

第四章 分类方法

内容提要

- 分类的基本概念与步骤
- 基于距离的分类算法
- 决策树分类方法
- 贝叶斯分类
- 规则归纳
- 与分类有关的问题





ID3算法

- ID3算法：Iterative Dichotomiser 3，迭代二叉树3代，是一种贪心算法。
- ID3算法最早是由罗斯昆（J. Ross Quinlan）于1975年在悉尼大学提出的一种分类预测算法，算法的核心是“信息熵”。
- ID3算法通过计算每个属性的信息增益，认为信息增益高的是好属性，每次划分选取信息增益最高的属性为划分标准，重复这个过程，直至生成一个能完美分类训练样例的决策树。



■ 信息熵：

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

■ 信息增益的计算公式如下：

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

学生膳食结构和缺钙调查表

学生	鸡肉	猪肉	牛肉	羊肉	鱼肉	鸡蛋	青菜	番茄	牛奶	健康情况
1	0	1	1	0	1	0	1	0	1	不缺钙
2	0	0	0	0	1	1	1	1	1	不缺钙
3	1	1	1	1	1	0	1	0	0	缺钙
4	1	1	0	0	1	1	0	0	1	不缺钙
5	1	0	0	1	1	1	0	0	0	缺钙
6	1	1	1	0	0	1	0	1	0	缺钙
7	0	1	0	0	0	1	1	1	1	不缺钙
8	0	1	0	0	0	1	1	1	1	缺钙
9	0	1	0	0	0	1	1	1	1	不缺钙
10	1	0	1	1	1	1	0	1	1	不缺钙

学生膳食结构和缺钙调查表

学生	鸡肉	猪肉	牛肉	羊肉	鱼肉	鸡蛋	青菜	番茄	牛奶	健康情况
1	0	1	1	0	1	0	1	0	1	不缺钙
2	0	0	0	0	1	1	1	1	1	不缺钙
3	1	1	1	1	1	0	1	0	0	缺钙
4	1	1	0	0	1	1	0	0	1	不缺钙
5	1	0	0	1	1	1	0	0	0	缺钙
6	1	1	1	0	0	1	0	1	0	缺钙
7	0	1	0	0	0	1	1	1	1	不缺钙
8	0	1	0	0	0	1	1	1	1	缺钙
9	0	1	0	0	0	1	1	1	1	不缺钙
10	1	0	1	1	1	1	0	1	1	不缺钙



信息增益 (information gain)

- 对于分类系统来说，类别是变量，每一个类别出现的概率分别是 $P(C_1), P(C_2), \dots, P(C_n)$

此时**分类系统的熵**就可以表示为：

$$H(C) = - \sum_{i=1}^n P(C_i) \log_2 P(C_i)$$

- **信息增益**是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。



ID3算法的性能分析

- ID3算法的**优点**是：
- 算法的理论清晰，方法简单，学习能力较强。
- ID3算法的假设空间包含所有的决策树，它是关于现有属性的有限离散值函数的一个完整空间。所以ID3算法避免了搜索不完整假设空间的一个主要风险：假设空间可能不包含目标函数。
- ID3算法在搜索的每一步都使用当前的所有训练样例，大大降低了对个别训练样例错误的敏感性。因此，通过修改终止准则，可以容易地扩展到处理含有噪声的训练数据。



- ID3算法的**缺点**是：
- 只对比较小的数据集有效，且对噪声比较敏感，当训练数据集加大时，决策树可能会随之改变。
- ID3算法在搜索过程中不进行回溯。收敛到局部最优而不是全局最优。
- ID3算法只能处理离散值的属性。信息增益度量存在一个内在偏置，它偏袒具有较多值的属性。如日期属性。
- ID3算法增长树的每一个分支的深度，直到恰好能对训练样例完美地分类。当数据中有噪声或训练样例的数量太少时，产生的树会过渡拟合训练样例。

第四章 分类方法

内容提要

- 分类的基本概念与步骤
- 基于距离的分类算法
- 决策树分类方法：ID3、C4.5
- 贝叶斯分类
- 规则归纳
- 与分类有关的问题



C4.5算法对ID3的主要改进

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



- **增益比率** gain ratio: 增益度量 $\text{Gain}(S, A)$ 和分裂信息度量 $\text{SplitInformation}(S, A)$ 来共同定义的, 如下所示:

$$\text{GainRatio}(S, A) \equiv \frac{\text{Gain}(S, A)}{\text{SplitInformation}(S, A)}$$

