K-Means算法简介

K-Means算法是集简单和经典于一身的基于距离的聚类算法,采用距离作为相似性的评价指标，即认为两个对象的距离越近，其相似度就越大。该算法认为类簇是由距离靠近的对象组成的，因此把得到紧凑且独立的簇作为最终目标。

K-Means通过迭代寻找k个类簇的一种划分方案，使得用这k个类簇的均值来代表相应各类样本时所得的总体误差最小。

k个类簇具有以下特点：各聚类本身尽可能的紧凑，而各聚类之间尽可能的分开。K-Means算法的基础是最小误差平方和准则。

K-Means算法简介

如果用数据表达式表示，假设簇划分为$(C_1, C_2, \ldots C_k)$，则我们的目标是最小化平方误差E

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|_2^2 \tag{6}$$

C_i中所有样本与C_i质心的距离的平方和

其中μ_i是簇C_i的均值向量，有时也称质心，表达式为

$$\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x \tag{7}$$

C_i中元素的个数，即 $Card(C_i)$

如果我们想直接寻求上式的最小值并不容易，这是一个NP难问题，只能采用启发式的迭代方法

K-Means算法简介

随机选取k个聚类质心点（cluster centroids）为$\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$。

重复下面过程直到收敛:

对于每一个样例i，计算其应该属于的类

样本点

$$c^{(i)} = \arg\min_j \left\| x^{(i)} - \mu_j \right\|^2 \tag{8}$$

⇒得到k个新类

迭代

对于每一个类j，重新计算该类的质心

$$\mu_i = \frac{1}{|C_j|} \sum_{x \in C_j} x \tag{9}$$

⇒得到k个新质心

收敛指如果新计算出来的质心和原来的质心之间的距离小于某一个设置的阈值，表示重新计算的质心的位置变化不大，趋于稳定

收敛条件（迭代的终止条件）

K-Means算法的计算过程

可通过示意图来解释K-Means的计算过程

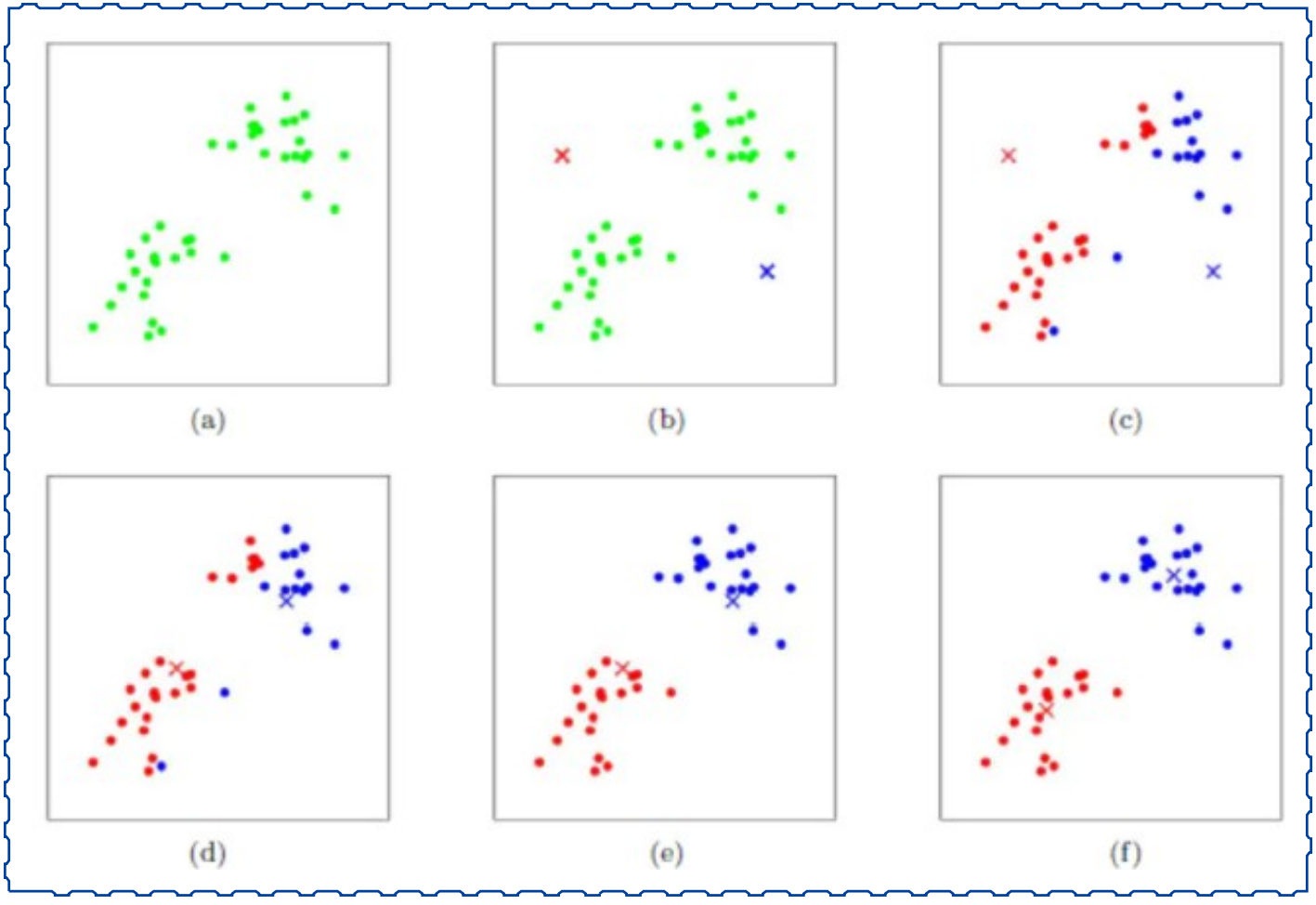


Figure 3: K-Means计算过程示意图

K-Means算法的计算过程

1. 图a表达了初始的数据集，假设$k = 2$。
2. 图b中，我们随机选择了两个类别所对应的类别质心，即图中的红色质心和蓝色质心。
3. 图c表示通过求样本中所有点到这两个质心的距离，并标记每个样本的类别为和该样本距离最小的质心的类别，得到了所有样本点的第一轮迭代后所属的类别。
4. 图d中，我们对我们当前标记为红色和蓝色的点分别求其新的质心新的红色质心和蓝色质心的位置已经发生了变动。
5. 图e和图f重复了我们在图c和图d的过程，即将所有点的类别标记为距离最近的质心的类别并求新的质心。最终我们得到的两个类别如图f所示。

K-Means算法实例的Python实现

本实例采用的是iris数据集：

```
from sklearn import datasets
import matplotlib.pyplot as plt
#加载数据集，是一个字典类似中的Javamap
lris_df = datasets.load_iris()
#挑选出前两个维度作为横轴和纵轴，你也可以选择其他维度
x_axis = lris_df.data[:,0]
y_axis = lris_df.data[:,2]
#绘制散点图
plt.scatter(x_axis, y_axis)
```

