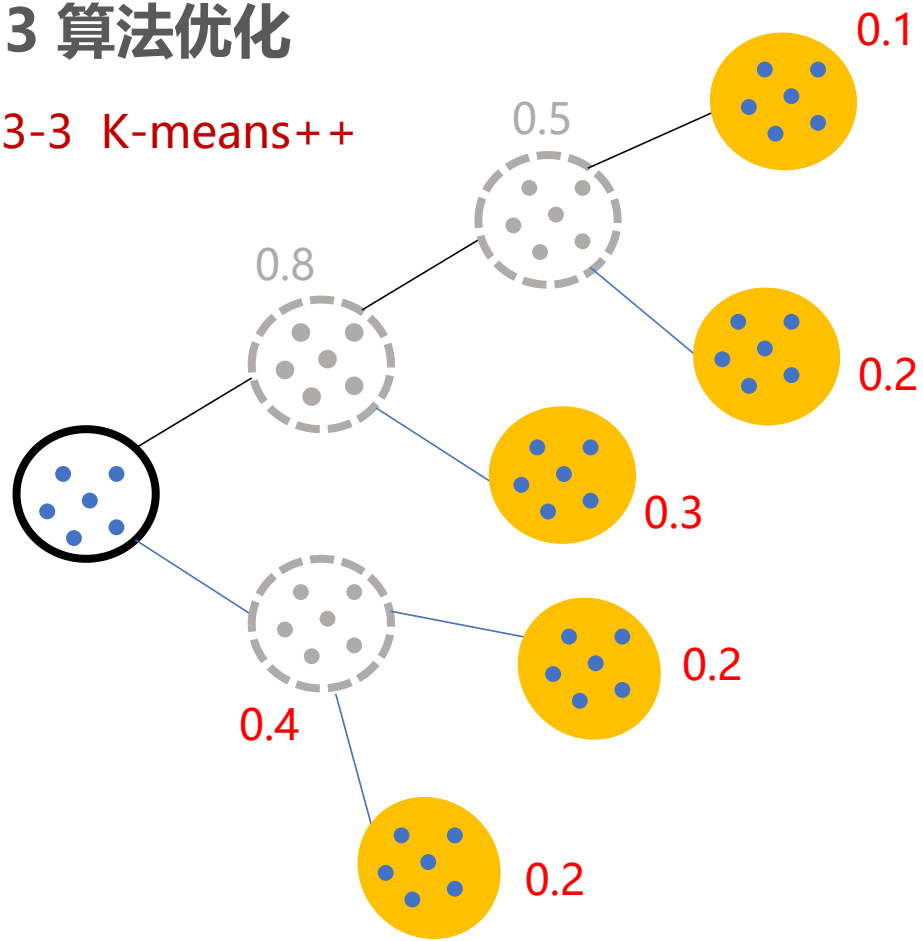


隐含的一个原则

因为聚类的误差平方和能够衡量聚类性能，该值越小表示数据点越接近于他们的质心，聚类效果就越好。所以需要对误差平方和最大的簇进行再一次划分，因为误差平方和越大，表示该簇聚类效果越不好，越有可能是多个簇被当成了一个簇，所以我们首先需要对这个簇进行划分。

3 算法优化

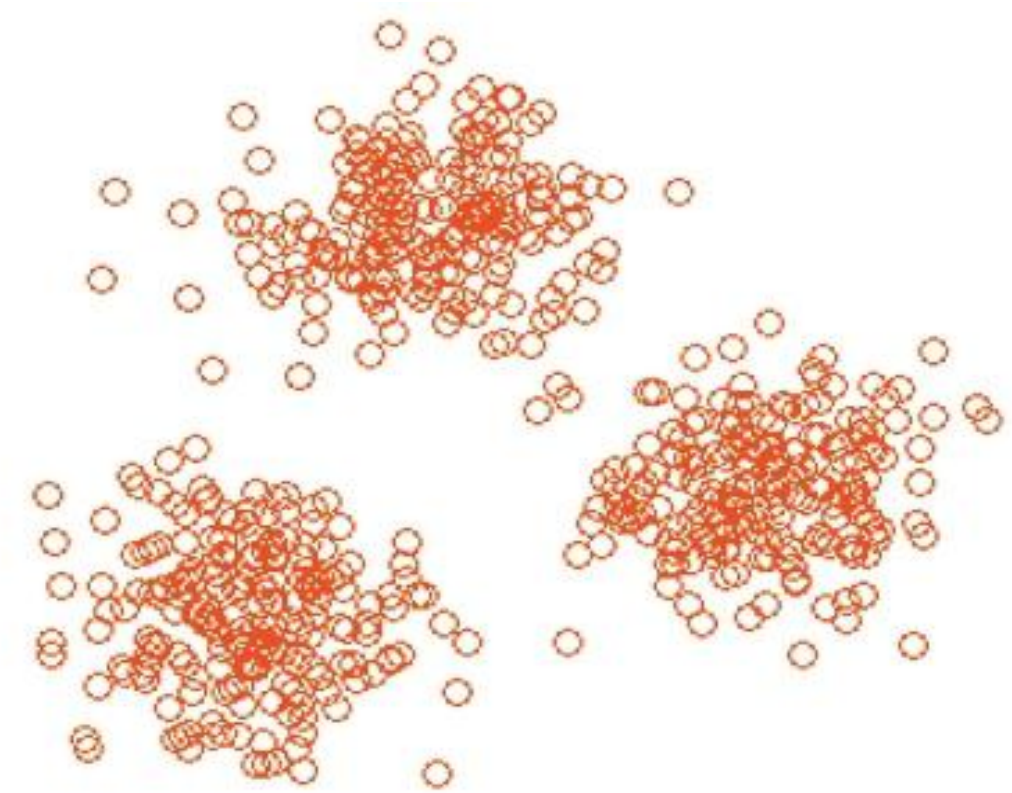
3-3 K-means++



二分K均值算法可以加速K-means算法的执行速度，因为它的相似度计算少了并且不受初始化问题的影响，因为这里不存在随机点的选取，且每一步都保证了误差最小

3 算法优化

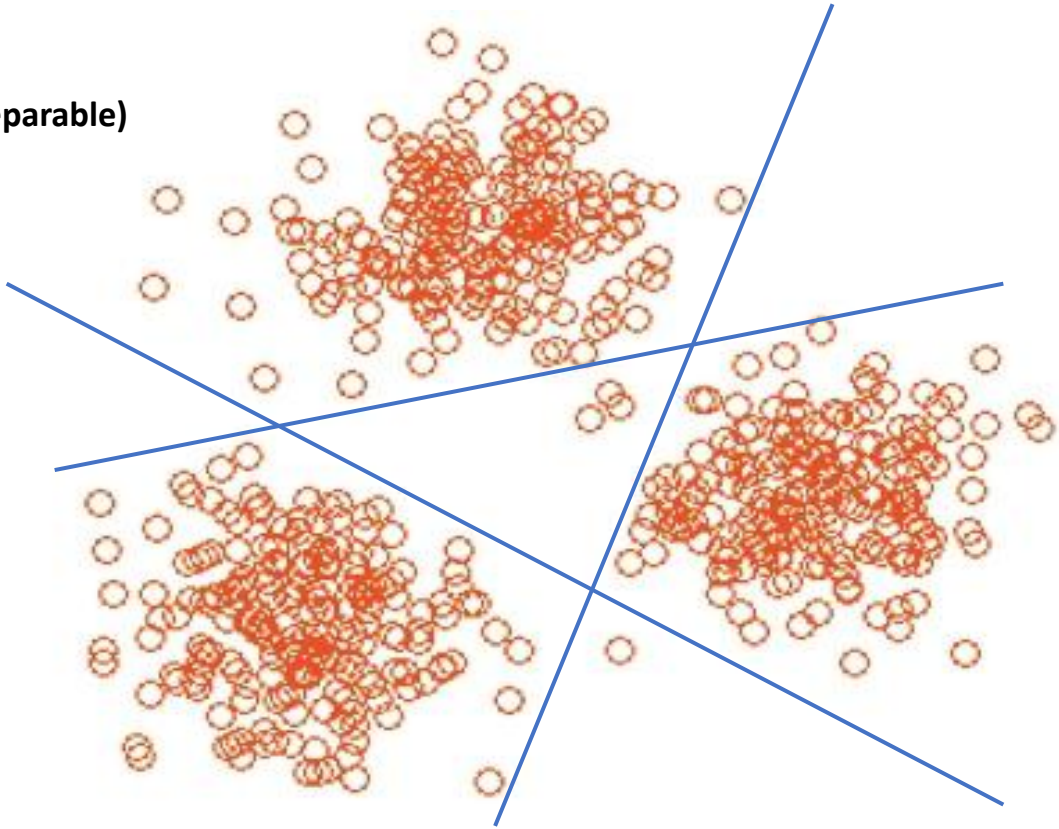
3-4 Kernel k-means



3 算法优化

3-4 Kernel k-means

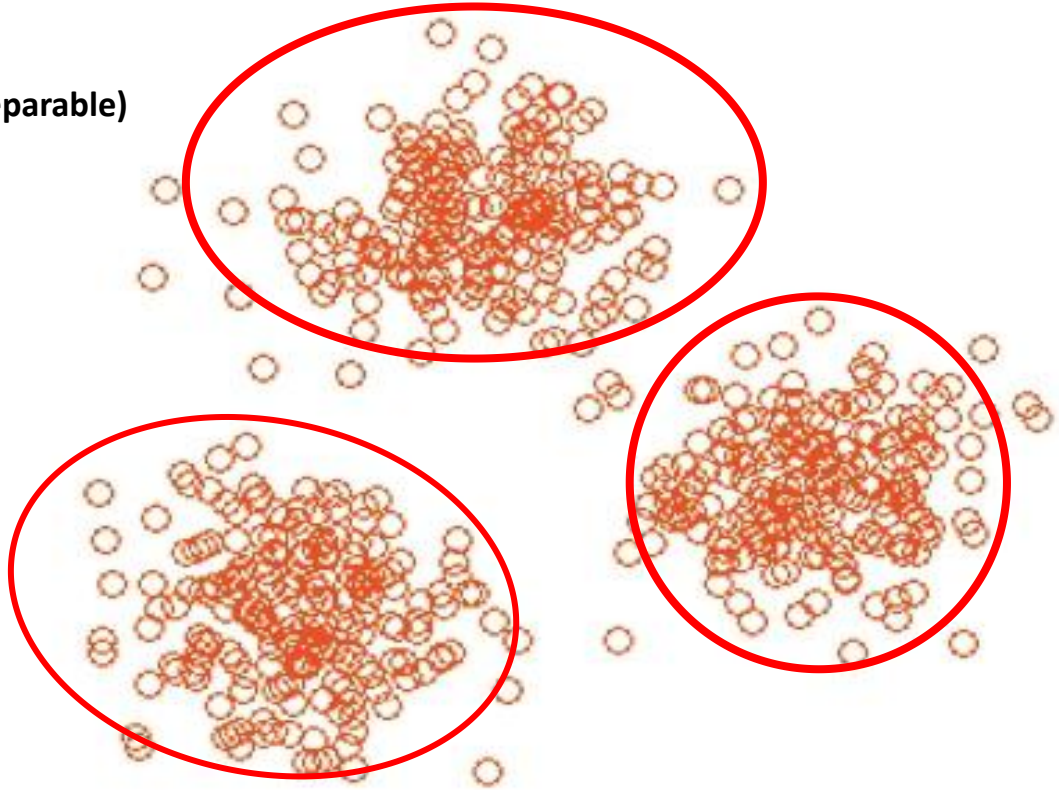
线性可分(linearly separable)



3 算法优化

3-4 Kernel k-means

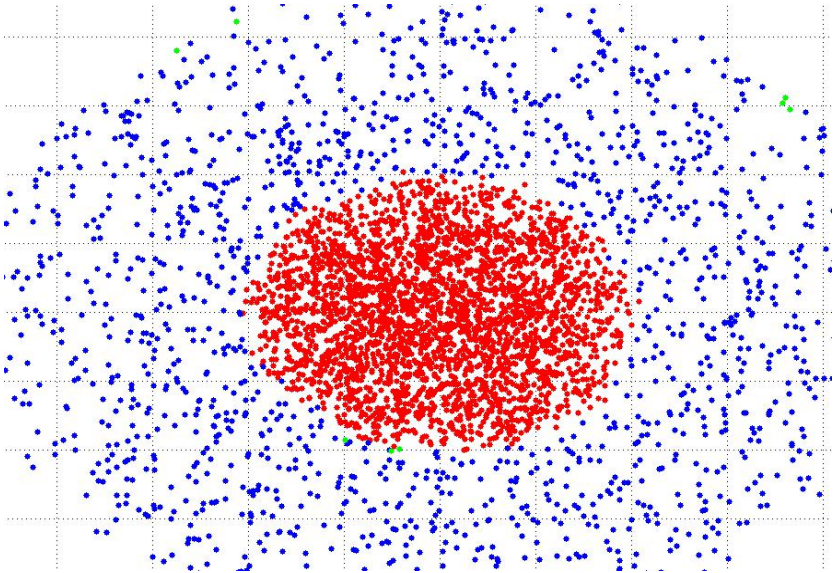
线性可分(linearly separable)

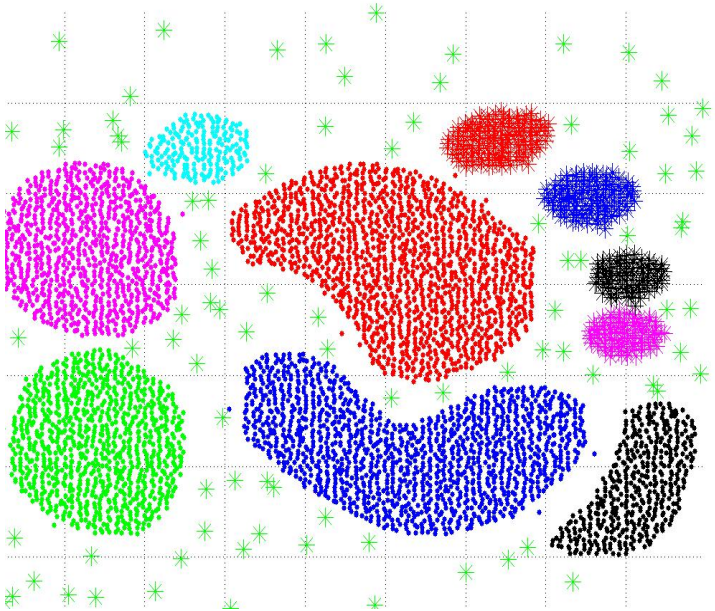


3 算法优化

3-4 Kernel k-means

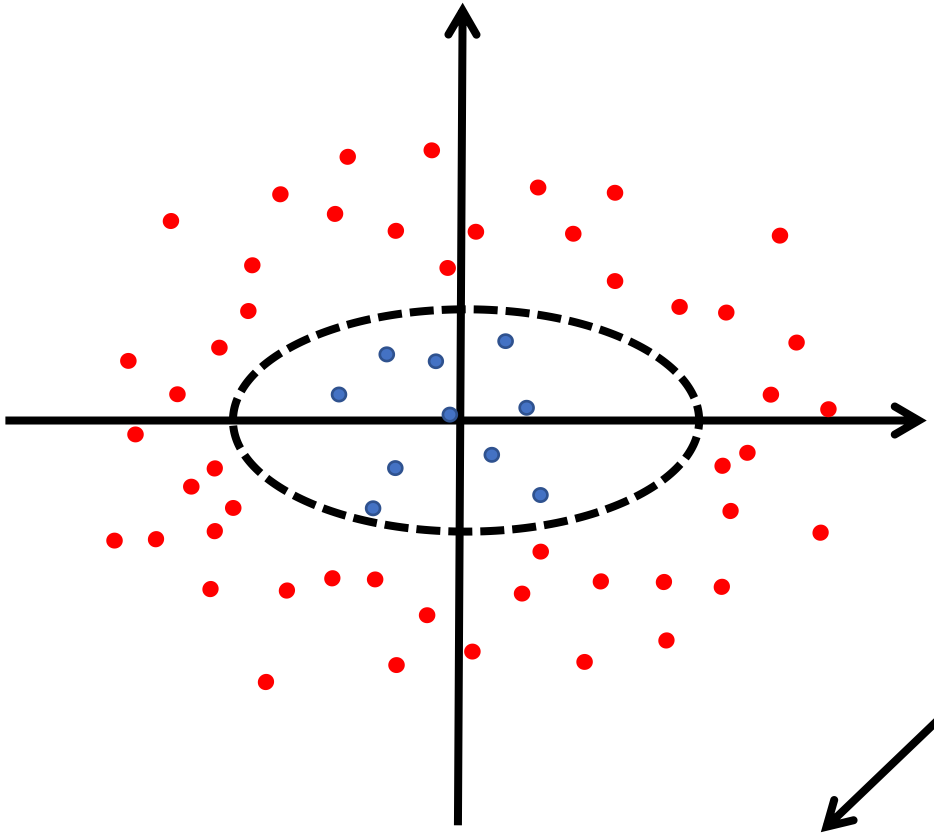
非线性可分(**not-linearly separable**)

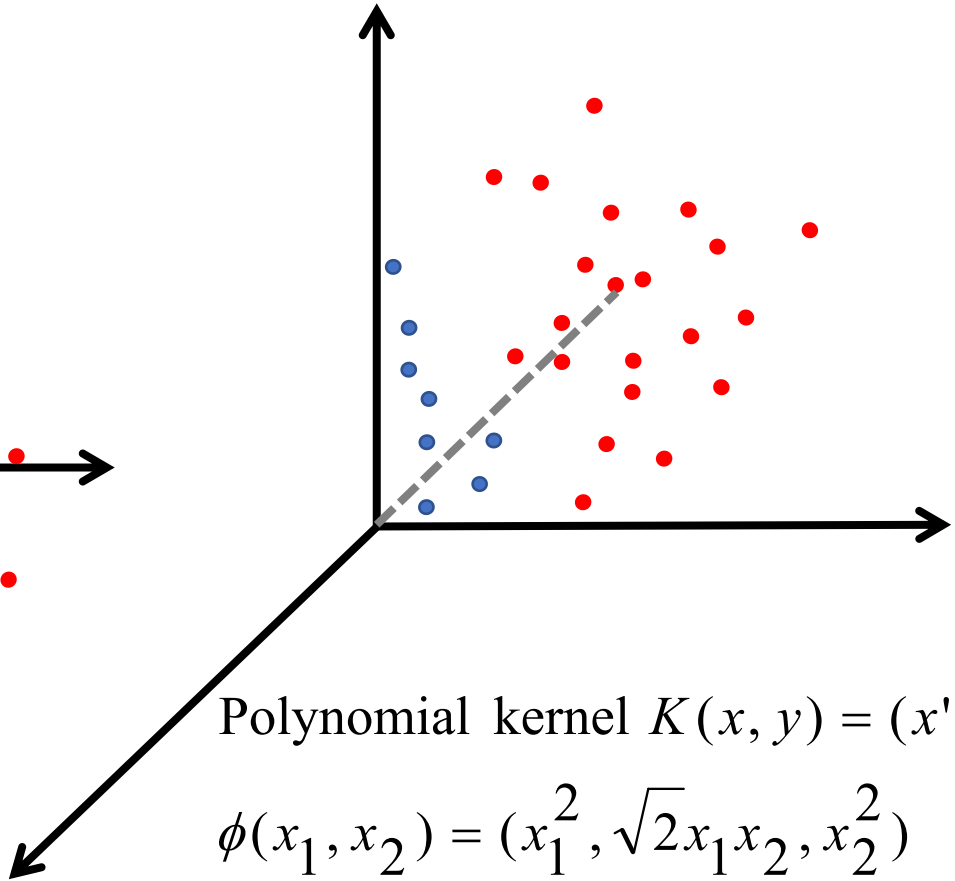




3 算法优化

3-4 Kernel k-means



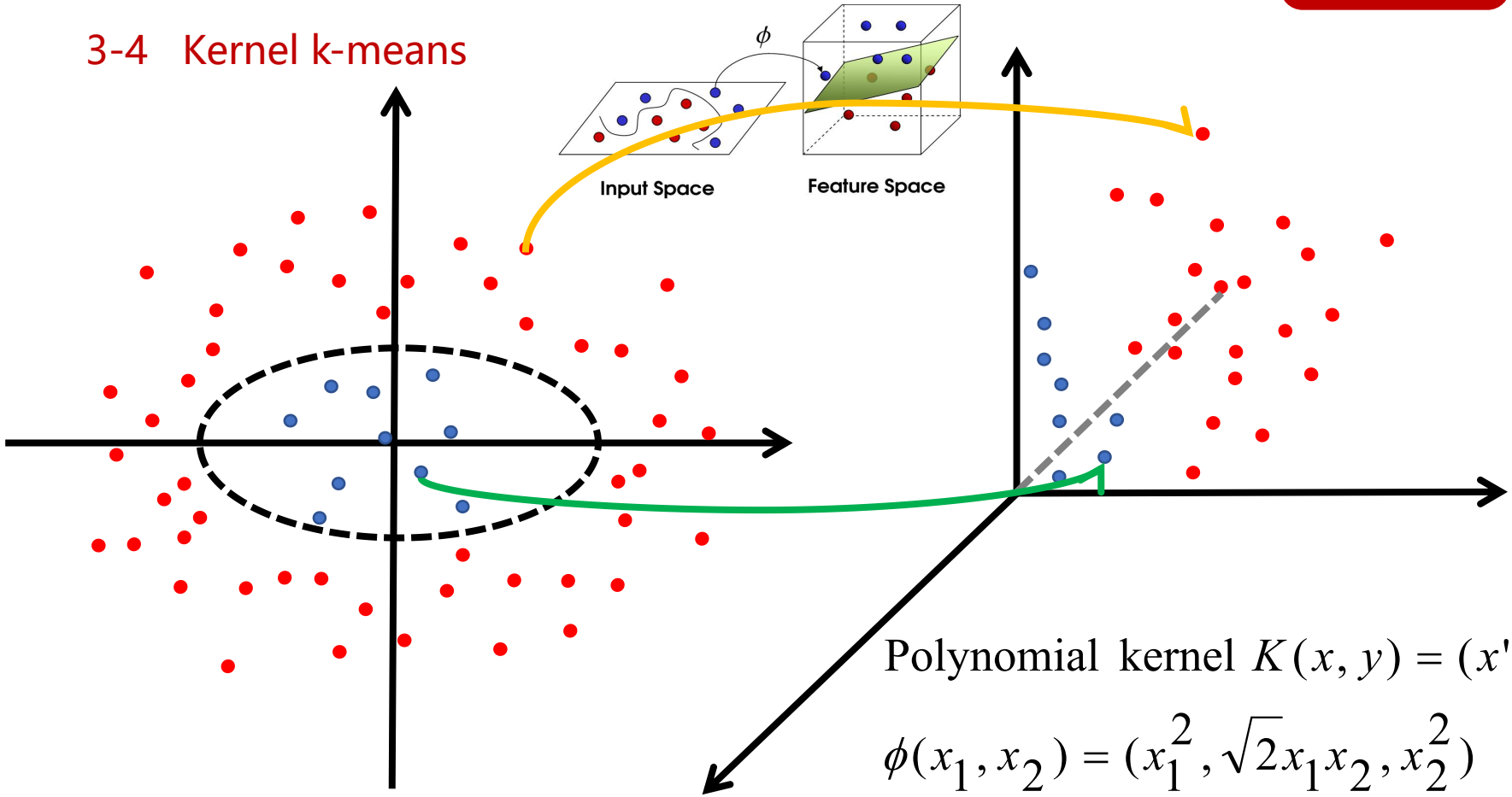


Polynomial kernel $K(x, y) = (x' y)^2$

$$\phi(x_1, x_2) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

3 算法优化

3-4 Kernel k-means

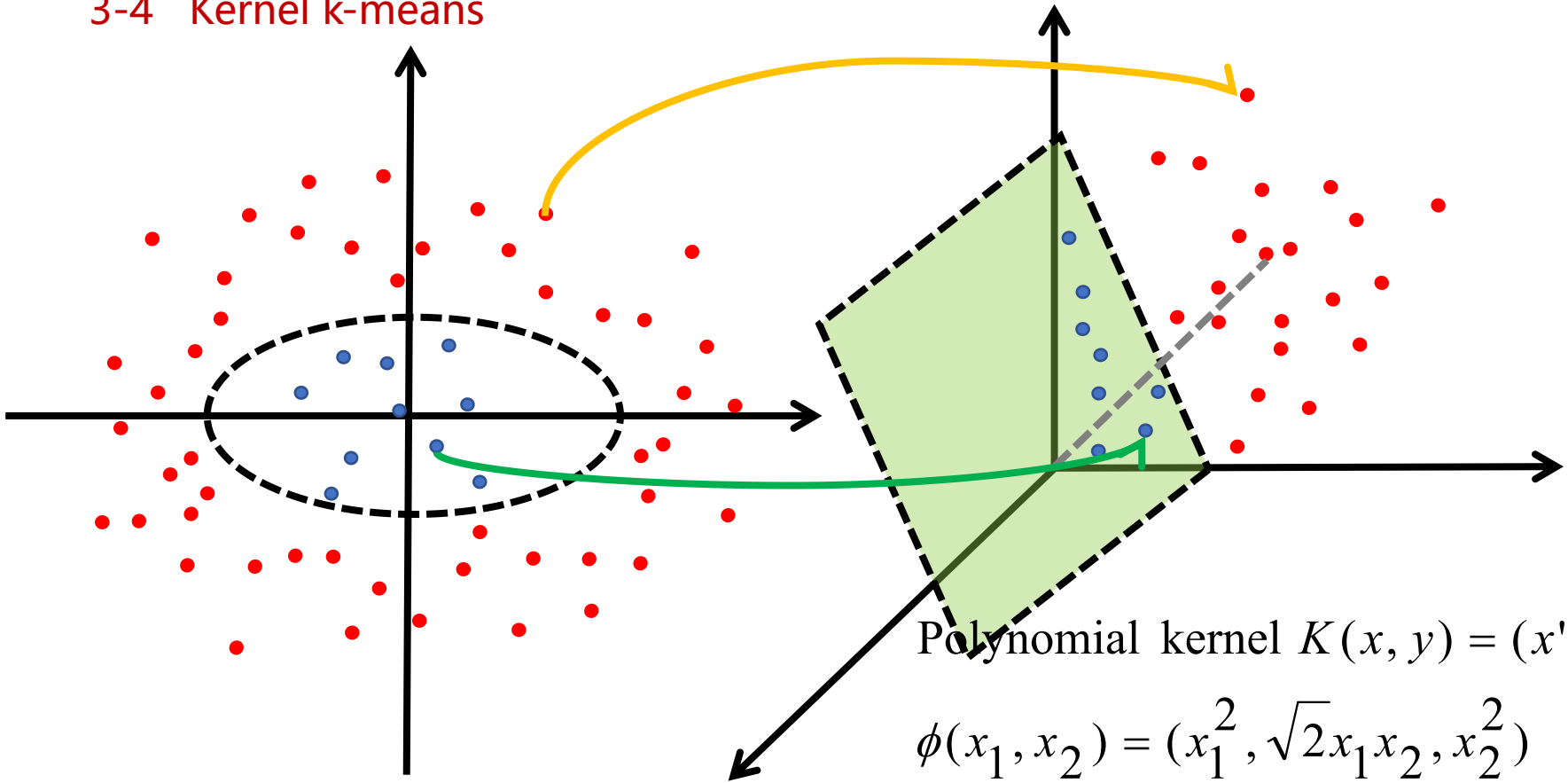


Polynomial kernel $K(x, y) = (x' y)^2$

$$\phi(x_1, x_2) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

3 算法优化

3-4 Kernel k-means

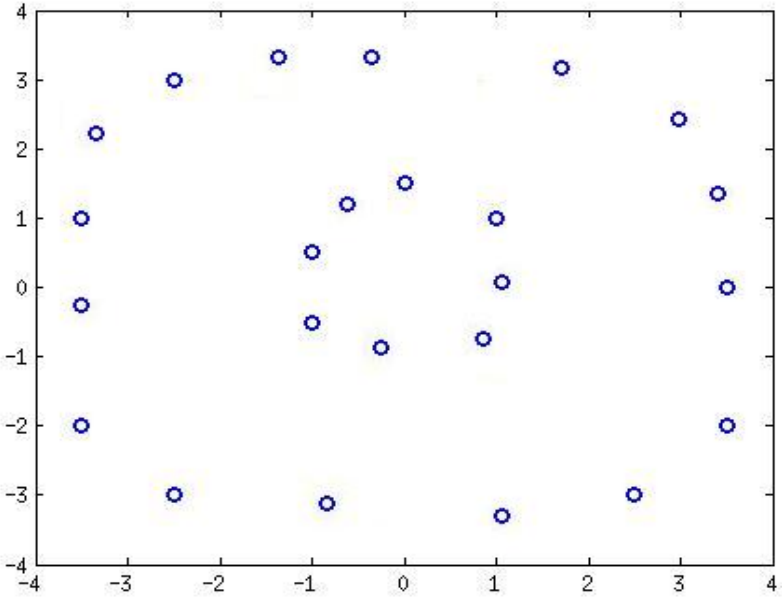


Polynomial kernel $K(x, y) = (x' y)^2$

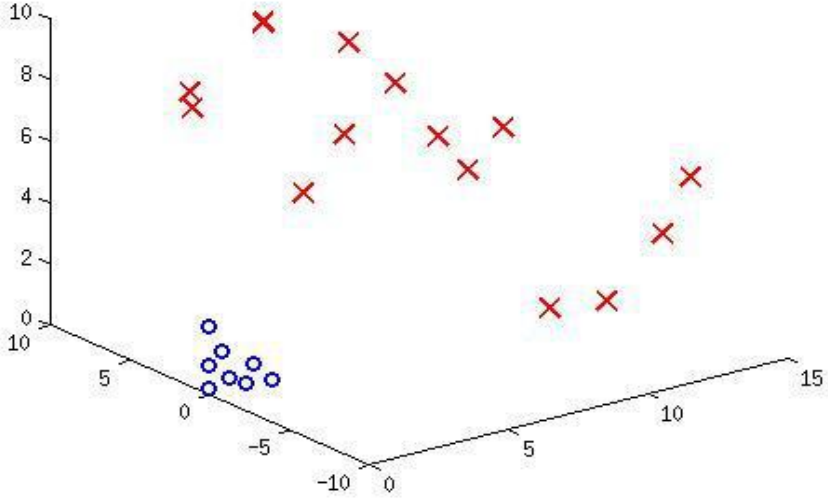
$$\phi(x_1, x_2) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

