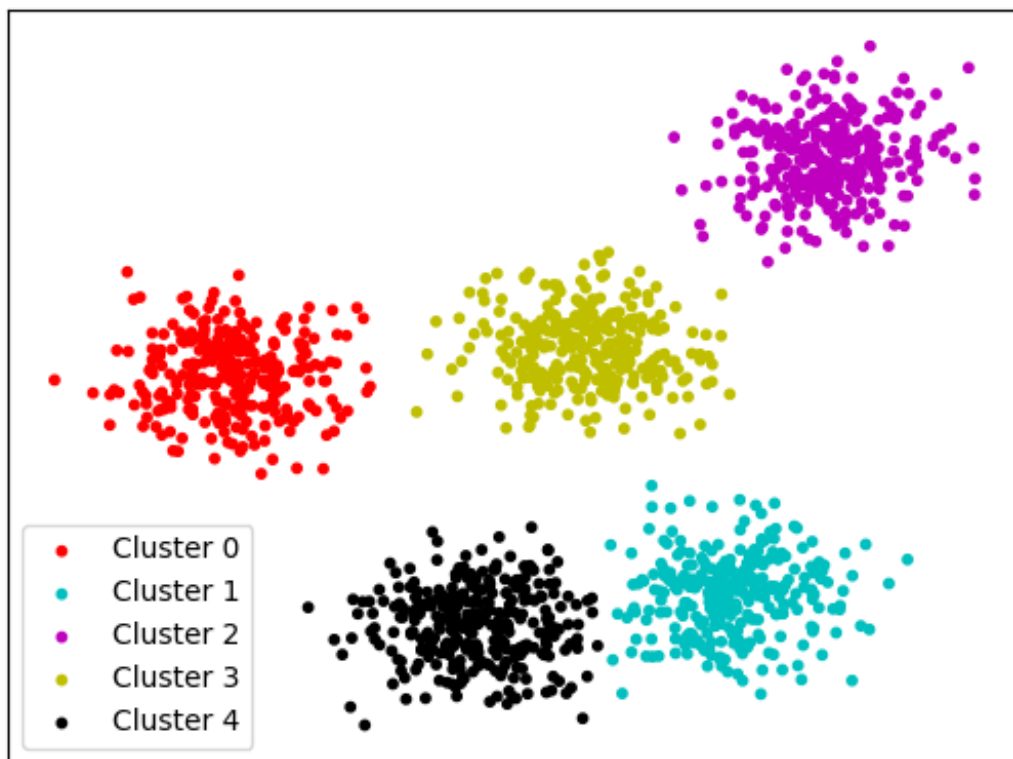


一篇文章透彻解读聚类分析（附数据和R代码）

知 zhuanlan.zhihu.com/p/37856153

不写代码的调参侠数据科学研究僧/调包侠/调参侠



最初是打算一个月写一篇文章的，正好当作自己复习了，但是自己拖延症真的是严重，从3月开始忙考试到忙完论文现在又实习，总觉得有其他事把这篇文章拖着，现在觉得真的不能再拖了。目前在一家公司的实验室做自然语言处理实习生，开发聊天机器人，负责意图识别和实体识别，正好这两天把手头上的工作做完了现在有空总结一下聚类了。

这篇文章主要是总结聚类分析的思想并提供数据和R的代码给大家实践。

我们很多时候逛电商网站都会收到一些推销活动的通知，但是我们之前也没关注过那个商品，这些电商网站是为什么决定给我们推销这个商品的呢？这是因为电商网站，可以根据用户的年龄、性别、地址以及历史数据等等信息，将其分为，比如“年轻白领”、“一家三口”、“家有一老”、“初得子女”等等类型，然后你属于其中的某一类，电商网站根据这类用户的特征向其发起不同的优惠活动。那在利用用户的这些数据将用户分为不同的类别时，就会用到聚类分析。

聚类分析的定义：

聚类分析是根据在数据中发现的描述对象及其关系的信息，将数据对象分组。目的是，组内的对象相互之间是相似的（相关的），而不同组中的对象是不同的（不相关的）。组内相似性越大，组间差距越大，说明聚类效果越好。

聚类效果的好坏依赖于两个因素：**1.衡量距离的方法（distance measurement）** **2.聚类算法（algorithm）**，这篇文章也将从这两个方面来展开介绍聚类分析

1.距离的计算（distance measurement）

计算数据之间距离的方法一般是根据数据的类型来选择的，数据的类型大概有数值变量，二元变量，类别变量，有序变量。

1.1数值变量（numerical）

- Minkowski 距离：X和Y是两个向量，，，那么

$$X = (x_1, x_2, \dots, x_p)$$

$$Y = (y_1, y_2, \dots, y_p)$$

$$d(X, Y) = \sqrt[q]{|x_1 - y_1|^q + |x_2 - y_2|^q + \dots + |x_p - y_p|^q}$$

,q 是正整数

- Euclidean 距离：是Minkowski，q=2时的特例

$$d(X, Y) = \sqrt{|x_1 - y_1|^2 + |x_2 - y_2|^2 + \dots + |x_p - y_p|^2}$$

- Manhattan 距离：是Minkowski, q=1时的特例

$$d(X, Y) = |x_1 - y_1| + |x_2 - y_2| + \dots + |x_p - y_p|$$

但是上面计算的距离有个缺点就是每个变量x1,x2...的重要性都是相同的，但是实际上有的变量更重要，有的没那么重要，这时候在计算距离就引进了权重（就好比衡量一个人和另外一个人学习成绩差异的时候，语数外的成绩明显要比美术音乐的成绩更重要），所以

Mahalanobis 距离：权重向量，那么

$$W = (w_1, w_2, \dots, w_p)$$

$$d(X, Y) = \sqrt[q]{w_1 * |x_1 - y_1|^q + w_2 * |x_2 - y_2|^q + \dots + w_p * |x_p - y_p|^q}$$

以上就是针对数值型数据计算距离的几种方法，但是我们知道数据的不同变量很多时候标度是不一样的，比如x1是（2000，3000）之间的变量，x2是（10-20）之间的变量，所以我们要对数据进行标准化，一般情况下我们都是对数据进行Z-score标准化：

2.2 二元变量

二元变量又分为对称二元变量和不对称二元变量。对称二元变量是指两个状态有相同的权重，比如性别，男性和女性就是对称二元变量。不对称二元变量是指两个状态的输出不是同样重要的，比如艾滋病阴性和阳性，阳性出现的几率更小。

$$Z_f = \frac{X_f - \text{mean}_f}{S_f}$$

那二元变量的距离如何计算呢？-----用列联表(contingency tabel)，下面用一个例子解释更直观：

我们有三个同学，他们有不同的特征，我们想衡量他们哪一对特征是更接近的

Name	Gender	Fever	Cough	Test-1	Test-2	Test-3	Test-4
Jack	M	Yes	No	Pos	Neg	Neg	Neg
Mary	F	Yes	No	Pos	Neg	Pos	Neg
Jim	M	Yes	Yes	Neg	Neg	Neg	Pos

我们首先将变量用0，1表示，这里我们只用到不对称二元变量来计算距离：

Name	Fever	Cough	Test-1	Test-2	Test-3	Test-4
Jack	1	0	1	0	0	0
Mary	1	0	1	0	1	0
Jim	1	1	0	0	0	1

列联表是什么呢？我们建立了X,Y之间的列联表，a是X和Y同时都是1的次数，b是指Y是0，X是1的次数，c,d类似

		Y		
		1	0	sum
X	1	a	b	a+b
	0	c	d	c+d
	sum	a+c	b+d	p

a: Number of variables with value '1' in both instances among the p binary variables

知乎 @郭磊 Wayne

X和Y之间的距离公式为，

例如：

$$d(X, Y) = \frac{b + c}{a + b + c}$$

,其中a=2(Fever,Test-1),b=0,c=1(Test-3)

,其中a=1(Fever),b=1(Test-1),c=2(Cough,Test-4)

所以Jack和Mary是最相近的两个。

$$d(Jack, Mary) = \frac{0 + 1}{2 + 0 + 1} = 0.33$$

2.3分类变量 (categorical)

分类变量是二元变量的一般情况，例如。计算分类变量的距离有两种方法


$$d(Jack, Jim) = \frac{1 + 2}{1 + 1 + 2} = 0.75$$

- 第一种是简单的匹配：,其中p是类别的个数，m是配对的个数
- 第二种是将分类变量二值化：

$$d(Mary, Jim) = \frac{2 + 2}{1 + 2 + 2} = 0.8$$

$$Color \in \{blue, green, red, \dots\}$$

$$d(X, Y) = \frac{p - m}{p}$$

ID	Color	...	Binarization				ID	Blue	Green	Red	...
1	Blue	...					1	1	0	0	...
2	Green	...					2	0	1	0	...
3	Red	...					3	0	0	1	...
...


