答题模板

2023年12月21日 22:17

- 概念：含义/定义&区别+主要思想、特征、类型
- 应用
- 流程：输入→算法过程→输出
- 评价：
    - 时空复杂度&证明
    - 优缺点/局限性：
        - 输入：适合处理的数据类型（高维等）
        - 过程：性能，时空复杂度，成本：计算量&时间→速度&效率，节省空间/存储需求
        - 整体：
            - 可解释性→合理性→可行性&问题解决（过拟合、非线性等）→易用性
            - 主/客观性&准确性：需要预先设定/估计初始值/参数[的多少]
            - 敏感性/灵敏度/稳定性/鲁棒性/健壮性：
                - 对缺失数据、离群点/孤立点/异常值、噪声（被测变量的随机错误或变化）（即抗噪性）
                - 对维度、初始值/参数、距离测量方法/测度
            - *可扩展性/适应性&局限性/理想化（需要假设条件独立）

概念

2024年1月2日 11:19

数据挖掘[、知识发现、数据预处理的目的]
数据、数据对象、数据集及类型、属性/特征及类型&特征的值、结构数据的特征
特征子集选择、创建方法
可视化：表示、安排、选择
分类及应用、决策树含义
广义误差、奥卡姆剃刀
*关联分析及应用、事务、项、项集、支持度计数
（空）、极大、 闭、闭频繁项集
聚类[分析]及应用

①什么是数据挖掘

2023年12月22日　　22:47

- 数据挖掘是在大型数据存储库中，自动地发现有用信息的过程
- 数据挖掘&知识发现：

数据挖掘是数据库中知识发现（knowledge discovery in database，KDD）不可缺少的一部分，而 KDD 是将未加工的数据转换为有用信息的整个过程，如图 1-1 所示。该过程包括一系列转换步骤，从数据的预处理到数据挖掘结果的后处理。

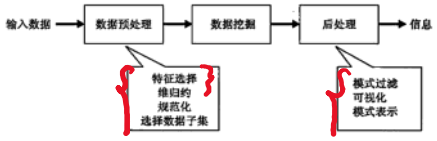


图 1-1　数据库中知识发现（KDD）过程

  - 归纳总结：
    - 数据预处理：规范（规范化）→维&特征（维归约&特征选择）→数据（选择数据子集）
    - 后处理：模式过滤&表示→可视化
    - 数据：未加工→（数据预处理）适合分析&大型数据存储库→（数据挖掘）有用信息
  - 数据预处理的目的：将未加工的输入数据转换成适合分析的形式

②数据挖掘要解决的问题

2023年12月23日 0:12

高维性、可伸缩、异种数据和复杂数据、数据的所有权与分布、非传统的分析（非假设-检验模式）

③数据挖掘的起源

2023年12月23日　0:15

- 起到支撑作用的领域：
  - 数据库系统技术：提供有效的存储、索引和查询处理支持
  - 高性能（并行）计算技术：处理海量数据集
  - 分布式计算技术：帮助处理海量数据，并且当数据不能集中到一起处理时更是至关重要
- 利用的领域思想：
  - 统计学：抽样、估计和假设检验
  - 数据挖掘
  - 人工智能、机器学习和模式识别：学习理论、建模技术、搜索算法

④数据挖掘任务

2023年12月23日　0:19

- 两大类：
  - 预测任务：根据其他属性（明变量/自变量）的值预测特定属性（目标变量/因变量）的值
  - 描述任务：
    - **描述任务。其目标是导出概括数据中潜在联系的模式（相关、趋势、聚类、轨迹和异常）。本质上，描述性数据挖掘任务通常是探查性的，并且常常需要后处理技术验证和解释结果。**
- 四种主要数据挖掘任务：
  - 预测建模：
    - 离散的目标变量：分类
    - 连续…：回归
  - 异常检测
  - 聚类分析
  - 关联分析

⑤本书的内容&组织

2023年12月23日 0:26

略~

①数据类型

2023年9月13日　10:34

- 数据含义：事实或观察的结果，是对客观事物的逻辑归纳，是用于表示客观事物的未经加工的原始素材
- 数据对象含义：具有相同性质的数据元素的类，有时也叫做事件/案例/模式、实体/点/向量、观测/记录/样本
- 数据[集]含义：数据对象的集合

属性/特征&度量

什么是属性

- 属性（也叫做特性/特征，字段/变量，维）：对象的性质或特性
- 测量标度：将数值或符号值与对象的属性相关联的规则（函数）

属性类型

分类型：标称、序数

数值型：区间、比率

属性类型		描　述	例　子
分类的（定性的）	标称	标称属性的值仅仅只是不同的名字，即标称值只提供足够的信息以区分对象（=，≠）	邮政编码、雇员 ID 号、眼球颜色、性别
	序数	序数属性的值提供足够的信息确定对象的序（<，>）	矿石硬度、{好，较好，最好}、成绩、街道号码
数值的（定量的）	区间	对于区间属性，值之间的差是有意义的，即存在测量单位（+，−）	日历日期、摄氏或华氏温度
	比率	对于比率变量，差和比率都是有意义的（*，/）	绝对温度、货币量、计数、年龄、质量、长度、电流

属性变换

属性类型		变　换	注　释
分类的（定性的）	标称	任何一对一变换，例如值的一个排列	如果所有雇员的 ID 号都重新赋值，不会出现任何不同
	序数	值的保序变换，即 新值=f(旧值)， 其中 f 是单调函数	包括好、较好、最好的属性可以完全等价地用值{1, 2, 3}或用{0.5, 1, 10}表示
数值的（定量的）	区间	新值 $=a*$旧值$+b$， 其中 a、b 是常数	华氏和摄氏温度的零度的位置不同，1 度的大小（即单位长度）也不同
	比率	新值$=a*$旧值	长度可以用米或英尺度量

值的个数

属性/特征值含义：分配给属性/特征的数字或符号

离散、连续

非对称的属性

略~

数据集的类型

有序、基于图形、记录数据

数据集的一般特性、结构数据的特征

- 维度（数据集中对象具有的属性数目）：维度灾难（高维数据）
- 稀疏性：只有出现的那些值才有意义
- 分辨率：取决于规模

有序数据

- 时序/时间数据：每个记录包含一个与之相关联的时间
- 序列数据：一个数据集合，是各个实体的序列
- 时间序列数据：一种特殊的时序数据，其中每个记录都是一个时间序列
- 空间数据：一个重要特征是空间自相关性

基于图形的数据

具有图形对象的数据、带有对象之间联系的数据

记录数据

记录矩阵&数据矩阵、文档-词矩阵、事务数据（一种特殊类型的记录数据，其中每个记录涉及一系列的项，如购物篮数据）

处理非记录数据

略~

②数据质量

2023年9月13日 11:19

测量和数据收集问题

测量和数据收集问题

- 测量误差和数据收集错误
- 噪声和伪像（数据的确定性失真，如一组照片在同一地方出现条纹）
- 精度、偏倚和准确率
  - 精度：同一个量的重复测量值之间的接近程度
  - 偏倚：测量值与被测量之间的系统的变差
  - 准确率：被测量的测量值与实际值之间的接近度
- 离群点：在某种意义上具有不同于数据集中其他大部分数据对象的特征的数据对象
- 遗漏值：
  - 在分析时忽略遗漏值
  - 删除数据对象或属性
  - 估计遗漏值
  - 用可能的值替代
- 不一致的值
- 重复数据
- 归纳总结：数据收集错误→遗漏值→重复数据&不一致的值→离群点（异常值）→噪声&伪像→精度&测量误差、偏倚和准确率

关于应用的问题

时效性、相关性、关于数据的知识

③数据预处理

2023年9月13日　11:36

- 归纳总结：
  - 数据：抽样→聚集/集成
  - 属性：特征子集选择&特征创建、维归约
  - 值（变换计算）：离散化（二元化）、变量变换

聚集/集成

略~

抽样

抽样方法

- 简单随机抽样：有、无放回抽样
- 分层抽样

渐进/自适应抽样

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

维归约

- 目的：
  - 使数据更易于可视化
  - 减少数据挖掘算法所需的时间和内存
  - 避免维度问题
  - 有助于消除无关特征、降噪
- 维灾难：随着数据维度的增加，许多数据分析变得非常困难
- 方法：维归约的线性代数技术（PCA、SVD）

特征子集选择

- 特征子集选择体系结构：
  - 将过滤和包装方法放到一个共同的体系结构中
  - 四部分：控制新的特征子集产生的搜索策略→子集评估度量→停止搜索判断→验证过程
  - 特征子集选择过程流程图：

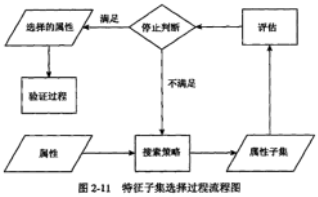


图 2-11　特征子集选择过程流程图

- 方法：
  - 蛮力法
  - 过滤方法：在数据挖掘算法运行前，使用某种独立于数据挖掘任务的方法进行特征选择
  - 嵌入方法：在数据挖掘算法运行期间，算法本身决定使用和忽略哪些属性
  - 包装方法：
    - 使用理想算法，将目标数据挖掘算法作为黑盒，但通常并不枚举所有可能的子集来找出最佳属性子集
    - 与过滤方法的唯一不同是它们使用了不同的特征子集评估方法
  - 特征加权

特征创建

- 特征提取：由原始数据创建新的特征集
- 特征构造
- 映射数据到新的空间（傅立叶分析、小波变换）

离散化和二元化

二元化、连续属性离散化（[非]监督离散化）、具有过多值的分类属性

变量变换

简单函数、规范化或标准化

④相似性和相异性度量

2023年9月20日　10:39

基础

- 两个对象之间相似度的非正式定义：两个对象相似程度的数值度量
- ...相异度：是这两个对象差异程度的数值度量
- 变换公式：

比较直截了当的。例如，如果对象之间的相似度在1（一点也不相似）和10（完全相似）之间变化，则我们可以使用如下变换将它变换到[0, 1]区间：$s' = (s-1)/9$，其中 s 和 s' 分别是相似度的原值和新值。一般来说，相似度到[0, 1]区间的变换由如下表达式给出：$s' = (s - min\_s) / (max\_s - min\_s)$，其中 max_s 和 min_s 分别是相似度的最大值和最小值。类似地，具有有限值域的相异度也能用 $d' = (d - min\_d) / (max\_d - min\_d)$ 映射到[0, 1]区间。

简单属性之间的相似度和相异度

属性类型	相异度	相似度
标称的	$d=\begin{cases}0 & \text{如果} x=y \\ 1 & \text{如果} x\neq y\end{cases}$	$s=\begin{cases}1 & \text{如果} x=y \\ 0 & \text{如果} x\neq y\end{cases}$
序数的	$d=\lvert x-y\rvert/(n-1)$（值映射到整数0到$n-1$，其中n是值的个数）	$s=1-d$
区间或比率的	$d=\lvert x-y\rvert$	$s=-d$，$s=\frac{1}{1+d}$，$s=e^{-d}$，$s=1-\frac{d-min\_d}{max\_d-min\_d}$

数据对象之间的相异度

- 欧式距离：

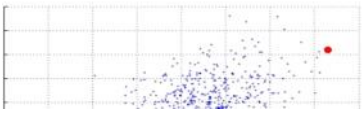
$$d(\mathbf{x},\mathbf{y})=\sqrt{\sum_{k=1}^{n}(x_k-y_k)^2}$$

- 闵可夫斯基距离：

$$d(\mathbf{x},\mathbf{y})=\left(\sum_{k=1}^{n}|x_k-y_k|^r\right)^{1/r}$$

- 马氏距离（Mahalanobis）：

$$mahalanobi\quad s(p,q)=(p-q)\sum^{-1}(p-q)^T$$



Σ is the covariance matrix of the input data X

$$\Sigma_{j,k}=\frac{1}{n-1}\sum_{i=1}^{n}(X_{ij}-\overline{X}_j)(X_{ik}-\overline{X}_k)$$

邻近性度量的例子

相似性

二元数据的相似性度量

设 x 和 y 是两个对象，都由 n 个二元属性组成。这样的两个对象（即两个二元向量）的比较可生成如下四个量（频率）：

f_{00} = x 取 0 并且 y 取 0 的属性个数
f_{01} = x 取 0 并且 y 取 1 的属性个数
f_{10} = x 取 1 并且 y 取 0 的属性个数
f_{11} = x 取 1 并且 y 取 1 的属性个数

简单匹配系数（Simple Matching Coefficient, SMC） 一种常用的相似性系数是简单匹配系数，定义如下：

$$SMC = \frac{\text{值匹配的属性个数}}{\text{属性个数}} = \frac{f_{11} + f_{00}}{f_{01} + f_{10} + f_{11} + f_{00}} \quad (2\text{-}5)$$

该度量对出现和不出现都进行计数。因此，*SMC* 可以在一个仅包含是非题的测验中用来发现回答问题相似的学生。

Jaccard 系数（Jaccard Coefficient） 假定 x 和 y 是两个数据对象，代表一个事务矩阵（见 2.1.2 节）的两行（两个事务）。如果每个非对称的二元属性对应于商店的一种商品，则 1 表示该商品被购买，而 0 表示该商品未被购买。由于未被顾客购买的商品数远大于被其购买的商品数，因而像 *SMC* 这样的相似性度量将会判定所有的事务都是类似的。这样，常常使用 Jaccard 系数来处理仅包含非对称的二元属性的对象。Jaccard 系数通常用符号 *J* 表示，由如下等式定义：

$$J = \frac{\text{匹配的个数}}{\text{不涉及0-0匹配的属性个数}} = \frac{f_{11}}{f_{01} + f_{10} + f_{11}} \quad (2\text{-}6)$$

余弦相似度

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$$

广义Jaccard系数

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

相关性

- [皮尔森]相关系数：

$$\text{corr}(\mathbf{x}, \mathbf{y}) = \frac{\text{covariance}(\mathbf{x}, \mathbf{y})}{\text{standard\_deviation}(\mathbf{x}) \times \text{standard\_deviation}(\mathbf{y})} = \frac{s_{xy}}{s_x s_y} \quad (2\text{-}10)$$

这里我们使用标准的统计学记号和定义：

$$\text{covariance}(\mathbf{x}, \mathbf{y}) = s_{xy} = \frac{1}{n-1} \sum_{k=1}^{n} (x_k - \bar{x})(y_k - \bar{y}) \quad (2\text{-}11)$$

$$\text{standard\_deviation}(\mathbf{x}) = s_x = \sqrt{\frac{1}{n-1} \sum_{k=1}^{n} (x_k - \bar{x})^2}$$

$$\text{standard\_deviation}(\mathbf{y}) = s_y = \sqrt{\frac{1}{n-1} \sum_{k=1}^{n} (y_k - \bar{y})^2}$$

