数据挖掘与算法

hycoder

一、名词翻译

英文	中文
k-Nearest Neighbors	k最近邻
Clustering	聚类

英文	中文
Data Classification	数据分类
Sequential Mining	序列挖掘
Web Content Mining	Web内容挖掘
Web Structure Mining	Web结构挖掘
Time Series	时间序列
Geographic Information System	地理信息系统
Data Mining	数据挖掘
Artificial Intelligence	人工智能
Knowledge Discovery	知识发现
OLAP(On-Line Analytic Processing)	在线分析处理
Association Rule	关联规则
Maximal Frequent ItemSet	最大频繁项目集
Data Classificaiton	数据分类
Decision Tree	决策树

二、名词解释

2.1 数据挖掘

数据挖掘：定义有广义和狭义之分。从广义的观点，数据挖据是从大型数据集中**挖掘隐含在其中的、人们事先不知道的、对决策有用的知识**的完整过程。从狭义的观点上出发，我们定义数据挖掘从特定形式的数据集中提拣知识的过程。(P12)

2.2 频繁项目集

频繁项目集：是指**出现频率最高的项目**对应的集合，反映交易数据库中项目出现的频度信息。挖掘频繁项目集是关联规则挖掘的基础。许多关联规则挖掘是基于频繁项目集发现的。(P74)

2.3 分类

分类：可以看作是从数据库到一组**预先定义的、非交叠的类别**的映射。数据挖掘中分类的主要任务是构造分类器，需要有一个训练样本数据集作为输入。分类的**目的**是分析输入数据，为每一个类找到一种准确的描述或者模型。(P166)

2.4 聚类

聚类：就是将数据对象分组成为多个类或簇，**划分的原则**是在同一个簇中的对象之间具有较高的相似度，而不同簇中的对象差别敏大。与分类不同的是，聚类操作中要划分的类是事先未知的，类的形成完全是数据驱动的，属种无指导的学习方法。(P172)

2.5 层次聚类法

层次聚类法：是对给定数据对象集合进行层次的分解。其基本思想是将数据样本按距离准则逐步聚类，直到满足分类要求为止。一般分为凝聚的和分裂的两种方案。(P188)

2.6 序列挖掘

序列挖掘：是指从数据库中发现相对**时间或者其他顺序**出现的高频率子序列。(P204)

2.7 时间序列数据挖掘

时间序列数据挖掘：就是要从**大量的时间序列数据中提取**人们实现不知道的，但又是潜在有用的与时间属性相关的信息和知识，并且用于短期、中期或长期预测，指导人们的社会、经济、军事和生活等行为。(P204)

2.8 后验概率

后验概率：又称条件概率，是在已知结果发生的情况下，求导致结果的某种原因的可能性的大小。(P140)

2.9 数字地球

数字地球：是一种关于地球嵌入海量地理数据的、多分辨率的和三维的表示，它提供一种机制引导用户寻找地理信息，可供生产者出版。(P372)

2.10 分类与聚类区别

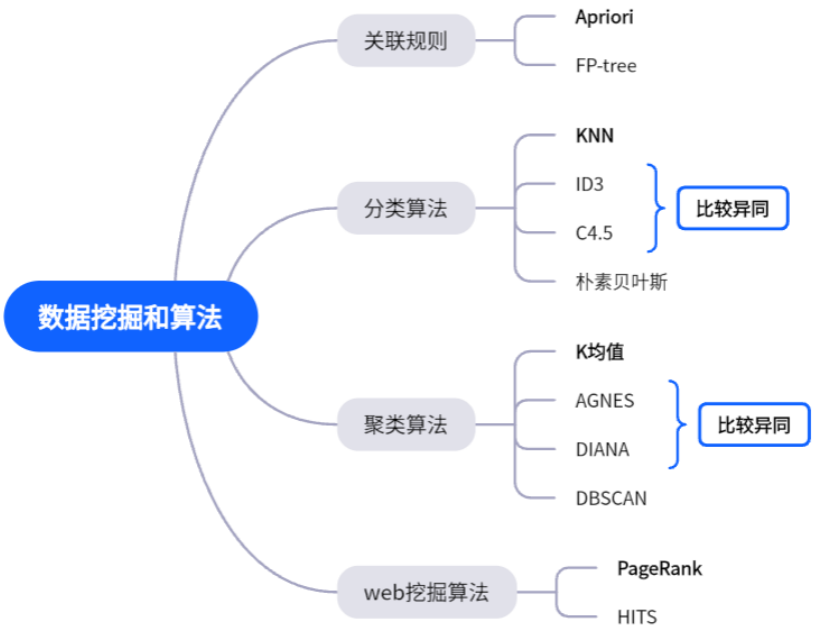
聚类：

- 在没有训练的条件下，把样本划分为若干类
- 无类别标签的样本
- 无监督学习

分类：

- 已知存在哪些类，将每一条记录分别属于哪一类标记出来。
- 有类别标签的样本
- 有监督学习

三、相关算法



考试重点掌握Apriori、KNN、K均值和PageRank四个算法的有关概念、算法思想、操作流程和计算相关值。比较ID3和C4.5的异同、AGNES和DIANA算法的异同。

3.1 Apriori算法

- 算法目标：找到事物之间潜在的关联关系(商品摆放：牛奶与尿布)。
- 相关概念：
  1. **项集**：在关联分析中，包含0个或多个项的集合被称为项集(itemset)。如果一个项集包含k个项，则称它为k-项集。例如:{床单，枕套，牛奶，花生} 是一个4-项集。空集是指不包含任何项的项集。
  2. **频数**：一个项集x在数据库D中出现的次数，count。

     X={beer，diaper}，是几项集，频数是多少？

     X={beer，diaper}，2-项集，count（X）=3

交易号（TID）	商品（Items）
1	beer, diaper, nuts
2	beer, biscuit, diaper
3	bread, butter, cheese
4	beer, cheese, diaper, nuts
5	beer, butter, cheese, nuts

  3. **关联规则**(association rule)：是形如 $X \rightarrow Y$ 的蕴含表达式，其中X和Y是不相交的项集，即：$X \cap Y = \varnothing$。
  4. **支持度**(Support)：一个项集或者规则在事物数据库中出现的频率。而**最小支持度**(Minsupport)是指用户给定(参数)的最小支持度%。

$$support(x) = \frac{count(x)}{|D|} * 100\%$$

其中$|D|$表示D的模，即：事务数据库中交易的次数(说白了就是给的数据集表格中数据有多少行)

显然可以得到最小支持数的计算公式：

$$minsup\_count = minsupport * |D|$$

最小支持数的意义是用于Apriori算法中评判当前项目集(Itemset)是否加入N-频繁项目集。

举例如下：

交易号码	商品
0	豆奶，莴苣
1	莴苣，尿布，葡萄酒，甜菜
2	豆奶，尿布，葡萄酒，橙汁
3	莴苣，豆奶，尿布，葡萄酒
4	莴苣，豆奶，尿布，橙汁

{豆奶}的支持度　　为4/5
{尿布}的支持度　　为4/5
{豆奶、尿布}的支持度　为3/5

  5. **频繁项目集**(Frequent Itemsets)：**一个项集X的支持度大于用户给定的一个最小支持度阈值**，则X被称为**频繁项集**(或频繁模式)，X是频繁的。最大频繁项目集：在频繁项目集中挑选出所有不被其他元素包含的频繁项目集。

6. **置信度或信任度**(Confidence)：针对某一关联规则$I_1 \Rightarrow I_2$的可信度的值为I_1、I_2项集出现的概率比上I_1项集出现的概率：

$$Confidence(I_1 \Rightarrow I_2) = \frac{Support(I_1 \cup I_2)}{Support(I_1)}$$
$$s.t: I_1, I_2 \subseteq I; I_1 \cap I_2 = \varnothing$$

当然，很显然在同一事务数据库中其事务交易次数为一致的，即$|D|$相同，因此上式可以改写为：

$$Confidence(I_1 \Rightarrow I_2) = \frac{count(I_1 \cup I_2)}{count(I_1)}$$
$$s.t: I_1, I_2 \subseteq I; I_1 \cap I_2 = \varnothing$$

其中**最小置信度**(Minconfidence)是给定的具体参数。

举例如下：

交易号（TID）	商品（Items）
1	beer, diaper, nuts
2	beer, biscuit, diaper
3	bread, butter, cheese
4	beer, cheese, diaper, nuts
5	beer, butter, cheese, nuts

计算规则{beer、diaper}→nuts的置信度？

confidence({beer, diaper}→nuts)=66.7%。

项集{beer,diaper,nuts}出现次数为2，{beer,diaper}出现次数为3。

因此其关联规则$\{beer, diaper\} \Rightarrow \{nuts\}$最小置信度的值为$\frac{2}{3}$。

7. **强关联规则**(Strong Association Rule)：D在I上满足**最小支持度**和**最小信任度**的关联规则称为强关联规则。

8. **项目集空间理论**：频繁项目集的子集是频繁项目集；非频繁项目集的超集是非频繁项目集。具体证明过程在第五章给出。

- **Apriori算法的两个致命的性能瓶颈**

1. 多次扫描事务数据库，需要很大的I/O负载。

   对每次k循环，侯选集C_k中的每个元素都必须通过扫描数据库一次来验证其是否加入L_k。假如有一个最大频繁项目集包含10个项的话，那么就至少需要扫描事务数据库10遍。

2. 可能产生庞大的侯选集

   由L_{k-1}产生k-侯选集C_k是指数增长的，例如10^4个1-频繁项目集就有可能产生接近10^7个元素的2-侯选集。如此大的侯选集对时间和主存空间都是一种挑战。

- 例题：

给出样本事务数据库且$minsupport = 40\%$，$minconfidence = 60\%$。

TID	Itemset
1	A，B，C，D
2	B，C，E
3	A，B，C，E
4	B，D，E
5	A，B，C，D

解题：

1. **发现频繁项目集**

首先最小支持数 $= minsupport * |D| = 40\% * 5 = 2$ 。

(1) L_1 生成：

生成 **候选集** 并通过扫描数据库得到它们的支持数，C_1 = {(A，3)，(B，5)，(C，4)，(D，3)，(E，3)}；满足大于最小支持数的项目集组成1-频繁项目集$L_1 = \{A，B，C，D，E\}$。

(2) L_2 生成：

由L_1 生成 **2-候选集** 并通过扫描数据库得到它们的支持数C_2 = {(AB，3)，(AC，3)，(AD，2)，(AE，1)，(BC，4)，(BD，3)，(BE，3)，(CD，2)，(CE，2)，(DE，1)}；满足大于最小支持数的最小支持数(挑选$minsup_count \geqslant 2$)的项目集组成2-频繁项目集
$L_2 = \{AB，AC，AD，BC，BD，BE，CD，CE\}$。

