数据挖掘原理与算法期中复习

> 作者：cqq
>
> 整理时间：2023-11-29
>
> 最近更新时间：2023-11-30

考试题型与提纲

题型：

1. 填空
2. 选择
3. 名词解释
    - 参考释义部分
4. 英汉互译
    - 参考术语部分
5. 简答题
    - 简述数据库中知识发现（KDD）阶梯处理过程模型。
    - 知识发现（KDD）一般包含哪些基本阶段？简述各阶段的特点。
    - 事务数据库中关联规则挖掘主要包括哪两个主要步骤？各步骤的主要任务和目标是什么？
    - 简述***算法的原理和步骤。
    - 简述时间序列挖掘的含义、意义、分类及其主要任务。
    - 简述空间数据挖掘的含义、意义、分类及其主要任务。
    - 简述Web挖掘的含义、意义、分类及其主要任务。
6. 证明

说明

> ★为复习重点，加粗为知识提炼；本复习资料主要用于概念记忆与算法步骤和伪代码的记忆，即侧重于记忆部分，对于算法的理解和例题见资料中标出的课本相对应的页码；练手的题目可参考课本例题、课后习题和往年试卷考题。

术语

1. 重点掌握★
    - Data Mining：数据挖掘
    - Artificial Intelligence：人工智能
    - Knowledge Discovery：知识发现
    - OLAP（Online Analytical Process）：在线分析处理
    - Association Rule：关联规则
    - Maximal Frequent ItemSet：最大频繁项集
    - Data Classification：数据分类
    - Decision Tree：决策树

- k-Nearest Neighbors：k最近邻
- Clustering：聚类
- 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

2. 补充

- Decision Support：决策支持
- Usage Mining：访问挖掘
- Link Mining：链接挖掘
- Data Visualization：数据可视化
- Information Retrieval：信息检索
- OLAP（Online Analytical Mining）：在线分析挖掘

释义

1. 重点掌握★

- ***数据挖掘***：定义有广义和狭义之分。从广义的观点，数据挖掘是从大型数据集中**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。从狭义的观点出发，我们定义数据挖掘从特定形式的数据集中提炼知识的过程。
- ***频繁项目集***：是指**出现频率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，许多关联规则挖掘方法是基于频繁项目集发现的。
- ***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，为每个类找到一种准确的描述或者模型。
- ***序列挖掘***：是指**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。**
- ***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较大。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，属于一种无指导的学习方法。
- ***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**按照距离准则逐步聚类，直至满足分类要求为止。**一般分为凝聚的和分裂的两种方案。
- ***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，知道人们的社会、经济、军事和生活等行为。
- ***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。**
- ***数字地球***：是一种**关于地球的可以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，可供出版者出版。

2. 补充

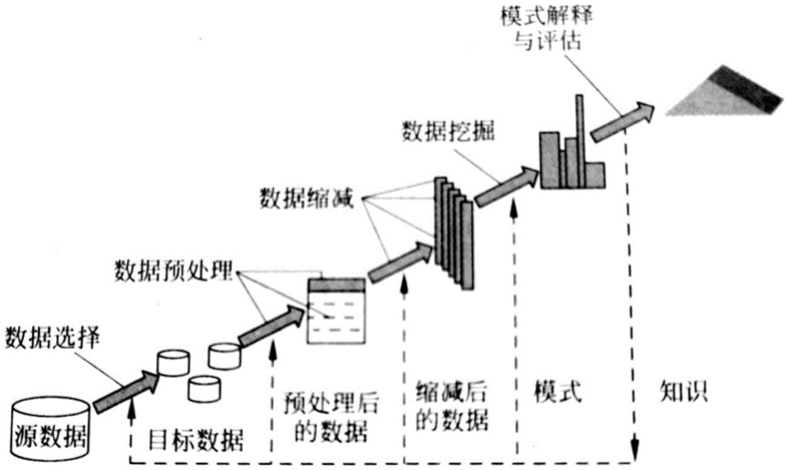
- ***机器学习（Machine Learning）***：研究**如何使用机器来模拟人类学习活动**的一门学科。
- ***人工神经网络（Artificial Neural Networks，ANNs）***：**简称神经网络**（NNs），是一种**模仿动物神经网络行为特征**，进行**分布式并行信息处理**的算法数学模型。
- ***人工智能（Artificial Intelligence）***：研究如何应用机器来**模拟人类某些智能行为**的基本理论、方法和技术的一门科学。
- ***大数据（Big Data）***：指一种**规模大到在获取、存储、管理、分析方面**大大**超出了传统数据库软件工具能力范围**的数据集合。（特征是）海量的数据规模、快速的数据流转、多样的数据类