$\bar{x} = \frac{1}{n} \sum_{k=1}^{n} x_k$ 是 x 的均值

$\bar{y} = \frac{1}{n} \sum_{k=1}^{n} y_k$ 是 y 的均值

$$= \frac{\sum_{k=1}^{n} (x_k - \bar{x})(y_k - \bar{y})}{\sqrt{\sum_{k=1}^{n} (x_k - \bar{x})^2 \sum_{k=1}^{n} (y_k - \bar{y})^2}}$$

数据对象的密度

基于中心、单元（cell）、图、概率的密度：略~

邻近度计算问题

- 距离度量的标准化和相关性：马氏距离（见上~）
- 组合异种属性的相似度：

算法 2.1 异种对象的相似度

1：对于第 k 个属性，计算相似度 $s_k(\mathbf{x}, \mathbf{y})$，在区间[0, 1]中。
2：对于第 k 个属性，定义一个指示变量 δ_k，如下：
  $\delta_k = 0$，如果第 k 个属性是非对称属性，并且两个对象在该属性上的值都是 0，或者如果一个对象的第 k 个属性具有遗漏值
  $\delta_k = 1$，否则
3：使用如下公式计算两个对象之间的总相似度：

$$\text{similarity}(\mathbf{x}, \mathbf{y}) = \frac{\sum_{k=1}^{n} \delta_k s_k(\mathbf{x}, \mathbf{y})}{\sum_{k=1}^{n} \delta_k} \quad (2\text{-}15)$$

- 使用权值：略~

0总体架构

2023年9月27日 10:36

- 数据探索：有助于选择合适的数据预处理和数据分析技术，着重于利用数据可视化来理解和解释数据挖掘结果，而不包含聚类分析、异常检测等主题（相比于EDA）
- 探索性数据分析（EDA）：更全面~

①鸢尾花数据集

2023年9月20日 11:47

略~

②汇总统计

2023年9月27日　10:39

- 频率度量：频率、众数
- 位置度量：均值、中位数、百分位数
- 散布度量：
  - 极差&方差（除以的是样本数[m]-1）
  - 更稳健的估计：绝对平均偏差（AAD）、中位数绝对偏差（MAD）、四分位数极差（IQR）：

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

$$\text{MAD}(x) = median(\{|x_1 - \bar{x}|, \ldots, |x_m - \bar{x}|\})$$

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

- 多元汇总统计：
  - 协方差矩阵：

对于多元数据，每个属性的散布可以独立于其他属性，使用 3.2.4 节介绍的方法计算。然而，对于具有连续变量的数据，数据的散布更多地用协方差矩阵（covariance matrix）**S** 表示，其中，**S** 的第 ij 个元素 s_{ij} 是数据的第 i 个和第 j 个属性的协方差。这样，如果 x_i 和 x_j 分别是第 i 个和第 j 个属性，则

$$s_{ij} = \text{covariance}(x_i, x_j) \quad (3\text{-}10)$$

而 $\text{covariance}(x_i, x_j)$由

$$\text{covariance}(x_i, x_j) = \frac{1}{m}\sum_{k=1}^{m}(x_{ki} - \overline{x_i})(x_{kj} - \overline{x_j}) \quad (3\text{-}11)$$

给出，其中 x_{ki}和 x_{kj}分别是第 k 个对象的第 i 和第 j 个属性的值。注意，$\text{covariance}(x_i, x_i) = \text{variance}(x_i)$。这样，协方差矩阵的对角线上是属性的方差。

  - 相关矩阵：

两个属性的协方差是两个属性一起变化并依赖于变量大小的度量。协方差的值接近于 0 表明两个变量不具有（线性）关系，但是不能仅靠观察协方差的值来确定两个变量之间的关联程度。因为两个属性的相关性直接指出两个属性（线性）相关的程度，对于数据探索，相关性比协方差更可取。（另见 2.4.5 节关于相关的讨论。）相关矩阵（correlation matrix）**R** 的第 ij 个元素是数据的第 i 个和第 j 个属性之间的相关性。如果 x_i 和 x_j 分别是第 i 个和第 j 个属性，则

$$r_{ij} = \text{correlation}(x_i, x_j) = \frac{\text{covariance}(x_i, x_j)}{s_i s_j} \quad (3\text{-}12)$$

其中，s_i 和 s_j 分别是 x_i 和 x_j 的方差。**R** 的对角线上的元素是 $\text{correlation}(x_i, x_i) = 1$，而其他元素在−1 和 1 之间。考虑包含每对对象而不是每对属性之间相关性的相关矩阵也是有用的。

  - 个人分析：此相关系数应该为皮尔森相关系数
- 汇总数据的其他方法：略~

③可视化

2023年9月27日 11:16

- 可视化技术在数据挖掘方面的应用可称作可视化数据挖掘

  # 可视化的动机

  略~

  # 一般概念：

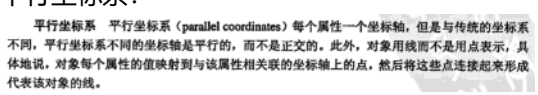
- 表示：将数据映射到图形元素
- 安排：正确选择对象和属性的可视化，即整合数据
- 选择：删除或不突出某些对象和属性，处理很多属性的方法是维归约（最常用方法是使用属性子集，其他还有PCA等）

  # 技术

- 少量属性的可视化：
  - 茎叶图：观测一维整型或连续数据的分布
  - [相对频率、二维]直方图&条形图：
    - [相对频率、二维]直方图：
      - 将可能的值分散到箱中，并显示落入每个箱中的对象数/相对频率，从而显示属性值的分布
      - 分类、连续属性都可以表示，其中连续属性通常是等宽的，但不必是等宽的
      - 二维直方图：将每个属性划分成区间
    - 条形图：
      - 在直方图的基础上，每个箱用一个条形表示，并且每个条形的面积正比于落在对应区间的值（对象）的个数
      - 可以不等宽
  - 盒状图/箱线图：显示一维数值属性值分布
  - 饼图：通常用于具有相对较少的值的分类属性
  - 百分位数图和经验累计分布函数：更定量地显示数据分布
    - 累计分布函数（CDF）：显示点小于该值的概率
    - 经验累计分布函数（ECDF）：显示小于该值的点的百分比
  - 散布图：
    - 使用数据对象两个属性的值作为x和y坐标值，每个数据对象都作为平面上的一个点绘制
    - 用途：
      - 图形化地显示两个属性之间的关系
      - 考察两个属性将类分开的程度
- 可视化时间空间数据：
  - 等高线图：应用场景是两个属性指定平面上的位置，而第三个属性具有连续值
  - 曲面图：*使用两个属性表示x和y坐标，曲面图的第三个属性用来指示高出前两个属性定义的平面的高度
  - 矢量场图：某些数据中，一个特性可能同时具有值和方向
  - 低维切片：（高维的）时间空间数据集，记录不同地点和时间上的某种量
  - 动画：显示数据的相继二维切片

  # 可视化高维数据

- 矩阵
- 平行坐标系：



平行坐标系 平行坐标系（parallel coordinates）每个属性一个坐标轴，但是与传统的坐标系不同，平行坐标系不同的坐标轴是平行的，而不是正交的。此外，对象用线而不是用点表示，具体地说，对象每个属性的值映射到与该属性相关联的坐标轴上的点，然后将这些点连接起来形成代表该对象的线。

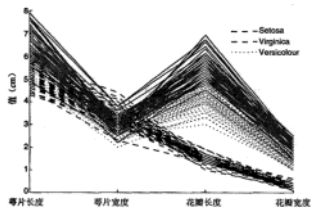
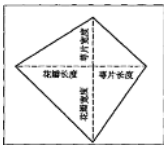


图 3-25　4 个鸢尾花属性的平行坐标图

- 星形坐标&Chemoff脸：
  - 星形坐标：



  - Chemoff脸：将属性数据分别对应脸部各特征的大小

*注意事项

ACCENT原则：理解、清晰性、一致性、有效性、必要性、真实性

- **理解（Apprehension）**。正确察觉变量之间关系的能力。图形能够最大化对变量之间关系的理解吗？
- **清晰性（Clarity）**。以目视识别图形中所有元素的能力。最重要的元素或关系在视觉上最突出吗？
- **一致性（Consistency）**。根据与以前的图形的相似性解释图形的能力。元素、符号形状和颜色与以前图形使用的一致吗？
- **有效性（Efficiency）**。用尽可能简单的方法描绘复杂关系的能力。图形元素的使用经济吗？图形容易解释吗？
- **必要性（Necessity）**。对图形和图形元素的需要。与其他替代方法（表、文本）相比，图形是提供数据的更有用的形式吗？为表示关系，所有的图形元素都是必要的吗？
- **真实性（Truthfulness）**。通过图形元素相对于隐式或显式尺度的大小，确定图形元素所代表的真实值的能力。图形元素可以准确地定位和定标吗？

*④OLAP&多维数据分析

2023年9月27日 11:40

用多维数组表示鸢尾花数据

略~

多维数据：一般情况

略~

分析多维数据

- 数据立方体：计算聚集量
- 维归约&转轴（在除两个维之外的所有维上聚集）
- 切片&切块
- 上卷和下钻

最后评述

略~

①预备知识

2023年9月27日 11:43

- 分类定义：分类任务就是通过学习得到一个目标函数，把每个属性集x映射到一个预先定义的类标号y
- 目标函数也称为分类模型，用于描述性建模、预测性建模
- 分类的应用：

不同的应用。例如：根据电子邮件的标题和内容检查出垃圾邮件，根据核磁共振扫描的结果区分肿瘤是恶性的还是良性的，根据星系的形状对它们进行分类（见图 4-1）。

②解决分类问题的一般方法

2023年9月27日　　11:56

- 混淆矩阵的准确率&错误率：

分类模型的性能根据模型正确和错误预测的检验记录计数进行评估，这些计数存放在称作混淆矩阵（confusion matrix）的表格中。表 4-2 描述二元分类问题的混淆矩阵。表中每个表项 f_{ij} 表示实际类标号为 i 但被预测为类 j 的记录数，例如，f_{01} 代表原本属于类 0 但被误分为类 1 的记录数。按照混淆矩阵中的表项，被分类模型正确预测的样本总数是（$f_{11}+f_{00}$），而被错误预测的样本总数是（$f_{10}+f_{01}$）。

表 4-2　二类问题的混淆矩阵

		预测的类	
		类 =1	类 =0
实际的类	类 =1	f_{11}	f_{10}
	类 =0	f_{01}	f_{00}

虽然混淆矩阵提供衡量分类模型性能的信息，但是用一个数汇总这些信息更便于比较不同模型的性能。为实现这一目的，可以使用性能度量（performance metric），如准确率（accuracy），其定义如下：

$$\text{准确率} = \frac{\text{正确预测数}}{\text{预测总数}} = \frac{f_{11}+f_{00}}{f_{11}+f_{10}+f_{01}+f_{00}} \tag{4-1}$$

同样，分类模型的性能可以用错误率（error rate）来表示，其定义如下：

$$\text{错误率} = \frac{\text{错误预测数}}{\text{预测总数}} = \frac{f_{10}+f_{01}}{f_{11}+f_{10}+f_{01}+f_{00}} \tag{4-2}$$

③决策树归纳

2023年9月27日　　11:58

应用：预测，具体为分类

决策树的工作原理

- 含义：

  在决策树中，每个叶结点都赋予一个类标号。非终结点（non-terminal node）（包括根结点和内部结点）包含属性测试条件，用以分开具有不同特性的记录。例如，在图 4-4 中，在根结点处，

  - 归纳总结：
    - 叶结点：赋予一个类标号
    - 非终结点（根结点&内部结点）：包含属性测试条件（用以分开具有不同特性的记录）

如何建立决策树

Hunt算法

- 递归（类似于构建二叉树）：

(1) 如果 D_t 中所有记录都属于同一个类 y_t，则 t 是叶结点，用 y_t 标记。
(2) 如果 D_t 中包含属于多个类的记录，则选择一个属性测试条件（attribute test condition），将记录划分成较小的子集。对于测试条件的每个输出，创建一个子女结点，并根据测试结果将 D_t 中的记录分布到子女结点中。然后，对于每个子女结点，递归地调用该算法。

- 图示：

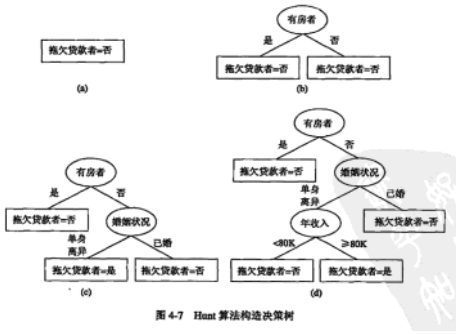


图 4-7　Hunt 算法构造决策树

决策树归纳的设计问题

略~

表示属性测试条件的方法

略~

选择最佳划分的度量

- 不纯程度：

$$\text{Entropy}(t) = -\sum_{i=0}^{c-1} p(i|t)\log_2 p(i|t)$$

$$\text{Gini}(t) = 1-\sum_{i=0}^{c-1}[p(i|t)]^2$$

$$\text{Classification error}(t) = 1-\max_i[p(i|t)]$$

  - 其中p(i|t)为结点t中属于类i的记录所占的比例
  - 二分类：

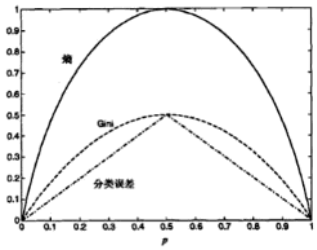


图 4-13　二元分类问题不纯性度量之间的比较

- 增益：
  - 通过比较父结点（划分前）和子女结点（划分后）的不纯程度来判断划分的效果
  - 熵作为不纯度计算方法时，增益又可具体称为信息增益
  - $$\Delta = I(\text{parent}) - \sum_{j=1}^{k}\frac{N(v_j)}{N}I(v_j) \qquad (4\text{-}6)$$

    其中，$I(.)$是给定结点的不纯性度量，N 是父结点上的记录总数，k 是属性值的个数，$N(v_j)$是与子女结点 v_j 相关联的记录个数。决策树归纳算法通常选择最大化增益Δ的测试条件，因为对所有的

- 信息增益率：

也考虑进去，例如，决策树算法 C4.5 采用称作增益率（gain ratio）的划分标准来评估划分。增益率定义如下：

$$Gain\ ratio = \frac{\Delta_{info}}{Split\ Info} \tag{4-7}$$

其中，划分信息 $Split\ Info = -\sum_{i=1}^{k} P(v_i)\log_2 P(v_i)$，而 k 是划分的总数。例如，如果每个属性值具有相同的记录数，则$\forall i$：$P(v_i) = 1/k$，而划分信息等于 $\log_2 k$。这说明如果某个属性产生了大量的划分，它的划分信息将会很大，从而降低了增益率。