- 其中, 分裂信息用来衡量属性分裂数据的广度和均匀, **信息分裂度**被定义为:

$$\text{SplitInformation}(S, A) \equiv - \sum_{i=1}^c \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

Outlook	Temperature	Humidity	Wind	PlayTennis
Sunny	Hot	85	false	No
Sunny	Hot	90	true	No
Overcast	Hot	78	false	Yes
Rain	Mild	96	false	Yes
Rain	Cool	70	true	No
Overcast	Cool	65	true	Yes
Sunny	Mild	95	false	No
Sunny	Cool	70	false	Yes
Rain	Mild	80	false	Yes
Sunny	Mild	70	true	Yes
Overcast	Mild	90	true	Yes
Overcast	Hot	75	false	Yes
Rain	Mild	80	true	No



C4.5算法例子

样本数据

Outlook	Temperature	Humidity	Wind	PlayTennis
Sunny	Hot	85	false	No
Sunny	Hot	90	true	No
Overcast	Hot	78	false	Yes
Rain	Mild	96	false	Yes
Rain	Cool	80	false	Yes
Rain	Cool	70	true	No
Overcast	Cool	65	true	Yes
Sunny	Mild	95	false	No
Sunny	Cool	70	false	Yes
Rain	Mild	80	false	Yes
Sunny	Mild	70	true	Yes
Overcast	Mild	90	true	Yes
Overcast	Hot	75	false	Yes
Rain	Mild	80	true	No

(1) 首先对 *Humidity* 进行属性离散化，针对上面的训练集合，通过检测每个划分而确定最好的划分在75处，则这个属性的范围就变为 { (≤ 75 , > 75) }。

(2) 计算目标属性 *PlayTennis* 分类的期望信息：

$$I(s_1, s_2) = I(9, 5) = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.940$$

Outlook	Temperature	Humidity	Wind	PlayTennis
Sunny	Hot	85	false	No
Sunny	Hot	90	true	No
Overcast	Hot	78	false	Yes
Rain	Mild	96	false	Yes
Rain	Cool	80	false	Yes
Rain	Cool	70	true	No
Overcast	Cool	65	true	Yes
Sunny	Mild	95	false	No
Sunny	Cool	70	false	Yes
Rain	Mild	80	false	Yes
Sunny	Mild	70	true	Yes
Overcast	Mild	90	true	Yes
Overcast	Hot	75	false	Yes
Rain	Mild	80	true	No

$$GainRatio (Outlook) = \frac{0.2467}{1.577} = 0.156 ;$$

$$GainRatio (windy) = 0.049 ;$$

$$GainRatio(Temperature) = 0.0248 ;$$

$$GainRatio(Humidity) = 0.0483$$

天气	温度	湿度	风速	活动
晴	炎热	高	弱	取消
晴	炎热	高	强	取消
阴	炎热	高	弱	进行
雨	适中	高	弱	进行
雨	寒冷	正常	弱	进行
雨	寒冷	正常	强	取消
阴	寒冷	正常	强	进行
晴	适中	高	弱	取消
晴	寒冷	正常	弱	进行
雨	适中	正常	弱	进行
晴	适中	正常	强	进行
阴	适中	高	强	进行
阴	炎热	正常	弱	进行
雨	适中	高	强	取消

天气	温度	湿度	风速	活动
晴	寒冷	正常	弱	进行
晴	适中	正常	强	进行
晴	炎热	高	弱	取消
晴	炎热	高	强	取消
晴	适中	高	弱	取消
阴	炎热	高	弱	进行
阴	寒冷	正常	强	进行
阴	适中	高	强	进行
阴	炎热	正常	弱	进行
雨	适中	高	弱	进行
雨	寒冷	正常	弱	进行
雨	适中	正常	弱	进行
雨	寒冷	正常	强	取消
雨	适中	高	强	取消