型和价值密度低。

- ***知识工程（Knowledge Engineering）***：研究**知识信息处理**并**探讨开发知识系统**的技术。
- ***广义知识（Generalization）***：描述**类别特征**的概括性知识。这类数据挖掘系统是对细节数据所蕴涵的概念特征信息的概括和抽象的过程。
- ***关联知识（Association）***：**反映**一个事件和其他事件之间的**依赖或关联。**目的是找出数据库中**隐藏的关联信息。**
- ***爬虫（spider/crawler）***：是一种按照一定的规则，**自动地抓取万维网信息的程序或者脚本。**
- ***数据仓库（Data Warehouse）***：是决策支持系统（dss）和联机分析**应用数据源的结构化数据环境。**数据仓库**研究和解决**从数据库中**获取信息的问题。**数据仓库的特征在于**面向主题、集成性、稳定性和时变性。**
- ***OLTP（On-Line Transaction Processing）***：**联机事务处理传统的关系型数据库的主要应用**，主要是基本的、日常的**事务处理**（增删改查），例如银行交易。
- ***OLAP（On-Line Analytic Processing）***：**联机分析处理数据仓库系统的主要应用**，支持**复杂的分析操作**，侧重**决策支持**，并且提供直观易懂的查询结果 。
- ***决策支持（Decision Support）***：决策者通过数据、模型和知识，以**人机交互方式**进行**半结构化或非结构化决策。**
- ***事务数据库（Transaction Database）***：一个**事务数据库**是对**事务型数据的收集。**
- ***分布式数据库（Distributed Database）***：**物理上分散**而**逻辑上集中**的数据库系统。

知识发现过程

1. 知识发现（Knowledge Discovery in Database，KDD）★
    - 定义：是一个**系统化**的工作，必须**对可以利用的源数据进行分析**，**确定合适的挖掘目标**，然后才能着手系统的设计和开发。KDD是一个多步骤的处理过程，一般分为**问题定义、数据采集、数据预处理、数据挖掘、模式评估**等基本阶段。
    - 过程
        1. 首先从数据源中**抽取感兴趣的数据**，并把它**组织成适合挖掘的数据组织形式**；
        2. 然后**调用相应的算法**生成所需的知识；
        3. 最后对生成的知识模式**进行评估**，并把有价值的知识集成到企业的智能系统中。
    - 基本阶段的特点
        - ***问题定义阶段***：KDD是用于在大量数据中**发现有用的令人感兴趣的信息**，因此发现何种知识就成为整个过程中第一个也是最重要的一个阶段。
        - ***数据抽取阶段***：目的是**选取相应的源数据库**，并**根据要求**从数据库中**提取相关的数据。**
        - ***数据预处理阶段***：主要对前一阶段抽取的数据**进行再加工**，**检查数据的完整性及数据的一致性**，包括消除噪声、推导计算缺值数据、消除重复记录、完成数据类型转换（如把连续值变成离散值，以便符号归纳等）。
        - ***数据挖掘阶段***：**运用选定的数据挖掘算法**，从数据中**提取出用户需要的知识。**
        - ***知识评估阶段***：数据挖掘阶段发现出来的模式，经过评估，**可能存在冗余或无关的模式**，这需要将其**剔除**；也有可能**不满足用户的需求**，这需要将整个发现过程**回退到前续阶段。**如重新选取数据、采用新的数据变换方法等。知识评估是KDD一个重要的必不可少的阶段。
2. 阶梯处理过程模型★
    - 基本处理流程图示



- 定义：阶梯处理过程模型将数据库中的知识发现看作是一个**多阶段的处理过程**，在整个知识发现的过程中包括很多处理阶段。**主要有九个处理阶段**，这九个处理阶段分别是**数据准备、数据选择、数据预处理、数据缩减、KDD目标确定、挖掘算法确定、数据挖掘、模式解释**及**知识评价。**
- 处理阶段
  - ***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。

关联规则挖掘

1. 项目集格空间理论★
   - 核心原理
     - 频繁项目集的**非空子集**都是频繁项目集。
     - 非频繁项目集的**超集**都是非频繁项目集。

2. 事务数据库中关联规则挖掘★

- 主要步骤

  1. ***挖掘频繁项集***

     - 主要任务：在事务数据库中**找到频繁项集**，即在数据集中经常同时出现的项的集合。
     - 目标：**识别出**在数据集中**频繁出现的项集**，这些项集的支持度（Support）**大于或等于预先设定的最小支持度阈值。**

  2. ***生成关联规则***

     - 主要任务：基于频繁项集，生成**满足最小置信度阈值**的关联规则。
     - 目标：**发现**项集之间的**关联规则**，以确定项集 A 的出现是否可能导致项集 B 的出现。

3. Apriori算法★

- 定义：是一种**挖掘关联规则**的**频繁项集算法**，其核心思想是通过**候选集生成**和**情节的向下封闭检测**两个阶段来挖掘频繁项集。
- 算法步骤

  1. 扫描全部数据，产生候选 1-项集的集合C_1
  2. 根据最小支持度，由候选 1-项集的集合C_1产生频繁 1-项集的集合L_1
  3. 对$k>1$,重复执行步骤 4,5,6
  4. 由L_k执行连接和剪枝操作，产生候选$(k+1)$-项集的集合C_{k+1}
  5. 根据最小支持度，由候选$(k+1)$-项集的集合 C_{k+1}，产生频繁$(k+1)$-项集的集合L_{k+1}
  6. 若L不等于∅，则$k=k+1$，步骤跳 4，否则结束
  7. 根据最小置信度，由频繁项集产生强关联规则