决策树归纳算法

算法 4.1 给出了称作 TreeGrowth 的决策树归纳算法的框架。该算法的输入是训练记录集 E 和属性集 F。算法递归地选择最优的属性来划分数据（步骤 7），并扩展树的叶结点（步骤 11 和步骤 12），直到满足结束条件（步骤 1）。算法的细节如下。

(1) 函数 createNode()为决策树建立新结点。决策树的结点或者是一个测试条件，记作 *node.test_cond*，或者是一个类标号，记作 *node.label*。

(2) 函数 find_best_split()确定应当选择哪个属性作为划分训练记录的测试条件。如前所述，测试条件的选择取决于使用哪种不纯性度量来评估划分，一些广泛使用的度量包括熵、Gini 指标和χ^2统计量。

(3) 函数 Classify()为叶结点确定类标号。对于每个叶结点 t，令 $p(i|t)$表示该结点上属于类 i 的训练记录所占的比例，在大多数情况下，都将叶结点指派到具有多数记录的类：

$$leaf.label = \underset{i}{\operatorname{argmax}}\ p(i|t) \tag{4-8}$$

其中，操作 argmax 返回最大化 $p(i|t)$的参数值 i。$p(i|t)$除了提供确定叶结点类标号所需要的信息之外，还可以用来估计分配到叶结点 t 的记录属于类 i 的概率。5.7.2 节和 5.7.3 节讨论如何使用这种概率估计，在不同的代价函数下，确定决策树的性能。

(4) 函数 stopping_cond()通过检查是否所有的记录都属于同一个类，或者都具有相同的属性值，决定是否终止决策树的增长。终止递归函数的另一种方法是，检查记录数是否小于某个最小阈值。

算法 4.1 决策树归纳算法的框架

```
TreeGrowth(E, F)
1: if stopping_cond(E, F) = true then
2:   leaf = createNode()
3:   leaf.label = Classify(E)
4:   return leaf
5: else
6:   root = createNode()
7:   root.test_cond = find_best_split(E, F)
8:   令 V = {v | v 是 root.test_cond 的一个可能的输出}
9:   for 每个 v∈V do
10:    E_v = {e | root.test_cond(e) = v 并且 e ∈ E}
11:    child = TreeGrowth(E_v, F)
12:    将 child 作为 root 的派生结点添加到树中，并将边(root→child)标记为 v
13:  end for
14: end if
15: return root
```

- 归纳总结-工作过程（步骤、停止判断、划分方式比建立决策树的Hunt算法更丰富、具体）：
  1. 若满足停止条件（记录都属于同一类或具有相同的属性值，或记录数小于某个最小阈值），则为叶结点确定类标号并停止构建
  2. 否则，确定并选择一个最佳的属性测试条件（包括熵、Gini系数、分类错误率等不纯程度度量方式）
  3. 将记录划分为较小的子集：对于测试条件的每个输出，创建一个子女节点，并根据测试结果将记录分布到子女结点中
  4. 递归调用此算法来构建子树

- 常见熵计算结果（若考试中log2(3)=1.58、log2(5)=2.32）：
  - 4+5：0.9822
  - 2+3：0.972
  - 1+3：0.815
  - （2+2）+（2+3）：0.9844
  - （1+3）+（2+3）：0.9022

④模型的过分拟合

2023年10月18日　10:35

误差分类：训练/表现/再代入误差、检验/泛化误差（测试集上的误差）、广义误差（从同一分布中随机选择记录时模型的预期误差）

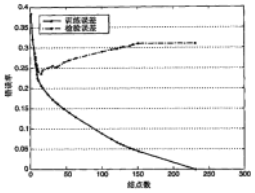


图 4-23　训练误差和检验误差

噪声导致的过分拟合

略~

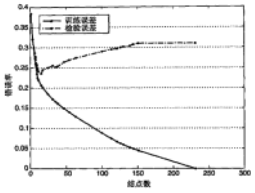


图 4-23　训练误差和检验误差

缺乏代表性样本导致的过分拟合

略~

过分拟合&多重比较过程

模型的过分拟合可能出现在多重比较过程的学习算法中

泛化误差估计

使用再代入/乐观估计

再代入估计方法假设训练数据集可以很好地代表整体数据，因而，可以使用训练误差（又称再代入误差）提供对泛化误差的乐观估计。在这样的前提下，决策树归纳算法简单地选择产生最低训练误差的模型作为最终的模型。然而，训练误差通常是泛化误差的一种很差的估计。

结合模型复杂度（在再代入/乐观估计的基础上）

- 奥卡姆剃刀/节俭原则：相同泛化误差的模型中选择更简单的那个
- 悲观误差估计：

悲观误差评估　第一种方法明确使用训练误差与模型复杂度罚项（penalty term）的和计算泛化误差。结果泛化误差可以看作模型的悲观误差估计（pessimistic error estimate）。例如，设 $n(t)$ 是结点 t 分类的训练记录数，$e(t)$ 是被误分类的记录数。决策树 T 的悲观误差估计 $e_g(T)$ 可以用下式计算：

$$e_g(T)=\frac{\sum_{i=1}^{k}[e(t_i)+\Omega(t_i)]}{\sum_{i=1}^{k}n(t_i)}=\frac{e(T)+\Omega(T)}{N_t}$$

其中，k 是决策树的叶结点数，$e(T)$ 决策树的总训练误差，N_t 是训练记录数，$\Omega(t_i)$ 是每个结点 t_i 对应的罚项。

- *最小描述长度原则（MDL）：没讲~

最小描述长度原则　另一种结合模型复杂度的方法是基于称作最小描述长度（minimum description length，MDL）原则的信息论方法。为了解释说明该原则，考虑图 4-28 中的例子。在该例中，A 和 B 都是已知属性 x 值的给定记录集。另外，A 知道每个记录的确切类标号，而 B 却不知道这些信息。B 可以通过要求 A 顺序传送类标号而获得每个记录的分类。一条消息需要 $\Theta(n)$ 比特的信息，其中 n 是记录总数。

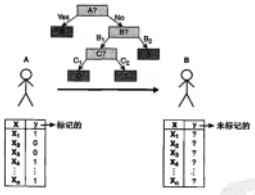


图 4-28　最小描述长度（MDL）原则

另一种可能是，A 决定建立一个分类模型，概括 x 和 y 之间的关系。在传送给 B 前，模型用压缩形式编码。如果模型的准确率是 100%，那么传输的代价就等于模型编码的代价。否则，A 还必须传输哪些记录被模型错误分类信息。传输的总代价是：

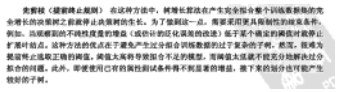
$$Cost(model,data)=Cost(model)+Cost(data\mid model)\qquad(4\text{-}9)$$

其中，等式右边的第一项是模型编码的开销，而第二项是误分类记录编码的开销。根据 MDL 原则，我们寻找最小化开销函数的模型。本章习题 9 给出了一个如何计算决策树总描述长度的例子。

处理决策树归纳的过分拟合

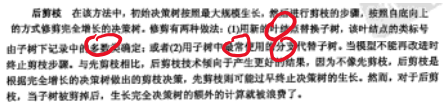
- 先/预剪枝（提前终止规则）：

先剪枝（提前终止规则）　在这种方法中，树增长算法在产生完全拟合整个训练数据集的完全增长的决策树之前就停止决策树的生长。为了做到这一点，需要采用更具限制性的终止条件。例如，当观察到的不纯性度量的增益（或估计的泛化误差的改进）低于某个确定的阈值时就停止扩展叶结点。这种方法的优点在于避免产生过分拟合训练数据的过于复杂的子树。然而，很难为提前终止选取正确的阈值。阈值太高将导致给分不足的模型。而阈值太低就不能充分地解决过分拟合的问题。此外，即便使用已有的属性测试条件得不到显著的增益，基于树的划分也可能产生



- 归纳总结：
  - 典型：所有实例都属于同一类别、属性值都相同
  - 更严格：阈值法（树深度、记录/实例数量、增益、泛化误差）、数据划分法；实例的类别分布与可用特征无关

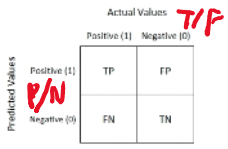
- 后剪枝：



⑤评估分类器的性能

2023年10月18日 11:23

- 混淆矩阵：



- 常见参数：
  - 准确率/正确率：(TP+TN)/(TP+TN+FP+FN)
  - 精确率/查准率（是否错报，主体是预测值）：TP/(TP+FP)
  - 召回率/查全率（是否漏报，主体是真实值）、TPR（真正类率，代表分类器预测的正类中实际正实例占所有正实例的比例）：TP/(TP+FN)
  - FPR（假正类率，代表分类器预测的正类中实际负实例占所有负实例的比例）：FPR = FP/(FP+TN)
  - F1 score：精确率和召回率的调和平均值：2*准确率*召回率/(准确率+召回率)
  - P-R曲线：横轴是召回率R，纵轴是精确率P
  - ROC曲线（receiveroperating characteristic，接收者操作特征）：横轴为FPR，纵轴为TPR
  - AUC（Area under Curve，ROC曲线下的面积），介于0.1和1之间，直观的评价分类器的好坏，值越大越好

保持方法

分比例通常根据分析家的判断（例如，50-50，或者 2/3 作为训练集、1/3 作为检验集）。分类器的准确率根据模型在检验集上的准确率估计。

保持方法有一些众所周知的局限性。第一，用于训练的被标记样本较少，因为要保留一部分记录用于检验，因此，建立的模型不如使用所有数据建立的模型好。第二，模型可能高度依赖于训练集和检验集的构成。一方面，训练集越小，模型的方差越大，另一方面，如果训练集太大，根据用较小的检验集估计的准确率又不太可靠，这样的估计具有很宽的置信区间。最后，训练集和检验集不再是相互独立的。因为训练集和检验集来源于同一个数据集，在一个子集中超出比例的类在另一个子集就低于比例，反之亦然。

- 归纳总结：训练→训练&检验集相互独立→模型对训练&检验集的依赖（训练集越小/大）

随机二次抽样

可以多次重复保持方法来改进对分类器性能的估计，这种方法称作随机二次抽样（random subsampling）。设 acc_i 是第 i 次迭代的模型准确率。总准确率是 $acc_{sub} = \sum_{i=1}^{k} acc_i / k$。随机二次抽样也会遇到一些与保持方法同样的问题。因为在训练阶段也没有利用尽可能多的数据，并且，由于它没有控制每个记录用于训练和检验的次数。因此，有些用于训练的记录使用的频率可能比其他记录高很多。

交叉验证

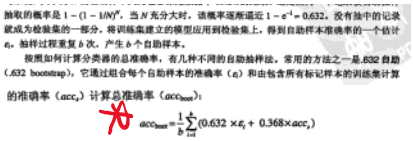
替代随机二次抽样的一种方法是交叉验证（cross-validation）。在该方法中，每个记录用于训练的次数相同，并且恰好检验一次。为了解释该方法，假设把数据分为相同大小的两个子集。首先，我们选择一个子集作训练集，而另一个作检验集，然后交换两个集合的角色，原先作训练集的现在做检验集。反之亦然。这种方法叫二折交叉验证。总误差通过对两次运行的误差率和得到。在这个例子中，每个样本各作一次训练样本和检验样本。k 折交叉验证是对该方法的推广，把数据分为大小相同的 k 份，在每次运行，选择其中一份作检验集，而其余的全作为训练集。该过程重复 k 次，使得每份数据都用于检验恰好一次。同样，总误差是所有 k 次运行的误差之和。k 折交叉验证方法的一种特殊情况是令 $k = N$，其中 N 是数据集的大小。在这种所谓留一（leave-one-out）方法中，每个检验集只有一个记录。该方法的优点是使用尽可能多的训练记录。此外，检验集之间是互斥的，并且有效地覆盖了整个数据集；该方法的缺点是整个过程重复 N 次，计算上开销极大。此外，因为每个检验集只有一个记录，性能估计度量的方差偏高。

- 归纳总结-步骤：
  - 将记录划分为相同大小的子集
  - 每次选择一个子集作为测试集，其他作为训练集
  - 重复K次，使得每个子集都恰好作为过一次测试集
  - 总误差为所有k次运行的误差之和
- 重复交叉验证：进行多次
- 分层交叉验证：保证训练和测试中类别标签的百分比相同

自助法

迄今为止，我们介绍的方法都是假定训练记录采用不放回抽样。因此，训练集和检验集都不包含重复记录。在自助（bootstrap）方法中，训练记录采用有放回抽样，即已经选作训练的记录将放回原来的记录集中，使得它等机率地被重新抽取。如果原始数据有 N 个记录，可以证明，平均来说，大小为 N 的自助样本大约包含原始数据中 63.2% 的纪录。这是因为一个记录被自助抽样抽取的概率是 $1 - (1 - 1/N)^N$，当 N 充分大时，该概率逐渐逼近 $1 - e^{-1} = 0.632$。没有抽中的记录就成为检验集的一部分，将训练集建立的模型应用到检验集上，得到自助样本准确率的一个估计 ϵ_i。抽样过程重复 b 次，产生 b 个自助样本。

按照如何计算分类器的总准确率，有几种不同的自助抽样法。常用的方法之一是 .632 自助（.632 bootstrap），它通过组合每个自助样本的准确率（ϵ_i）和由包含所有标记样本的训练集计算的准确率（acc_s）计算总准确率（acc_{boot}）：



抽取的概率是 $1-(1-1/N)^N$。当 N 充分大时，该概率逐渐逼近 $1-e^{-1}=0.632$。没有抽中的记录就成为检验集的一部分，将训练集建立的模型应用到检验集上，得到自助样本准确率的一个估计 ε_i。抽样过程重复 b 次，产生 b 个自助样本。

按照如何计算分类器的总准确率，有几种不同的自助抽样法。常用的方法之一是.632 自助（.632 bootstrap），它通过组合每个自助样本的准确率（ε_i）和由包含所有标记样本的训练集计算的准确率（acc_s）计算总准确率（acc_{boot}）：

$$acc_{boot}=\frac{1}{b}\sum_{i=1}^{b}(0.632\times\varepsilon_i+0.368\times acc_s)$$

- 归纳总结：保持方法→随机二次抽样、交叉验证（重复&分层）、自助法

⑥比较分类器的方法

2023年10月25日 10:49

估计准确度的置信区间

略~

比较两个模型的性能

略~

比较两种分类法的性能

略~

0归纳总结

2023年12月23日 23:36

- 决策树含义：

  在决策树中，每个叶结点都赋予一个类标号。非终结点（non-terminal node）（包括根结点和内部结点）包含属性测试条件，用以分开具有不同特性的记录。例如，在图 4-4 中，在根结点处，