(3) L_3 生成：

由L_2 生成 **3-候选集** 并通过扫描数据库得到它们的支持数C_3 = {(ABC，3)，(ABD，2)，(ACD，2)，(BCD，2)，(BCE，2)}；满足大于最小支持数的最小支持数的项目集组成3-频繁项目集
$L_3 = \{ABC，ABD，ACD，BCD，BCE\}$。

注意：*3-候选集* 中不含有 (BDE,1) 的原因是：在 L_2 中 DE 已经确定为不满足频繁项目集了，即根据 *非频繁项目集的超集是非频繁项目集* ，因此没有必要将 (BDE,1) 加入 *3-候选集* 。

(4) L_4 生成：

由L_3 生成 **4-候选集** 并通过扫描数据库得到它们的支持数C_4 = {(ABCD，2)}；满足大于最小支持数的最小支持数(挑选$minsup_count \geqslant 2$)的项目集组成3-频繁项目集$L_4 = \{ABCD\}$。

注意：*4-候选集* 中不含有 (ABCE,1) 的原因是：在 L_2 中 AE 已经确定为不满足频繁项目集了，即根据 *非频繁项目集的超集是非频繁项目集* ，因此没有必要将 (ABCE,1) 加入 *4-候选集* 。

(5) L_5 生成：

由L_4 生成 **5-候选集** $C_5 = \varnothing, L_5 = \varnothing$，算法停止。

于是所有的**频繁项目集**为{A，B，C，D，E，AB，AC，AD，BC，BD，BE，CD，CE，ABC，ABD，ACD，BCD，ABCD}。且易知 **最大频繁项目集** 为{ABCD，BCE}。

2. **关联规则生成**

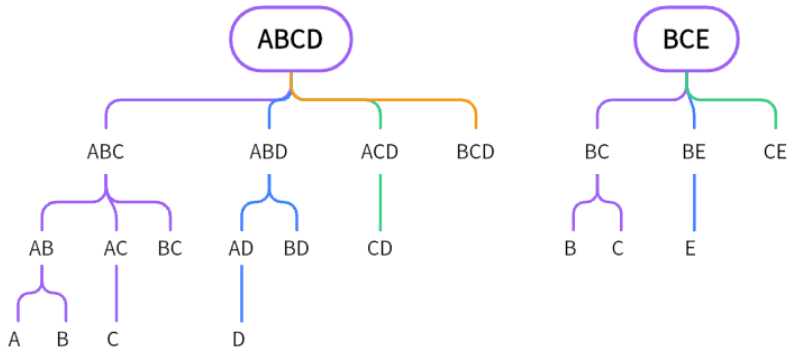
采用深度优先算法展示关联规则生成过程：

序号	[illegible]	[illegible]	[illegible]	[illegible]	规则(是否为强规则)
1	ABCD	ABC	67%	40%	$ABC \Rightarrow D$ (是)
2	ABCD	AB	67%	40%	$AB \Rightarrow CD$ (是)
3	ABCD	A	67%	40%	$A \Rightarrow BCD$ (是)
4	ABCD	B	40%	40%	$B \Rightarrow ACD$ (否)
5	ABCD	AC	67%	40%	$AC \Rightarrow BD$ (是)
6	ABCD	C	50%	40%	$C \Rightarrow ABD$ (否)
7	ABCD	BC	50%	40%	$BC \Rightarrow AD$ (否)
8	ABCD	ABD	100%	40%	$ABD \Rightarrow C$ (是)
9	ABCD	AD	100%	40%	$AD \Rightarrow BC$ (是)
10	ABCD	D	67%	40%	$D \Rightarrow ABC$ (是)
11	ABCD	BD	67%	40%	$BD \Rightarrow AC$ (是)
12	ABCD	ACD	100%	40%	$ACD \Rightarrow B$ (是)
13	ABCD	CD	100%	40%	$CD \Rightarrow AB$ (是)

序号	l_k	x_{m-1}	$confidence(\frac{l_k}{x_{m-1}})$	$support(l_k)$	规则(是否为强规则)
14	ABCD	BCD	100%	40%	$BCD \Rightarrow A$ (是)
15	BCE	BC	50%	40%	$BC \Rightarrow E$ (否)
16	BCE	B	40%	40%	$B \Rightarrow CE$ (否)
17	BCE	C	50%	40%	$C \Rightarrow BE$ (否)
18	BCE	BE	67%	40%	$BE \Rightarrow C$ (是)
19	BCE	E	67%	40%	$E \Rightarrow BC$ (是)
20	BCE	CE	100%	40%	$CE \Rightarrow B$ (是)

注释:

- l_k 表示最大频繁项目集；x_{m-1} 表示非空子集；confidence是$x_{m-1} \Rightarrow l_k - x_{m-1}$的规则信任度；support是项目集$l_k$ 的支持度。
- 采用深度优先展示。那么具体顺序可以画树形结构采用先序遍历得到：



- 某关联规则的信任度(confidence)：

$$Confidence(I_1 \Rightarrow I_2) = \frac{count(I_1 \cup I_2)}{count(I_1)}$$
$$s.t: I_1, I_2 \subseteq I; I_1 \cap I_2 = \varnothing$$

  根据上式和表格元素含义易得：

$$Confidence = \frac{l_k}{x_{m-1}}$$

- 某项目集的支持度(support)：

$$support(x) = \frac{count(x)}{|D|} * 100\%$$

  其中$|D|$表示D的模，即：事务数据库中交易的次数(说白了就是给的数据集表格中数据有多少行)

- 判断是否为强关联规则：

  满足$confidence \geqslant minconfidence$且$support \geqslant minspport$即为强关联规则。

总结:

1. 确定最小支持度和最小信任度
2. 计算出最小支持数
3. 迭代找到候选集和频繁项目集(支持数和判别)
4. 找到最大频繁项目集
5. 画出 **关联规则生成过程表** (序号、l_k、x_{m-1}、confidence、suppport、规则)

- l_k：最大频繁项目集
- x_{m-1}：根据l_k采用深度或广度得到
- confidence：$\frac{l_k}{x_{m-1}}$
- support：是l_k的支持度
- 规则：$x_{m-1} \Rightarrow l_k - x_{m-1}$

3.2 FP-tree算法

- 基本原理：进行2次数据库扫描：一次对所有1-项目的频度排序；一次将数据库信息转变成紧缩内存结构。不使用侯选集，直接压缩数据库成一个频繁模式树，通过频繁模式树可以直接得到频集。
- 算法步骤：
  1. **两次扫描数据库，生成频繁模式树FP-Tree：**
     - 扫描数据库一次，得到所有1-项目的频度排序表T；
     - 依照T，再扫描数据库，得到FP-Tree。
  2. **使用FP-Tree，生成频集(挖掘频繁模式，参考：FP Tree算法原理总结)：**
     - 为FP-tree中的每个节点生成**条件模式库**；
     - 用条件模式库构造对应的条件FP-tree；
     - 递归挖掘条件FP-trees同时增长其包含的频繁集；
     - 如果**条件FP-tree只包含一个路径，则直接生成所包含的频繁集。**
- 例题：

  给定项目集，要求画出FP-tree，给出以m为后缀模式的条件模式基。该例题最小支持数阈值是3。

TID数据库	原始项目集
100	f,a,c,d,g,i,m,p
200	a,b,c,f,l,o
300	b,f,h,j,m,p
400	b,c,k,m,o,s
500	a,f,c,e,l,n,o,p

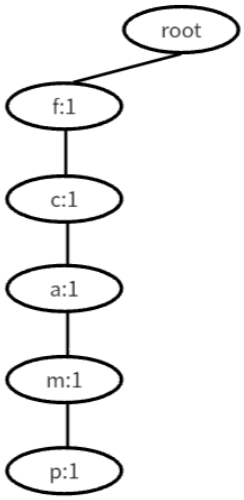
  解题：

  (1) 第一次扫描数据库，导出 1-频繁项集 $L = [f4, c4, a3, b3, m3, o3, p3]$(字母表示项目集中元素，数字表示支持数)。因此，可以通过**去掉**不频繁的单项目来简化原始的数据库元组，并按照项目的**支持数高低排序**整理(**支持数相同的按照ASCII码排序**)，如下表所示。

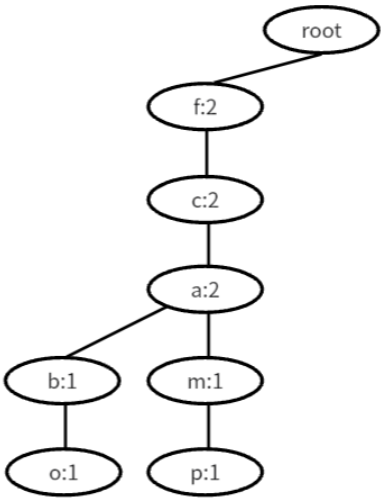
TID数据库	原始项目集	整理后的项目集
100	f,a,c,d,g,i,m,p	f,c,a,m,p
200	a,b,c,f,l,o	f,c,a,b,o
300	b,f,h,j,m,p	f,b,m,p
400	b,c,k,m,o,s	c,b,m,o
500	a,f,c,e,l,n,o,p	f,c,a,o,p

  (2)创建树的根节点，用"root"标记。第二次扫描数据库DB，对每一个事务创建一个分支。

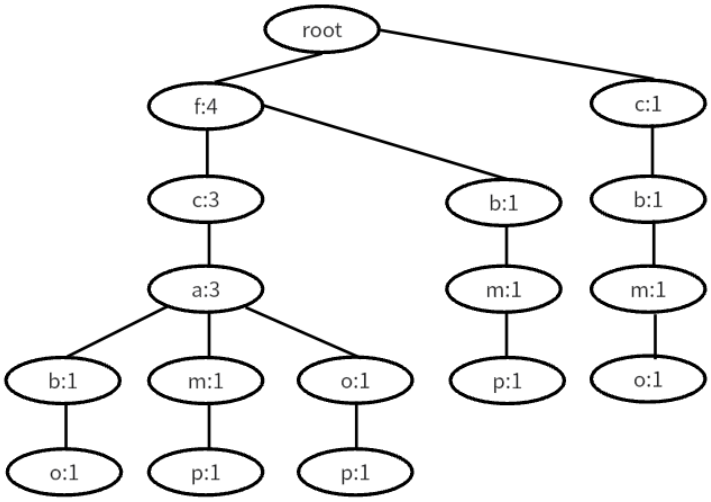
- 第一个事务“T100：f,a,c,d,g,i,m,p”按L的次序包含5个项{f,c,a,m,p}，得到构造树的第一个分支
  $< f1 \rightarrow c1 \rightarrow a1 \rightarrow m1 \rightarrow p1 >$



