聚类算法

Clustering Algorithms

Mr. Black

目录

K-means

K-means

K-means是一种简单的迭代性的聚类算法。对于数据集 $D = \{x_1, x_2, \ldots, x_n\}$，其中 $x_i \in \mathbb{R}^d$，需要指定利用K-means算法对数据划分成 k个簇。对于数据集 D的每个点 x_i 仅属于一个簇 S_i，则K-means算法的目标函数可以表示为：

$$\text{argmin}_S \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|_2^2$$

其中 μ_i是簇 S_i的均值向量。从的目标函数不难看出，K-means是通过一种“紧密程度”的形式对数据进行划分的，衡量这种“紧密程度”一般我们会用到“距离”的概念。距离可以理解为在集合 M上的一个度量（Metric），即

$$dist : M \times M \to \mathbb{R}$$

K-means

对于集合 M中的 x, y, z，下列条件均成立：

1. $dist\left(x, y\right) \geq 0$（非负性）
2. $dist\left(x, y\right) = 0$, 当且仅当$x = y$（同一性）
3. $dist\left(x, y\right) = dist\left(y, x\right)$（对称性）
4. $dist\left(x, z\right) \leq dist\left(x, y\right) + dist\left(y, z\right)$（三角不等式）

对于点 $x = (x_1, x_2, \ldots, x_n)$和点 $y = (y_1, y_2, \ldots, y_n)$，常用的距离为 p阶明可夫斯基距离（Minkowski distance）：

$$dist\left(x, y\right) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

K-means

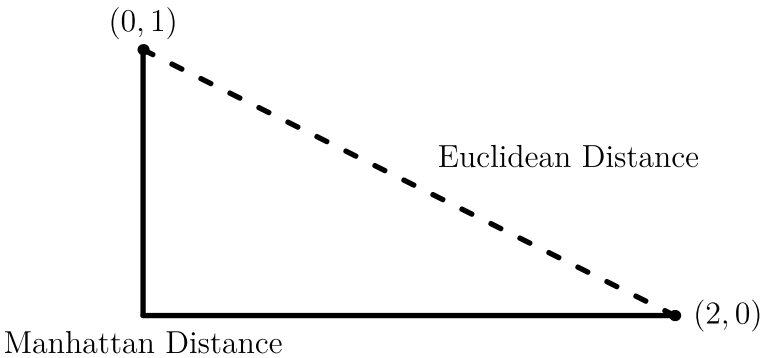
当 $p=1$时，称之为曼哈顿距离（Manhattan distance）或出租车距离：

$$dist_{man}\left(x,y\right)=\sum_{i=1}^{n}\left|x_i-y_i\right|$$

当 $p=2$时，称之为欧式距离（Euclidean distance）：

$$dist_{ed}\left(x,y\right)=\sqrt{\sum_{i=1}^{n}\left(x_i-y_i\right)^2}$$

曼哈顿距离和欧式距离直观比较如图所示：



K-means

对于K-means算法，具体的计算过程如下：

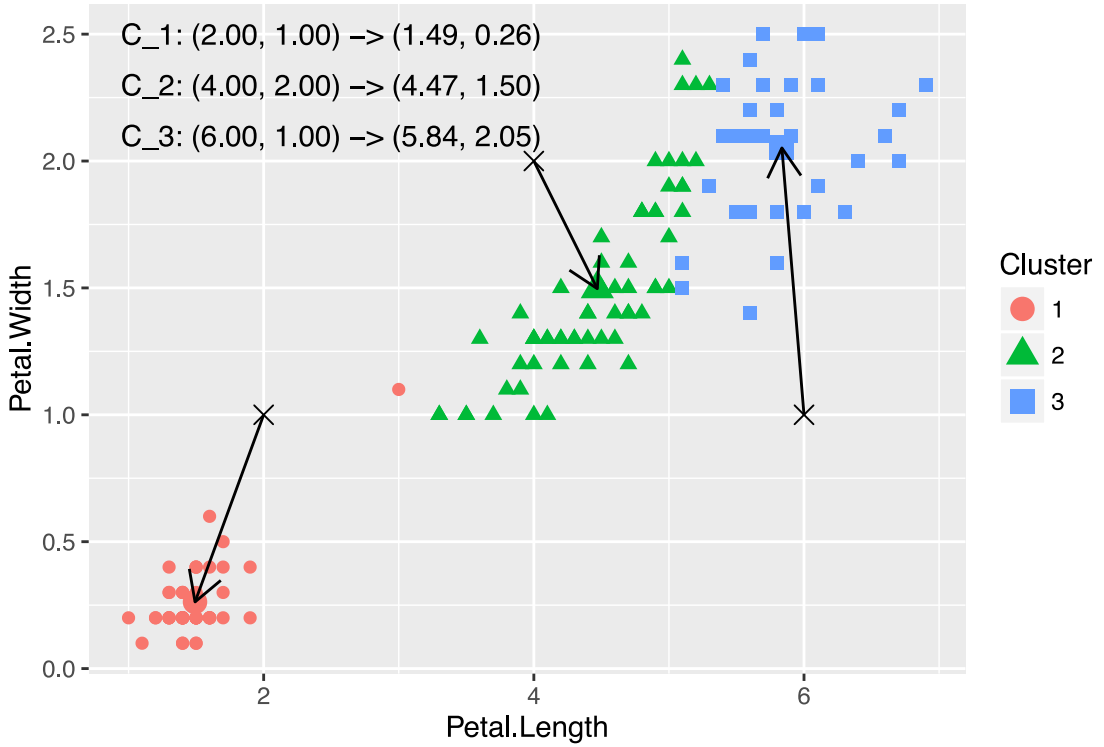
1. 指定簇的个数为 k，并随机设置 k个簇的中心，对于簇 S_i其中心为 μ_i。
2. 计算数据集 $D = \{x_1, x_2, \ldots, x_n\}$中的所有点 x_j到每个簇的中心 μ_i的距离 $dist\left(x_j, \mu_i\right)$。
3. 对于点 x_j，从其到每个簇中心 μ_i的距离中选择距离最短的簇作为本轮计算中该点所隶属的簇。
4. 对于隶属于同一个簇的样本 D_{S_i}，计算这些样本点的中心，作为该簇新中心 μ_i'。
5. 重复执行步骤2到步骤4直至簇中心不再发生变化或超过最大迭代次数。