- 伪代码

  1. ***发现频繁项目集***

     - 输入：数据集D；最小支持度minsup_count
     - 输出：频繁项目集L

     (1) L_1＝{large 1-itemsets}；//所有支持度不小于 minsupport 的 1-项目集
     (2) FOR （k＝2；$L_{k-1}\neq\Phi$；k＋＋）DO BEGIN
     (3) C_k＝apriori-gen(L_{k-1})；//C_k 是 k 个元素的候选集
     (4) FOR all transactions t∈D DO BEGIN
     (5) C_t＝subset(C_k,t)；//C_t 是所有 t 包含的候选集元素
     (6) FOR all candidates c∈C_t DO c. count＋＋；
     (7) END
     (8) L_k＝{c∈C_k | c. count≥minsup_count}
     (9) END
     (10) L＝$\bigcup L_k$；

  2. ***候选集产生***

     - 输入：(k-1)-频繁项目集Lk-1
     - 输出：k-候选项目集Ck

```
(1) FOR all itemset p∈L_{k-1} DO
(2)     FOR all itemset q∈L_{k-1} DO
(3)         IF p.item_1 = q.item_1, p.item_2 = q.item_2, ···, p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1} THEN BEGIN
(4)             c=p∞q;                    //把 q 的第 k-1 个元素连到 p 后
(5)             IF has_infrequent_subset(c, L_{k-1}) THEN
(6)                 delete c;             //删除含有非频繁项目子集的候选元素
(7)             ELSE add c to C_k;
(8)         END
(9) Return C_k;
```

- 例题

  见课本P76-79

4. Close算法
   - 目的：**减少搜索空间**和**提高挖掘效率。**
   - 优点：Close 算法的**优势**在于它的**剪枝操作**，它能够**减少不必要的搜索**，提高了关联规则挖掘的效率。
   - 闭合项目集：一个项目集C，当且仅当对于在C中的任何元素，不可能在C中存在小于或等于它的支持度的子集。或者A的**闭合项集**是指所有包含A的**项目的交集**，**支持度**是指包含A的项目的**交集出现的频数。**

TID	Item
1	a, b, c
2	a, b, c, d
3	b, c, e
4	a, c, d, e
5	d, c

请找出a的闭合项集?

{abc}∩{abcd}∩{acde}={ac}

count{ac}= 3　闭合项集：{ac}　支持度：3

   - 算法步骤
     1. ***初始化***:
        - 生成所有单个项的频繁项集。
        - 对频繁项集按照字典序排序。
     2. ***迭代生成频繁项集***:
        - 从长度为 k-1 的频繁项集中生成长度为 k 的候选项集。
        - 对候选项集进行剪枝，移除非频繁的项。
        - 对剩余的候选项集进行扩展，生成新的频繁项集。
        - 对新生成的频繁项集按照字典序排序。
     3. ***判断是否终止***:
        - 如果没有生成更多频繁项集，则算法终止。
        - 否则，回到步骤 2。
     4. ***提取关联规则***:
        - 对于每个频繁项集，生成其所有非空子集。
        - 对每个子集，计算其支持度。
        - 利用支持度和信任度等指标提取关联规则。
   - 算法关键点
     - ***剪枝操作***：Close 算法引入了剪枝操作，以去除非频繁项，减少搜索空间。

- ***按字典序排序***：对生成的频繁项集按字典序排序，确保结果的一致性。
- ***频繁项集的生成***：使用候选项集的扩展和剪枝来生成新的频繁项集。

- 伪代码

算法 3-6 Close 算法

```
(1) generators in FCC1 = {1-itemsets};                          //候选频繁闭合 1-项目集
(2) FOR(i=1; FCCi. generators=Φ; i++) DO BEGIN
(3)    closures in FCCi = Φ;
(4)    supports in FCCi = 0;
(5)    FCCi = Gen_Closure(FCCi);                                //计算 FCC 的闭合
(6)    FOR all candidate closed itemsets c∈FCCi DO BEGIN
(7)       IF(c. support >= minsupport) THEN FCi = FCi ∪ {c};    //修剪小于最小支持度的项
(8)    END
(9)    FCCi+1 = Gen_Generator(FCCi);                            //生成 FCCi+1
(10) END
(11) FC = ∪i FCi(FCi. closure, FCi. support);                   //返回 FC
(12) Deriving frequent itemsets(FC, L);
```