- 对于第二个事务，由于其排序后的频繁项表<f,c,a,b,o>与已有的分支路径<f,c,a,m,p>共享前缀<f,c,a>,所有前缀中的每个节点计数加1，只创建两个新的节点b1和o1，形成新的分支链
  $< f2 \rightarrow c2 \rightarrow a2 \rightarrow b1 \rightarrow o1 >$

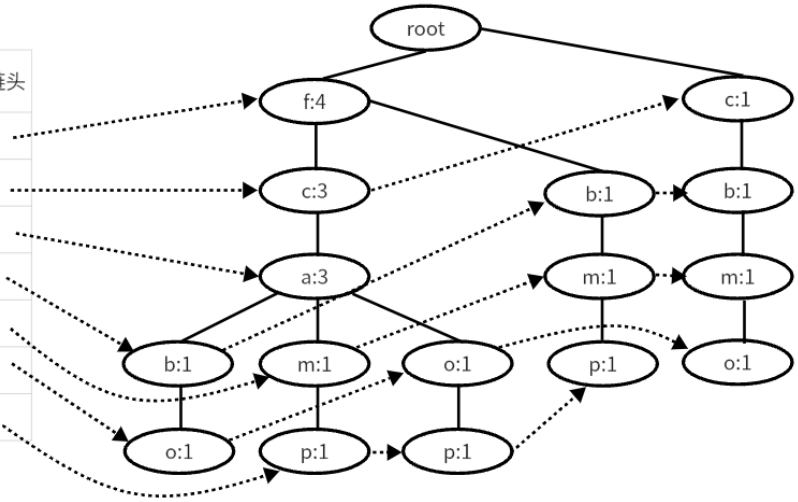


- 按照此方法处理第3~5个事务，可以得到下图：



- 最后按要求连接到项表头和把相同的项目连接起来。使用虚线箭头连接起来相同的元素，得到最终的FP-tree。

项	支持度计数	结点链头
f	4	
c	4	
a	3	
b	3	
m	3	
o	3	
p	3	



(3)从FP-Tree中找出频繁项集。按题目要求先找到以m为后缀的频繁项目集。(找频繁项集参考：FP Tree算法原理总结)

遍历表头项中的m，从FP-tree中找到所有的m结点，向上遍历它的祖先结点，得到以下路径：

ID	Items
1	f:4，c:3，a:3，m:1
2	f:4，b:1，m:1
3	c:1，b:1，m:1

(4)对于每一条路径上的节点，其count都设置为m的count(路径中最末尾的数值)

ID	Items
1	f:1，c:1，a:1，m:1
2	f:1，b:1，m:1
3	c:1，b:1，m:1

(5)把m去掉，得到 **条件模式基** ，此时的后缀模式为：m

ID	Items
1	f:1，c:1，a:1
2	f:1，b:1
3	c:1，b:1

3.3 KNN算法

- 算法思想：k-近邻（kNN, k-Nearest Neighbor）通过计算每个训练数据到待分类元组的距离，取和待分类元组距离最近的k个训练数据，k个数据中哪个类别的训练数据占多数，则待分类元组就属于哪个类别。(其中k值是给定的参数)
- 算法优缺点：
  - 优点：
    - 简单、易于理解、容易实现
    - 通过对K的选择可具备丢噪音数据的健壮性

- 缺点：
  - 算法**复杂度高**（需要比较所有已知实例与要分类的实例）
  - 需要**大量空间储存**所有已知实例
  - 对k值的依赖性
  - 当其**样本分布不平衡**时，比如其中一类样本占主导的时候，新的未知实例容易被归类为这个主导样本。

- 例题：

  K = 5，根据下表训练数据，假如高度参与距离计算，对<范可可，女，1.50>进行分类。

序号	姓名	性别	身高(m)	类别
1	李莉	女	1.50	矮
2	吉米	男	1.92	高
3	马大华	女	1.70	中等
4	王小华	女	1.73	中等
5	刘敏杰	女	1.60	矮
6	包博	男	1.75	中等
7	张烨	女	1.50	矮
8	戴维	男	1.60	矮
9	马天雨	男	2.05	高
10	张晓晓	男	1.90	高
11	刘冰冰	女	1.68	中等
12	陶德德	男	1.78	中等
13	高洁洁	女	1.70	中等
14	张小艺	女	1.68	中等
15	徐甜甜	女	1.65	中等

  解题：

  1. 前 5个记录， N={<李莉，女， 1.50>，<吉米，男，1.92>，<马大华，女，1.70>，<王小华，女，1.73>，<刘敏杰，女，1.60>}。
  2. 第6个记录 =<包博，男，1.75>，相比测试记录<范可可，女，1.50>，需要替换掉*N*中和测试记录差别最大的< 吉米，男，1.92>，得到 N={<李莉，女， 1.50>，<包博，男，1.75>，<马大华，女，1.70>，<王小华，女，1.73>，< 刘敏杰，女，1.60>}。
  3. 第7个记录 =<张烨，女，1.50>，需要替换掉<包博，男，1.75>，得到 N={<李莉，女， 1.50>，<张烨，女，1.50>，<马大华，女，1.70>，<王小华，女，1.73>，< 刘敏杰，女，1.60>}。
  4. 第8个记录 =<戴维，男，1.60>，需要替换掉<王小华，女，1.73>，得到N ={<李莉，女， 1.50>，<张烨，女，1.50>，<马大华，女，1.70>，<戴维，男，1.60>，< 刘敏杰，女，1.60>}。
  5. 对 的第9、10个记录，没变化。
  6. 第11个记录 =<刘冰冰，女，1.68>，需要替换掉<马大华，女，1.70>，得到N ={<李莉，女， 1.50>，<张烨，女，1.50>，<刘冰冰，女，1.68>，<戴维，男，1.60>，< 刘敏杰，女，1.60>}。
  7. 第12～14个记录，没变化。

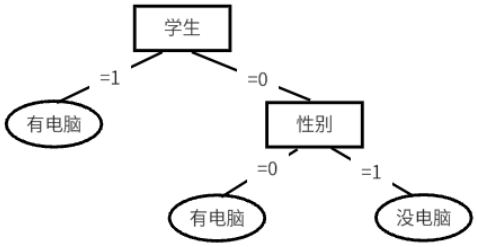
8. 第15个记录 =<徐甜甜，女，1.65>，需要替换掉<刘冰冰，女，1.68>，得到 N={<李莉，女， 1.50>，<张烨，女，1.50>，<徐甜甜，女，1.65>，<戴维，男，1.60>，< 刘敏杰，女，1.60>}。
9. **最后输出** = {<李莉，女， 1.50，**矮**>，<张烨，女，1.50，**矮**>，<徐甜甜，女，1.65，**中等**>，<戴维，男，1.60，**矮**>，< 刘敏杰，女，1.60，**矮**>}。

   在这五项中，四个属于矮个，一个属于中等。最终-最临近算法认为范可可为矮。

3.4 ID3与C4.5算法

3.4.1 ID3算法

- 算法目标：得到决策树。如下图所示：



- 算法思想：在决策树的每一个非叶子结点划分之前，先计算每一个属性所带来的信息增益，选择最大信息增益的属性来划分，因为信息增益越大，区分样本的能力就越强，越具有代表性，很显然这是一种自顶向下的贪心策略。重复这个过程，直至生成一个能完美分类训练样例的决策树。
- 相关概念
  1. **信息熵**：是对随机变量不确定度的度量，熵越大，随机变量的不确定性就越大。一个属性的熵越大，它蕴含的不确定信息就越大，越有利于数据的分类。

     对于某一给定样本分类所需的期望信息其计算公式为：

     $$I(s_1, s_2, \ldots, s_m) = -\sum_{i=1}^{m} p_i * log_2(p_i)$$

     其中：

     - $s_1, s_2, \ldots, s_m$ ：表示某一样本所具有的不同属性出现的次数。例如性别有男女之分，那么s_1可以表示为男生出现的次数，s_2可以表示为女生出现的次数。
     - p_i是样本属性出现的概率，使用$\frac{s_i}{s}$来计算，其中s是样本集合的个数。比如样本集合中总共有6个样本，五位女生，那么$p_2 = \frac{s_2}{s} = \frac{5}{6}$。
  2. **信息增益**：是针对一个一个特征而言的，就是看一个特征，系统有它和没有它时的信息量各是多少，两者的差值就是这个特征给系统带来的信息量，即信息增益。
- 相关公式：

  针对考试题目，列出解题所用公式的一般步骤：

  1. 计算 **决策属性** (最终需要分类的属性) 的信息熵：

     $$I(s_1, s_2, \ldots, s_m) = -\sum_{i=1}^{m} \frac{s_i}{s} * log_2(\frac{s_i}{s})$$

     譬如：最终需要分类的属性为“电脑”，它有两个不同值0和1(表示有无电脑)；1有4个样本，0有两个样本。

     那么有：

     $$I(s_1, s_2) = I(4, 2) = -\frac{4}{6} log_2 \frac{4}{6} - \frac{2}{6} log_2 \frac{2}{6}$$

  2. 对于 **非决策属性** 的信息熵计算，需要计算的是：某一非决策属性值下的决策属性的熵值。

譬如计算非决策属性的熵值：性别为1的有3个有电脑，2个没电脑；性别为0的有1个有电脑，0个没电脑。

那么我们需要先计算：$I(s_{11}, s_{21})$和$I(s_{12}, s_{22})$

$$I(s_{11}, s_{21}) = I(3, 2) = -\frac{3}{5}log_2\frac{3}{5} - \frac{2}{5}log_2\frac{2}{5}$$

$$I(s_{12}, s_{22}) = I(1, 0) = -\frac{1}{1}log_2\frac{1}{1} - \frac{0}{1}log_2\frac{0}{1}$$

其中$I(s_{11}, s_{21})$表示在性别为1时，有无电脑的熵值。$I(s_{12}, s_{22})$表示在性别为0时，有无电脑的熵值。

3. 随后，需要对其 **非决策属性** 计算其期望信息：

$$E(A) = \sum_{j=1}^{v}\frac{s_{1j} + s_{2j} + \ldots + s_{mj}}{s}I(s_{1j}, s_{2j}, \ldots, s_{mj})$$

上式说白了就是对步骤2中的非决策属性求**加权平均和**，即期望。

还是拿上述例子，由于总人数为6，性别为1的有5个，为0的有1个，那么有：

$$E(\text{性别}) = \frac{5}{6}I(s_{11}, s_{21}) + \frac{1}{6}I(s_{12}, s_{22})$$

4. 拿第一步计算的**决策属性信息熵**减去第三步的**非决策属性期望**即可得到该分支上的**信息增益**。即：

$$Gain(A) = I(s_1, s_2, \ldots, s_m) - E(A)$$

5. 重复步骤2、3、4直到完成所有非决策属性的信息增益的计算。
6. 根据**信息增益大小**选择信息增益大的作为决策树当前的非叶子节点。
7. 重复步骤1，2，3，4，5，6直到其决策树的叶子节点为决策属性的值。