3 算法优化

3-4 Kernel k-means



ϕ

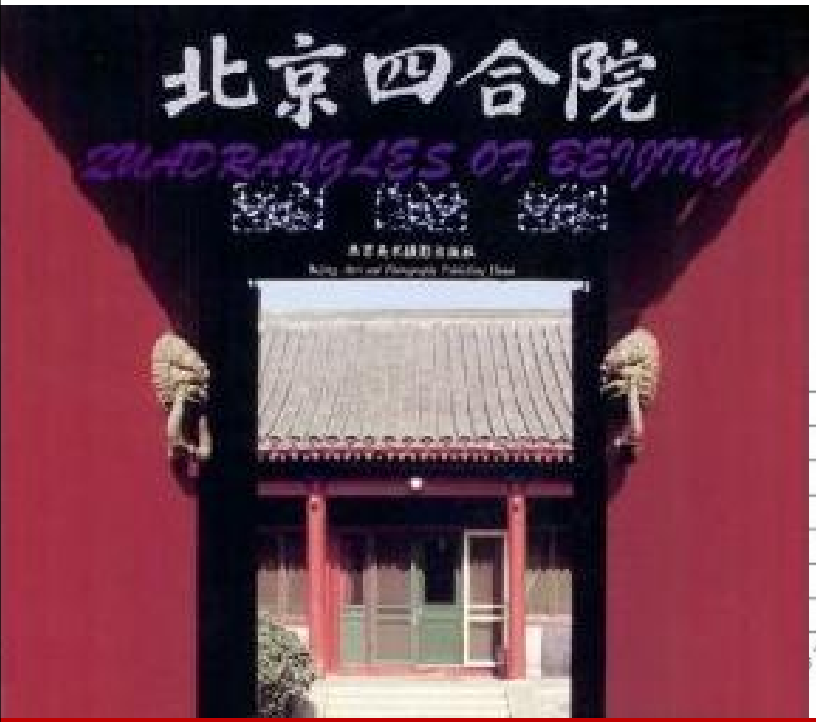


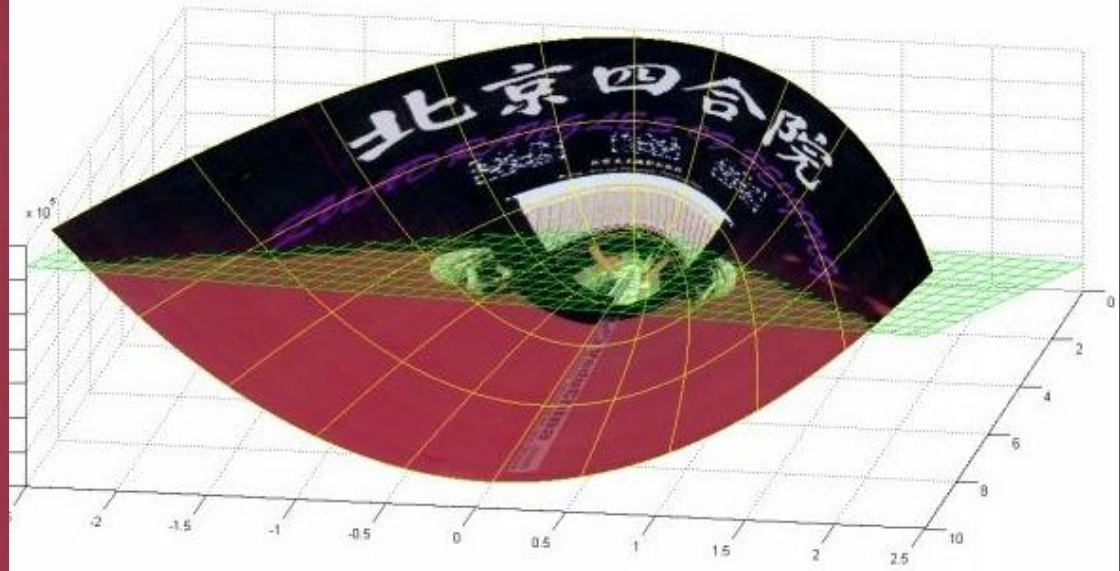
Polynomial kernel $K(x, y) = (x' y)^2$

$$\phi(x_1, x_2) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

3 算法优化

3-4 Kernel k-means

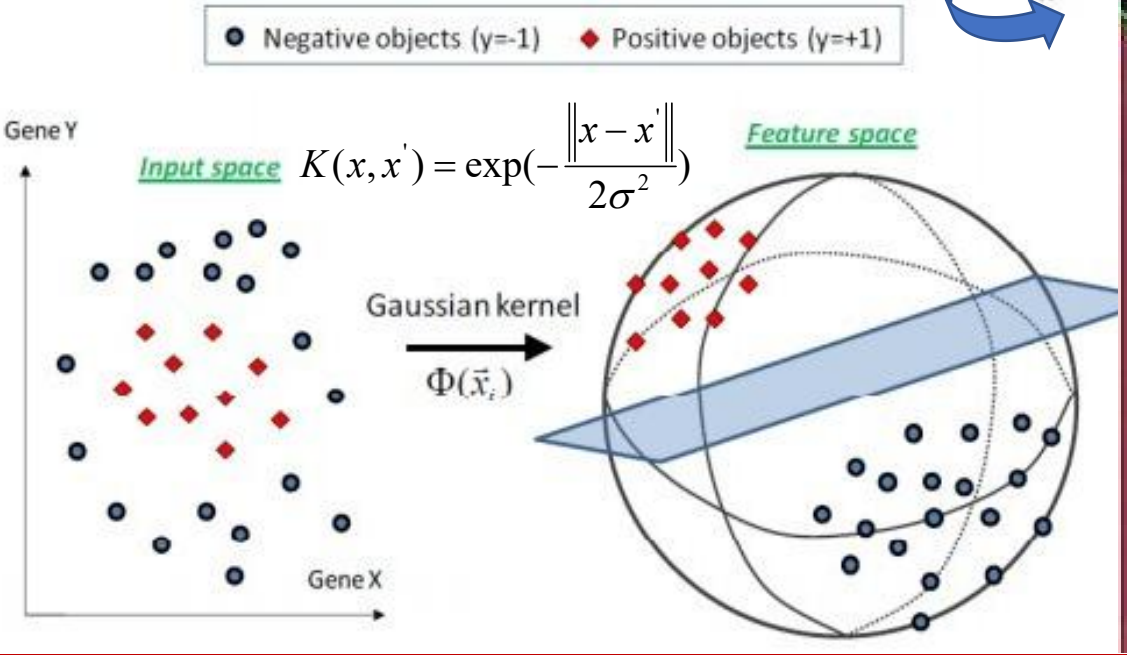




3 算法优化

3-4 Kernel k-means

$x_1(0, 0), x_2(4, 4), x_3(-4, 4), x_4(-4, -4), x_5(4, -4)$



3 算法优化

3-4 Kernel k-means

$$||x_1 - x_2||^2 = (0-4)^2 + (0-4)^2 = 32, \text{ therefore, } K(x_1, x_2) = e^{-\frac{32}{2\cdot 4^2}} = e^{-1}$$

Original Space

	x	y
x_1	0	0
x_2	4	4
x_3	-4	4
x_4	-4	-4
x_5	4	-4

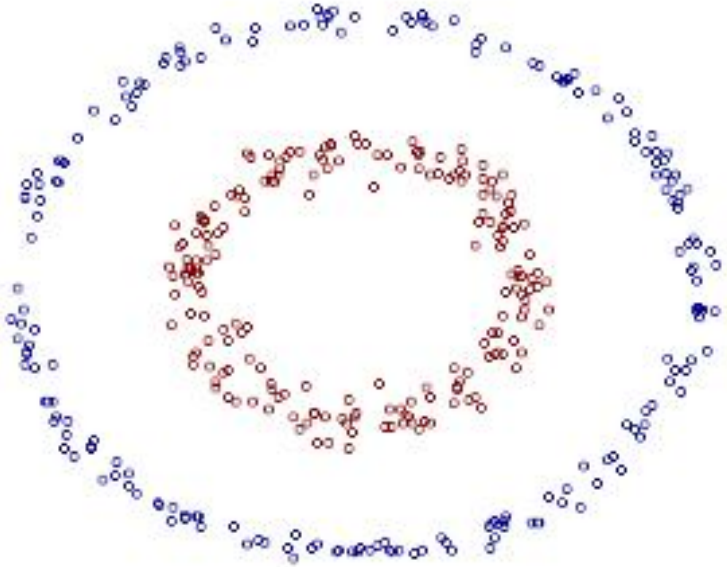
RBF Kernel Space ($\sigma = 4$)

$K(x_i, x_1)$	$K(x_i, x_2)$	$K(x_i, x_3)$	$K(x_i, x_4)$	$K(x_i, x_5)$
0	$e^{-\frac{4^2+4^2}{2\cdot 4^2}} = e^{-1}$	e^{-1}	e^{-1}	e^{-1}
e^{-1}	0	e^{-2}	e^{-4}	e^{-2}
e^{-1}	e^{-2}	0	e^{-2}	e^{-4}
e^{-1}	e^{-4}	e^{-2}	0	e^{-2}
e^{-1}	e^{-2}	e^{-4}	e^{-2}	0

3 算法优化

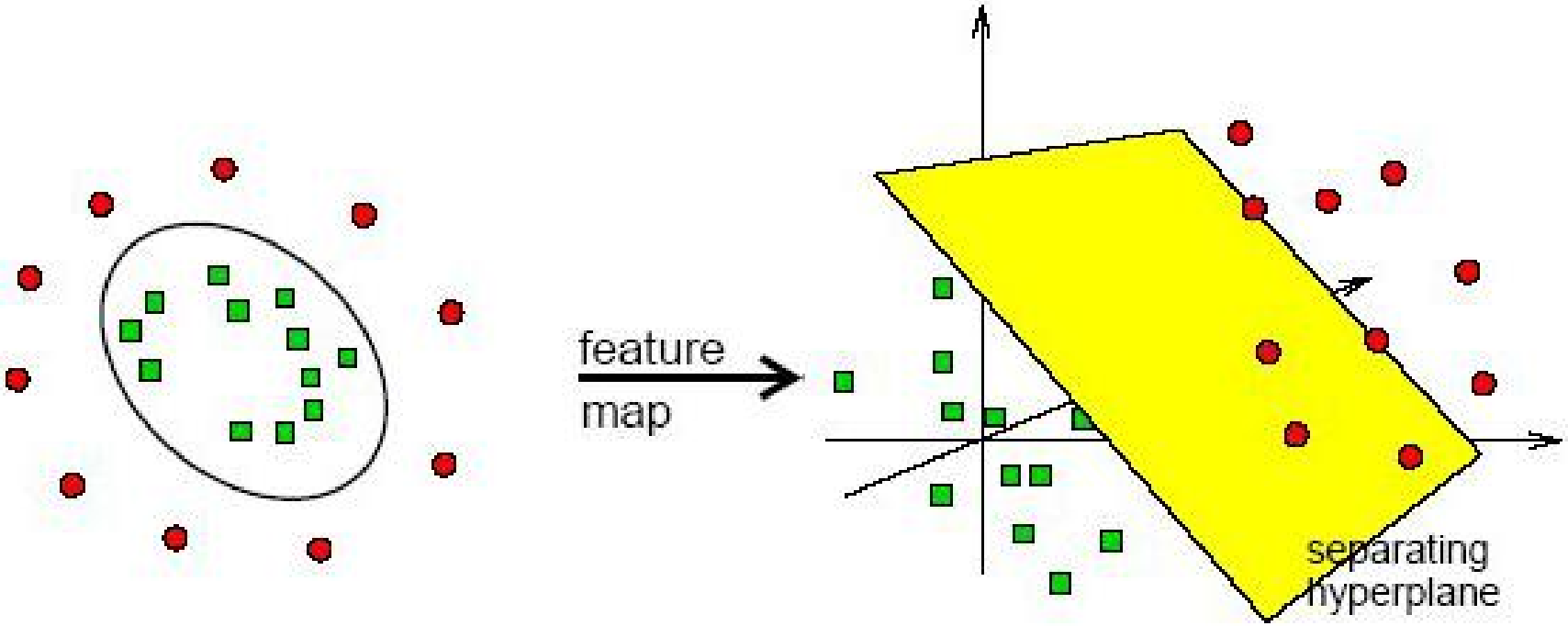
3-4 Kernel k-means

kernel k-means，将每个样本进行一个投射到高维空间的处理，然后再将处理后的数据使用普通的k-means算法思想进行聚类。



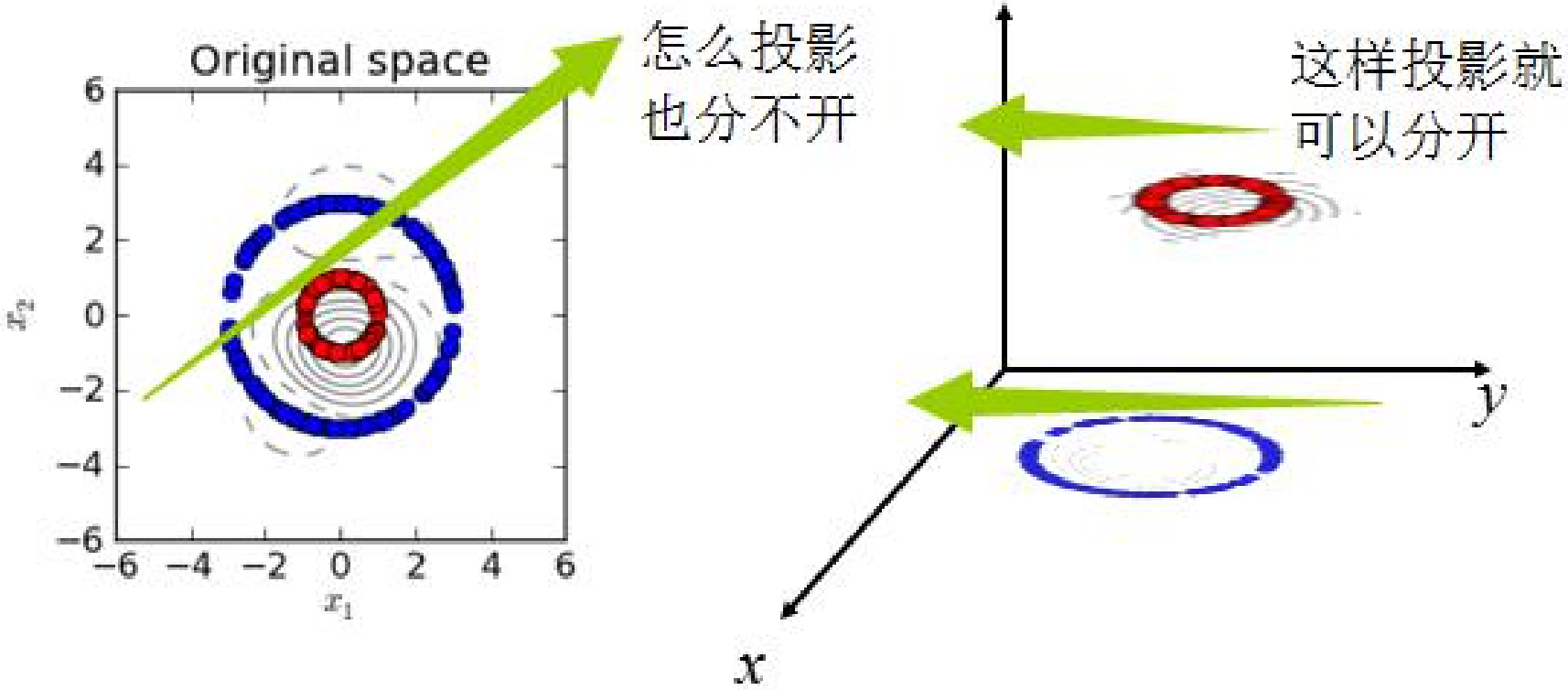
3 算法优化

3-4 Kernel k-means



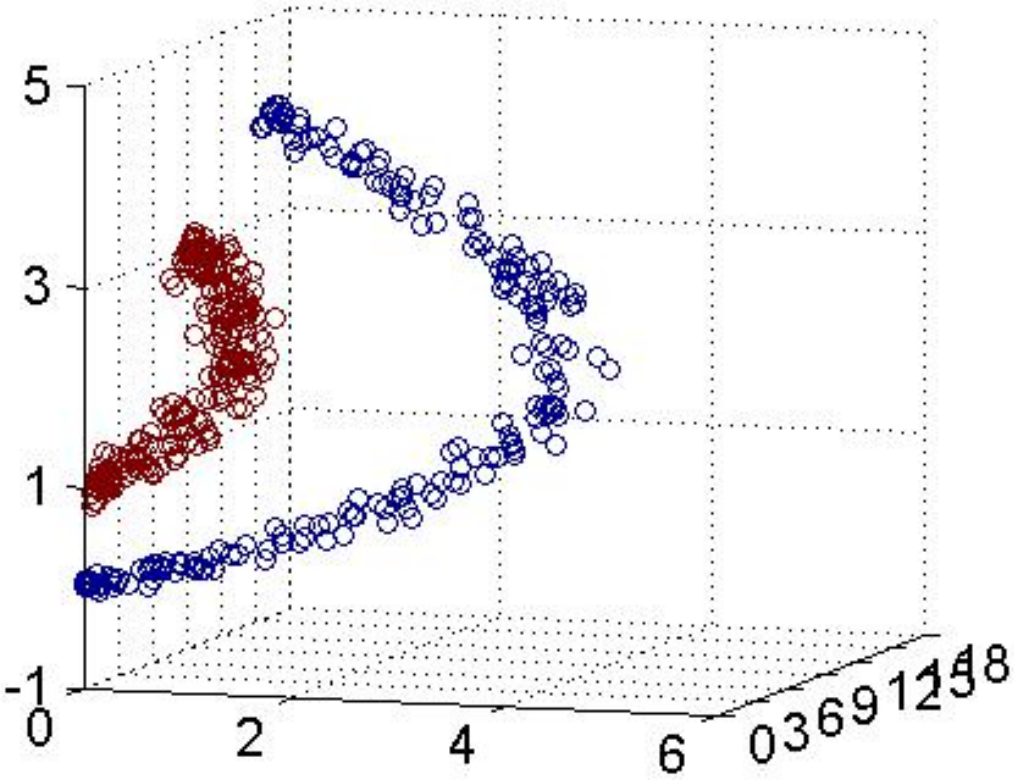
3 算法优化

3-4 Kernel k-means



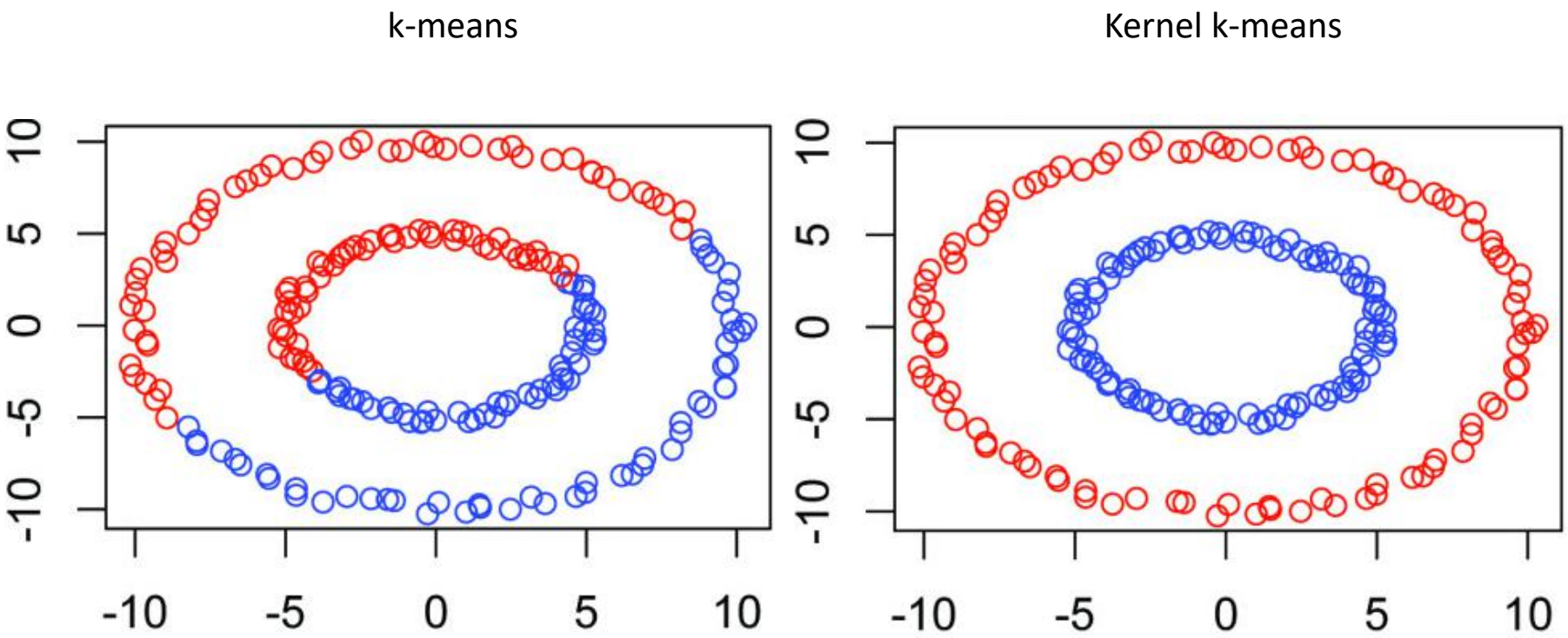
3 算法优化

3-4 Kernel k-means



3 算法优化

3-4 Kernel k-means



3 算法优化

3-4 Kernel k-means

Clustering accuracy (%) achieved by each clustering algorithm for 10 data sets

Data set	Conventional				Kernel			
	k-means	FCM	Average	Mountain	*k*-means	FCM	Average	Mountain
BENSAID	79.59	73.47	100.0	85.71	83.67	93.88	100.0	100.0
DUNN	70.00	70.00	100.0	83.33	71.11	95.56	100.0	100.0
IRIS	89.33	89.33	90.67	52.67	96.00	93.33	89.33	93.33
ECOLI	42.86	49.11	76.49	51.19	68.75	61.01	77.38	69.05
CIRCLE	50.76	52.79	62.44	55.84	100.0	93.40	82.74	62.94
BLE-3	65.67	65.67	56.00	70.33	76.33	74.67	100.0	71.67
BLE-2	88.50	87.75	100.0	85.25	100.0	94.00	100.0	100.0
UE-4	77.25	66.00	71.45	73.50	100.0	98.50	100.0	84.75
UE-3	95.83	95.00	100.0	51.17	98.83	96.67	100.0	95.67
ULE-4	76.25	94.75	76.25	96.25	98.00	96.25	100.0	96.25
Avg. (%)	73.60	74.39	83.33	70.52	89.27	89.73	94.95	87.37

Evaluation of the performance of clustering algorithms in kernel-induced feature space, Pattern Recognition, 2005

3 算法优化

3-4 Kernel k-means

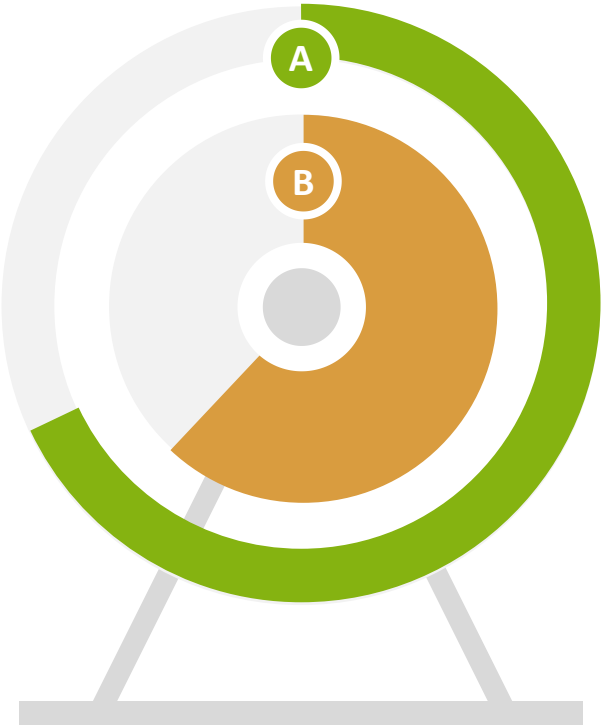
Clustering accuracy (%) achieved by each clustering algorithm for 10 data sets

Data set	Conventional				Kernel			
	k-means	FCM	Average	Mountain	*k*-means	FCM	Average	Mountain
BENSAID	79.59	73.47	100.0	85.71	83.67	93.88	100.0	100.0
DUNN	70.00	70.00	100.0	83.33	71.11	95.56	100.0	100.0
IRIS	89.33	89.33	90.67	52.67	96.00	93.33	89.33	93.33
ECOLI	42.86	49.11	76.49	51.19	68.75	61.01	77.38	69.05
CIRCLE	50.76	52.79	62.44	55.84	100.0	93.40	82.74	62.94
BLE-3	65.67	65.67	56.00	70.33	76.33	74.67	100.0	71.67
BLE-2	88.50	87.75	100.0	85.25	100.0	94.00	100.0	100.0
UE-4	77.25	66.00	71.45	73.50	100.0	98.50	100.0	84.75
UE-3	95.83	95.00	100.0	51.17	98.83	96.67	100.0	95.67
ULE-4	76.25	94.75	76.25	96.25	98.00	96.25	100.0	96.25
Avg. (%)	73.60	74.39	83.33	70.52	89.27	89.73	94.95	87.37

Evaluation of the performance of clustering algorithms in kernel-induced feature space, Pattern Recognition, 2005

3 算法优化

3-4 Kernel k-means



A **kernel kmeans**

适合线性不可分情况
复杂度高计算量大

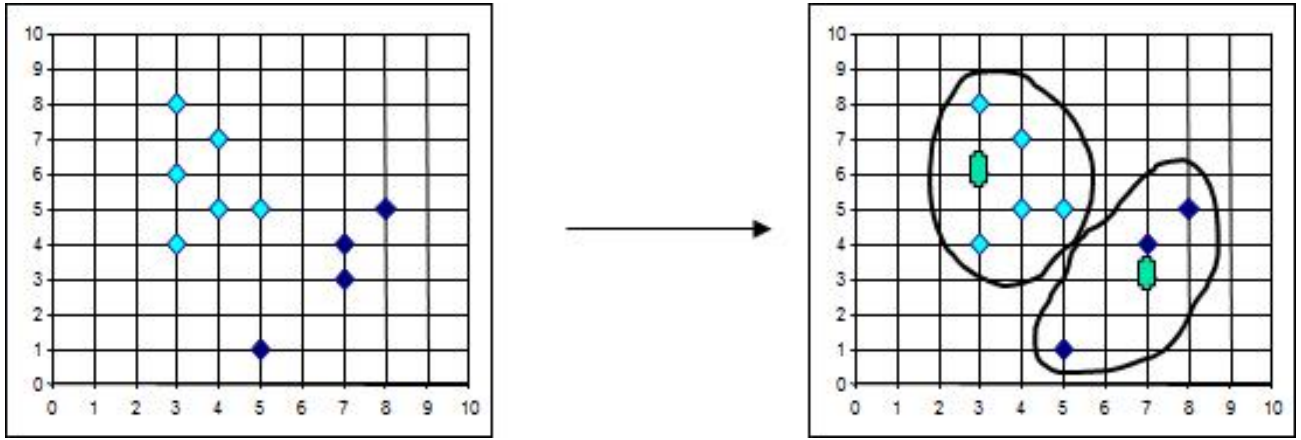
B **kmeans**

适合线性可分情况
复杂度低计算量小

3-5 k-medoids（k-中心聚类算法）

K-medoids和K-means是有区别的，**不一样的地方在于中心点的选取**：

- K-means中，将中心点取为当前cluster中所有数据点的平均值，对异常点很敏感!
- K-medoids中，将从当前cluster 中选取到其他所有（当前cluster中的）点的距离之和最小的点作为中心点。



3 算法优化

3-5 k-medoids（k-中心聚类算法）

K-medoids使用绝对差值和（Sum of Absolute Differences，SAD）的度量来衡量聚类结果的优劣，在n维空间中，计算SAD的公式如下所示：

$$\mathrm{SAD}=\sum_{m=1}^{k}\sum_{p_i\in C_i} dist(p_i,o_i)=\sum_{m=1}^{k}\sum_{p_i\in C_i}\sqrt{\sum_{j=1}^{n_{C_i}}(p_{ij}-o_{ij})^2}$$

3 算法优化

3-5 k-medoids（k-中心聚类算法）

算法流程：

（1）总体n个样本点中任意选取k个点作为medoids

（2）按照与medoids最近的原则，将剩余的n-k个点分配到当前最佳的medoids代表的类中

（3）对于第i个类中除对应medoids点外的所有其他点，按顺序计算当其为新的medoids时，代价函数的值，遍历所有可能，选取代价函数最小时对应的点作为新的medoids

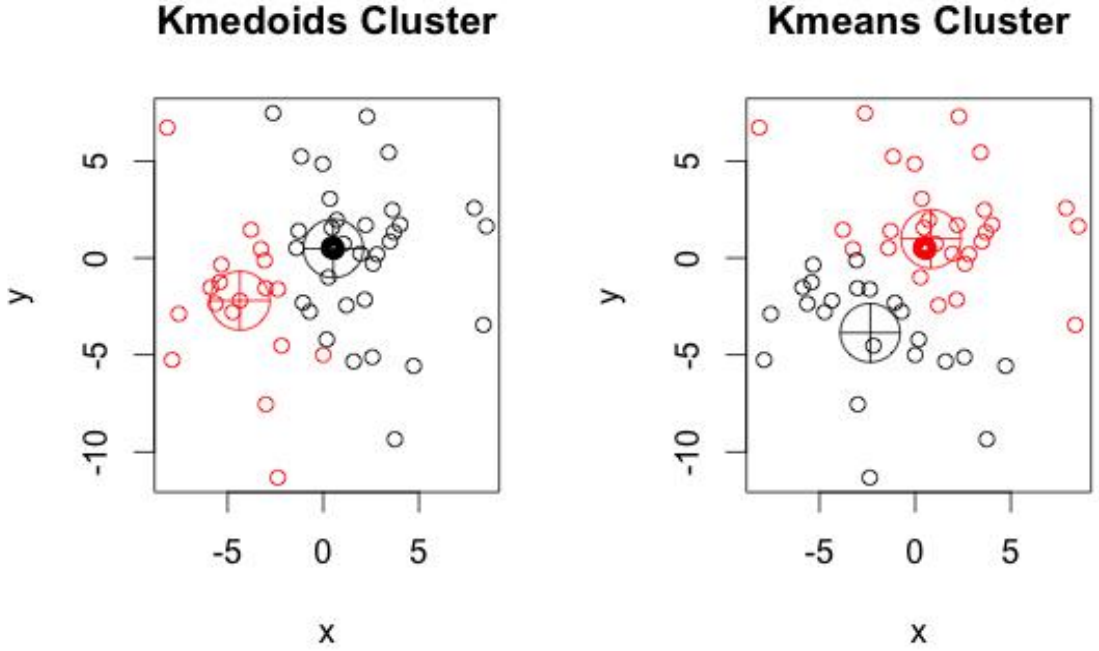
（4）重复2-3的过程，直到所有的medoids点不再发生变化或已达到设定的最大迭代次数

（5）产出最终确定的k个类

3 算法优化

3-5 k-medoids（k-中心聚类算法）

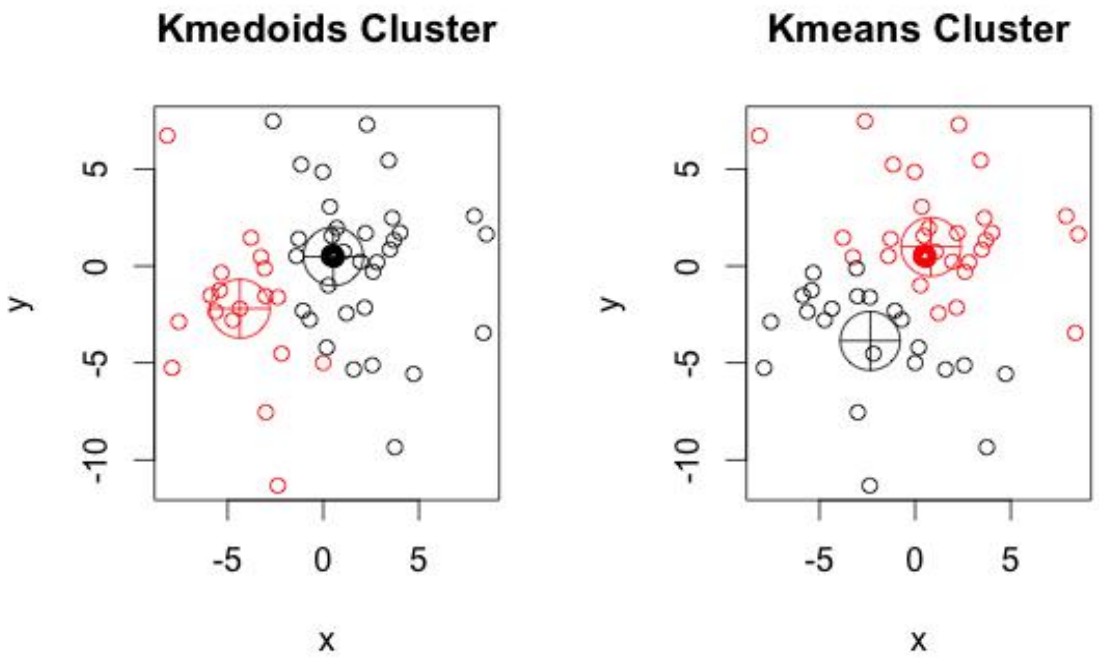
K-medoids较k-means运行结果有哪些区别呢?