晴

温度	湿度	风速	活动
寒冷	正常	弱	进行
适中	正常	强	进行
炎热	高	弱	取消
炎热	高	强	取消
适中	高	弱	取消

阴

温度	湿度	风速	活动
炎热	高	弱	进行
寒冷	正常	强	进行
适中	高	强	进行
炎热	正常	弱	进行

雨

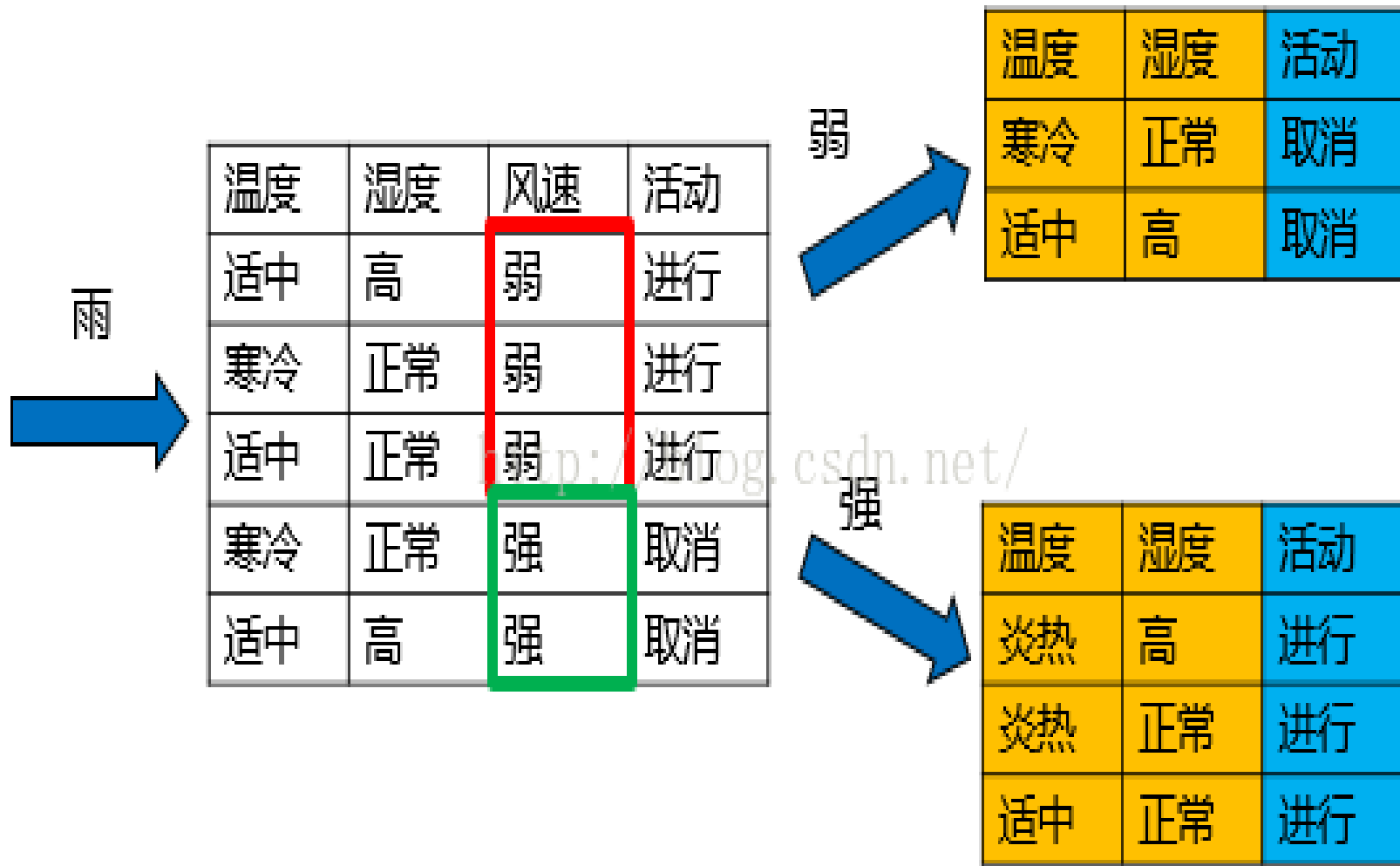
温度	湿度	风速	活动
适中	高	弱	进行
寒冷	正常	弱	进行
适中	正常	弱	进行
寒冷	正常	强	取消
适中	高	强	取消



