聚类分析

> 参考文献：
>
> 1. 【数学建模算法与应用第二版】
> 2. 【R语言实战（第二版）】

模型的原理——建模

原理部分

聚类分析是一种**数据归约**技术，旨在揭露一个数据集中观测值的**子集**。它可以把大量的观测值归约为若干个类。这里的类被定义为若干个观测值组成的群组，群组内观测值的**相似度**比群间相似度高。一般把数据聚类归纳为一种非監督式學習。

> 这不是一个精确的定义，从而导致了**各种**聚类方法的出现。

样本的相似性度量：

对于定量变量（通常所说的**连续量**），最常用的是闵式距离（Minkowski）距离，即

$$d_q(\mathbf{x},\mathbf{y})=[\sum_{k=1}^{p}|x_k-y_k|^q]^{\frac{1}{q}},q>0$$

当 $q=1,2$ 或者 $q\to+\infty$ 则分别得到

1. 绝对值距离:
2. 欧几里得距离:
3. 切比雪夫距离：$d_\infty(\mathbf{x},\mathbf{y})=\max\limits_{1\le k\le p}|x_k-y_k|$

在闵式距离中最常用的是欧氏距离，它的主要优点是当坐标轴进行正交旋转时，欧氏距离是保持不变的。

> 值得注意的是在采用 Minkowski 距离时，一定要采用**相同量纲**的变量。故而首先要进行数据的标准化处理‘

在采用Minkowski 距离时， 还应尽可能地避免变量的**多重相关性** （multicollinearity）。多重相关性所造成的**信息重叠**，会片面强调某些变量的重要性。

> 有研究表明主成分分析不能消除多变量多重相关性的影响。这会导致参数估计不够准确。相应的解决方案可以采取**岭回归**：岭回归是一种专用于共线性数据分析的有偏估计回归方法，实质上是一种改良的最小二乘估计法，通过放弃最小二乘法的无偏性，以损失部分信息、降低精度为代价，获得回归系数更为符合实际、更可靠的回归方法。原理待续~

由于Minkowski 距离的这些缺点，一种改进的距离就是**马氏距离**（Mahalanobis），定义如下：

$$d(x,y)=\sqrt{(x-y)^T\Sigma^{-1}(x-y)}$$

其中 x,y 为 p 维总体 Z 的观测值，Σ 为其协方差矩阵，实际中 Σ 往往是不知道的，常常需要用**样本协方差**来**估计**。马氏距离对一切**线性变换**是不变的，故不受**量纲**的影响。

类与类之间的相似性度量

给定样本类 G_1,G_2 ，有以下一系列方法

一、最短距离法（单联动）： （Nearest Neighbor or Single Linkage Method）

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

它的直观意义为两个类中**最近两点**间的距离

二、最长距离法（全联动）： （farthest neighbor or complete linkage method）

它的直观意义为两个类中最远两点间的距离。

三、重心法（质心）：

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

四、类平均法（平均联动）：(group average method)

$$D(G_1,G_2)=\frac{1}{n_1n_2}\sum_{x_i\in G_1}\sum_{x_j\in G_2}d(x_i,x_j)$$

五、离差平方和法（Ward法）： （sum of squares method）

$$D(G_1, G_2) = D_{12} - D_1 - D_2$$

> 离差平方和法最初是由 Ward 在 1936 年提出，后经Orloci 等人1976 年发展起来的，故又称为Ward 方法。

聚类分析根据分类对象的不同可以分为 Q 型和 R 型两大类。

1. Q 型聚类分析的作用：对样本进行分类
    - 利用多个变量（指标）对样本进行分类
    - 所得结果比传统的定性分析更细致、全面
2. R 型聚类分析的作用：对变量（指标）进行分类处理
    - 了解变量之间及**变量组合之间的亲疏关系**
    - 根据聚类结果选择**主要变量**进行回归分析或者 R 型聚类分析

    > 可见实质上是起到降维的作用

Q 型聚类

最常用的两种 Q 型聚类方法是**层次聚类**（hierarchical agglomerative clustering）和**划分聚类**（partitioning clustering）。

1. **层次**聚类中，每一个观测值自成一类，这些类每次两两合并，直到所有的类被聚成一类为止。
2. 在划分聚类中，首先指定类的个数K，然后观测值被随机分成K类，再重新形成聚合的类。

R 型聚类法结果可用一个聚类图（树状图）展示出来，树状图应该从下往上读，它展示了这些条目如何**被结合**成类，高度刻度代表了该高度类之间合并的**判定值**。

层次聚类法（系统聚类法）

设样本空间 $\Omega = w_1, w_2, \cdots, w_n$

1. 计算样本点俩俩之间的距离，得距离矩阵 $D = (d_{ij})_{n \times n}$
2. 首先构造 n 个类，每一类平台高度均为 0。
3. 依据类间相似性度量算法选取两类合为一类，并以此两类间的距离值作为新的平台高度

> 显然平台高度依赖于类间度量算法，而类间度量算法又依赖于两点间距离测度的选取。

4. 更新类别与距离矩阵，若类别已经等于1则转至5，否则转至3
5. **画聚类图**
6. 决定类的个数与类

> 类别的个数得自己定，一般参考广泛采用的一些指标

当需要嵌套**聚类**和**有意义的层次结构**时，层次聚类或许特别有用。在生物科学中这种情况很常见。比如需要解释聚类的意义时可以分析不同变量之间的共性。

在某种意义上分层算法是贪婪的，一旦一个观测值被分配给一个类，它就不能在后面的过程中被重新分配。

另外，层次聚类**难以应用**到有**数百甚至数千观测值的大样本**中。不过划分方法 可以在大样本情况下做得很好。

划分聚类分析

在划分方法中，观测值被分为 K 组并根据**给定的规则**改组成**最有粘性**的类

K 均值聚类

算法步骤：

1. 选择 k 个中心点（随机选择 k 行）；
2. 对每个点确定其聚类中心点。（分配到离他最近的中心点）；
3. 重新计算每类中的新的聚类中心点，即该类数据点的平均值（质心）
4. 分配每个数据到它最近的中心点；
5. 重复步骤(3)和步骤(4)直到所有的观测值不再被分配或是达到最大的迭代次数