- 分类应用：

  不同的应用。例如：根据电子邮件的标题和内容检查出垃圾邮件，根据核磁共振扫描的结果区分肿瘤是恶性的还是良性的，根据星系的形状对它们进行分类（见图 4-1）。

- 过程：
  - 决策树（Hunt算法&归纳）：
    - Hunt算法：

(1) 如果 D_t 中所有记录都属于同一个类 y_t，则 t 是叶结点，用 y_t 标记。
(2) 如果 D_t 中包含属于多个类的记录，则选择一个属性测试条件（attribute test condition），将记录划分成较小的子集。对于测试条件的每个输出，创建一个子女结点，并根据测试结果将 D_t 中的记录分布到子女结点中。然后，对于每个子女结点，递归地调用该算法。

    - 归纳：
      1. 若满足停止条件（记录都属于同一类或具有相同的属性值，或记录数小于某个最小阈值），则为叶结点确定类标号并停止构建
      2. 否则，确定并选择一个最佳的属性测试条件（包括熵、Gini系数、分类错误率等不纯程度度量方式）
      3. 将记录划分为较小的子集：对于测试条件的每个输出，创建一个子女节点，并根据测试结果将记录分布到子女结点中
      4. 递归调用此算法来构建子树
  - KNN：
    1. 计算每个测试样例与训练样例之间的距离
    2. 选择离测试样例最近的k个训练样例的集合
    3. 用多数表决或加权算法确定测试样例所属类别
  - 朴素贝叶斯：
    1. 计算先验概率P(类=...)
    2. 计算条件概率P(属性=...|类=...)
    3. 根据贝叶斯定理计算后验概率P(类=...|属性=...)
    4. 选择分类依据：选取后验概率最大的分类情况
- 优缺点：

算法	优点	缺点
朴素贝叶斯	所需估计的参数少；对于缺失数据不敏感	假设属性之间相互独立；需要知道先验概率；分类决策错误率高
决策树	不需要参数假设；适合高维数据；简单易于理解；能获得较好的性能	容易产生过拟合；忽略了属性之间的相关性；不支持在线学习
支持向量机(SVM)	可以解决小样本下机器学习的问题；可解决高维、非线性问题；可避免神经网络结构选择和局部极小的问题	缺失数据敏感；内存消耗大，难以解释；运行和调参复杂
KNN	速度极快，近似于线性；结果可解释性高	对于参数极为敏感；准确性低
神经网络	分类准确率高；并行处理能力强；分布式存储和学习能力强；鲁棒性较高	参数多；模型解释困难；训练时间过长
深度学习	分类准确率高；并行处理能力强；分布式存储和学习能力强	参数多；样本需求大；训练时间过长

    - 朴素贝叶斯优缺点完整版：

      - 关于优点中第三点敏感性的补充：面对孤立的噪声点、无关属性
    - 决策树优缺点完整版：
      - 优点：
        - 适合高维数据，可处理不相关的属性
        - 性能较好，分类速度快&构建成本低
        - 可解释性高→简单（不需要参数假设）&易于理解
        - 抗噪性强（尤其是在采用避免过度拟合的方法时），冗余属性不会对准确率造成不利的影响
      - 缺点：
        - 问题：不支持在线学习→不能处理非线性问题→解空间大→容易产生过拟合
        - 每个决策边界只涉及一个属性；交互属性（特征之间的相关性）可能会被忽略

- 时空复杂度：
    - 朴素贝叶斯&决策树：不考~（决策树时间复杂度为O(m*logm&k)，参数含义见下）
    - KNN：都为O(m*k)，其中m为样本数量，k为单个样本特征的维度

①基于规则的分类器

2023年10月25日 10:58

略~

②最近邻（KNN）分类器

2023年10月25日　11:42

- 积极（决策树、基于规则的分类器）&消极（Rote）学习方法（KNN）：

图 4-3 中显示的分类框架包括两个步骤：(1)归纳步，由训练数据建立分类模型；(2)演绎步，把模型应用于测试样例。决策树和基于规则的分类器是积极学习方法（eager learner）的例子，因为如果训练数据可用，它们就开始学习从输入属性到类标号的映射模型。与之相反的策略是推迟对训练数据的建模，直到需要分类测试样例时再进行。采用这种策略的技术被称为消极学习方法（lazy learner）。消极学习的一个例子是 Rote 分类器（Rote classifier），它记住整个训练数据，仅当测试实例的属性和某个训练样例完全匹配时才进行分类。该方法一个明显的缺点是有些测试记录不能被分类，因为没有任何训练样例与它们相匹配。

算法

算法 5.2　k-最近邻分类算法

1：令 k 是最近邻数目，D 是训练样例的集合
2：**for** 每个测试样例 $z=(\mathbf{x}', y')$ **do**
3：　计算 z 和每个样例 $(\mathbf{x}, y) \in D$ 之间的距离 $d(\mathbf{x}', \mathbf{x})$
4：　选择离 z 最近的 k 个训练样例的集合 $D_z \subseteq D$
5：　$y' = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} I(v = y_i)$
6：**end for**

其中，v 是类标号，y_i 是一个最近邻的类标号，$I(.)$是指示函数，如果其参数为真，则返回 1，否则，返回 0。

- 归纳总结：
  1. 计算每个测试样例与训练样例之间的距离
  2. 选择离测试样例最近的k个训练样例的集合
  3. 用多数表决或加权算法确定测试样例所属类别
- 加权算法：

在多数表决方法中，每个近邻对分类的影响都一样，这使得算法对 k 的选择很敏感，如图 5-7 所示。降低 k 的影响的一种途径就是根据每个最近邻 $\mathbf{x}_i$ 距离的不同对其作用加权：$w_i=1/d(\mathbf{x}', \mathbf{x}_i)^2$。结果使得远离 z 的训练样例对分类的影响要比那些靠近 z 的训练样例弱一些。使用距离加权表决方案，类标号可以由下面的公式确定：

$$\text{距离加权表决：}\quad y' = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} w_i \times I(v = y_i) \qquad (5\text{-}8)$$

- 特点：
  - 最近邻分类属于一类更广泛的技术，这种技术称为基于实例的学习，它使用具体的训练实例进行预测，而不必维护源自数据的抽象（或模型）。基于实例的学习算法需要邻近性度量来确定实例间的相似性或距离，还需要分类函数根据测试实例与其他实例的邻近性返回测试实例的预测类标号。
  - 像最近邻分类器这样的消极学习方法不需要建立模型，然而，分类测试样例的开销很大，因为需要逐个计算测试样例和训练样例之间的相似度。相反，积极学习方法通常花费大量计算资源来建立模型，模型一旦建立，分类测试样例就会非常快。
  - 最近邻分类器基于局部信息进行预测，而决策树和基于规则的分类器则试图找到一个拟合整个输入空间的全局模型。正是因为这样的局部分类决策，最近邻分类器（k 很小时）对噪声非常敏感。
  - 最近邻分类器可以生成任意形状的决策边界，这样的决策边界与决策树和基于规则的分类器通常所局限的直线决策边界相比，能提供更加灵活的模型表示。最近邻分类器的决策边界还有很高的可变性，因为它们依赖于训练样例的组合。增加最近邻的数目可以降低这种可变性。
  - 除非采用适当的邻近性度量和数据预处理，否则最近邻分类器可能做出错误的预测。例如，我们想根据身高（以米为单位）和体重（以磅为单位）等属性来对一群人分类。属性高度的可变性很小，从 1.5 米到 1.85 米，而体重范围则可能是从 90 磅到 250 磅。如果不考虑属性值的单位，那么邻近性度量可能就会被人的体重差异所左右。

③贝叶斯分类器

2023年10月25日　　11:09

贝叶斯定理

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

贝叶斯定理在分类中的应用

在描述贝叶斯定理怎样应用于分类之前，我们先从统计学的角度对分类问题加以形式化。设 $\mathbf{X}$ 表示属性集，Y 表示类变量。如果类变量和属性之间的关系不确定，那么我们可以把 $\mathbf{X}$ 和 Y 看作随机变量，用 $P(Y|\mathbf{X})$以概率的方式捕捉二者之间的关系。这个条件概率又称为 Y 的后验概率（posterior probability），与之相对地，$P(Y)$称为 Y 的先验概率（prior probability）。

准确估计类标号和属性值的每一种可能组合的后验概率非常困难，因为即便属性数目不是很大，仍然需要很大的训练集。此时，贝叶斯定理很有用，因为它允许我们用先验概率 $P(Y)$、类条件（class-conditional）概率 $P(\mathbf{X}|Y)$和证据 $P(\mathbf{X})$来表示后验概率：

$$P(Y|\mathbf{X})=\frac{P(\mathbf{X}|Y)P(Y)}{P(\mathbf{X})} \qquad (5\text{-}11)$$

朴素贝叶斯分类器

分类测试记录时，朴素贝叶斯分类器对每个类 Y 计算后验概率：

$$P(Y|\mathbf{X})=\frac{P(Y)\prod_{i=1}^{d}P(X_i|Y)}{P(\mathbf{X})} \qquad (5\text{-}15)$$

由于对所有的 Y，$P(\mathbf{X})$是固定的，因此只要找出使分子 $P(Y)\prod_{i=1}^{d}P(X_i|Y)$ 最大的类就是够了。在接

- 工作过程：
  1. 计算先验概率P(类=...)
  2. 计算条件概率P(属性=...|类=...)
  3. 根据贝叶斯定理计算后验概率P(类=...|属性=...)
  4. 选择分类依据：选取后验概率最大的分类情况
- *概率估计：

than simple fractions

- Probability estimation:

original: $P(X_i=c|y)=\frac{n_c}{n}$

Laplace Estimate: $P(X_i=c|y)=\frac{n_c+1}{n+v}$

m − estimate: $P(X_i=c|y)=\frac{n_c+mp}{n+m}$

n: number of training instances belonging to class y

n_c: number of instances with $X_i = c$ and $Y = y$

v: total number of attribute values that X_i can take

p: initial estimate of $\hat{P}$ ($P(X_i = c|y)$ known apriori

m: hyper-parameter for our confidence in p

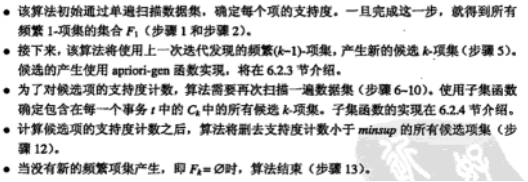
2/08/2021　Introduction to Data Mining, 2nd Edition

- 规范-估计条件概率P(A|+/-)：题目没有出错，而是需要分别算出A的所有取值对应的条件概率！

0归纳总结

2024年1月3日 20:05

- 含义：发现隐藏在大型数据集中的有意义的联系
- 应用：网页挖掘&科学数据分析、生物信息学&医疗诊断
- 过程：
  - Apriori算法的频繁项集产生



- 该算法初始通过单遍扫描数据集，确定每个项的支持度。一旦完成这一步，就得到所有频繁 1-项集的集合 F_1（步骤 1 和步骤 2）。
- 接下来，该算法将使用上一次迭代发现的频繁$(k-1)$-项集，产生新的候选 k-项集（步骤 5）。候选的产生使用 apriori-gen 函数实现，将在 6.2.3 节介绍。
- 为了对候选项的支持度计数，算法需要再次扫描一遍数据集（步骤 6–10）。使用子集函数确定包含在每一个事务 t 中的 C_k 中的所有候选 k-项集。子集函数的实现在 6.2.4 节介绍。
- 计算候选项的支持度计数之后，算法将删去支持度计数小于 *minsup* 的所有候选项集（步骤 12）。
- 当没有新的频繁项集产生，即 $F_k = \varnothing$时，算法结束（步骤 13）。

    - 补充：重复步骤2~5
  - Apriori算法中规则的产生：
    1. 对每个频繁项集，若规则后件少于频繁项集大小，则两两合并当前规则后件，从而产生新的候选规则，否则停止
    2. 判断新的候选规则置信度是否达到阈值，若否则删去包含此候选规则对于结点的整个子图
    3. 递归调用此算法
- 支持度-置信度框架的局限性：
  - 在除去没有意义的模式时，支持度的缺点在于许多潜在的有意义的模式由于包含支持度小的项而被删去
  - 置信度的缺陷在于该度量忽略了规则后件中项集的支持度

①问题定义

2023年11月1日　10:38

- 概念含义：
  - 关联分析：发现隐藏在大型数据集中的有意义的联系
  - 项：数据集中的一个元素
  - 项集：包含0个或多个项的集合
  - 事务：交易的项的集合
  - 支持度计数：包含特定项集的事务个数
- 应用：网页挖掘&科学数据分析、生物信息学&医疗诊断

项集&支持度计数

项集和支持度计数　令 $I=\{i_1, i_2, \cdots, i_d\}$ 是购物篮数据中所有项的集合，而 $T=\{t_1, t_2, \cdots, t_N\}$ 是所有事务的集合。每个事务 t_i 包含的项集都是 I 的子集。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，{啤酒，尿布，牛奶}是一个 3-项集。空集是指不包含任何项的项集。

事务的宽度定义为事务中出现项的个数。如果项集 X 是事务 t_j 的子集，则称事务 t_j 包括项集 X。例如，在表 6-2 中第二个事务包括项集{面包，尿布}，但不包括项集{面包，牛奶}。项集的一个重要性质是它的支持度计数，即包含特定项集的事务个数。数学上，项集 X 的支持度计数 $\sigma(X)$ 可以表示为：

$$\sigma(X)=|\{t_i \mid X \subseteq t_i, t_i \in T\}|$$

其中，符号 $|\cdot|$ 表示集合中元素的个数。在表 6-2 显示的数据集中，项集{啤酒，尿布，牛奶}的支持度计数为 2，因为只有 2 个事务同时包含这 3 个项。

关联规则

关联规则（association rule）　关联规则是形如 $X \rightarrow Y$ 的蕴涵表达式，其中 X 和 Y 是不相交的项集，即 $X \cap Y=\varnothing$。关联规则的强度可以用它的**支持度**（support）和**置信度**（confidence）度量。支持度确定规则可以用于给定数据集的频繁程度，而置信度确定 Y 在包含 X 的事务中出现的频繁程度。支持度（s）和置信度（c）这两种度量的形式定义如下：

$$s(X \rightarrow Y)=\frac{\sigma(X \cup Y)}{N} \quad (6\text{-}1)$$

$$c(X \rightarrow Y)=\frac{\sigma(X \cup Y)}{\sigma(X)} \quad (6\text{-}2)$$

- 两步法：根据支持度产生频繁项集→根据置信度生成规则
- 频繁项集定义：满足最小支持度阈值的所有项集

②频繁项集的产生

2023年11月1日　11:23

先验原理

- 原理内容：如果一个项集是频繁的，则它的所有子集一定也是频繁的
- 基于支持度的剪枝-利用支持度度量的反单调性：一个项集的支持度决不会超过它的子集的支持度。即如果一个项集是频繁（达到支持度）的，则它的所有子集一定也是频繁的；相反，如果项集是非频繁的，则它的所有超集也一定是非频繁的