K-Means算法实例的Python实现

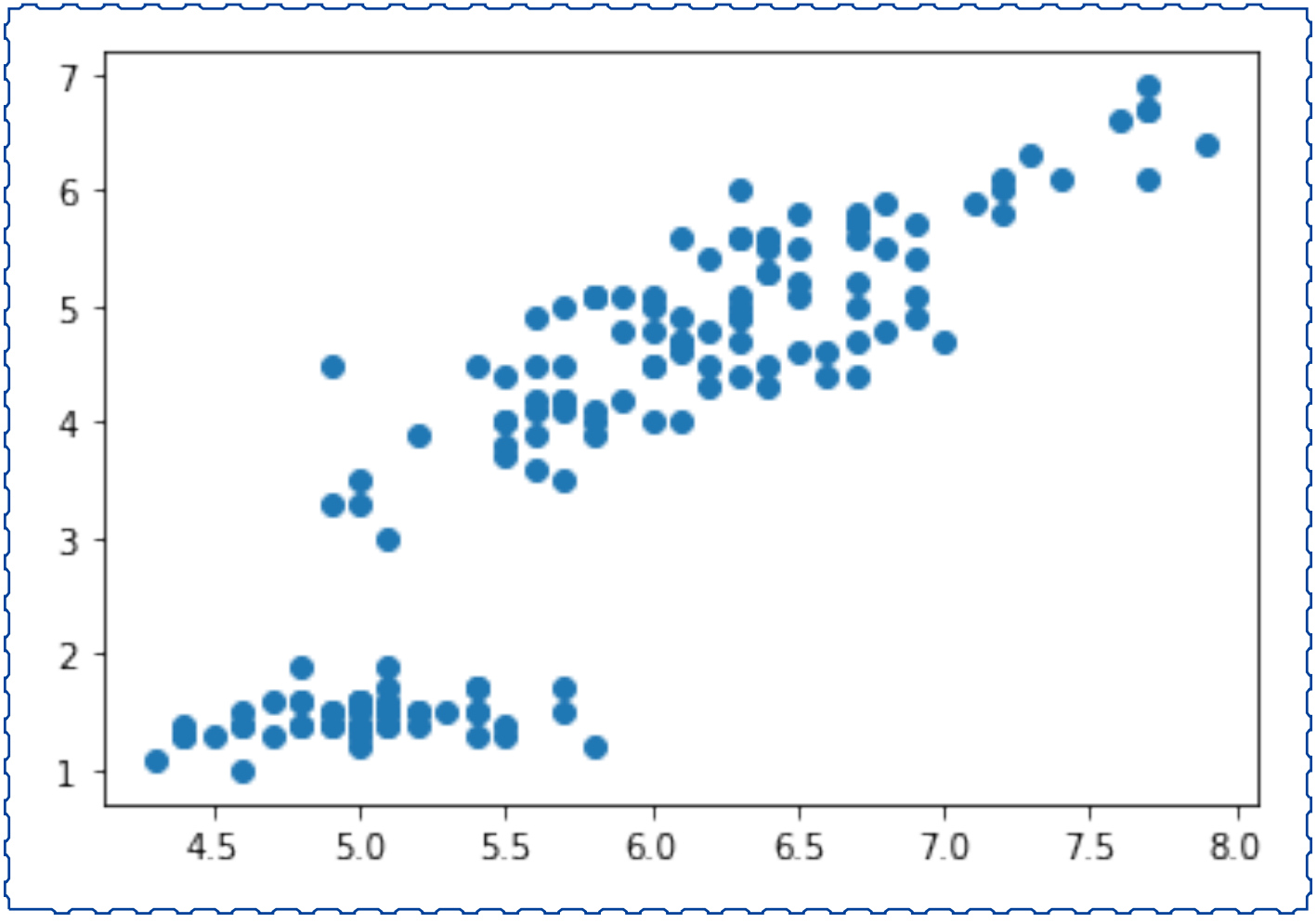


Figure 4: iris数据集分布散点图

K-Means算法实例的Python实现

使用K-Means算法将其分成3类：

```
from sklearn import datasets
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
lris_df = datasets.load_iris()
#挑选出前两个维度作为横轴和纵轴，你也可以选择其他维度
x_axis = lris_df.data[:,0]
y_axis = lris_df.data[:,2]
#调试需要分的类别数
model = KMeans(n_clusters=3)
model.fit(lris_df.data) #训练模型
all_predictions = model.predict(lris_df.data) #预测全部数据
plt.scatter(x_axis, y_axis, c=all_predictions)
plt.show() #打印聚类散点图
```