> K-means面对的第一个问题是如何保证收敛，可以证明的是K-means完全可以保证收敛性。

对于每一个样例 i，计算其应该属于的类

$$c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2$$

对于每一个类 j，重新计算其质心

$$\mu_j := \frac{\sum_{i=1}^{m} \mathbf{1}\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} \mathbf{1}\{c^{(i)} = j\}}$$

> 不像层次聚类方法，K均值聚类要求你事先确定要提取的聚类个数。另外，在K均值聚类中，类中**总的平方值**对聚类数量的曲线可能是有帮助的。可 根据图中的弯曲选择适当的类的数量。

围绕中心点的划分（PAM）

因为K均值聚类方法是基于**均值**的，所以它对**异常值是敏感**的。一个更稳健的方法是围绕中心点的划分（PAM）。与其用质心（**变量均值向量**）表示类，不如用一个最有代表性的观测值来表示（称为中心点）。K均值聚类一般使用欧几里得距离，而PAM可以使用任意的距离来计算。因此，**PAM可以容纳混合数据类型，并且不仅限于连续变量。**

1. 随机选择 K个观测值（每个都称为中心点）；
2. 计算观测值到各个中心的**距离/相异性**；
3. 把每个观测值分配到最近的中心点；
4. 计算每个中心点到每个观测值的距离的总和（总成本）
5. **选择一个该类中不是中心的点，并和中心点互换；**
6. 重新把每个点分配到距它最近的中心点；
7. 再次计算总成本；
8. 如果总成本比步骤(4)计算的总成本少，把新的点作为中心点；
9. 重复步骤(5)～(8)直到中心点不再改变。

> 有点类似于单纯形法中基变量的替换，关键是如何寻找新的替换点

R 型聚类分析

在系统分析或评估过程中，为避免遗漏某些重要因素，往往在一开始选取指标时，尽可能**多地**考虑所有的相关因素。而这样做的结果，则是变量过多，变量间的相关度高，给系统分析与建模带来很大的不便。因此，人们常常希望能研究**变量间的相似关系**，按照变量的相似关系把它们聚合成若干类，进而找出影响系统的**主要因素。**

变量相似性度量

一、相关系数：

计变量 x_j 的取值为 $(x_{1j}, x_{2j}, \cdots, x_{nj})^T \in R^n, j = 1, 2, \cdots, m$ 。则变量 x_j, x_k 相关性度量：

$$r_{jk} = \frac{\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k)}{[\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)^2 \sum_{i=1}^{n}(x_{ik} - \bar{x}_k)^2]^{1/2}}$$

在对变量进行聚类分析时，利用相关系数矩阵是最多的。

二、夹角余弦：

$$r_{jk} = \frac{x_j \cdot x_k}{||x_j|| \cdot ||x_k||}$$

变量聚类法

类似于样本集合聚类分析中最常用的**最短距离法、最长距离法**等，变量聚类法采用了与系统聚类法相同的思路和过程。

其中单一变量间的距离：

$$d_{jk} = 1 - |r_{jk}| \quad or \quad d_{jk}^2 = 1 - r_{jk}^2$$

基于密度的聚类算法

是为了挖掘有**任意形状特性**的类别而发明的。此算法把一个类别视为数据集中大于某阈值的一个区域。DBSCAN和OPTICS是两个典型的算法。

DBSCAN是一种基于密度的聚类算法，这类密度聚类算法一般**假定类别可以通过样本分布的紧密程度决定。**

> 同一类别的样本，他们之间的紧密相连的，也就是说，在该类别任意样本周围不远处一定有同类别的样本存在。

通过将紧密相连的样本划为一类，这样就得到了一个聚类类别。通过将所有各组紧密相连的样本划为各个不同的类别，则我们就得到了最终的所有聚类类别结果。

DBSCAN，英文全写为 **Density-based spatial clustering of applications with noise**，是在1996年由Martin Ester, Hans-Peter Kriegel , Jörg Sander及Xiaowei Xu提出的聚类分析算法，这个算法是以**密度**为本的：给定某空间里的一个点集合，这算法能把附近的点分成一组（有很多相邻点的点），并标记出位于低密度区域的局外点（最接近它的点也十分远），DBSCAN是其中一个**最常用**的聚类分析算法，也是其中一个科学文章中最常引用的。

密度定义

DBSCAN 是基于一组邻域来描述样本集的紧密程度的，参数 $(\epsilon, MinPts)$ 用来描述邻域的样本分布紧密程度。其中，ϵ 描述了某一样本的邻域距离阈值，$MinPts$ 描述了某一样本的距离为 ϵ 的邻域中样本个数的阈值。