- 例题

  见课本P86-88

5. FP-tree算法

   - 优势：只进行**两次数据库扫描。它不使用候选集**，直接压缩数据库成一个**频繁模式树**，最后通过这棵树生成关联规则。
   - 主要步骤

     1. ***利用事务数据库中的数据构造FP-tree***

        说明：在FP-tree构造过程中，总是尽量将出现**频度高的项目放在靠近根结点。**

        1. 首先扫描数据库一次生成1-频繁集，并将它们按降序排序，放入L表中。
        2. 再扫描数据库一次，对每个数据库的元组，把它对于项目集的关联和频度信息放入到FP-tree中。

     2. ***从FP-tree中挖掘频繁模式***

        说明：用FP-tree挖掘频繁集的基本思想是**分而治之。**

        1. 对每个项，生成它的条件模式集（一个“子数据库”，由FP-tree中与后缀模式一起出现的前缀路径集组成），然后是它的条件FP-tree。
        2. 对每个新生成的条件FP-tree，重复这个步骤。
        3. 直到结果FP-tree为空，或只含唯一的一个路径（此路径的每个子路径对应的项目集都是频繁集）

分类方法

1. 定义：分类是一种**监督学习**，即每个训练样本的数据对象已经有类标识，通过学习可以形成表达数据对象与类标识间对应的知识。
2. 目的：利用历史数据纪录来**自动学习一个分类模型/函数（分类器）**，利用该模型把数据库中的**数据项映射到给定类别中的某一个类别**，从而能对未来数据进行类别预测。
3. 基于距离的分类方法（KNN算法）★

   - 定义：KNN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。**

- 算法步骤

  1. ***收集数据***:
     - 收集带有标签的训练数据，其中每个样本都有一个类别标签。
  2. ***选择K值***:
     - 选择要用于分类的邻居数量（K值）。K值的选择通常通过交叉验证来确定。
  3. ***计算距离***:
     - 对于给定的未标记样本，计算它与训练集中每个已标记样本之间的距离。常用的距离度量包括欧氏距离、曼哈顿距离、闵可夫斯基距离等。
  4. ***找到K个最近邻***:
     - 选择与未标记样本距离最近的K个训练样本。
  5. ***投票决策***:
     - 对于分类问题，统计K个最近邻中每个类别的出现次数。未标记样本被赋予与其最近邻中出现最多的类别。

- 伪代码

  - 输入：训练数据T；近邻数目K；待分类的元组t。
  - 输出：输出类别c

```
(1) N=Φ;
(2) FOR each d∈T DO BEGIN
(3)   IF |N|≤k THEN
(4)     N=N∪{d};
(5)   ELSE
(6)       IF ∃ u∈N such that sim(t,u)< sim(t,d) THEN
(7)        BEGIN
(8)          N=N−{u};
(9)          N=N∪{d};
(10)       END
(11) END
(12) c=class related to such u∈N which has the most number;
```

- 例题

  见课本P126-127

4. 决策树分类方法（ID3算法，C4.5算法）

- ID3算法

  - 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。

$$H(p)=\sum_x p(x)log_2(\frac{1}{p(x)})=-\sum_x p(x)log_2(p(x))$$

  - 信息增益：**针对一个特征**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。

$$IG(T)=Entropy(S)-Entropy(S|T)$$

  - 算法步骤

1. ***选择最佳特征***:
    - 计算每个特征的信息增益（或信息增益比、基尼指数等）。
    - 选择信息增益最大的特征作为当前节点的分裂特征。
2. ***将数据集划分***:
    - 使用选定的分裂特征将数据集划分为子集，每个子集对应于该分裂特征的一个取值。
3. ***递归构建子树***:
    - 对于每个子集，重复步骤1和步骤2，递归构建子树。
    - 如果子集纯净（属于同一类别）或达到预定的树深度或停止条件，则停止递归。
4. ***构建决策树***:
    - 将选择的分裂特征作为当前节点的特征。
    - 对于每个子集，将其递归构建的子树作为当前节点的子树。
5. ***停止条件***:
    - 构建树的过程中，可以设置一些停止条件，如达到预定的树深度、子集纯净度达到阈值等。

