

数据挖掘 复习

简单 20 分

应用题 80 分 5-6 个

平时成绩 2 个实验各 15 分，报告 20 分

第一章

为什么要数据挖掘

数据量急剧上升

存储技术发展

什么是数据挖掘

从大量数据中挖掘那些令人感兴趣的、隐含的、先前未知的和可能有用的模式或知识。

与传统数据库查询本质区别

数据挖掘是在**没有明确假设的前提下**去挖掘信息、发现知识；

数据挖掘所得到的信息应具有**先前未知、有效和可实用**三个特征。

两大类任务：

➤ 分类预测型任务

利用一些已知变量来预测未知的或其他变量将来的值。典型的方法是回归分析

➤ 描述型任务

找出人们可以解释，并能够描述数据的模式。

数据

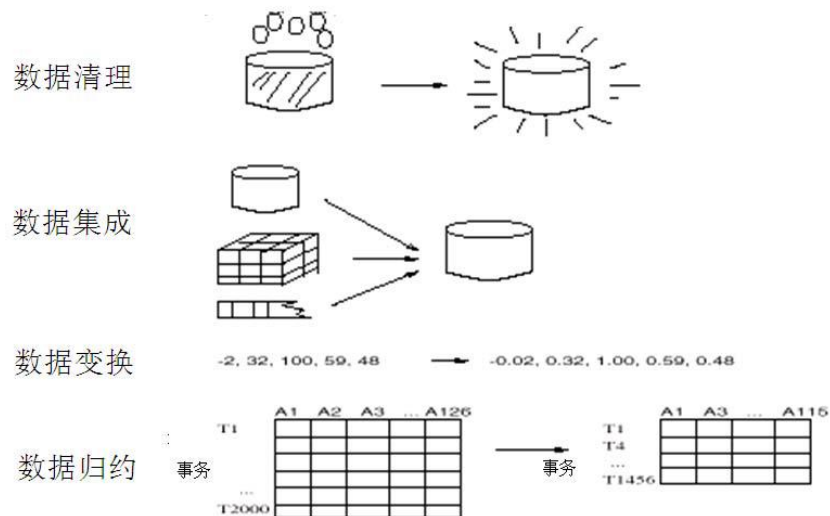
1. 关系数据库
2. 数据仓库
3. 事务数据库
4. 高级数据库系统和信息库
 - 空间数据库
 - 时间数据库和时间序列数据库
 - 流数据
 - 多媒体数据库
 - 文本数据库和万维网(www)

第二章

为什么进行数据预处理

不完整、有噪声、数据不一致

数据预处理 4 个步骤



数据清理

任务:

填写空缺的值

识别离群点和平滑噪声数据

纠正不一致的数据

解决数据集成造成的冗余

处理空缺值

人工填写空缺值：工作量大

在所有样本上，使用属性的平均值填充空缺值

使用给定元组属同一类的所有样本的平均值

噪声数据

计算机和人工检查结合

回归：通过让数据适应回归函数来平滑数据

聚类：监测并去除孤立点

数据集成

将多个数据源中的数据整合到一个一致的存储中的过程。

处理冗余数据：

对象识别

构建同义词表

数据变换

规范化：将数据按比例缩放，使之落入一个特定区间

最小-最大规范化

z-score 规范化

数据规约

维规约：删除贡献低的属性或维→减少数据量，常用 PCA 方法

数据压缩：有损压缩、无损压缩

数值规约：通过选择替代的或者较小的数据来减少数据量

有参方法：线性回归

无参方法：直方图、聚类、选样

第三章 关联挖掘

支持度：包含 X 的交易数/交易总数

最小支持度：用于发现关联规则的项集必须满足的最小阈值

关联规则： $R: X \rightarrow Y, X \cap Y = \emptyset$

最小可信度：包含 X 和 Y 的交易数/包含 X 的交易数

例子：

交易号 TID	顾客购买商品 Items
T1	bread cream milk tea
T2	bread cream milk
T3	cake milk
T4	milk tea
T5	bread cake milk
T6	bread tea
T7	beer milk tea
T8	bread tea
T9	bread cream milk tea
T10	bread milk tea

对于2-项集 $X=\{\text{bread}, \text{milk}\}$ ，它出现在T1, T2, T5, T9和T10中，

$$\text{support}(X)=5/10=0.5$$

关联规则支持度

假设 $\text{sup}_{\min}=0.5$ ，对于关联规则 $R_1: \{\text{bread}\} \Rightarrow \{\text{milk}\}$ ，则

$$\text{support}(R_1)=\text{support}(\{\text{bread}, \text{milk}\})=0.5。$$

对于关联规则 $R_2: \{\text{milk}\} \Rightarrow \{\text{bread}\}$ ，则

$$\text{support}(R_2)=\text{support}(\{\text{bread}, \text{milk}\})=0.5。$$

可信度

关联规则 $R_1: \{\text{bread}\} \Rightarrow \{\text{milk}\}$ 的可信度为

$$\text{confidence}(R_1) = \frac{\text{support}(\{\text{bread}, \text{milk}\})}{\text{support}(\{\text{bread}\})} = 0.5/0.7 = 5/7。$$