设样本定义为 $D = (x_1, x_2, \cdots, x_n)$，则 DBSCAN 具体的密度描述如下：

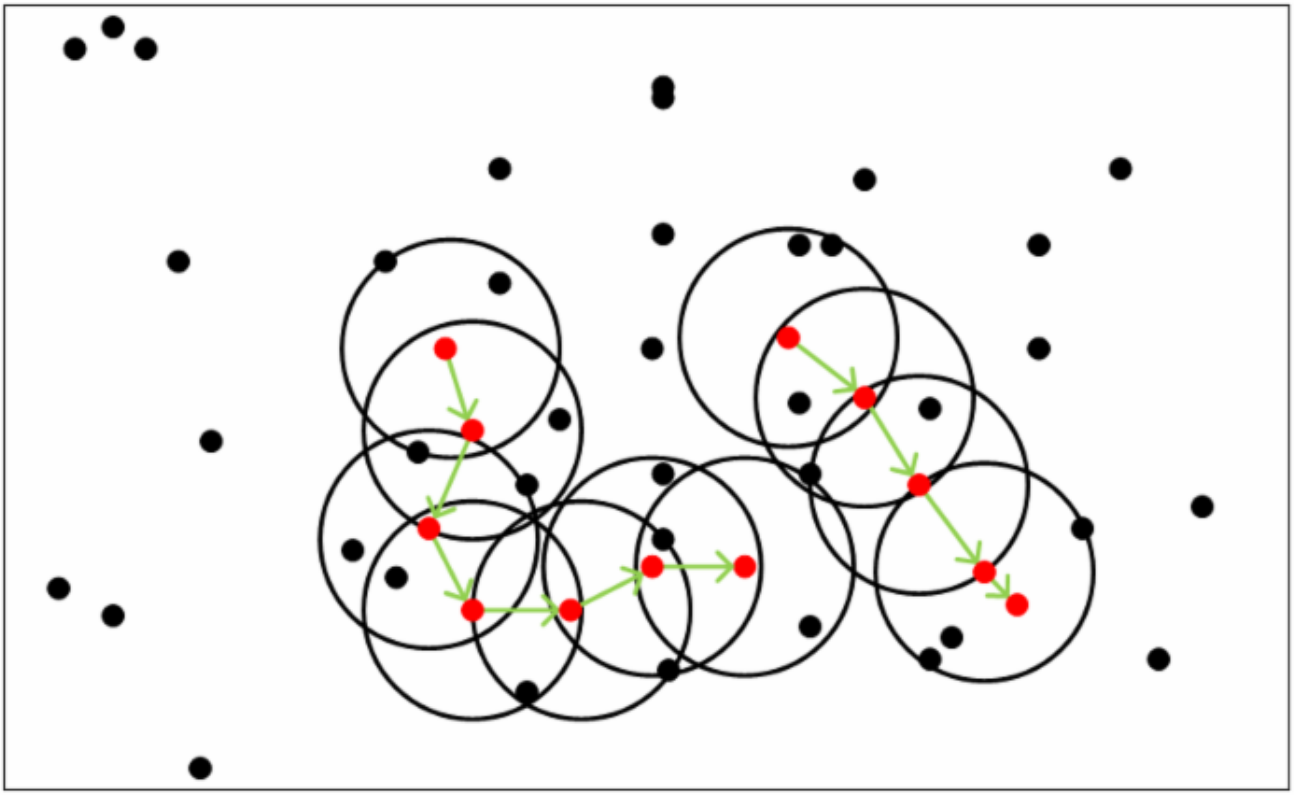
ϵ **邻域**：对于 $x_j \in D$，其 ϵ 邻域包含样本集 D 中与 x_j 的距离不大于 ϵ 的子样本集

$$N_\epsilon(x_j) = \{x_i \in D | distance(x_i, x_j) \leq \epsilon\}$$

核心对像：对于任一样本 $x_j \in D$，如果其 ϵ 邻域对应的 $N_\epsilon(x_j)$ 至少包含 $MinPts$ 个样本，即如果

$$|N_\epsilon(x_j)| \geq MinPts$$

则 x_j 是核心对象。



DBSCAN密度聚类思想与算法

由密度可达关系导出的**最大密度相连的样本集合**，即为我们最终聚类的一个类别，或者说一个簇。

那么怎么才能找到这样的簇样本集合呢？DBSCAN使用的方法很简单，它任意选择一个没有类别的核心对象作为**种子**，然后找到所有这个核心对象能够密度可达的样本集合，即为一个聚类簇。接着继续选择另一个没有类别的核心对象去寻找密度可达的样本集合，这样就得到另一个聚类簇。一直运行到所有核心对象都有类别为止。

算法：

Step 1 初始化

1. 给定参数 ϵ 和 $\mathcal{M}$.

2. 生成 $N_\epsilon(i), i = 1, 2, \cdots, N$.

3. 令 $k = 1$; $m_i = 0, i = 1, 2, \cdots, N$.

4. 令 $I = \{1, 2, \cdots, N\}$.

Step 2 生成 *cluster* 标记数组.

WHILE ($I \neq \emptyset$)

{

从 I 中任取一个元素 i, 并令 $I := I \setminus \{i\}$.

IF ($m_i = 0$) // 即 i 号节点还没有被处理过

{

(1) 初始化 $T := N_\epsilon(i)$

(2) 若 $|T| < \mathcal{M}$, 则令 $m_i = -1$. (暂时将 i 号节点标记为**噪音点**)

(3) 若 $|T| \geq \mathcal{M}$ (即 i 为核心点), 则

(3.1) 令 $m_i = k$. (将 i 号节点归属于第 k 个聚类)

(3.2) *WHILE* ($T \neq \emptyset$)

{

(a) 从 T 中任取元素 j, 并令 $T := T \setminus \{j\}$.

(b) 若 $m_j = 0$ 或 -1, 则令 $m_j = k$.

(c) 若 $|N_\epsilon(j)| \geq \mathcal{M}$(即 j 为核心点), 则令 $T := T \cup N_\epsilon(j)$.

}

(3.3) 令 $k = k + 1$(第 k 个聚类已经完成, 开始下一个聚类).