3 算法优化

3-5 k-medoids（k-中心聚类算法）

K-medoids较k-means来说，多次运行每次结果偏差较小，而k-means对初值依赖大，导致每次运行结果相异程度大。



3 算法优化

3-5 k-medoids（k-中心聚类算法）

K-Means	K-Medoids
初始据点随机选取	初始随机据点限定在样本点中
使用Means(均值)作为聚点，对极值很敏感	使用Medoids(中位数)作为聚点
对数据要求高，要求数据点处于欧式空间中	可适用类别（categorical）类型的特征
时间复杂度：O(n*k*t)，t为迭代次数	时间复杂度：O(n^2 *k*t)，t为迭代次数
K-Means 算法对大规模数据集较高效	K-Medoids算法对小规模数据性能更好，但伸缩性较差
都有可能陷入局部最优解的困境之中	
K的含义相同,都需要开始人为设定簇数目	
都是无监督算法	

3 算法优化

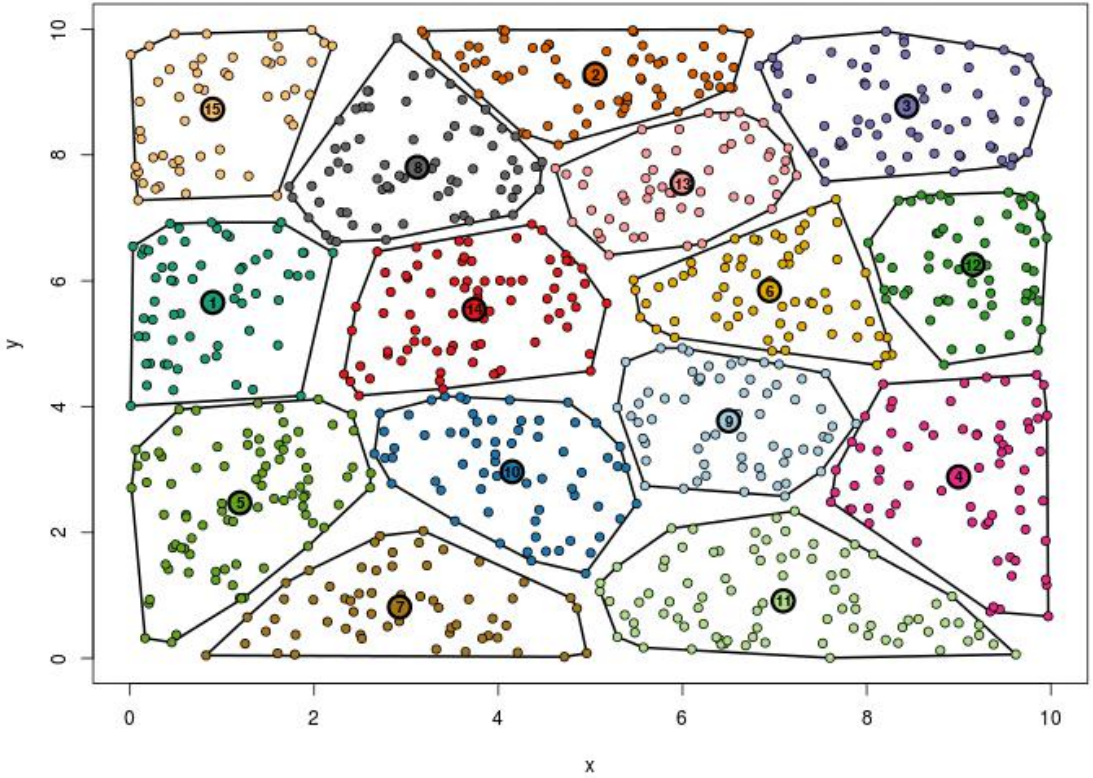
3-5 k-medoids（k-中心聚类算法）

k-medoids对噪声鲁棒性好。例：当一个cluster样本点只有少数几个，如（1,1）（1,2）（2,1）（1000,1000）。其中（1000,1000）是噪声。如果按照k-means质心大致会处在（1,1）（1000,1000）中间，这显然不是我们想要的。这时k-medoids就可以避免这种情况，他会在（1,1）（1,2）（2,1）（1000,1000）中选出一个样本点使cluster的绝对误差最小，计算可知一定会在前三个点中选取。

k-medoids只能对小样本起作用，样本大，速度就太慢了，当样本多的时候，少数几个噪音对k-means的质心影响也没有想象中的那么重，所以k-means的应用明显比k-medoids多。

3 算法优化

3-6 ISODATA



3 算法优化

3-6 ISODATA

- 类别数目随着聚类过程而变化；
- 对类别数的“合并”：（当聚类结果某一类中样本数太少，或两个类间的距离太近时）
- “分裂”（当聚类结果中某一类的类内方差太大，将该类进行分裂）

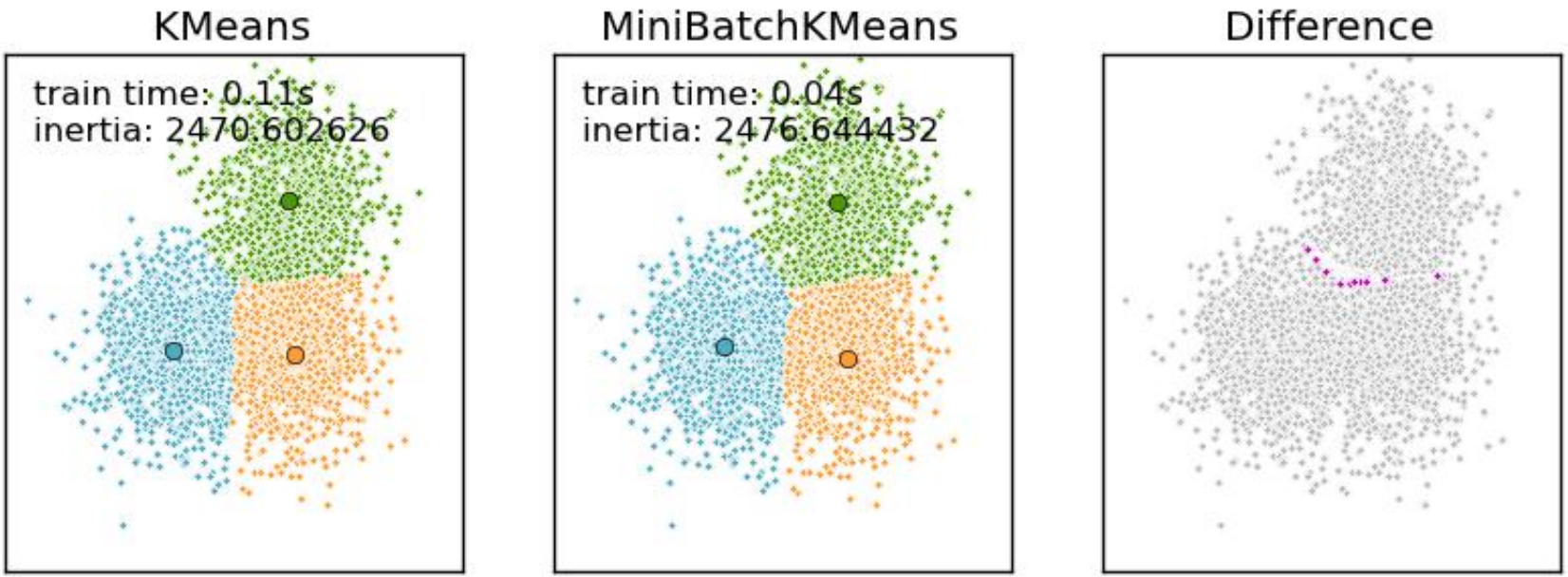
3-7 Mini Batch K-Means（适合大数据的聚类算法）

大数据量是什么量级？通过当样本量大于1万做聚类时，就需要考虑选用Mini Batch K-Means算法。

Mini Batch KMeans使用了Mini Batch（分批处理）的方法对数据点之间的距离进行计算。Mini Batch计算过程中不必使用所有的数据样本，而是从不同类别的样本中抽取一部分样本来代表各自类型进行计算。由于计算样本量少，所以会相应的减少运行时间，但另一方面抽样也必然会带来准确度的下降。

3 算法优化

3-7 Mini Batch K-Means（适合大数据的聚类算法）



3 算法优化

3-7 Mini Batch K-Means（适合大数据的聚类算法）

该算法的迭代步骤有两步：

(1)从数据集中随机抽取一些数据形成小批量，把他们分配给最近的质心

(2)更新质心

与Kmeans相比，数据的更新在每一个小的样本集上。对于每一个小批量，通过计算平均值得到更新质心，并把小批量里的数据分配给该质心，随着迭代次数的增加，这些质心的变化是逐渐减小的，直到质心稳定或者达到指定的迭代次数，停止计算。

小结

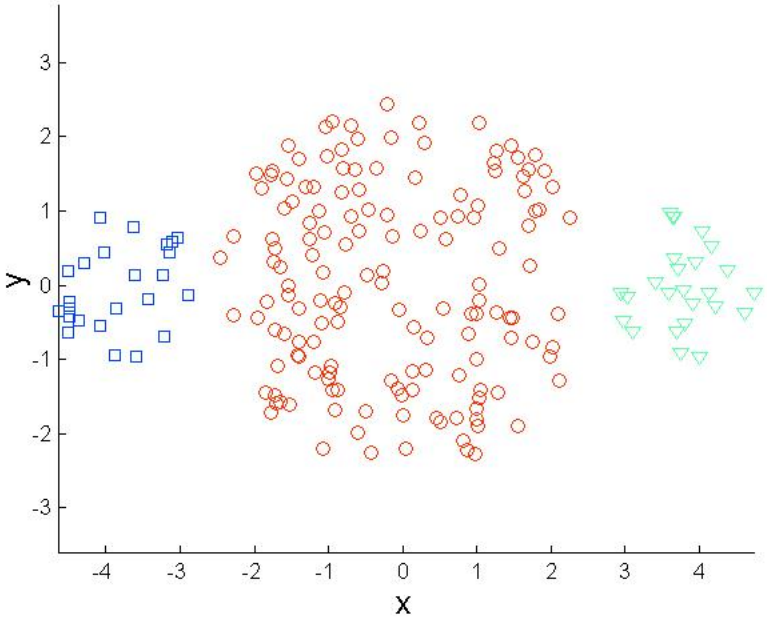
优化方法	思路
Canopy+kmeans	Canopy粗聚类配合kmeans
kmeans++	距离越远越容易成为新的质心
二分k-means	拆除SSE最大的簇
ISODATA	动态聚类
kernel kmeans	映射到高维空间
Mini-batch K-Means	大数据集分批聚类

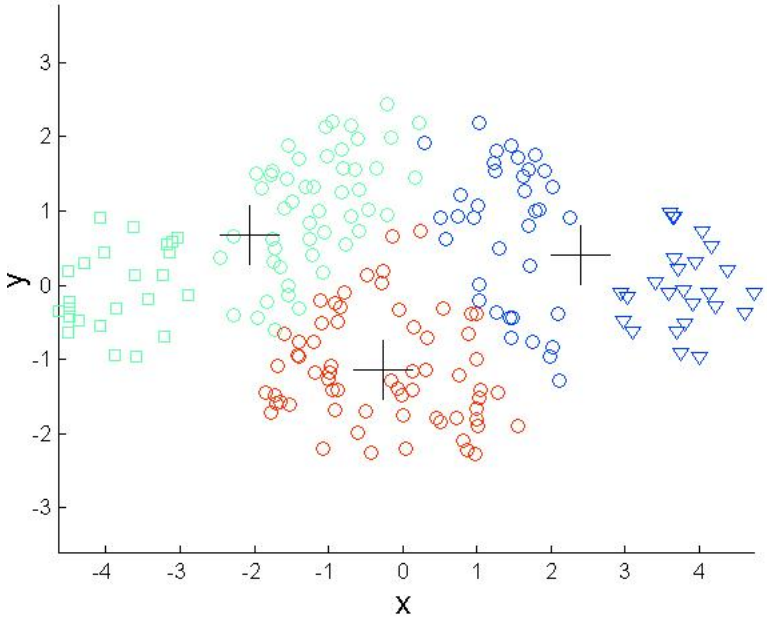
黑马程序员
www.itheima.com
传智播客旗下高端IT教育品牌

目录 Contents

4 算法进阶

4-1 DBSCAN(基于密度聚类)

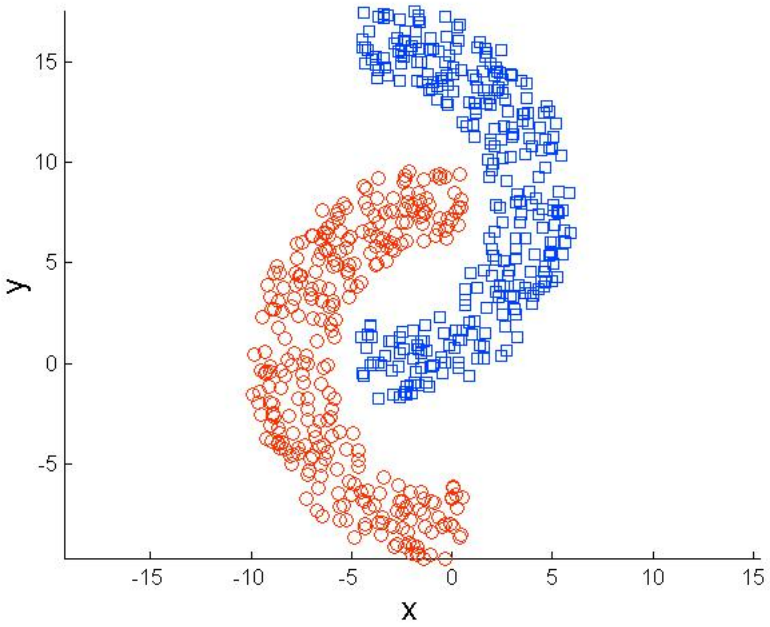


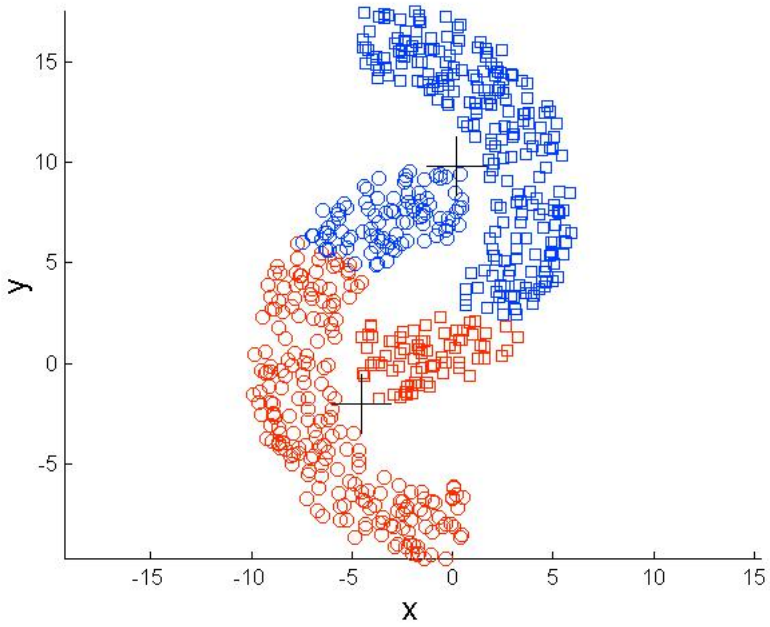


https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

4 算法进阶

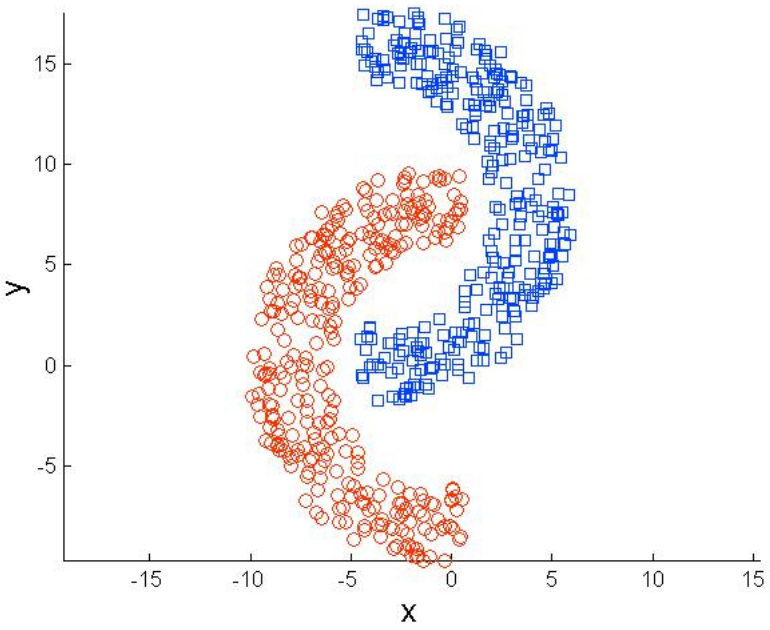
4-1 DBSCAN(基于密度聚类)

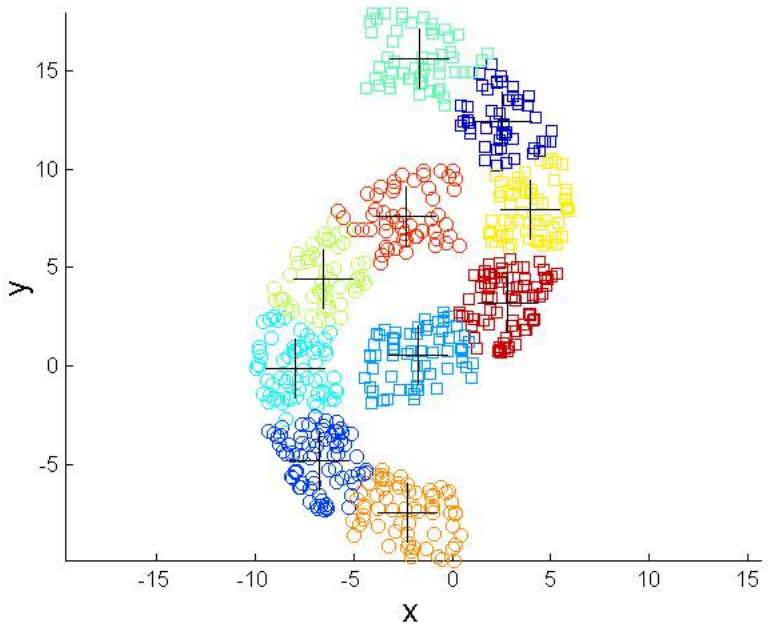




4 算法进阶

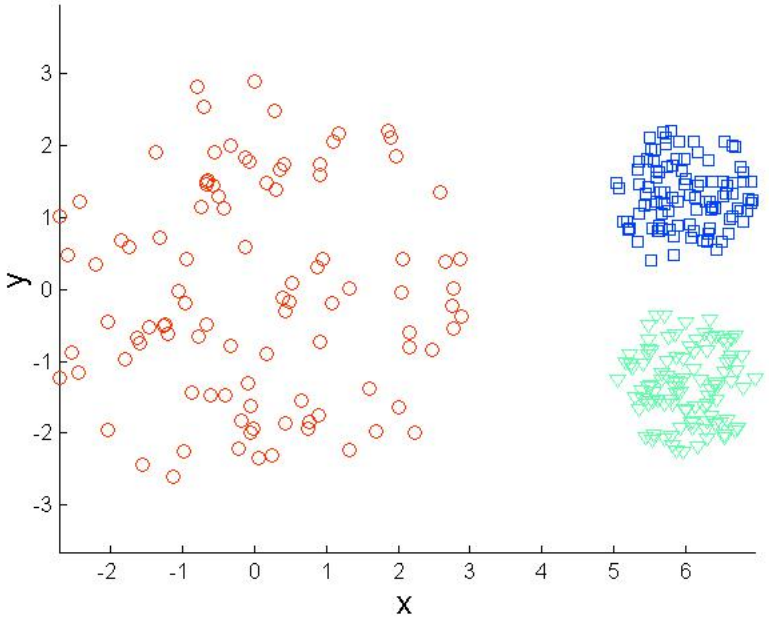
4-1 DBSCAN(基于密度聚类)

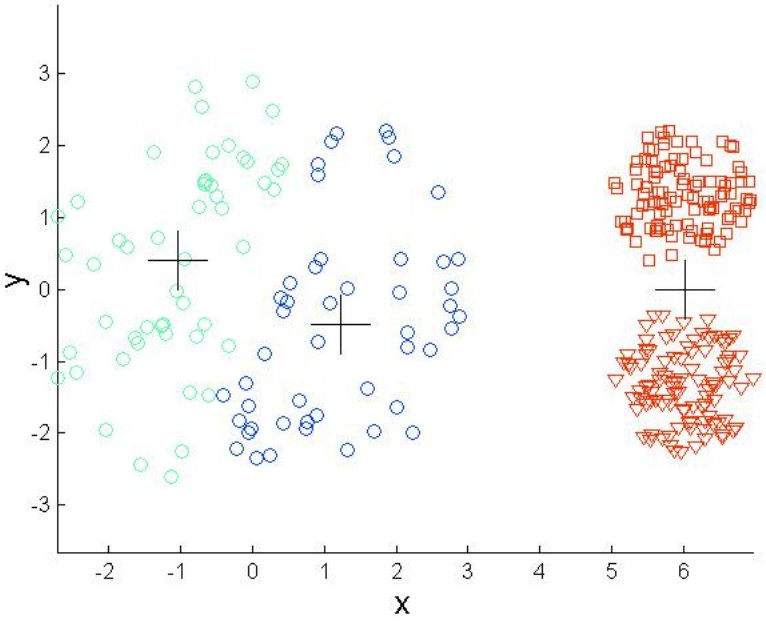




4 算法进阶

4-1 DBSCAN(基于密度聚类)

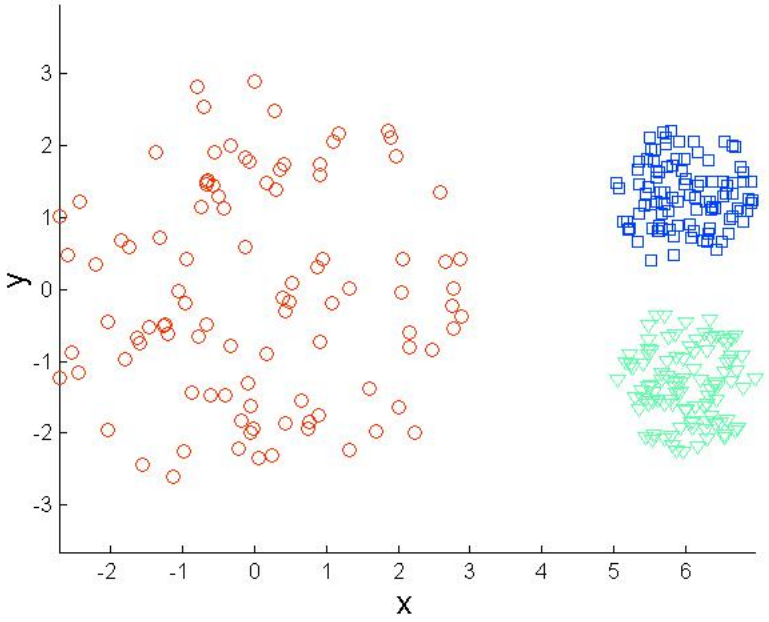


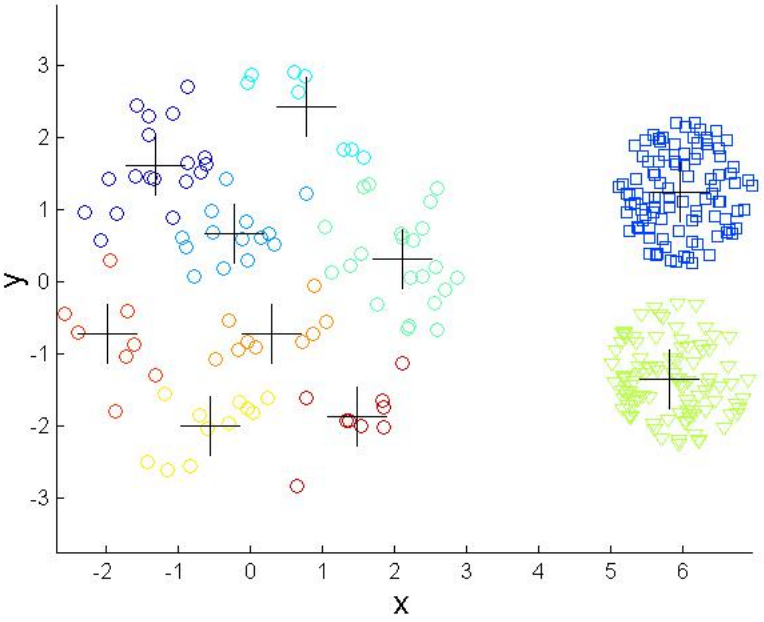


https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

4 算法进阶

4-1 DBSCAN(基于密度聚类)





4 算法进阶

4-1 DBSCAN(基于密度聚类)

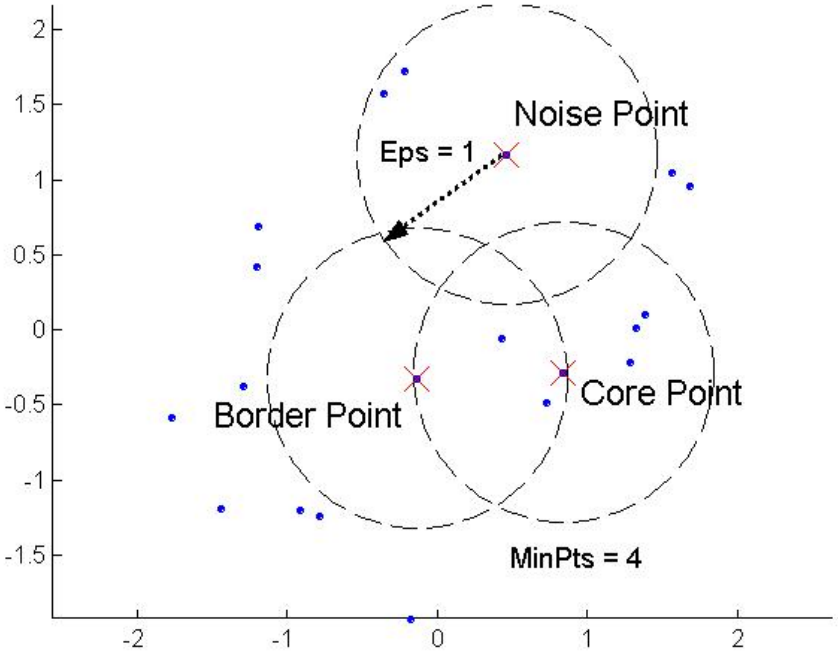
参数1

半径（Eps），表示以给定点P为中心的圆形邻域的范围；

参数2

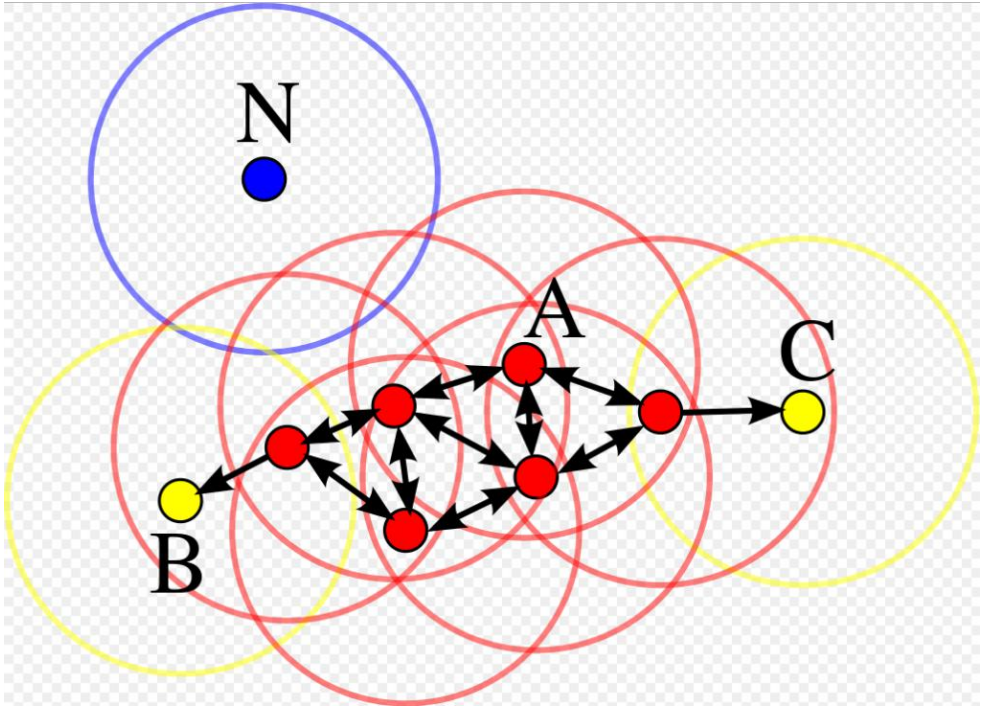
以点P为中心的邻域内最少点的数量（MinPts）。

如果满足：以点P为中心、半径为Eps的邻域内的点的个数不少于MinPts，则称点P为核心点。

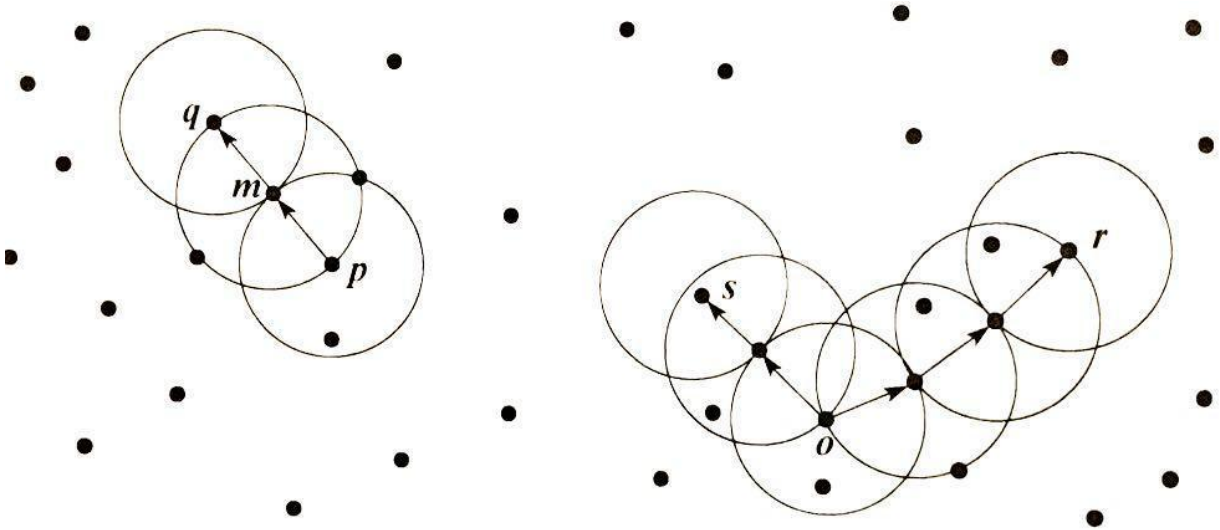


4 算法进阶

4-1 DBSCAN(基于密度聚类)