转化成二元变量之后再列联表分析。

2.4有序变量(Ordinal)

其实就是类别变量进行了排序，例如,这里Low<Medium<High。计算这类型数据的方法是：

$$Level \in \{Low, Medium, High\}$$

1. 用每个值对应的排名 $r \in [1...N]$ 来代替这个值。
2. 计算z-scores来标准化排名，让r在[0,1]之间
3. 计算Minkowski距离（因为这时候的z-scores是连续性变量了）

ID	Level	...	Ranks			Z-scores		
1	Medium	...				1	0.5	...
2	Low	...				2	0.0	...
3	High	...				3	1.0	...
4	Medium	...				4	0.5	...
...

根据数据类型选择了相应的距离计算方法之后，我们就要选择聚类的方法了。

2. 聚类算法

2.1 K-均值聚类(k-means)

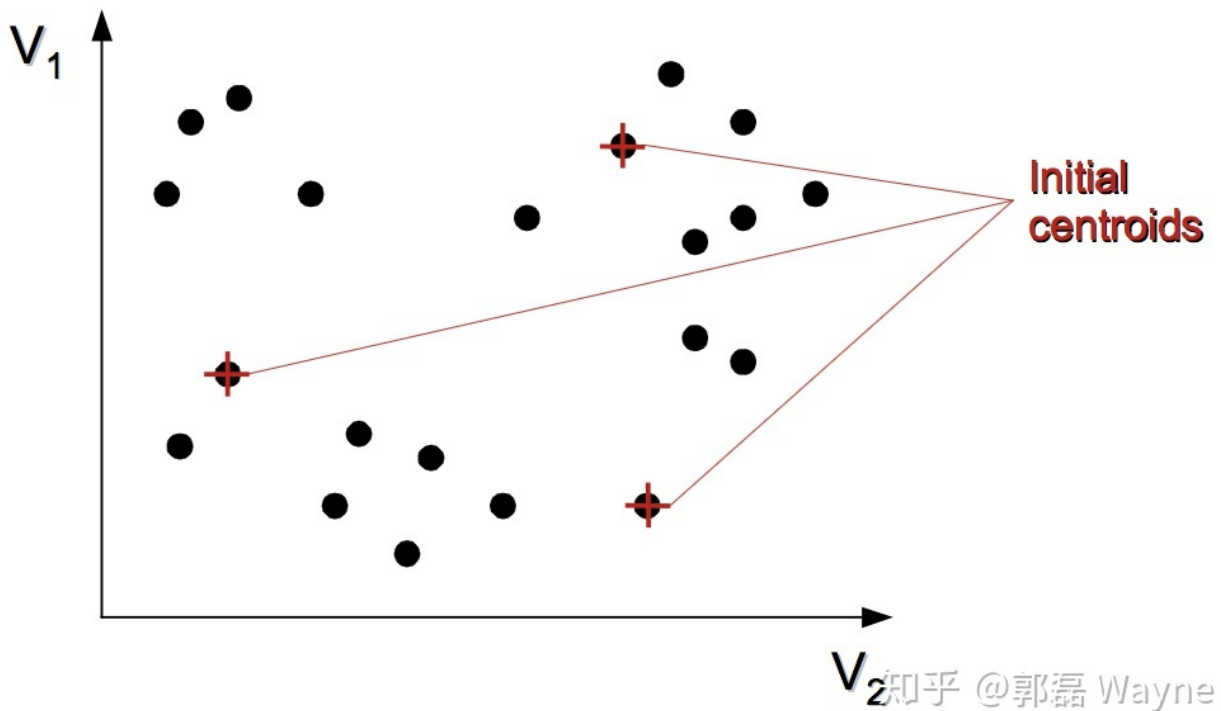
2.1.1 算法

1. 选择 K 个初始质心，初始质心随机选择即可，每一个质心为一个类
2. 把每个观测指派到离它最近的质心，与质心形成新的类
3. 重新计算每个类的质心，所谓质心就是一个类中的所有观测的平均向量（这里称为向量，是因为每一个观测都包含很多变量，所以我们把一个观测视为一个多维向量，维数由变量数决定）。
4. 重复2. 和 3.
5. 直到质心不在发生变化时或者到达最大迭代次数时