通过上述步骤的计算，K-means算法可以将样本点划分为 k个簇，并得到每个簇的最终中心 μ_i。

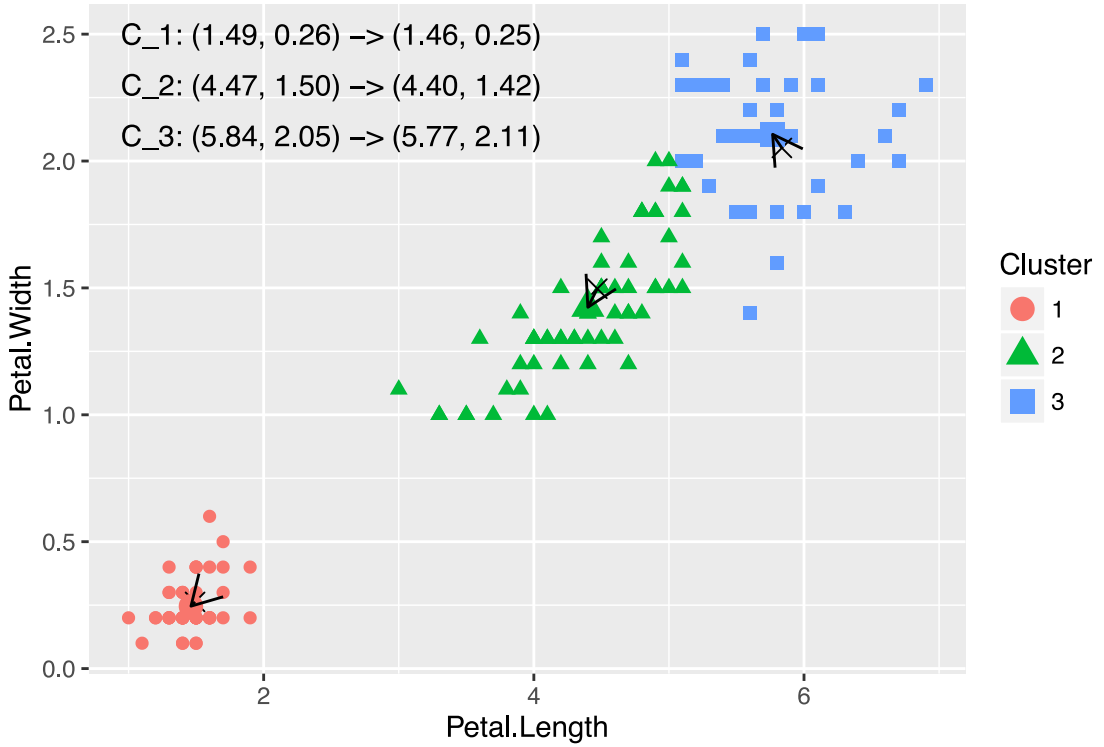
K-means

利用K-means算法，我们对iris数据集进行聚类分析。iris数据集包含了Sepal.Length，Sepal.Width，Petal.Length，Petal.Width以及花的种类共5列数据。为了能够更直观的演示，我们进采用Petal.Length和Petal.Width两列数据。K-means是一种无监督的学习算法，因此我们并没有先验知识知道数据最适合分为几个簇，同时K-means算法又是一个对于聚类中心初始点敏感的算法，因此同样为了便于演示效果，在此我们设置簇的个数 $k=3$，3个簇对应的初始中心点分别为 $\mu_1=(2,1)$, $\mu_2=(4,2)$, $\mu_3=(6,1)$。