Apriori算法的频繁项集产生

算法伪代码：

- 该算法初始通过单遍扫描数据集，确定每个项的支持度。一旦完成这一步，就得到所有频繁 1-项集的集合 F_1（步骤 1 和步骤 2）。
- 接下来，该算法将使用上一次迭代发现的频繁(k−1)-项集，产生新的候选 k-项集（步骤 5）。候选的产生使用 apriori-gen 函数实现，将在 6.2.3 节介绍。
- 为了对候选项的支持度计数，算法需要再次扫描一遍数据集（步骤 6–10）。使用子集函数确定包含在每一个事务 t 中的 C_k 中的所有候选 k-项集。子集函数的实现在 6.2.4 节介绍。
- 计算候选项的支持度计数之后，算法将删去支持度计数小于 *minsup* 的所有候选项集（步骤 12）。
- 当没有新的频繁项集产生，即 $F_k = \varnothing$ 时，算法结束（步骤 13）。

- 补充：重复步骤2~5

候选的产生&剪枝

- 要求：

算法 6.1 步骤 5 的 apriori-gen 函数通过如下两个操作产生候选项集。

(1) 候选项集的产生。该操作由前一次迭代发现的频繁(k−1)-项集产生新的候选 k-项集。

(2) 候选项集的剪枝。该操作采用基于支持度的剪枝策略，删除一些候选 k-项集。

为了解释候选项集剪枝操作，考虑候选 k-项集 $X = \{i_1, i_2, \cdots, i_k\}$。算法必须确定它的所有真子集 $X - \{i_j\}(\forall j = 1, 2, \cdots, k)$是否都是频繁的，如果其中一个是非频繁的，则 X 将会被立即剪枝。这种方法能够有效地减少支持度计数过程中所考虑的候选项集的数量。对于每一个候选 k-项集，该操作的复杂度是 $O(k)$。然而，随后我们将明白，并不需要检查给定候选项集的所有 k 个子集。如果 k 个子集中的 m 个用来产生候选项集，则在候选项集剪枝时只需要检查剩下的 k−m 个子集。

理论上，存在许多产生候选项集的方法。下面列出了对有效的候选项集产生过程的要求。

(1) 它应当避免产生太多不必要的候选。一个候选项集是不必要的，如果它至少有一个子集是非频繁的。根据支持度的反单调属性，这样的候选项集肯定是非频繁的。

(2) 它必须确保候选项集的集合是完全的，即候选项集产生过程没有遗漏任何频繁项集。为了确保完全性，候选项集的集合必须包含所有频繁项集的集合，即$\forall k: F_k \subseteq C_k$。

(3) 它应该不会产生重复候选项集。例如：候选项集$\{a, b, c, d\}$可能会通过多种方法产生，如合并$\{a, b, c\}$和$\{d\}$，合并$\{b, d\}$和$\{a, c\}$，合并$\{c\}$和$\{a, b, d\}$等。候选项集的重复产生将会导致计算的浪费，因此为了效率应该避免。

接下来，将简要地介绍几种候选产生过程，其中包括 apriori-gen 函数使用的方法。

  - 归纳总结：完全，重复→不必要

- 常用方法：
  - 蛮力方法
  - Fk-1*F1方法
  - Fk-1*Fk-1方法：

$F_{k-1} \times F_{k-1}$ 方法　函数 apriori-gen 的候选产生过程合并一对频繁(k−1)-项集，仅当它们的前 k−2 个项都相同。令 $A = \{a_1, a_2, \cdots, a_{k-1}\}$和 $B = \{b_1, b_2, \cdots, b_{k-1}\}$是一对频繁(k−1)-项集，合并 A 和 B，如果它们满足如下条件：

$$a_i = b_i \quad (i = 1, 2, \cdots, k-2) \text{ 并且 } a_{k-1} \neq b_{k-1}$$

在图 6-8 中，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。实际上，如果{啤酒，尿布，牛奶}是可行的候选，则它应当由{啤酒，尿布}和{啤酒，牛奶}合并而得到。这个例子表明了候选项产生过程的完全性和使用字典序避免重复的候选的优点。然而，由于每个候选都由一对频繁(k−1)-项集合并而成，因此需要附加的候选剪枝步骤来确保该候选的其余k−2个子集是频繁的。

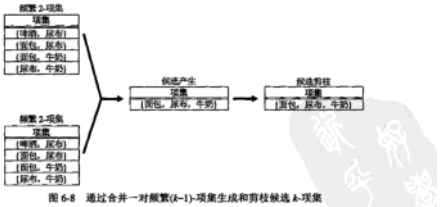


图 6-8　通过合并一对频繁(k−1)-项集生成和剪枝候选 k-项集

支持度计数

支持度计数过程确定在apriori-gen函数的候选项剪枝步骤保留下来的每个候选项集出现的频繁程度。支持度计数在算法 6.1 的第 6 步到第 11 步实现。支持度计数的一种方法是，将每个事

务与所有的候选项集进行比较（见图 6-2）。并且更新包含在事务中的候选项集的支持度计数。这种方法是计算昂贵的，尤其当事务和候选项集的数目都很大时。

另一种方法是枚举每个事务所包含的项集，并且利用它们更新对应的候选项集的支持度。例如，考虑事务 t_1。它包含 5 个项{1, 2, 3, 5, 6}。该事务包含 $C_5^3=10$ 个大小为 3 的项集，其中的某些项集可能对应于所考察的候选 3-项集。在这种情况下，增加它们的支持度。那些不与任何候选项集对应的事务 t 的子集可以忽略。

图 6-9 显示了枚举事务 t 中所有 3-项集的系统的方法。假定每个项集中的项都以递增的字典序排列。则项集可以这样枚举：先指定最小项，其后跟随较大的项。例如，给定 $t=\{1, 2, 3, 5, 6\}$，它的所有 3-项集一定以项 1、2 或 3 开始。不必构造以 5 或 6 开始的 3-项集，因为事务 t 中只有两个项的标号大于等于 5。图 6-9 中第一层的前缀结构描述了指定包含在事务 t 中的 3-项集的第一项的方法。例如，1 2 3 5 6 表示这样的 3-项集，它以 1 开始，后随两个取自集合{2, 3, 5, 6}的项。

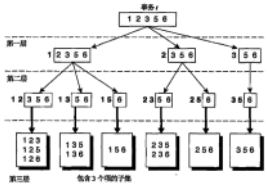


图 6-9　枚举事务 t 的所有包含 3 个项的子集

确定了第一项之后，第二层的前缀结构表示选择第二项的方法。例如：1 2 3 5 6表示以{1,2}为前缀、后随项 3、5 或 6 的项集。最后，第三层的前缀结构显示了事务 t 包含的所有的 3-项集。例如，以{1, 2}为前缀的 3-项集是{1, 2, 3}，{1, 2, 5}和{1, 2, 6}；而以{2, 3}为前缀的 3-项集是{2, 3, 5}和{2, 3, 6}。

图 6-9 中所示的前缀结构展示了如何系统地枚举事务所包含的项集，即通过从最左项到最右项依次指定项集的项。然而还必须确定每一个枚举的 3-项集是否对应于一个候选项集，如果它与一个候选匹配，则相应候选项集的支持度计数增值。下面，将解释如何使用 Hash 树来有效地进行匹配操作。

- 使用Hash树进行支持度计数：

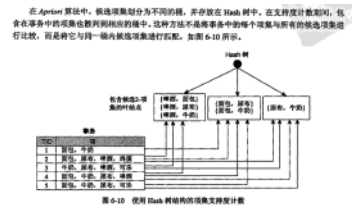


图 6-10　使用 Hash 树结构的项集支持度计数

图6-11是一棵 Hash 树结构的例子。树的每个内部结点都使用 Hash 函数 $h(p)=p \bmod 3$来确定应当沿着当前结点的哪个分支向下。例如，项1、4和7应当散列到相同的分支（即最左分支）。因为除以3之后它们都具有相同的余数。所有的候选项集都存放在 Hash 树的叶结点中。图6-11中显示的 Hash 树包含15个候选3-项集，分布在9个叶结点中。

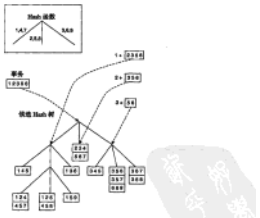


图 6-11　在 Hash 树的根结点散列一个事务

考虑一个事务 $t=\{1, 2, 3, 5, 6\}$。为了更新候选项集的支持度计数，必须这样遍历 Hash 树，所有包含属于事务 t 的候选 3-项集的叶结点至少访问一次。注意，包含在 t 中的候选 3-项集必须以项 1、2 或 3 开始，如图 6-9 中第一层前缀结构所示。这样，在 Hash 树的根结点，事务中的项 1、2 和 3 将分别散列。项 1 被散列到根结点的左子女，项 2 被散列到中间子女，而项 3 被散列到右子女。在树的下一层，事务根据图 6-9 中的第二层结构所出的第二项进行散列。例如，在根结点散列项 1 之后，散列事务的项 2、3 和 5。项 2 和 5 散列到中间子女，而 3 散列到右子女，如图 6-12 所示。继续该过程，直至到达 Hash 树的叶结点。存放在被访问的叶结点中的候选项集与事务进行比较，如果候选项集是该事务的子集，则增加它的支持度计数。在这个例子中，访问了 9 个叶结点中的 5 个，15 个项集中的 9 个与事务进行比较。

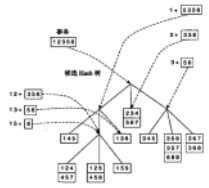


图 6-12　在候选项集 Hash 树根的最左子树上的子集操作

- （补充）影响 Apriori 复杂性的因素：
  - 项数（维度），事务数&事务的平均宽度
  - 支持度阈值

③规则产生

2023年11月8日 11:15

本节介绍如何有效地从给定的频繁项集中提取关联规则。忽略那些前件或后件为空的规则（$\varnothing \rightarrow Y$ 或 $Y \rightarrow \varnothing$），每个频繁 k-项集能够产生多达 2^k-2 个关联规则。关联规则可以这样提取：将项集 Y 划分成两个非空的子集 X 和 $Y-X$，使得 $X \rightarrow Y-X$ 满足置信度阈值。注意，这样的规则必然已经满足支持度阈值，因为它们是由频繁项集产生的。

计算关联规则的置信度并不需要再次扫描事务数据集。考虑规则$\{1,2\} \rightarrow \{3\}$，它是由频繁项集 $X=\{1,2,3\}$产生的。该规则的置信度为$\sigma(\{1,2,3\})/\sigma(\{1,2\})$。因为$\{1,2,3\}$是频繁的，支持度的反单调性确保项集$\{1,2\}$一定也是频繁的。由于这两个项集的支持度计数已经在频繁项集产生时得到，因此不必再扫描整个数据集。

- 强关联规则：s(X→Y)≥s_minsup且c(X→Y)≥minconf

基于置信度的剪枝

不像支持度度量，置信度不具有任何单调性。例如：规则 $X \rightarrow Y$ 的置信度可能大于、小于或等于规则 $\tilde{X} \rightarrow \tilde{Y}$ 的置信度，其中 $\tilde{X} \subset X$ 且 $\tilde{Y} \subset Y$（见本章习题 3）。尽管如此，当比较由频繁项集 Y 产生的规则时，下面的定理对置信度度量成立。

定理 6.2 *如果规则 $X \rightarrow Y-X$ 不满足置信度阈值，则形如 $X' \rightarrow Y-X'$ 的规则一定也不满足置信度阈值，其中 X' 是 X 的子集。*

Apriori算法中规则的产生

*Apriori*算法使用一种逐层方法来产生关联规则，其中每层对应于规则后件中的项数。初始，提取规则后件只含一个项的所有高置信度规则，然后，使用这些规则来产生新的候选规则。例如，如果$\{acd\} \rightarrow \{b\}$和$\{abd\} \rightarrow \{c\}$是两个高置信度的规则，则通过合并这两个规则的后件产生候选规则$\{ad\} \rightarrow \{bc\}$。图 6-15 显示了由频繁项集$\{a, b, c, d\}$产生关联规则的格结构。如果格中的任意结点具有低置信度，则根据定理 6.2，可以立即剪掉该结点生成的整个子图。假设规则$\{bcd\} \rightarrow \{a\}$具有低置信度，则可以丢弃后件包含$a$的所有规则，包括$\{d\} \rightarrow \{abc\}$，$\{bd\} \rightarrow \{ac\}$，$\{bc\} \rightarrow \{ad\}$和$\{d\} \rightarrow \{abc\}$。

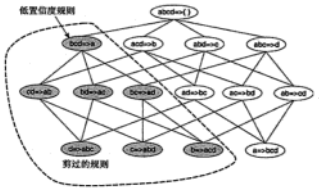


图 6-15 使用置信度度量对关联规则进行剪枝

算法 6.2 和算法 6.3 给出了关联规则产生的伪代码。注意，算法 6.3 中的 ap-genrules 过程与算法 6.1 中的频繁项集产生的过程类似。二者唯一不同是，在规则产生时，不必再次扫描数据集来计算候选规则的置信度，而是使用在频繁项集产生时计算的支持度计数来确定每个规则的置信度。

算法 6.2 *Apriori* 算法中的规则产生

```
1: for 每一个频繁 k-项集 f_k，k≥2 do
2:   H_1={i | i∈f_k}   {规则的 1-项后件}
3:   call ap-genrules(f_k, H_1)
4: end for
```

算法 6.3 过程 ap-genrules(f_k, H_m)

```
1: k=|f_k|   {频繁项集的大小}
2: m=|H_m|   {规则后件的大小}
3: If k>m+1 then
4:   H_{m+1}=apriori-gen(H_m)
5:   for 每个 h_{m+1}∈H_{m+1} do
6:     conf=σ(f_k)/σ(f_k−h_{m+1})
7:     if conf≥minconf then
8:       output: 规则 (f_k−h_{m+1}) → h_{m+1}
9:     else
10:      从 H_{m+1} delete h_{m+1}
11:    end if
12:  end for
13:  call ap-genrules(f_k, H_{m+1})
14: end if
```

- 归纳总结-算法步骤：
  1. 对每个频繁项集，若规则后件少于频繁项集大小，则两两合并当前规则，从而产生新的候选规则，否则停止
  2. 判断新的候选规则置信度是否达到阈值，若否则删去包含次候选规则对于结点的整个子图
  3. 递归调用此算法
- 支持度-置信度框架的局限性：
  - 在除去没有意义的模式时，支持度的缺点在于许多潜在的有意义的模式由于包含支持度小的项而被删去
  - 置信度的缺陷在于该度量忽略了规则后件中项集的支持度