<http://blog.csdn.net/xuxurui007>



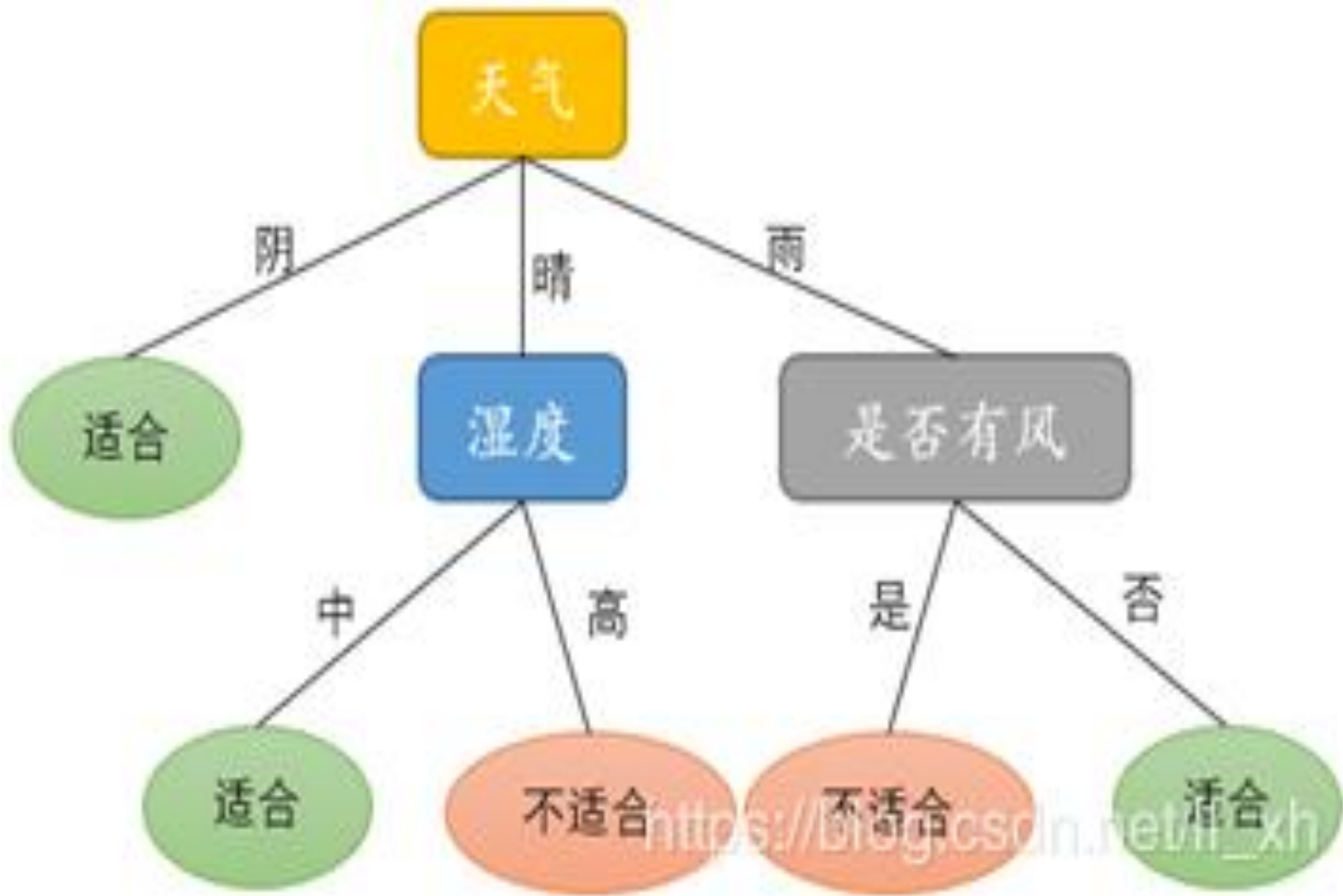
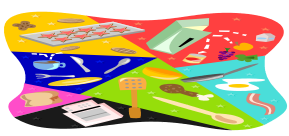
C4.5算法