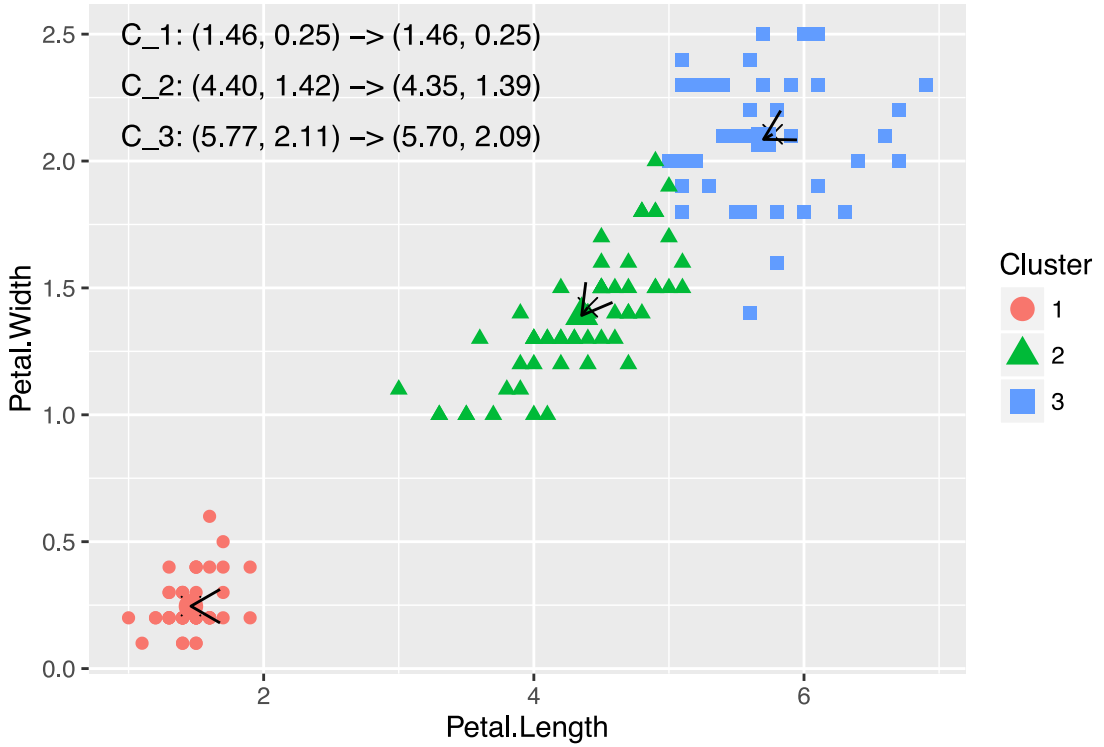
K-means


C_1: (2.00, 1.00) –> (1.49, 0.26)
C_2: (4.00, 2.00) –> (4.47, 1.50)
C_3: (6.00, 1.00) –> (5.84, 2.05)
Petal.Width
Petal.Length
2.5
2.0
1.5
1.0
0.5
0.0
2
4
6
Cluster
1
2
3