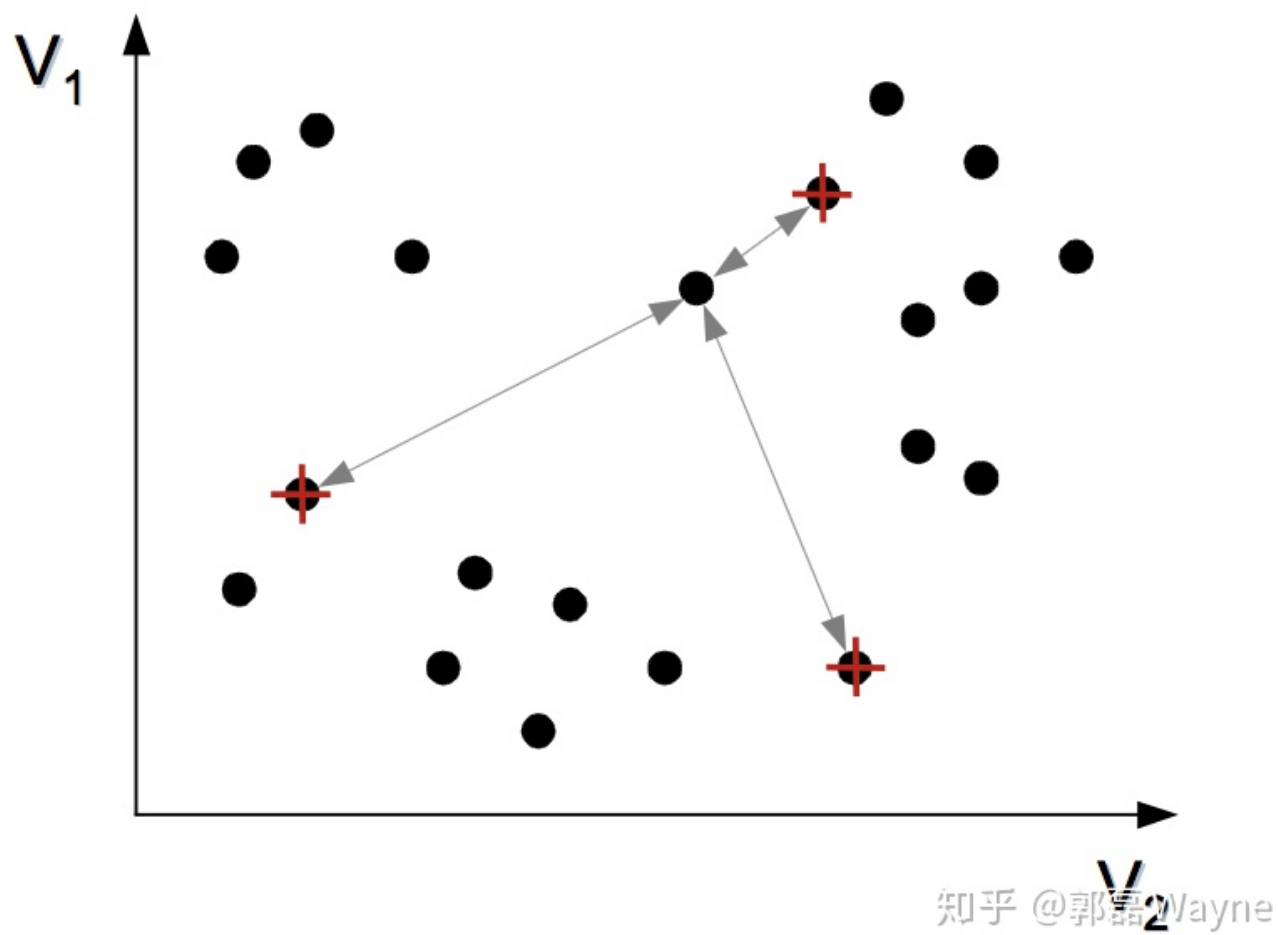
2.1.2 例子

有一个二维空间的一些点，我们要将它们分成3个类，即 $K=3$ 。

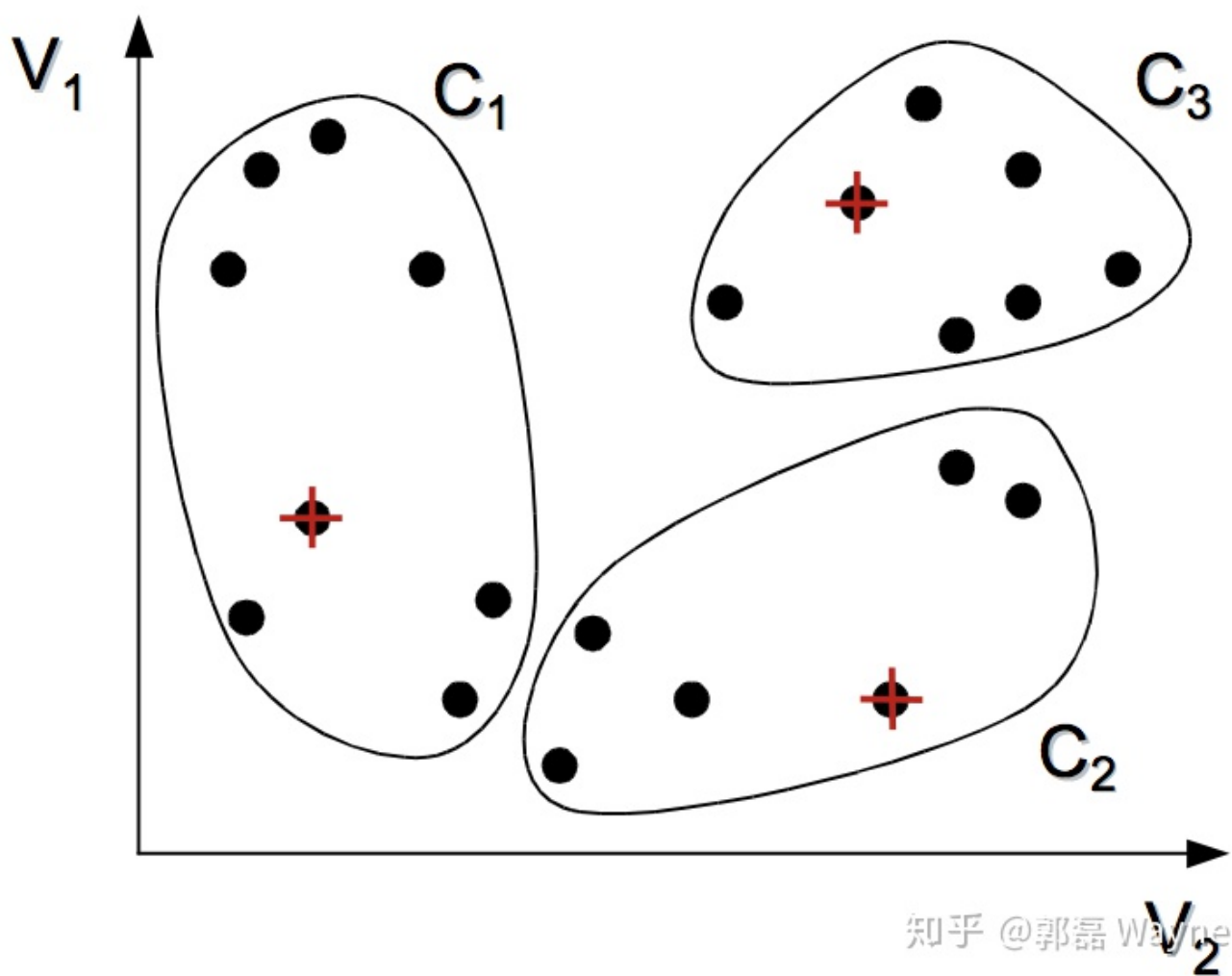
我们首先随机选择3个初始质心，每一个质心为一类：



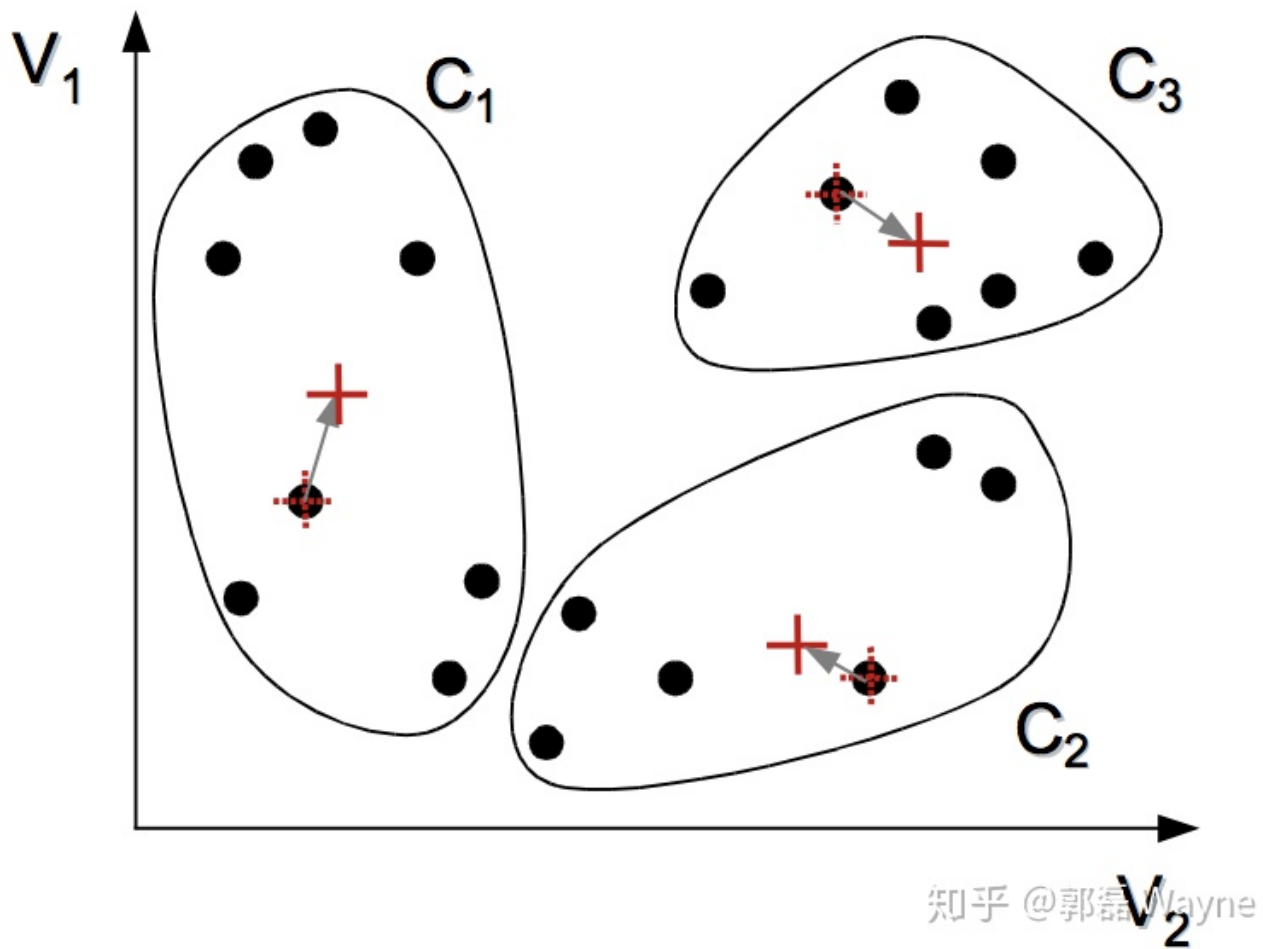
然后我们计算每一个不是质心的点到这三个质心的距离：