晴

温度	湿度	风速	活动
寒冷	正常	弱	进行
适中	正常	强	进行
炎热	高	弱	取消
炎热	高	强	取消
适中	高	弱	取消



第三章 分类方法

内容提要

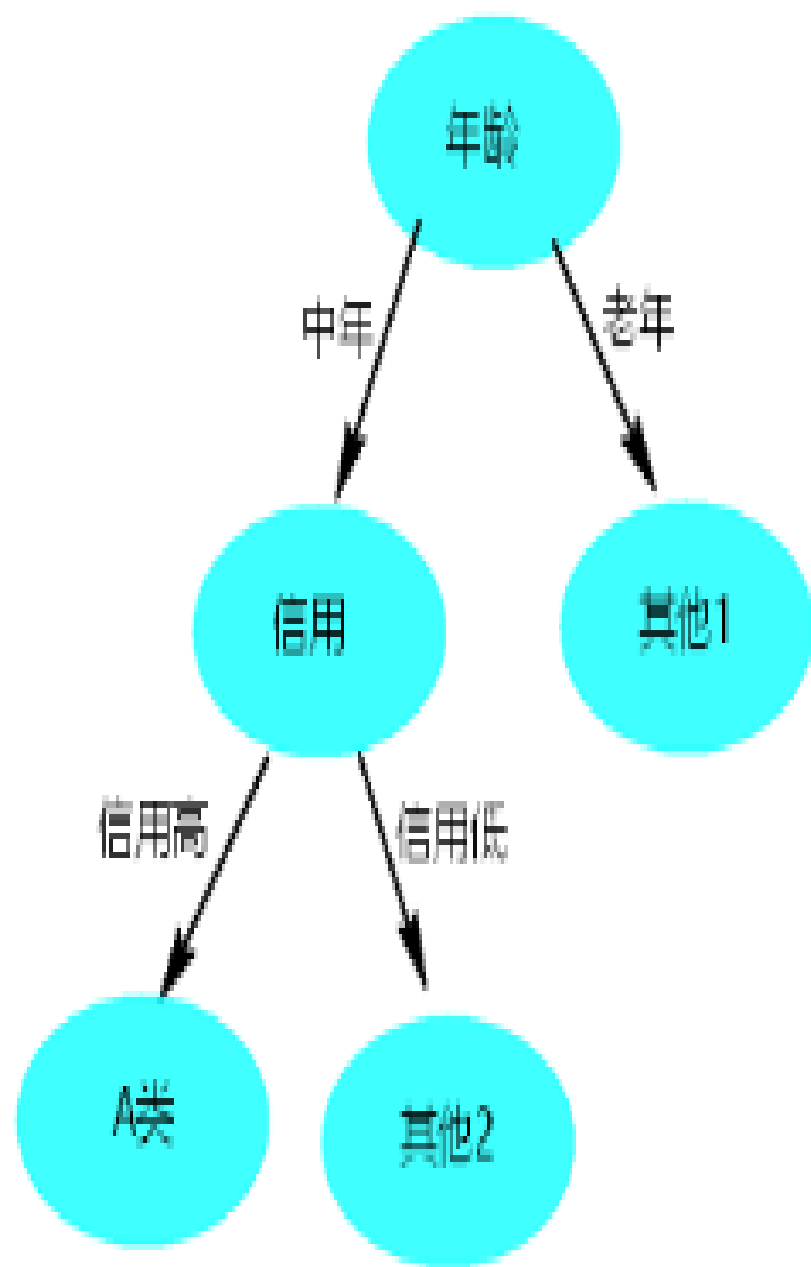