}

}

注 5.2.1 *Wiki* 上提供的关于 *DBSCAN* 算法伪代码 *([5])*

```
DBSCAN(D, eps, MinPts)
   C = 0
   for each unvisited point P in dataset D
      mark P as visited
      NeighborPts = regionQuery(P, eps)
```

基本上这就是DBSCAN算法的主要内容了，但是我们还是有三个问题没有考虑。

1. 第一个是一些异常样本点或者说少量游离于簇外的样本点，这些点不在任何一个核心对象在周围，在DBSCAN中，我们一般将这些样本点标记为噪音点。
2. 第二个是距离的**度量问题**，即如何计算某样本和核心对象样本的距离。在DBSCAN中，一般采用**最近邻思想**，采用某一种距离度量来衡量样本距离，比如欧式距离。这和KNN 分类算法的最近邻思想完全相同。对应少量的样本，寻找最近邻可以直接去计算所有样本的距离，**如果样本量较大，则一般采用 KD 树或者 球树 来快速的搜索最近邻。**如果对于最近邻的思想，距离度量，KD树和球树不熟悉，建议参考另一篇文章K近邻法(KNN)原理小结。
3. 第三种问题比较特殊，某些样本可能到两个核心对象的距离都很小，但是这两个核心对象由于不是密度直达，又不属于同一个聚类簇，那么如果界定这个样本的类别呢？一般来说，此时DBSCAN采用先来后到，先进行聚类的类别簇会标记这个样本为它的类别。也就是说BDSCAN的算法不是完全稳定的算法。

K-means 与 Bdscan 算法比较

一般来说，如果数据集是稠密的，并且数据集不是凸的，那么用DBSCAN会比K-Means聚类效果好很多。如果数据集不是稠密的，则不推荐用DBSCAN来聚类。

> 所谓凸的，即指同一类集合构成的集合近似为凸集

聚类算法评价指标

样本点有标签时

Rand index(兰德指数)(RI)