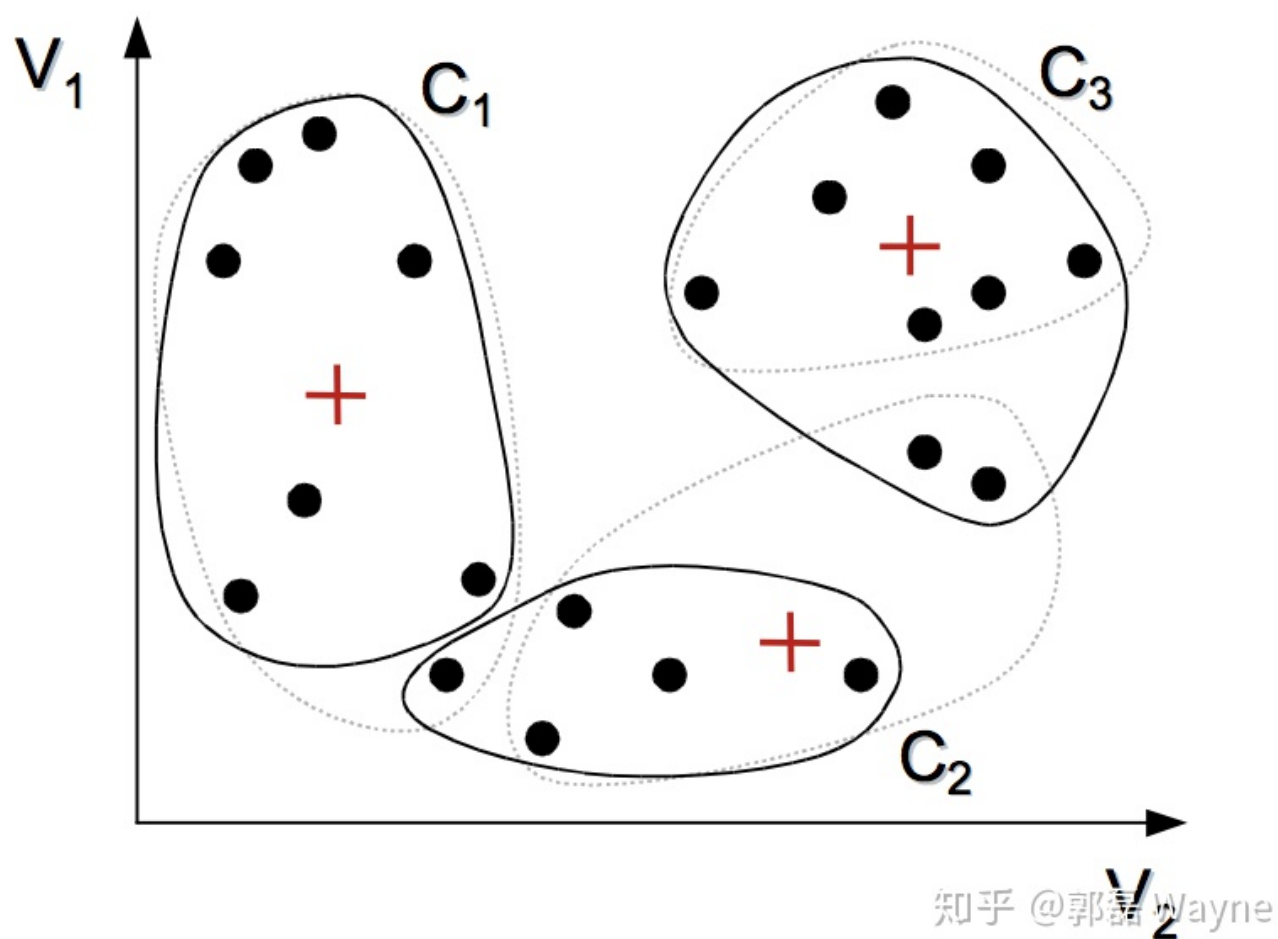
将这些点归类于距离最近的那个质心的一类：



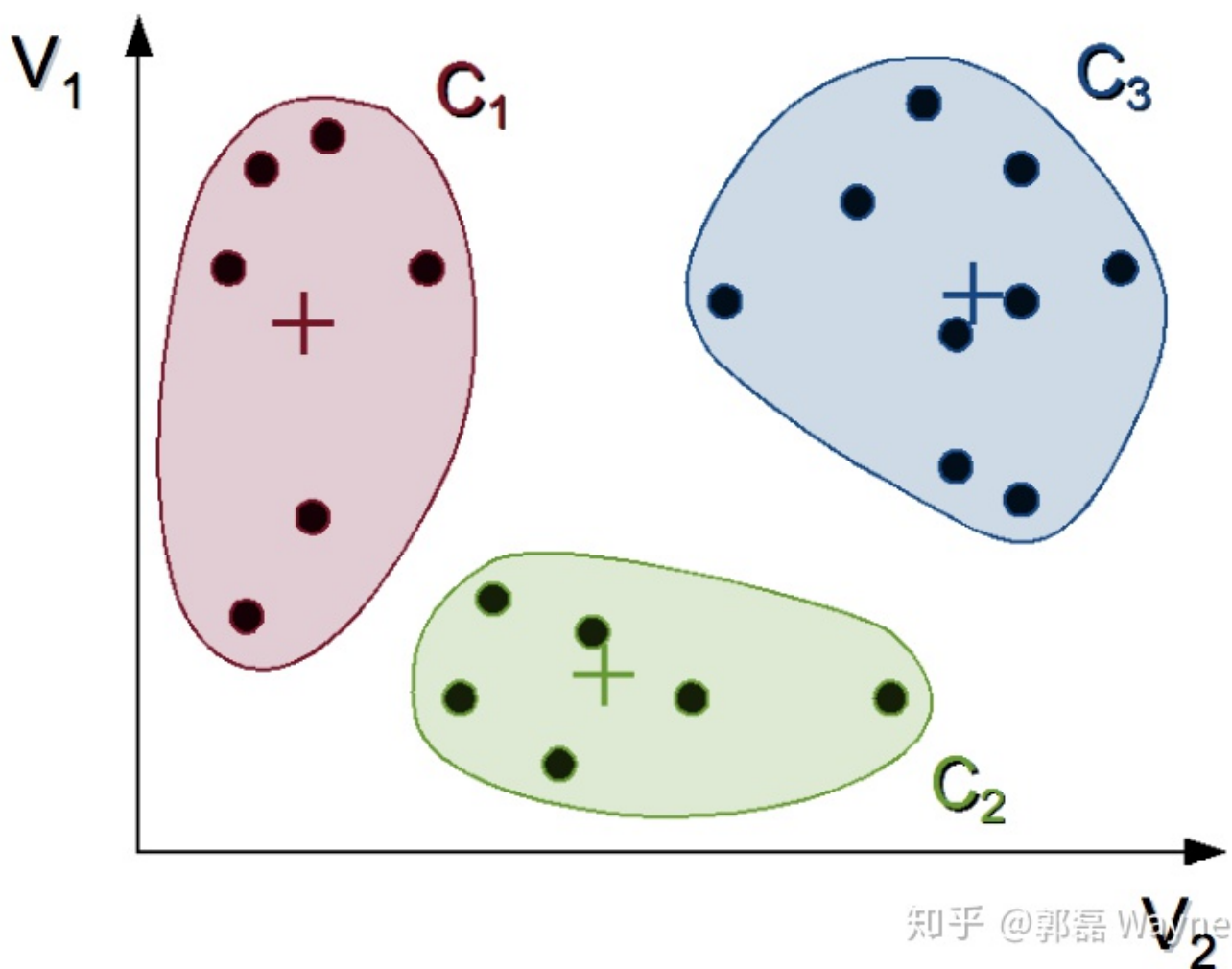
重新计算这三个分类的质心：



不断重复上述两步，更新三个类：



当稳定以后，迭代停止，这时候的三个类就是我们得到的最后的三个：



2.2 层次聚类(Hierarchical)