model.labels_

K-Means算法实例的Python实现

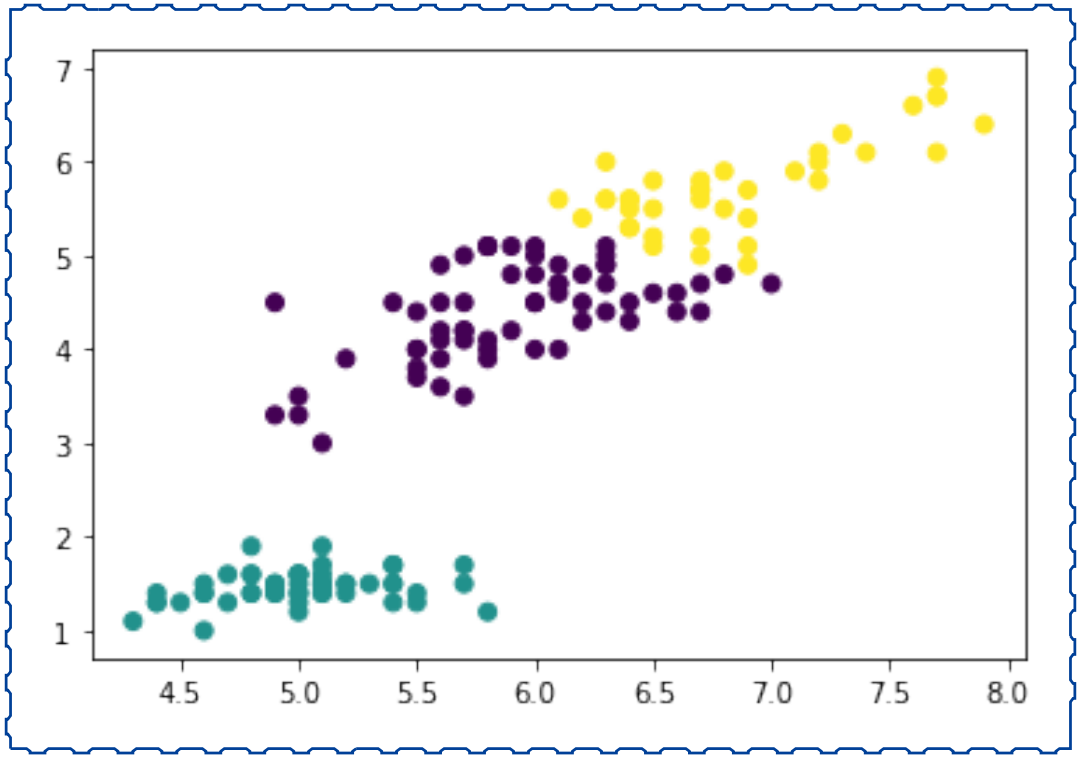


Figure 5: K-Means算法分类结果

- 请你自行尝试分2类或者分4类的情况，看看有什么发现?

层次聚类法简介

层次聚类法又叫系统聚类法，是聚类分析中最常用的一种方法。它的优点在于可以指出由粗到细的多种分类情况，典型的层次聚类结果可由一个聚类图展示出来。

例如，在平面上有 5 个点 w_1, w_2, w_3, w_4, w_5（如图6(a)），其坐标分别为(1,1),(2,3),(3,2),(4,0),(6,1)，可以用聚类图（如图6(b)）来表示聚类结果。