注意：**此时的决策属性值为某一结点确定的情况下的，与第一次计算的决策属性不一致。**

- 算法优点和缺点：
  - 算法优点：
    - 算法的理论清晰，方法简单，学习能力较强。
    - ID3算法的假设空间包含所有的决策树，它是关于现有属性的有限离散值函数的一个完整空间。所以ID3算法避免了搜索不完整假设空间的一个主要风险：假设空间可能不包含目标函数。
    - ID3算法在搜索的每一步都使用当前的所有训练样例，大大降低了对个别训练样例错误的敏感性。因此，通过修改终止准则，可以容易地扩展到处理含有噪声的训练数据。
  - 算法缺点：
    - 只对比较小的数据集有效，且对噪声比较敏感，当训练数据集加大时，决策树可能会随之改变。
    - ID3算法在搜索过程中不进行回溯。收敛到局部最优而不是全局最优。
    - ID3算法只能处理离散值的属性。信息增益度量存在一个内在偏置，它偏袒具有较多值的属性。如日期属性。
    - ID3算法增长树的每一个分支的深度，直到恰好能对训练样例完美地分类。当数据中有噪声或训练样例的数量太少时，产生的树会过渡拟合训练样例。
- 例题：

决策属性“买计算机”，根据下表使用ID3算法画出决策树并写出详细过程。

计数	年龄	收入	学生	信誉	归类：买计算机？
64	青	高	否	良	不买
64	青	高	否	优	不买
128	中	高	否	良	买
60	老	中	否	良	买
64	老	低	是	良	买
64	老	低	是	优	不买
64	中	低	是	优	买
128	青	中	否	良	不买
64	青	低	是	良	买
132	老	中	是	良	买
64	青	中	是	优	买
32	中	中	否	优	买
32	中	高	是	良	买
63	老	中	否	优	不买
1	老	中	否	优	买

解题：

1. 首先计算决策属性(买计算机)的信息熵：

$$s_1 = 641 \quad s_2 = 383 \quad s = s_1 + s_2 = 1024$$

$$I(s_1, s_2) = I(641, 383) = -P_1 log_2 P_1 - P_2 log_2 P_2 = -\frac{641}{1024} log_2 \frac{641}{1024} - \frac{383}{1024} log_2 \frac{383}{1024} = 0.9537$$

2. 非决策属性有四个，年龄、收入、学生和信誉，要计算其不同属性的信息增益。先计算年龄的期望信息：

   根据表中数据，年龄为青年的有384个样本，买电脑的有128，不买电脑的有256，所以：

$$I(128, 256) = -\frac{128}{384} log_2 \frac{128}{384} - \frac{256}{384} log_2 \frac{256}{384} = 0.9183$$

   根据表中数据，年龄为中年的有256个样本，买电脑的有256，不买电脑的有0，所以：

$$I(256, 0) = -\frac{256}{256} log_2 \frac{256}{256} - \frac{0}{256} log_2 \frac{0}{256} = 0$$

   根据表中数据，年龄为老年的有252个样本，买电脑的有125，不买电脑的有127，所以：

$$I(125, 127) = -\frac{125}{252} log_2 \frac{125}{252} - \frac{127}{252} log_2 \frac{127}{252} = 0.9157$$

   对此年龄的期望信息(平均加权和)为：

$$E(\text{年龄}) = \frac{384}{1024} * 0.9183 + \frac{256}{1024} * 0 + \frac{384}{1024} * 0.9157 = 0.6877$$

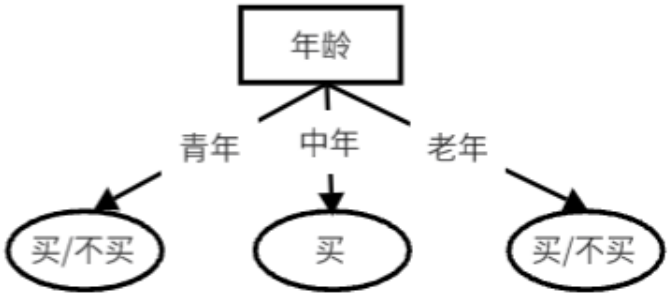
   则，年龄的信息增益为：

$$G(\text{年龄的信息增益}) = I(s_1, s_2) - E(\text{年龄}) = 0.9537 - 0.6877 = 0.2660$$

3. 重复步骤2，计算出收入、学生、信誉的信息增益：

$$G(\text{收入信息增益}) = 0.0176 \quad G(\text{学生信息增益}) = 0.1726 \quad G(\text{信誉信息增益}) = 0.0453$$

4. 由于年龄的信息增益最大，即以年龄来划分最具有价值，区分样本的能力最强。可以根据此画出当前的决策树：



由于青年和老年的所分类的结果存在二义性，故，需要再对**该条件(老年或青年)下**计算信息增益继续划分得到决策树。

5. **在年龄为青年的条件下**，计算决策属性的信息熵：

$$s_1 = 128 \quad s_2 = 256 \quad s = s_1 + s_2 = 384$$

$$I(s_1, s_2) = I(128, 256) = -P_1 log_2 P_1 - P_2 log_2 P_2 = -\frac{128}{384} log_2 \frac{128}{384} - \frac{256}{384} log_2 \frac{256}{384} = 0.9183$$

注意此时决策属性的信息熵与第一步不是同一个值。

6. 计算非决策属性 收入、学生、信誉的信息增益，这里给出计算收入的信息增益具体过程：

根据表格数据，在年龄为青年下，高收入人数有128个，0个买计算机，128个不买计算机，所以：

$$I(0, 128) = -\frac{0}{128} log_2 \frac{0}{128} - \frac{128}{128} log_2 \frac{128}{128} = 0$$

根据表格数据，在年龄为青年下，中收入人数有192个，64个买计算机，128个不买计算机，所以：

$$I(64, 128) = -\frac{64}{192} log_2 \frac{64}{192} - \frac{128}{192} log_2 \frac{128}{192} = 0.9183$$

根据表格数据，在年龄为青年下，低收入人数有64个，64个买计算机，0个不买计算机，所以：

$$I(64, 0) = -\frac{64}{64} log_2 \frac{64}{64} - \frac{0}{64} log_2 \frac{0}{64} = 0$$

对此在年龄为青年下，收入的期望信息(平均加权和)为：

$$E(\text{收入}) = \frac{128}{384} * 0 + \frac{192}{384} * 0.9183 + \frac{64}{384} * 0 = 0.4592$$

则，在年龄为青年下，收入的信息增益为：

$$G(\text{收入信息增益}) = I(s_1, s_2) - E(\text{收入}) = 0.9183 - 0.4592 = 0.4591$$

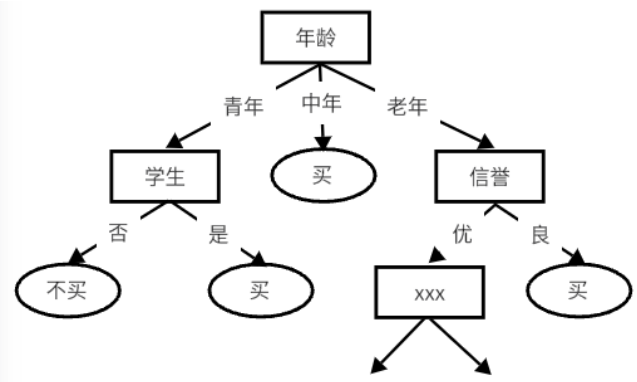
7. 重复步骤6，计算出在年龄为青年下，学生、信誉的信息增益：

略

8. 还要计算在年龄为老年人的条件下，其收入、学生、信誉的信息增益：

略

9. 略



3.4.2 C4.5算法

- 算法思想：
  1. 选择当前节点的最优划分属性，即使得**信息增益率**最大的属性，如果不存在则该节点为叶子节点。
  2. 对选择的最优属性的每个取值，分别构建一个子节点，并将样本点分配到这些子节点中。
  3. 对每个子节点，递归地执行步骤1-2，直至满足终止条件，即到达叶子节点或无法继续划分。
  4. 构建好决策树，用它进行测试数据的分类预测。
- 相关概念
  1. **信息增益**$Gain(A)$：是针对一个一个特征而言的，就是看一个特征，系统有它和没有它时的信息量各是多少，两者的差值就是这个特征给系统带来的信息量，即信息增益。
  2. **分裂信息度量**$SplitI(A)$：衡量属性分裂数据的广度和均匀，分裂信息度量被定义为：

$$SplitI(A) = -\sum_{j=1}^{v} p_j * log_2(p_j)$$

     注意这里v表示的是属性值的划分的类别数量。
  3. **信息增益率**$GainRatio(A)$：避免出现因为信息增益偏向取值比较多的属性的问题。

$$GainRatio(A) = \frac{Gain(A)}{SplitI(A)}$$

- 例题：

3.4.3 两者异同

C4.5算法同ID3算法类似，只不过在计算过程中采用**信息增益率**来选择特征而不是**信息增益**。通过引入**信息增益比**，一定程度上对取值比较多的特征进行惩罚，避免出现过拟合的特性，提升决策树的泛化能力。

C4.5算法是从ID3算法演变而来，除了拥有ID3算法的功能外，C4.5算法引入了新的方法和增加了新的功能。例如：

- 用**信息增益比例**的概念。
- 合并具有**连续属性**的值。
- 可以**处理缺少属性值的训练样本**。
- 通过使用不同的**修剪**技术以避免树的过度拟合。
- k交叉验证。
- 规则的产生方式等。

3.5 朴素贝叶斯算法

- 算法思想：

  计算后验概率，谁的后验概率大就属于哪一个类，下面给出后验概率计算公式：

$$P(\text{所属类别}|\text{某种特征}) = \frac{P(\text{某种特征}|\text{所属类别})P(\text{所属类别})}{P(\text{某种特征})}$$

  使用该算法的前提条件：**假定属性值相互条件独立，即在属性间不存在依赖关系。**

  (考试只考离散的情况)
- 相关概念：

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}$$

- X：样本数据，是样本特征。
- H：是某种假定，X属于C类。
- P(X)：样本表征出现的概率，对所有类而言为常数。
- P(H)：先验概率，根据训练集得到，训练集中某种假定出现的概率。
- P(X|H)：假定成立的条件下，X表征出现的概率。
- P(H|X)：后验概率，在某一特征出现的情况下，属于该类的概率。