其核心思想是，把每一个单个的观测都视为一个类，而后计算各类之间的距离，选取最相近的两个类，将它们合并为一个类。新的这些类再继续计算距离，合并到最近的两个类。如此往复，最后就只有一个类。然后用树状图记录这个过程，这个树状图就包含了我们所需要的信息。

2.2.1 算法

1. 计算类与类之间的距离，用邻近度矩阵记录
2. 将最近的两个类合并为一个新的类
3. 根据新的类，更新邻近度矩阵
4. 重复2. 3.
5. 到只剩下一个类的时候，停止

2.2.2 例子

有一组数据 $D=\{a,b,c,d,e\}$ 给了它们之间的距离矩阵。

首先，每一个例子都是一个类：

- Initialization: Each instance constitutes a cluster

Distance matrix

	a	b	c	d	e
a	0.00				
b	0.18	0.00			
c	0.39	0.32	0.00		
d	0.43	0.34	0.25	0.00	
e	0.39	0.41	0.27	0.21	0.00

a

b

c

d

e

知乎 @郭磊 Wayne

将最近的两个类合并为一个新的类，并重新计算类之间的距离然后更新距离矩阵：

$d(a,b)=0.18$ 距离最近，合并为一类ab,然后计算ab,c,d,e之间的距离

$$d(ab, c) = \text{avg}(0.39, 0.32) = 0.355;$$

$$d(ab, d) = \text{avg}(0.43, 0.34) = 0.385;$$

$$d(ab, e) = \text{avg}(0.39, 0.41) = 0.4$$

- Iteration: Merge the two nearest clusters
- The distances $d(ab,c)$, $d(ab,d)$, $d(ab,e)$, $d(c,d)$, $d(c,e)$ and $d(d,e)$ are compared in the distance matrix

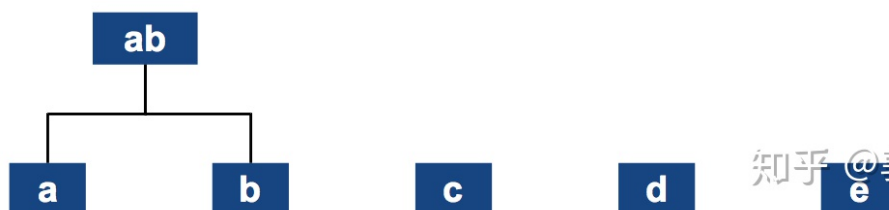
Distance matrix

	a	b	c	d	e
a	0.00				
b	0.18	0.00			
c	0.39	0.32	0.00		
d	0.43	0.34	0.25	0.00	
e	0.39	0.41	0.27	0.21	0.00

$$d(ab,c) = \text{avg}(0.39, 0.32) = 0.355$$

$$d(ab,d) = \text{avg}(0.43, 0.34) = 0.385$$

$$d(ab,e) = \text{avg}(0.39, 0.41) = 0.40$$



知乎 @郭磊 Wayne

选择距离最近的两个类合并为新的类：

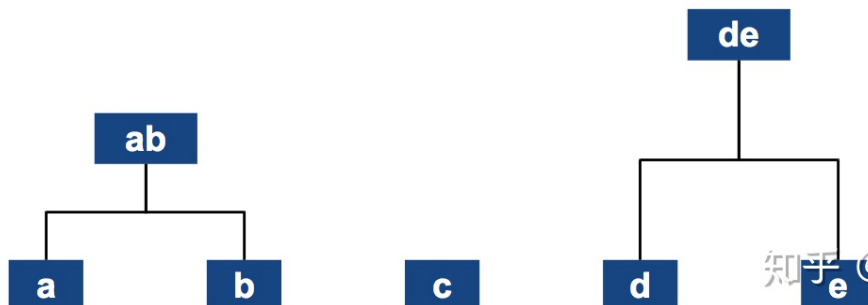
距离最近，因此d,e合并为一类

$$d(d,e) = 0.21$$

- Iteration: Merge the two nearest clusters
- Clusters $\{d\}$ and $\{e\}$ are merged since $d(d,e)$ is the lowest distance

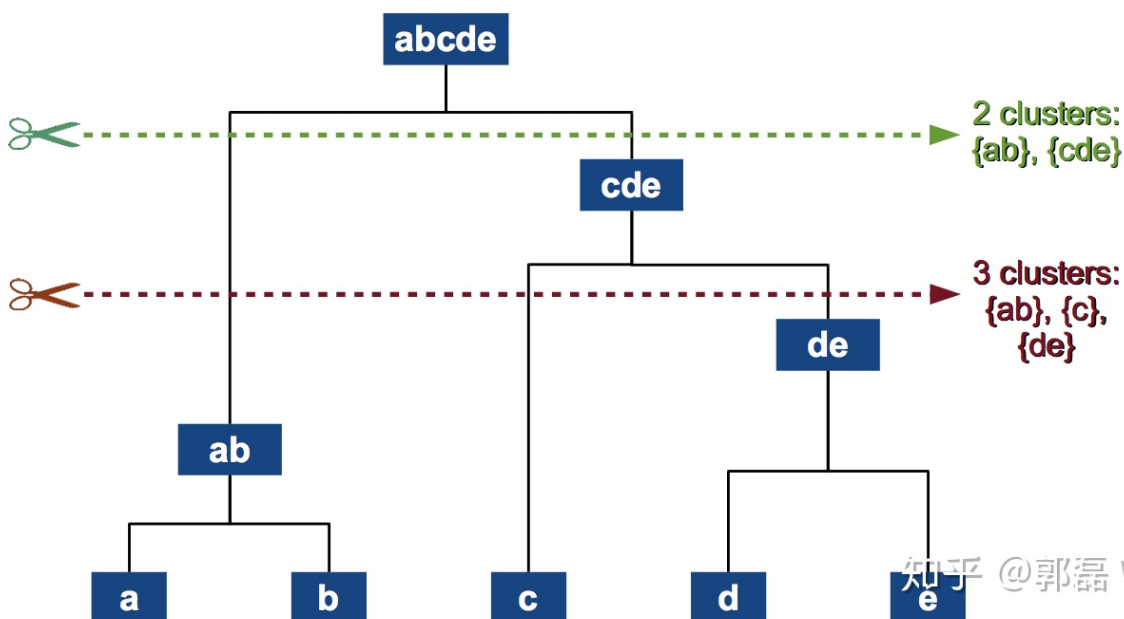
Distance matrix

	a	b	c	d	e
a	0.00				
b	0.18	0.00			
c	0.39	0.32	0.00		
d	0.43	0.34	0.25	0.00	
e	0.39	0.41	0.27	0.21	0.00



