一、绪论

1.数据挖掘的定义：

数据挖掘是在大型数据存储库中，自动地发现有用的信息的过程。

2.数据挖掘的过程：

输入数据->数据预处理->数据挖掘->后处理->信息

数据预处理：**预处理是将未加工的数据转化为适合分析的形式**。包括融合来自多个数据源的数据、清洗数据、数据去重、去噪。

3.数据挖掘要解决的问题：

- **可伸缩**

  处理海量的数据集、算法必须是可伸缩的。

  使用**特殊的搜索策略**处理指数级搜索问题；**新的数据结构**；

  使用抽样技术或者开发并行和分布算法也可以提高可伸缩的程度。

- **高维性**

  传统的数据分析技术不能很好的处理高维数据。

  对某些算法，随着维度的增加，计算复杂度迅速增加。

- **异种数据类型和复杂数据**

  半结构化文本、Web 页面集、DNA 结构、气象数据等。

- **数据的所有权和分布**

  分布式数据挖掘：通信量；有效性；安全性。

- **非传统分析**

  假设检验不再适用；

  随机样本->时机样本。

4.数据挖掘涉及的学科：

- 统计学：抽样、估计、假设检验等；
- 人工智能、机器学习、模式识别；
- 并行计算和分布计算。

4.数据挖掘的任务：

- **预测性任务**：根据其他属性值来预测特定属性的值（预测建模）

  自变量（说明变量）-> 因变量（目标变量）

  eg：分类（鸢尾花分类）、回归（预测股票的价格）

- **描述性任务**：导出数据集中潜在的模式（相关、趋势、聚类、轨迹、异常等）

  eg：聚类分析（文档聚类）、关联分析（购物篮数据分析）、异常检测（信用卡欺诈检测）

二、数据

1.属性

（1）属性类型

标称：相异性
序数：相异性+序
区间：相异性+序+加法
比率：相异性+序+加法+乘法
注：定性属性（分类属性）：标称和序数；
定量属性（数值属性）：区间和比率；
对于区间属性，值之间的差是有意义的；
对于比率属性，值之间的差和比都是有意义的；
非对称的属性：关注非零属性的值。

3.数据集的一般特性

- 维度

维灾难->维归约

- 稀疏性

只有非零值才需要存储和处理，节省时间、空间。

- 分辨率

在不同的分辨率下得到的数据的性质是不同的；
数据的模式依赖数据的分辨率；
当分辨率太低，模式可能不出现；
当分辨率太高，可能看不出或者淹没在噪声中。

补充说明：
（1）时间自相关：如果两次测量时间很接近，则这些测量值通常很相似；
（2）空间自相关：在物理上靠近的对象趋于在其他方面也相似。

4.数据质量

- 测量误差和数据收集错误

数据收集错误：遗漏数据对象或属性值、不当的包含了其他数据对象等错误。

- 噪声和伪像

噪声是随机产生的；
伪像是确定性现象的结果（比如一组照片在同一位置出现条纹）。

- 精度、偏倚量、准确率
- 离群点（异常）

离群点也可以是合法的数据对象或值；
离群点有时是人们感兴趣的对象。

- 遗漏值

对遗漏值的处理方法：
（1）删除属性或对象；
（2）估计遗漏值；
（3）在分析时忽略遗漏值。

- 不一致的值

纠正不一致的值。

- 重复值

进行去重，要避免将两个相似的但并非重复的数据对象合并在一起（如两个人都叫张三）。

5.数据的预处理

预处理的操作大致分为两类:**选择分析所需要的数据对象和属性；创建/改变属性。**

- **聚集**

聚集是指将两个或者多个对象合并成单个对象。

聚集的优点：

（1）数据规约导致较小的数据集需要较小的内存和处理时间，因此可以使用开销更大的数据挖掘算法；

（2）通过高层数据视图，聚集起到了范围或者标度转换的作用；

（3）数据对象或者属性群的行为通常比单个对象或属性的行为更加稳定。

聚集的缺点：更能会丢失一些信息。

- **抽样**

（1）简单随机抽样：有放回和无放回。

当总体由不同类型的对象组成，每种类型的对象数量差别很大时，简单随机抽样不能充分地代表不太

出现的对象类型，此时就会出现问题。

（2）分层抽样

（3）自适应抽样/渐进抽样

从一个小样本开始，然后增加样本容量直到得到足够容量的样本（适用于样本容量不好确定时）。

- **维归约**

维归约的好处：

（1）如果数据的维度较低，许多数据挖掘算法的效果就更好。

（删除了不相关的特征并降低噪音；避免维灾难）

（2）降低了数据挖掘算法的时间和内存需求；

（3）使模型更容易理解；

（4）更容易进行数据可视化。

维灾难：随着数据维度的增加，许多数据分析变得越来越困难，数据在它所占据的空间中越来越稀疏。

对于分类，没有足够的数据对象创建模型；对于聚类，点之间的距离和密度的定义失去了意义。

维灾难的解决办法：主成分分析（PCA）和奇异值分解（SVD）。

注：主成分分析是找出新属性，这些属性是原属性的线性组合并相互正交的并捕获了数据的最大变差。

- **特征子集的选择（特征工程）**

理想算法：将所有可能的特征子集作为感兴趣的数据挖掘算法的输入，然后选取产生最好结果的子集。

n 个属性有 2^n 个子集，实现起来不现实。

特征子集选择方法：嵌入、过滤、包装

（1）嵌入：将特征选择作为数据挖掘算法的一部分，由算法本身决定使用和忽略哪些属性。

（2）过滤：把特征子集的选择作为一个独立的部分，在数据挖掘算法开始前进行特征选择。

（3）包装：将目标数据挖掘算法看作一个黑盒子，使用类似于理想算法的方法，但并不

枚举所有子集。

- **特征创建**

  特征提取->将特征映射到新的数据空间（傅里叶变化/小波变换）->特征构造

- **离散化和二元化**

  离散化就是决定选择多少个分割点和确定分割点的位置。

  无监督离散化（不使用类标签）：等宽、等频、等深。

  有监督离散化（使用类标签）：基于熵。

- **变量变换**

  使用一些简单的函数；

  规范化和标准化。

$$x' = \left(x - \overline{x}\right) / s_x$$

6. 相似性和相异性度量

- **常见的变换**

  s'=(s-min_s)/(max_s-min_s)

  d'=(d-min_d)/(max_d-min_d)

  d'=d/(1+d)