4 算法进阶



由上图可看出m,p,o.r 都是核心对象，因为他们的内都只是包含3个对象。

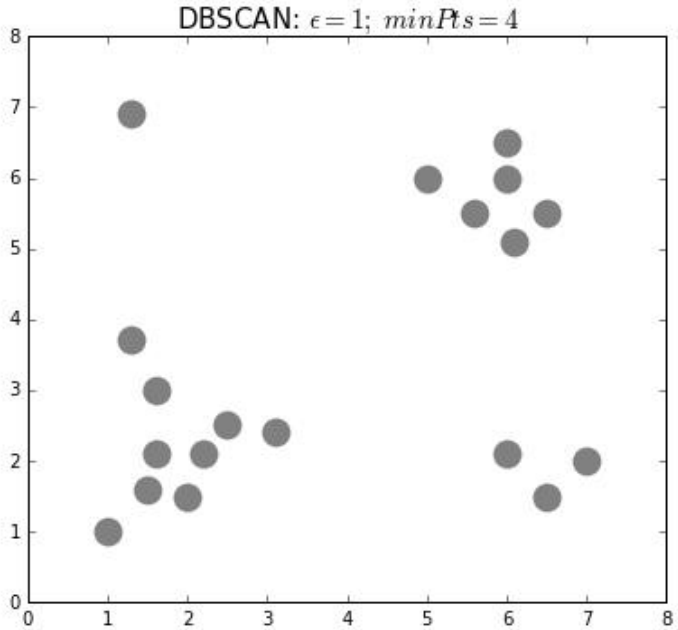
（1） 对象q是从m直接**密度可达**的。对象m从p直接密度可达的。

（2）对象q是从p（间接）密度可达的,因为q从m直接密度可达，m从p直接密度可达。

（3）r和s是从o密度可达的，而o是从r密度可达的，所有o,r和s都是密度相连的。

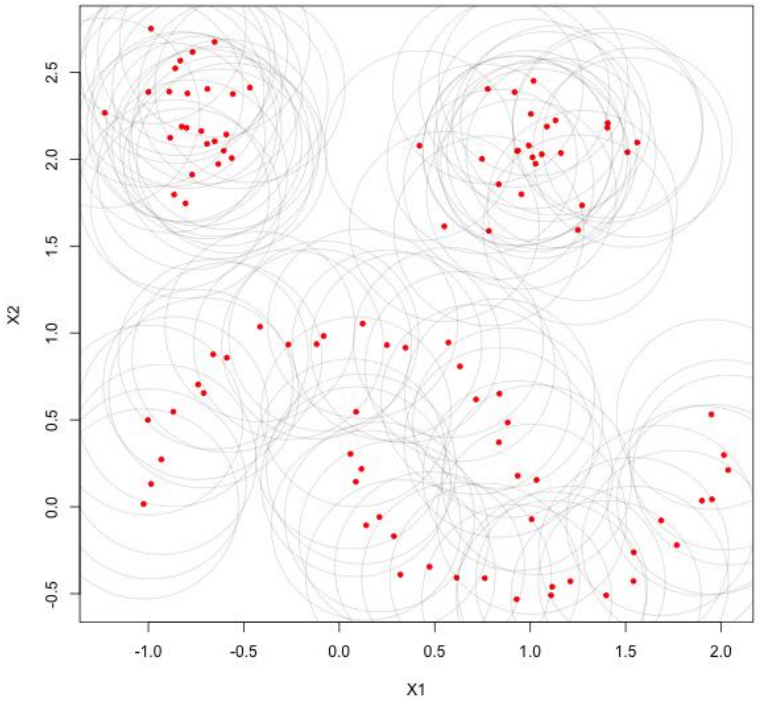
4 算法进阶

4-1 DBSCAN(基于密度聚类)



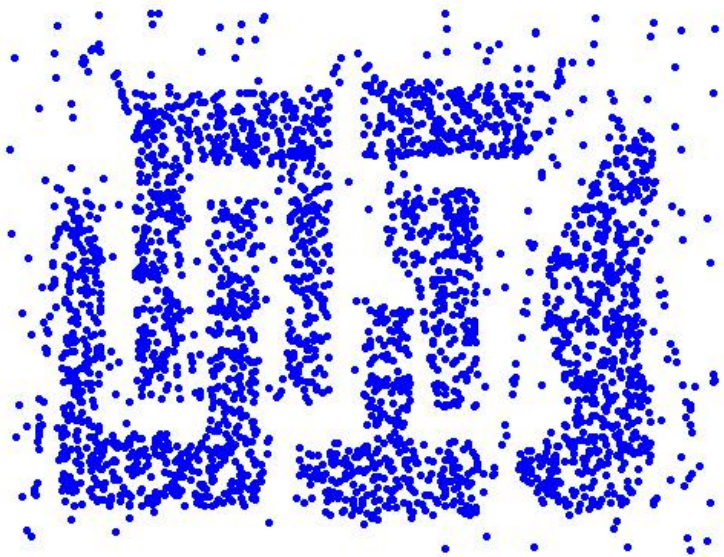
4 算法进阶

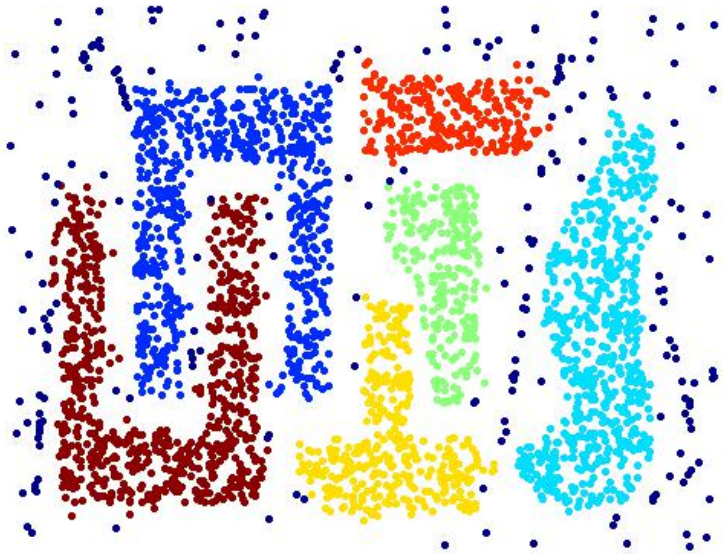
4-1 DBSCAN(基于密度聚类)



4 算法进阶

4-1 DBSCAN(基于密度聚类)

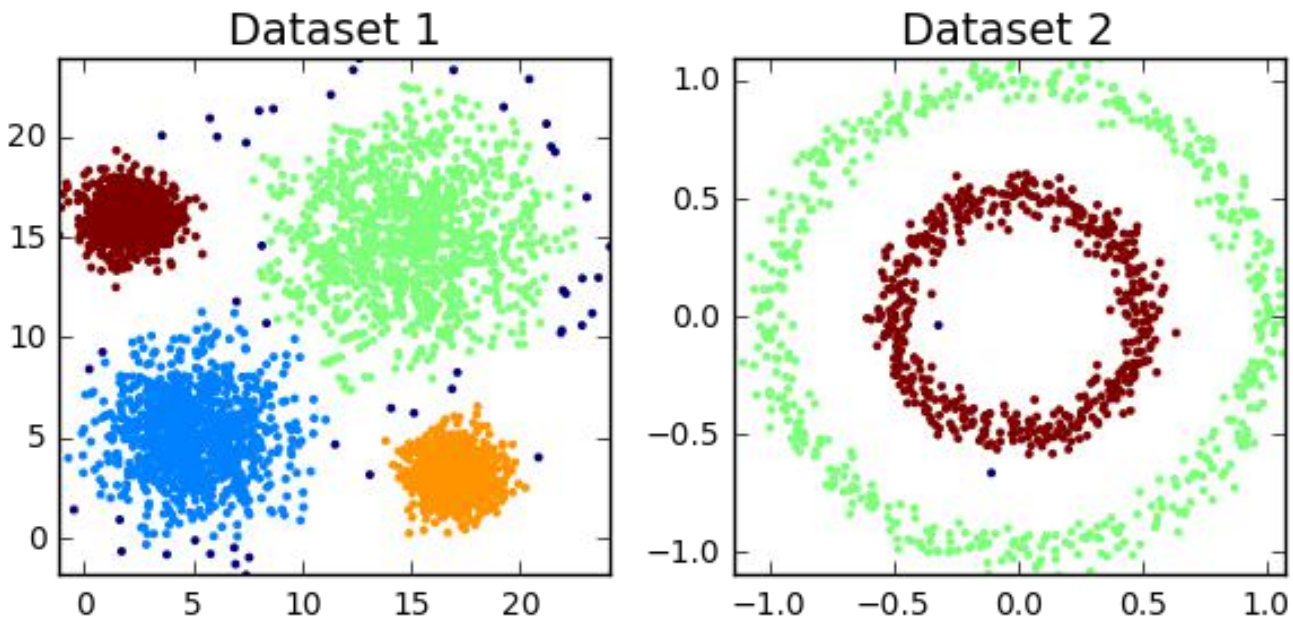




- 抗噪性好
- 对聚类形状捕捉敏锐

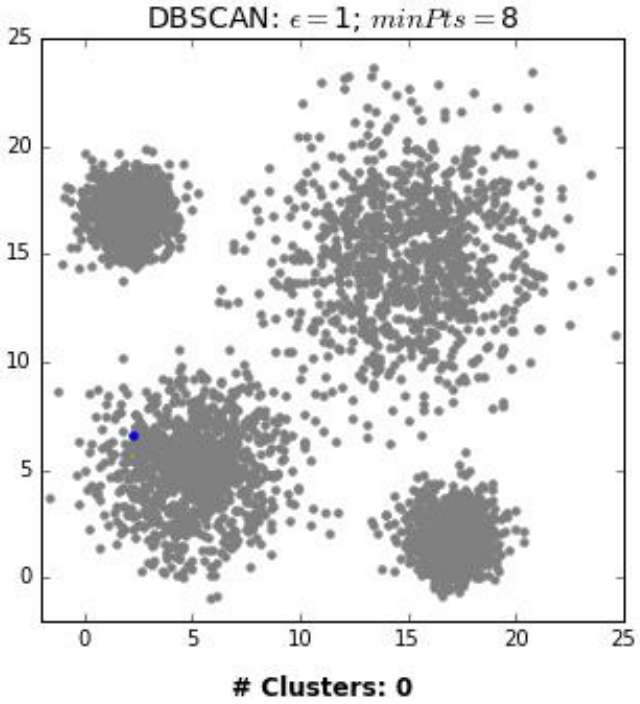
4 算法进阶

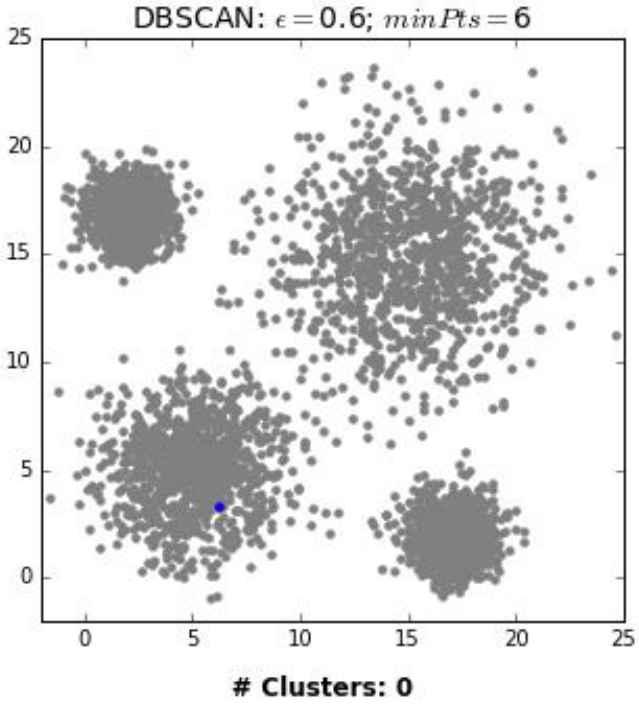
4-1 DBSCAN(基于密度聚类)



4 算法进阶

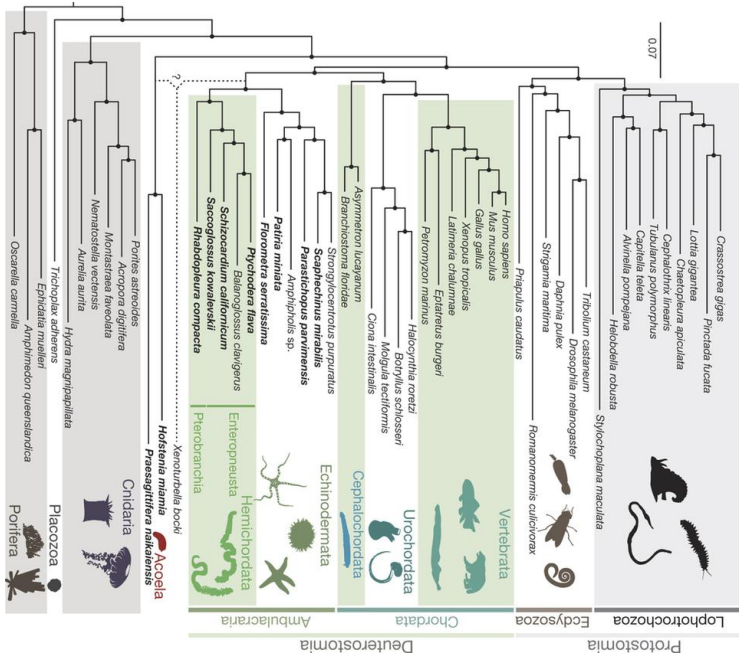
4-1 DBSCAN(基于密度聚类)

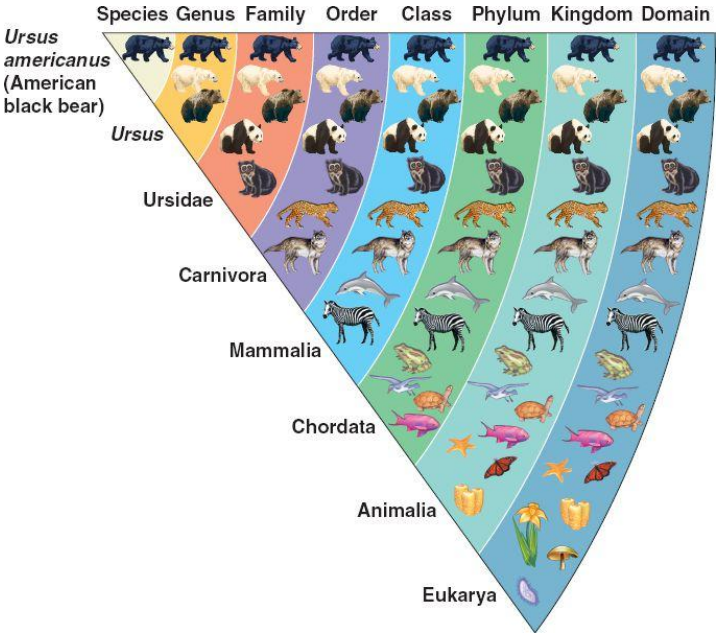




4 算法进阶

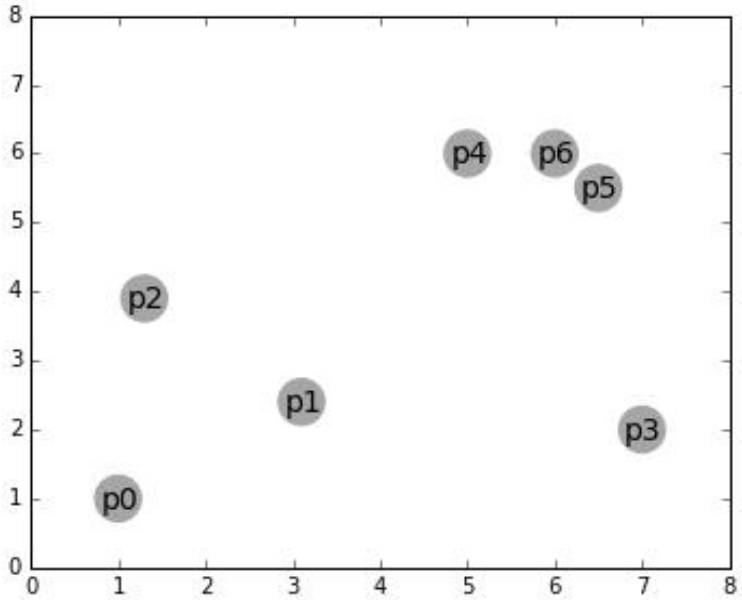
4-2 层次聚类

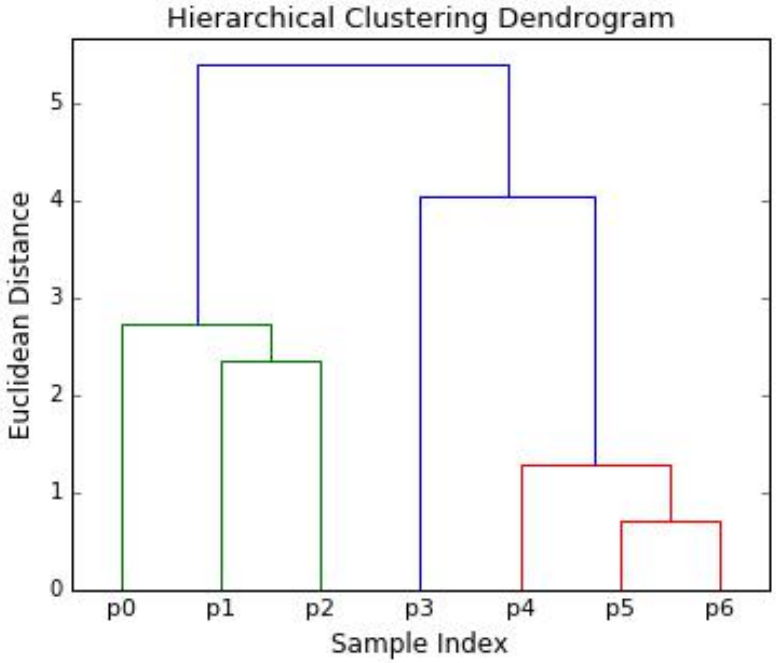




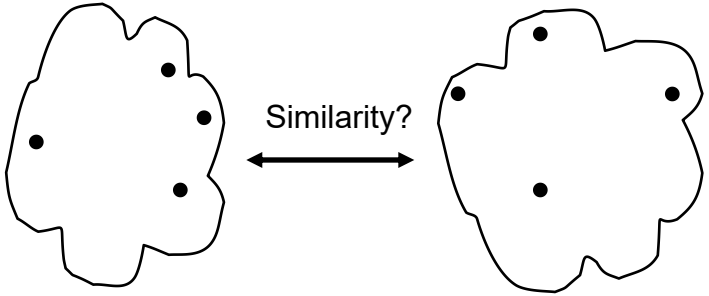
4 算法进阶

4-2 层次聚类





4-2 层次聚类



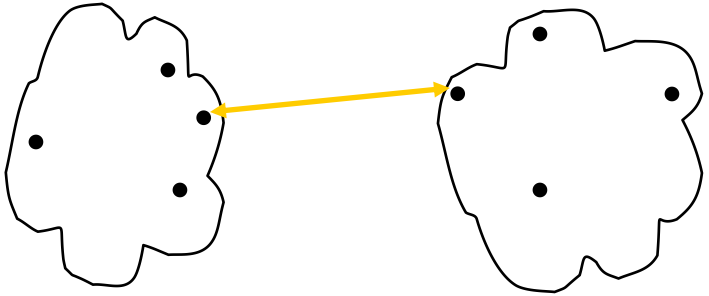
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						

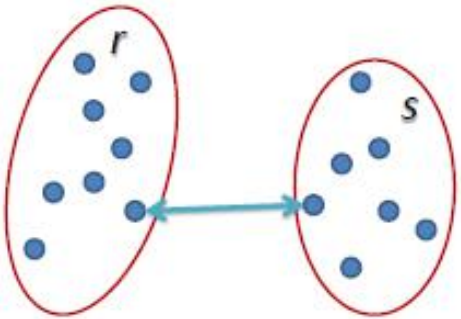
Proximity Matrix

- MIN度量
- MAX度量
- Group Average（平均距离度量）
- Distance Between Centroids（质心度量）
- Other methods driven by an objective function
  - Ward’s Method uses squared error

4 算法进阶

4-2 层次聚类





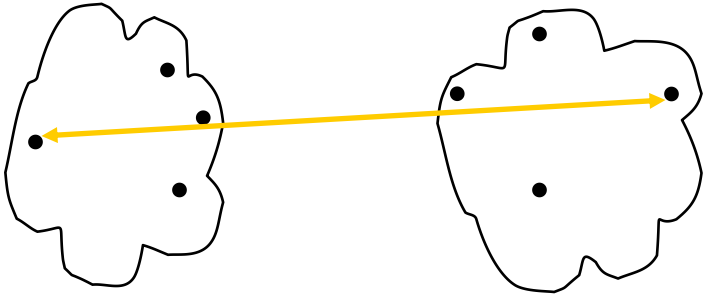
$$L(r,s)=\min(D(x_{ri},x_{sj}))$$

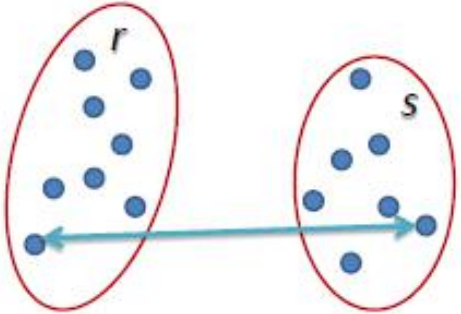
- MIN度量
- MAX度量
- Group Average（平均距离度量）
- Distance Between Centroids（质心度量）
- Other methods driven by an objective function

  -Ward's Method uses squared error

4 算法进阶

4-2 层次聚类



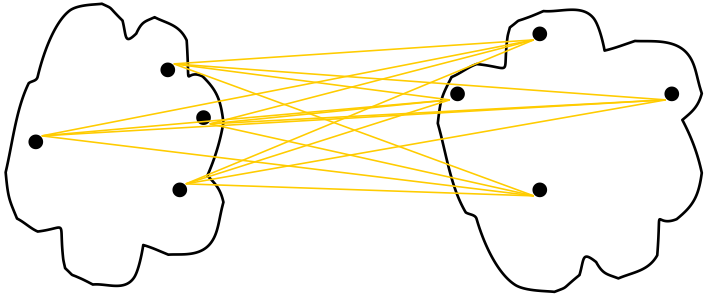


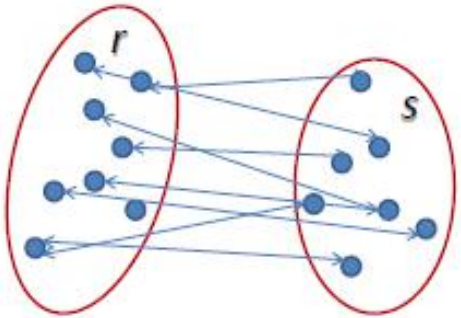
$$L(r,s)=\max(D(x_{ri},x_{sj}))$$

- MIN度量
- MAX度量
- Group Average（平均距离度量）
- Distance Between Centroids（质心度量）
- Other methods driven by an objective function
  - Ward’s Method uses squared error

4 算法进阶

4-2 层次聚类



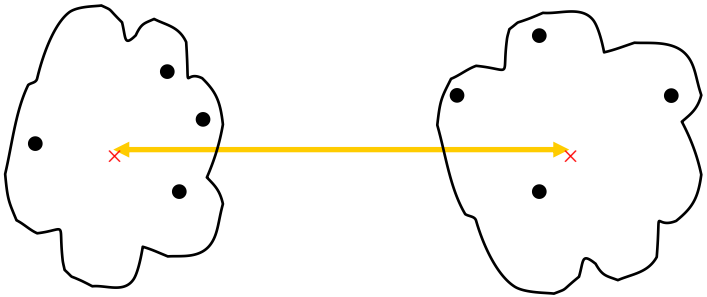


$$L(r,s)=\frac{1}{n_r n_s}\sum_{i=1}^{n_r}\sum_{j=1}^{n_s}D(x_{ri},x_{sj})$$

- MIN度量
- MAX度量
- Group Average（平均距离度量）
- Distance Between Centroids（质心度量）
- Other methods driven by an objective function
  - Ward's Method uses squared error

4 算法进阶

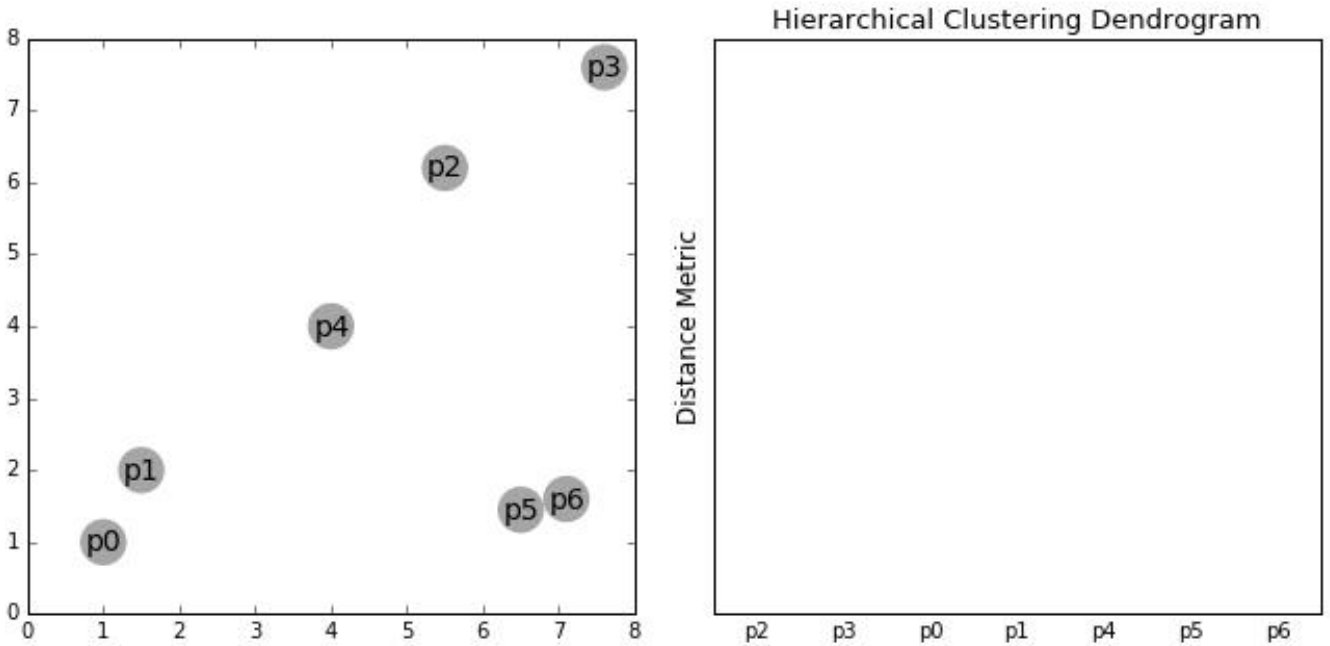
4-2 层次聚类



- MIN度量
- MAX度量
- Group Average（平均距离度量）
- Distance Between Centroids（质心度量）
- Other methods driven by an objective function
  - Ward’s Method uses squared error

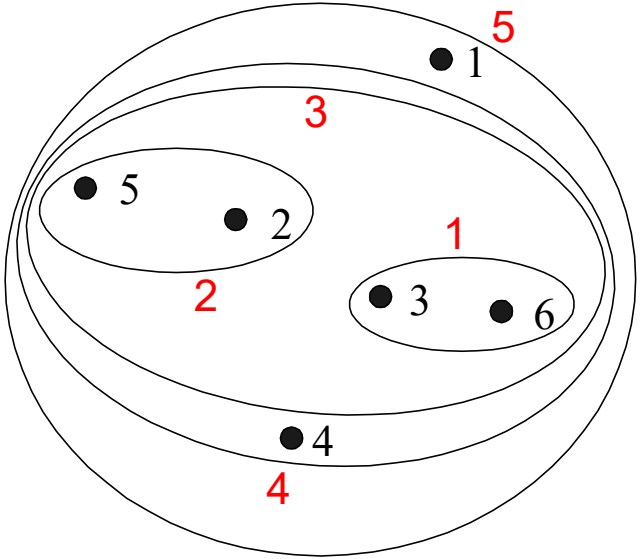
4 算法进阶

4-2 层次聚类

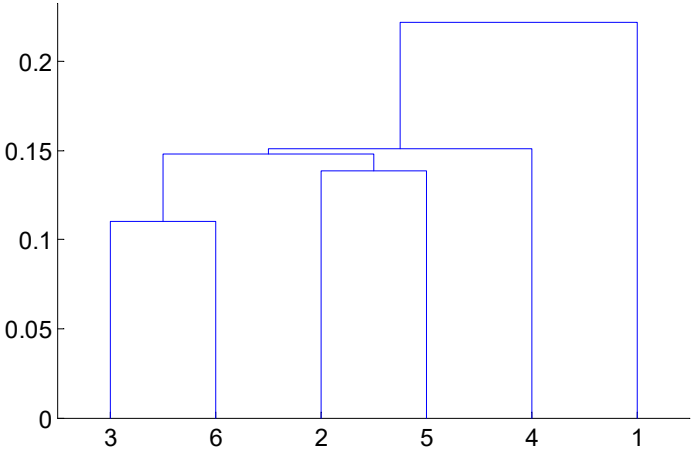


Hierarchical clustering is heavily influenced by two factors...