④频繁项集的紧凑表示

2023年11月8日 11:44

极大频繁项集

直接超集都不是频繁的的频繁项集

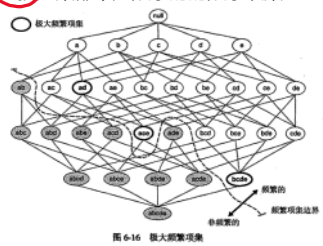
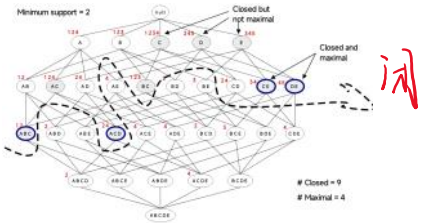


图 6-16 极大频繁项集

- 极大频繁项集有效地提供了频繁项集的紧凑表示
- 极大频繁项集不包含它们子集的支持度信息

闭频繁项集

- 闭项集：直接超集都不具有和它相同的支持度计数的项集
- 闭频繁项集：支持度大于等于最小支持度阈值（即频繁）的闭项集（如果已经陈述了频繁项集的定义，则可以改为直接超集都不具有和它相同的支持度计数的频繁项集）



  - maximal指支持度刚好为阈值
- 包含关系：频繁项集>>闭频繁项集>>极大频繁项集

⑤产生频繁项集的其他方法

2023年11月8日 11:45

都没讲~

- 项集格遍历：一般到特殊&特殊到一般、等价类、宽度优先&深度优先

⑥FP增长算法

2023年11月8日　　11:47

用于发现频繁项集

FP树表示法

- 方法步骤：

(1) 扫描一次数据集，确定每个项的支持度计数。丢弃非频繁项，而将频繁项按照支持度的递减排序。对于图 6-24 中的数据集，a 是最频繁的项，接下来依次是 b, c, d 和 e。

(2) 算法第二次扫描数据集，构建 FP 树。读入第一个事务$\{a, b\}$之后，创建标记为 a 和 b 的结点。然后形成 null→a→b 路径，对该事务编码。该路径上的所有结点的频度计数为 1。

(3) 读入第二个事务$\{b, c, d\}$之后，为项 b, c 和 d 创建新的结点集。然后，连接结点 null→b→c→d，形成一条代表该事务的路径。该路径上的每个结点的频度计数也等于 1。尽管前两个事务具有一个共同项 b，但是它们的路径不相交，因为这两个事务没有共同的前缀。

(4) 第三个事务$\{a, c, d, e\}$与第一个事务共享一个共同前缀项 a，所以第三个事务的路径 null→a→c→d→e 与第一个事务的路径 null→a→b 部分重叠。因为它们的路径重叠，所以结点 a 的频度计数增加为 2，而新创建的结点 c, d 和 e 的频度计数等于 1。

(5) 继续该过程，直到每个事务都映射到 FP 树的一条路径。读入所有的事务后形成的 FP 树显示在图 6-24 的底部。

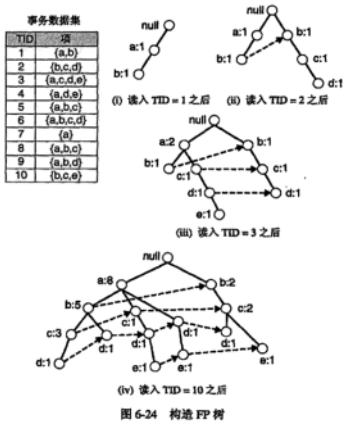


图 6-24　构造 FP 树

- 分析：

通常，FP树的大小比未压缩的数据小，因为购物篮数据的事务常常共享一些共同项。在最好情况下，所有的事务都具有相同的项集，FP树只包含一条结点路径。当每个事务都具有唯一项集时，导致最坏情况发生，由于事务不包含任何共同项，FP树的大小实际上与原数据的大小一样，然而，由于需要附加的空间为每个项存放结点间的指针和计数，FP树的存储需求增大。

FP 树的大小也取决于项的排序方式。如果颠倒前面例子的序，即项按照支持度由小到大排列，则结果 FP 树显示在图 6-25 中。该树显得更加茂盛，因为根结点上的分支数由 2 增加到 5，并且包含了高支持度项 a 和 b 的结点数由 3 增加到 12。尽管如此，支持度计数递减序并非总是导致最小的树。例如，假设加大图 6-24 给定的数据集，增加 100 个事务包含$\{e\}$、80 个事务包含$\{d\}$、60 个事务包含$\{c\}$、40 个事务包含$\{b\}$。现在，项 e 是最频繁的，接下来依次是 d, c, b 和 a。使用加大的事务数据，支持度计数递减序将导致类似于图 6-25 中的 FP 树，而基于支持度计数递增序将产生一棵类似于图 6-24(iv)的较小的 FP 树。

FP增长算法的频繁项集的产生

FP 增长（FP-growth）是一种以自底向上方式探索树，由 FP 树产生频繁项集的算法。给定图 6-24 所示的树，算法首先查找以 e 结尾的频繁项集，接下来依次是 d, c, b，最后是 a。这种用于发现以某一个特定项结尾的频繁项集的自底向上策略等价于 6.5 节介绍的基于后缀的方法。由于每一个事务都映射到 FP 树中的一条路径，因而通过仅考察包含特定结点（例如 e）的路径，就可以发现以 e 结尾的频繁项集。使用与结点 e 相关联的指针，可以快速访问这些路径。图 6-26a 显示了所提取的路径。稍后详细解释如何处理这些路径，以得到频繁项集。

FP增长采用分治策略将一个问题分解为较小的子问题，从而发现以某个特定后缀结尾的所有频繁项集。例如，假设对发现所有以e结尾的频繁项集感兴趣。为了实现这个目的，必须首先检查项集$\{e\}$本身是否频繁。如果它是频繁的，则考虑发现以de结尾的频繁项集子问题，接下来是ce和ae。依次，每一个子问题可以进一步划分为更小的子问题。通过合并这些子问题得到的结果，就可以找到所有以e结尾的频繁项集。这种分治策略是FP增长算法采用的关键策略。

为了更具体地说明如何解决这些子问题，考虑发现所有以 e 结尾的频繁项集的任务。

(1) 第一步收集包含 e 结点的所有路径。这些初始的路径称为前缀路径（prefix path），如图 6-27a 所示。

(2) 由图 6-27a 中所显示的前缀路径，通过把与结点 e 相关联的支持度计数相加得到 e 的支持度计数。假定最小支持度为 2，因为$\{e\}$的支持度是 3 所以它是频繁项集。

(3) 由于$\{e\}$是频繁的，因此算法必须解决发现以 de、ce、be 和 ae 结尾的频繁项集的子问题。在解决这些子问题之前，必须先将前缀路径转化为条件 FP 树（conditional FP-tree）。除了用于发现以某特定后缀结尾的频繁项集之外，条件 FP 树的结构与 FP 树类似。条件 FP 树通过以下步骤得到。

(a) 首先，必须更新前缀路径上的支持度计数，因为某些计数包括那些不含项 e 的事务。例如，图 6-27a 中的最右边路径 null→b:2→c:2→e:1，包括并不含项 e 的事务$\{b, c\}$。因此，必须将该前缀路径上的计数调整为 1，以反映包含$\{b, c, e\}$的事务的实际个数。

(b) 删除 e 的结点，修剪前缀路径。删除这些结点是因为，沿这些前缀路径的支持度计数已经更新，以反映包含 e 的那些事务，并且发现以 de, ce, be 和 ae 结尾的频繁项集的子问题不再需要结点 e 的信息。

(c) 更新沿前缀路径上的支持度计数之后，某些项可能不再是频繁的。例如，结点 b 只出现了 1 次，它的支持度计数等于 1，这就意味着只有一个事务同时包含 b 和 e。因为所有以 be 结尾的项集一定都是非频繁的，所以在其后的分析中可以安全地忽略 b。

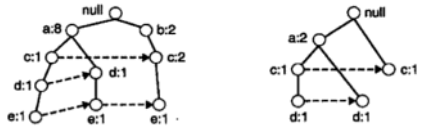
如，图 6-27a 中的最右边路径 null→*b*:2→*c*:2→*e*:1，包括并不含项 *e* 的事务{*b*, *c*}。因此，必须将该前缀路径上的计数调整为 1，以反映包含{*b*, *c*, *e*}的事务的实际个数。

(b) 删除 *e* 的结点，修剪前缀路径。删除这些结点是因为，沿这些前缀路径的支持度计数已经更新，以反映包含 *e* 的那些事务，并且发现以 *de*, *ce*, *be* 和 *ae* 结尾的频繁项集的子问题不再需要结点 *e* 的信息。

(c) 更新沿前缀路径上的支持度计数之后，某些项可能不再是频繁的。例如，结点 *b* 只出现了 1 次，它的支持度计数等于 1，这就意味着只有一个事务同时包含 *b* 和 *e*。因为所有以 *be* 结尾的项集一定都是非频繁的，所以在其后的分析中可以安全地忽略 *b*。

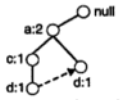
e 的条件 FP 树显示在图 6-27b 中。该树看上去与原来的前缀路径不同，因为频度计数已经更新，并且结点 *b* 和 *e* 已被删除。

(4) FP 增长使用 *e* 的条件 FP 树来解决发现以 *de*, *ce*, *be* 和 *ae* 结尾的频繁项集的子问题。为了发现以 *de* 结尾的频繁项集，从项 *e* 的条件 FP 树收集 *d* 的所有前缀路径（图 6-27c）。通过将与结点 *d* 相关联的频度计数求和，得到项集{*d*, *e*}的支持度计数。因为项集{*d*, *e*}的支持度计数等于 2，所以它是频繁项集。接下来，算法采用第 3 步介绍的方法构建 *de* 的条件 FP 树。更新了支持度计数并删除了非频繁项 *c* 之后，*de* 的条件 FP 树显示在图 6-27d 中。因为该条件 FP 树只包含一个支持度等于最小支持度的项 *a*，算法提取出频繁项集{*a*, *d*, *e*}并转到下一个子问题，产生以 *ce* 结尾的频繁项集。处理 *c* 的前缀路径后，只发现项集{*c*, *e*}是频繁的。接下来，算法继续解决下一个子问题并发现项集{*a*, *e*}是剩下唯一的频繁项集。



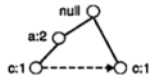
(a) 以 *e* 结尾的前缀路径

(b) *e* 的条件 FP 树



(c) 以 *de* 结尾的前缀路径

(d) *de* 的条件 FP 树



(e) 以 *ce* 结尾的前缀路径

(f) *ae* 的条件 FP 树

图 6-27　使用 FP 增长算法发现以 *e* 结尾的频繁项集的例子

- 更新结点计数&删除未达到支持度计数阈值的结点：前缀路径都不做，条件FP树再都做
- 优点：
  - 避免多次扫码数据集，提高效率
  - 将输入数据压缩表示，节约存储空间

⑦关联模式的评估

2023年11月8日　11:56

- 支持度、置信度
- 主观信息：可视化、基于模板的方法、主观兴趣度度量

兴趣度的客观度量

- 2路相依表：

	B	$\bar{B}$	
A	f_{11}	f_{10}	f_{1+}
$\bar{A}$	f_{01}	f_{00}	f_{0+}
	f_{+1}	f_{+0}	N

- 提升度（lift）&兴趣因子：

兴趣因子　茶与咖啡的例子表明，由于置信度度量忽略了规则后件中出现的项集的支持度，高置信度的规则有时可能出现误导。解决这个问题的一种方法是使用称作提升度（lift）的度量：

$$lift(A \rightarrow B) = \frac{c(A \rightarrow B)}{s(B)} \tag{6-4}$$

它计算规则置信度和规则后件中项集的支持度之间的比率。对于二元变量，提升度等价于另一种称作兴趣因子（interest factor）的客观度量，其定义如下：

$$I(A,B) = \frac{s(A,B)}{s(A) \times s(B)} = \frac{Nf_{11}}{f_{1+}f_{+1}} \tag{6-5}$$

  - 意义：

$$I(A,B)\begin{cases} =1 & \text{如果 } A \text{ 和 } B \text{ 是独立的} \\ >1 & \text{如果 } A \text{ 和 } B \text{ 是正相关的} \\ <1 & \text{如果 } A \text{ 和 } B \text{ 是负相关的} \end{cases}$$

  - 具有一定的局限性

- 相关分析：

相关分析　相关分析是分析一对变量之间关系的基于统计学的技术。对于连续变量，相关度用皮尔森相关系数定义（参看 2.4.5 节公式（2-10））。对于二元变量，相关度可以用ϕ系数度量。ϕ系数定义如下：

$$\phi = \frac{f_{11}f_{00} - f_{01}f_{10}}{\sqrt{f_{1+}f_{+1}f_{0+}f_{+0}}} \tag{6-8}$$

相关度的值从−1（完全负相关）到+1（完全正相关）。如果变量是统计独立的，则$\phi=0$。例

  - 局限性：
    - 把项在事务中同时出现和不出现视为同等重要
    - 当样本大小成比例变化时，它不能够保持不变

- IS度量：

IS 度量　*IS* 是另一种度量，用于处理非对称二元变量。该度量定义如下：

$$IS(A,B) = \sqrt{I(A,B) \times s(A,B)} = \frac{s(A,B)}{\sqrt{s(A)s(B)}} \tag{6-9}$$

注意，当模式的兴趣因子和模式支持度都很大时，*IS* 也很大。例如，表 6-9 中显示的词对$\{p, q\}$

  - 局限性：与置信度度量类似，即使是不相关或负相关的模式，度量值也可能相当大

- *其他度量方式：

表 6-11　项集{A,B}的对称的客观度量

度量（符号）	定　义
相关性（ϕ）	$\frac{Nf_{11}-f_{1+}f_{+1}}{\sqrt{f_{1+}f_{+1}f_{0+}f_{+0}}}$
几率（α）	$(f_{11}f_{00})\ /\ (f_{10}f_{01})$
κ（k）	$\frac{Nf_{11}+Nf_{00}-f_{1+}f_{+1}-f_{0+}f_{+0}}{N^2-f_{1+}f_{+1}-f_{0+}f_{+0}}$
兴趣因子（I）	$(Nf_{11})\ /\ (f_{1+}f_{+1})$
余弦（IS）	$(f_{11})/\left(\sqrt{f_{1+}f_{+1}}\right)$
Piatetsky-Shapiro（PS）	$\frac{f_{11}}{N}-\frac{f_{1+}f_{+1}}{N^2}$
集体强度（S）	$\frac{f_{11}+f_{00}}{f_{1+}f_{+1}+f_{0+}f_{+0}}\times\frac{N-f_{1+}f_{+1}-f_{0+}f_{+0}}{N-f_{11}-f_{00}}$
Jaccard（ζ）	$f_{11}/\ (f_{1+}+f_{+1}-f_{11})$
全置信度（h）	$\min\left[\frac{f_{11}}{f_{1+}},\ \frac{f_{11}}{f_{+1}}\right]$