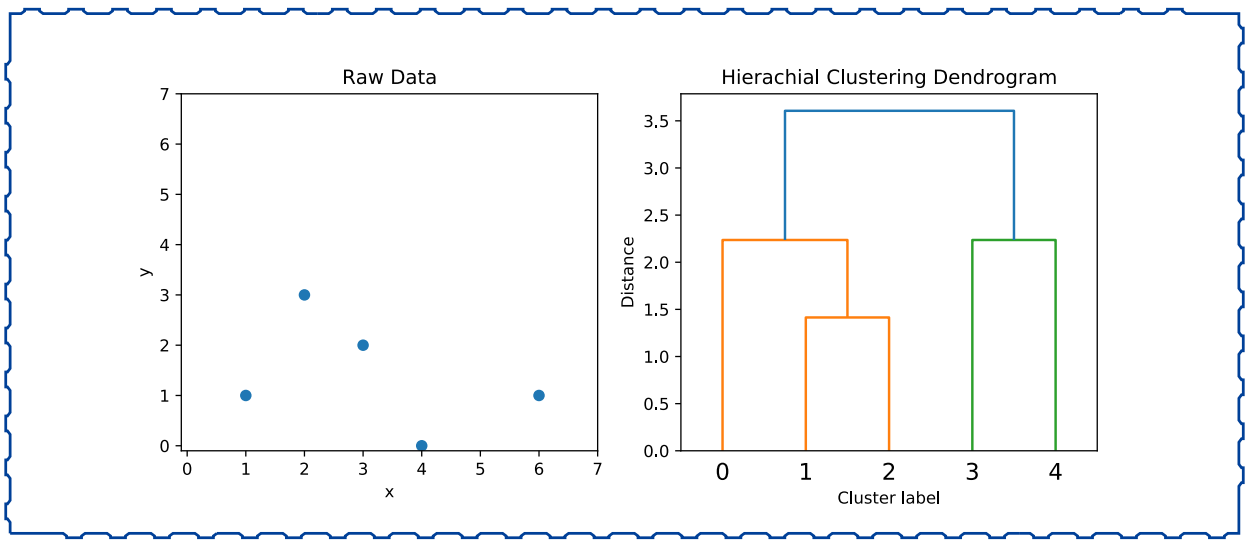


Figure 6: 层次聚类点分布示意图及聚类结果

层次聚类法的计算过程

怎样才能生成这样的聚类图呢？步骤如下：

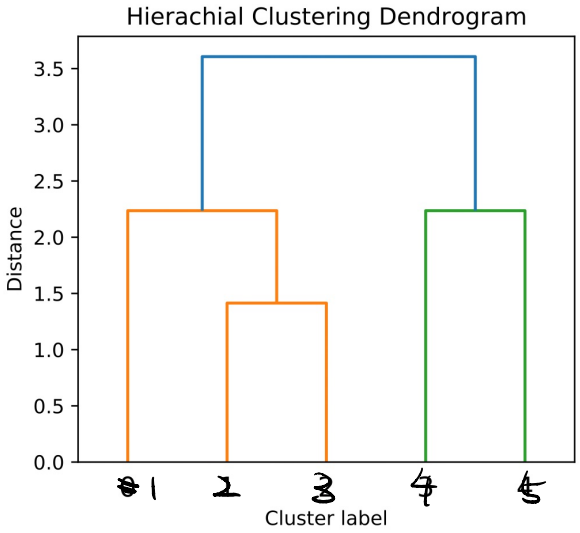
1. 计算n 个样本点两两之间的距离$\{d_{ij}\}$，记为矩阵$D = (d_{ij})_{n\times n}$
2. 首先构造n个类，每一个类中只包含一个样本点，每一类的平台高度均为零；
3. 合并距离最近的两类为新类，并且以这两**类间的距离值**作为聚类图中的平台高度；
4. 计算新类与当前各类的距离，若类的个数已经等于1，转入步骤5，否则，回到步骤3；
5. 画聚类图；
6. 决定类的个数和类。

迭代

w_1 w_2 w_3 w_4 w_5

$(1,1),(2,3),(3,2),(4,0),(6,1)$

$$D = \begin{array}{c} \\ w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{array} \begin{array}{c} \begin{array}{ccccc} w_1 & w_2 & w_3 & w_4 & w_5 \end{array} \\ \begin{bmatrix} 0 & \sqrt{5} & \sqrt{5} & \sqrt{10} & 5 \\ \sqrt{5} & 0 & \sqrt{2} & \sqrt{13} & \sqrt{20} \\ \sqrt{5} & \sqrt{2} & 0 & \sqrt{5} & \sqrt{10} \\ \sqrt{10} & \sqrt{13} & \sqrt{5} & 0 & \sqrt{5} \\ 5 & \sqrt{20} & \sqrt{10} & \sqrt{5} & 0 \end{bmatrix} \end{array}$$



合并 w_2 和 w_3，平台高度为 $\sqrt{2}$