$$RI = \frac{a+b}{C_2^{n_{samples}}}$$

未完待续~

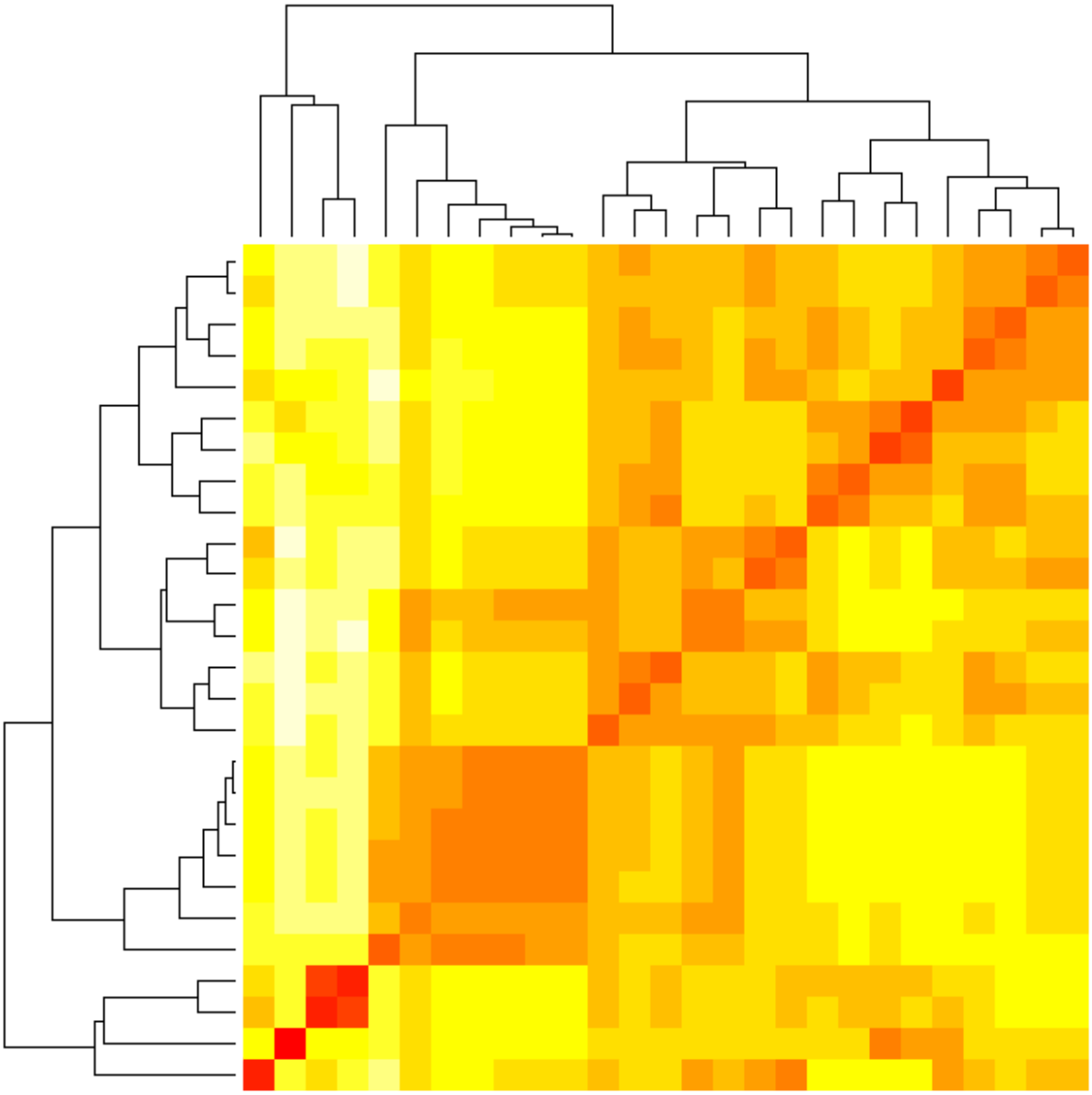
可移植程序实现——编程

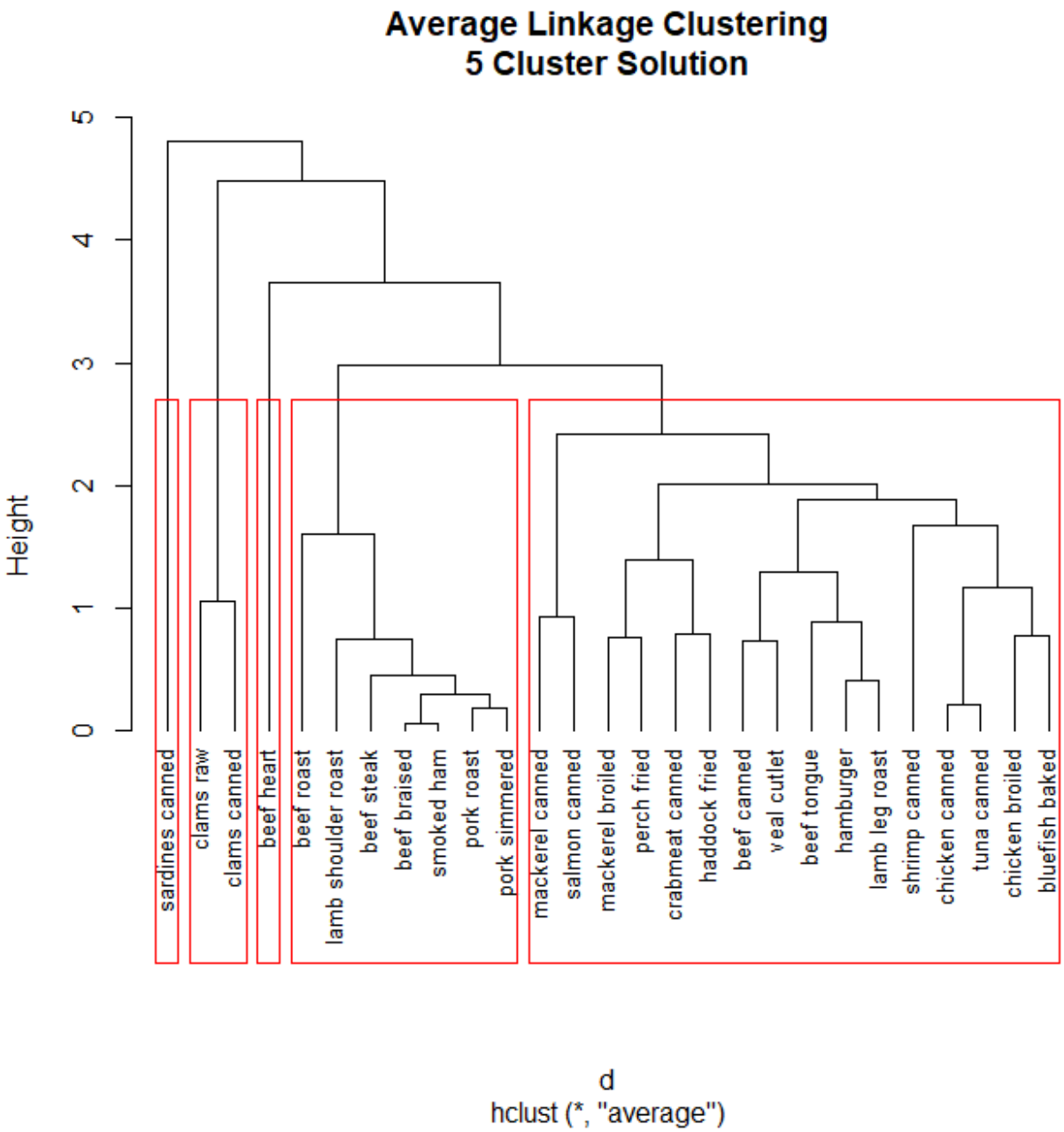
层次、系统、直接聚类

输入: 二维矩阵的聚类样本， ***行表示样本，列表示指标***

```
# 设置路径
setwd("C:/Users/16053/Documents/R/18guosai/聚类分析/")
# 读取数据
nutrient<-read.csv(file="nutrient.csv",header = TRUE)
row.names(nutrient) <- nutrient$X # 第一列改为行名
nutrient<-nutrient[,-1] # 删除第一列


# 标准差法（z-score）进行标准化处理 （无量纲化的一种，均值为0，标准差为1。还有极值差法，【0,1】区间，但要知道最大最小值）
nutrient.scaled <- scale(nutrient)

# 计算距离矩阵
d <- dist(nutrient.scaled,method = "euclidean")
## method 可选：manhattan—曼哈顿距离、maximum—切比雪夫距离、minkowski—闵式距离
as.matrix(d)[1:4,1:4] #  展现前几个变量的距离

# 绘制距离热力图
heatmap(as.matrix(d),labRow = F, labCol = F)

# 选取聚类方法
fit.average <- hclust(d, method="average")
## method 可选：ward.D、ward.D2、single、complete、average、mcquitty（质心）
plot(fit.average, hang=-1, cex=.8, main="Average Linkage Clustering")

# 选取聚类个数—依赖包中的评价标准
library(NbClust)
nc <- NbClust(nutrient.scaled, distance="euclidean",
              min.nc=2, max.nc=15, method="average")
par(opar)
table(nc$Best.n[1,])
barplot(table(nc$Best.n[1,]),
        xlab="Numer of Clusters", ylab="Number of Criteria",
        main="Number of Clusters Chosen by 26 Criteria")
```

```
# 确定最终模型
clusters <- cutree(fit.average, k=5) # 5 参考之前评价标准给出
table(clusters)
aggregate(nutrient, by=list(cluster=clusters), median)
aggregate(as.data.frame(nutrient.scaled), by=list(cluster=clusters),
          median)
plot(fit.average, hang=-1, cex=.8,
     main="Average Linkage Clustering\n5 Cluster Solution")
rect.hclust(fit.average, k=5)
```