- 伪代码

算法 4-4 ID3

```
输入：T：table                          //训练数据
      C：classification attribute       //类别属性
输出：decision tree                     //决策树
(1) BEGIN
(2) IF (T is empty) THEN return (null);
(3) N=a new node; //创建结点 N
(4) IF (there are no predictive attributes in T) THEN //第一种情况
(5)    label N with most common value of C in T(deterministic tree) or with frequencies of C in T
(probabilistic tree);
//如没有剩余属性来进一步划分 T,把给定的结点转换成树叶,用 T 中多数元组所在的类标记它
(6) ELSE IF (all instances in T have the same value V of C) THEN //第二种情况
(7)          label N, "X. C=V with probability 1";
//如果 T 中所有样本的类别都一样,标记 N,类别为 V
(8)        ELSE BEGIN
(9)          FOR each attribute A in T compute AVG ENTROPY(A,C,T);
//对 T 中每个属性 A 对其计算 AVG ENTROPY(A,C,T)
(10)           AS=the attribute for which AVG ENTROPY(A,C,T) is minimal;
//把 AVG ENTROPY(A,C,T)最小的属性标记 AS
(11) IF (AVG ENTROPY(AS,C,T) is not substantially smaller than ENTROPY(C,T)) THEN
                    //第三种情况
(12) label N with most common value of C in T(deterministic tree) or with frequencies of C in
T(probabilistic tree);
//如果 AVG ENTROPY(AS,C,T)不比 ENTROPY(C,T)小,用 T 中多数元组所在的类标记 N
(13)           ELSE BEGIN
(14)               label N with AS;
(15)               FOR each value V of AS DO BEGIN
(16)               N1=ID3(SUBTABLE(T,A,V),C); //递归调用
(17)               IF (N1 !=null) THEN make an arc from N to N1 labelled V;
(18)               END
(19)           END
(20)       END
(21) return N;
(22) END
```

- 例题

见课本P131-132

- C4.5算法
  - 相对ID3增加的新功能
    - 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**以避免树的过拟合
    - **k交叉验证**
    - 规则的产生方式
  - 信息增益比例

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

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

  - 例题

    见课本P136-138

- 两者的异同
  - 相同点：
    - 都是**基于信息熵的决策树算法**，都采用**自顶向下的贪心策略。**
    - 都是通过对数据集进行**递归划分**，生成决策树模型。
    - 都可以**处理离散型数据和连续型数据。**
  - 不同点：
    - **ID3算法使用信息增益**来选择最优特征，而**C4.5算法使用信息增益率**来选择最优特征。信息增益率在信息增益的基础上，对可取值数目较少的特征有所偏好，**避免了信息增益选择特征时偏向取值较多的特征的问题。**
    - **C4.5算法可以处理缺失值**，而ID3算法不能。
    - **C4.5算法使用后剪枝来避免过拟合**，而ID3算法没有使用后剪枝。

5. 贝叶斯分类（朴素贝叶斯算法）

- 基本思想
  - 贝叶斯定理：贝叶斯定理描述了在**给定先验知识**的情况下，如何通过新的观测数据来**更新对一个事件的概率**估计。
  - 朴素假设：朴素贝叶斯算法假设样本的特征之间相互独立，即**给定类别，特征之间不存在依赖关系。**
- 例题

  见课本P140-143

聚类方法

1. 定义：把一组个体**按照相似性归为若干类别。**聚类属于**无监督学习。**
2. 目的：使**同一类别**的个体之间的**差别尽可能地小**，**不同类别**上的个体间的**差别尽可能地大。**
3. 评价准则
    - **类间**距离**大**，即不同簇的数据尽量**不相似。**
    - **类内**距离**小**，即一个簇内的数据尽量**相似。**
4. 距离函数
    - 距离条件
        - 非负性
        - 对称性
        - 三角不等式
    - 明可夫斯基距离

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

    - 曼哈顿距离（绝对值距离）

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

    - 欧氏距离

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

    - 切比雪夫距离

$$d(x,y) = max(|x_1 - x_2|, |y_1 - y_2|)$$

    - 余弦距离

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

5. 划分聚类方法（k-means算法，k-medoids算法）
    - 主要思想
        - 每一个簇至少包含一个对象
        - 每一个对象属于且**仅**属于一个簇
    - 簇的表现形式
        - 通过它们的**中心**或者**类中关系远的（边界）点**表示空间的一类点

- 使用聚类树中的结点**图形化**地表示一个类
  - 使用样本属性的**逻辑表达式**表达类
- k-means算法★
  - 定义：也被称为**k-均值**，是一种被广泛使用的聚类算法。k-平均算法以k为参数，把n个对象分为k个簇，以使**簇内具有较高的相似度。**相似度的计算根据一个簇中的**对象的平均值**来进行。
  - 算法描述
    1. 给定集合D，有**n个样本点**
    2. **随机指定k个点**，**作为**k个子集的**质心**
    3. 根据样本点与k个质心的**距离远近**，将每个样本点**划归最近质心所在的子集**
    4. 对k个子集**重新计算质心**
    5. 根据新的质心，重复操作3.
    6. 重复操作4.和5.，**直至结果足够收敛或者不再变化**
  - 伪代码

    输入：簇的数目 k 和包含 n 个对象的数据库。
    输出：k 个簇，使平方误差准则最小。

    (1) assign initial value for means; //任意选择 k 个对象作为初始的簇中心
    (2) REPEAT
    (3) FOR j=1 to n DO assign each xj to the cluster which has the closest mean;
    //根据簇中对象的平均值，将每个对象赋给最类似的簇
    (4) FOR i=1 to k DO $\overline{x}_i = \sum_{x \in C_i} x / | C_i |$;
    //更新簇的平均值，即计算每个对象簇中对象的平均值
    (5) Compute E; //计算准则函数 E
    (6) UNTIL E 不再明显地发生变化;
  - 性能分析
    - 主要优点
      - 是解决聚类问题的一种经典算法，**简单、快速。**
      - 对处理大数据集，该算法是相对**可伸缩和高效率**的。
      - 当**结果簇是密集的**，它的**效果较好。**
    - 主要缺点
      - 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。
      - 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。
      - **不适合**于发现**非凸面形状的簇**或者**大小差别很大的簇。**对于**"噪声"和孤立点数据**是**敏感**的。
  - 例题