  注：使用任何单调减函数都可以用来将相异度转化为相似度。

Attribute Type	Dissimilarity	Similarity
Nominal	$d = \begin{cases} 0 & \text{if } p = q \\ 1 & \text{if } p \neq q \end{cases}$	$s = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{if } p \neq q \end{cases}$
Ordinal	$d = \frac{\lvert p-q \rvert}{n-1}$ (values mapped to integers 0 to $n-1$, where n is the number of values)	$s = 1 - \frac{\lvert p-q \rvert}{n-1}$
Interval or Ratio	$d = \lvert p - q \rvert$	$s = -d$, $s = \frac{1}{1+d}$ or $s = 1 - \frac{d - min\_d}{max\_d - min\_d}$

Table 5.1. Similarity and dissimilarity for simple attributes

- **相异性**

  常见的距离：欧氏距离、闵可夫斯基距离、曼哈顿距离。

- Euclidean Distance

$$dist = \sqrt{\sum_{k=1}^{n} (p_k - q_k)^2}$$

- Minkowski Distance is a generalization of Euclidean Distance

$$dist = \left(\sum_{k=1}^{n} \lvert p_k - q_k \rvert^r\right)^{\frac{1}{r}}$$

- $r = 1$. City block (Manhattan, taxicab, L_1 norm) distance.
  - A common example of this is the Hamming distance, which is just the number of bits that are different between two binary vectors

  距离的性质：非负性、对称性、三角不等式。

1. $d(p, q) \geq 0$ for all p and q and $d(p, q) = 0$ only if $p = q$. (Positive definiteness)
2. $d(p, q) = d(q, p)$ for all p and q. (Symmetry)
3. $d(p, r) \leq d(p, q) + d(q, r)$ for all points p, q, and r. (Triangle Inequality)

集合差和时间相异度度量：

D(A,B) = Size(A-B) + Size(B-A)

- 相似度

非对称的相似度度量

S'(A,B) = (S(A,B)+S(B,A))/2

SMC、Jaccard、余弦相似度、广义 Jaccard、相关性

$$SMC = \frac{f_{11} + f_{00}}{f_{01} + f_{10} + f_{11} + f_{00}}$$

$$J = \frac{f_{11}}{f_{01} + f_{10} + f_{11}}$$

$$\cos(x, y) = \frac{\langle x, y \rangle}{\|x\| \|y\|}$$

$$EJ(x, y) = \frac{\langle x, y \rangle}{\|x\|^2 + \|y\|^2 - \langle x, y \rangle}$$

在进行距离度量时，可以对属性加权：

- May not want to treat all attributes the same.
  - Use weights w_k which are between 0 and 1 and sum to 1.

$$similarity(p, q) = \frac{\sum_{k=1}^{n} w_k \delta_k s_k}{\sum_{k=1}^{n} \delta_k}$$

$$distance(p, q) = \left(\sum_{k=1}^{n} w_k |p_k - q_k|^r \right)^{1/r}$$

7. 汇总统计

频率和众数

百分位数：给定一个有序的或连续属性 X 和取值在 0-100 之间的 P，Xp 是这样的一个 x 值，使得 X 的 P%的观测值都比 Xp 小。

注：求百分位数的方法

第 1 步：以递增地顺序排列原始数据；

第 2 步：计算指数 i=n*p%

第 3 步：若 i 不是整数，将 i 向上取整。大于 i 的毗邻整数即就是 Xp 的位置；

若 i 是整数，Xp 即就是第 i 项与第 i+1 项的平均值。

均值和中位数

截断均值：指定 0-100 的数 P，丢弃高端和低端的(P/2)%的数据，然后再求均值。

特别地，当 P=100 时，即就是中位数；当 P=0 时，即就是普通的均值。

极差和方差

注意：均值和方差对离群点和噪音点十分敏感

绝对平均偏差

$$\mathrm{AAD}(x) = \frac{1}{m}\sum_{i=1}^{m}|x_i - \overline{x}|$$

中位数绝对偏差

$$\mathrm{MAD}(x) = median\left(\{|x_1 - \overline{x}|, \ldots, |x_m - \overline{x}|\}\right)$$

四分位数极差

$$\text{interquartile range}(x) = x_{75\%} - x_{25\%}$$

协方差矩阵和相似矩阵

$$covariance(x_i, x_j) = \frac{1}{m-1}\sum_{k=1}^{m}(x_{ki} - \overline{x_i})(x_{kj} - \overline{x_j})$$

$$correlation(x_i, x_j) = \frac{covariance(x_i, x_j)}{s_i s_j}$$

8. 数据可视化

数据可视化的一般步骤：表示->安排->选择

- **表示**：将数据映射为图形元素。

  将数据对象、属性、对象之间的联系映射成可视化的图像元素（如：点、线、形状、颜色等）。
- **安排**：在可视化中，项的安排至关重要。
- **选择**：删除或者不突出某些对象或者属性。