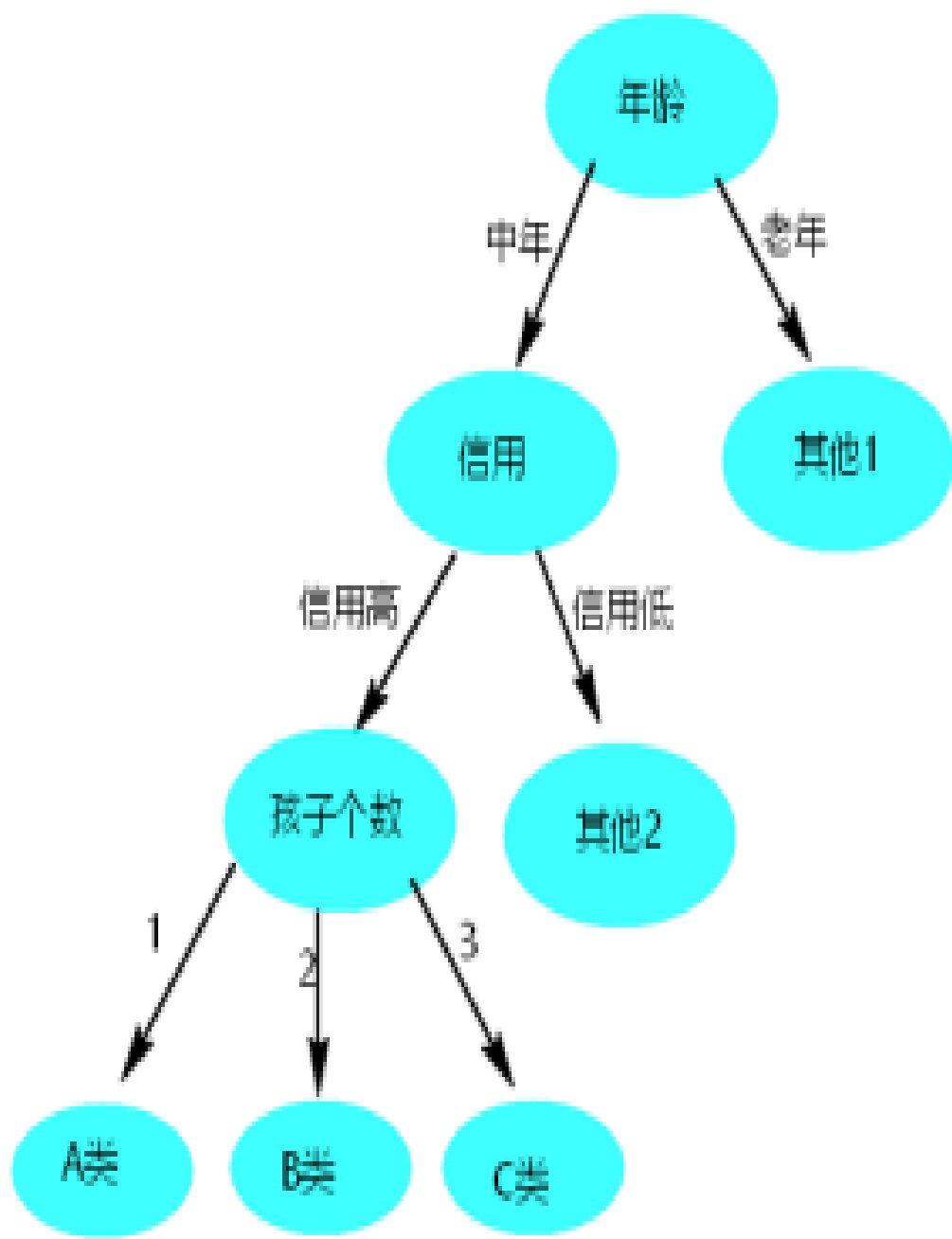
- 分类的基本概念与步骤
- 基于距离的分类算法
- 决策树分类方法
- 与决策树有关的问题
- 贝叶斯分类
- 规则归纳