对于所有类而言，P(X)一致，故实际上只需要比较$P(X|H)P(H)$的大小即可。

假定属性值相互条件独立，即在属性间不存在依赖关系时有：

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i)$$

其中，概率$P(X_1|C_i), P(X_2|C_i), \ldots, P(X_n|C_i)$可以由训练样本估计

- 例题：

  给定训练数据表，将未知样本X=(age="$\leqslant 30$",income="medium",student="yes",credit_rating="fair")分类是否购买电脑。

RID	age	income	student	credit_rating	buys_computer
1	$\leqslant 30$	High	No	Fair	No
2	$\leqslant 30$	High	No	Excellent	No
3	31~40	High	No	Fair	Yes
4	> 40	Medium	No	Fair	Yes
5	> 40	Low	Yes	Fair	Yes
6	> 40	Low	Yes	Excellent	No
7	31~40	Low	Yes	Excellent	Yes
8	$\leqslant 30$	Medium	No	Fair	No
9	$\leqslant 30$	Low	Yes	Fair	Yes
10	> 40	Medium	Yes	Fair	Yes
11	$\leqslant 30$	Medium	Yes	Excellent	Yes
12	31~40	Medium	No	Excellent	Yes
13	31~40	High	Yes	Fair	Yes
14	> 40	Medium	No	Excellent	No

解题：

1. 先计算先验概率P(H)：

$$P(buys\_computer = No) = \frac{5}{14}$$
$$P(buys\_computer = Yes) = \frac{9}{14}$$

2. 计算概率$P(X_1|C_i), P(X_2|C_i), \ldots, P(X_n|C_i)$

   所给未知样本为：X=(age="$\leqslant 30$",income="medium",student="yes",credit_rating="fair")

   在buy_computer=No时：

$$P(age \leqslant 30 \mid buy\_computer = No) = \frac{3}{5}$$

$$P(income = medium \mid buy\_computer = No) = \frac{2}{5}$$

$$P(student = yes \mid buy\_computer = No) = \frac{1}{5}$$

$$P(credit_rating = fair \mid buy\_computer = No) = \frac{2}{5}$$

在buy_computer=Yes时：

$$P(age \leqslant 30 \mid buy\_computer = Yes) = \frac{2}{9}$$

$$P(income = medium \mid buy\_computer = Yes) = \frac{4}{9}$$

$$P(student = yes \mid buy\_computer = Yes) = \frac{6}{9}$$

$$P(credit_rating = fair \mid buy\_computer = Yes) = \frac{6}{9}$$

3. **假定属性值相互条件独立，即属性间不存在依赖关系时**：

   计算P(X|H)有：

$$P(X \mid buy\_computer = No) = \frac{3}{5} * \frac{2}{5} * \frac{1}{5} * \frac{2}{5} = 0.019$$

$$P(X \mid buy\_computer = Yes) = \frac{2}{9} * \frac{4}{9} * \frac{6}{9} * \frac{6}{9} = 0.044$$

4. 最后计算P(X|H)*P(H)，即第一步的值和第三步的值相乘：

$$P(X \mid buy\_computer = No)P(buys\_computer = No) = \frac{5}{14} * 0.019 = 0.007$$

$$P(X \mid buy\_computer = Yes)P(buys\_computer = Yes) = \frac{9}{14} * 0.044 = 0.028$$

因此，对于样本X，朴素贝叶斯分类预测 buy_computer = "Yes"。

3.6 K均值算法

- 算法思想：算法**随机的选择k个对象**(k值给定)，每个对象**初始的代表了一个簇的平均值或中心。**对剩余的每个对象根据其与各个簇中心的**距离**，将它赋值给最近的簇。然后**重新计算**每个簇的平均值。这个过程不断重复，**直到准则函数收敛。**

  K-平均算法的准则函数定义为：

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} \mid x - \overline{x}_i \mid^2$$

- 相关概念

  1. **聚类分析**：是指根据给定一组对象的描述信息，发现由具有共同特性的对象构成**簇**的过程。
  2. **评判聚类好坏**：类间距离大，类内距离小
  3. **明可夫斯基距离(Minkowski)**：

$$d(x, y) = [\sum_{i=1}^{n} |x_i - y_i|^r]^{\frac{1}{r}}$$

   当$r = 1$时，明可夫斯基距离演变为**绝对值距离**：

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

当$r=2$时，明可夫斯基距离演变为**欧式距离**：

$$d(x,y)=[\sum_{i=1}^{n}|x_i-y_i|^2]^{\frac{1}{2}}$$

4. **类间距离**：
    1. 最短距离法：定义两个类中最靠近的两个元素间的距离为类间距离。

$$D_S(C_a,C_b)=min\{d(x,y)\mid x\in C_a,y\in C_b\}$$

    2. 最长距离法：定义两个类中最远的两个元素间的距离为类间距离。

$$D_L(C_a,C_b)=max\{d(x,y)\mid x\in C_a,y\in C_b\}$$

    3. 中心法：定义两个类的两个中心间距离为类间距离。
        - 类中心定义：

$$\overline{x}_i=\frac{1}{n_i}\sum_{x\in C_i}x$$

        - 基于类中心，进一步定义两个类C_a和C_b的类间距离为：

$$D_C(C_a,C_b)=d(r_a,r_b)$$

    4. 类平均法：将两个类中任意两个元素间的距离的平均值定义为类间距离，即

$$D_C(C_a,C_b)=\frac{1}{mh}\sum_{x\in C_a}\sum_{y\in C_b}d(x,y)$$

    其中，m 和 h 是两个类C_a和C_b的元素个数。

- 算法优缺点：
    - 优点：
        1. 是解决聚类问题的一种经典算法，简单、快速。
        2. 对处理大数据集，该算法是相对可伸缩和高效率的。
        3. 当结果簇是密集的，它的效果较好。
    - 缺点：
        1. 在簇的平均值被定义的情况下才能使用，可能不适用于某些应用。
        2. 必须事先给出*k*（要生成的簇的数目），而且对初值敏感，对于不同的初始值，可能会导致不同结果。
        3. 不适合于发现非凸面形状的簇或者大小差别很大的簇。而且，它对于“躁声”和孤立点数据是敏感的。
- 例题：

    设$n=8$，$k=2$，事先随机选取序号1和序号3作为初始点，给定样本事务数据库，请对其使用k-平均算法分类。

序号	属性1	属性2
1	1	1
2	2	1
3	1	2
4	2	2
5	4	3

序号	属性1	属性2
6	5	3
7	4	4
8	5	4

解题：

1. 第一次迭代：将序号1和序号3作为初始点，由准则函数可得：

	2:(2,1)	4:(2,2)	5:(4,3)	6:(5,3)	7:(4,4)	8:(5,4)
1:(1,1)	1	2	13	20	18	25
3:(1,2)	2	1	10	17	13	20

   根据上表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(对应属性值累加除以簇目数)**，**得到平均值点**：

   对于簇{1，2}的平均值点为(1.5，1)；

   对于簇{3，4，5，6，7，8}的平均值点位(3.5，3)。

2. 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)和(3.5，1)最近的原则重新分配。由准则函数可得：

	1:(1,1)	2:(2,1)	3:(1,2)	4:(2,2)	5:(4,3)	6:(5,3)	7:(4,4)	8:(5,4)
(1.5,1)	0.25	0.25	1.25	1.25	10.25	16.25	15.25	21.25
(3.5,3)	10.25	6.25	7.25	3.25	0.25	2.25	1.25	3.25

   根据上表找到离两点最近的对象，并产生两个簇{1，2，3，4}和{5，6，7，8}

   对于簇{1，2，3，4}的平均值点为(1.5，1.5)；

   对于簇{5，6，7，8}的平均值点为(4.5，3.5)。

3. 第三次迭代：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(1.5，1.5)和(4.5，3.5)最近的原则重新分配。由准则函数可得：

	1: (1,1)	2: (2,1)	3: (1,2)	4: (2,2)	5: (4,3)	6: (5,3)	7: (4,4)	8: (5,4)
(1.5,1.5)	0.5	0.5	0.5	0.5	8.5	14.5	12.5	18.5
(4.5,3.5)	18.5	12.5	14.75	8.5	0.5	0.5	0.5	0.5

   根据上表找到离两点最近的对象，并产生两个簇{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。

4. 给出整个过程中平均值计算和簇生成的过程和结果。

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)

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)
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)

3.7 AGNES与DIANA算法

3.7.1 AGNES算法

- 算法思想：AGNES (AGglomerative NESting)算法最初将**每个对象作为一个簇**，然后这些簇根据**某些准则**被一步步地**合并。**两个簇间的相似度由这两个不同簇中**距离最近的数据点对的相似度来确定。**聚类的**合并过程反复进行**直到所有的对象最终**满足簇数目(作为参数给定)。**
- 算法评价：
  - AGNES算法比较简单，但经常会遇到合并点选择的困难。假如一旦一组对象被合并，下一步的处理将在新生成的簇上进行。已做处理不能撤消，聚类之间也不能交换对象。如果在某一步没有很好的选择合并的决定，可能会导致低质量的聚类结果。
  - 这种聚类方法不具有很好的可伸缩性，因为合并的决定需要检查和估算大量的对象或簇。
  - 假定在开始的时候有n个簇，在结束的时候有1个簇，因此在主循环中有n次迭代，在第i次迭代中，我们必须在n-i+1个簇中找到最靠近的两个聚类。另外算法必须计算所有对象两两之间的距离，因此这个算法的复杂度为 $O(n2)$，该算法对于n很大的情况是不适用的。
- 例题：

  给定样本事务数据库，终止条件为**两**个簇，使用AGNES算法进行层次聚类分析。

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

  解题：

  1. 将每个事务划分为一个簇，初始簇为{1}，{2}，{3}，{4}，{5}，{6}，{7}，{8}。并计算每个簇之间的距离，这里采用最短距离法计算(若题目有要求，按照题目要求计算)：

簇	{1}	{2}	{3}	{4}	{5}	{6}	{7}	{8}
{1}	0	1	1	2	5	6	6	7
{2}		0	2	1	4	5	5	6
{3}			0	1	3	4	4	5

簇	{1}	{2}	{3}	{4}	{5}	{6}	{7}	{8}
{4}				0	3	4	4	5
{5}					0	1	1	2
{6}						0	2	1
{7}							0	1
{8}								0

找到最近的两个簇{1} 和{2} 合并得到新簇：{1,2}，{3}，{4}，{5}，{6}，{7}，{8}。

2. 对上一次合并后的簇计算簇间距离，找到距离最近的两个簇进行合并。

簇	{1,2}	{3}	{4}	{5}	{6}	{7}	{8}
{1,2}	0	1	1	4	5	5	6
{3}		0	1	3	4	4	5
{4}			0	3	4	4	5
{5}				0	1	1	2
{6}					0	2	1
{7}						0	1
{8}							0