关联规则 $R_2: \{\text{milk}\} \Rightarrow \{\text{bread}\}$ 的可信度为

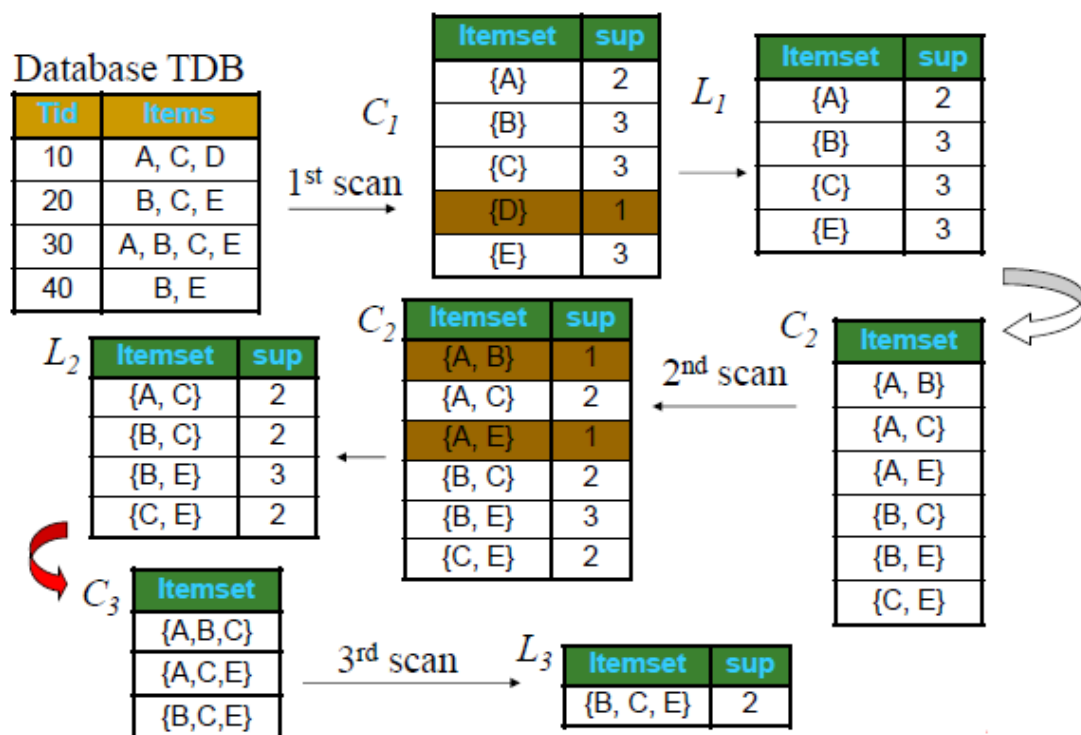
$$\text{confidence}(R_2) = \frac{\text{support}(\{\text{bread}, \text{milk}\})}{\text{support}(\{\text{milk}\})} = 0.5/0.8 = 5/8。$$

Apriori 算法—多次扫描

通过迭代来穷举出数据集中的所有频繁集

过程:

1. 产生 1-频繁集 L_1
2. 在 L_1 上通过连接和修剪产生 2-频繁集 L_2
3. 重复上述流程, 在 L_k 上产生 $(k+1)$ -频繁集 L_{k+1}
4. 直到无法产生新的频繁集, 结束



连接: 只相差一个项目的两个项集才能进行连接 (集合“并”操作)。

修剪: 去除子集不是频繁集的项集。

上面例子中的3-频繁集{B,C,E}来说, 它的非空子集包括{{B}, {C}, {E}, {B,C}, {B,E}, {C,E}}。

选取其中2个规则:

- 关联规则{B,C}→{E}的可信度 $\text{Conf}=2/2=100\%$ 。

- 关联规则{B,E}→{C}的可信度Conf=2/3=66.7%。
- 当最小可信度设置为80%时，关联规则{B,C}→{E}是强关联规则，而关联规则{B,E}→{C}不是。
- 当最小可信度设置为50%时，关联规则{B,C}→{E}和{B,E}→{C}都是强关联规则。

FP-growth—两次扫描

例子：

第一步：扫描一遍数据集 D，得到 1-频繁项集 F 和每个频繁项的支持度。并将 F 按支持度递降排序，存入列表 L 中。

■ 根据一个数据集D，建立一棵FP-tree。

- 第一步：扫描一遍数据集D，得到1-频繁项集F和每个频繁项的支持度。并将F按支持度递降排序，存入列表L中。

<i>TID</i>	<i>Items</i>		<i>Item</i>	<i>counts</i>
100	{f, a, c, d, g, i, m, p}	$sup_{min} = 3$ 次	f	4
200	{a, b, c, f, l, m, o}		c	4
300	{b, f, h, j, o}		a	3
400	{b, c, k, s, p}		b	3
500	{a, f, c, e, l, p, m, n}		m	3
			p	3

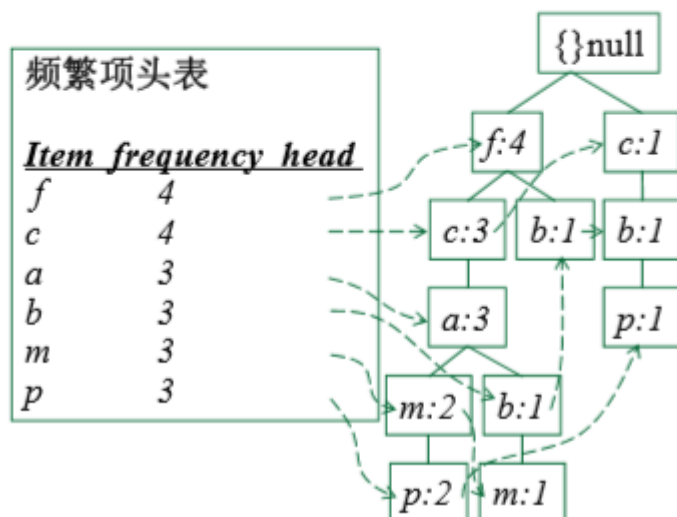
<i>TID</i>	<i>Items</i>	<i>(ordered) frequent items</i>
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

第二步：

- 创建树的根节点，用null标记；
- 将每个交易中的项按支持度递减排列，并对每个交易创建一个分枝； 例如：为第一个交易{f, c, a, m, p}构建一个分枝。
- 当为一个交易增加分枝时，沿共同前缀上的每个节点的计数加1，为跟随前缀后的项创建节点并连接。 例如：将第二个交易{f, c, a, b, m}加到树上时，需要为f, c, a各增计数1，然后为{b, m}创建分枝。
- 创建一个频繁项头表，以方便遍历，每个项通过一个指针指向它在树中出现的位置。

<i>TID</i>	<i>Items bought</i>	<i>(ordered) frequent items</i>
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