    见课本P181-182
- k-medoids算法
  - 定义：K 中心点算法中，每次迭代后的质点都是从**聚类的样本点中选取**，k中心点算法不采用簇中对象的平均值作为簇中心，而选用簇中**离平均值最近的对象**作为簇中心。这样划分方法仍然是基于最小化所有对象与其参照点之间的相异度之和的原则来执行的。

6. 层次聚类算法（AGNES算法，DIANA算法）
   - 定义：层次聚类方法对给定的数据集**进行层次的分解**，**直到满足某种条件为止。**具体又可分为**凝聚的、分裂的两种方案。**

- ***凝聚的层次聚类***是一种**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，直到所有的对象都在一个簇中，或者某个终止条件被满足，绝大多数层次聚类属于这一类，他们只是在簇间相似度的定义上有所不同。
- ***分裂的层次聚类***与凝聚的层次聚类相反，采用**自顶向下**的策略，它首先**将所有对象置于一个簇**，然后**逐渐细分为越来越小的簇**，直到每个对象自成一簇，或者达到了某个终结条件。

- AGNES算法
  - 定义：**自底向上凝聚的算法**，先**将每个对象作为一个簇**，然后这些簇**根据某些准则**（类间距离最近的两个点）被**一步步地合并**，直到某个终结条件被满足（达到定义的簇的数目）。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。**
  - 伪代码

    输入：包含n个对象的数据库，终止条件簇的数目k。
    输出：k个簇，达到终止条件规定簇数目。

    1. 将每个对象当成一个初始簇；
    2. REPEAT
    3. 根据两个簇中最近的数据点的距离找到最近的两个簇；
    4. 合并两个簇，生成新的簇的集合；
    5. UNTIL 达到定义的簇的数目；
  - 例题

    见课本P189-190

- DIANA算法
  - 定义：**自顶向下分裂的算法**，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（达到了某个希望的簇数目，或两个最近簇之间的距离超过了某个阈值）。
  - 簇的直径：在一个簇中的**任意两个数据点**的**欧氏距离中的最大值。**
  - 平均相异度（平均距离）

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

  - 伪代码
    - 输入：包含n个对象的数据库，终止条件簇的数目k。
    - 输出：k个簇，达到终止条件规定簇数目。

（1）将所有对象整个当成一个初始簇；
（2） FOR （i=1; i≠k; i++) DO BEGIN
（3） 在所有簇中挑出具有最大直径的簇C；
（4） 找出C中与其它点平均相异度最大的一个点p并把p放入splinter group，剩余的放在old party中；
（5）. REPEAT
（6） 在old party里找出到最近的splinter group中的点的距离不大于到old party中最近点的距离的点，并将该点加入splinter group。
（7） UNTIL 没有新的old party的点被分配给splinter group；
（8） splinter group和old party为被选中的簇分裂成的两个簇，与其它簇一起组成新的簇集合。
（9） END.

- 例题

  见课本P191-192

- 两者的异同★
  - 相同点
    - AGNES 和 DIANA **都属于层次聚类算法**，这意味着它们通过递归地划分或合并簇，构建一个层次结构，形成一个聚类树（或树状图）。
    - **都需要定义簇间的距离度量或相似性度量**，以确定合并或分裂的标准。
    - AGNES 和 DIANA **都能够形成一个树状结构**，其中每个节点代表一个簇，边表示合并或分裂的过程。
    - 两者**都无需事先指定聚类数目**，而是通过合并或分裂的过程自适应地形成不同层次的簇。
  - 不同点
    - **AGNES 采用自底向上**的策略，**DIANA 采用自顶向下**的策略。
    - **AGNES 的计算复杂度相对较高**，因为在每一步都需要计算所有簇对之间的距离。DIANA 在初始时计算较少的距离，但递归过程中可能需要计算的距离逐渐增多。
    - **AGNES 对于缺失值较为敏感**，因为在计算簇间距离时，需要考虑所有数据点的距离。DIANA 对于缺失值的影响相对较小，因为它主要关注簇内的距离。