表 6-12　规则 $A\rightarrow B$ 的非对称的客观度量

度量（符号）	定　义
Goodman-Kruskal（λ）	$\left(\sum_j \max_k f_{jk}-\max_k f_{+k}\right)/\left(N-\max_k f_{+k}\right)$
互信息（M）	$\left(\sum_i\sum_j \frac{f_{ij}}{N}\log\frac{Nf_{ij}}{f_{i+}f_{+j}}\right)/\left(-\sum_i \frac{f_{i+}}{N}\log\frac{f_{i+}}{N}\right)$
J 度量（J）	$\frac{f_{11}}{N}\log\frac{Nf_{11}}{f_{1+}f_{+1}}+\frac{f_{10}}{N}\log\frac{Nf_{10}}{f_{1+}f_{+0}}$
Gini 指标（G）	$\frac{f_{1+}}{N}\times\left[\left(\frac{f_{11}}{f_{1+}}\right)^2+\left(\frac{f_{10}}{f_{1+}}\right)^2\right]-\left(\frac{f_{+1}}{N}\right)^2+\frac{f_{0+}}{N}\times\left[\left(\frac{f_{01}}{f_{0+}}\right)^2+\left(\frac{f_{00}}{f_{0+}}\right)^2\right]-\left(\frac{f_{+0}}{N}\right)^2$
拉普拉斯（L）	$(f_{11}+1)/(f_{1+}+2)$
信任度（V）	$(f_{1+}f_{+0})/(Nf_{10})$
可信度因子（F）	$\left(\frac{f_{11}}{f_{1+}}-\frac{f_{+1}}{N}\right)/\left(1-\frac{f_{+1}}{N}\right)$
Added Value（AV）	$\frac{f_{11}}{f_{1+}}-\frac{f_{+1}}{N}$

- 客观度量的一致性：没讲~
- 客观度量的性质：
  - 反演性：

    定义 6.6　反演性　客观度量 M 在反演操作下是不变的，如果交换频度计数 f_{11} 和 f_{00}、f_{10} 和 f_{01} 它的值保持不变。
  - 零加性：

    定义 6.7　零加性　客观度量 M 在零加操作下是不变的，如果增加 f_{00} 而保持相依表中所有其他的频度不变并不影响 M 的值。
  - 缩放性：

    定义 6.8　缩放不变性　客观度量 M 在行/列缩放操作下是不变的，如果 $M(T)=M(T')$，其中，T 是频度计数为$[f_{11};f_{10};f_{01};f_{00}]$的相依表，$T'$是频度计数为$[k_1k_3f_{11};k_2k_3f_{10};k_1k_4f_{01};k_2k_4f_{00}]$的相依表，而 k_1, k_2, k_3, k_4 是正常量。

多个二元变量的度量

支持度、全置信度、兴趣因子、IS、PS、Jaccard系数：没讲~

辛普森悖论

在某些情况下，隐藏的变量可能会导致观察到的一堆变量之间的联系消失或逆转方向的现象

⑧倾斜支持度分布的影响

2023年11月8日 20:34

没讲~

0归纳总结

2023年12月22日　11:12

- 应用：
  - 旨在理解的聚类：气候、生物学、心理学&医学、信息检索、商业
  - 旨在实用的聚类：汇总、压缩、有效地发现最近邻
- 过程：
  - [二分]K均值：
    - K均值：

1：选择 K 个点作为初始质心。
2：**repeat**
3：　将每个点指派到最近的质心，形成 K 个簇。
4：　重新计算每个簇的质心。
5：**until** 质心不发生变化。

    - 二分K均值：
      1. 初始化簇表，使之包含由所有点组成的簇
      2. 从簇表中依次取出每一个簇，分别用基本K均值进行二分
      3. 选择具有最小总SSE的两个簇，将其加到簇表中
      4. 重复步骤2、3，直到簇表中包含K个簇
  - 凝聚层次聚类：

算法 8.3　基本凝聚层次聚类算法
1：如果需要，计算邻近度矩阵
2：**repeat**
3：　合并最接近的两个簇
4：　更新邻近性矩阵，以反映新的簇与原来的簇之间的邻近性
5：**until** 仅剩下一个簇

  - DBSCAN：

算法 8.4　DBSCAN 算法
1：将所有点标记为核心点、边界点或噪声点。
2：删除噪声点。
3：为距离在 Eps 之内的所有核心点之间赋予一条边。
4：每组连通的核心点形成一个簇。
5：将每个边界点指派到一个与之关联的核心点的簇中。

- 优缺点：
  - 答题模板：
    - 时空复杂度：快速高效/计算量&节省空间/存储需求
    - 主/客观性&准确性：需要估计/预先设定参数[的多少]
    - 敏感性/灵敏度/稳定性/鲁棒性：
      - 对离群点/孤立点/异常值、噪声（被测变量的随机错误或变化，即抗噪性）
      - 对维度、初始值、距离测量方法/测度
      - 簇：
        - 维度、尺寸/大小、形状（球形&非凸）、变密度
        - 容易将大簇拆分
    - 全局目标函数
    - 合并决策是否是最终的/能倒退
  - 具体：
    - K均值聚类：
      - *优点：
        - 快速，简单
        - 当单个簇是密集的，而簇与簇间区别明显时，效果较好
      - 缺点：
        - k值需自己设定（有一定的主观成分）
        - 对包含离群点和噪声的数据进行聚类时有问题，不能处理不同尺寸、非球形簇&非凸、不同密度的簇
    - 凝聚层次聚类：
      - 优点：
        - 高质量

        - *不需要事先给出聚类数
    - 缺点：
        - 计算量大→时间复杂度高，存储需求昂贵
        - 可能对离群点和噪声敏感、难于处理不同尺寸/大小和非球状的簇&非凸情况、容易将大簇拆分
        - 缺乏全局目标函数
        - 合并决策是最终的（即不能反悔倒退）
    - 单链：不同簇之间两个最近的点之间的近邻度
        - 优点：可以处理非椭圆形状
        - 缺点：对噪声和离群点敏感
    - 全链：...最远
        - 优点：对噪声和离群点不太敏感
        - 缺点：
            - 偏好球形的簇
            - 容易将大簇拆分
    - 组平均：...所有点...平均值
        - 优点：对噪声和离群点不太敏感
        - 缺点：偏好球形的簇
- DBSCAN：
    - 优点：
        - *不需要事先给出聚类个数
        - 相对抗噪声，能够处理任意尺寸/大小和形状（可以处理非凸聚类）
    - 缺点：
        - *当数据量大时，I/O开销大
        - *对参数Eps、MinPts敏感
        - *聚类效果依赖于距离公式的选取
        - 对于高维的、变密度的簇可能会有问题

- 时间复杂度（其中m为数据点的个数）：
    - K均值分析：O(kmid)，其中k为聚类中心个数，i为算法的迭代次数，d为属性/维度个数
    - 凝聚层次聚类：O(m^2*logm)
    - DBSCAN：
        - 基本时间复杂度：O(m*找出Eps邻域中的点所需要的时间)
        - 最坏时间复杂度：O(m^2)，kd树：可以降到O(mlogm)
- 空间复杂度：
    - K均值分析：O((k+m)*i)，其中k为聚类中心个数，i为算法的迭代次数
    - 凝聚层次聚类：O(m^2)
    - DBSCAN：O(m)

①概述

2023年11月8日 20:47

什么是聚类分析

- 仅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
- 又称非监督分类
- 相近词汇：分割、划分

不同的聚类模型

完全/部分聚类（是否每个对象都被指派到一个簇）、层次聚类、模糊聚类、划分聚类（最简单）

不同的簇类型

明显分离的簇、基于中心/原型的簇、基于连续性的簇、基于密度的簇

②K均值

2023年11月15日 11:32

基本K均值算法

1：选择 K 个点作为初始质心。
2：**repeat**
3：　将每个点指派到最近的质心，形成 K 个簇。
4：　重新计算每个簇的质心。
5：**until** 质心不发生变化。

- 输入：所有点及其坐标
- 输出：所有点所属类别与所有类结果
- 时间复杂度：O(mkid)，其中m是数据点的总数，k是聚类中心个数，i是算法的迭代次数，d是属性/维度个数
- 距离度量：
  - 欧几里得空间数据：曼哈顿距离（L1）、欧几里得距离（L2），其中欧几里得距离：

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist(c_i, x)^2$$

$$\mathbf{c}_i = \frac{1}{m_i} \sum_{x \in C_i} \mathbf{x}$$

  - 文档数据 - 余弦相似度、Jaccard度量，其中余弦相似度：

$$\text{Total Cohesion} = \sum_{i=1}^{K} \sum_{x \in C_i} cosine(\mathbf{x}, \mathbf{c}_i)$$

  - 一般情况：

邻近度函数	质　心	目标函数
曼哈顿距离（L_1）	中位数	最小化对象到其簇质心的 L_1 距离和
平方欧几里得距离（L_2^2）	均值	最小化对象到其簇质心的 L_2 距离的平方和
余弦	均值	最大化对象与其簇质心的余弦相似度和
Bregman 散度	均值	最小化对象到其簇质心的 Bregman 散度和

    - Bregman散度：实际上为一类邻近性度量，包括平方欧几里得距离L2^2、Mahaloanobis距离和余弦相似度

K均值：附加的问题

- 处理空簇：需用某种策略来选择一个替补质心（否则平方误差会偏大），常用方法有：
  - 选择一个距离当前任何质心最远的点
  - 从具有最大SSE的簇中选择一个替补质心
- 离群点：无~
- 用后处理降低SSE：
  - 降低总SSE-增加簇个数：分裂一个簇、引进一个新的质心
  - 最小化总SSE的增长-减少簇个数：拆散一个簇、合并两个簇
- 增量地更新质心：在点到簇的每次指派之后增量地更新质心，而不是在所有点点都指派到簇中之后才更新簇质心

二分K均值

算法步骤：

算法 8.2　二分 K 均值算法

1：初始化簇表，使之包含由所有的点组成的簇。
2：**repeat**
3：　从簇表中取出一个簇。
4：　{对选定的簇进行多次二分“试验”。}
5：　**for** $i = 1$ to 试验次数 **do**
6：　　使用基本 K 均值，二分选定的簇。
7：　**end for**
8：　从二分试验中选择具有最小总 SSE 的两个簇。
9：　将这两个簇添加到簇表中。
10：**until** 簇表中包含 K 个簇。

- 归纳总结：

1. 初始化簇表，使之包含由所有点组成的簇
2. 从簇表中依次取出每一个簇，分别用基本K均值进行二分
3. 选择具有最小总SSE的两个簇，将其加到簇表中
4. 重复步骤2、3，直到簇表中包含K个簇

优点&缺点

K 均值简单并且可以用于各种数据类型。它也相当有效，尽管常常多次运行。K 均值的某些变种（包括二分 K 均值）甚至更有效，并且不太受初始化问题的影响。然而，K 均值并不适合所有的数据类型。它不能处理非球形簇、不同尺寸和不同密度的簇，尽管指定足够大的簇个数时它通常可以发现纯子簇。对包含离群点的数据进行聚类时，K 均值也有问题。在这种情况下，离群点检测和删除大有帮助。最后，K 均值仅限于具有中心（质心）概念的数据。一种相关的 K 中心点聚类技术没有这种限制，但开销更大。

- 归纳总结：
  - *优点：
    - 简单快速
    - 当单个簇是密集的，而簇与簇间区别明显时，效果较好
  - 缺点：
    - k值需自己设定（有一定的主观成分）
    - 不能处理非球形簇、不同尺寸和不同密度的簇
    - 对包含离群点和噪声的数据进行聚类时也有问题
    - *不能有效处理非凸情况
- 初始中心点问题的解决方案：
  - 多次运行
    - 有帮助，但概率不在你这边
  - 使用某种策略选择 k 个初始中心点，然后在这些初始中心点中进行选择
    - 选择分隔最远的
      - K-means++ 是进行这种选择的一种稳健方法
    - 使用分层聚类确定初始中心点
  - 分段 K-means
    - 归纳总结：分段→多次→使用某种策略...

③凝聚层次聚类

2023年11月22日　15:59

- 分类：
  - 凝聚的：从点作为个体簇开始，每一步合并两个最接近的簇。这需要定义簇的邻近性概念。
  - 分裂的：从包含所有点的某个簇开始，每一步分裂一个簇，直到仅剩下单点簇。在这种
    - 凝聚使用的更多

基本凝聚层次聚类算法

- 算法步骤：

算法 8.3　基本凝聚层次聚类算法

1：如果需要，计算邻近度矩阵
2：**repeat**
3：　合并最接近的两个簇
4：　更新邻近性矩阵，以反映新的簇与原来的簇之间的邻近性
5：**until** 仅剩下一个簇

1. Compute the proximity matrix
2. Let each data point be a cluster
3. Repeat
4. 　　Merge the two closest clusters
5. 　　Update the proximity matrix
6. Until only a single cluster remains

- 定义簇之间的近邻性：
  - 单链：不同簇之间两个最近的点之间的近邻度
    - 优点：可以处理非椭圆形状
    - 缺点：对噪声和离群点敏感
  - 全链：...最远
    - 优点：对噪声和离群点相对更不敏感
    - 缺点：
      - 容易将大簇拆分
      - 对球形的簇不太合适
  - 组平均：...所有点...平均值
    - 优点：对噪声和离群点相对更不敏感
    - 缺点：对球形的簇不太合适
- 时间和空间复杂性：时间复杂度为O(m^2*logm)，空间复杂度为O(m^2)，其中m为数据点的个数

特殊技术

- Ward方法&质心方法：

对于 Ward 方法，两个簇的邻近度定义为两个簇合并时导致的平方误差的增量。这样一来，该方法使用的目标函数与 K 均值相同。尽管看上去这一特点使得 Ward 方法不同于其他层次聚类技术，但是可以从数学上证明：当两个点之间的邻近度取它们之间距离的平方时，Ward 方法与组平均非常相似。

  - 合并的两个簇可能比后一步合并的簇更相似
  - 优缺点同组平均

簇邻近度的Lance-Williams公式

本节我们讨论过的任何簇邻近度都可以看作簇 Q 和 R 之间邻近度的不同参数（下面公式（8-7）显示的 Lance-Williams 公式）的一种选择，其中 R 是通过合并簇 A 和 B 形成的。在公式（8-7）中，$p(.,.)$是邻近度函数，而 m_A、m_B和 m_Q分别是簇 A、B 和 Q 的点数。换言之，合并簇 A 和 B 形成簇 R 之后，新簇 R 与原簇 Q 的邻近度是 Q 与原来的簇 A 和 B 的邻近度的线性函数。表8-5对我们讨论过的技术显示了这些系数的值。

$$p(R,Q)=\alpha_A p(A,Q)+\alpha_B p(B,Q)+\beta p(A,B)+\gamma|p(A,Q)-p(B,Q)| \quad (8\text{-}7)$$