4-2 层次聚类



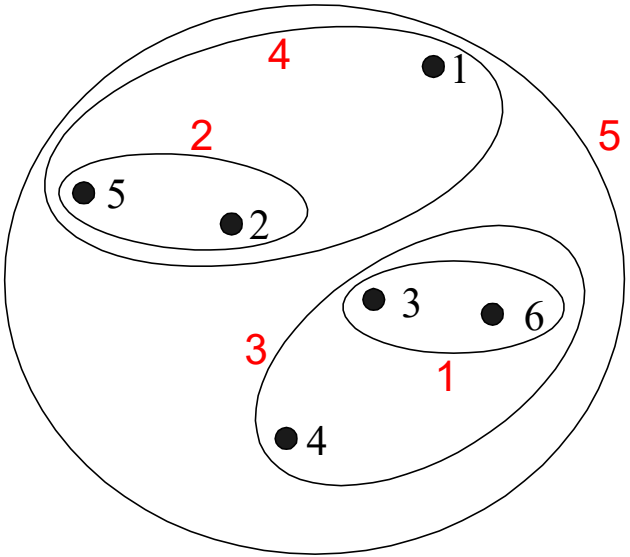
Nested Clusters



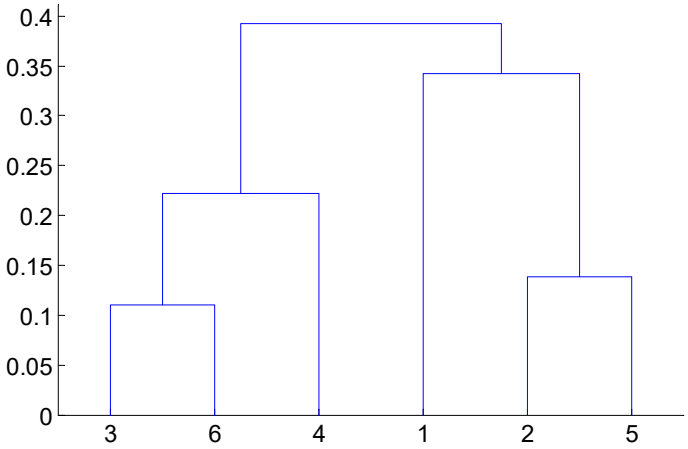
Dendrogram

4 算法进阶

4-2 层次聚类

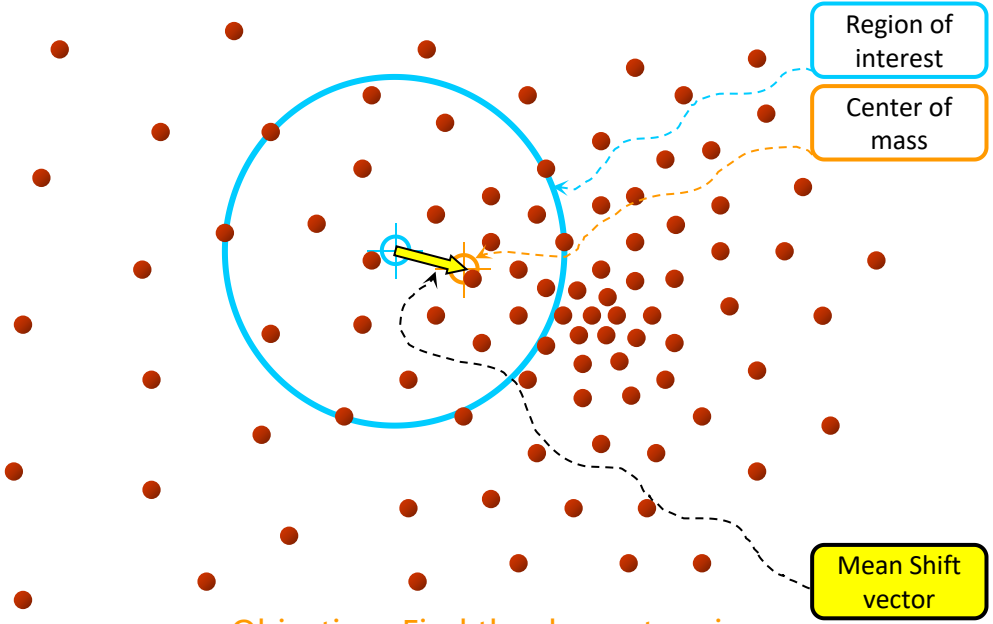


Nested Clusters



Dendrogram

4-3 Mean Shift聚类

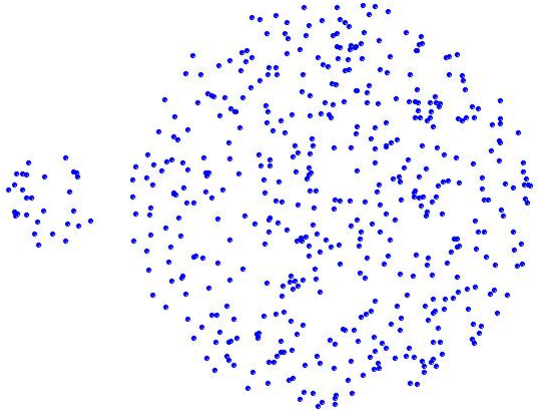


Objective : Find the densest region

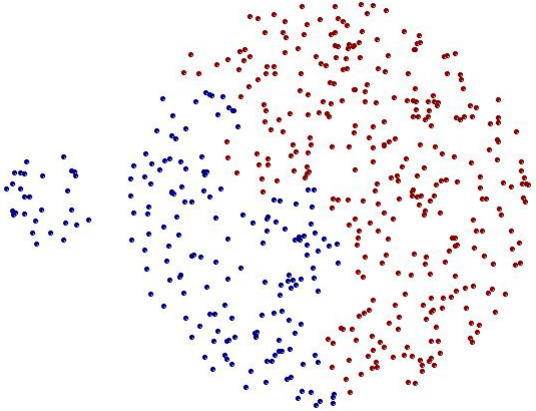
Distribution of identical billiard balls

4 算法进阶

4-2 层次聚类



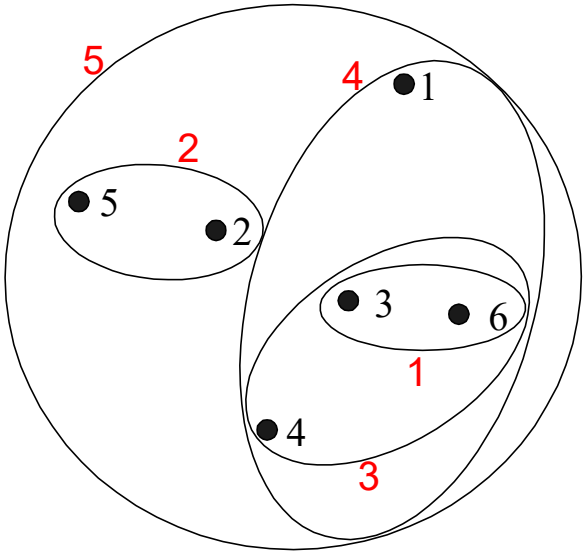
Original Points



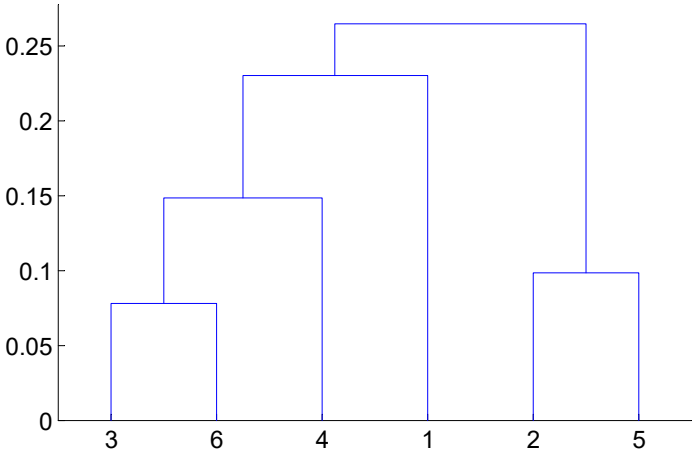
Two Clusters

- Tends to break large clusters
- Biased towards globular clusters

4-2 层次聚类



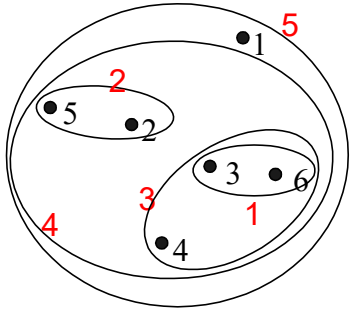
Nested Clusters



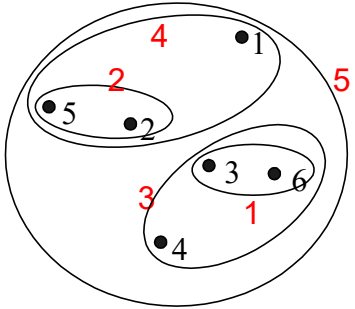
Dendrogram

4 算法进阶

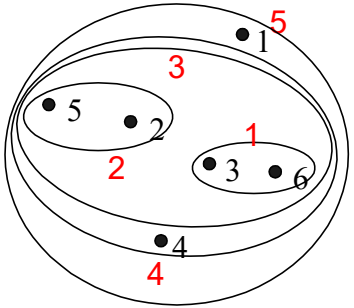
4-2 层次聚类



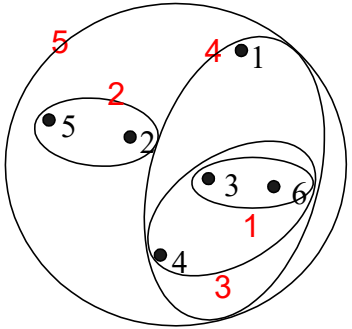
MIN



MAX

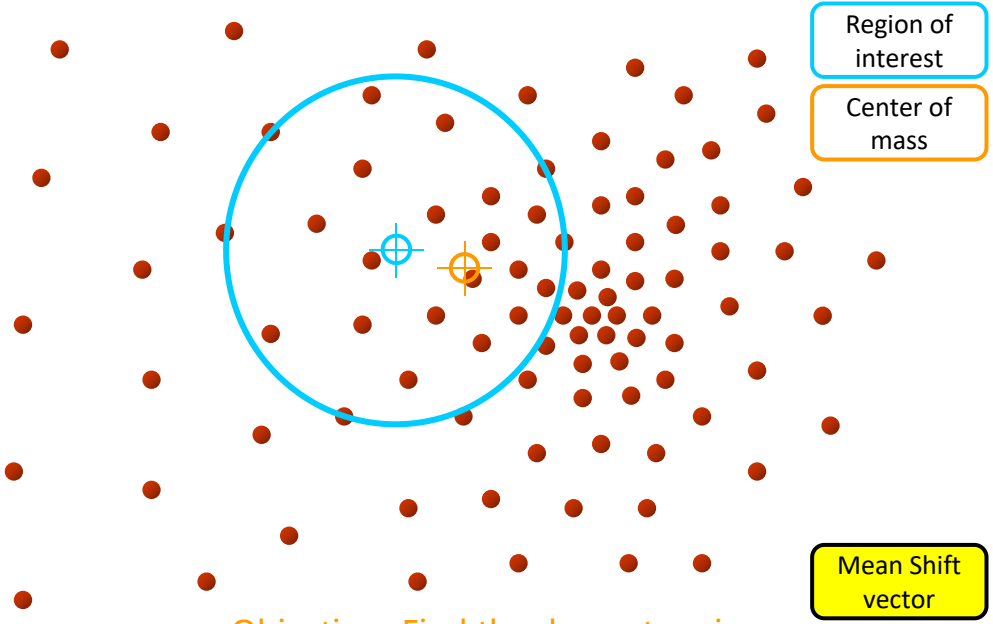


Group Average



Ward's Method

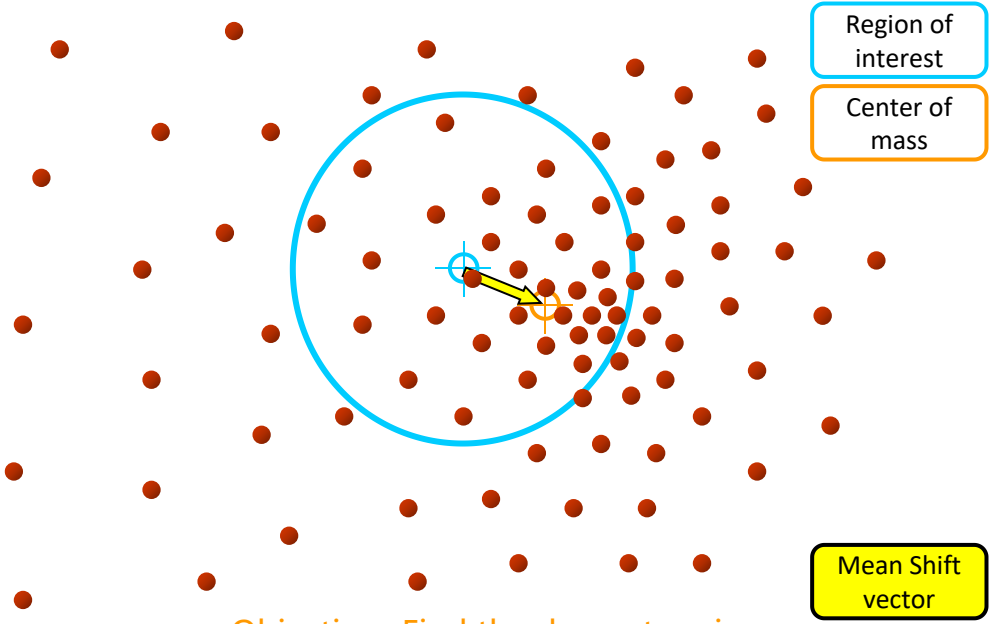
4-3 Mean Shift聚类



Objective : Find the densest region

Distribution of identical billiard balls

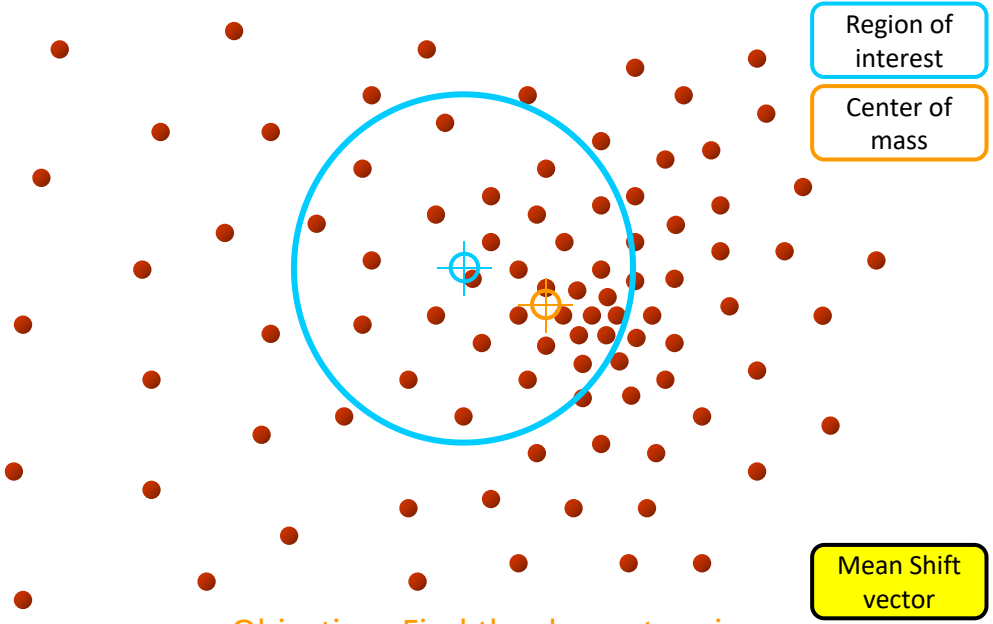
4-3 Mean Shift聚类



Objective : Find the densest region

Distribution of identical billiard balls

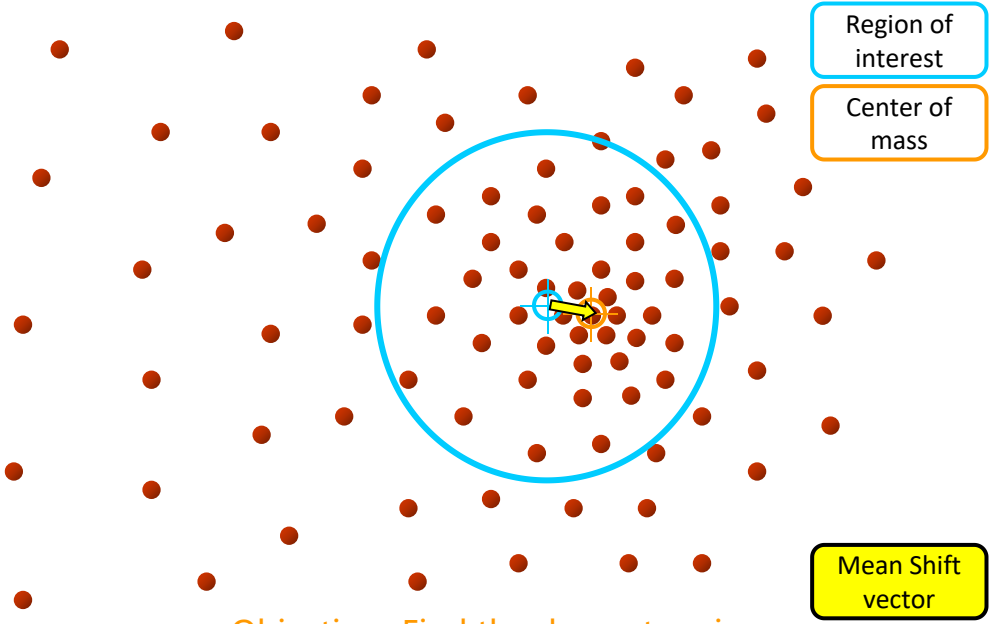
4-3 Mean Shift聚类



Objective : Find the densest region

Distribution of identical billiard balls

4-3 Mean Shift聚类

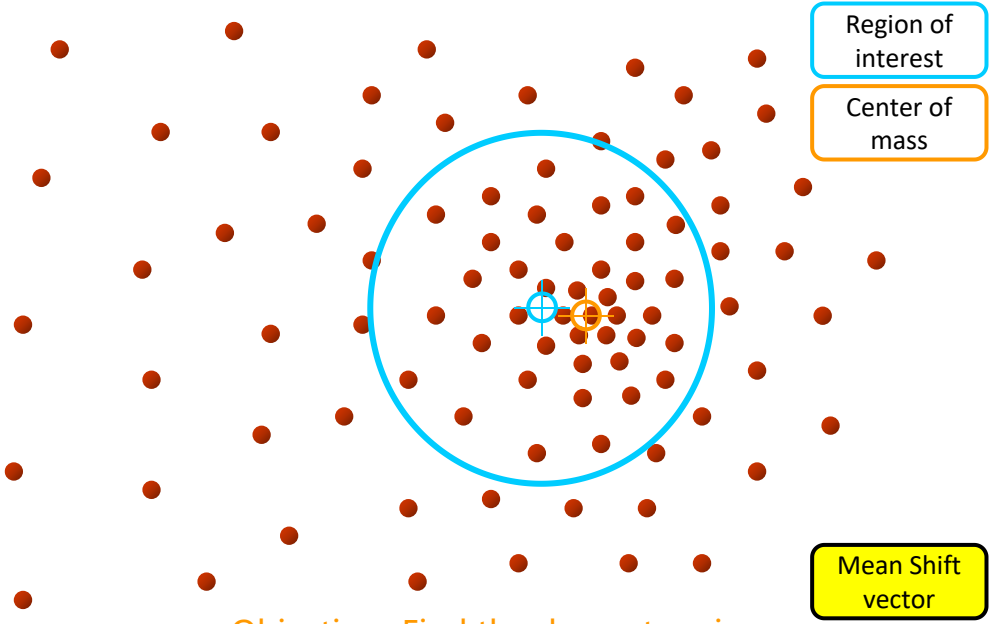


Objective : Find the densest region

Distribution of identical billiard balls

4 算法进阶

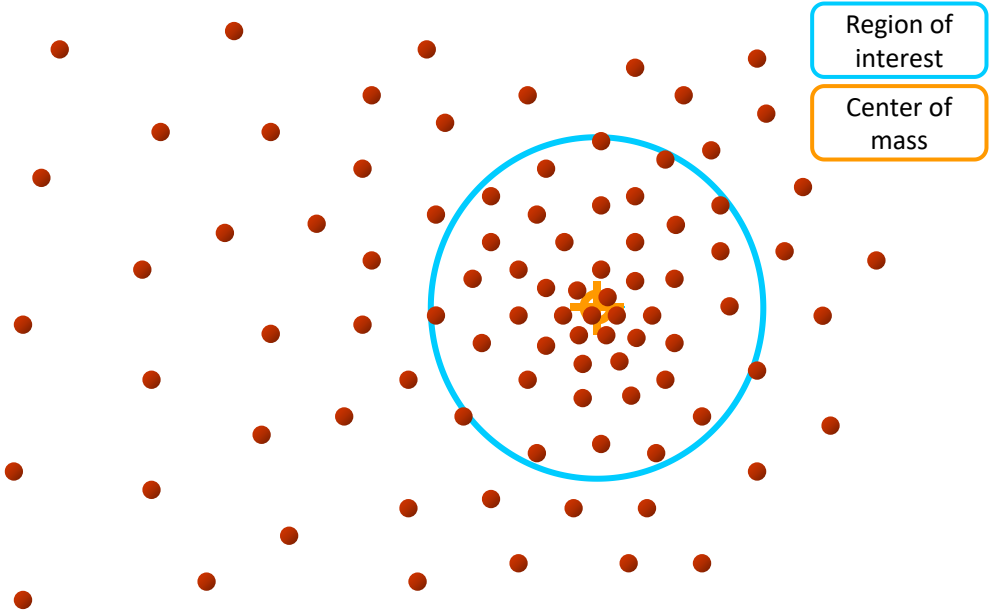
4-3 Mean Shift聚类



Objective : Find the densest region

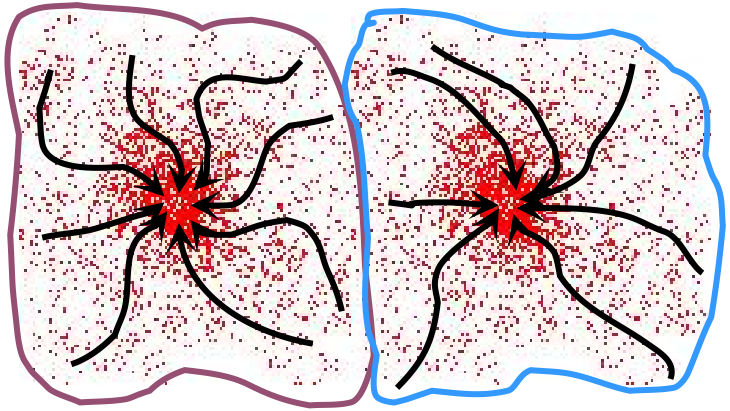
Distribution of identical billiard balls

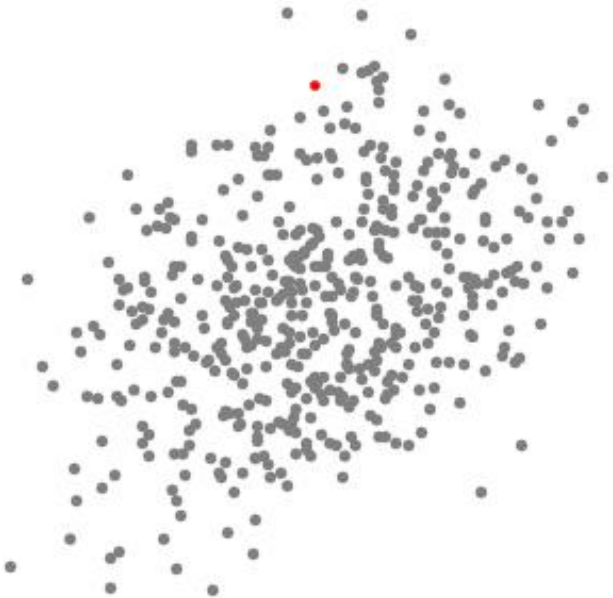
4-3 Mean Shift聚类



4-3 Mean Shift聚类

利用Mean Shift进行跟踪已经相当成熟,通过已知的图像帧中的目标位置找到目标在下一帧中的位置。在一定条件下，MeanShift算法能收敛到局部最优点，从而实现对运动体准确地定位。