数据可视化技术：

- **低维数据**：茎叶图、直方图、盒装图、饼图、百分位数图、散布图。

  注：经验累计分布函数（ECDF）：显示小于该值的点的百分比。

  散布图优势：图形化地显示了两个属性之间的关系；当类标号给出时，可以使用散布图考察两个属性

  将类分开的程度。
- **时间、空间数据**：等高线图、曲面图、矢量场图。
- **高维数据**：平行坐标系（坐标轴是平行的而不是正交的；对象用线表示而不是用点表示）

星型坐标：对每个属性使用一个坐标轴，从中心点向四周辐射，均匀的散开。通常将属性值映射到 0-1。

cherrnoff 脸：将每个属性与脸的特征联系起来、每一个对象都是一个独立的脸。

- **OLAP（联机处理分析）**：非常关注交互式的数据分析，并且提供可视化数据和汇总统计的能力。

几个名词解释：

转轴：在除了 2 个维之外的所有维度上聚集。

切片：通过对一个或者多个维度指定特定的值，从整个数组中选择一组单元。

切块：通过指定属性值区间选择单元子集。

上卷和下钻：与聚集相关的操作。

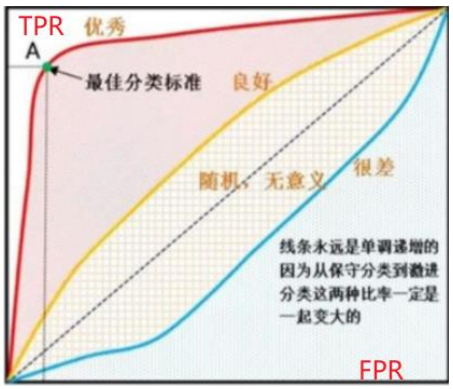
9.评价指标

预测 / 已知	阳性	阴性
正类	真阳（TP）	假阴（FN）
负类	假阳（FP）	真阴（TN）

ACC = right/all

Precision = TP/(TP+FP)

Recall = TP/(TP+FN)



TPR = TP/(TP+FN)

FPR = FP/(FP+TN)

AUC = **曲线下面积（越接近 1 效果越好）**

三、分类

- ## 决策树（Hunt、ID3、C4.5、CART）

（1）决策树的构成：根节点、内部节点、叶节点

（2）决策树设计要考虑的两个问题：

Q1：如何分裂训练记录？--选择最佳属性测试条件

Q2：如何停止分裂过程？--设置结束/终止条件

（3）决策树的发展：Hunt -> ID3 -> C4.5 -> CART

（4）**不纯性度量：**

Entropy：

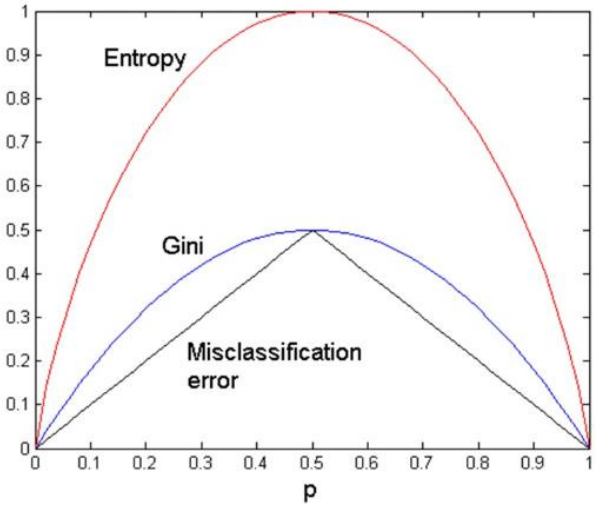
$$H(X) = -\sum_{i=1}^{n} p_i \log p_i$$

Gini：

$$\text{Gini}(p) = \sum_{k=1}^{K} p_k(1-p_k) = 1 - \sum_{k=1}^{K} p_k^2$$

Classification-error：

$$\text{Error}(t) = 1 - \max P(i \mid t)$$



（5）Hunt 算法：

简介：通过将训练记录相继划分成较纯的子集，以递归的方式建立决策树。

步骤：step1：如果 Dt(与节点 t 相关联的训练记录集)中所有的结点都属于同一类 Yt，则 t 是叶子节点；

step2：如果 Dt 中包含多个类的记录，则选择一个属性测试条件，将记录划分成较小的子集。

对测试条件的每个输出都创建一个子节点，并根据测试条件将 Dt 中的记录分布到子节点中，

然后对每个子节点递归地使用这种算法。

（6）ID3：

ID3 是建立在奥卡姆剃刀的基础上，越是小型的决策树越是优于较大的决策树。

（奥卡姆剃刀：给定两个具有相同泛化误差的模型，较简单的模型比较复杂的模型更可取。）

分类标准：信息增益。

信息增益 = 信息熵-条件熵（表示得知特征 A 的信息而使得样本集合不确定性减少的程度）

信息增益越大，表示使用特征 A 划分所获得的纯度提升越大。

$$H(D)=-\sum_{k=1}^{K}\frac{|C_k|}{|D|}log_2\frac{|C_k|}{|D|}$$

$$\begin{aligned}H(D|A)&=\sum_{i=1}^{n}\frac{|D_i|}{|D|}H(D_i)\\&=-\sum_{i=1}^{n}\frac{|D_i|}{|D|}(\sum_{k=1}^{K}\frac{|D_{ik}|}{|D_i|}log_2\frac{|D_{ik}|}{|D_i|})\end{aligned}$$

$$Gain(D,A)=H(D)-H(D|A)$$

缺点：（1）没有剪枝策略，容易导致过拟合现象；

（2）倾向于选择属性值较多的特征；

（3）只能处理离散分布的特征；

（4）没有考虑缺失值。

（7）C4.5：

分类标准：信息增益率 => 优先选择属性值较少的特征；

$$Gain_{ratio}(D,A)=\frac{Gain(D,A)}{H_A(D)}$$

$$H_A(D)=-\sum_{i=1}^{n}\frac{|D_i|}{|D|}log_2\frac{|D_i|}{|D|}$$

采用“启发式方法”：先从候选划分特征中找到信息增益高于平均值的特征，再从中选择信息增益率最高的；

优点：（1）采用后剪枝 => 避免过拟合；C4.5 采用悲观剪枝方法，用递归的方法从低往上针对每一个非叶子节点，评估用一个最佳子节点去替代这颗子树是否有益。如果剪枝后错误率保持不变或者下降，这颗子树就可以被替代。