知乎 @郭磊 Wayne

不断重复上述两个步骤，最终只剩下一个类的时候，停止：



类别的数量取决于你剪的位置

2.3 根据密度的聚类

其核心思想是在数据空间中找到分散开的密集区域，简单来说就是画圈，其中要定义两个参数，一个是圈的最大半径，一个是一个圈里面最少应该容纳多少个点。

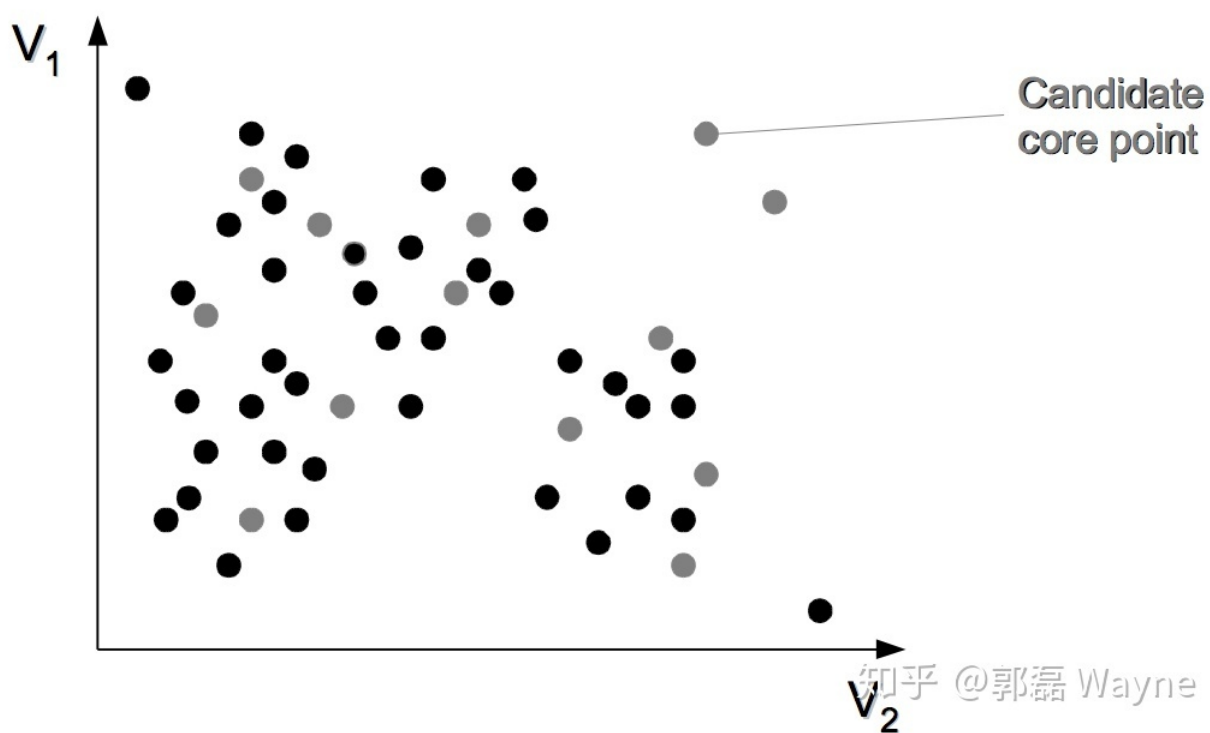
2.3.1 算法

1. 从数据集中随机选择核心点
2. 以一个核心点为圆心，做半径为 V 的圆，选择圆内圈入点的个数满足密度阈值的核心点，因此称这些点为核心对象，且将圈内的点形成一个簇，其中核心点直接密度可达周围的其他实心原点；
3. 合并这些相互重合的簇