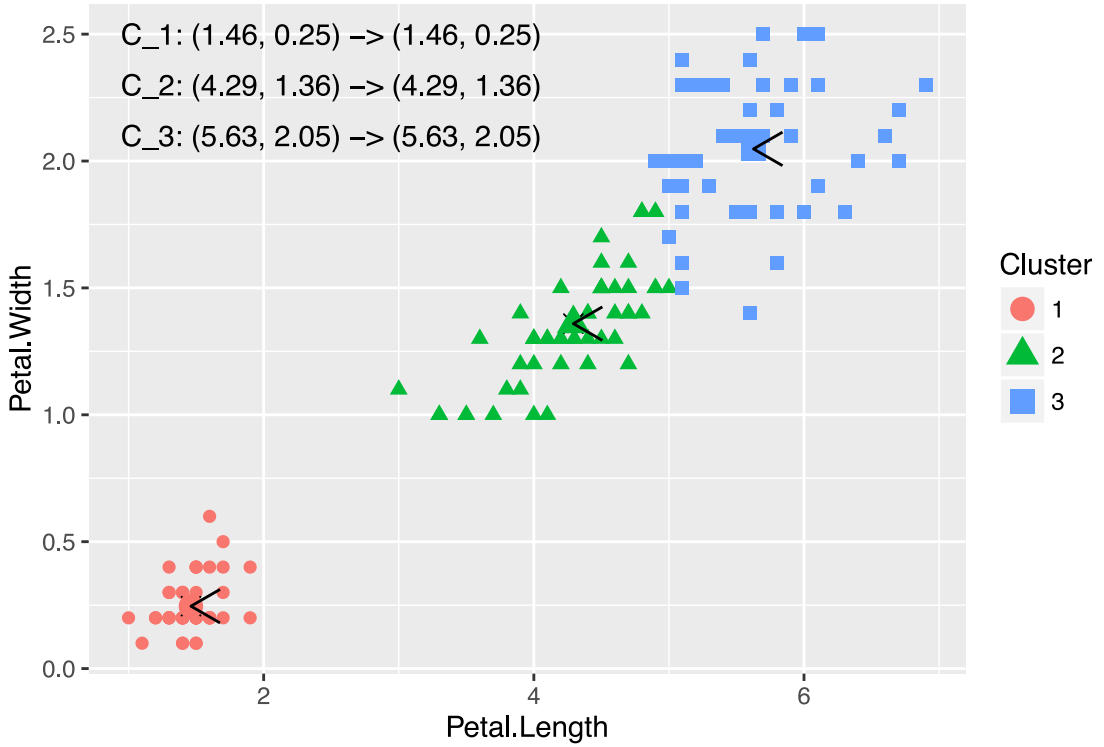
K-means


C_1: (1.49, 0.26) –> (1.46, 0.25)
C_2: (4.47, 1.50) –> (4.40, 1.42)
C_3: (5.84, 2.05) –> (5.77, 2.11)
Petal.Width
Petal.Length
2.5
2.0
1.5
1.0
0.5
0.0
2
4
6
Cluster
1
2
3

K-means


C_1: (1.46, 0.25) –> (1.46, 0.25)
C_2: (4.40, 1.42) –> (4.35, 1.39)
C_3: (5.77, 2.11) –> (5.70, 2.09)
Petal.Width
Petal.Length
2.5
2.0
1.5
1.0
0.5
0.0
2
4
6
Cluster
1
2
3

K-means


C_1: (1.46, 0.25) –> (1.46, 0.25)
C_2: (4.29, 1.36) –> (4.29, 1.36)
C_3: (5.63, 2.05) –> (5.63, 2.05)
Petal.Width
Petal.Length
2.5
2.0
1.5
1.0
0.5
0.0
2
4
6
Cluster
1
2
3

K-means

第1轮，第2轮，第3轮和第7轮（最终轮）计算得出的结果。其中每幅图左上角3组坐标分别表示了3个簇的中心更新前和更新后的位置。图中的x号即为更新前簇的中心，箭头指向的方向即为更新后簇的中心，每轮计算中隶属不同簇的样本点利用颜色和形状加以了区分。

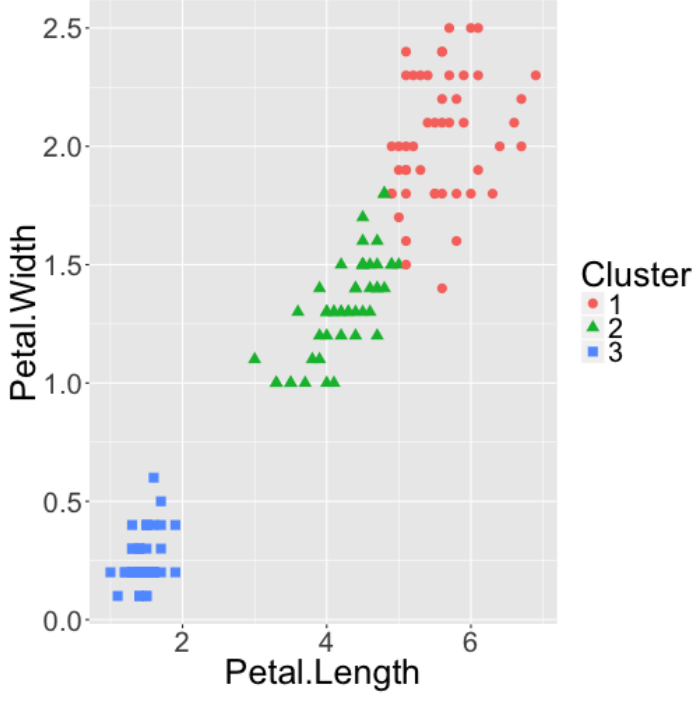
K-means算法在一般数据集上可以的到较好的聚类效果，但同时也存在若干问题：

1. K-means算法需要预先设置聚类个数 k。
2. K-means是一个对于簇中心点起始位置敏感的算法，设置不同的簇中心点的起始位置可能得到不同的聚类结果。
3. 噪音数据对K-means算法的聚类结果影响较大。
4. 只能发现球状簇。