（2）将连续属性离散化 => 既能处理离散分布的特征，也能处理连续分布的特征；

（3）对缺失值进行处理：特征选择时，对于具有缺失值的特征，用没有缺失的样本子集所占的比重来折算；在划分记录时，对于缺失值样本，以不同的概率值划分到不同节点。

缺点：（1）使用熵模型拥有大量耗时的对数运算；

（2）对于连续值要进行排序，从中选择分割点；

（3）只适合于能够驻留于内存的数据集，当训练集大得无法容纳内存时，程序无法运行。

剪枝：过拟合的树在泛化能力上表现得很差，所以要进行剪枝。

预剪枝：（1）含义：在节点划分前确定是否继续增长。

（2）停止增长的策略：1）节点内数据样本数低于某一阈值；

2）所有的节点特征都已分裂（没有特征可用）；

3）节点分裂后准确率不升反降。

（3）预剪枝的好处：不仅可以降低过拟合的风险，还可以减少训练时间。

（4）预剪枝的弊端：基于“贪心”策略会带来欠拟合的风险。

后剪枝：（1）含义：初始决策树按最大规模生长，然后进行剪枝。

（2）后剪枝的好处：后剪枝的欠拟合的风险很小，泛化性能往往优于预剪枝。

（3）后剪枝的弊端：训练时间会多很多。

（8）CART：

分类标准：基尼系数。

步骤：**分裂 -> 剪枝 -> 树选择**

分裂：没有停止准则，会一直生长下去；

剪枝：采用代价复杂度剪枝，从最大树开始，每次选择对整体性能影响最小的那个分裂节点

作为下一个剪枝对象，直至只剩下根节点。CART 会产生一系列嵌套的剪枝树，需要从

中选出一颗最优的决策树。

树选择：用单独的测试集评估每颗剪枝树的预测性能（也可以使用交叉验证）

优点：（1）不仅可以用于分类，还可以用于回归；

（2）使用二叉树，计算速度快；

（3）使用基尼系数作为不纯度度量，避免了复杂的对数运算。

- KNN

简介：k 邻近算法是一种基于实例的学习方法，是一种懒惰学习，没有显式的学习过程，即就是没有训练

阶段。数据集有特征值和类标签，待接收新样本后直接进行处理。

思想：每个样本的都可以用它最邻近的 k 个邻居来代表。

步骤：（1）计算该未知样本到所有已知样本的距离；

（2）选取与该未知节点最近的 k 个训练对象；

（3）统计（2）步骤中 k 个对象的类别；

（4）确定一种决策规则来判定未知样本所属类别（一般采用基于投票的分类规则）。

若采用多数表决法，每个近邻对分类的影响都是一样的，这使得算法对 K 值很敏感。降低 K 的

影响的一种途径就是**根据每个近邻到该点的距离进行加权**(W=1/d^2),使得距离较远的训练样例比

靠近的训练样例对分类结果的影响小。

注意：（1）k 值的选择：

k 值的选择会极大程度上影响 KNN 分类结果。

k 值较小：对噪声敏感；会使训练误差减小而使泛化误差增大即模型过于复杂导致过拟合。

k 值较大：导致模型过于简单，容易出现欠拟合的情况。

极端情况下，把 k 值设为样本总数，无论新输入的点落在何处，该模型都会简单地

把它预测为训练样本中出现最多的类别。

k 值通常取奇数：如果采用投票机制，避免因两种票数相同而难以决策。

（2）距离的度量：欧几里得距离、闵可夫斯基距离、曼哈顿距离。

在传统的欧氏距离中，各特征的权重相同，即认为各个特征对于分类的贡献是相同的，显然

这并不符合实际情况。同等权重使得特征向量的相似度计算不够准确，进而影响分类度。

距离会被大量不相关的属性所支配=>对每个属性加权：越近权值越高。

Wij = 1/（dij^2)

优点：（1）算法简单、易于理解、易于实现；

（2）适合处理多分类问题；

（3）比较合适应用在低维空间。

缺点：（1）时间复杂度高，需要计算未知样本到所有已知样本的距离；

（2）样本平衡度依赖高，当出现极端情况样本不平衡时，分类会出现偏差；

（3）可解释性差，无法给出类似于决策树那样的规则；

（4）特征向量维度越高，距离的区分能力就越弱。

- ## 朴素贝叶斯算法

贝叶斯公式：

$$P(A|B)=\frac{P(A)P(B|A)}{P(B)}$$

换一种表达形式：

$$P(c|x)=\frac{P(c)P(x|c)}{P(x)}$$

“朴素”是指假设各个特征之间相互独立。

问题：为什么要假设各个特征之间相互独立？

答：现实生活中，被研究的对象会有非常多的特征，每个特征都有多个取值，此时通过统计来估计概率

会变得很难；由于数据的稀疏性，很容易统计到 0 的情况。

$$P(c|x)=\frac{P(c)P(x|c)}{P(x)}=\frac{P(c)}{P(x)}\prod_{i=1}^{d}P(x_i|c)$$

优点：（1）算法简单，易于实现；

（2）时空开销小。

缺点：（1）理论上讲朴素贝叶斯分类与其它分类相比具有最小的误差率，但实际上并非如此；

（2）假设模型中各个属性相互独立，这个条件在现实生活中往往并不成立；

（3）当属性个数比较多或者属性之间的相关性较大时，分类效果并不好。

- **分类器性能的评估**

（1）保持方法：将数据集划分成训练集和验证集。

在训练集上训练分类模型，在验证集上评估模型的性能。

（2）随机二次抽样：多次重复保持方法来改进对分类器性能的估计。

ACC = mean(acc)