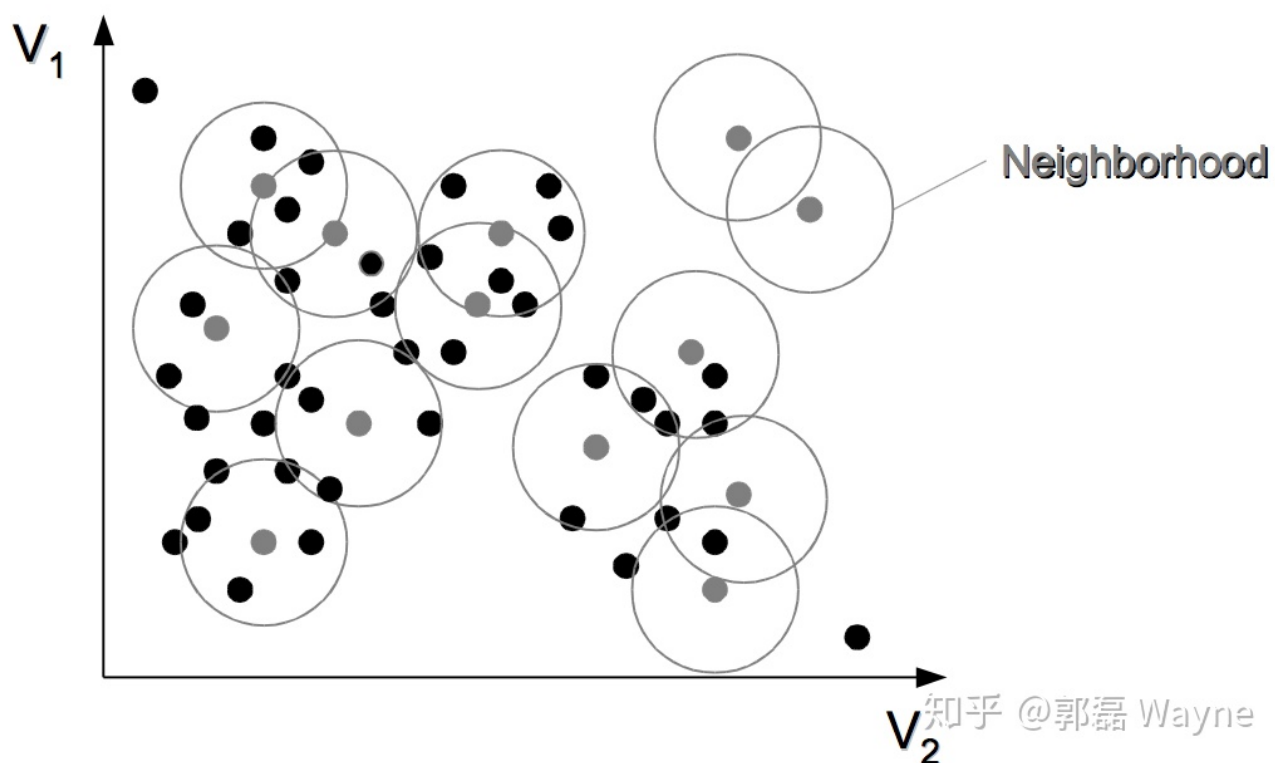
2.3.2 例子

我们设定参数密度阈值 $N=3$ ，也就是每个圈里必须满足3个点，才能称为一个簇，首先我们随机选取一些候选的核心点：

这些灰色点为核心点的候选：

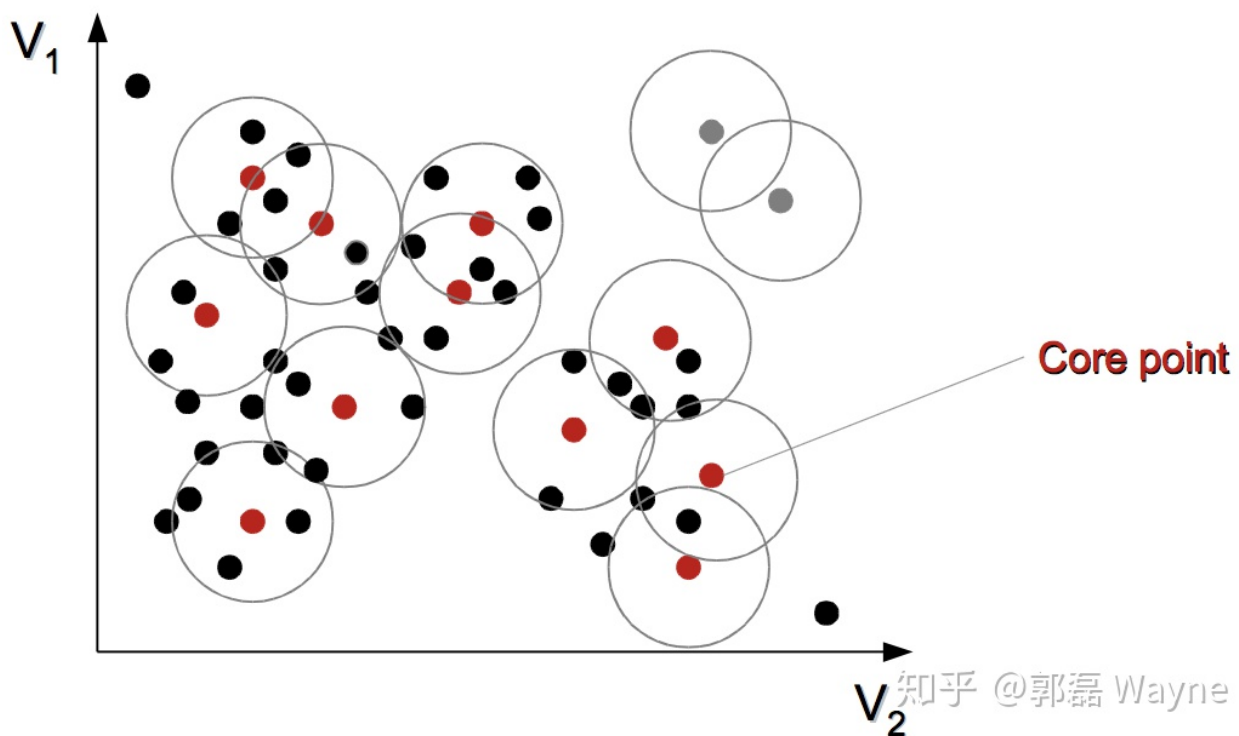


以这些候选的核心点为圆心按照设定的半径画圆圈：

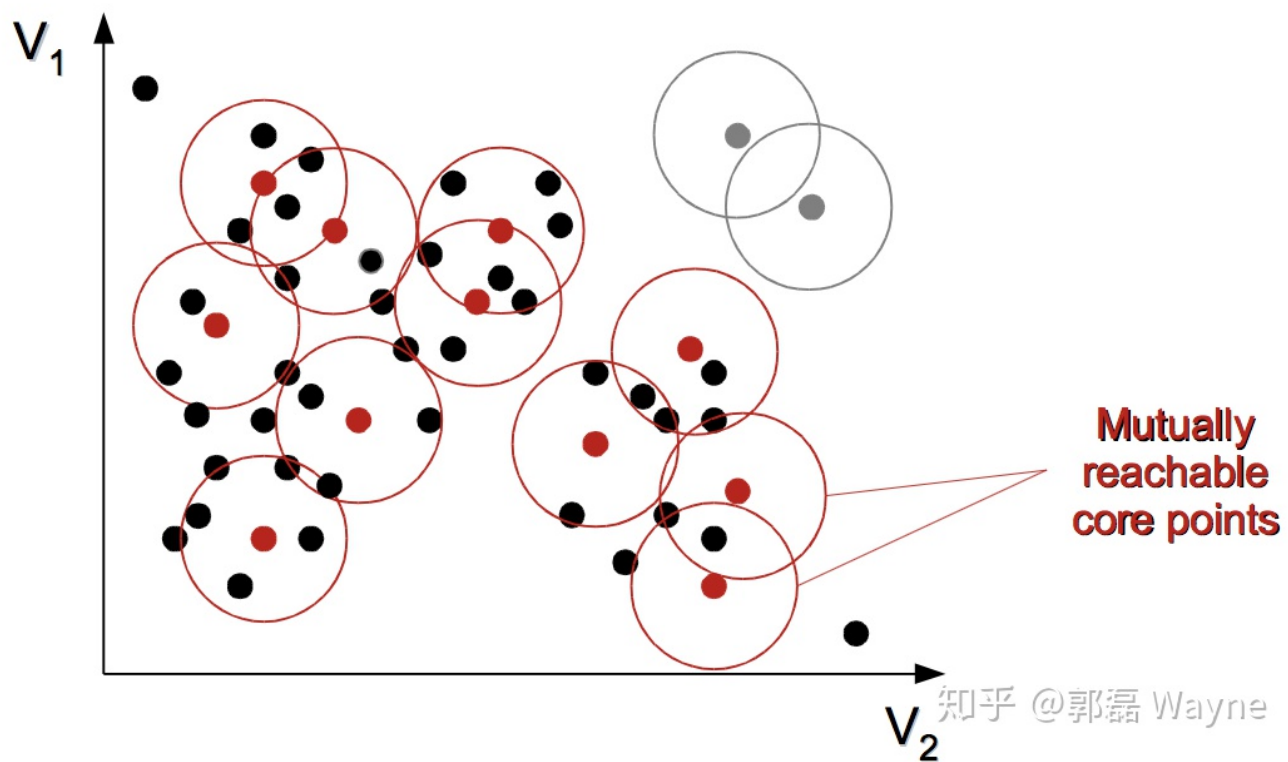


如果圈内满足三个点，那就是一个簇，簇内点候选的核心点就是核心点：

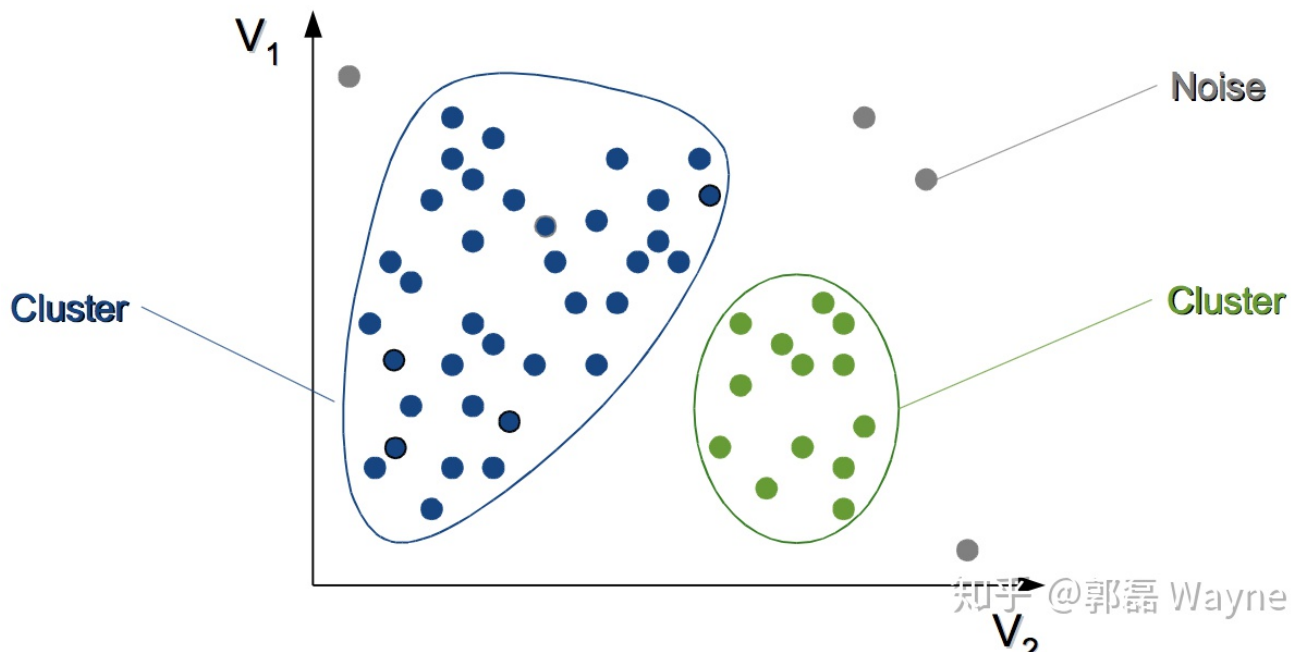
这些红色的点就是核心点，而灰色的点因为其圈内点数不满足阈值，所以不是核心点



合并重合的簇：



得到两个cluster:



2.4根据网格的聚类

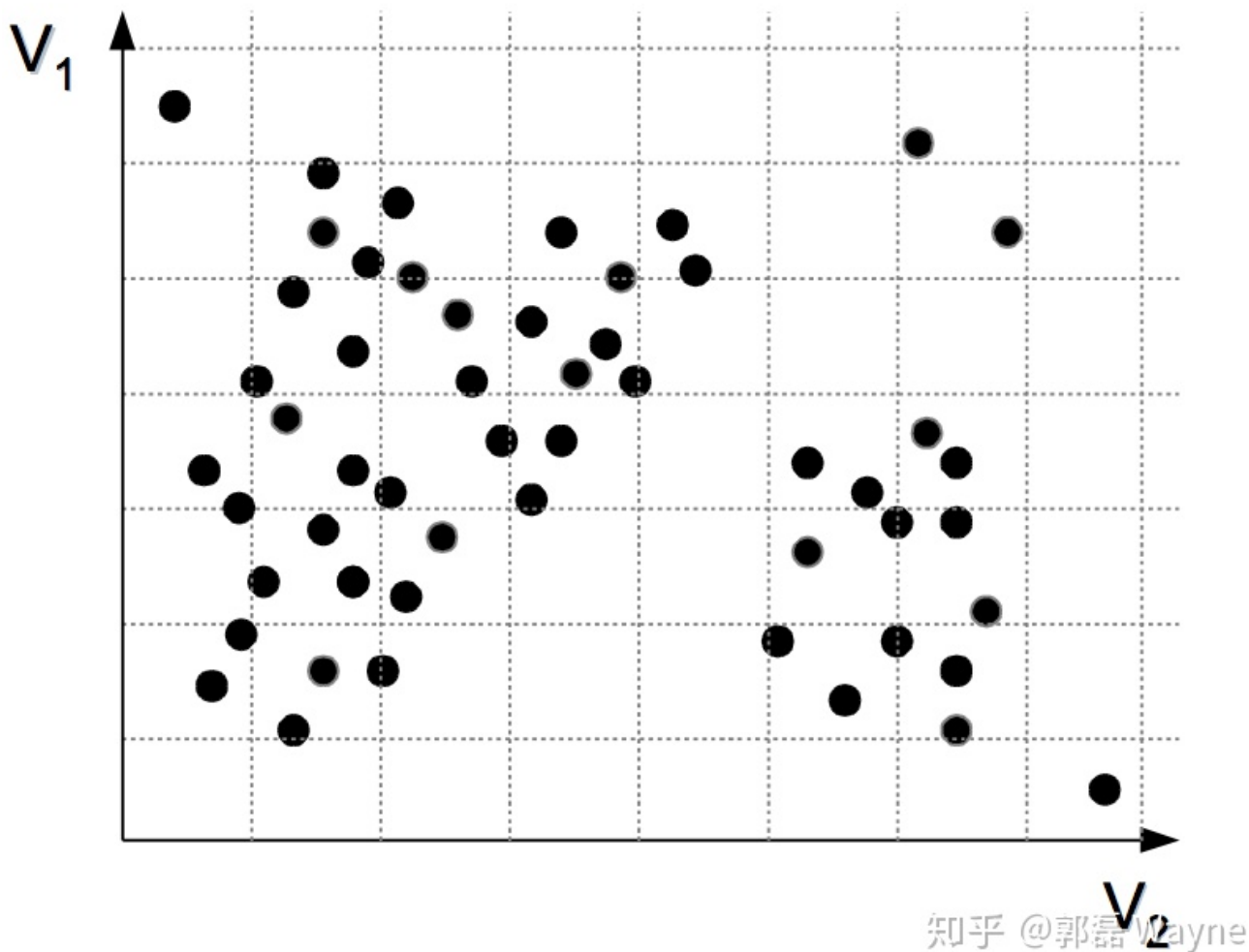
其原理是将数据空间划分为网格单元，将数据对象映射到网格单元中，并计算每个单元的密度。根据预设阈值来判断每个网格单元是不是高密度单元，由邻近的稠密单元组成“类”。

2.4.1算法

- 1.将数据空间划分为网格单元
- 2.依照设置的阈值，判定网格单元是否稠密
- 3.合并相邻稠密的网格单元为一类

2.4.2例子

选择一定宽度的格子来分割数据空间：



设置阈值为2，将相邻稠密的格子合并形成一个“类”：

以上就是关于聚类详细介绍，至于每个聚类算法在什么时候应用，这个是需要了解每个方法的优缺点，然后针对具体问题选择最适合的算法。关于每个聚类算法的优缺点，这个回答介绍的很详细。

用于数据挖掘的聚类算法有哪些，各有何优势？ - 郭小贤的回答 - 知乎

用于数据挖掘的聚类算法有哪些，各有何优势？ www.zhihu.com

数据：

链接：pan.baidu.com/s/1DGyhAB

密码:27sj

```
# Required libraries
library(cluster)
library(ggplot2)
```

```

## DATA PREPARATION ##
# Load data
product <- read.csv("*/M2_IFI_Data_Product.csv", header = TRUE, sep = ",", dec = ".")
attach(product)
View(product)

# Delete variable ID
product <- product[,-1]

# Modify Children variable type
product$Children <- as.factor(product$Children)

# Compute distance matrix
dmatrix <- daisy(product)
summary(dmatrix)

## K-MEANS CLUSTERING ##

# K-means clustering for k in [4,8]
km4 <- kmeans(dmatrix, 4)
km5 <- kmeans(dmatrix, 5)
km6 <- kmeans(dmatrix, 6)
km7 <- kmeans(dmatrix, 7)
km8 <- kmeans(dmatrix, 8)

# Add column with cluster number in the product data frame
product <- data.frame(product, km4$cluster)
product <- data.frame(product, km5$cluster)
product <- data.frame(product, km6$cluster)
product <- data.frame(product, km7$cluster)
product <- data.frame(product, km8$cluster)
attach(product)

# Distribution of classes (Product=Oui/Non) by cluster
table(km4.cluster, Product)
table(km5.cluster, Product)
table(km6.cluster, Product)
table(km7.cluster, Product)
table(km8.cluster, Product)

# Histograms of cluster sizes with class in color
qplot(km4$cluster, data=product, fill=Product)
qplot(km5$cluster, data=product, fill=Product)
qplot(km6$cluster, data=product, fill=Product)
qplot(km7$cluster, data=product, fill=Product)
qplot(km8$cluster, data=product, fill=Product)

#??? Display distribution of classes for 4-means clusters vs. Children and Family_Quotient
qplot(Children, km4$cluster, data=product, color=Product) + geom_jitter(width = 0.3, height = 0.3)
qplot(Family_Quotient, km4$cluster, data=product, color=Product) + geom_jitter(width = 0,

```

```
height = 0.3)
```

```
## HIERARCHICAL CLUSTERING ##
```

```
# Run the hclust algorithm
```

```
hc <- hclust(dmatrix, method="ward.D2")
```

```
# Display the dendrogram
```

```
plot(hc)
```

```
# Generate clusters for different k values
```

```
ghc4 <- cutree(hc, k=4)
```

```
ghc5 <- cutree(hc, k=5)
```

```
ghc6 <- cutree(hc, k=6)
```

```
ghc7 <- cutree(hc, k=7)
```

```
ghc8 <- cutree(hc, k=8)
```

```
# Distribution of classes (Product=Oui/Non) by cluster
```

```
product <- data.frame(product, ghc4)
```

```
product <- data.frame(product, ghc5)
```

```
product <- data.frame(product, ghc6)
```

```
product <- data.frame(product, ghc7)
```

```
product <- data.frame(product, ghc8)
```

```
table(product$ghc4, Product)
```

```
table(product$ghc5, Product)
```

```
table(product$ghc6, Product)
```

```
table(product$ghc7, Product)
```

```
table(product$ghc8, Product)
```

```
# Display clusters for k in [4,8]
```

```
rect.hclust(hc, k=4, border="red")
```

```
rect.hclust(hc, k=5, border="blue")
```

```
rect.hclust(hc, k=6, border="green")
```

```
rect.hclust(hc, k=7, border="gold")
```

```
rect.hclust(hc, k=8, border="skyblue")
```

```
# Histograms of cluster sizes with class in color
```

```
qplot(product$ghc4, data=product, fill=Product)
```

```
qplot(product$ghc5, data=product, fill=Product)
```

```
qplot(product$ghc6, data=product, fill=Product)
```

```
qplot(product$ghc7, data=product, fill=Product)
```

```
qplot(product$ghc8, data=product, fill=Product)
```

```
## FOR LOOPS #
```

```
# Computing distance matrix
```

```
dmatrix <- daisy(product)
```

```
# K-means executions for a range of k values
```

```
for (i in 4:8){
```

```
  km <- kmeans(dmatrix, i)
```

```
product <- data.frame(product, km$cluster)
print(table(km$cluster, Product))
print(qplot(km$cluster, data=product, fill=Product))
}
```