7. 密度聚类方法（DBSCAN算法)
   - 指导思想：只要有一个区域中，**点的密度大于某个阈值**，就把它**加到与之相连的簇**中去。
   - 噪声点：不包含在任何簇中的对象被认为是“噪声”。
   - 边界点：落在某个核心点的邻域内，是一个稠密区域边缘上的点。（非核心对象点）
   - 阈值：包含多于MinPts个对象
   - 定义：DBSCAN是**噪声环境**下的**密度聚类算法**，将密度相连的点的**最大集合聚成簇**，并可在有“噪声”的空间数据库中发现任意形状的聚类。
   - 特点：事先不知道会有多少个簇
   - 相关概念

定义 5-3 对象的ε-临域：给定对象在半径ε内的区域。

定义 5-4 核心对象：如果一个对象的ε-临域至少包含最小数目MinPts个对象，则称该对象为核心对象。

定义 5-5 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。

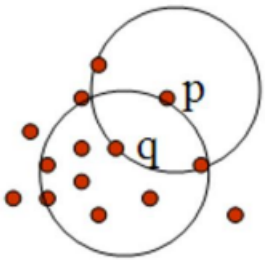


图5-6直接密度可达

定义 5-6 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$p_i \in D$，（$1<=i<=n$），$pi+1$是从p_i关于ε和MitPts直接密度可达的，则对象p是从对象q关于ε和MinPts间接密度可达。

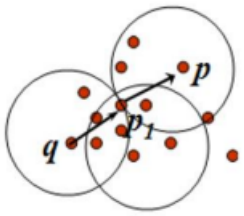


图5-7 密度可达

除了P点可以不是核心对象，其它对象链中的都必须是核心对象！

定义 5-7 密度相连：如果对象集合D中存在一个对象o，使得对象p和q是从o关于ε和MinPts间接密度可达，那么对象p和q是关于ε和MinPts密度相连。

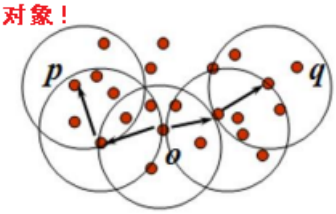


图5-8密度相连

- **步骤：**
  1. 先找到核心对象
  2. 找簇中的直接密度可达
  3. 在直接密度可达点中密度可达点（已经被包含在其他簇的点不用加入）
- 算法描述
  1. DBSCAN通过检查数据集中每点的Eps领域来搜索簇，如果一个点q的区域内包含多于MinPts个对象，则创建一个q作为核心对象的簇。
  2. 然后，反复地从这些核心对象中寻找直接密度可达的对象，把一些密度可达簇进行合并。
  3. 当没有新的点可以被添加到任何簇时，该过程结束。
- 伪代码
  - 输入：包含n个对象的数据库，半径ε，最少数目MinPts。
  - 输出：所有生成的簇，达到密度要求。

  1. REPEAT
  2. 从数据库中抽取一个未处理过的点；
  3. IF 抽出的点是核心点 THEN找出所有从该点密度可达的对象，形成一个簇
  4. ELSE 抽出的点是边缘点(非核心对象)，跳出本次循环，寻找下一点；
  5. UNTIL 所有点都被处理；
- 性能分析
  - 优点
    - 聚类**速度快**且能够**有效处理噪声点**和**发现任意形状的空间聚类**；
    - 与K-MEANS比较起来，**不需要输入要划分的聚类个数**；

        - 聚类簇的**形状没有偏倚**；
        - **对噪声数据不敏感。**
    - 缺点
        - 当**数据量增大**时，要求**较大的内存支持I/O消耗也很大**；
        - 当空间聚类的**密度不均匀**、聚类间**距差相差很大**时，聚类**质量较差**，因为这种情况下参数MinPts和 ϵ 选取困难（对半径和Minpoints敏感）。
        - 算法聚类效果**依赖与距离公式选取**， 实际应用中常用欧式距离，对于高维数据，存在“维数灾难”。
- 例题

  见课本P195-196

时间序列挖掘技术

1. 时间序列挖掘的含义★：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。
2. 时间序列挖掘的意义★：
    - 用于短期、中期或长期预测等**决策性工作**，**指导**人们的**社会、经济、军事和生活**等行为。
    - 时间序列数据挖掘通过**研究信息的时间特性**，**深入洞悉事物进化的机制**，是**获得知识的有效途径。**
3. 时间序列的分类★：
    - ***绝对数时间序列***：由时期总量指标排列而成的时间序列。
    - ***相对数时间序列***：把一系列同种相对数指标按时间先后顺序排列而成的时间序列叫做相对数时间序列。
    - ***平均数时间序列***：平均数时间序列是指由一系列同类平均指标按时间先后顺序排列的时间序列。
4. 时间序列挖掘的主要任务★：
    - 时间序列相似性搜索
    - 时间序列聚类
    - 时间序列分类
    - 时间序列分割与模式发现
    - 海量时间序列可视化
    - 时间序列预测
5. 应用场景：时间序列挖掘在**宏观的经济预测、市场营销、客流量分析、太阳黑子数、月降水量、河流流量、股票价格变动等**众多领域得到应用。事实上，**社会、科学、经济、技术等领域**中广泛存在着大量的时间序列数据有待进一步的分析和处理。