（3）交叉验证：二折交叉验证 -> K-折交叉验证

优点：使用了尽可能多的训练记录；

验证集之间是互斥的；

缺点：过程重复 k 次，计算开销很大；

方差偏高。

（4）自助法：有放回抽样，即已经选为训练的记录将放回原来的记录集中，使得它被等几率的抽取。

1 - (1 - 1/N)^N = 1 - e^-1 ≈ 0.623

四、聚类

Q1：聚类和分类区别？

聚类的目的是把相似的数据聚合在一起；具体划分时并不关心这一类的标签；属于无监督学习；

分类是通过训练集获得分类器，用分类器去预测未知数据的标签;属于有监督学习。

Q2：数据对象之间的相似度度量？

相似度度量准则	相似度度量函数
Euclidean 距离	$d(x,y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$
Manhattan 距离	$d(x,y)=\sum_{i=1}^{n}\|x_i-y_i\|$
Chebyshev 距离	$d(x,y)=\max_{i=1,2,\ldots,n}^{n}\|x_i-y_i\|$
Minkowski 距离	$d(x,y)=[\sum_{i=1}^{n}(x_i-y_i)^p]^{\frac{1}{p}}$

Q3：cluster 之间的相似性度量？

相似度度量准则	相似度度量函数
Single-link	$D(C_i,C_j)=\min_{x\subseteq C_i,y\subseteq C_j} d(x,y)$
Complete-link	$D(C_i,C_j)=\max_{x\subseteq C_i,y\subseteq C_j} d(x,y)$
UPGMA	$D(C_i,C_j)=\frac{1}{\|C_i\|\|C_j\|}\sum_{x\subseteq C_i,y\subseteq C_j} d(x,y)$
WPGMA	-

Single-Link(单链)：两个簇之间的距离定义为为两个簇之间距离最近的两个点之间的距离。

会在聚类中产生“链式效应”，即就是有可能出现非常大的簇。

Complete-Link（全链）：两个簇之间的距离定义为为两个簇之间距离最远的两个点之间的距离。

能避免“链式效应”，但是容易受到异常点的影响，容易产生不合理的聚类。

注意：单链和全链仅考虑了某个有特点的数据，没有考虑数据的整体特性。

UPGMA（平均值）：两个簇之间的距离定义为两个簇之间所有点的距离的平均值。

WPGMA（加权平均值）：两个簇之间的距离定义为两个簇之间所有点的距离的加权平均值。

加权后使距离的计算不受簇的大小的影响。

Q4：聚类的方法？

数据聚类的方法可以分为：划分式聚类、层次化聚类、基于密度的聚类等。

- **划分式聚类**：需要事先指定簇类的数目或者聚类的中心，通过反复迭代，直至最后达到簇

  内的点足够近，簇间的点足够远的目的。

kmeans：步骤：（1）随机选取 K 个点作为初始质心；

（2）把其它数据点指派到离它最近的质心，形成新的类；

（3）重新计算每个类的质心（平均向量）；

（4）重复 2-3，直到质心不再变化或者达到最高迭代次数。

特点：（1）需要提前确定 K 值；

（2）对初始的质心点敏感；

（3）对异常点敏感。

kmeans++：是对 kmeans 中初始质心点选取的优化算法。

步骤：（1）随机选取一个数据点作为初始的聚类中心；

（2）当聚类中心的的数量小于 K 时：计算每个数据点到当前已有聚类中心的最短距离，用

D(x)表示，这个值越大，表示被选取为下一个聚类中心的概率就越大，最后使用轮盘

法选取下一个聚类中心。

bi-kmeans：是针对 kmeans 算法陷入局部最优的缺陷进行的改进算法。

思想：该算法基于 SSE(Sum of Squared Error)最小化原理。首先将所有的数据点视为一个簇，然

后将该簇一分为二，之后选择其中的一个簇继续划分，选择标准是能最大程度降低 SSE 的值

的簇的划分。

步骤：（1）将所有的点视为一个簇；

（2）当簇的个数小于 K 时：对于每一个簇计算总误差 -> 在给定的簇上面进行 2-means 聚类

-> 计算将该簇一分为二后的总误差 -> 选取使得误差最小的那个簇进行划分操作。

- **基于密度的聚类**

参数：邻域半径 R 和最小点数目 minpoints。

点的类别：核心点、边界点、噪声点。

核心点：其邻域半径 R 内样本点的数量大于等于 minpoints；

边界点：不属于核心点但是在核心点的 R 邻域内；

噪声点：既不是核心点也不是边界点定义为噪声点。

几个名词：密度直达、密度可达、密度相连、非密度相连。

密度直达：P 是核心点，Q 在 P 的 R 邻域内，则 P->Q 密度直达；（不具有对称性）

密度可达：如果存在核心点 P1，P2，……，Pn 且 P1 到 P2 密度直达，P2 到 P3 密度直达，……，

且 Pn 到 Q 密度直达，则 P1 到 Q 密度可达；（不具有对称性）

密度相连：存在核心点 S，S 到 P 和 Q 都密度可达，则 P 和 Q 密度相连；

密度相连具有对称性：P 和 Q 密度相连则 Q 和 P 密度相连；

密度相连的两个点属于同一个簇；

非密度相连：两个点不属于同一个簇或者存在噪声点。

步骤：

1. 标记所有对象为unvisited
2. 当有标记对象时
    1. 随机选取一个unvisited对象p
    2. 标记p为visited
    3. 如果p的ε邻域内至少有M个对象，则
        1. 创建一个新的簇C，并把p放入C中
        2. 设N是p的ε邻域内的集合，对N中的每个点p'
            1. 如果点p'是unvisited
                1. 标记p'为visited
                2. 如果p'的ε邻域至少有M个对象，则把这些点添加到N
                3. 如果p'还不是任何簇的成员，则把p'添加到C
        3. 保存C
    4. 否则标记p为噪声