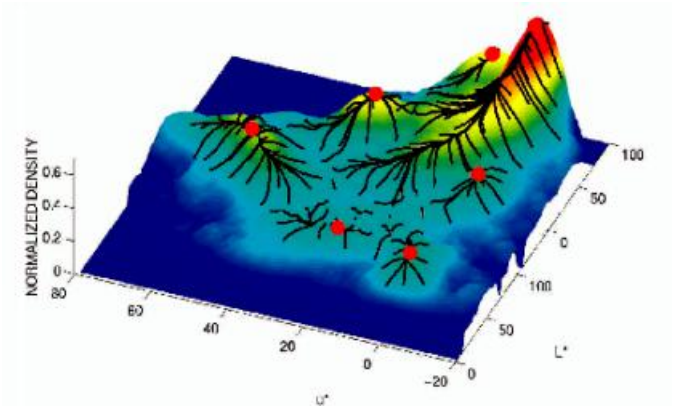


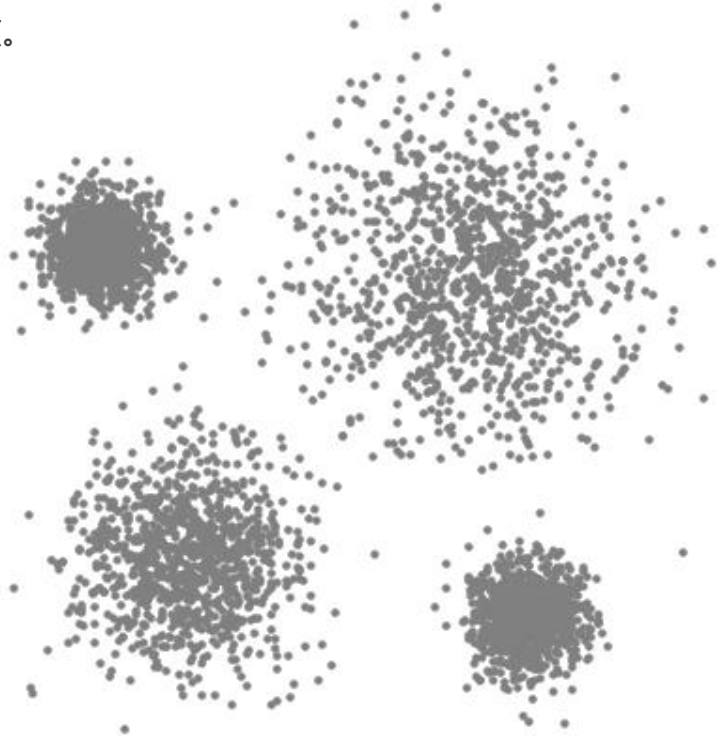


4-3 Mean Shift聚类

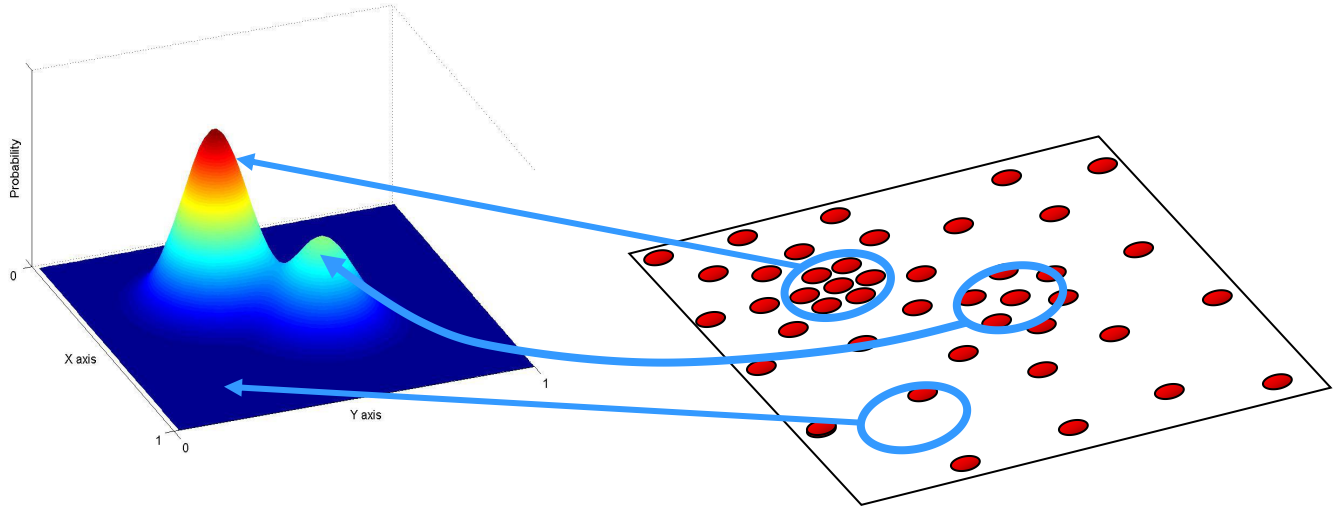
mean shift就是沿着密度上升的方向寻找同属一个簇的数据点。

球的中心被反复地推向高密度区域。这个过程重复进行，直到球表现出很小的运动。当多个球重叠时，包含最多点的球被保留。然后根据它们的球对观察结果进行聚类。

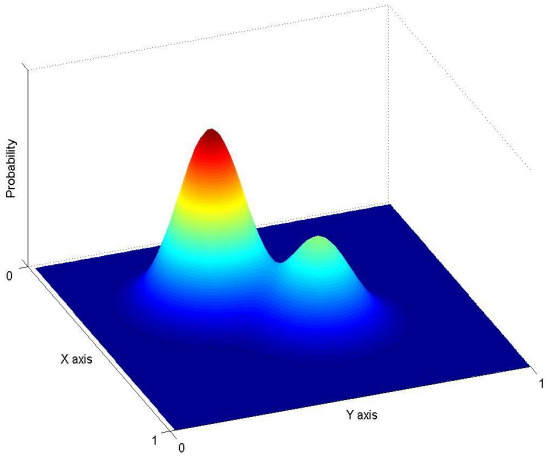




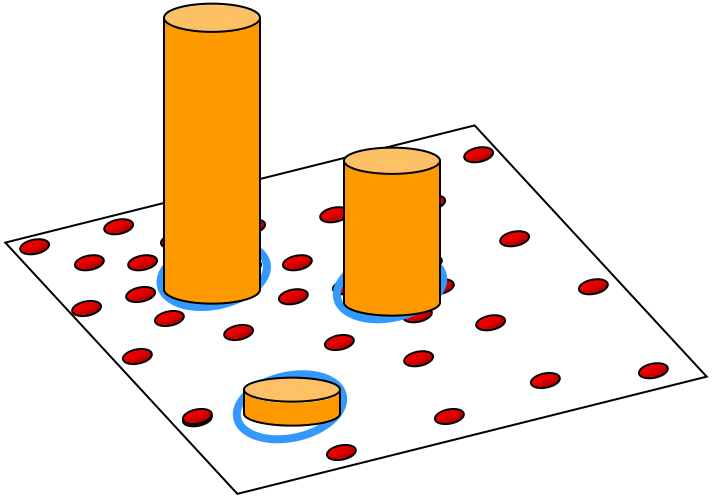
4-3 Mean Shift聚类



4-3 Mean Shift聚类

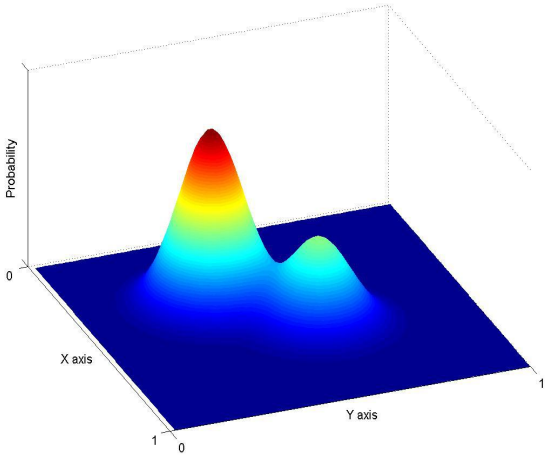


Assumed Underlying PDF

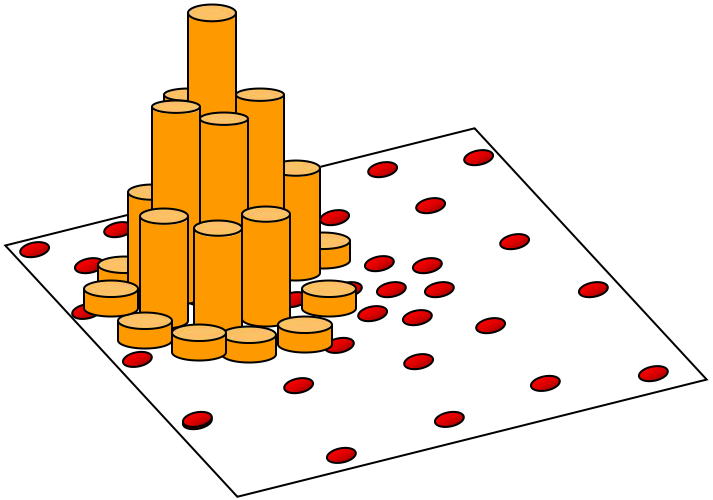


Real Data Samples

4-3 Mean Shift聚类

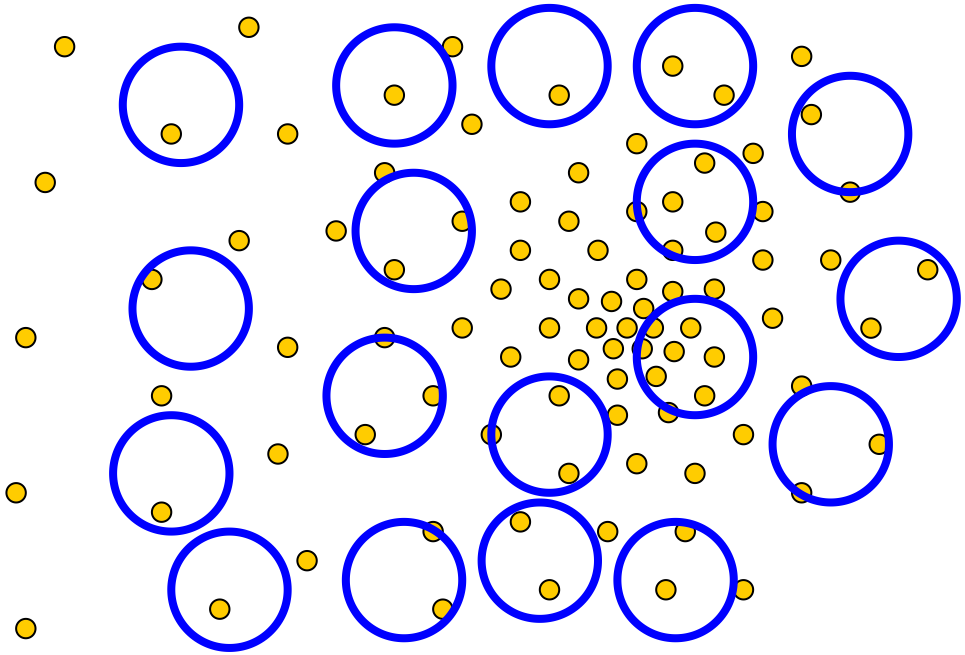


Assumed Underlying PDF



Real Data Samples

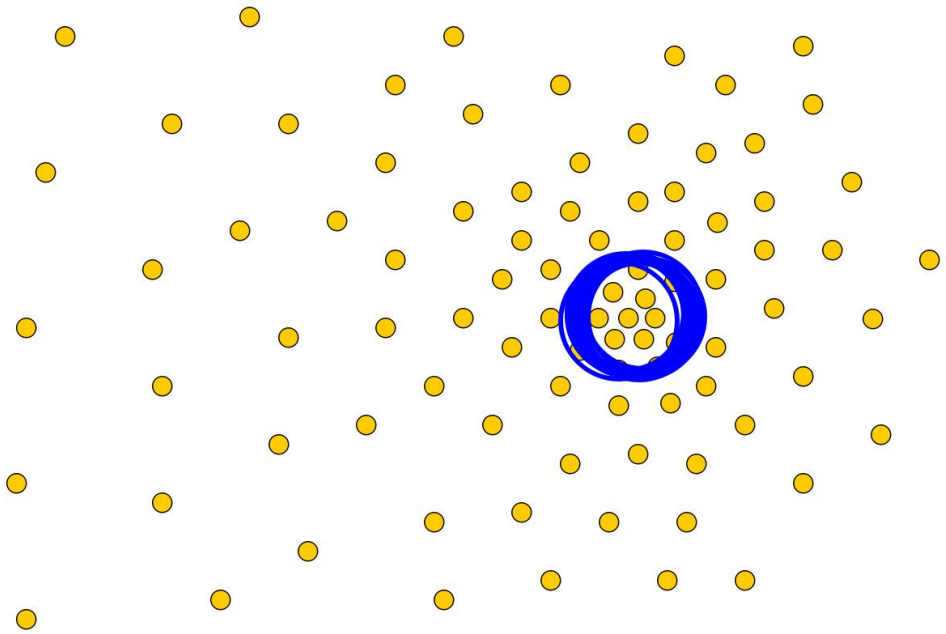
4-3 Mean Shift聚类



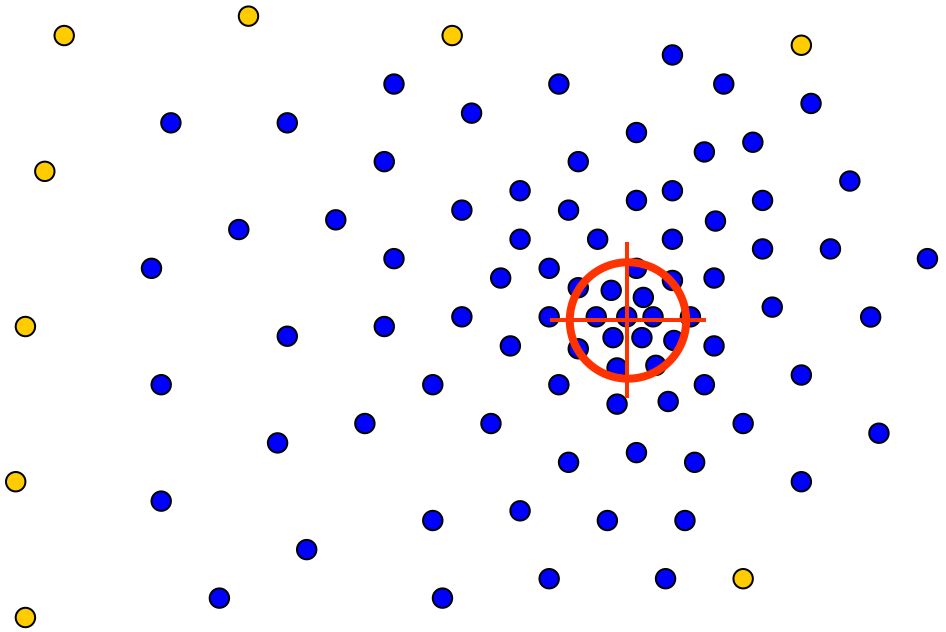
Tessellate the space with windows

Run the procedure in parallel

4-3 Mean Shift聚类



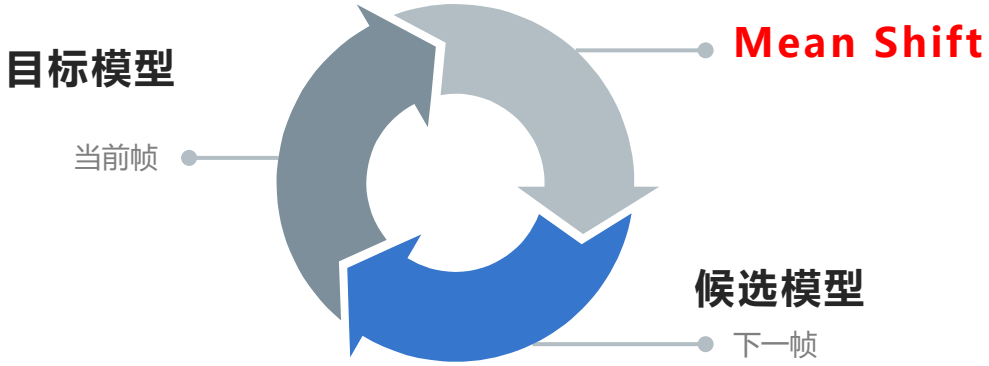
4-3 Mean Shift聚类



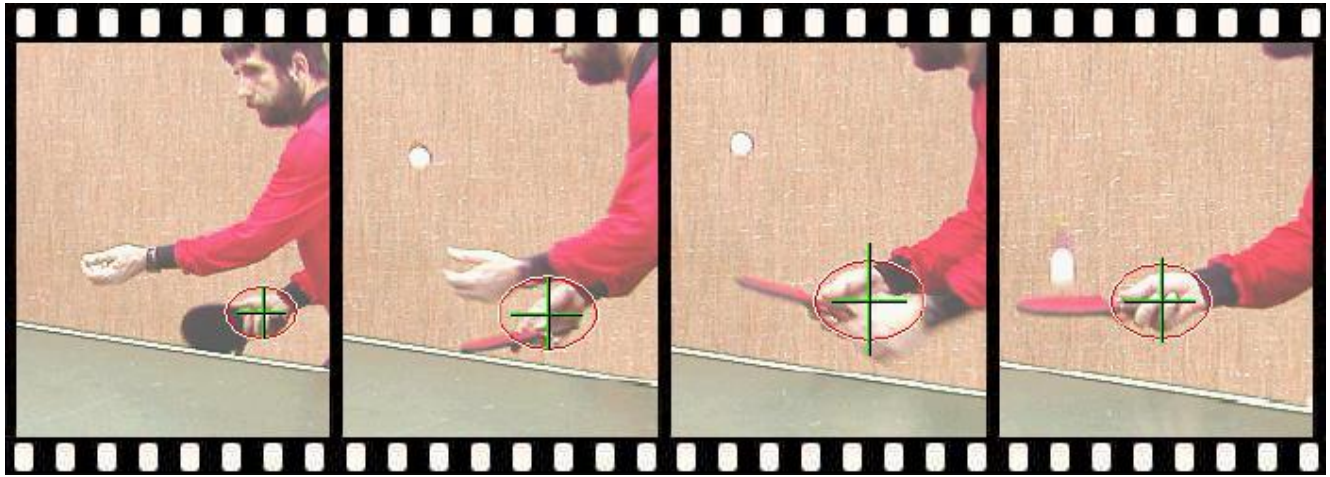
The blue data points were traversed by the windows towards the mode

4 算法进阶

4-3 Mean Shift聚类



4-3 Mean Shift聚类



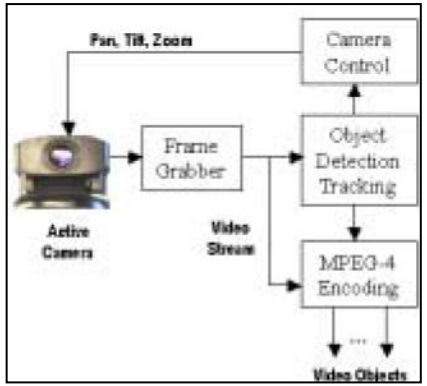
… → …

4-3 Mean Shift聚类

Surveillance

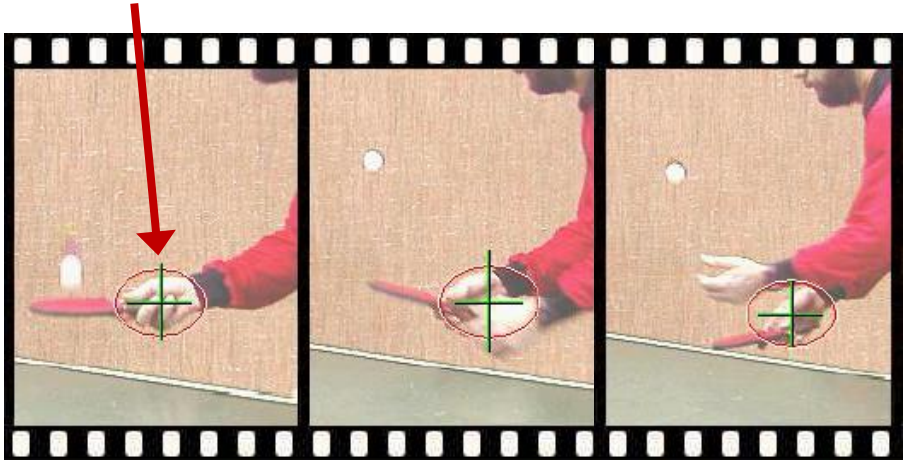
Driver Assistance

Object-Based Video Compression



4-3 Mean Shift聚类

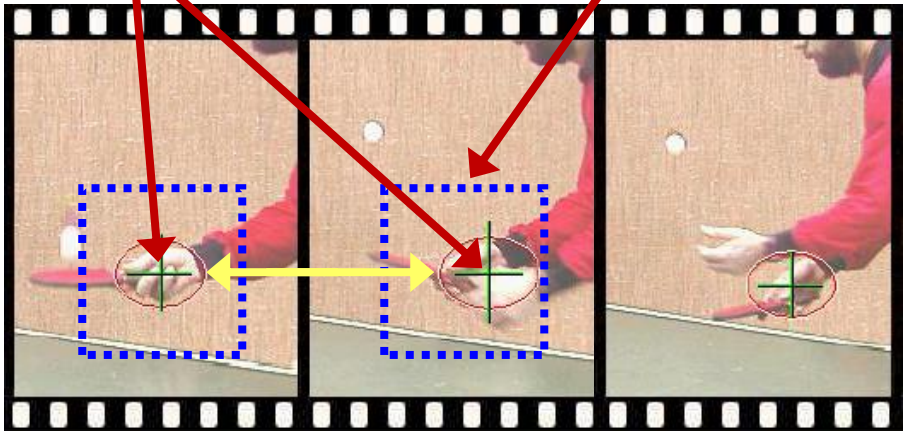
Choose a reference model in the current frame → Choose a feature space → Represent the model in the chosen feature space



… Current frame → …

4-3 Mean Shift聚类

Start from the position of the model in the current frame → Search in the model’s neighborhood in next frame → Find best candidate by maximizing a similarity func.



Model

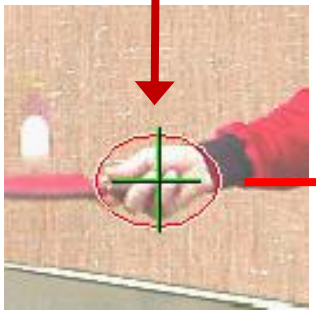
Candidate

Repeat the same process in the next pair of frames

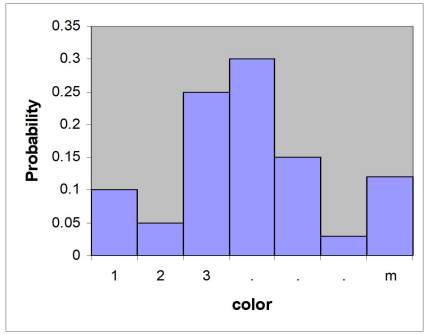
… Current frame → …

4-3 Mean Shift聚类

Choose a reference target model → Choose a feature space → Represent the model by its PDF in the feature space



Quantized Color Space



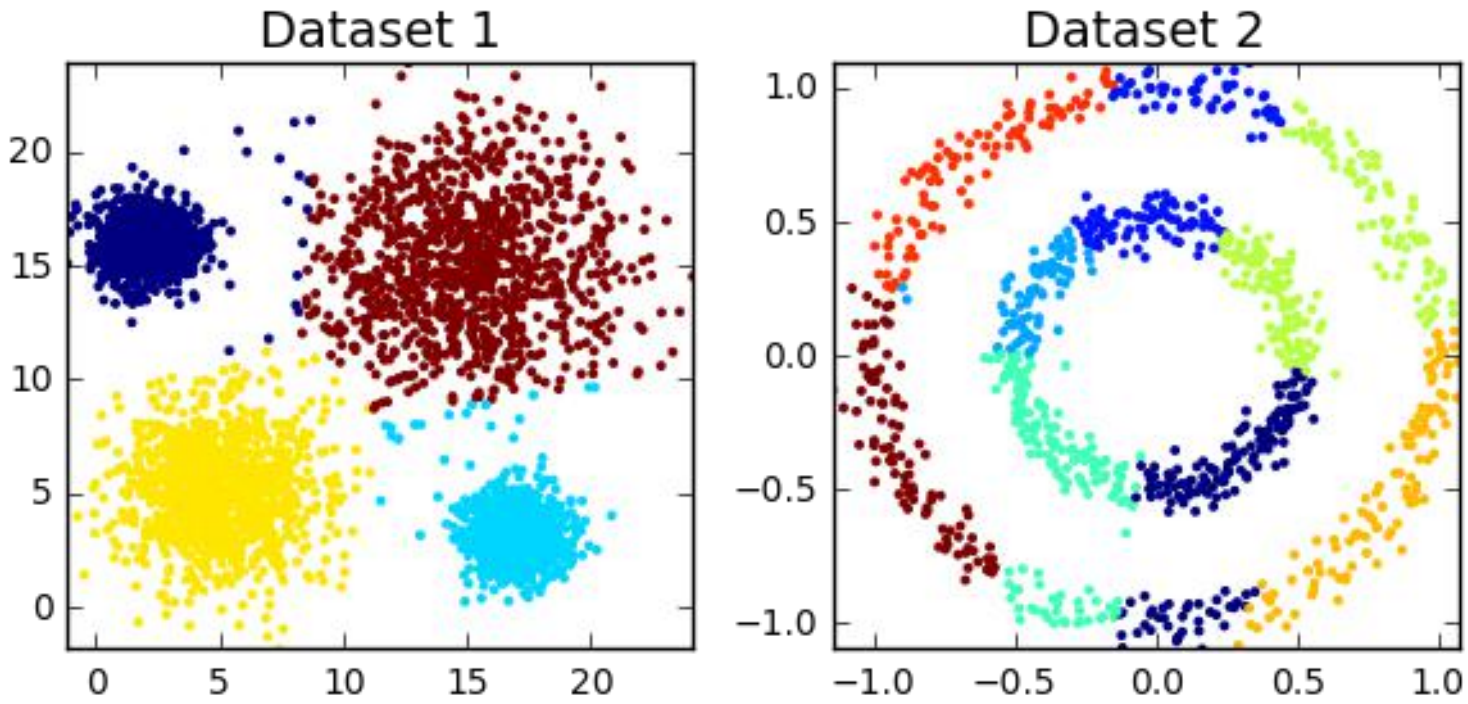
4-3 Mean Shift聚类

首先对跟踪目标进行描述，这个描述是将跟踪目标区域转换为颜色HSV空间，然后得到H的这个通道的分布直方图，有了这个描述之后，我们就是要在下一个视频帧中找到和这个描述的一样的区域，但是我们知道要找到完全一样的区域很难，所以我们就用了一个相似函数来衡量我们找到的区域和我们的目标区域的相似度，通过这个相似函数，相似函数值越大说明我们找打的区域和目标区域越相似，所以我们的目标就是要找这个对应最大相似值的区域，那么怎么来找呢？这个时候meanshift就排上用场了，它可以通过不断地迭代得到有最大相似值的区域（具体里面的是怎么算的，可以参考博文地底下的参考博客），meanshift的作用可以让我们的搜索窗口不断向两个模型相比颜色变化最大的方向不断移动，直到最后两次移动距离小于阈值，即找到当前帧的位置，并以此作为下一帧的起始搜索窗口中心，如此重复，这个过程每两帧之间都会产生一个meanshift向量，整个过程的meanshift向量连起来就是目标的运动路径。

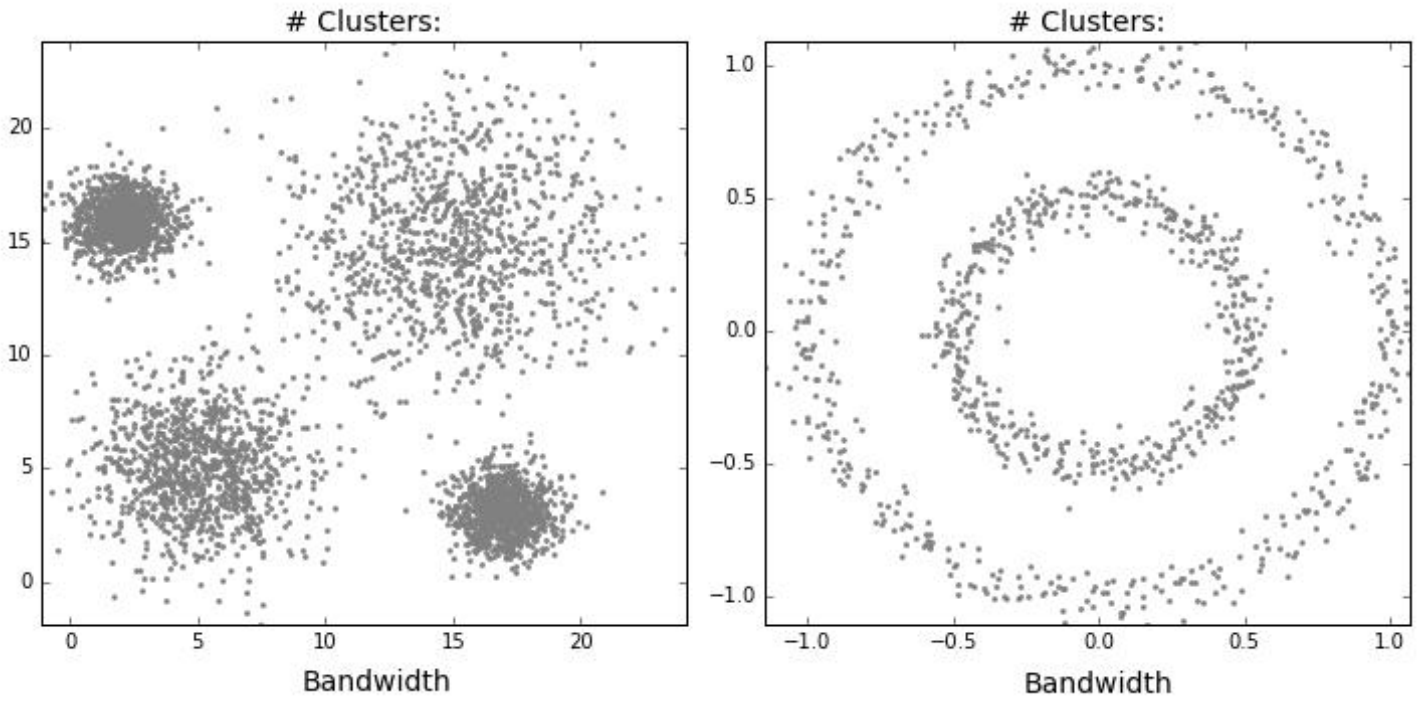
可删

4 算法进阶

4-3 Mean Shift聚类

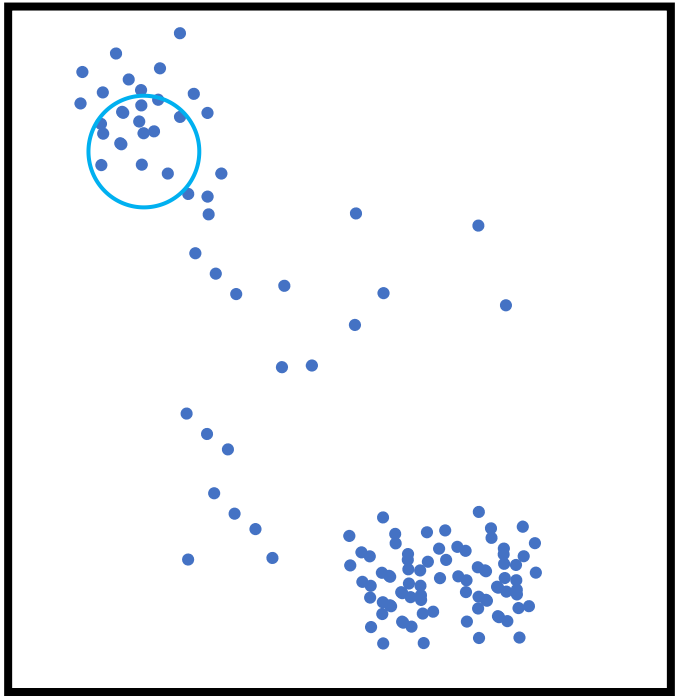


4-3 Mean Shift聚类



4-3 Mean Shift聚类

恰当的选择

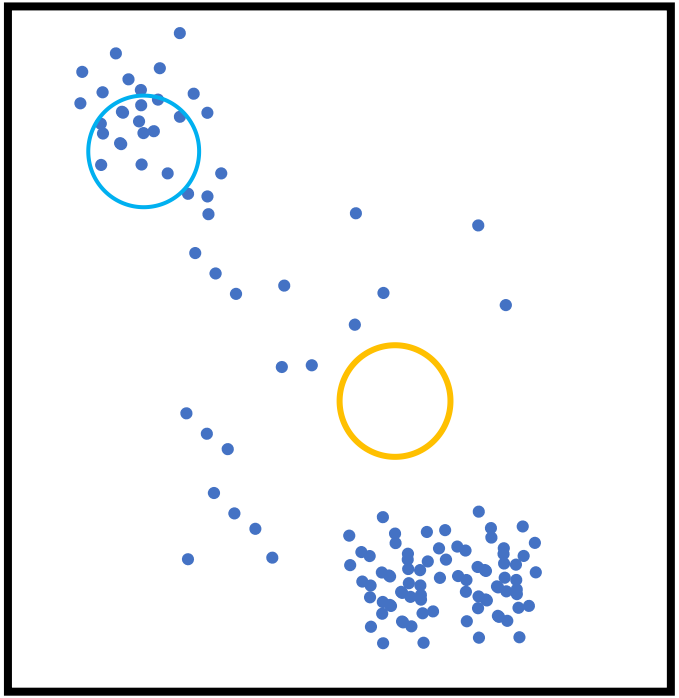


4 算法进阶

4-3 Mean Shift聚类

恰当的选择

初始位置选择不恰当



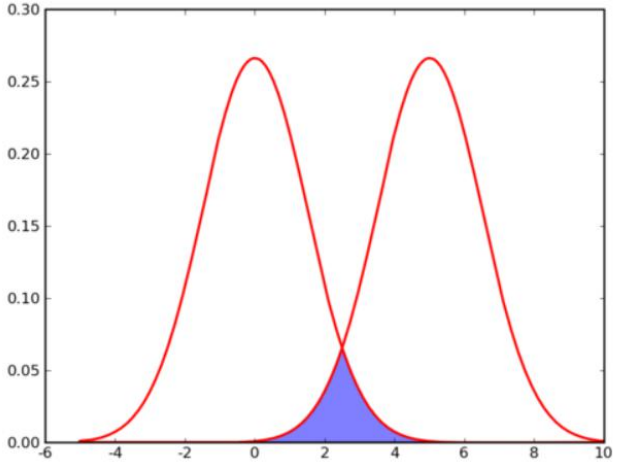
4 算法进阶

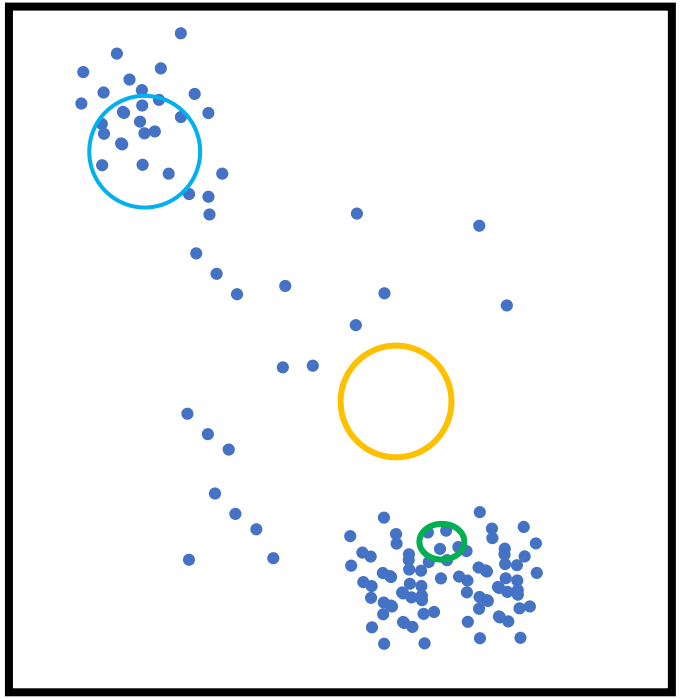
4-3 Mean Shift聚类

恰当的选择

初始位置选择不恰当

处于概率密度峰值之间，算法震荡不收敛





4 算法进阶

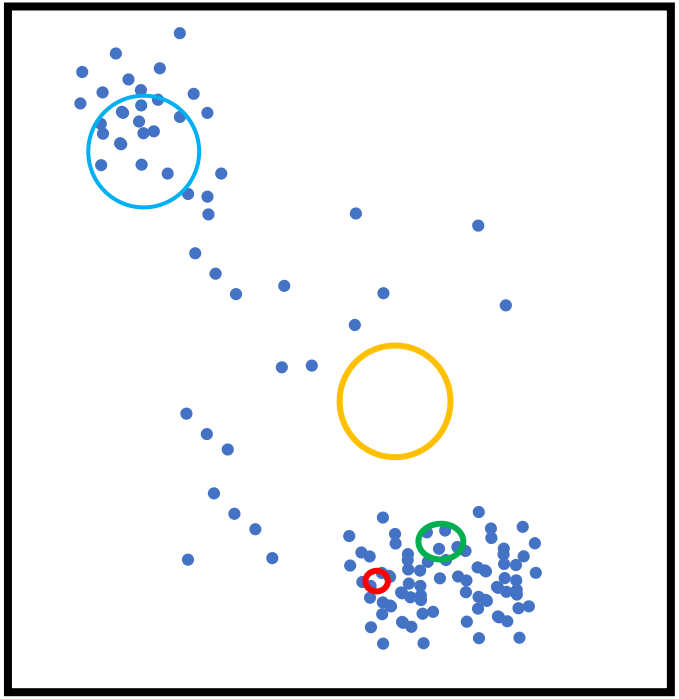
4-3 Mean Shift聚类

恰当的选择

初始位置选择不恰当

处于概率密度峰值之间，算法震荡不收敛

选取窗口过小也会震荡



4 算法进阶

4-4 Affinity Propagation (AP)聚类

Exemplar：聚类族中心点；

s(i,j)：数据点i与数据点j的相似度值，理解为数据点j作为数据点i的聚类中心的能力；

一般使用欧氏距离的的负值表示，即s(i,j)值越大表示点i与j的距离越近；

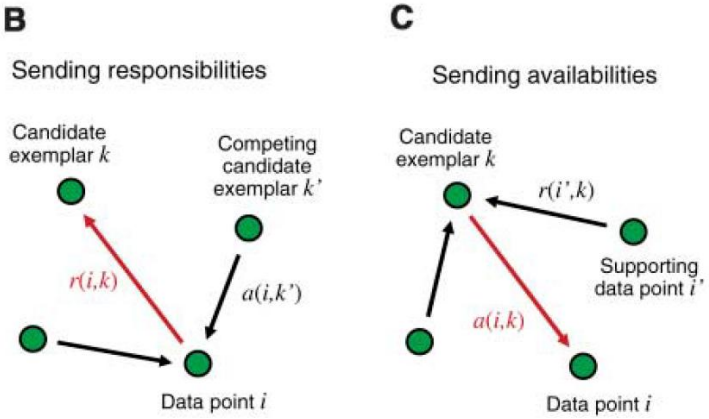
相似度矩阵：作为算法的初始化矩阵，n个点就有由n乘n个相似度值组成的矩阵；

Responsibility，r(i,k)：吸引度信息，表示数据点k适合作为数据点i的聚类中心的程度；

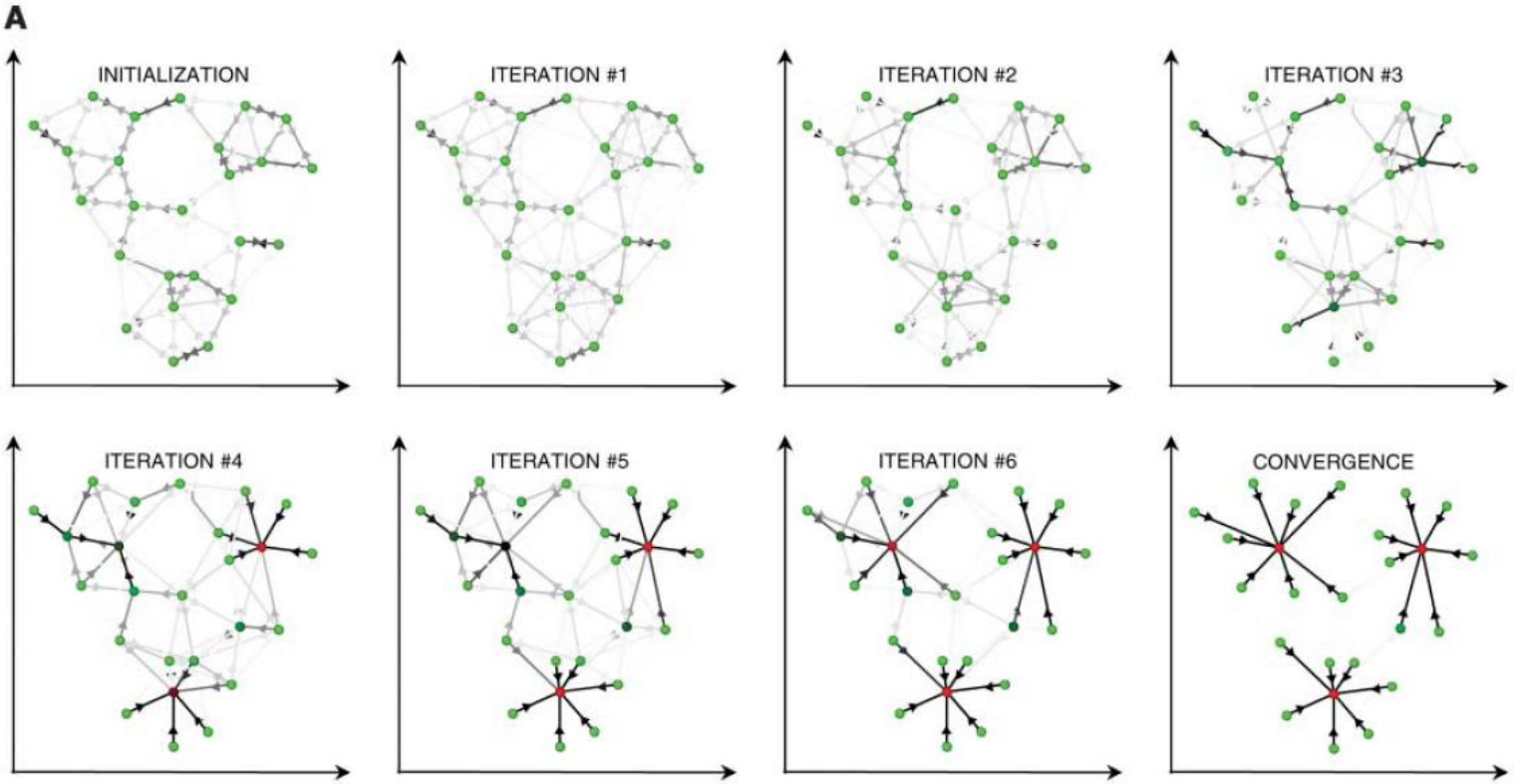
$$r(i,k) \leftarrow s(i,k) - \max_{k' \text{ s.t. } k' \neq k} \{a(i,k') + s(i,k')\}$$