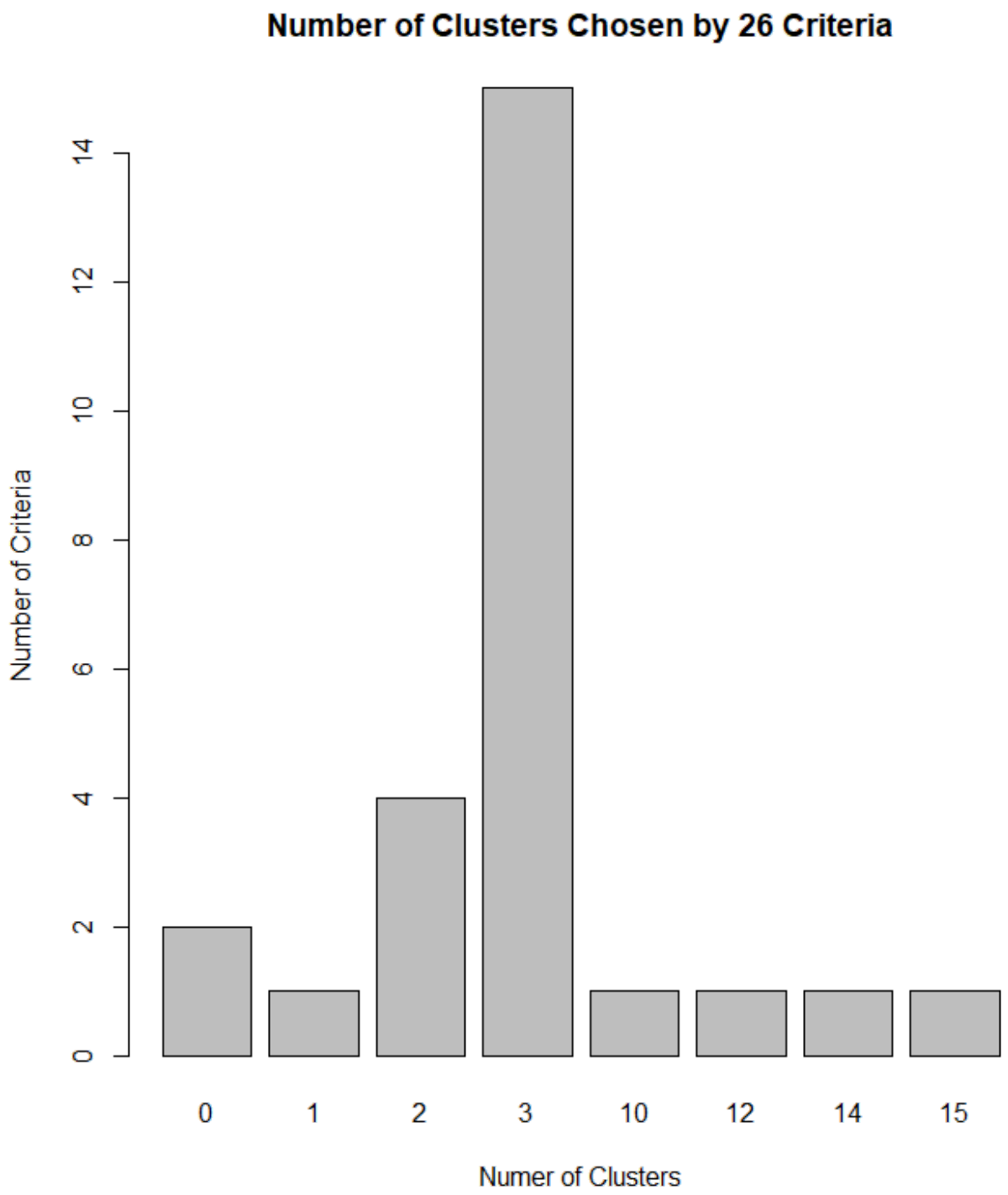




K-means 聚类

```
##library(rattle)
##data(wine, package="rattle")
# 此程序包中没有数据集 wine.data 下载后存于文件夹data中的 csv 文件
wine<-read.csv(file="wine.csv",header = FALSE);wine
# 备份类别属性，用于ggplot可视化
shape_wine<-wine$V1

df <- scale(wine[-1]) ;df # 对去掉类别属性的一列进行Z-score标准化
```