特点：（1）需要提前确定半径 R 和阈值 N；

（2）不需要提前设置聚类的个数；

（3）对初值的设置敏感，但是对噪声不敏感；

（4）对密度不均的数据聚合效果不好。

- **层次化聚类**

思想：将数据集划分成一层一层的簇，后边一层生成的簇基于前面一层簇的结果。

分类：Agglomerative 和 Divisive。

（1）Agglomerative 即**自底向上**（bottom-up）的层次聚类：

最开始时，每一个对象都是一个独立的簇，每次按照一定的准则将最相近的两个簇合并生成一个新

的簇，如从往复，直至最终所有的对象都属于一个簇。

（2）Divisive 即**自顶向下**（top-down）的层次聚类：

开始时所有的对象属于同一个簇，每次按照一定的准则将某个簇划分为多个簇，如此往复，直至最

终每一个对象都属于一个独立的簇。

步骤：以 bottom-up 层次聚类的为例

（1）初始时每个对象都是一个簇，计算距离矩阵 D(或者相似矩阵 S)；

（2）遍历距离矩阵 D，找出其中最小的距离（对角线除外），并由此得到两个簇的编号，将这两个簇合

并并根据簇之间的距离的度量标准更新距离矩阵，存储本次合并的相关信息；

（3）重复步骤 2 直至所有的对象属于同一个簇。

特点：层次聚类实际上是一种“贪心”策略，每次合并或者划分都是基于某种局部最优的选择。

- **各种聚类算法的比较**

算法类型	抗噪点性能	聚类形状	算法效率
kmeans	较差	球形	很高
kmeans++	一般	球形	较高
bi-kmeans	一般	球形	较高
DBSCAN	较好	任意形状	一般
Agglomerative	较好	任意形状	较差

五、关联分析

关联分析用于发现藏匿在大型数据集中的令人感兴趣的联系，所发现的模式通常用关联规则或

频繁项集的形式表示。关联分析可用于生物信息学、医疗诊断、网页挖掘、科学数据分析等。

- 项集（Itemset）
  - 包含0个或多个项的集合
    - 例子: {Milk, Bread, Diaper}
  - k-项集
    - 如果一个项集包含k个项
- 支持度计数（Support count）(σ)
  - 包含特定项集的事务个数
  - 例如： σ({Milk, Bread,Diaper}) = 2
- 支持度（Support）
  - 包含项集的事务数与总事务数的比值
  - 例如： s({Milk, Bread, Diaper}) = 2/5
- 频繁项集（Frequent Itemset）
  - 满足最小支持度阈值（ *minsup* ）的所有项集

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

- 关联规则
  - 关联规则是形如 $X \rightarrow Y$的蕴含表达式, 其中 X 和 Y 是不相交的项集
  - 例子:
    {Milk, Diaper} → {Beer}

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

3

- 关联规则的强度
  - 支持度 Support (s)
    - 确定项集的频繁程度
  - 置信度 Confidence (c)
    - 确定Y在包含X的事务中出现的频繁程度

Example:

$$\{Milk, Diaper\} \Rightarrow Beer$$

$$s = \frac{\sigma(Milk, Diaper, Beer)}{|T|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(Milk, Diaper, Beer)}{\sigma(Milk, Diaper)} = \frac{2}{3} = 0.67$$

关联规则挖掘问题

定义：给定事务的集合 T, 关联规则发现是指找出支持度大于等于 *minsup* 并且置信度大于等于 *minconf* 的所有规则, *minsup* 和 *minconf* 是对应的支持度和置信度阈值。

一种方法：Brute-force approach

- 计算每个可能规则的支持度和置信度
- 这种方法计算代价过高，因为可以从数据集提取的规则的数量达指数级
- 从包含d个项的数据集提取的可能规则的总数 $R = 3^d - 2^{d+1} + 1$ ，如果d等于6，则R=602