$$D = \begin{array}{c} \\ w_1 \\ w_2 \& w_3 \\ w_4 \\ w_5 \end{array} \begin{array}{c} \begin{array}{cccc} w_1 & w_2 \& w_3 & w_4 & w_5 \end{array} \\ \begin{bmatrix} 0 & \sqrt{5} & \sqrt{10} & 5 \\ \sqrt{5} & 0 & \frac{\sqrt{13}+\sqrt{5}}{2} & \frac{\sqrt{20}+\sqrt{10}}{2} \\ \sqrt{10} & \frac{\sqrt{13}+\sqrt{5}}{2} & 0 & \sqrt{5} \\ 5 & \frac{\sqrt{20}+\sqrt{10}}{2} & \sqrt{5} & 0 \end{bmatrix} \end{array}$$

合并 w_1 和 $w_2 \& w_3$，平台高度为 $\sqrt{5}$

$$D = \begin{array}{c} \\ w_1 \& w_2 \& w_3 \\ w_4 \\ w_5 \end{array} \begin{array}{c} \begin{array}{ccc} w_1 \& w_2 \& w_3 & w_4 & w_5 \end{array} \\ \begin{bmatrix} 0 & \frac{\sqrt{10}+\sqrt{13}+\sqrt{5}}{3} & \frac{5+\sqrt{20}+\sqrt{10}}{3} \\ \frac{\sqrt{10}+\sqrt{13}+\sqrt{5}}{3} & 0 & \sqrt{5} \\ \frac{5+\sqrt{20}+\sqrt{10}}{3} & \sqrt{5} & 0 \end{bmatrix} \end{array}$$

合并 w_4 和 w_5，平台高度为 $\sqrt{5}$

$$D = \begin{array}{c} \\ w_1 \& w_2 \& w_3 \\ w_4 \& w_5 \end{array} \begin{array}{c} \begin{array}{cc} w_1 \& w_2 \& w_3 & w_4 \& w_5 \end{array} \\ \begin{bmatrix} 0 & \frac{\sqrt{10}+5+\sqrt{13}+\sqrt{20}+\sqrt{5}+\sqrt{10}}{6} \\ \sim & 0 \end{bmatrix} \end{array}$$