决策树分类的特点

- 决策树分类方法采用自顶向下的递归方式，在决策树的内部结点进行属性值的比较并根据不同的属性值判断从该结点向下的分枝，在决策树的叶结点得到结论。所以从决策树的根到叶结点的一条路径就对应着一条**合取规则**，整棵决策树就对应着一组析取表达式规则。
- 决策树分类模型的建立通常分为两个步骤：
 - 决策树生成
 - 决策树修剪。





决策树修剪算法

■ 剪枝两种基本的剪枝策略：

- **预剪枝** (Pre-Pruning)：在构造决策树的同时进行剪枝。设定一个阈值，如决策树的高度等，使构造的决策树不能大于此阈值。
- **后剪枝** (Post-Pruning)：决策树构造完成后进行剪枝，从树的叶子开始剪枝，逐步向根的方向剪。剪枝的过程是对拥有同样父节点的一组节点进行检查，判断如果将其合并，熵的增加量是否小于某一个阈值。如果确实小，则这一组节点可以合并一个节点，其中包含了所有可能的结果。



预剪枝依据

- 作为叶结点需要含的最少样本个数
- 决策树的层数
- 结点的经验熵小于某个阈值才停止



后剪枝算法

- Reduced-Error Pruning (REP, 错误率降低剪枝)
- 若完全的决策树过度拟合, 则用一个测试数据集来纠正它。对于完全决策树中的每一个非叶子节点的子树, 尝试着把它替换成一个叶子节点, 该叶子节点的类别用子树所覆盖训练样本中存在最多的那个类来代替, 这样就产生了一个简化决策树, 然后比较这两个决策树在测试数据集中的表现, 如果简化决策树在测试数据集中的错误比较少, 那么该子树就可以替换成叶子节点。
- 该算法以bottom-up的方式遍历所有的子树, 直至没有任何子树可以替换使得测试数据集的表现得以改进时, 算法就可以终止。

第四章 分类方法

内容提要

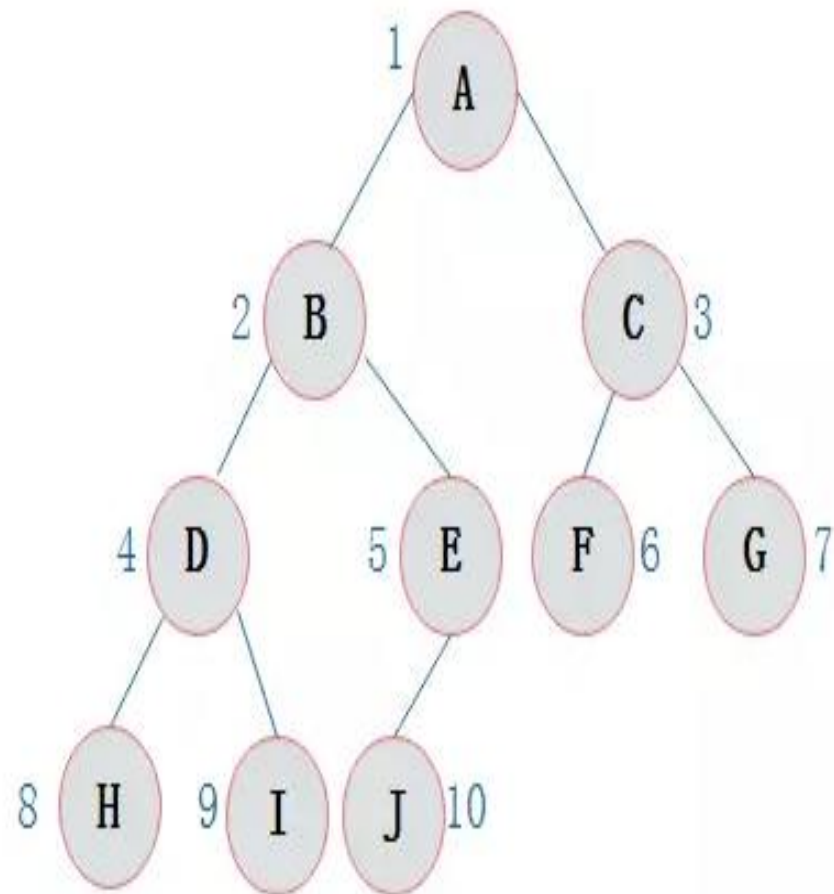
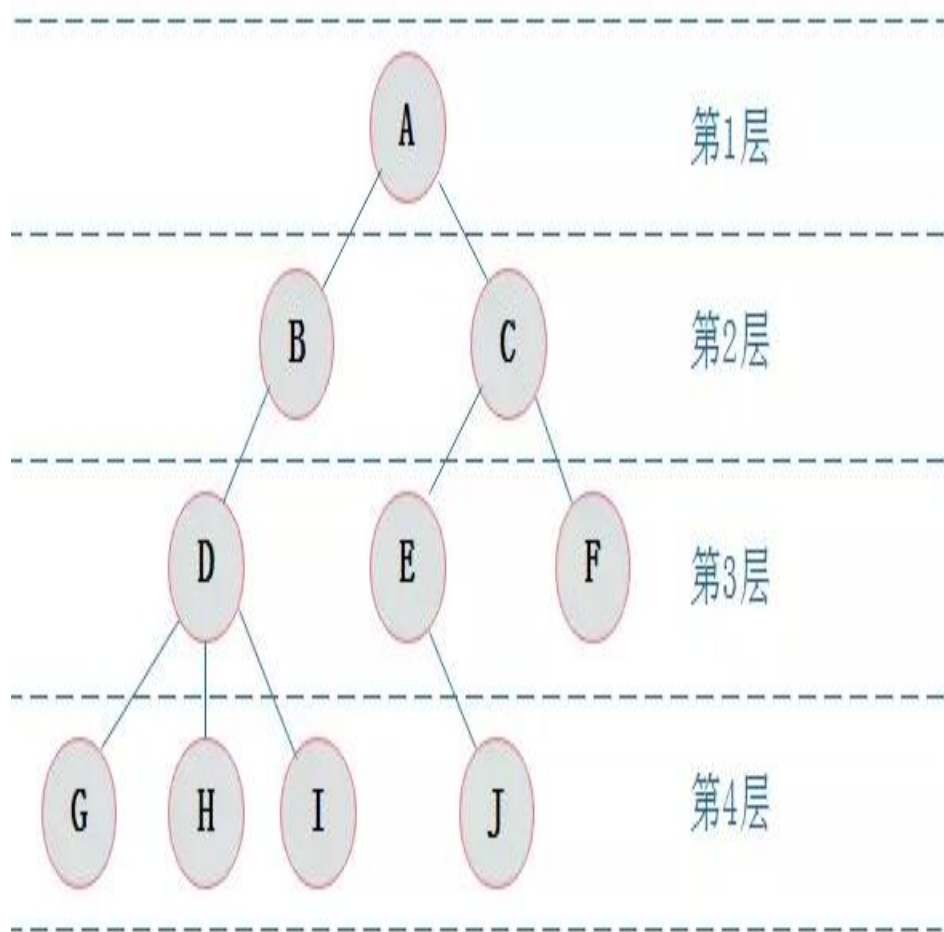
- 分类的基本概念与步骤
- 基于距离的分类算法
- 决策树分类方法：ID3、C4.5、CART
- 贝叶斯分类
- 规则归纳
- 与分类有关的问题