由上表知，簇{3}和簇{4}合并，得到新簇{1,2}，{3,4}，{5}，{6}，{7}，{8}。(**注**：至于为啥不先把{1,2}和{3}合并成{1,2,3}，事实上，自己写的算法不同，执行合并顺序不同(这也是AGNES在处理簇合并上的缺点)，但是在此题上无论怎么个合并顺序，其结果形成的最终簇是一致的。)

3. 对上一次合并后的簇计算簇间距离，找到距离最近的两个簇进行合并。

簇	{1,2}	{3,4}	{5}	{6}	{7}	{8}
{1,2}	0	1	4	5	5	6
{3,4}		0	3	4	4	5
{5}			0	1	1	2
{6}				0	2	1
{7}					0	1
{8}						0

由上表知，簇{5}和簇{6}合并，得到新簇{1,2}，{3,4}，{5,6}，{7}，{8}。

4. 对上一次合并后的簇计算簇间距离，找到距离最近的两个簇进行合并。

簇	{1,2}	{3,4}	{5,6}	{7}	{8}
{1,2}	0	1	4	5	6
{3,4}		0	3	4	5
{5,6}			0	1	1

簇	{1,2}	{3,4}	{5,6}	{7}	{8}
{7}				0	1
{8}					0

由上表知，簇{7}和簇{8}合并，得到新簇{1,2}，{3,4}，{5,6}，{7,8}。

5. 对上一次合并后的簇计算簇间距离，找到距离最近的两个簇进行合并。

簇	{1,2}	{3,4}	{5,6}	{7,8}
{1,2}	0	1	4	5
{3,4}		0	3	4
{5,6}			0	1
{7,8}				0

合并簇{1,2}和簇{3,4}，新簇：{1,2,3,4}，{5,6}，{7,8}。

6. 对上一次合并后的簇计算簇间距离，找到距离最近的两个簇进行合并。

簇	{1,2,3,4}	{5,6}	{7,8}
{1,2,3,4}	0	3	4
{5,6}		0	1
{7,8}			0

合并簇{5,6}和簇{7,8}，新簇：{1,2,3,4}，{5,6,7,8}。由于合并后的簇数目已经达到了用户输入的终止条件，程序结束。

7. 给出上述步骤1-6的执行过程：

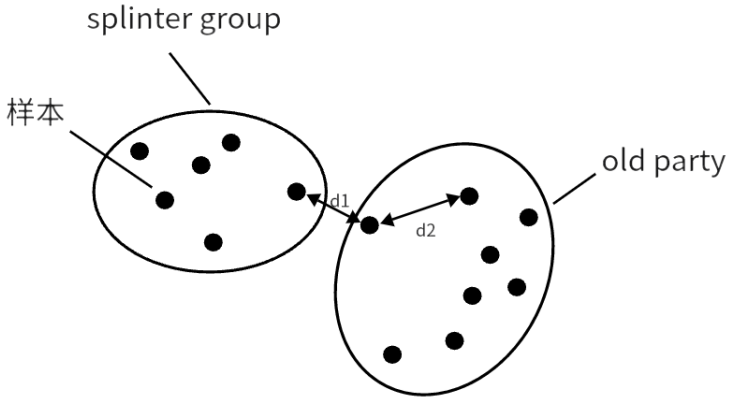
步骤	最近的簇距离	最近的两个簇	合并后的新簇
1	1	{1}，{2}	{1,2}，{3}，{4}，{5}，{6}，{7}，{8}
2	1	{3}，{4}	{1,2}，{3,4}，{5}，{6}，{7}，{8}
3	1	{5}，{6}	{1,2}，{3,4}，{5,6}，{7}，{8}
4	1	{7}，{8}	{1,2}，{3,4}，{5,6}，{7,8}
5	1	{1,2}，{3,4}	{1,2,3,4}，{5,6}，{7,8}
6	1	{5,6}，{7,8}	{1,2,3,4}，{5,6,7,8}

3.7.2 DIANA算法

- 算法思想：所有的对象**初始都放在一个簇**中。根据**一些原则**(如簇中最邻近对象的最大欧式距离)，将该簇分裂。簇的**分裂过程反复进行**，直到最终每个新的簇只包含一个对象。在聚类中，用户能定义希望得到的簇数作为一个结束条件。同时使用下面两种测度方法：
  1. **簇的直径**：在一个簇的**任意**两个数据点的**欧式距离**中的**最大值**。优先选择直径大的簇进行分裂。
  2. **平均相异度(在某同一个簇中，该簇下某点对该簇下其他点的平均距离)**：

$$d_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{x \in C_i} \sum_{y \in C_j} | x - y |$$

分裂原则：在 **old party** 里找到 **splinter group** 中点的最近距离不大于到 **old party** 中点的最近距离的点，并将该点加入 **splinter group** 。



上图所示：

- d1：表示 old party 中的某点到 splinter group 的距离。
- d2：表示 old party 中的该点到本簇中其他点的最近距离。
- 若有 $d_1 \leqslant d_2$ ，则将该点从 old party 中移动到 splinter group 中。

- 相关概念：
  1. **splinter group**：簇A中某些样本的**平均相异度**过大而被簇A剔除而形成的一个新簇称为**splinter group。**
  2. **old party**：簇A中剔除某些**平均相异度**大的样本剩下的称为**old party。**
- 具体步骤：
  1. 在所有簇中挑出具有**最大直径**的簇。
  2. 从该簇中挑选具有**最大平均相异度**的点放在 splinter group 中，其他放在 old party 中
  3. 若某点**满足分裂规则**，则将该点从old party中移动到splinter group中，直到移不动为止。
  4. 重复步骤1，2，3，直到分裂的簇**满足给定的簇数目**为止。
- 例题：

  在下表中给定的样本上运行DIANA算法，假定算法的终止条件为3个簇，初始簇为{1,2,3,4,5,6,7,8}。

序号	属性1	属性2
1	2	10
2	2	5
3	8	4
4	5	8
5	7	5
6	6	4
7	1	2
8	4	9

解题：

为了解题方便，先将各个点间的绝对值距离计算出来如下表：

序号	1	2	3	4	5	6	7	8
1	0	5	12	5	10	10	9	3
2	5	0	7	6	5	5	4	6
3	12	7	0	7	2	2	9	9
4	5	6	7	0	5	5	10	2
5	10	5	2	5	0	2	9	7
6	10	5	2	5	2	0	7	7
7	9	4	9	10	9	7	0	10
8	3	6	9	2	7	7	10	0

1. 选择具有最大直径的簇{1,2,3,4,5,6,7,8}进行分裂，计算该簇中所有点的平均相异度，题目未要求，这里使用绝对值距离求解。

   1的平均相异度：(5+12+5+10+10+9+3)/7=7.714

   2的平均相异度：(5+7+6+5+5+4+6)/7=5.429

   3的平均相异度：(12+7+7+2+2+9+9)/7=6.857

   4的平均相异度：(5+6+7+5+5+10+2)/7=5.714

   5的平均相异度：(10+5+2+5+2+9+7)/7=5.714

   6的平均相异度：(10+5+2+5+2+7+7)/7=5.429

   7的平均相异度：(9+4+9+10+9+7+10)/7=8.286

   8的平均相异度：(3+6+9+2+7+7+10)/7=6.286

   计算发现，序号7的平均相异度最大，故 splinter group = {7}，old party = {1,2,3,4,5,6,8}。

2. 注意到$d(7,2)=4$且点2在old party中距离其他点的距离$d_{min}=5$。由于$d\leqslant d_{min}$所以将点2移动到splinter group中，即splinter group = {7,2}，old party = {1,3,4,5,6,8}。

3. 注意到$d(2,1)=5$，点1在old party中距离其他点的距离$d_{min}=3$。不满足，继续搜索。

4. 注意到$d(2,5)=5$，点5在old party中距离其他点的距离$d_{min}=3$。不满足，继续搜索。

5. 注意到$d(2,6)=5$，点5在old party中距离其他点的距离$d_{min}=2$。不满足，继续搜索。

6. ……不断寻找，发现没有新的old party中的点分配给splinter group，此时分裂的簇数为2，未达到终止条件。此时正式分为两个簇：{2,7}和{1,3,4,5,6,8}。

7. 查表知：簇{2,7}的直径为4，簇{1,3,4,5,6,8}的直径为12。接下来对簇{1,3,4,5,6,8}继续分裂。

8. 计算该簇{1,3,4,5,6,8}中所有点的平均相异度。

   1的平均相异度：(12+5+10+10+3)/5=8

   3的平均相异度：(12+7+2+2+9)/5=6.4

   4的平均相异度：(5+7+5+5+2)/5=4.8

   5的平均相异度：(10+2+5+2+7)/5=3.25

   6的平均相异度：(10+2+5+2+7)/5=3.25

   8的平均相异度：(3+9+2+7+7)/5=3.5

   计算发现，点1的平均相异度在本簇中最大，因此簇{1,3,4,5,6,8}分为 splinter group = {1} 和 old party = {3,4,5,6,8}。

9. 注意到$d(1,8)=3$，点8在old party中距离其他点的距离$d_{min}=2$。不满足，继续搜索。

10. ……不断寻找，发现没有新的old party中的点分配给splinter group，此时分裂的簇数为3，达到终止条件。最终分为三个簇：{2,7}和{3,4,5,6,8}和{1}。

3.7.3 两者异同

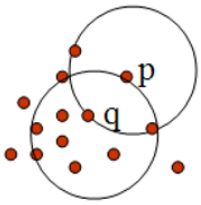
AGNES算法和DIANA算法都属于**层次聚类方法。**层次聚类方法对给定的数据集进行层次的分解，直到某种条件满足为止。**两者都需要定义簇间的距离度量或现实性度量，都能形成一个树状结构，都无需事先指定聚类数目。**具体又可分为：

- **凝聚的层次聚类**：一种自底向上的策略，首先将每个对象作为一个簇，然后合并这些原子簇为越来越大的簇，直到某个终结条件被满足。层次凝聚的代表是**AGNES算法。**
- **分裂的层次聚类**：采用自顶向下的策略，它首先将所有对象置于一个簇中，然后逐渐细分为越来越小的簇，直到达到了某个终结条件。层次分裂的代表是**DIANA算法。**

3.8 DBSCAN算法

- 算法思想：DBSCAN 通过检查数据集中**每个对象**的***ε*-邻域**(半径给定参数)来寻找聚类。如果一个点*p*的*ε*-邻域包含多于**MinPts**(给定参数)个对象，则创建一个*p*作为**核心对象**的新簇。然后，DBSCAN反复地寻找从这些**核心对象直接密度可达**的对象，这个过程可能涉及一些**密度可达簇的合并。**当没有新的点可以被添加到任何簇时，该过程结束。
- 相关概念：
  1. ***ε*-邻域**：给定对象在半径***ε***内的区域。
  2. **核心对象**：如果一个对象的*ε*-领域**至少包含**最小数目**MinPts**(给定参数)个对象，则称该对象为核心对象。
  3. **直接密度可达**：给定一个对象集合*D*，如果***p*是在*q*的*ε*-邻域内**，而***q*是一个核心对象**，我们说对象*p*从对象*q*出发是直接密度可达的。