关联规则挖掘的步骤：频繁项集的产生 -> 生成规则

Apriori 算法

- 如何降低产生频繁项集的时间复杂度？

利用先验原理减少候选项集的数量 M；

使用更高级的数据结构，或者存储候选项集或者压缩数据集来减少比较次数 NM。

- **什么是先验原理？**

如果一个项集是频繁的，则它的所有子集也一定是频繁的；

如果一个项集是非频繁的，则它的所有超集也一定是非频繁的。

支持度度量的反单调性：一个项集的支持度决不会超出它的子集的支持度。

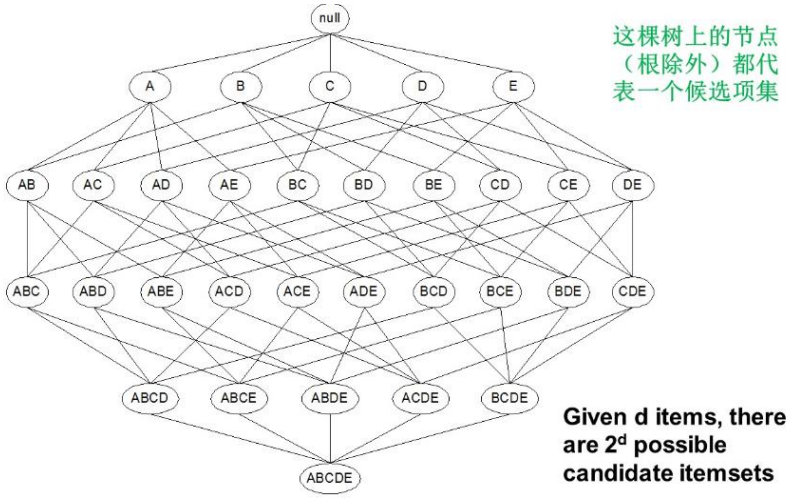
注：基于支持度度量修剪指数搜索空间的策略称为**基于支持度的剪枝**。

- **Apriori 算法的特点：**

（1）是一个逐层的算法，即就是从 1-频繁项集到最长的频繁项集，它每次遍历项集格中的一层；

（2）使用**产生-测试**的策略来发现频繁项集。在每次迭代过程中，新的候选项集由前一次迭代发现的频繁

项集产生，然后对每个候选项集的支持度进行计数，并与最小支持度阈值进行比较。



- **产生候选项集的三种方法**

蛮力法

思想：把格结构中的每个项集作为候选项集；

将每个候选项集和每个事务进行比较，确定每个候选项集的支持度计数。

缺点：时间复杂度为 O(NMW)，时间开销可能非常大。

$F_{k-1} \times F_1$方法

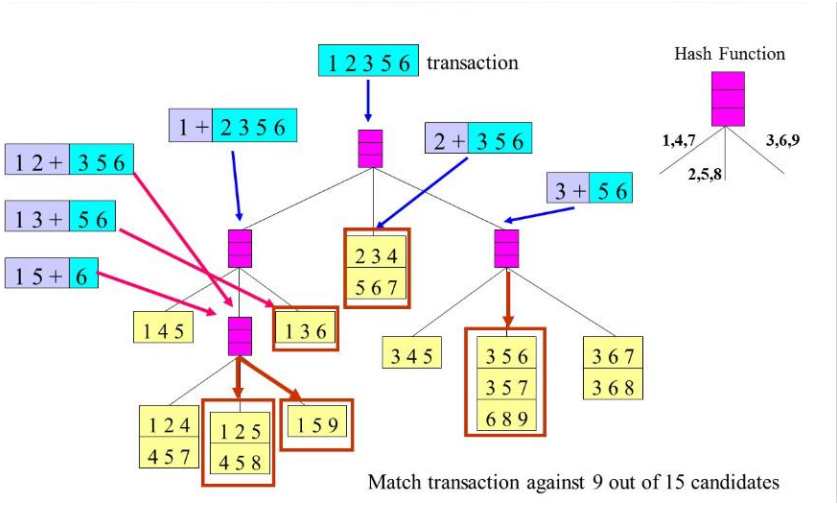
- 这种方法用其他频繁项来扩展每个频繁（k-1）-项集
- 这种方法将产生$O(|F_{k-1}| \times |F_1|)$个候选k-项集，其中$|F_j|$表示频繁j-项集的个数。这种方法总复杂度是$O(\sum_k k|F_{k-1}||F_1|)$
- 这种方法是完全的，因为每一个频繁k-项集都是由一个频繁（k-1）-项集和一个频繁1-项集组成的。因此，所有的频繁k-项集是这种方法所产生的候选k-项集的一部分。
- 然而，这种方法很难避免重复地产生候选项集。

$F_{k-1} \times F_{k-1}$方法

– 这种方法合并一对频繁（k-1）-项集，仅当它们的前k-2个项都相同。

如频繁项集{面包，尿布}和{面包，牛奶}合并，形成了候选3-项集{面包，尿布，牛奶}。算法不会合并项集{啤酒，尿布}和{尿布，牛奶}，因为它们的第一个项不相同。

采用哈希结构：

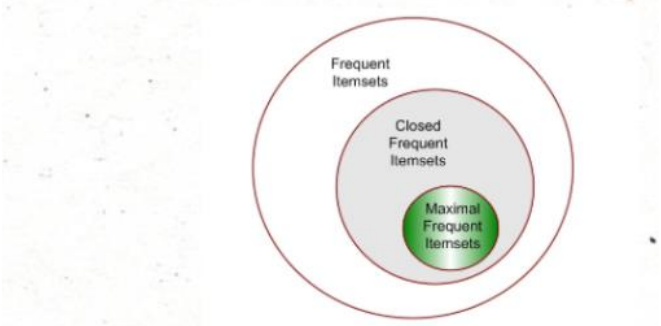


- **极大频繁项集和频繁闭项集：**

极大频繁项集：它的直接超集都不是频繁的；
有效地提供了频繁项集的紧凑表示；
形成了可以导出所有频繁项集的最小项的集合。

闭项集：它的直接超集都不具有和它相同的支持度计数。
即就是它的直接超集的支持度计数比它小。

频繁闭项集：是闭项集且其支持度大于等于最小支持度阈值。



- **如何根据闭频繁项集的支持度，得到非闭频繁项集的支持度？**

从最大的频繁项集向项集格的根部计算，非闭频繁项集的支持度等于其直接超集的最大支持度。

FP-Tree

- **产生背景：**Apriori 算法需要多次扫描数据，I/O 是很大的瓶颈；

FP-Tree 算法只需要两次扫描数据集，因此提高了算法运行的效率。

第一次扫描数据是得到所有频繁 1 项集，将频繁 1 项集放入项头表并按照支持度降序排列；

第二次扫描数据是将读到的原始数据剔除非频繁 1 项集，并按照支持度降序排列。

- **数据结构：**项头表、FP 树、节点链表。

  项头表：记录所有频繁 1 项集出现的次数，按照次数降序排列；

  FP-Tree：将原始数据集映射到该 Tree；

  节点链表：项头表里的所有频繁 1 项集都是一个节点链表的头，它依次指向 FPTree 中该频繁 1 项集出现的位置。这样便于项头表和 FPTree 之间的联系、查找和更新。