K-means

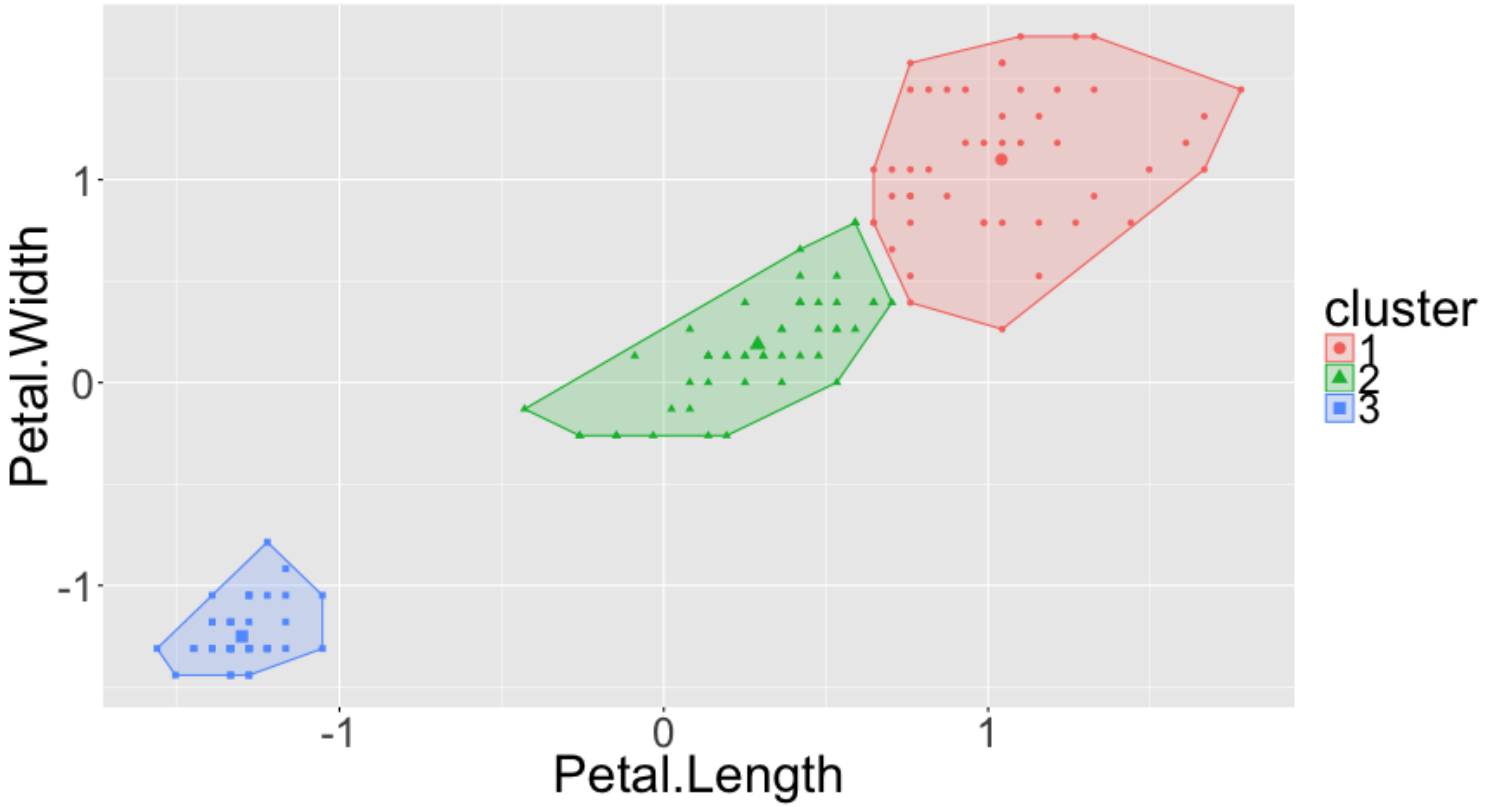
```
kmeans(x, centers, iter.max = 10, nstart = 1,
       algorithm = c("Hartigan-Wong", "Lloyd", "Forgy",
                     "MacQueen"), trace=FALSE)
```

```
iris_ <- iris[, 3:4]
iris_kmeans <- kmeans(iris_, 3,
    nstart=25)
iris_res <- dplyr::bind_cols(
    iris_,
    Cluster=as.factor(
        iris_kmeans$cluster))
p <- ggplot(iris_res,
            aes(Petal.Length,
                Petal.Width)) +
    geom_point(aes(
        color = Cluster,
        shape = Cluster))
print(p)
```



K-means

```
factoextra::fviz_cluster(iris_kmeans, data=iris_, geom=c('point'))
```



层次聚类

层次聚类

层次聚类（hierarchical clustering）不同于K-means那种基于划分的聚类，通过对数据集在不同层次上进行划分，直至达到某种条件。层次聚类根据分层的方法不同，可以分为凝聚（agglomerative）层次聚类和分裂（divisive）层次聚类。

AGNES（Agglomerative NESting）算法是一种凝聚层次聚类算法，其基本思想如下：

1. 将数据集中每个样本作为一个簇。
2. 在每一轮计算中，找出两个距离最近的簇进行合并，生成一个新的簇。
3. 重复步骤2，直至达到预设的聚类簇的个数。

层次聚类

因此，对于AGNES算法而言，最关键的是如何计算两个簇之间的距离，对于簇C_i和簇C_j，常用的距离计算方法有：

- 最小距离，即两个簇内部样本点之间距离的最小值：

$$dist_{min} = \min\{dist\left(x, y\right) | x \in C_i, y \in C_j\}$$

- 最大距离，即两个簇内部样本点之间距离的最大值：

$$dist_{max} = \max\{dist\left(x, y\right) | x \in C_i, y \in C_j\}$$

- 平均距离，即两个簇内部样本点之间距离的均值：

$$dist_{avg} = \frac{1}{|C_i||C_j|} \sum_{x \in C_i} \sum_{y \in C_j} dist\left(x, y\right)$$