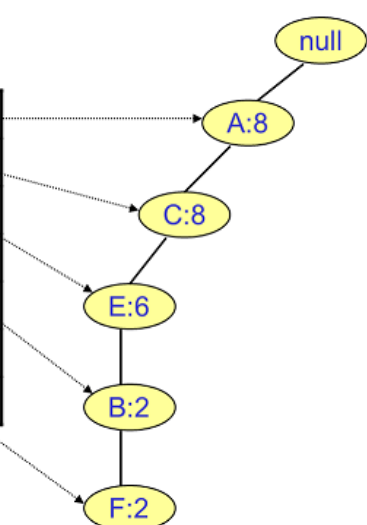
$sup_{min} = 3$



我们看看先从最底下的F节点开始，我们先来寻找F节点的条件模式基，由于F在FP树中只有一个节点，因此候选就只有下图左所示的一条路径，对应{A:8,C:8,E:6,B:2, F:2}。我们接着将所有的祖先节点计数设置为叶子节点的计数，即FP子树变成{A:2,C:2,E:2,B:2, F:2}。一般我们的条件模式基可以不写叶子节点，因此最终的F的条件模式基如下图所示。

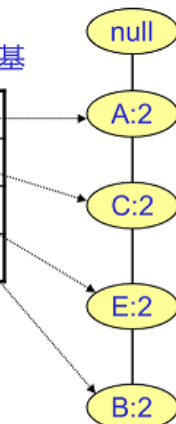
项头表

A:8	
C:8	
E:8	
G:5	
B:2	
D:2	
F:2	



F的条件模式基

A:2	
C:2	
E:2	
B:2	



第四章 分类与预测

决策树

期望 Info

熵

信息增益

信息增益比

ID3—只适用离散型

最高信息增益的属性作为数据划分

倾向于选取种类较多的属性进行划分

C4.5—适用离散型、连续型

最佳分割点：具有最大信息增益比

神经网络设计

神经元个数的确定

SVM

硬间隔

软间隔：松弛变量

核函数：

多项式核

高斯核

Sigmoid

多分类问题

1 vs (N-1): 训练 N 个分类器, 第 i 个分类器判断样本是否属于第 i 类

1 vs 1: 训练 $N*(N-1)/2$ 个分类器, 分类器(i,j)判断样本是否属于第 i 类 or 第 j 类

贝叶斯

一个贝叶斯网络由网络结构和条件概率表两部分组成。

➤ 网络结构是一个有向无环图, 由若干结点和有向弧组成。

结点的取值必须是完备互斥的

➤ 条件概率表: 是指网络中的每个结点都有一个条件概率表, 用于表示其父结点对该结点的影响。

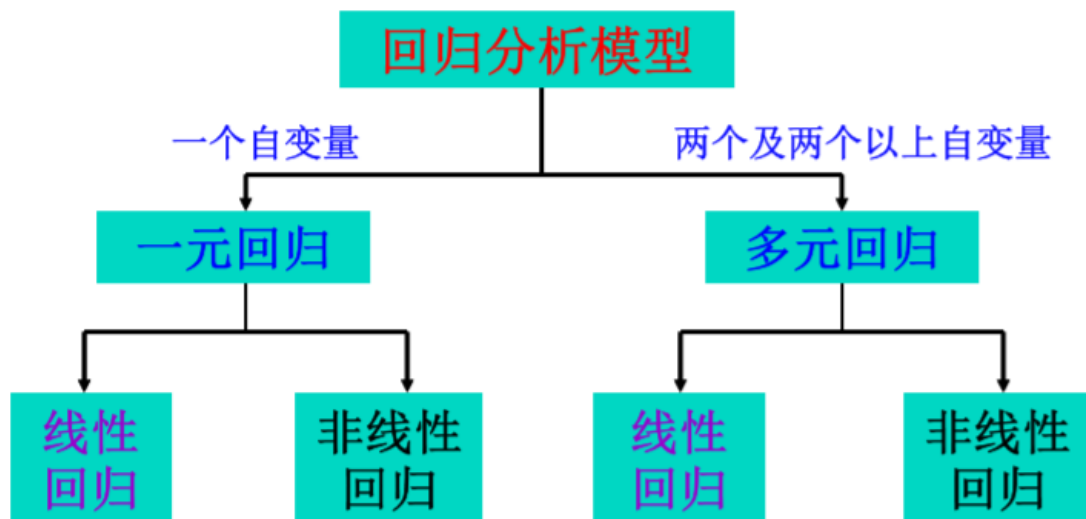
数据预测

构建一个模型, 利用模型来估计未知样本对应的连续值。

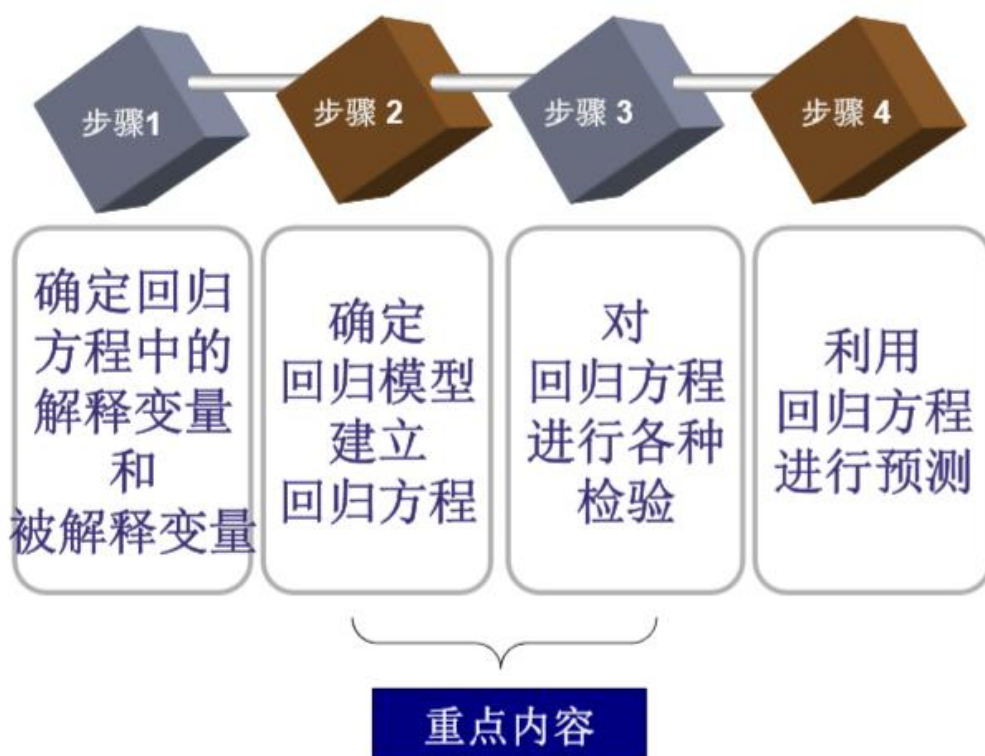
数据预测的主要方法:

神经网络、SVM 等均可以实现数据预测

回归分析 (Regression Analysis)



回归分析的一般步骤



非线性—变换

一元非线性回归

只涉及一个自变量的回归分析，但因变量与自变量之间的关系是非线性的。

建模方法：

首先，使用适合的一元非线性模型，构建因变量与自变量之间的关系；
其次，进行变量替换，将一元非线性回归模型转换为一元线性回归模型；
后，根据样本数据，并使用小二乘法求解（估计）模型的参数。

常用的一元非线性模型包括：

- 指数函数
- 幂函数
- 双曲函数
- 对数函数

第五章 聚类分析

划分法

k-means

实例

序号	属性 1	属性 2
1	1	1
2	2	1
3	1	2
4	2	2
5	4	3
6	5	3
7	4	4
8	5	4

根据所给的数据通过对其实施k-means (设 $n=8$, $k=2$)，其主要执行步骤：

第一次迭代：假定随机选择的两个对象，如序号1和序号3当作初始点，分别找到离两点最近的对象，并产生两个簇{1, 2}和{3, 4, 5, 6, 7, 8}。

对于产生的簇分别计算平均值，得到平均值点。

对于{1, 2}，平均值点为 (1.5, 1)；

对于{3, 4, 5, 6, 7, 8}，平均值点为 (3.5, 3)。

第二次迭代：通过平均值调整对象的所在的簇，重新聚类，即将所有点按离平均值点 (1.5, 1)、(3.5, 3) 最近的原则重新分配。得到两个新的簇：{1, 2, 3, 4}和{5, 6, 7, 8}。重新计算簇平均值点，得到新的平均值点为 (1.5, 1.5) 和 (4.5, 3.5)。

第三次迭代：将所有点按离平均值点 (1.5, 1.5) 和 (4.5, 3.5) 最近的原则重新分配，调整对象，簇仍然为{1, 2, 3, 4}和{5, 6, 7, 8}，发现没有出现重新分配，而且准则函数收敛，程序结束。

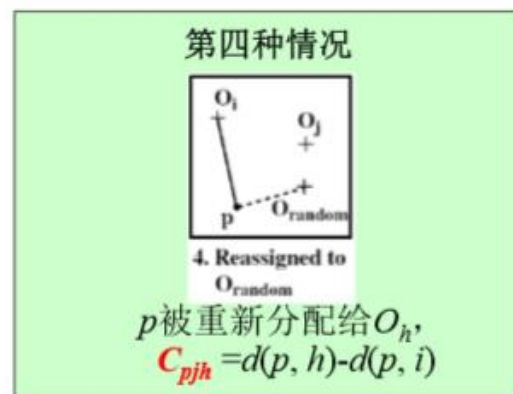
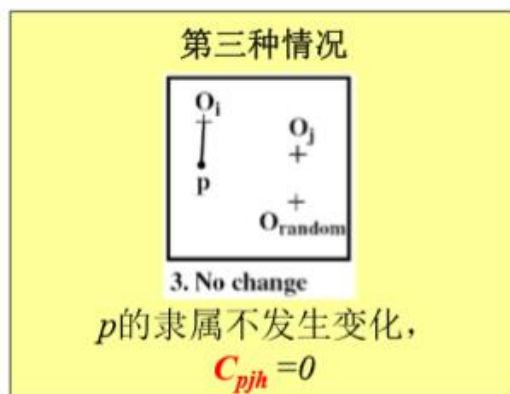
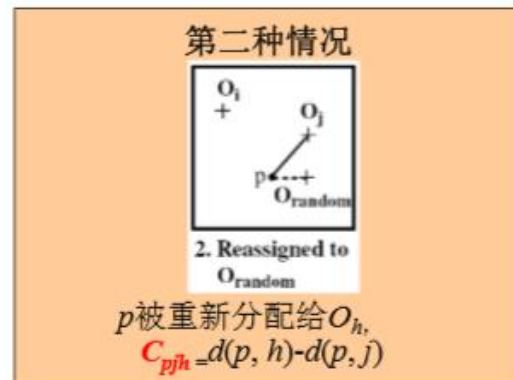
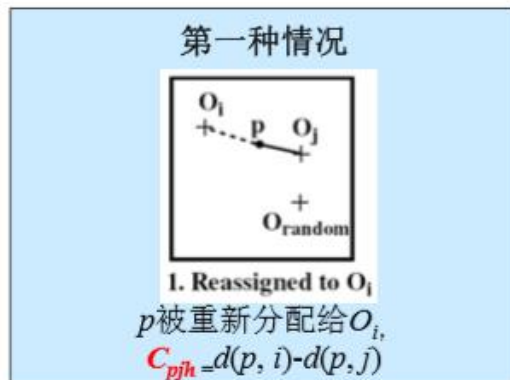
迭代次数	平均值 (簇1)	平均值 (簇2)	产生的新簇	新平均值 (簇1)	新平均值 (簇2)
1	(1, 1)	(1, 2)	{1, 2}, {3, 4, 5, 6, 7, 8}	(1.5, 1)	(3.5, 3)
2	(1.5, 1)	(3.5, 3)	{1, 2, 3, 4}, {5, 6, 7, 8}	(1.5, 1.5)	(4.5, 3.5)
3	(1.5, 1.5)	(4.5, 3.5)	{1, 2, 3, 4}, {5, 6, 7, 8}	(1.5, 1.5)	(4.5, 3.5)