Web挖掘技术

1. Web挖掘的意义★
    - 从大量的信息中**发现用户感兴趣的信息**
    - 将Web上的丰富信息**转变成有用的知识**
    - 对用户**进行信息个性化**
2. Web挖掘的分类★

- ***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。
- ***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。
- ***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。

3. Web挖掘的含义★：是**数据挖掘在Web上的应用**，它**利用数据挖掘技术**从与WWW相关的资源和行为中**抽取感兴趣的、有用的模式和隐含信息。**

4. Web挖掘的主要数据源

   - Web服务器**日志**数据
   - Web上的**电子商务**数据
   - Web上的**网页**
   - Web上的网页之间的**链接**
   - Web上的**多媒体**数据

5. PageRank算法★

   - 核心思想

     - 如果**一个网页被很多其他网页链接到**的话说话这个网页比较重要，也就是**PageRank值会相对较高。**
     - 如果**一个PageRank值很高的网页链接到一个其他的网页**，那么**被链接到的网页的PageRank值**会相应地因此而**提高。**

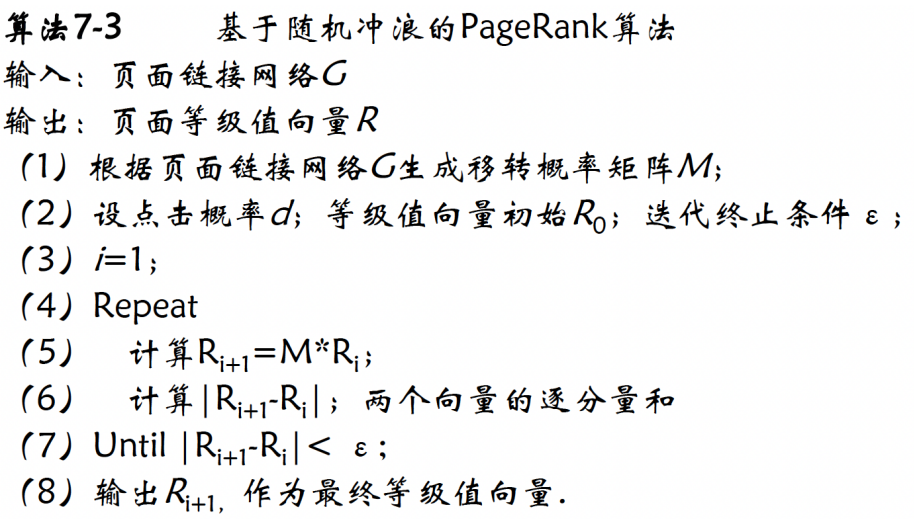
   - 算法步骤

     1. 计算每个网页一个PageRank（PR）值
     2. 通过（投票）算法不断迭代，直至达到平稳分布为止。
     3. 根据这个值的大小对网页的重要性进行排序。

     概括：计算每一个网页的PageRank值，然后根据这个值的大小对网页的重要性进行排序。

   - 基于随机冲浪的PageRank算法

     - 伪代码

     - 例题

       见课本P327-329

6. 权威页面和中心页面

- ***权威页面***：是指**包含需求信息的最佳资源页面。**是指**与某个领域或者某个话题相关的高质量网页**，比如搜索引擎领域，Google和百度首页即该领域的高质量网页，比如视频领域，优酷和土豆首页即该领域的高质量页面。
- ***中心页面***：是一个**包含权威页面链接的页面。**

空间数据挖掘技术

1. 空间挖掘的含义★：也称作空间数据挖掘，或者空间数据库的知识发现。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。**
2. 空间挖掘的意义★：
   - ***帮助理解空间数据***：通过挖掘空间数据，可以揭示地理空间中存在的模式和关系，帮助人们更好地理解空间数据。
   - ***地理决策支持***：在城市规划、资源管理、环境保护等领域，空间数据挖掘可以为决策提供有力的支持。
   - ***预测和规划***：通过挖掘历史空间数据，可以预测未来的趋势，并为规划提供参考。
   - ***应用于导航和位置服务***：空间数据挖掘对于导航、位置服务和地理信息系统等应用具有重要价值。

   总结：空间数据挖掘实质上是**空间信息技术发展的必然结果。**
3. 空间挖掘的分类★：
   - ***空间关系挖掘***：发现不同地理实体之间的空间关系，如邻近、重叠、包含等。
   - ***空间模式挖掘***： 发现在空间数据中出现的模式，如簇、群集、异常点等。
   - ***空间序列挖掘***：结合时间信息，挖掘地理空间中随时间变化的模式和趋势。
   - ***地理数据分类与聚类***：将地理空间数据进行分类和聚类，以识别不同地理区域或类型。
4. 空间挖掘的主要任务★：
   - 空间关系分析
   - 空间模式发现
   - 轨迹分析
   - 地理数据分类与聚类
   - 地理数据可视化