     如下图，*ε*=1cm，MinPts=5，*q*是一个核心对象，对象*p*从对象*q*出发是直接密度可达的。

  4. **密度可达**(间接密度可达)：如果存在一个对象链p_1，p_2，…，p_n。$p_1=q$，$p_n=p$，对$p_1 \in D$，$(1 \leqslant i \leqslant n)$，且有$p_{i+1}$是从$p_i$关于*ε*和MitPts**直接密度可达(注意：这里隐喻着p_i是核心对象**)的，则对象*p*是从对象*q*关于*ε*和MinPts密度可达的。
- 例题：

  对所给的数据进行DBSCAN算法，给定$\varepsilon=1$，$MinPts=4$。

序号	属性1	属性2
1	1	0
2	4	0
3	0	1
4	1	1
5	2	1
6	3	1
7	4	1

序号	属性1	属性2
8	5	1
9	0	2
10	1	2
11	4	2
12	1	3

解题：

1. 对于每一个点，寻找在 $\varepsilon = 1$ 领域内的点的个数，并结合给定的$MinPts = 4$ 列出下表：

选择的点	在该点的 ε 领域内	在 ε 中个数	是否为核心对象
1	{1,4}	2	否
2	{2,7}	2	否
3	{3,4,9}	3	否
4	{1,3,4,5,10}	5	是
5	{4,5,6}	3	否
6	{5,6,7}	3	否
7	{2,6,7,8,11}	5	是
8	{7,8}	2	否
9	{3,9,10}	3	否
10	{4,9,10,12}	4	是
11	{7,11}	2	否
12	{10,12}	2	否

2. 在数据库中选择一点1，由于在以它为圆心的，以1为半径的圆内包含2个点(小于4)，因此它不是核心对象，选择下一点。
3. 点2、点3同上。
4. 在数据库中选择一点4，由于在以它为圆心的，以1为半径的圆内包含5个点，因此它是核心对象，其直接密度可达的有{1,3,5,10}，检查**核心对象**点10的 ε 领域：{4,9,12}，这些点{4,9,12}对于点4是间接密度可达，故聚类的新簇C_1 = {1,3,4,5,9,10,12}。选择下一个点。
5. 在数据库中选择一点5，已经在簇1中，选择下一个点。
6. 在数据库中选择一点6，由于在以它为圆心的，以1为半径的圆内包含3个点(小于4)，因此它不是核心对象，选择下一点。
7. 在数据库中选择一点7，由于在以它为圆心的，以1为半径的圆内包含5个点，因此它是核心对象，其直接密度可达的有{2,6,8,11}，其直接密度可达的点中没有核心对象，故不存在间接密度可达，故聚类的新簇C_2 = {2,7,8,11}。选择下一个点。
8. 在数据库中选择一点8，已经在簇2中，选择下一个点。
9. 在数据库中选择一点9，已经在簇1中，选择下一个点。
10. 在数据库中选择一点10，已经在簇1中，选择下一个点。
11. 在数据库中选择一点11，已经在簇2中，选择下一个点。

12. 在数据库中选择一点12，已经在簇1中。完成所有点的扫描，程序终止。
13. 给出步骤1~12DBSCAN算法执行过程示意图

步骤	选择的点	在 ε 中个数	通过计算可达点而找到的新簇
1	1	2	无
2	2	2	无
3	3	3	无
4	4	5	簇C_1:{1,3,4,5,9,10,12}
5	5	3	已在一个簇C_1中
6	6	3	已在一个簇C_1中
7	7	4	簇C_2:{2,6,7,8,11}
8	8	2	已在一个簇C_2中
9	9	3	已在一个簇C_1中
10	10	4	已在一个簇C_1中
11	11	2	已在一个簇C_2中
12	12	2	已在一个簇C_1中

3.9 PageRank算法

- 算法思想：PageRank算法是一种网页排名算法，其基本准则有两条：
  - 链接数量。一个网页被越多的其他网页链接，说明这个网页很重要。
  - 链接质量。一个网页被一个越高权值的网页链接，也能表明这个网页越重要。
- 相关概念：
  1. 设u为一个Web页，F_u为所有u指向的页面(扇出)的集合，B_u为所有指向u的页面(扇入)的集合。设$c(<1)$为一个调整参数，那么u页面的PageRank($R(u)$)被定义为：

$$R(u)=c\sum_{v\in B_u}\frac{R(v)}{|F_v|}$$

  2. 邻接矩阵***A*** **= (**$a_{i,j}$**)**中的元素$a_{i,j}(\in[0,1])$表示从页面 j 指向页面 i 的概率。页面的等级值为1，该页面上共有 n 个超链接，其分配给每个超链接页面的等级值即 $a_{i,j}=\frac{1}{n}$ 。
  3. 把邻接矩阵 ***A*** 转化成所谓的 **转移概率矩阵M** 来实现 PageRank算法：

$$M=(1-d)\times Q+d\times A$$

     其中：
     - **d**： 经验参数，默认取值为0.85 。
     - **Q**：常量矩阵，默认$Q=(q_{i,j})$, $q_{i,j}=\frac{1}{n}$ 这里n是页面数量。
     - 对此一般默认M的计算公式写为：

$$M=0.15\times[\frac{1}{n}]+0.85\times A$$

  4. 得到 **转移概率矩阵M** 后，使用下面的公式，不断迭代计算等级值向量 R_{i+1} 。

$$R_{i+1} = M \times R_i$$

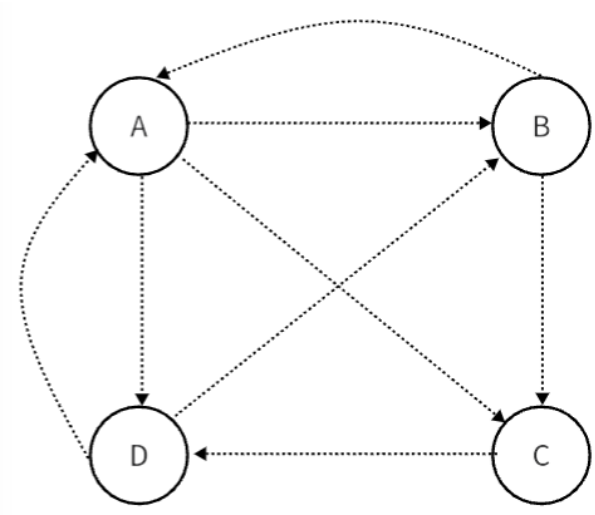
其中初始值 R_0为常量1矩阵

5. 每次计算 R_{i+1} 后，计算 ε_{i+1} ，与给定参数 ε 比较。算法终止条件是$\varepsilon_{i+1} \leqslant \varepsilon$ 。

$$\varepsilon_{i+1} = | R_{i+1} - R_i |$$

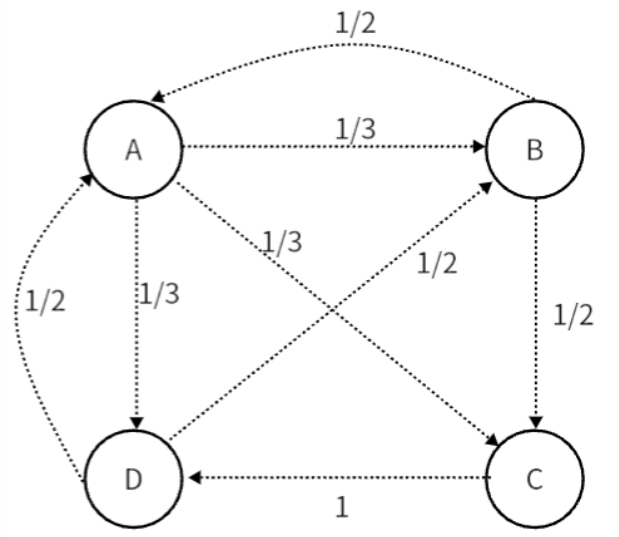
- 例题：

  假定存在A、B、C、D四个页面，各页面之间的链接关系如下图所示。请计算这四个页面的PageRank值。$\varepsilon = 0.1$



解题：

1. 标记其转移概率，如下图：



由此得到邻接矩阵A：

$$A = \begin{bmatrix} 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} & 0 & 0 \\ \frac{1}{3} & 0 & 1 & 0 \end{bmatrix}$$

2. 计算概率转移矩阵M：

$$M = 0.15 \times \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix} + 0.85 \times \begin{bmatrix} 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} & 0 & 0 \\ \frac{1}{3} & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0.04 & 0.46 & 0.04 & 0.46 \\ 0.32 & 0.04 & 0.04 & 0.46 \\ 0.32 & 0.46 & 0.04 & 0.04 \\ 0.32 & 0.04 & 0.89 & 0.04 \end{bmatrix}$$

3. 接下来开始迭代计算：

$$R_1 = M \times R_0 = \begin{bmatrix} 0.04 & 0.46 & 0.04 & 0.46 \\ 0.32 & 0.04 & 0.04 & 0.46 \\ 0.32 & 0.46 & 0.04 & 0.04 \\ 0.32 & 0.04 & 0.89 & 0.04 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1.00 \\ 0.86 \\ 0.86 \\ 1.29 \end{bmatrix}$$

计算 $\varepsilon_1 = \mid R_1 - R_0 \mid = 0.57 > \varepsilon$，继续迭代。

4. 计算R_2:

$$R_2 = M \times R_1 = \begin{bmatrix} 0.04 & 0.46 & 0.04 & 0.46 \\ 0.32 & 0.04 & 0.04 & 0.46 \\ 0.32 & 0.46 & 0.04 & 0.04 \\ 0.32 & 0.04 & 0.89 & 0.04 \end{bmatrix} \times \begin{bmatrix} 1.00 \\ 0.86 \\ 0.86 \\ 1.29 \end{bmatrix} = \begin{bmatrix} 1.06 \\ 0.98 \\ 0.80 \\ 1.17 \end{bmatrix}$$

计算 $\varepsilon_2 = \mid R_2 - R_1 \mid = 0.36 > \varepsilon$，继续迭代。

5. 计算R_3:

$$R_3 = M \times R_2 = \begin{bmatrix} 0.04 & 0.46 & 0.04 & 0.46 \\ 0.32 & 0.04 & 0.04 & 0.46 \\ 0.32 & 0.46 & 0.04 & 0.04 \\ 0.32 & 0.04 & 0.89 & 0.04 \end{bmatrix} \times \begin{bmatrix} 1.06 \\ 0.98 \\ 0.80 \\ 1.17 \end{bmatrix} = \begin{bmatrix} 1.06 \\ 0.95 \\ 0.87 \\ 1.14 \end{bmatrix}$$