合并 $w_1 \& w_2 \& w_3$ 和 $w_4 \& w_5$，平台高度为 $\frac{\sqrt{10}+5+\sqrt{13}+\sqrt{20}+\sqrt{5}+\sqrt{10}}{6}$

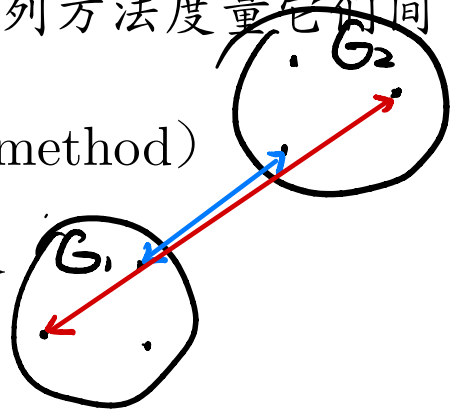
层次聚类法的计算过程

如何计算类与类之间的间距呢?

如果有两个样本类$G1$和$G2$，我们可以用下面的一系列方法度量它们间的距离：

1）最短距离法（nearest neighbor or single linkage method）

$$D(G_1, G_2) = \min_{x_i \in G_1, y_j \in G_2} \{d(x_i, y_j)\}$$



它的直观意义为两个类中最近两点间的距离。

2）最长距离法（farthest neighbor or complete linkage method）

$$D(G_1, G_2) = \max_{\substack{x_i \in G_1 \\ y_j \in G_2}} \{d(x_i, y_j)\}$$

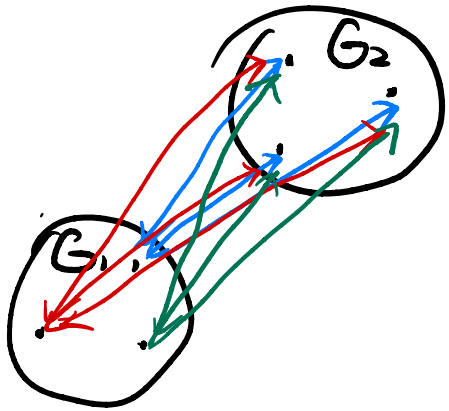
它的直观意义为两个类中最远两点间的距离。

层次聚类法的计算过程

3）重心法（centroid method）

$$D\left(G_1, G_2\right) = d(\bar{x}, \bar{y})$$

其中 $\bar{x}, \bar{y}$ 分别为 G_1, G_2 的重心。



4）类平均法（group average method）

$$D\left(G_1, G_2\right) = \frac{1}{n_1 n_2} \sum_{x_i \in G_1 x_j \in G_2} d\left(x_i, x_j\right)$$

它等于 G_1 ,G_2 中两两样本点距离的平均，式中n_1,n_2 分别为 G_1 ,G_2 中的样本点个数。

层次聚类法计算流程

计算n 个样本点两两之间的距离$\{d_{ij}\}$，记为矩阵$D = (d_{ij})_{n\times n}$

	w_1	w_2	w_3	w_4	w_5
w_1	0	$\sqrt{5}$	$\sqrt{5}$	$\sqrt{10}$	5
w_2	$\sqrt{5}$	0	$\sqrt{2}$	$\sqrt{13}$	$\sqrt{20}$
w_3	$\sqrt{5}$	$\sqrt{2}$	0	$\sqrt{5}$	$\sqrt{10}$
w_4	$\sqrt{10}$	$\sqrt{13}$	$\sqrt{5}$	0	$\sqrt{5}$
w_5	5	$\sqrt{20}$	$\sqrt{10}$	$\sqrt{15}$	0

层次聚类法计算流程

构造n个类，每一个类中只包含一个样本点，每一类的平台高度均为零；合并距离最近的两类为新类，并且以这两**类间的距离值**作为聚类图中的平台高度；
需要合并的两类是：w_2和w_3，合并后类别之间的距离矩阵变为

	w_1	w_2 & w_3	w_4	w_5
w_1	0	$\sqrt{5}$	$\sqrt{10}$	5
$w_2\&w_3$	$\sqrt{5}$	0	$\frac{\sqrt{13}+\sqrt{5}}{2}$	$\frac{\sqrt{20}+\sqrt{10}}{2}$
w_4	$\sqrt{10}$	$\frac{\sqrt{13}+\sqrt{5}}{2}$	0	$\sqrt{5}$
w_5	5	$\frac{\sqrt{20}+\sqrt{10}}{2}$	$\sqrt{5}$	0