$$a(i,k) \leftarrow \min\left\{0, r(k,k) + \sum_{i' \text{ s.t. } i' \notin \{i,k\}} \max\{0, r(i',k)\}\right\}$$

$$a(k,k) \leftarrow \sum_{i' \text{ s.t. } i' \neq k} \max\{0, r(i',k)\}$$



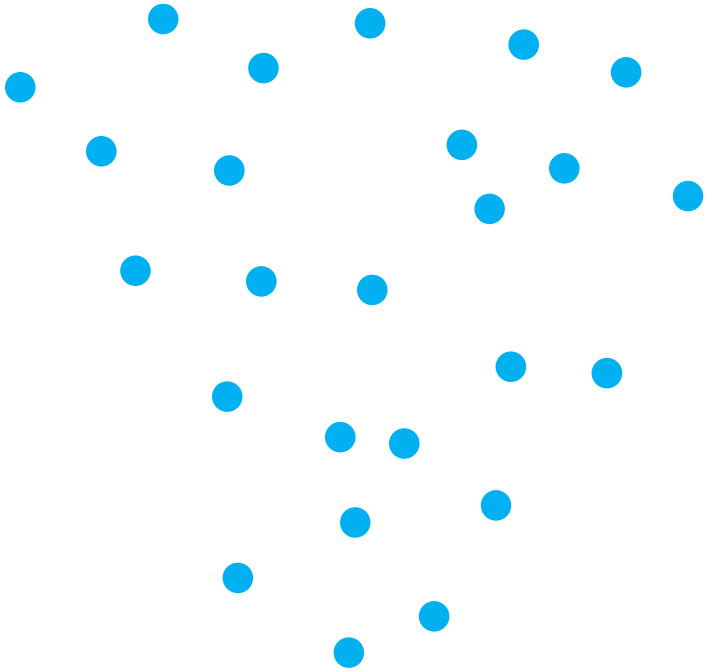
4-4 Affinity Propagation (AP)聚类



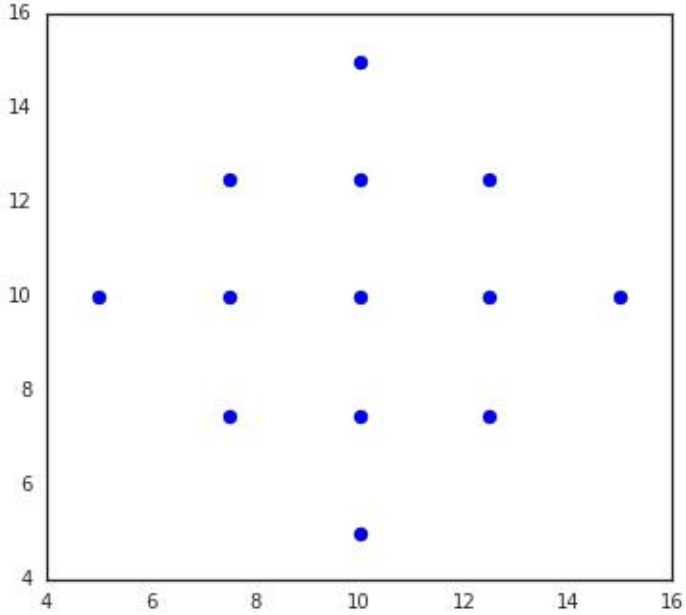
4-4 Affinity Propagation (AP)聚类

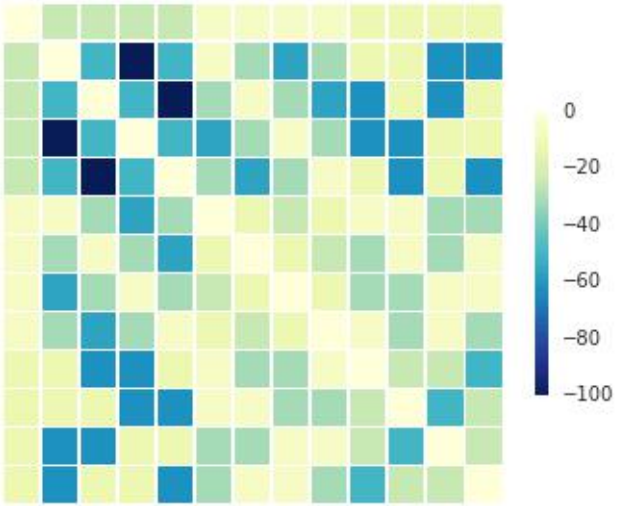
Preference参考度或称为偏好参数：

是相似度矩阵中横轴纵轴索引相同的点，如s(i,i)，若按欧氏距离计算其值应为0，但在AP聚类中其表示数据点i作为聚类中心的程度，因此不能为0。迭代开始前假设所有点成为聚类中心的能力相同，因此参考度一般设为相似度矩阵中所有值的最小值或者中位数，参考度越大则说明个数据点成为聚类中心的能力越强，则最终聚类中心的个数则越多



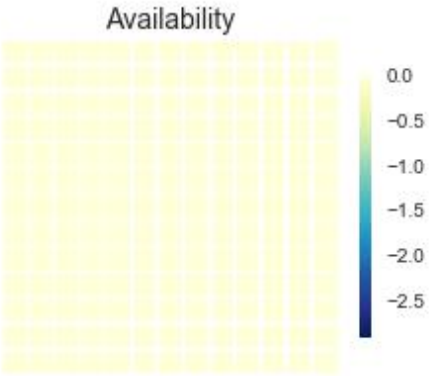
4-4 Affinity Propagation (AP)聚类

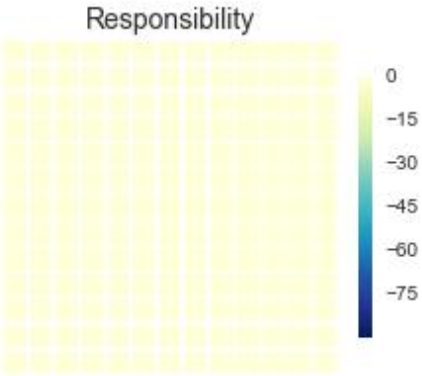




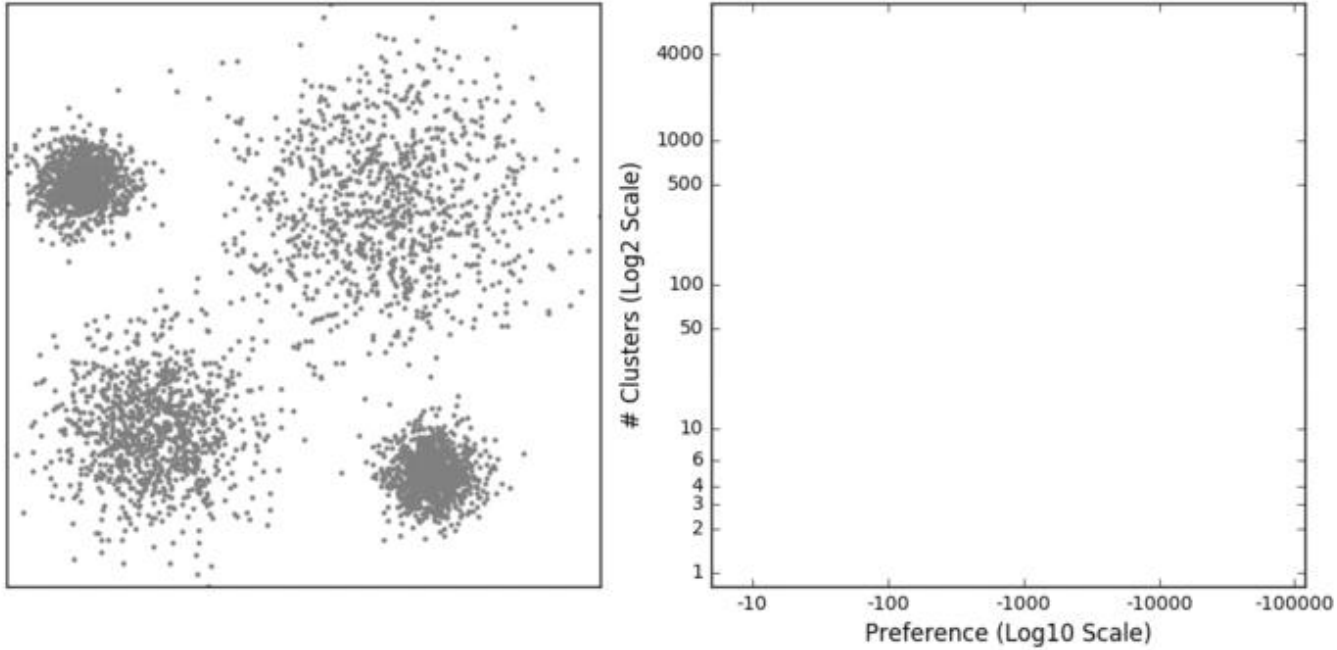
4 算法进阶

4-4 Affinity Propagation (AP)聚类





4-4 Affinity Propagation (AP)聚类



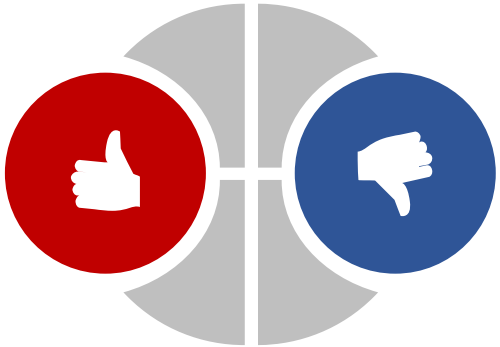
4-4 Affinity Propagation (AP)聚类

（1）不需要制定最终聚类族的个数

（2）已有的数据点作为最终的聚类中心，而不是新生成一个族中心。

（3）模型对数据的初始值不敏感。

（4）对初始相似度矩阵数据的对称性没有要求。



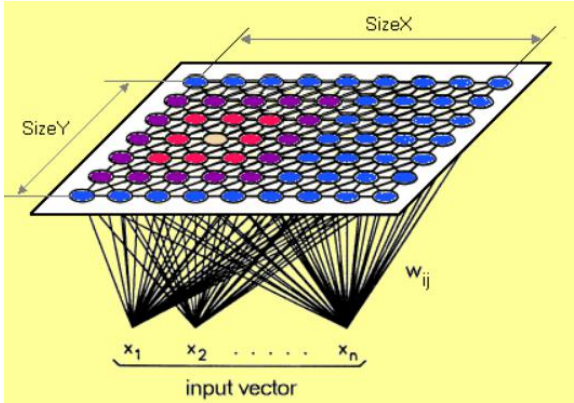
（1）AP算法需要事先计算每对数据对象之间的相似度，如果数据对象太多的话，内存放不下，若存在数据库，频繁访问数据库也需要时间。

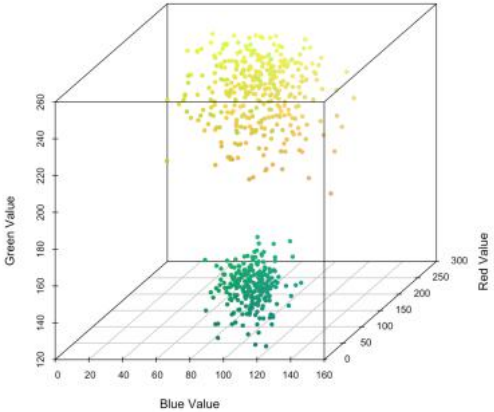
（2）AP算法的时间复杂度较高，一次迭代大概$O(N^3)$

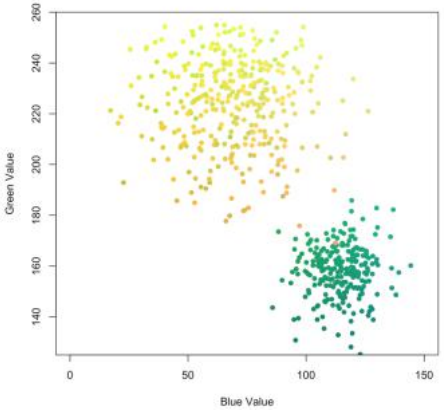
（3）聚类的好坏受到参考度和阻尼系数的影响。

4-5 SOM聚类

SOM 即自组织映射，是一种用于特征检测的无监督学习神经网络。它模拟人脑中处于不同区域的神经细胞分工不同的特点，即不同区域具有不同的响应特征，而且这一过程是自动完成的。SOM 用于生成训练样本的低维空间，可以将高维数据间复杂的非线性统计关系转化为简单的几何关系，且以低维的方式展现，因此通常在降维问题中会使用它。







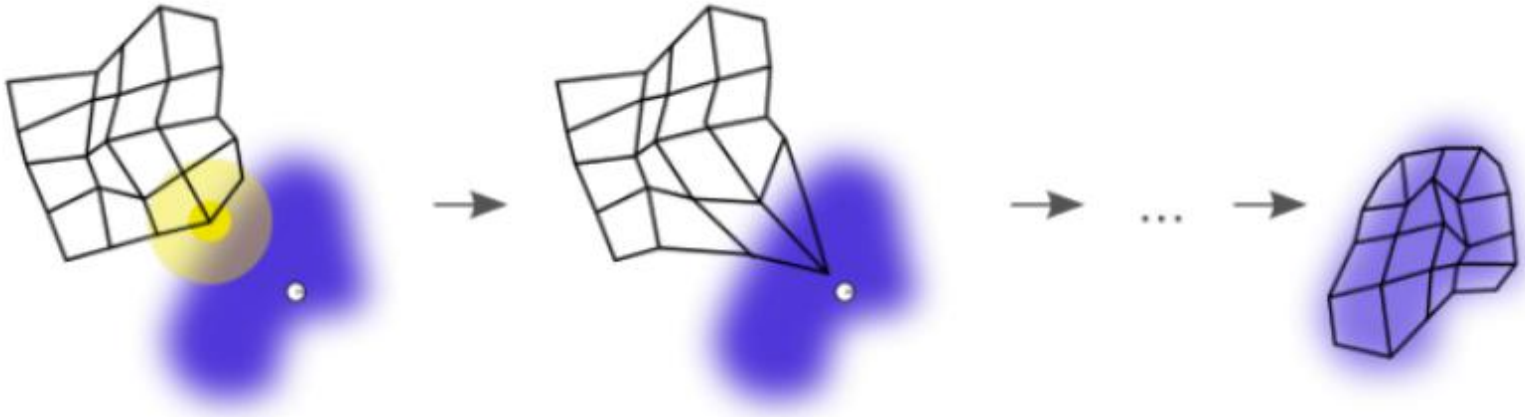
4-5 SOM聚类

SOM 的训练过程：

紫色区域表示训练数据的分布状况，白色网格表示从该分布中提取的当前训练数据。

（1） SOM 节点位于数据空间的任意位置，最接近训练数据的节点（黄色高亮部分）会被选中。它和网格中的邻近节点一样，朝训练数据移动。

（2）在多次迭代之后，网格倾向于近似该种数据分布（下图最右）。

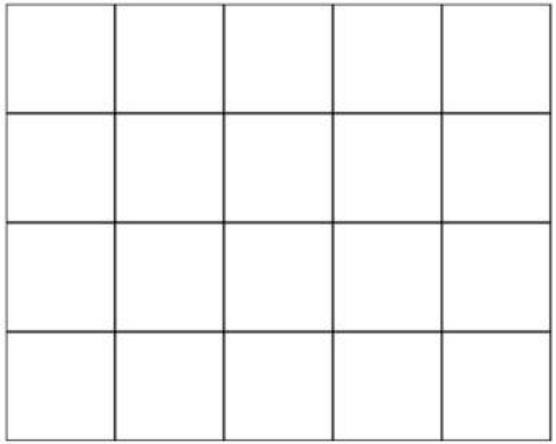


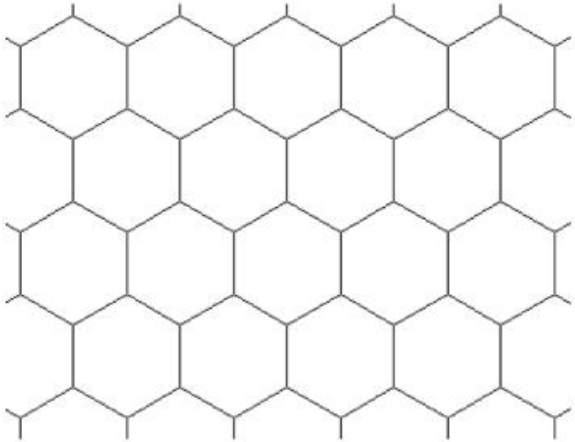
4-5 SOM聚类

所有的神经元组织成一个网格，网格可以是六边形、四边形……，甚至是链状、圆圈……

网络的结构通常取决于输入的数据在空间中的分布。

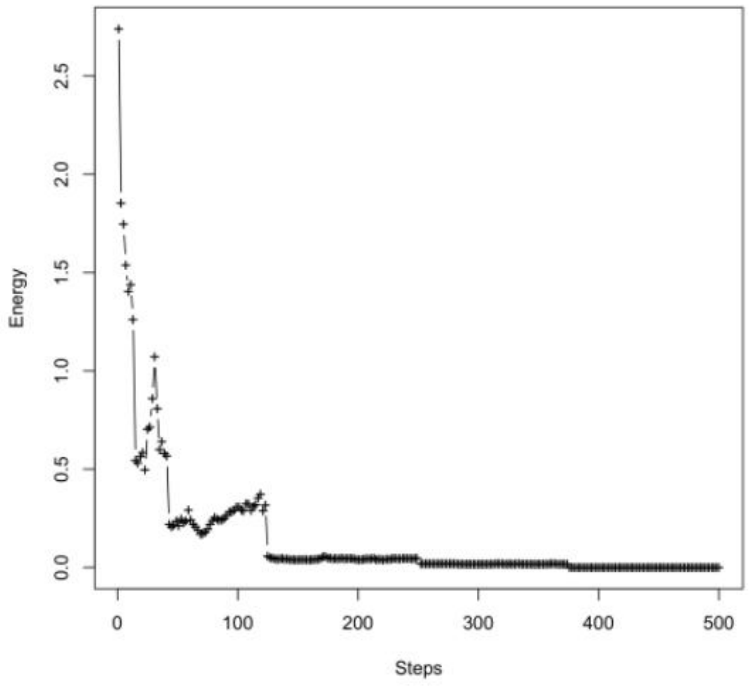
SOM的作用是将这个网格铺满数据存在的空间。

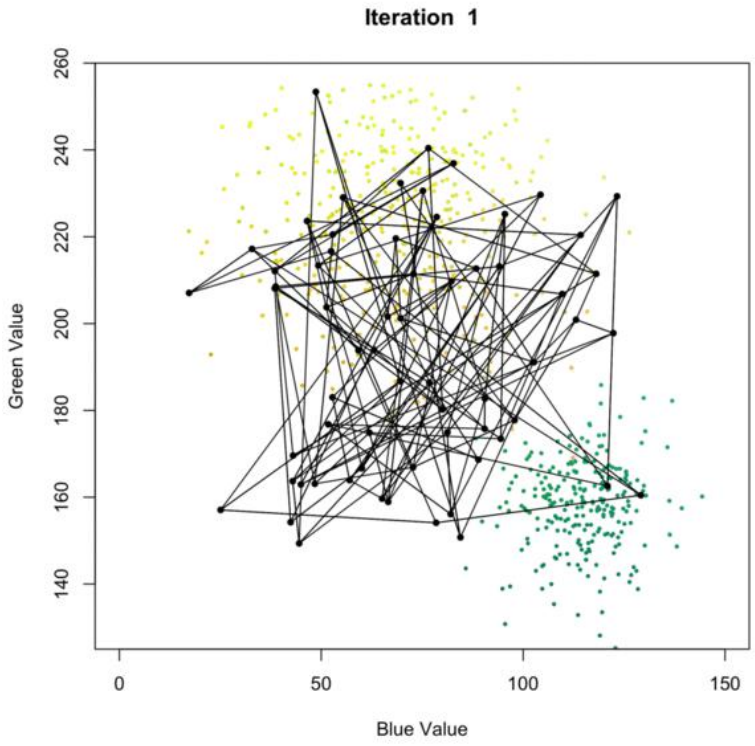




4 算法进阶

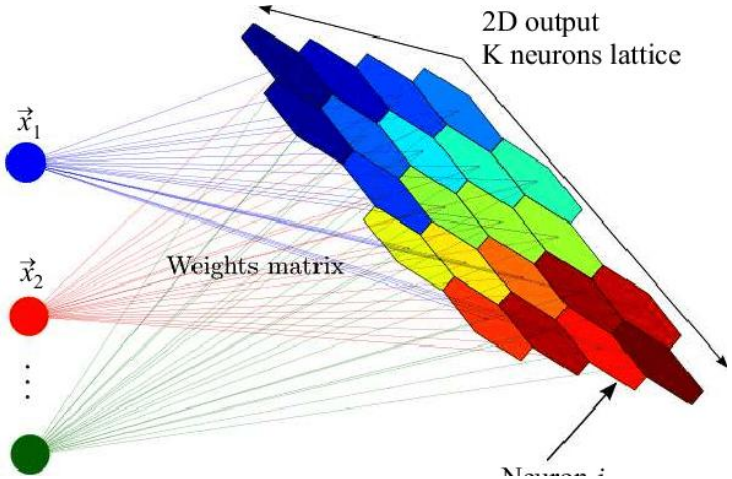
4-5 SOM聚类

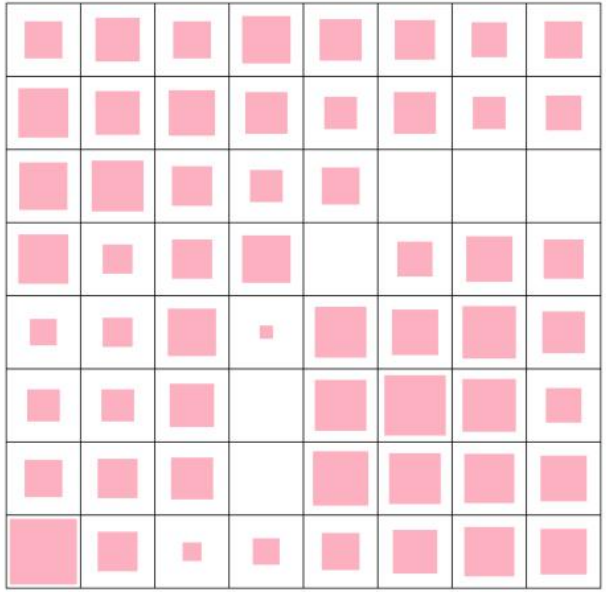




4-5 SOM聚类

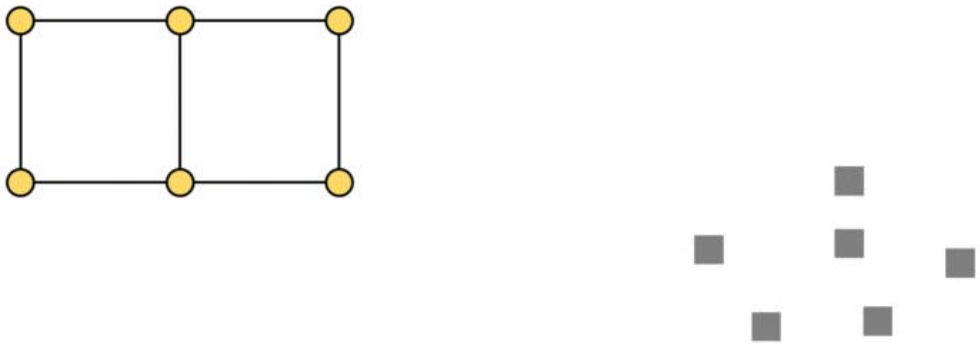
每个神经元由正方形表示，正方形内的粉红色区域表示神经元最接近的数据点的相对数量 - 粉红色区域越大，该神经元表示的数据点越多。





4-5 SOM聚类

当我们将训练数据输入到网络中时，会计算出所有权重向量的欧几里德距离。权重向量与输入最相似的神经元称为最佳匹配单元（BMU）。BMU 的权重和 SOM 网格中靠近它的神经元会朝着输入矢量的方向调整。一旦确定了 BMU，下一步就是计算其它哪些节点在 BMU 的邻域内。



Step 0: Position neurons (orange) in data space.

4-5 SOM聚类

（1）将网格的神经元随机定位在数据空间中。

（2）选择一个数据点，按顺序随机或系统地循环遍历数据集

（3）找到最接近所选数据点的神经元。这种神经元被称为最佳匹配单元（BMU）。

（4）将BMU移近该数据点。 BMU移动的距离由学习速率确定，学习速率在每次迭代后减小。

（5）将BMU的邻居移动到更靠近该数据点的位置，远处的邻居移动得更少。使用BMU周围的半径来识别邻居，并且在每次迭代之后该半径的值减小。

（6）在重复步骤1到4之前，更新学习速率和BMU半径。迭代这些步骤，直到神经元的位置稳定。

4 算法进阶

4-5 SOM聚类

SOM 通常用在可视化中。比如右图，世界各国贫困数据的可视化。生活质量较高的国家聚集在左上方，而贫困最严重的国家聚集在右下方。

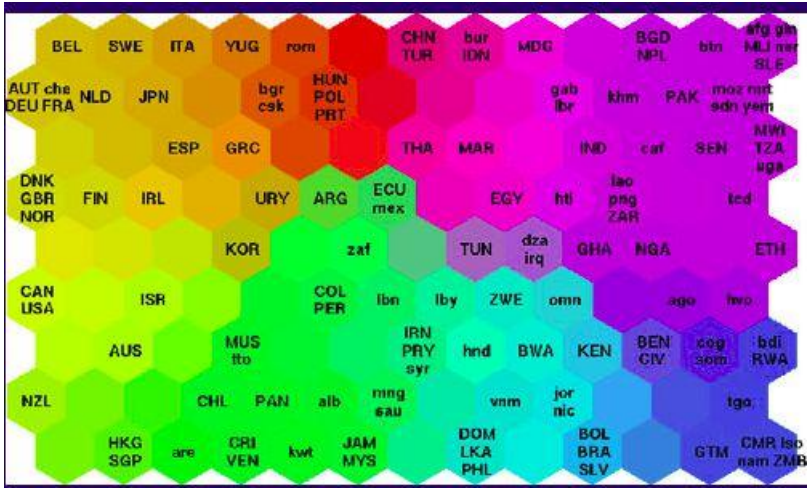
SOM 的其它一些应用还包括：

数据压缩

语音识别

分离音源

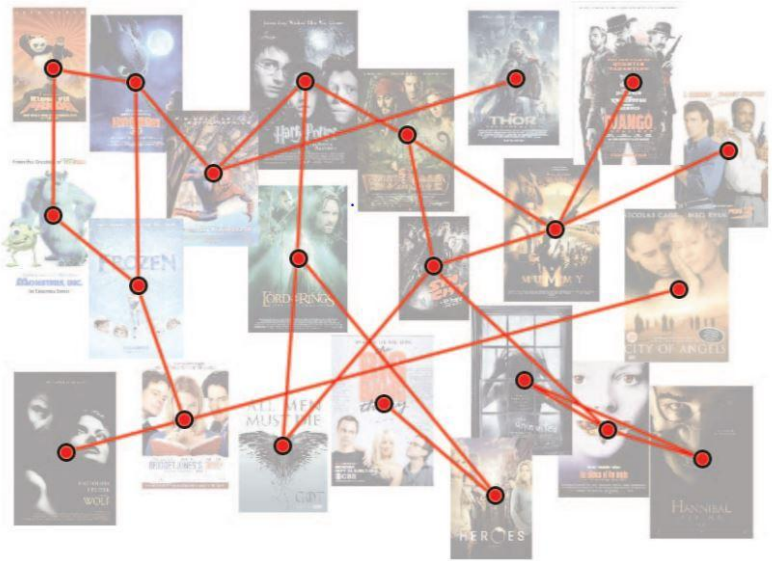
欺诈检测



4-6 谱聚类

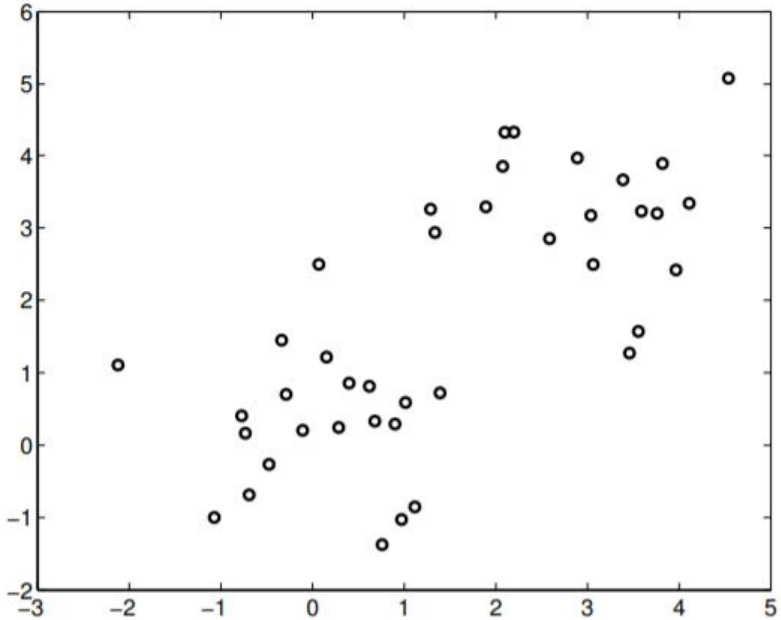
谱聚类：是一种基于**图论**的聚类方法，通过对样本数据的**拉普拉斯矩阵**的**特征向量**进行聚类。