计算 $\varepsilon_3 = \mid R_3 - R_2 \mid = 0.13 > \varepsilon$，继续迭代。

6. 计算R_4:

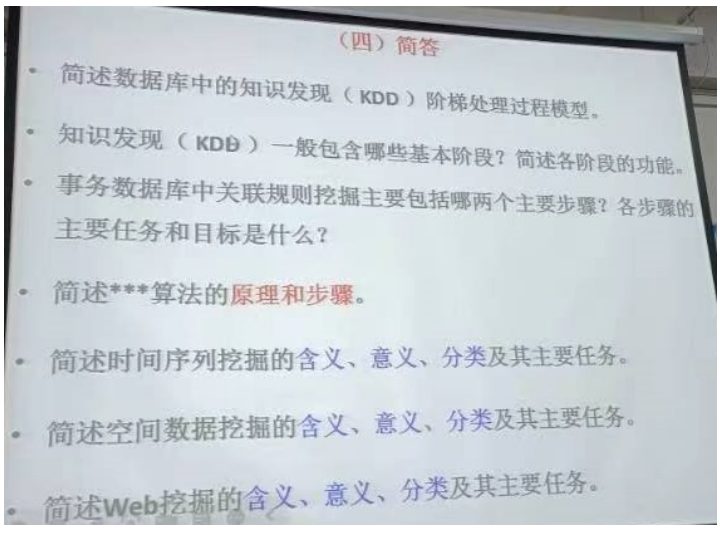
$$R_3 = M \times R_2 = \begin{bmatrix} 0.04 & 0.46 & 0.04 & 0.46 \\ 0.32 & 0.04 & 0.04 & 0.46 \\ 0.32 & 0.46 & 0.04 & 0.04 \\ 0.32 & 0.04 & 0.89 & 0.04 \end{bmatrix} \times \begin{bmatrix} 1.06 \\ 0.95 \\ 0.87 \\ 1.14 \end{bmatrix} = \begin{bmatrix} 1.04 \\ 0.94 \\ 0.86 \\ 1.19 \end{bmatrix}$$

计算 $\varepsilon_3 = \mid R_3 - R_2 \mid = 0.07 \leq \varepsilon$，结束迭代。最后A、B、C、D页面等级值为：1.04、0.94、0.86、1.19.

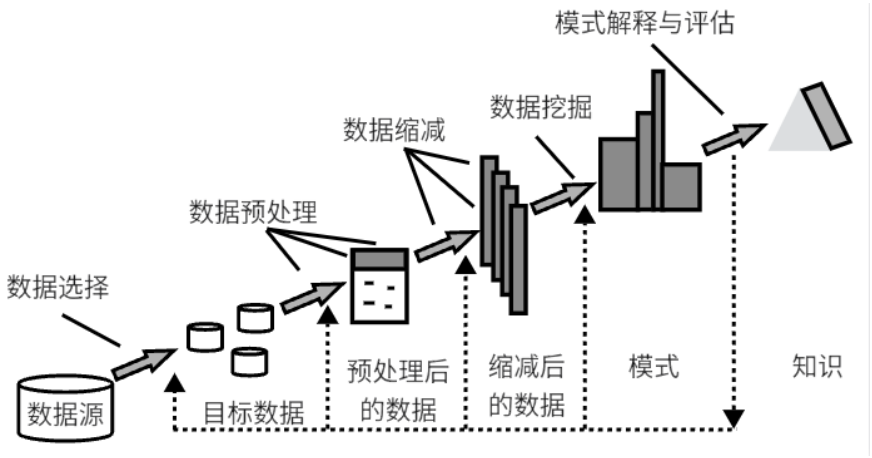
3.10 HITS算法

- HITS（Hyperlink-Induced Topic Search）是遵照寻找权威页面和中心页面的典型方法，基于一组给定的关键字，可以找到相关的页面。
- 相关概念：
  1. **权威页面**是指包含需求信息的最佳资源页面。
  2. **中心页面**是一个包含权威页面链接的页面。

四、简答题



4.1 知识发现(KDD)阶梯处理过程模型



具体步骤解释如下：

- 数据准备，了解KDD相关领域的有关情况，熟悉有关的背景知识，并弄清楚用户的要求，**确定挖掘的总体目标和方法。**了解相关的源数据结构并加以分析，确定数据选择的原则。
- 数据选择：根据用户的要求从数据库中**提取与KDD目标相关的数据。**在此过程中,KDD系统将从备选的源数据中进行知识提取。这种数据选择工作可以借助于数据库操作语言或专门的工具来进行。
- 数据预处理：主要是对上一阶段产生的数据进行再加工，**检查数据的完整性及数据的一致性，对其中的噪声数据进行处理**， 对丢失的数据可以利用统计方法进行填补。对一些不适合于操作的数据进行必要的处理等。
- 数据缩减：对经过预处理的数据，根据知识发现的任务对数据进行必要的再处理，**使数据集中在用户的挖掘目标上。**此过程对KDD系统的精度和效率起着至关重要的作用。它也可以通过数据库中的投影等相关操作或专门的工具来完成。
- KDD目标确定：根据挖掘的目标和用户的要求，**确定KDD所发现的具体知识模式和类型**，为选择或开发适合用户要求的数据挖掘算法提供模式或模板。
- 挖掘算法确定：根据上一阶段所确定的模式，**选择合适的数据挖掘算法**，这包括选取合适的参数、知识表示方式,并保证数据挖掘算法与整个KDD的评判标准相一致。数据挖掘:运用选定的算法，从数据中提取出用户所需要的知识。这些知识可以用一种特定的方式表示或使用一些常用的表示方式，如产生式规则等。
- 模式解释：对**发现的模式进行解释。**在此过程中，为了取得更为有效的知识,可能会返回前面处理步骤中的某些步以改进结果，**保证提取出的知识是有效的和可用的。**
- 知识评价：**将发现的知识以用户能了解的方式呈现给用户。**这期间也包含对知识的一致性的检查,以确信本次发现的知识与以前发现的知识不相抵触。

上述的每个处理阶段，KDD系统都可以借助相应的处理工具来完成相应的工作。掘的知识进行评价后，根据结果可以决定是否重新进行某些处理过程，在处理的任都可以返回以前的阶段进行再处理。整个KDD模型呈现出阶梯状的称作阶梯处理过程模型。

4.2 事务数据库中的关联规则挖掘

1. 发现所有的**频繁项集**：通过用户给定Minsupport ，寻找所有频繁项目集或者最大频繁项目集。
2. 从频繁项集中**发现关联规则**：通过用户给定Minconfidence ，在频繁项目集中，寻找强关联规则。

4.3 时间序列挖掘

简述时间序列挖掘的含义、意义、分类及其主要任务。

时间序列挖掘就是要从大量的时间序列数据中提取人们事先不知道的、但又是**潜在有用的与时间属性相关的信息和知识**，并用于短期、中期或长期预测，指导人们的社会、经济、军事和生活等行为。通过对过去历史行为的客观记录分析，揭示其内在规律，进而完成预测未来行为等决策性工作。

4.4 空间数据挖掘

简述空间数据挖掘的含义、意义、分类及其主要任务。

- 空间挖掘也被称作空间数据挖掘，或者空间数据库的知识发现。顾名思义，它是数据挖掘在空间数据库或其他数据空间数据上的应用。简言之，空间挖掘，就是从空间数据库等空间数据中挖掘隐含的知识模式，包括空间相关的基于空间的关联关系、聚类分类等模式，用于理解和归纳空间数据。

4.5 Web挖掘

简述Web挖掘的含义、意义、分类及其主要任务。

4.5.1 Web挖掘的含义

针对包括Web页面内容、页面之间的结构、用户访问信息、电子商务信息等在内的各种Web数据，应用数据挖掘方法以帮助人们从WWW中提取知识。另外由于Web挖掘的特点，像信息检索(Information Retrieval，IR)和信息抽取(Information Extraction，IE)等这样的研究领域的交叉研究也值得关注。

4.5.2 Web挖掘的意义

1. 从大量的信息中发现用户感兴趣的信息。
2. 将Web上的丰富信息转变成有用的知识。
3. 对用户进行信息个性化。

4.5.3 Web挖掘的分类

Web挖掘依靠的站点信息来源可以分为**Web内容挖掘**(Web Content Mining)、**Web访问信息挖掘**(Web Usage Mining)和**Web结构挖掘**(Web Structure Mining)三种主要类型。

1. **Web内容挖掘**：对站点的Web页面的各类信息进行集成、概化、分类等，挖掘某类信息所蕴含的知识模式。
2. **Web访问信息挖掘**：Web访问信息挖掘是对用户访问Web时在服务器方留下的访问记录进行挖掘。通过分析日志记录中的规律，可以识别用户的忠实度、喜好、满意度，可以发现潜在用户，增强站点的服务竞争力。
3. **Web结构挖掘**：Web结构挖掘是对Web页面之间的链接结构进行挖掘。在整个Web空间里，有用的知识不仅包含在Web页面的内容之中，而且也包含在页面的链接结构之中。对于给定的Web页面集合，通过结构挖掘可以发现页面之间的关联信息，页面之间的包含、引用或者从属关系等。

五、证明题

项目集空间理论：频繁项目集的子集是频繁项目集；非频繁项目集的超集是非频繁项目集。

5.1 Apriori属性1

请证明：如果项目集X是频繁项目集，那么它的所有非空子集都是频繁项目集。

证明如下：

设X是一个项目集，事务数据库T中支持X的元组数为s。对X的任一非空子集为Y，设T中支持Y的元组数为s_1。

根据项目集支持数的定义，很容易知道支持X的元组一定支持Y，所以$s_1 \geq s$，即support(Y) ≥ support(X)。

按假设：项目集X 是频繁项目集，即support(X) ≥ minsupport，所以support(Y) ≥ support(X) ≥ minsupport，因此Y是频繁项目集。

5.2 Apriori属性2

请证明：如果项目集X 是非频繁项目集，那么它的所有超集都是非频繁项目集。

证明如下：

设X是一个项目集，事务数据库T中支持X的元组数为s。对X的任一超集为Z，设T中支持Z的元组数为s_2。

根据项目集支持数的定义，很容易知道支持Z的元组一定支持X，所以$s \geq s_2$，即support(X) ≥ support(Z)。

按假设：项目集X是非频繁项目集，即minsupport ≥ support(X) ，所以minsupport ≥ support(X) ≥ support(Z) ，因此Z不是频繁项目集。