层次聚类法计算流程

进一步地，合并w_1, w_2, w_3为一类，w_4, w_5为另一类。合并后类别之间的距离矩阵变为

	w_1 & w_2 & w_3	w_4 & w_5
w_1 & w_2 & w_3	0	$\frac{\sqrt{10}+5+\sqrt{13}+\sqrt{20}+\sqrt{5}+\sqrt{10}}{6}$
w_4 & w_5	$\frac{\sqrt{10}+5+\sqrt{13}+\sqrt{20}+\sqrt{5}+\sqrt{10}}{6}$	0

最后一步，将所有的数据点聚为一类，结束。

层次聚类法实例的Python实现

```
1  import numpy as np
2  from matplotlib import pyplot as plt
3  from scipy.cluster.hierarchy import dendrogram,
       linkage
4  data = np.array([[1,1],
5                   [2,3],
6                   [3,2],
7                   [4,0],
8                   [6,1]])
9  plt.scatter(data[:,0], data[:,1])
10 plt.xlabel('x')
11 plt.ylabel('y')
12 plt.axis('equal') #把两坐标轴的比例设为相等
```

层次聚类法实例的Python实现

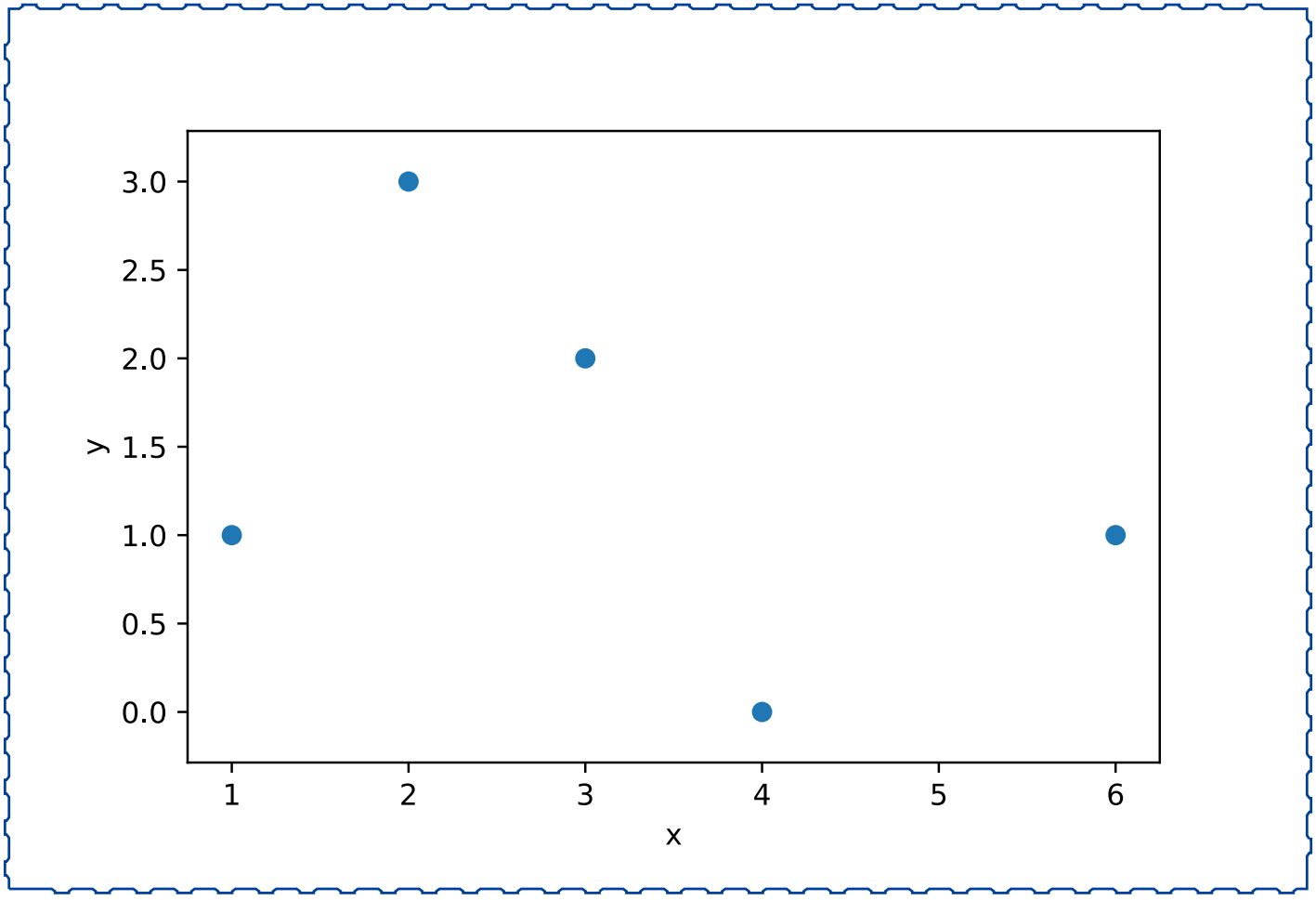