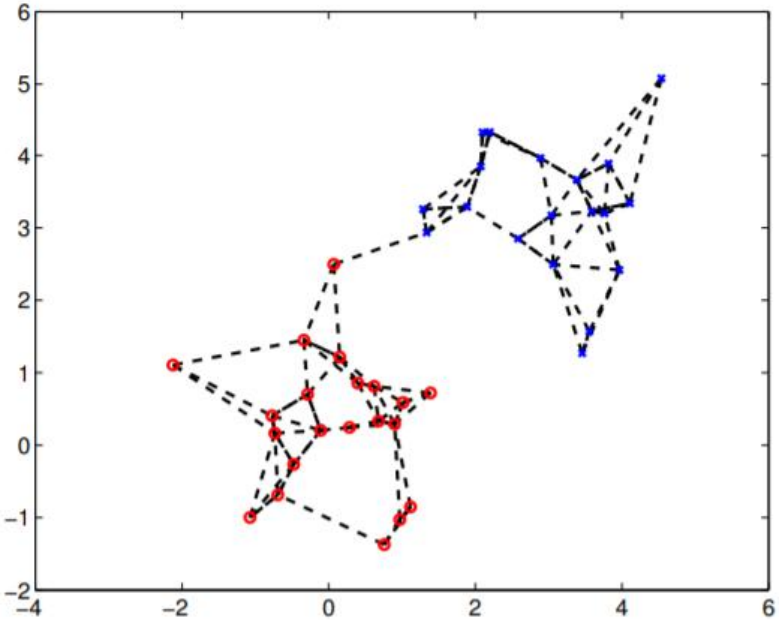
图（Graph）：由若干**点**及连接两点的**线**所构成的图形，通常用来描述某些事物之间的某种关系，用点代表事物，线表示对应两个事物间具有这种关系。



4 算法进阶

4-6 谱聚类



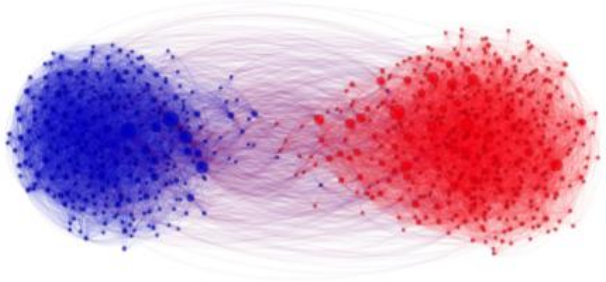


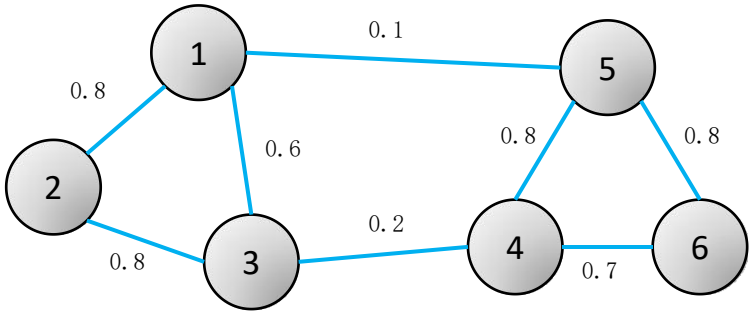
4-6 谱聚类

$G(V,E)$ 表示无向图， $V=\{v_1,v_2,\ldots,v_n\}$ 表示点集，E表示边集

w_{ij} 表示 v_i 与 v_j 之间的关系，称作权重，对于无向图

$W_{ij}=W_{ji}$ 而且 $w_{ii}=0$ $w_{ij}\geq 0$

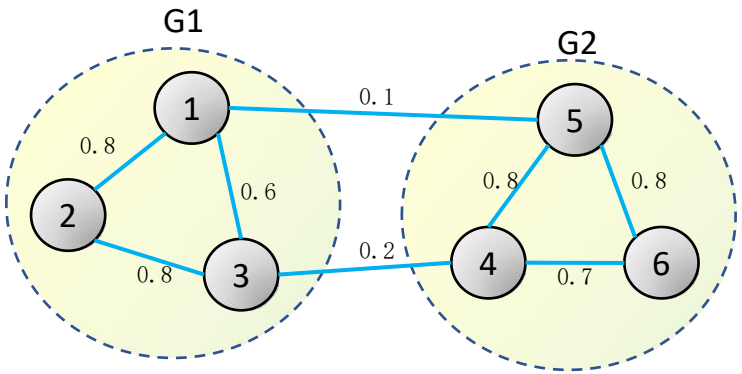


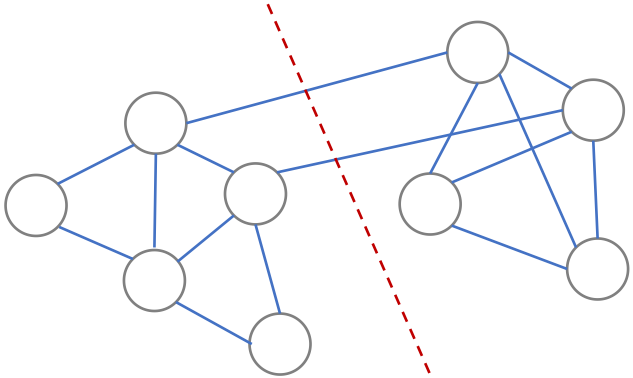


4-6 谱聚类

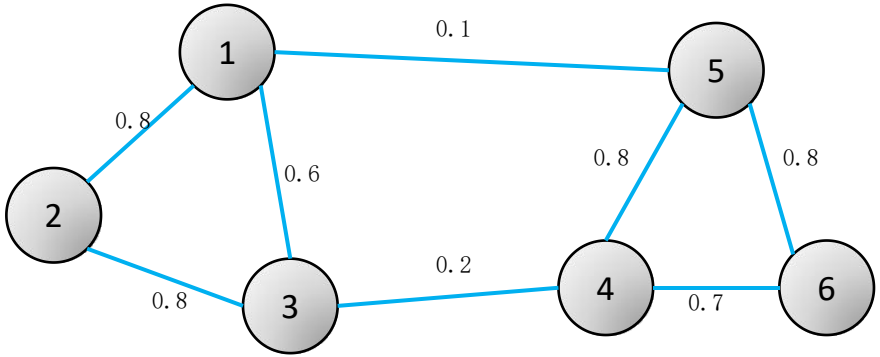
划分时子图之间被“截断”的边的权重和

$$Cut(G_1, G_2) = \sum_{i \in G_1, j \in G_2} w_{ij}$$

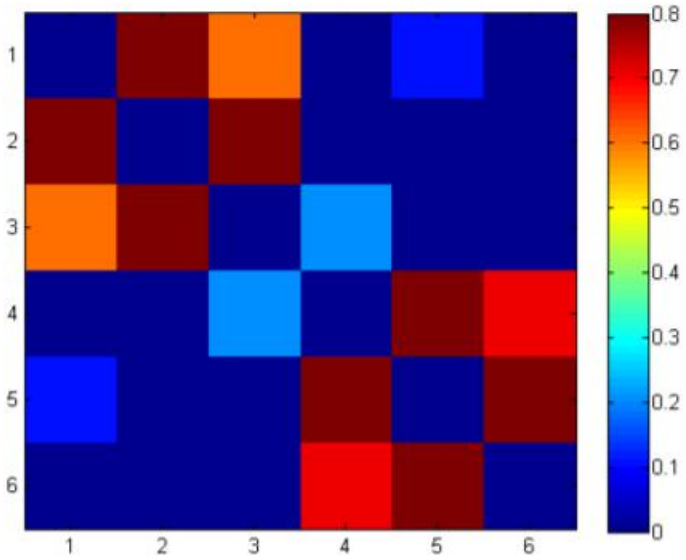




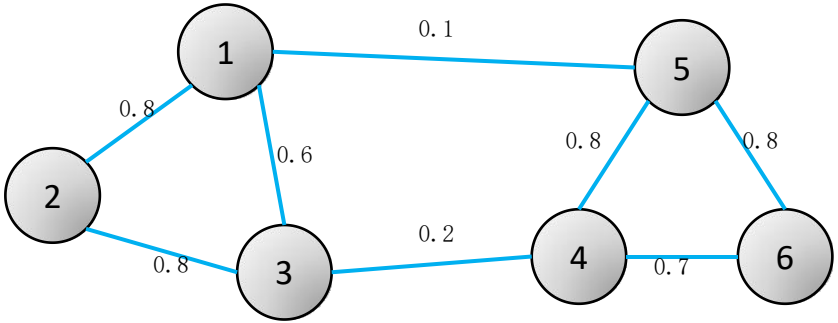
4-6 谱聚类



	1	*2*	*3*	*4*	*5*	*6*
1	0.0	0.8	0.6	0.0	0.1	0. 0
2	0.8	0.0	0.8	0.0	0.0	0.0
3	0.6	0.8	0.0	0.2	0.0	0.0
4	0.0	0.	0.2	0.0	0.8	0.7
5	0.1	0.0	0.0	0.8	0.0	0.8
6	0.0	0.0	0.0	0.7	0.8	0.0



4-6 谱聚类



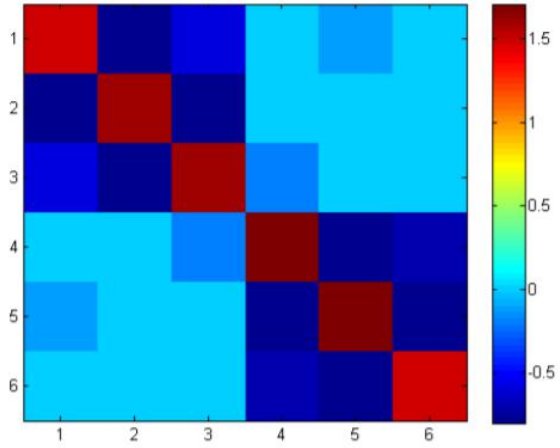
	1	*2*	*3*	*4*	*5*	*6*
1	0.0	0.8	0.6	0.0	0.1	0. 0
2	0.8	0.0	0.8	0.0	0.0	0.0
3	0.6	0.8	0.0	0.2	0.0	0.0
4	0.0	0.	0.2	0.0	0.8	0.7
5	0.1	0.0	0.0	0.8	0.0	0.8
6	0.0	0.0	0.0	0.7	0.8	0.0

	1	*2*	*3*	*4*	*5*	*6*
1	1.5	0.0	0.0	0.0	0.0	0. 0
2	0.0	1.6	0.0	0.0	0.0	0.0
3	0.0	0.0	1.6	0.0	0.0	0.0
4	0.0	0.0	0.0	1.7	0.0	0.0
5	0.0	0.0	0.0	0.0	1.7	0.0
6	0.0	0.0	0.0	0.0	0.0	1.5

4-6 谱聚类

邻接矩阵W

	1	2	3	4	5	6
1	0.0	0.8	0.6	0.0	0.1	0. 0
2	0.8	0.0	0.8	0.0	0.0	0.0
3	0.6	0.8	0.0	0.2	0.0	0.0
4	0.0	0.0	0.2	0.0	0.8	0.7
5	0.1	0.0	0.0	0.8	0.0	0.8
6	0.0	0.0	0.0	0.7	0.8	0.0



拉普拉斯矩阵L=D-W

	1	*2*	*3*	*4*	*5*	*6*
1	1.5	-0.8	-0.6	0.0	-0.1	0. 0
2	-0.8	1.6	-0.8	0.0	0.0	0.0
3	-0.6	-0.8	1.6	-0.2	0.0	0.0
4	0.0	0.0	-0.2	1.7	-0.8	-0.7
5	-0.1	0.0	0.0	-0.8	1.7	-0.8
6	0.0	0.0	0.0	-0.7	-0.8	1.5

度矩阵D

	1	*2*	*3*	*4*	*5*	*6*
1	1.5	0.0	0.0	0.0	0.0	0. 0
2	0.0	1.6	0.0	0.0	0.0	0.0
3	0.0	0.0	1.6	0.0	0.0	0.0
4	0.0	0.0	0.0	1.7	0.0	0.0
5	0.0	0.0	0.0	0.0	1.7	0.0
6	0.0	0.0	0.0	0.0	0.0	1.5

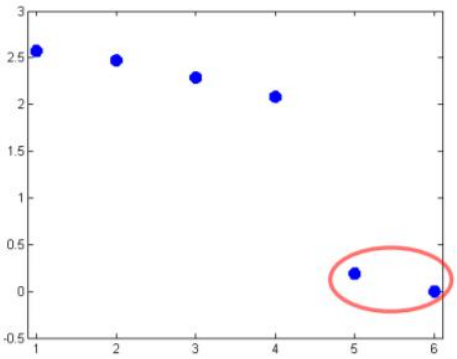
4-6 谱聚类

U=

-0.4082	0.4084
-0.4082	0.4418
-0.4082	0.3713
-0.4082	-0.3713
-0.4082	-0.4050
-0.4082	-0.4452

Eigen-values λ=

0
0.18
2.08
2.28
2.46
2.57



Eigen-vector v=

-0.4082	0.4084	……
-0.4082	0.4418	……
-0.4082	0.3713	……
-0.4082	-0.3713	……
-0.4082	-0.4050	……
-0.4082	-0.4452	……

U

N x K

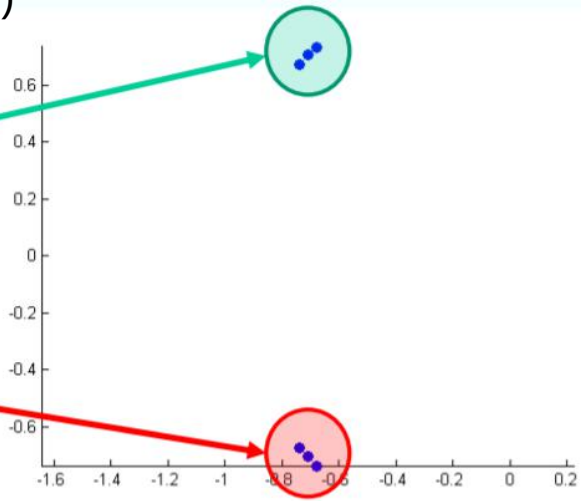
4-6 谱聚类

U=

-0.4082	0.4084
-0.4082	0.4418
-0.4082	0.3713
-0.4082	-0.3713
-0.4082	-0.4050
-0.4082	-0.4452

Y=row_normalise(U)

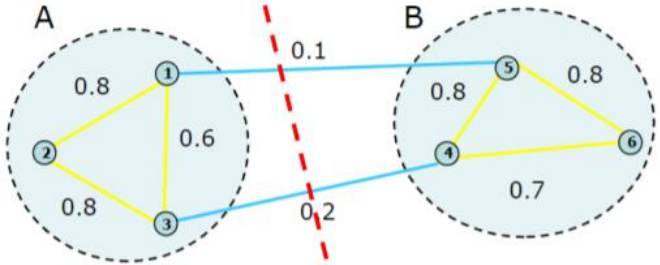
-0.7070	0.7072
-0.6786	0.7345
-0.7398	0.6729
-0.7398	-0.6729
-0.7099	-0.7043
-0.6759	-0.7370

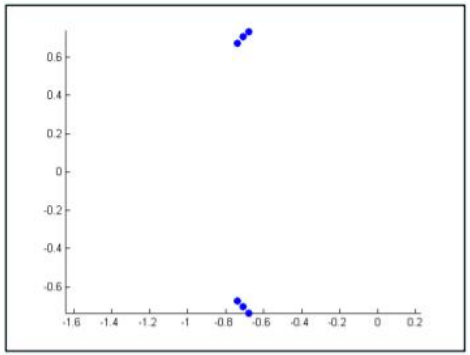


4 算法进阶

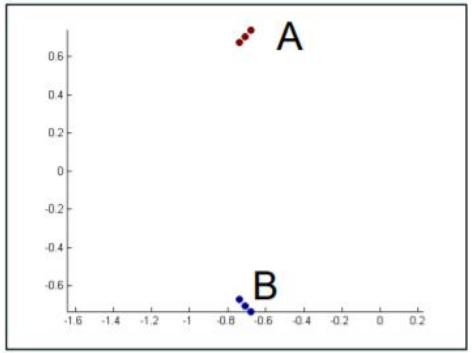
4-6 谱聚类

- K-means clustering with 2 clusters
- Easy，convex clustering problem



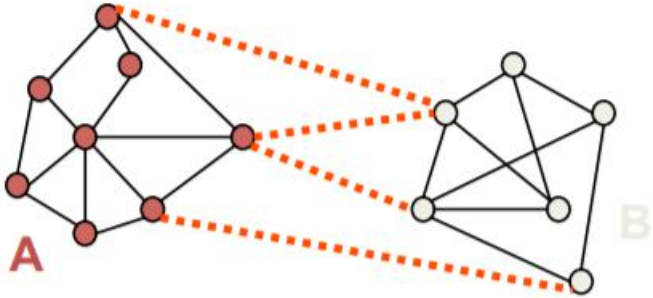


K-means



4-6 谱聚类

$$q_i = \begin{cases} c & i \in G_1 \\ -c & i \in G_2 \end{cases}$$



求: $\min(q^T L q)$

条件: $q^T q = \sum_{i=1}^{n} q_i^2 = nc^2$

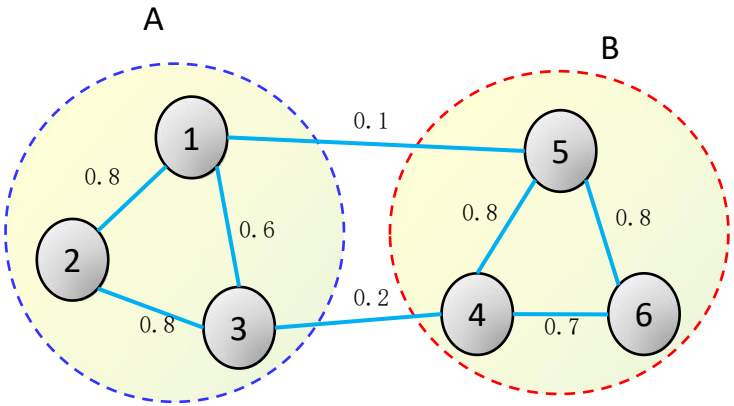
4 算法进阶

4-6 谱聚类

拉普拉斯矩阵L

	1	2	3	4	5	6
1	1.5	-0.8	-0.6	0.0	-0.1	0. 0
2	-0.8	1.6	-0.8	0.0	0.0	0.0
3	-0.6	-0.8	1.6	-0.2	0.0	0.0
4	0.0	0.0	-0.2	1.7	-0.8	-0.7
5	-0.1	0.0	0.0	-0.8	1.7	-0.8
6	0.0	0.0	0.0	-0.7	-0.8	1.5

1	2	3	4	5	6
0.408	-0.408	-0.647	-0.306	-0.379	0.106
0.408	-0.442	0.014	0.305	0.706	0.215
0.408	-0.371	0.638	0.045	-0.388	-0.368
0.408	0.371	0.339	-0.455	-0.001	0.612
0.408	0.405	-0.167	-0.305	0.351	-0.652
0.408	0.445	-0.178	0.716	-0.289	0.087

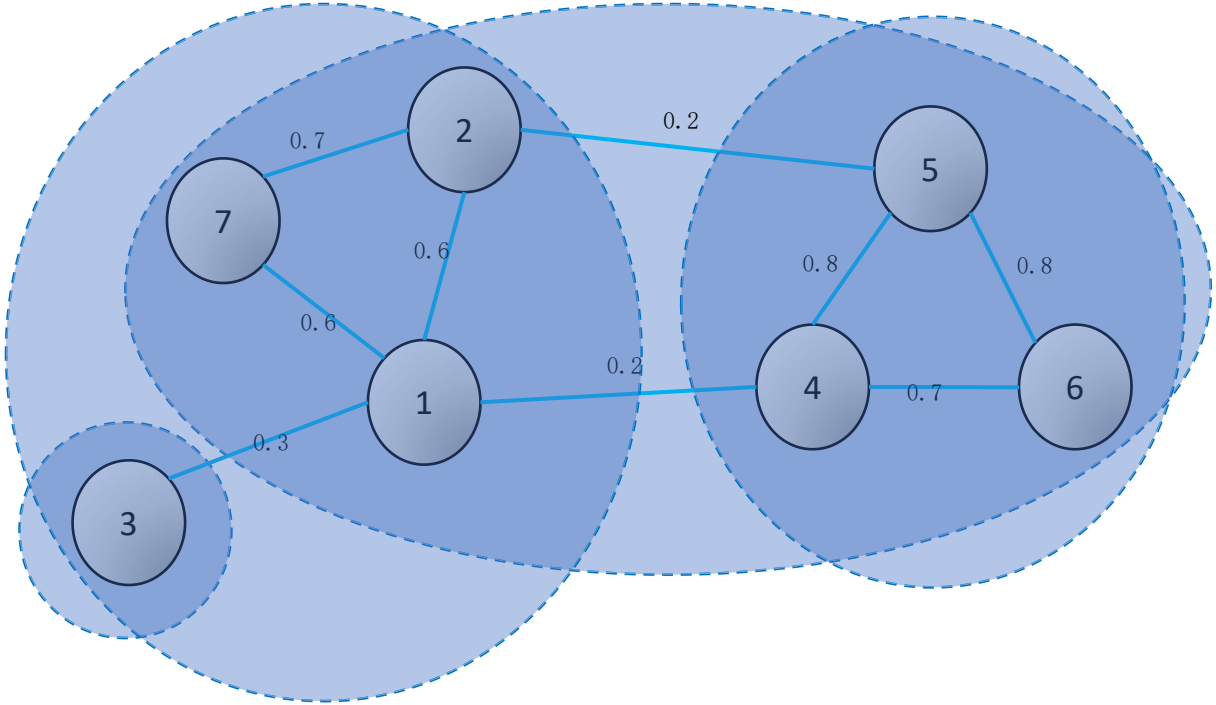


次小特征值的特征向量

-0.408
-0.442
-0.371
0.371
0.405
0.445

$$cut(A,B) = \sum_{i \in A, j \in B} w_{ij} = 0.3$$

4-6 谱聚类



Minimum Cut划分不均衡

小结

聚类方法	思路
DBSCAN	与划分和层次聚类方法不同，它将簇定义为密度相连的点的最大集合，能够把具有足够高密度的区域划分为簇，并可在噪声的空间数据库中发现任意形状的聚类。
层次聚类	合并算法通过计算两类数据点间的相似性，对所有数据点中最为相似的两个数据点进行组合，并反复迭代这一过程。
Mean Shift	多用于跟踪目标
AP聚类	将全部数据点都当作潜在的聚类中心(称之为exemplar)，然后数据点两两之间连线构成一个网络(相似度矩阵)，再通过网络中各条边的消息(responsibility和availability)传递计算出各样本的聚类中心。
SOM	WTA(Winner Takes All)竞争机制反映了自组织学习最根本的特征。
谱聚类	只要数据之间的相似度矩阵即可，计算复杂度小，且更健壮。

黑马程序员
www.itheima.com
传智播客旗下高端IT教育品牌

目录 Contents

5 综合实践

案例 4 聚类算法的文本文档实战应用

（1）简介需求：

结合DBSCAN和K-means对文本文档进行聚类分析，并生成对应章节的标签，如右图所示：

（2）分析提示：

可以尝试不同的数据集分别进行聚类分析，右图仅供参考。

Cluster 0：商家 商品 物流 品牌 支付 导购 网站 购物 平台 订单

Cluster 1：投资 融资 美元 公司 资本 市场 获得 国内 中国 去年

Cluster 2：手机 智能 硬件 设备 电视 运动 数据 功能 健康 使用

Cluster 3：数据 平台 市场 学生 app 移动 信息 公司 医生 教育

Cluster 4：企业 招聘 人才 平台 公司 it 移动 网站 安全 信息

Cluster 5：社交 好友 交友 宠物 功能 活动 朋友 基于 分享 游戏

Cluster 6：记账 理财 贷款 银行 金融 p2p 投资 互联网 基金 公司

Cluster 7：任务 协作 企业 销售 沟通 工作 项目 管理 工具 成员

Cluster 8：旅行 旅游 酒店 预订 信息 城市 投资 开放 app 需求

Cluster 9：视频 内容 游戏 音乐 图片 照片 广告 阅读 分享 功能

5 综合实践

案例 4 聚类算法的文本文档实战应用

（3）基本步骤：

- 使用jieba结巴分词对文本进行中文分词，同时插入字典关于关键词；
- scikit-learn对文本内容进行tfidf计算并构造N*M矩阵(N个文档 M个特征词)；
- 再使用K-means/DBSCAN进行文本聚类(省略特征词过来降维过程)；
- 最后对聚类的结果进行简单的文本处理，按类簇归类，也可以计算P/R/F特征值；

Cluster 0：商家 商品 物流 品牌 支付 导购 网站 购物 平台 订单
Cluster 1：投资 融资 美元 公司 资本 市场 获得 国内 中国 去年
Cluster 2：手机 智能 硬件 设备 电视 运动 数据 功能 健康 使用
Cluster 3：数据 平台 市场 学生 app 移动 信息 公司 医生 教育
Cluster 4：企业 招聘 人才 平台 公司 it 移动 网站 安全 信息
Cluster 5：社交 好友 交友 宠物 功能 活动 朋友 基于 分享 游戏
Cluster 6：记账 理财 贷款 银行 金融 p2p 投资 互联网 基金 公司
Cluster 7：任务 协作 企业 销售 沟通 工作 项目 管理 工具 成员
Cluster 8：旅行 旅游 酒店 预订 信息 城市 投资 开放 app 需求
Cluster 9：视频 内容 游戏 音乐 图片 照片 广告 阅读 分享 功能

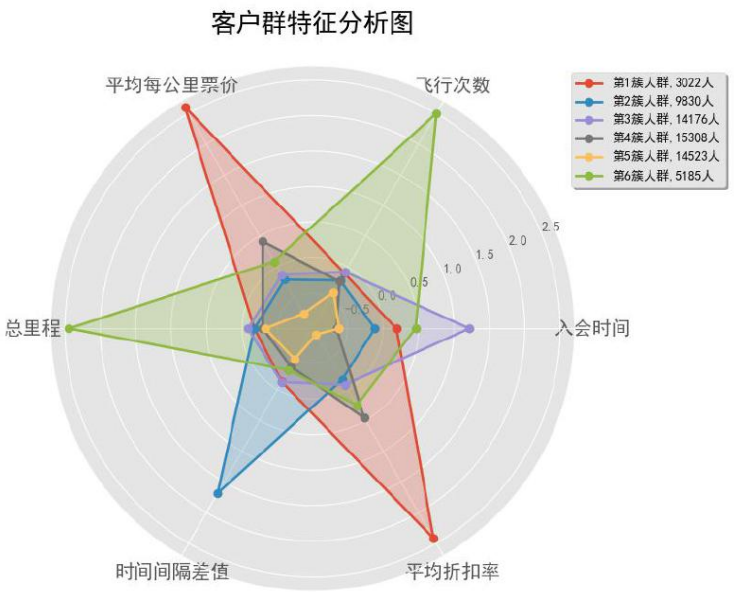
案例5 聚类算法的客户价值分析

（1）简介需求：

通过Sklearn框架和聚类学习相关技巧，通过数据分析手段比较不同类客户的客户价值，并是实现左图。

（2）分析提示：

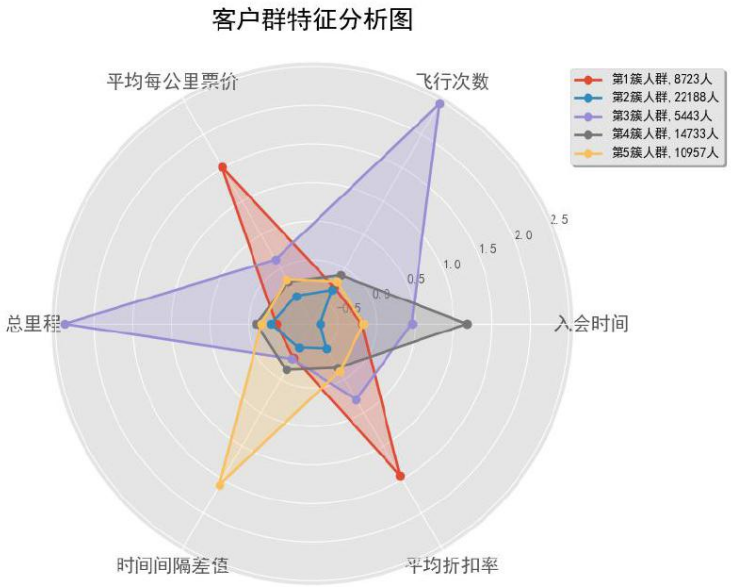
数据载入须先进行预处理，清除无关维度，并合理划分数据及大小。

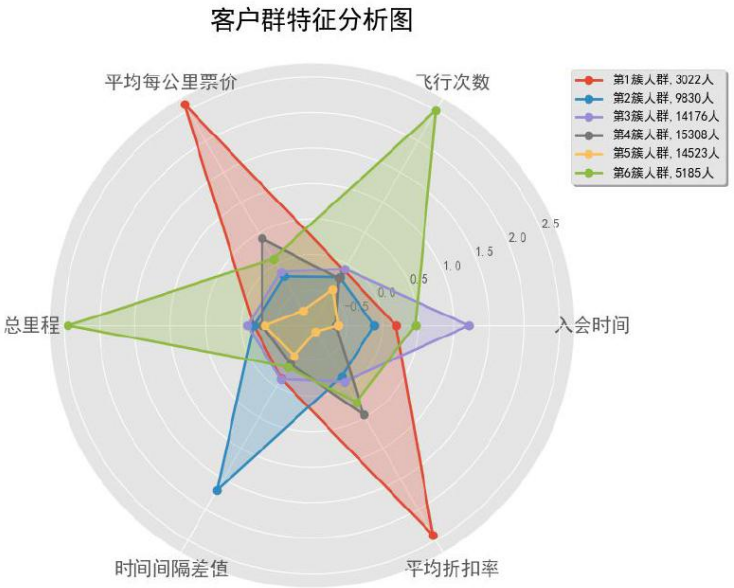


5 综合实践

（3）实现效果：

你也可以尝试不同划分依据及参数选择依据并形成不一样的图表





作业

1. 请梳理出K-means算法流程及常见优化方案

2. 聚类算法的簇评估指标有哪些，你还能提出其他的模型判别标准吗？
   （提示： ROC，混淆矩阵等）

3. 聚类算法中止条件有哪些？

总结

- K-means
  - 原理
    - 给定的样本集，按样本之间的距离大小，将样本集划分为K个簇。
    - 让簇内的点尽量紧密的连在一起，而让簇间的距离尽量的大。
  - 特点
    - 优点
      - 原理简单（靠近中心点），实现容易
      - 聚类效果中上（依赖K的选择）
      - 计算复杂度低O(Nkq)，q为迭代次数
    - 缺点
      - 对离群点，噪声敏感（中心点易偏移）
      - 确定K的个数及初值选取问题
      - 很难发现大小差别很大的簇及进行增量计算
      - 结果不一定是全局最优，只能保证局部最优
  - 改进
    - ISODATA
    - 谱聚类Spectral Clustering
    - 二分K-means；K-means++
    - Kernel K-means；K-MCMC
  - 应用
    - Tensorflow框架下Python实现K-means
    - 应用场景举例：视觉特征中的字典学习
  - 比较
    - 基于层次：BIRCH、CURE、CHAMELEON
    - 基于划分：K-medoids、CLARANS算法
    - 基于密度：DBSCAN、OPTICS、DENCLUE算法
    - 基于网络：STING、CLIQUE、WAVE-CLUSTER
  - 文献
    - [1]Fast and Provably Good Seedings for k-means
    - [2]Approximate K-menas++ in Sublinear Time
    - [3]k-means++：the advantages of careful seeding

黑马程序员
www.itheima.com
传智播客旗下高端IT教育品牌