层次聚类的主要问题

缺乏全局目标函数、处理不同大小簇的能力、合并决策是最终的（即不能反悔倒退）

优点&缺点

- 优点：
    - 高质量
    - *不需要事先给出聚类数
- 缺点：
    - 但计算量和存储需求昂贵&上一部分内容（主要问题）
    - 可能对噪声不敏感
    - 难于处理不同大小和非球状的簇
    - 容易将大簇拆分
    - 时间复杂度高
    - 无法处理非凸情况

④DBSCAN

2023年11月29日　15:28

传统的密度：基于中心的方法

- 核心点（core point）：这些点在基于密度的簇内部。点的邻域由距离函数和用户指定的距离参数 *Eps* 决定。核心点的定义是，如果该点的给定邻域内的点的个数超过给定的阈值 *MinPts*，其中 *MinPts* 也是一个用户指定的参数。在图 8-21 中，如果 *MinPts*≤7，则对于给定的半径（*Eps*），点 *A* 是核心点。
- 边界点（border point）：边界点不是核心点，但它落在某个核心点的邻域内。在图 8-21 中，点 *B* 是边界点。边界点可能落在多个核心点的邻域内。
- 噪声点（noise point）：噪声点是既非核心点也非边界点的任何点。在图 8-21 中，点 *C* 是噪声点。

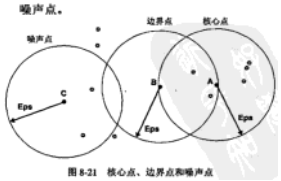


图 8-21　核心点、边界点和噪声点

DBSCAN算法

算法 8.4　DBSCAN 算法

1：将所有点标记为核心点、边界点或噪声点。
2：删除噪声点。
3：为距离在 *Eps* 之内的所有核心点之间赋予一条边。
4：每组连通的核心点形成一个簇。
5：将每个边界点指派到一个与之关联的核心点的簇中。

- 时间复杂性&空间复杂性：基本时间复杂度为O(m*找出Eps邻域中的点所需要的时间），最坏时间复杂度为O(m^2)，空间复杂度为O(m)
- 选择DBSCAN参数：

  当然，还有如何确定参数 *Eps* 和 *MinPts* 的问题。基本方法是观察点到它的 *k* 个最近邻的距离（称为 *k*-距离）的特性。对于属于某个簇的点，如果 *k* 不大于簇的大小的话，则 *k*-距离将很小。注意，尽管因簇的密度和点的随机分布不同而有一些变化，但是如果簇密度的差异不是很极端的话，在平均情况下变化不会太大。然而，对于不在簇中的点（如噪声点），*k*-距离将相对较大。因此，如果我们对于某个 *k*，计算所有点的 *k*-距离，以递增次序将它们排序，然后绘制排序后的值，则我们会看到 *k*-距离的急剧变化，对应于合适的 *Eps* 值。如果我们选取该距离为 *Eps* 参数，而取 *k* 的值为 *MinPts* 参数，则 *k*-距离小于 *Eps* 的点将被标记为核心点，而其他点将被标记为噪声或边界点。

- 变密度的簇：若簇的密度变化很大，DBSCAN可能会有问题

优点&缺点

- 优点：
  - 相对抗噪声
  - 能够处理任意形状和大小（可以处理非凸聚类）
  - *不需要事先给出聚类个数
- 缺点：
  - 对于变密度的簇、高维的簇可能会有问题
  - *当数据量大时，I/O开销大
  - *聚类效果依赖于距离公式的选取
  - *对参数Eps、MinPts敏感

⑤簇评估/簇确认

2023年11月29日　15:49

概述

- 非监督的。聚类结构的优良性度量，不考虑外部信息。例如，SSE。簇的有效性的非监督度量常常可以进一步分成两类：簇的凝聚性（cluster cohesion）（紧凑性，紧致性）度量确定簇中对象如何密切相关，簇的分离性（cluster separation）（孤立性）度量确定某个簇不同于其他簇的地方。非监督度量通常称为内部指标（internal index），因为它们仅使用出现在数据集中的信息。
- 监督的。度量聚类算法发现的聚类结构与某种外部结构的匹配程度。例如，监督指标的熵，它度量簇标号与外部提供的标号的匹配程度。监督度量通常称为外部指标（external index），因为它们使用了不在数据集中出现的信息。
- 相对的。比较不同的聚类或簇。相对簇评估度量是用于比较的监督或非监督评估度量。因而，相对度量实际上不是一种单独的簇评估度量类型，而是度量的一种具体使用。例如，两个 K 均值聚类可以使用 SSE 或熵进行比较。

非监督簇评估：使用凝聚度（簇内）&分离度（簇间）

名称	簇 度 量	簇 权 值	类　型
$\mathcal{I}_1$	$\sum_{\mathbf{x}\in C_i,\mathbf{y}\in C_i} proximity(\mathbf{x},\mathbf{y})$	$\frac{1}{m_i}$	基于图的凝聚度
$\mathcal{I}_2$	$\sum_{\mathbf{x}\in C_i} proximity(\mathbf{x},\mathbf{c}_i)$	1	基于原型的凝聚度
$\mathcal{E}_1$	$proximity(\mathbf{c}_i,\mathbf{c})$	m_i	基于原型的分离度
$\mathcal{G}_1$	$\sum_{i=1}^{k}\sum_{\mathbf{x}\in C_i,\mathbf{y}\in C_i} proximity(\mathbf{x},\mathbf{y})$	$\frac{1}{\sum_{\mathbf{x}\in C_i,\mathbf{y}\in C_i} proximity(\mathbf{x},\mathbf{y})}$	基于图的凝聚度和分离度

- 凝聚度&分离度的基于图的观点：

从数学上讲，基于图的簇的凝聚度和分离度可以分别用公式（8-9）和公式（8-10）表示。其中，*proximity* 函数可以是相似度、相异度，或者是这些量的简单函数。

$$cohesion(C_i) = \sum_{\mathbf{x}\in C_i,\mathbf{y}\in C_i} proximity(\mathbf{x},\mathbf{y}) \quad (8\text{-}9)$$

$$separation(C_i, C_j) = \sum_{\mathbf{x}\in C_i,\mathbf{y}\in C_j} proximity(\mathbf{x},\mathbf{y}) \quad (8\text{-}10)$$

- 凝聚度和分离度的基于原型的观点：

基于原型的凝聚度由公式（8-11）给出，而两个分离性度量分别由公式（8-12）和公式（8-13）给出，其中 $\mathbf{c}_i$ 是簇 C_i 的原型（质心），而 $\mathbf{c}$ 是总体原型（质心）。对于分离性，存在两种度量（稍后就会看到），这是因为簇原型与总原型的分离度有时与簇原型之间的分离度直接相关。注意，如果我们取邻近度为平方欧几里得距离，则公式（8-11）是簇的 SSE。

$$cohesion(C_i) = \sum_{\mathbf{x}\in C_i} proximity(\mathbf{x},\mathbf{c}_i) \quad (8\text{-}11)$$

$$separation(C_i, C_j) = proximity(\mathbf{c}_i, \mathbf{c}_j) \quad (8\text{-}12)$$

$$separation(C_i) = proximity(\mathbf{c}_i, \mathbf{c}) \quad (8\text{-}13)$$

- 凝聚度&分离度的总度量：略~

- 基于原型的凝聚度&基于图的凝聚度之间的联系

尽管度量簇的凝聚性和分离性的基于图的方法与基于原型的方法看上去截然不同，但是对于某些邻近性度量它们是等价的。例如，对于 SSE 和欧几里得空间的点，可以证明（公式（8-14））簇中逐对点的平均距离等于簇的 SSE（见本章习题 27）。

- *两种基于原型的分离性度量方法：

当邻近度用欧几里得距离度量时，簇之间分离性的传统度量是组平方和（SSB），即簇质心 $\mathbf{c}_i$ 到所有数据点的总均值 $\mathbf{c}$ 的距离的平方和。通过在所有簇上对 SSB 求和，我们得到总 SSB，由公式（8-15）给出，其中 $\mathbf{c}_i$ 是第 i 个簇的均值，而 $\mathbf{c}$ 是总均值。总 SSB 越高，簇之间的分离性越好。

$$总\ SSB = \sum_{i=1}^{K} m_i dist(\mathbf{c}_i,\mathbf{c})^2 \quad (8\text{-}15)$$

可以直接证明，总 SSB 与质心之间的逐对距离有直接关系。特殊地，如果簇的大小相等，即 $m_i = m/K$，则该关系取公式（8-16）给出的简单形式（见本章习题 28）。正是这类等价性诱导了公式（8-12）和公式（8-13）的原型分离度定义。

$$总\ SSB = \frac{1}{2K}\sum_{i=1}^{K}\sum_{j=1}^{K}\frac{m}{K} dist(\mathbf{c}_i,\mathbf{c}_j)^2 \quad (8\text{-}16)$$

- 凝聚度和分离度之间的联系：

在某些情况下，凝聚度和分离度之间也存在很强的联系。具体地说，可以证明总 SSE 和总 SSB 之和是一个常数，它等于总平方和（TSS）——每个点到数据的总均值的距离的平方和。这个结果的重要性在于：最小化 SSE（凝聚度）等价于最大化 SSB（分离度）。

下面，我们给出该事实的证明，该证明所用的方法也适用于证明前两节陈述的联系。为了简化证明过程，我们假定数据是一维的，即 $dist(x, y) = (x-y)^2$。证明中还使用了交叉项 $\sum_{i=1}^{K}\sum_{x\in C_i}(x-c_i)(c-c_i)$ 为 0 的事实（见本章习题 29）。

$$\begin{aligned}
TSS &= \sum_{i=1}^{K}\sum_{x\in C_i}(x-c)^2 \\
&= \sum_{i=1}^{K}\sum_{x\in C_i}((x-c_i)-(c-c_i))^2 \\
&= \sum_{i=1}^{K}\sum_{x\in C_i}(x-c_i)^2 - 2\sum_{i=1}^{K}\sum_{x\in C_i}(x-c_i)(c-c_i) + \sum_{i=1}^{K}\sum_{x\in C_i}(c-c_i)^2 \\
&= \sum_{i=1}^{K}\sum_{x\in C_i}(x-c_i)^2 + \sum_{i=1}^{K}\sum_{x\in C_i}(c-c_i)^2 \\
&= \sum_{i=1}^{K}\sum_{x\in C_i}(x-c_i)^2 + \sum_{i=1}^{K}|C_i|(c-c_i)^2 \\
&= SSE+SSB
\end{aligned}$$

$$= \sum_{i=1}^{K}\sum_{x\in C_i}(x-c_i)^2 + \sum_{i=1}^{K}\sum_{c\in C_i}(c-c_i)^2$$
$$= \sum_{i=1}^{K}\sum_{x\in C_i}(x-c_i)^2 + \sum_{i=1}^{K}|C_i|(c-c_i)^2$$
$$=SSE+SSB$$

- 评估个体簇&对象：略~
- 轮廓系数：

流行的轮廓系数（silhouette coefficient）方法结合了凝聚度和分离度。下面的步骤解释如何计算个体点的轮廓系数。此过程由如下三步组成。我们使用距离，但是类似的方法可以使用相似度。

(1) 对于第 i 个对象，计算它到簇中所有其他对象的平均距离。该值记作 a_i。

(2) 对于第 i 个对象和不包含该对象的任意簇，计算该对象到给定簇中所有对象的平均距离。关于所有的簇，找出最小值；该值记作 b_i。

(3) 对于第 i 个对象，轮廓系数是 $s_i = (b_i - a_i)/\max(a_i, b_i)$。

轮廓系数的值在–1 和 1 之间变化。我们不希望出现负值，因为负值表示点到簇内点的平均距离 a_i 大于点到其他簇的最小平均距离 b_i。我们希望轮廓系数是正的（$a_i < b_i$），并且 a_i 越接近 0 越好，因为当 $a_i = 0$ 时轮廓系数取其最大值 1。

我们可以简单地取簇中点的轮廓系数的平均值，计算簇的平均轮廓系数。通过计算所有点的平均轮廓系数，可以得到聚类优良性的总度量。

非监督簇评估：使用邻近度矩阵

通过相关性度量簇的有效性、通过相似度矩阵可视地评价聚类：略~

层次聚类的非监督评估

共性分类距离、共性分类相关系数（CPCC，该矩阵与原来的相异度矩阵的项之间的相关度）：略~

确定正确的簇个数

略~

聚类趋势

略~

簇有效性的监督度量

面向分类的度量

- 熵：每个簇由单个类的对象组成的程度。对于每个簇，首先计算数据的类分布，即对于簇 i，计算簇 i 的成员属于类 j 的概率 $p_{ij} = m_{ij}/m_i$，其中 m_i 是簇 i 中对象的个数，而 m_{ij} 是簇 i 中类 j 的对象个数。使用类分布，用标准公式 $e_i = -\sum_{j=1}^{L} p_{ij}\log_2 p_{ij}$ 计算每个簇 i 的熵，其中 L 是类的个数。簇集合的总熵用每个簇的熵的加权和计算，即 $e = \sum_{i=1}^{K}\frac{m_i}{m}e_i$，其中 K 是簇的个数，而 m 是数据点的总数。
- 纯度：簇包含单个类的对象的另一种度量程序。使用前面的术语，簇 i 的纯度是 $p_i = \max_j p_{ij}$，而聚类的总纯度是 $purity = \sum_{i=1}^{K}\frac{m_i}{m}p_i$。
- 精度：簇中一个特定类的对象所占的比例。簇 i 关于类 j 的精度是 $precision(i, j) = p_{ij}$。
- 召回率：簇包含一个特定类的所有对象的程度。簇 i 关于类 j 的召回率是 $recall(i, j) = m_{ij}/m_j$，其中 m_j 是类 j 的对象个数。
- F 度量：精度和召回率的组合，度量在多大程度上，簇只包含一个特定类的对象和包含该类的所有对象。簇 i 关于类 j 的 F 度量是 $F(i, j) = (2 \times precision(i, j) \times recall(i, j))/(precision(i, j) + recall(i, j))$。

面向相似性的度量

$$\text{Rand 统计量} = \frac{f_{00}+f_{11}}{f_{00}+f_{01}+f_{10}+f_{11}}$$

$$\text{Jaccard 系数} = \frac{f_{11}}{f_{01}+f_{10}+f_{11}}$$

*层次聚类的簇有效性

该方法的关键思想是，评估层次聚类是否对于每个类，至少有一个簇相对较纯，并且包含了该类的大部分对象。为了根据此目标评估层次聚类，我们对每个类，计算簇层次结构中每个簇的 F 度量。对于每个类，取最大的 F 度量。最后，通过计算每类的 F 度量的加权平均，计算层次聚类的总 F 度量，其中，权值基于类的大小。该层次 F 度量的形式上的定义如下：

$$F = \sum_j \frac{m_j}{m}\max_i F(i, j)$$

其中，最大值在所有层的所有簇 i 上取，m_j 是类 j 中对象的个数，而 m 是对象的总数。

评估簇有效性度量的显著性

略~