- **算法步骤：**

1）扫描数据，得到所有频繁一项集的的计数。然后删除支持度低于阈值的项，将1项频繁集放入项头表，并按照支持度降序排列。

2）扫描数据，将读到的原始数据剔除非频繁1项集，并按照支持度降序排列。

3）读入排序后的数据集，插入FP树，插入时按照排序后的顺序，插入FP树中，排序靠前的节点是祖先节点，而靠后的是子孙节点。如果有共用的祖先，则对应的公用祖先节点计数加1。插入后，如果有新节点出现，则项头表对应的节点会通过节点链表链接上新节点。直到所有的数据都插入到FP树后，FP树的建立完成。

4）从项头表的底部项依次向上找到项头表项对应的条件模式基。从条件模式基递归挖掘得到项头表项项的频繁项集。

5）如果不限制频繁项集的项数，则返回步骤4所有的频繁项集，否则只返回满足项数要求的频繁项集。

- **举例**

数据

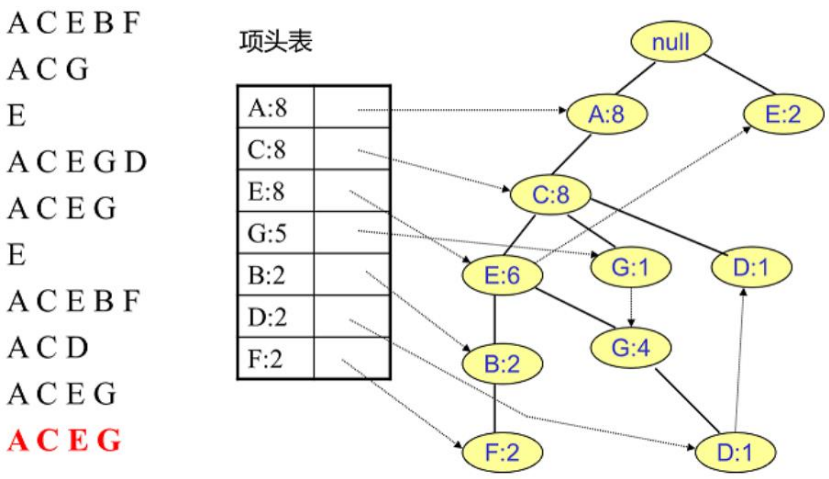
A B C E F O
A C G
E I
A C D E G
A C E G L
E J
A B C E F P
A C D
A C E G M
A C E G N

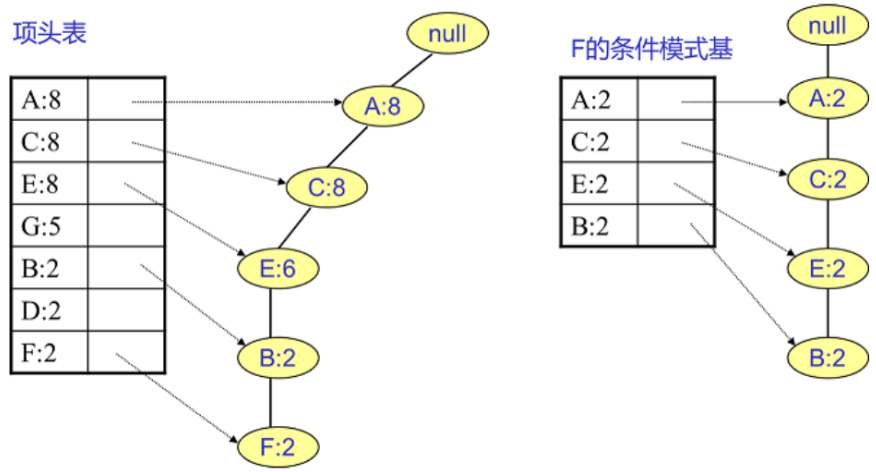
项头表.
支持度大于20%

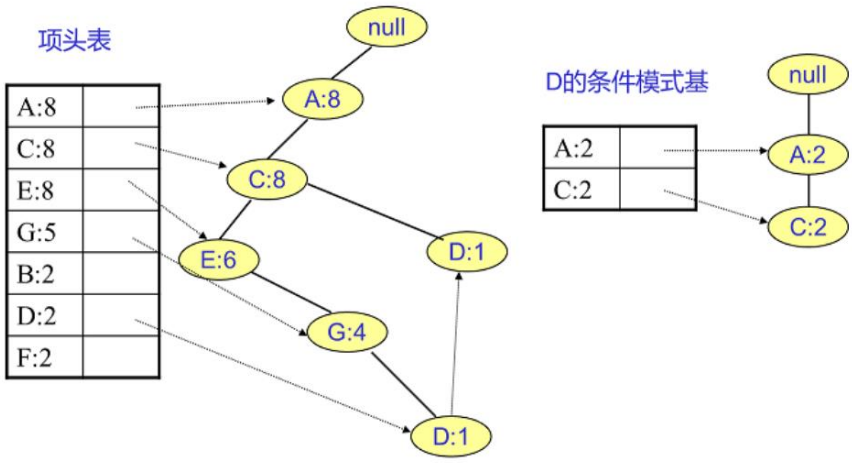
A:8	
C:8	
E:8	
G:5	
B:2	
D:2	
F:2	

排序后的数据集

A C E B F
A C G
E
A C E G D
A C E G
E
A C E B F
A C D
A C E G
A C E G







- **算法优点：**改进了 Apriori 算法的 I/O 瓶颈，巧妙地使用了树结构来提高算法的运行速度。

  利用内存数据结构**以空间换取时间**是常用的提高算法运行时间瓶颈的方法。