k-medoids—例子

基本思想：

选取有代表性的样本（而不是均值）来表示整个簇，即：选取靠近中心点(medoid)的那个样本来代表整个簇。

以降低聚类算法对离群点的敏感度。

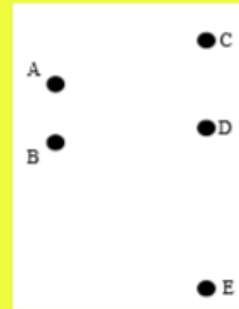


注： $d(p, \text{新中心点}) - d(p, \text{旧中心点})$

例子

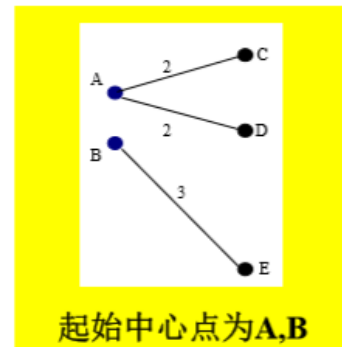
假设空间中的五个点{A、B、C、D、E}，如下图所示。各点之间的距离关系如下表所示，根据所给的数据对其运行k-medoids算法实现划分聚类（设 $k=2$ ）。

样本点	A	B	C	D	E
A	0	1	2	2	3
B	1	0	2	4	3
C	2	2	0	1	5
D	2	4	1	0	3
E	3	3	5	3	0



■ 第一步 建立阶段：假如从5个对象中随机抽取的2个中心点为{A, B}，则样本被划分为{A、C、D}和{B、E}，如图所示。

■ 第二步 交换阶段：假定中心点A、B分别被非中心点{C、D、E}替换，根据PAM算法需要计算下列代价 TC_{AC} 、 TC_{AD} 、 TC_{AE} 、 TC_{BC} 、 TC_{BD} 、 TC_{BE} 。



以 TC_{AC} 为例说明计算的过程：

- 当A被C替换以后，A不再是一个中心点，因为A离B比A离C更近，A被分配到B中心点代表的簇， $C_{AAC}=d(A,B)-d(A,A)=1$ 。
 - B是一个中心点，当A被C替换以后，B不受影响， $C_{BAC}=0$ 。
 - C原先属于A中心点所在的簇，当A被C替换以后，C是新中心点，符合PAM算法代价函数的第二种情况 $C_{CAC}=d(C,C)-d(C,A)=0-2=-2$ 。
 - D原先属于A中心点所在的簇，当A被C替换以后，离D近的中心点是C，根据PAM算法代价函数的第二种情况 $C_{DAC}=d(D,C)-d(D,A)=1-2=-1$ 。
 - E原先属于B中心点所在的簇，当A被C替换以后，离E近的中心仍然是B，根据PAM算法代价函数的第三种情况 $C_{EAC}=0$ 。
- 因此， $TC_{AC}=C_{AAC}+C_{BAC}+C_{CAC}+C_{DAC}+C_{EAC}=1+0-2-1+0=-2$ 。

TC_{AD} 计算过程：

- a) 当 A 被 D 替换以后, A 不再是一个中心点, 因为 A 离 B 比 A 离 C 更近, A 被分配到 B 中心点代表的簇, $C_{AAD}=d(A,B)-d(A,A)=1$ 。
- b) B 是一个中心点, 当 A 被 C 替换以后, B 不受影响, $C_{BAD}=0$ 。
- c) C 原先属于 A 中心点所在的簇, 当 A 被 D 替换以后, 离 C 近的中心点是 D, 根据 PAM 算法代价函数的第二种情况 $C_{CAD}=d(C,D)-d(C,A)=1-2=-1$ 。
- d) D 原先属于 A 中心点所在的簇, 当 A 被 D 替换以后, D 是新中心点, 根据 PAM 算法代价函数的第二种情况 $C_{DAD}=d(D,D)-d(D,A)=0-2=-2$ 。
- e) E 原先属于 B 中心点所在的簇, 当 A 被 D 替换以后, 离 E 近的中心仍然是 B, 根据 PAM 算法代价函数的第三种情况 $C_{EAD}=0$ 。
- 因此, $TC_{AD}=C_{AAD}+C_{BAD}+C_{CAD}+C_{DAD}+C_{EAD}=1+0-1-2+0=-2$ 。

TC_{AE} 计算过程:

- a) 当 A 被 E 替换以后, A 不再是一个中心点, 因为 A 离 B 比 A 离 C 更近, A 被分配到 B 中心点代表的簇, $C_{AAE}=d(A,B)-d(A,A)=1$ 。
- b) B 是一个中心点, 当 A 被 C 替换以后, B 不受影响, $C_{BAE}=0$ 。
- c) C 原先属于 A 中心点所在的簇, 当 A 被 E 替换以后, 离 C 近的中心点是 B, 根据 PAM 算法代价函数的第二种情况 $C_{CAE}=d(C,B)-d(C,A)=2-2=0$ 。
- d) D 原先属于 A 中心点所在的簇, 当 A 被 E 替换以后, 离 D 近的中心是 E, 根据 PAM 算法代价函数的第二种情况 $C_{DAE}=d(D,E)-d(D,A)=3-2=1$ 。
- e) E 原先属于 B 中心点所在的簇, 当 A 被 E 替换以后, E 是新中心点, 根据 PAM 算法代价函数的第二种情况 $C_{EAE}=d(E,E)-d(E,B)=0-3=-3$ 。
- 因此, $TC_{AE}=C_{AAE}+C_{BAE}+C_{CAE}+C_{DAE}+C_{EAE}=1+0+0+1-3=-1$ 。

TC_{BC} 计算过程:

- a) A 是一个中心点, 当 B 被 C 替换以后, A 不受影响, $C_{ABC}=0$ 。
- b) 当 B 被 C 替换以后, B 不再是一个中心点, 因为 B 离 A 比 B 离 C 更近, B 被分配到 A 中心点代表的簇, $C_{BBC}=d(B,A)-d(B,B)=1$ 。
- c) C 原先属于 A 中心点所在的簇, 当 B 被 C 替换以后, C 是新中心点, 符合 PAM 算法代价函数的第二种情况 $C_{CBC}=d(C,C)-d(C,B)=0-2=-2$ 。
- d) D 原先属于 A 中心点所在的簇, 当 B 被 C 替换以后, 离 D 近的中心是 C, 根据 PAM 算法代价函数的第二种情况 $C_{DBC}=d(D,C)-d(D,A)=1-2=-1$ 。
- e) E 原先属于 B 中心点所在的簇, 当 B 被 C 替换以后, 离 E 近的中心是 A, 根据 PAM 算法代价函数的第二种情况 $C_{EBC}=d(E,A)-d(E,B)=3-3=0$ 。
- 因此, $TC_{BC}=C_{ABC}+C_{BBC}+C_{CBC}+C_{DBC}+C_{EBC}=0+1-2-1+0=-2$ 。

k-means 更加稳定, k-medoids 执行代价高

层次法

自顶向下 (合并) - Agnes

合并准则: 每次找到距离最近的两个簇进行合并。

两个簇之间的距离由这两个簇中最近的样本点之间的距离来表示

单链接方法用于确定任意两个簇之间的距离；

相异度矩阵用于记录任意两个簇之间的距离（它是一个下三角矩阵，即：主对角线及其上方元素全部为零）。

例：为了研究辽宁省等五省区某年度城镇居民生活消费的分布规律，对如下调查数据进行聚类。

表1 数据集

省份	X1	X2	X3	X4	X5	X6	X7	X8
辽宁	7.90	39.77	8.49	12.94	19.27	11.05	2.04	13.29
浙江	7.68	50.37	11.35	13.30	19.25	14.59	2.75	14.87
河南	9.42	27.93	8.20	8.14	16.17	9.42	1.55	9.76
甘肃	9.16	27.8	9.01	9.32	15.99	9.10	1.82	11.35
青海	10.06	28.64	10.52	10.05	16.18	8.39	1.96	10.81

$G_1 = \{\text{辽宁}\}$, $G_2 = \{\text{浙江}\}$, $G_3 = \{\text{河南}\}$, $G_4 = \{\text{甘肃}\}$, $G_5 = \{\text{青海}\}$

采用欧氏距离：

$$d_{12} = 11.67$$

$$d_{13} = 13.80 \quad d_{14} = 13.12 \quad d_{15} = 12.80 \quad d_{23} = 24.63 \quad d_{24} = 24.06 \quad d_{25} = 23.54 \quad d_{34} = 2.20$$

$$D_1 = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{pmatrix} 0 & & & & \\ 11.67 & 0 & & & \\ 13.80 & 24.63 & 0 & & \\ 13.12 & 24.06 & 2.20 & 0 & \\ 12.80 & 23.54 & 3.51 & 2.21 & 0 \end{pmatrix} \end{matrix}$$

河南与甘肃的距离最近，先将二者（3和4）合为一类 $G_6 = \{G_3, G_4\}$

$G_1 = \{\text{辽宁}\}$, $G_2 = \{\text{浙江}\}$, $G_3 = \{\text{河南}\}$, $G_4 = \{\text{甘肃}\}$, $G_5 = \{\text{青海}\}$

采用欧氏距离:

$$d_{61} = d_{(3,4)1} = \min\{d_{13}, d_{14}\} = 13.12 \quad d_{62} = d_{(3,4)2} = \min\{d_{23}, d_{24}\} = 24.06$$

$$d_{65} = d_{(3,4)5} = \min\{d_{35}, d_{45}\} = 2.21$$

$$D_2 = \begin{matrix} & \begin{matrix} 6 & 1 & 2 & 5 \end{matrix} \\ \begin{matrix} 6 \\ 1 \\ 2 \\ 5 \end{matrix} & \begin{pmatrix} 0 & & & \\ 13.12 & 0 & & \\ 24.06 & 11.67 & 0 & \\ 2.21 & 12.80 & 23.54 & 0 \end{pmatrix} \end{matrix} \quad \longrightarrow \quad \begin{matrix} \text{河南、甘肃与青海并为一新类} \\ G_7 = \{G_6, G_5\} = \{G_3, G_4, G_5\} \end{matrix}$$

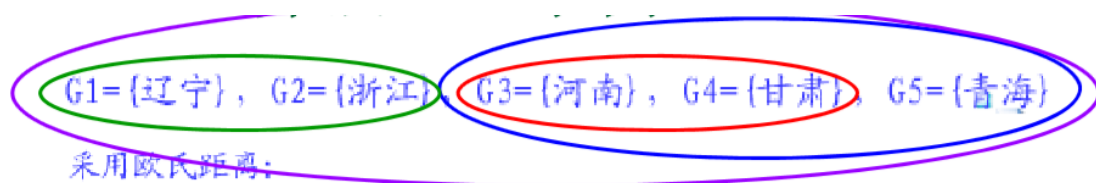
$G_1 = \{\text{辽宁}\}$, $G_2 = \{\text{浙江}\}$, $G_3 = \{\text{河南}\}$, $G_4 = \{\text{甘肃}\}$, $G_5 = \{\text{青海}\}$

采用欧氏距离:

$$d_{71} = d_{(3,4,5)1} = \min\{d_{13}, d_{14}, d_{15}\} = 12.80$$

$$d_{72} = d_{(3,4,5)2} = \min\{d_{23}, d_{24}, d_{25}\} = 23.54$$

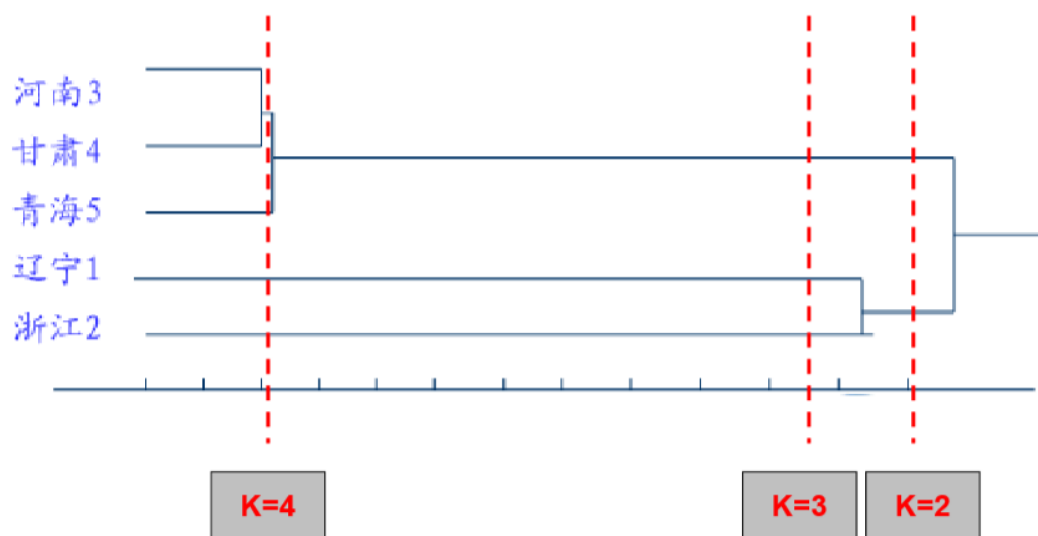
$$D_3 = \begin{matrix} & \begin{matrix} 7 & 1 & 2 \end{matrix} \\ \begin{matrix} 7 \\ 1 \\ 2 \end{matrix} & \begin{pmatrix} 0 & & \\ 12.80 & 0 & \\ 23.54 & 11.67 & 0 \end{pmatrix} \end{matrix} \quad \longrightarrow \quad G_8 = \{G_1, G_2\}$$



$$d_{78} = \min\{d_{71}, d_{72}\} = 12.80$$

$$D_4 = \begin{matrix} & \begin{matrix} 7 & 8 \end{matrix} \\ \begin{matrix} 7 \\ 8 \end{matrix} & \begin{bmatrix} 0 & 8 \\ 12.8 & 0 \end{bmatrix} \end{matrix} \longrightarrow G9 = \{G7, G8\}$$

$G1 = \{\text{辽宁}\}, G2 = \{\text{浙江}\}, G3 = \{\text{河南}\}, G4 = \{\text{甘肃}\}, G5 = \{\text{青海}\}$



自底向上（分裂）-DIana


例：有如下表所示的数据集，使用 DIANA 算法对该数据集进行分裂层次聚类。

序号	属性 1	属性 2
1	1	1
2	1	2
3	2	1

4	2	2
5	3	4
6	3	5
7	4	4
8	4	5

第 1 步，首先找到具有大直径的簇，然后计算该簇中每个样本的平均相异度（假定采用是欧式距离）。

序号	属性 1	属性 2								
1	1	1								
2	1	2								
3	2	1								
4	2	2								
5	3	4								
6	3	5								
7	4	4								
8	4	5								



0									
1	0								
1	1.4	0							
1.4	1	1	0						
3.6	2.8	3.2	2.2	0					
4.5	3.6	4.1	3.2	1	0				
4.2	3.6	3.6	2.8	1	1.4	0			
5	4.2	4.5	3.6	1.4	1	1	0		

第 1 步，首先找到具有大直径的簇，然后计算该簇中每个样本的平均相异度（假定采用是欧式距离）。

- 样本 1 的平均距离为：(1+1+1.4+3.6+4.5+4.2+5)/7=2.96
- 样本 2 的平均距离为：2.53
- 样本 3 的平均距离为：2.68
- 样本 4 的平均距离为：2.18
- 样本 5 的平均距离为：2.18
- 样本 6 的平均距离为：2.68
- 样本 7 的平均距离为：2.53
- 样本 8 的平均距离为：2.96

从上述值中挑出具有大平均相异度的样本 1(或者样本 8)，将其放到 splinter group 中，剩余点在 old party 中。

第六章 文本与 Web 挖掘

布尔模型 特点

关键字匹配、不能排序

向量空间模型：将一组文档表示为公共向量空间中的向量

将每篇文档表示成实数型分量所构成的向量，每个分量对应一个词项，并往往采用 **tf-idf** 权重来计算

两篇文档的相似度计算：

- 1) 将两个文档向量做差运算。
- 2) 求文档向量的余弦相似度 (cosine similarity)

Idf、tf-idf

文档频率 (document frequency) 表示的是出现 **t** 的所有文档的数目，记为 **df_t**

逆文档频率 (inverse document frequency) 由于 **df** 本身往往较大，所以通常需要将它映射到一个较小的取值范围中去。定义如下

$$\text{idf}_t = \log \frac{N}{df_t}$$

tf-idf 权重计算

对于每篇文档中的每个词项，可以将其 **tf** 和 **idf** 组合在一起形成最终的权重。

tf-idf 权重机制对文档 **d** 中的词项 **t** 赋予的权重如下：

$$\text{tf-idf}_{t,d} = tf_{t,d} \times \text{idf}_t$$

维度的确定—词汇数量决定

欧氏距离 升序

cos 降序

支持向量文本分类

Rocchio 分类方法：基于质心或原型(prototype)将整个向量空间划分成多个区域。每个质心或原型代表一类，计算该类中所有文档的质心。

步骤：

- (1) 先把属于一个类别的样本文档转换成文档向量；
- (2) 求属于一个类别的样本文档的质心向量（原型向量）；
- (3) 判断新文档属于哪个类别。

k 近邻(kNN, k-NearestNeighbor)分类算法通过局部信息来确定类别边界

kNN 的基本依据在于：邻近假设，一篇测试文档 d 将和其邻域中的训练文档应该具有相同的类别。

语言模型：词汇概率分布

一元语言模型：去掉所有条件概率中的条件，独立地估计每个词项的概率

二元语言模型：计算条件概率时只考虑前一个词项的出现情况

N 元语言模型

极大似然相似估计—词项概率

查询似然模型：对文档集中的每篇文档 d 构建其对应的语言模型 M_d 。

目标是将文档按照其与查询相关的似然 $P(d|q)$ 排序

$$P(d|q)=p(q|d)$$

检索的具体方法：

- (1) 对每篇文档推导出其语言模型;
- (2) 估计查询在每个文档 d_i 的语言模型下的生成概率 $P(q|M d_i)$
- (3) 按照上述概率对文档进行排序。