- 重心距离，即两个簇重心之间的距离：

$$dist_{med} = dist\left(Median_{C_i}, Median_{C_j}\right)$$

层次聚类

DIANA（Divisive Analysis）算法指一种分裂层次聚类算法，其基本思想如下：

1. 将数据集中全部样本作为一个簇。
2. 在每一轮计算中，对于“最大”的簇 C，找到 C中与其他点的平均相异度最大的点 p_0，将其放在一个新的簇 C_{new}中，剩余的点此时所组成的簇为 C_{old}。
3. 在簇 C_{old}找到一个距离簇 C_{new}最近，且距离小于到簇 C_{old}的点 p_i，并将其加入到簇 C_{new}中。
4. 重复步骤3，直至无法找到符合条件的点 p_i，此时得到两个新簇 C_{old}和 C_{new}。
5. 重复步骤2和步骤3，直至达到预设的聚类簇的个数。

在DIANA算法中，衡量一个簇 C的大小，一般利用簇的直径，即簇中任意两个样本之间距离的最大值；衡量簇 C中一个点 p的平均相异度，一般利用该点到簇中其他点距离的平均值。

层次聚类

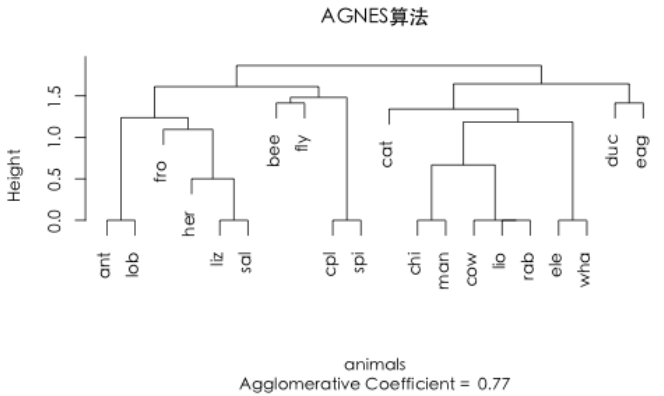
以R中cluster扩展包中的animals数据集为例，animals数据集记录了20中不同昆虫和动物的6种属性值，数据示例如表所示，列分别为：温血， 会飞，脊椎动物，濒危，群居动物，有毛发：

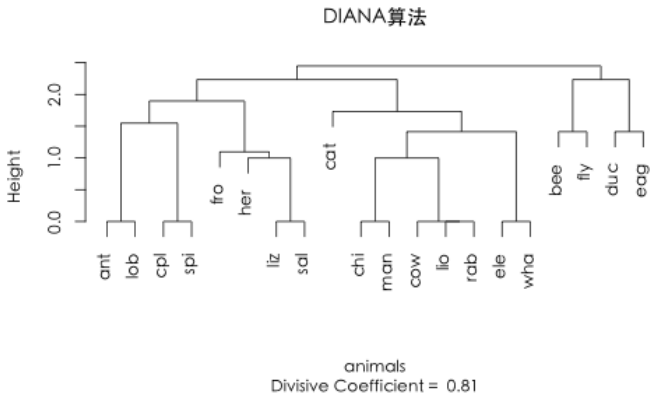
样本\特征	war	fly	ver	end	gro	hai
ant	1	1	1	1	2	1
bee	1	2	1	1	2	2
cat	2	1	2	1	1	2
...	...	...	...	...	...	...
spi	1	1	1	NA	1	2
wha	2	1	2	2	2	1

层次聚类

利用AGNES和DIANA算法对animals数据集进行层次聚类分析，可以得到层次聚类分析的树状图（dendrogram），如图所示：

```
require(cluster)
data('animals')

animals_agnes <- agnes(animals)
plot(animals_agnes,
     which.plot=2,
     main='AGNES算法')

animals_diana <- diana(animals)
plot(animals_diana,
     which.plot=2,
     main='DIANA算法')
```





基于密度的聚类

基于密度的聚类

基于密度的聚类（density-based clustering）是一种通过样本的稠密程度划分聚类簇的方法。不同于基于距离的K-means和层次聚类方法往往只能生成球状的聚类簇，基于密度的聚类可以发现任意形状的聚类簇。

DBSCAN（density-based spatial clustering of applications with noise）是一种基于密度的聚类算法。DBSCAN算法最重要的两个参数为 ϵ和 $MinPts$，两个参数分别确定了领域半径和定义了核心点的阈值，通过这两个参数可以刻画样本分布的稠密程度。对于数据集 $D=\{x_1, x_2, \ldots, x_n\}$，引入如下概念和记号：