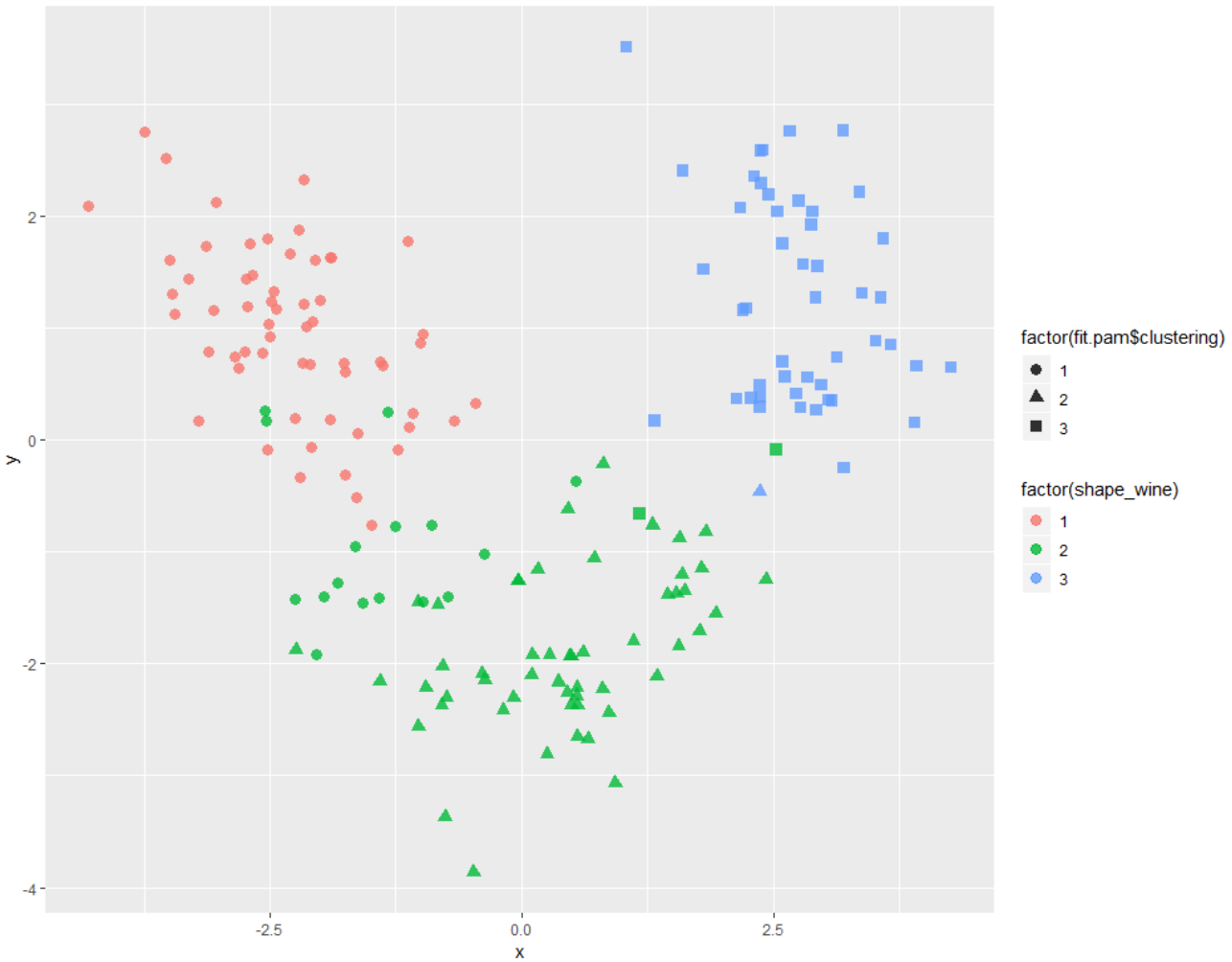
```r
# Plot function for within groups sum of squares by number of clusters
wssplot <- function(data, nc=15, seed=1234){
  wss <- (nrow(data)-1)*sum(apply(data,2,var))
  for (i in 2:nc){
    set.seed(seed)
    wss[i] <- sum(kmeans(data, centers=i)$withinss)}
  plot(1:nc, wss, type="b", xlab="Number of Clusters",
       ylab="Within groups sum of squares")}

wssplot(df)    # 看图初步判断聚类个数
library(NbClust) # 用指标说话，判断聚类个数
set.seed(1234)
nc <- NbClust(df, min.nc=2, max.nc=15, method="kmeans")
par(opar)
table(nc$Best.n[1,])
barplot(table(nc$Best.n[1,]),
        xlab="Numer of Clusters", ylab="Number of Criteria",
        main="Number of Clusters Chosen by 26 Criteria")
set.seed(1234)
fit.km <- kmeans(df, 3, nstart=25)
fit.km$size
fit.km$centers    # 聚类中心
aggregate(wine[-1], by=list(cluster=fit.km$cluster), mean)# 统计信息

# evaluate clustering
ct.km <- table(wine$V1, fit.km$cluster)
ct.km   # 统计的总体个数分错的情况，对角则完全准确
library(flexclust)
randIndex(ct.km) # 兰德指数

# ggplot
d.kmeans<- dist(df)    # 距离矩阵
mds=cmdscale(d.kmeans,k=2,eig=T) # MDS 缩放降维至2维，方便绘图
x = mds$points[,1]
y = mds$points[,2]
library(ggplot2)
p=ggplot(data.frame(x,y),aes(x,y))
p+geom_point(size=3,alpha=0.8,aes(colour=factor(shape_wine),shape=factor(fit
.km$cluster)))
```


Number of Clusters Chosen by 26 Criteria
Number of Criteria
0
2
4
6
8
10
12
14
0
1
2
3
10
12
14
15
Numer of Clusters

基于中心点的划分

```
#基于中心点的划分 PAM
wine<-read.csv(file="wine.csv",header = FALSE);wine
shape_wine<-wine$V1
df <- scale(wine[-1]) ;df

library(cluster)
set.seed(1234)
fit.pam <- pam(wine[-1], k=3, stand=TRUE)
fit.pam$medoids
clusplot(fit.pam, main="Bivariate Cluster Plot")

# evaluate clustering

ct.pam <- table(wine$V1, fit.pam$clustering)
```

```
ct.pam
randIndex(ct.pam)

# ggplot
d.center<- dist(df)
mds=cmdscale(d.center,k=2,eig=T)
x = mds$points[,1]
y = mds$points[,2]
library(ggplot2)
p=ggplot(data.frame(x,y),aes(x,y))
p+geom_point(size=3,alpha=0.8,aes(colour=factor(shape_wine),shape=factor(fit
.pam$clustering)))
```