文本分类

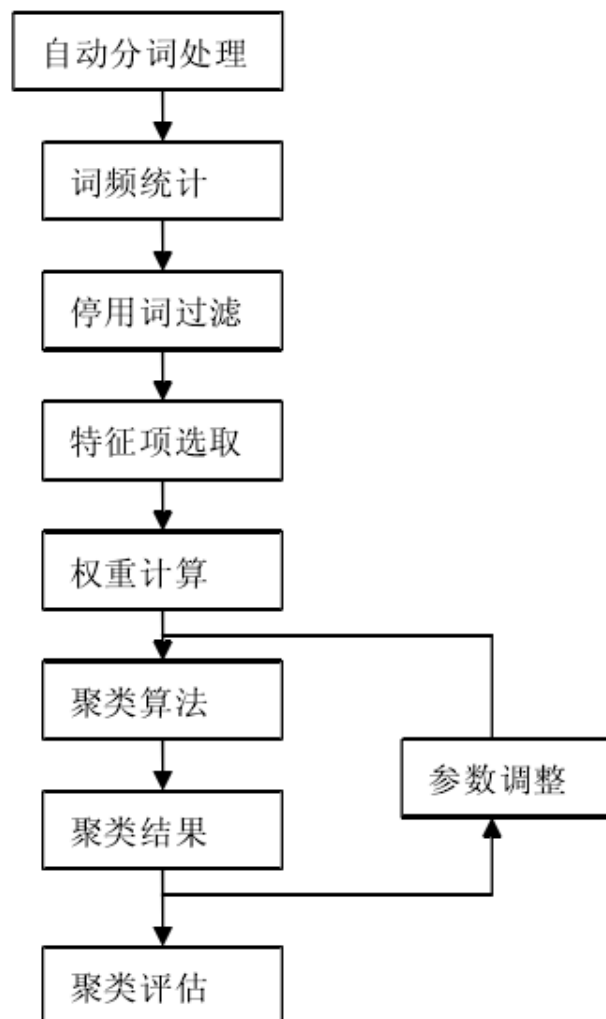
- 文本分类中，给定文档 $d \in X$ 和一个固定的类别集合 C ，其中 X 表示文档空间，类别（class）也通常称为类 category 或类标签 label。将文档分到某个类别。

朴素贝叶斯文本分类

多项式朴素贝叶斯是一种基于概率的学习方法。该方法中，计算文档 d 属于类别 c 的概率。

多元贝努利模型

文本聚类—无监督学习



扁平聚类--给出一系列扁平结构的簇，簇间无结构表明关联性

层次聚类--层次性聚类结果

硬聚类：每篇文档仅属于一个簇 k-means

软聚类：一篇文档可能属于多个簇 EM 算法(期望最大化算法)

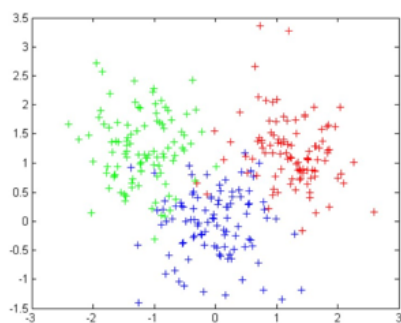


图1. 扁平聚类

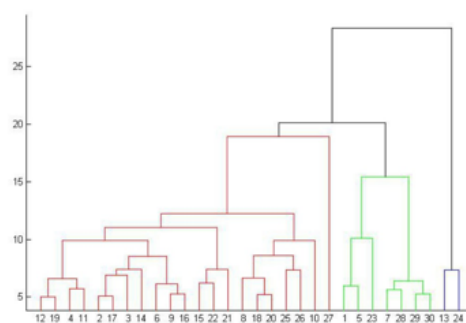


图2. 层次聚类

层次聚类

➤ 层次凝聚式聚类

在不同层次对数据集进行划分，从而形成一棵二叉树形式的类别层

对于单篇文档构成的簇，它的结合相似度定义成文档的**自相似度**，当采用**余弦相似度**计算时，这个值为 1.0。

每次选择**最好的**合并，即**结合相似度最大的**

➤ 单连接及全连接聚类算法

单连接：**最大相似度**

计算任意两篇文档之间的相似度，取其中的最大值

全连接：**最小相似度**

计算任意两篇文档之间的相似度，取其中的最小值

单连接形成的簇不均匀，全连接比较均匀

单连接方法中计算的是两篇最相似的文档之间的相似度；全连接方法中计算的是两篇最不相似的文档之间的相似度

➤ 组平均凝聚式聚类

计算所有文档之间相似度的平均值 **SIM-GA**，其中也包括来自同一簇的文档。

➤ 质心聚类

通过两个簇的质心相似度来定义这两个簇的相似度。

GAAC 在计算平均相似度时考虑了所有文档之间的相似度，而质心聚类中仅仅考虑来自同簇的文档之间的相似度

➤ 分裂式聚类

簇层次结构也可以自顶向下生成

索引构建

➤ 基于块的排序索引方法

- 内存式单遍扫描索引构建方法
- 分布式索引构建方法
- 动态索引构建方法

通配查询：指对词项中带有*通配符的查询

作用：

用户需要进行拼写不确定的查询；(sydney/sidney—>s*dney)

用户需要查找某个查询词项的所有变形；(judicia*)

用于一般通配符查询的索引：轮排(permuterm) 索引（是倒排索引的一种特殊形式）

基本思想：

- 将每个通配查询旋转，使*出现在末尾
- 引入一个新的符号\$，用于标识词项结束。例如：hello——>hello\$
- 构造一个轮排索引，对扩展词项的每个旋转结果都构造一个指针来指向原始词项。

轮排索引开销大

k-gram 索引：枚举一个词项中所有连读的 k 个字符构成 k-gram

例如：对于词项 castle 来说，cas、ast、stl 都是 3-gram

自己讲的部分