- ϵ邻域（ ϵ neighborhood）

$$N_\epsilon(x) = \{y \in X | dist(x, y) \leq \epsilon\}$$

对于 $x \in D$，称 $N_\epsilon(x)$为 x的 ϵ邻域。

基于密度的聚类

- 密度（density）

$$\rho(x) = |N_\epsilon(x)|$$

对于 $x \in D$，称 $\rho(x)$为 x的密度。

- 核心点（core point）

对于 $x \in D$，若 $\rho(x) \geq MinPts$，则称 x为一个核心点。假设 D中所有核心点构成的集合为 D_{core}，记 $D_{n-core} = D \setminus D_{core}$为所有非核心点的集合。

- 边界点（border point）

对于 $x \in D_{n-core}$，且 $\exists y \in D$，满足

$$y \in N_\epsilon(x) \cap D_{core}$$

即点 x所在的 ϵ邻域中存在核心点，则称 x为 D的边界点，记所有的边界点的集合为 D_{border}。

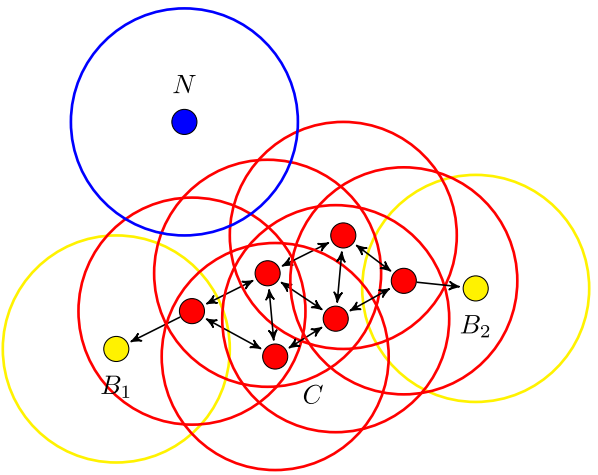
基于密度的聚类

- 噪音点（noise point）

记 $D_{noise} = D \setminus (D_{core} \cup D_{border})$，对于 $x \in D_{noise}$，则称 x为噪音点。

核心点，边界点和噪音点示例如右图所示：

其中 C为6个核心点， B_1和 B_2为2个边界点， N为1个噪音点。



- 密度直达（directly density-reachable）

对于 $x, y \in D$，若 $x \in D_{core}$，并且 $y \in N_{\epsilon}(x)$，则称 y由 x密度直达。

基于密度的聚类

- 密度可达（density-reachable）

若存在一个序列 $p_1, p_2, \ldots, p_m \in D$，满足 p_{i+1}由 p_i密度直达，则称 p_m由 p_1密度可达。

- 密度相连（density-connected）

对于 $x, y, z \in D$，若 y和 z均由 x密度可达，则称 y和 z密度相连。

- 簇（cluster）

对于非空子集 $C \in D$，如果称 C为一个簇，则对于 $x, y \in D$满足：

1. 连接性（connectivity）：对于 $x, y \in C$，则 x和 y密度相连。
2. 最大性（maximality）：对于 $x \in C$，且 y由 x密度可达，则 $y \in C$。

根据如上概念，DBSCAN算法的基本为：从一个核心点 x出发，寻找到 x密度可达的所有样本点的集合 $X = \{x' \in D | x'\textbf{由}x\textbf{密度可达}\}$，则 X即为一个满足要求的簇。

基于密度的聚类

DBSCAN算法如下：

Algorithm 1 DBSCAN算法

Require: 数据集d，参数$(\epsilon, minpts)$
Ensure: 簇划分$c = \{c1, c2, ..., c_k\}$

procedure DBSCAN$(d, \epsilon, minpts)$
  初始化核心对象集合：$i \leftarrow \varnothing$
  for $i = 1$ to n **do**
    对于样本x_i，生成ϵ邻域$n_\epsilon(x_i)$
    if $\rho(x_i) \geq minpts$ **then**
      $i \leftarrow i \cup \{x_i\}$
    end if
  end for
  初始化聚类个数：$k \leftarrow 0$
  初始化未访问到集合：$u \leftarrow d$
  #接下文
end procedure

procedure DBSCAN$(d, \epsilon, minpts)$
  #接上文
  while $i \neq \varnothing$ **do**
    当前为访问的样本集合：$u_{old} \leftarrow u$
    随机选取一个核心点$p \in i$，并初始化一个队列$q \leftarrow \{p\}$
    $u \leftarrow u \setminus \{p\}$
    while $q \neq \varnothing$ **do**
      $q \leftarrow q$的队首
      if $\rho \geq minpts$ **then**
        $r \leftarrow n_\epsilon(q) \cap u$
        $q \leftarrow q \cup r$
        $u \leftarrow u \setminus r$
      end if
    end while
    $k \leftarrow k + 1$
    生成簇$c_k \leftarrow u_{old} \setminus u$
    $i \leftarrow i \setminus c_k$
  end while
  return $c = \{c_1, c_2, ..., c_k\}$
end procedure