Figure 7: 初始数据分布

层次聚类法实例的Python实现

```
z = linkage(data, "average") #用类平均算法
fig, ax = plt.subplots(figsize=(8,8))
dendrogram(z, leaf_font_size=14) #画图
plt.title("Hierachial Clustering Dendrogram")
plt.xlabel("Cluster label")
plt.ylabel("Distance")
plt.axhline(y=10) #画一条分类线
plt.show()
```

层次聚类法实例的Python实现

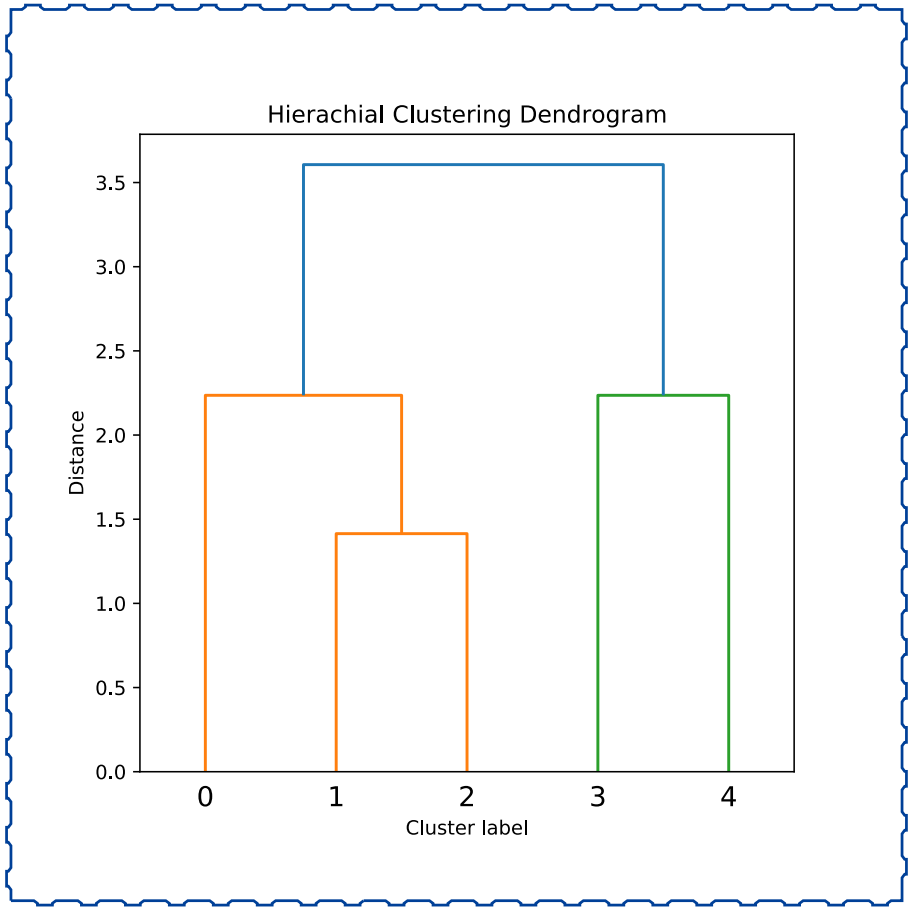


Figure 8: 层次聚类法结果

课后作业

数据集如下，使用K-Means算法对该数据集进行聚类（聚成4类）。

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn import datasets
iris = datasets.load_iris()
X = iris.data[:, :2] # 表示我们取特征空间中的维度数
    目#
print(X.shape)
# 绘制数据分布图
plt.scatter(X[:, 0], X[:, 1], c="red", marker='o',
    label='see')
plt.xlabel('sepal length')
plt.ylabel('sepal width')
plt.legend(loc=2)
```

Thank You

· 聚类分析