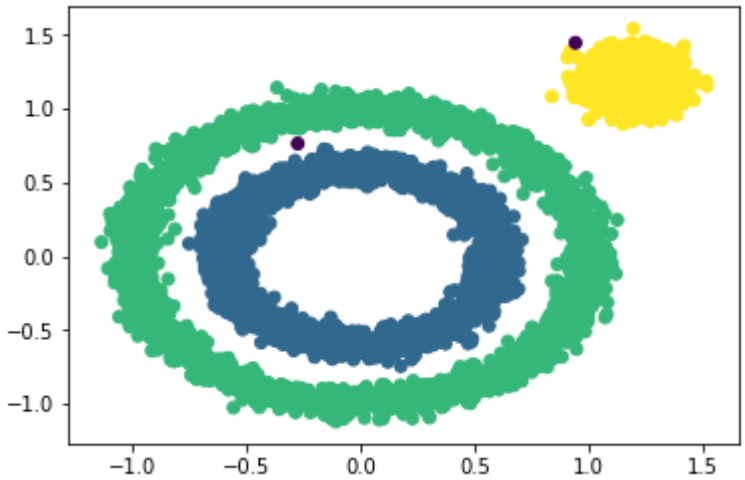
> 此处中心点的结果没有 kmeans 好

dbscan 聚类求解

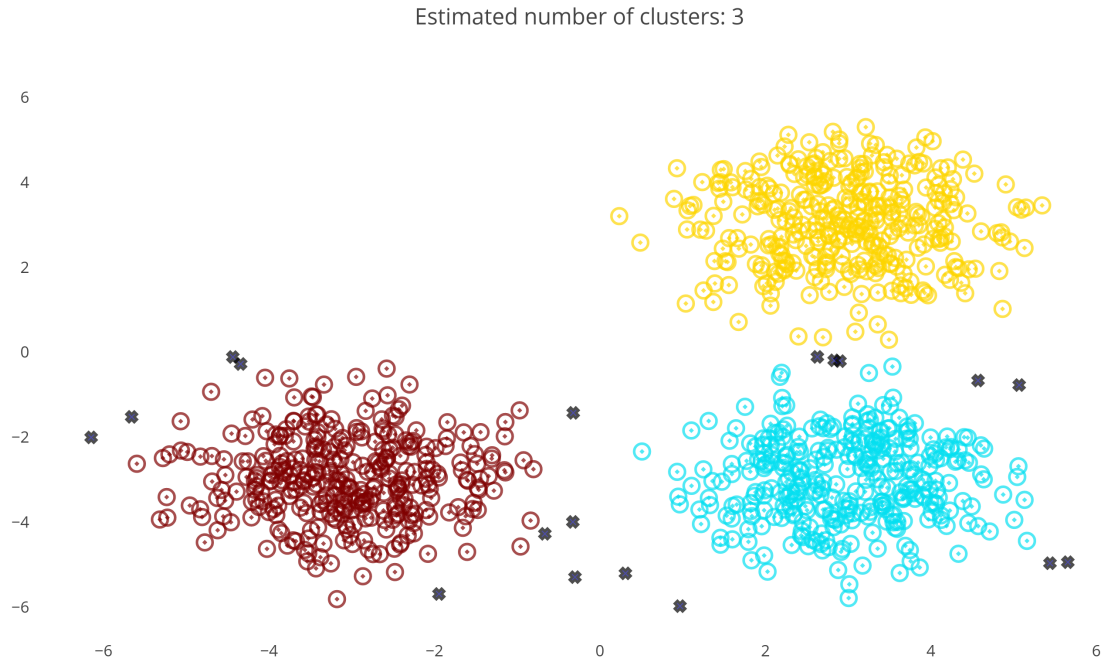
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
X1, y1=datasets.make_circles(n_samples=5000, factor=.6,
                                      noise=.05)
X2, y2 = datasets.make_blobs(n_samples=1000, n_features=2, centers=
[[1.2,1.2]], cluster_std=[[.1]],
               random_state=9)
X = np.concatenate((X1, X2))
plt.scatter(X[:, 0], X[:, 1], marker='o')
plt.show()
print(X1)
print(y1)
from sklearn.cluster import DBSCAN
y_pred = DBSCAN(eps = 0.1, min_samples = 10).fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=y_pred)
plt.show()
```

有待推广：

> 但是这个是自己生成的数据，如果一般的数据不止两维且需**自己读入数据**怎么办？试将 data 中的几个数据集用此方法求解



下述可视化为 plotly 实现，动态可修改，效果较好。但是目前尚未理解，有待进一步研究推广。代码参见 **code/dbscan_plotly.py**



> 这个代码我有点看不懂。。。

SPSS 实现聚类分析

模型的建立与结果分析——论文

可以参考以下模版：

Q 型聚类分析

1. 变量（指标）筛选

   待补充，通常通过相关性分析或者说 R 型聚类分析。

2. 数据标准化：$\bar{x}_i = \frac{x_i - \mu_i}{\sigma_i}$

3. 寻找异常点

   > 尤其当使用 kmeans 时，其他方法可以跳过

4. 计算距离矩阵：

   选取一个合适的样本点之间的距离函数，通常时欧几里得距离

   > 若要求马氏距离还需要估计协方差

5. 选择聚类算法，画出其算法流程图

Algorithm 4 An example for format For & While Loop in Algorithm

```
for each i ∈ [1, 9] do
    initialize a tree T_i with only a leaf (the root);
    T = T ∪ T_i;
end for
for all c such that c ∈ RecentMBatch(E_{n−1}) do
    T = T ∪ PosSample(c);
end for
for i = 1; i < n; i++ do
    // Your source here;
end for
for i = 1 to n do
    // Your source here;
end for
// Reusing recent base classifiers.
while (|E_n| ≤ L_1)and(D ≠ φ) do
    Selecting the most recent classifier c_i from D;
    D = D − c_i;
    E_n = E_n + c_i;
end while
```

6. 根据指标确定聚类个数

> 若采用的划分聚类则最先通过指标估计聚类个数

7. 获得最终聚类方案

8. 结果可视化：

绘制聚类效果与评估指标图，并结合实际情况进行分析，并检验聚类效果