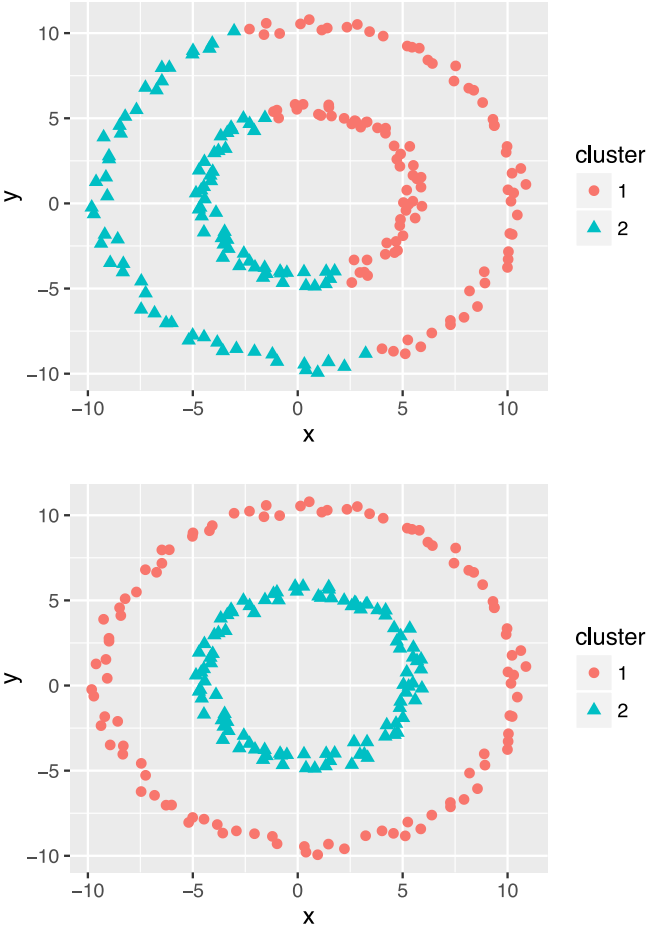
基于密度的聚类

相比K-means算法，DBSCAN算法有如下优势：

1. 不需要事先指定簇的个数 k。
2. 可以发现任意形状的簇。
3. 对噪音数据不敏感。

尽管相比K-means，DBSCAN算法有很多优势，但是对于不同的数据集，DBSCAN算法的参数 ϵ和 $MinPts$有时很难选取和优化。

一个非球形簇的数据分别利用DBSCAN算法和K-means算法进行聚类分析，对比结果如图所示：



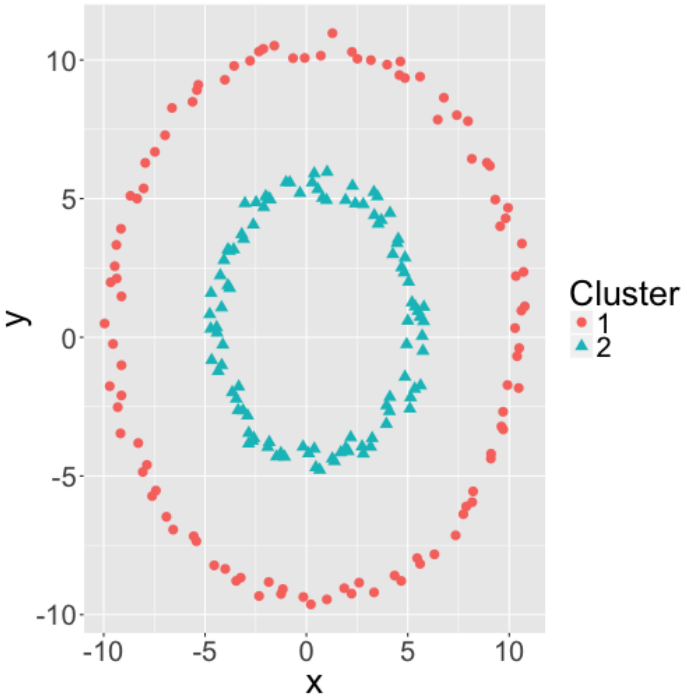
基于密度的聚类

```
dbscan(x, eps, minPts = 5, weights = NULL, borderPoints = TRUE, ...)
```

```
library(dbscan)

# 构造数据
set.seed(123)
a <- seq(0, 2*pi, by = pi/50)
x_1 <- 10 * cos(a) + runif(length(a), 0, 1)
y_1 <- 10 * sin(a) + runif(length(a), 0, 1)
x_2 <- 5 * cos(a) + runif(length(a), 0, 1)
y_2 <- 5 * sin(a) + runif(length(a), 0, 1)
points <- data.frame(
    x = c(x_1, x_2),
    y = c(y_1, y_2)
)

# 聚类
points_dbscan <- dbscan(points, eps=3.5, minPts=10)
```

基于密度的聚类

```
# 构造绘图数据
points_plt <- dplyr::bind_cols(
    points,
    Cluster=as.factor(
        points_dbscan$cluster))

# 绘图
p <- ggplot(points_plt,
            aes(x, y)) +
    geom_point(
        aes(color = Cluster,
            shape = Cluster))
print(p)
```



Thanks