分类回归树(CART)

- 分类回归树(CART, Classification And Regression Tree)也属于一种决策树。分类回归树是一棵**二叉树**，且每一个非叶子节点都有两个孩子，所以对于第一棵子树其叶子节点数比非叶子节点数多1。
- CART树是二叉树，不像多叉树那样形成过多的数据碎片。
- **基尼指数**：样本被选中的概率*样本被分错的概率

$$\text{Gini}(D) = 1 - \sum_{k=1}^K \left(\frac{|C_k|}{|D|} \right)^2$$



1	2	3	4	5	6	7	8	9	10
A	B	C	D	E	F	G	H	I	J

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



- Outlook有三个属性值，因为CART是一个二叉树，我们把三个属性值按照2+1的组合（有三种）即

	Sunny and Rain	Overcast
Yes	5	4
No	5	0

$$GINI(Sunny \cup Rain) = 1 - \left(\frac{5}{10}\right)^2 - \left(\frac{5}{10}\right)^2$$

$$GINI = \frac{10}{14}GINI(Sunny \cup Rain) + \frac{4}{10}GINI(Overcast)$$



CART算法

	Sunny and Overcast	Rain
Yes	6	3
No	3	2

$$GINI(Sunny \cup Overcast) = 1 - \left(\frac{6}{9}\right)^2 - \left(\frac{3}{9}\right)^2$$

$$GINI = \frac{9}{14}GINI(Sunny \cup Overcast) + \frac{5}{14}GINI(Rain)$$



	Rain and Overcast	Sunny
Yes	7	2
No	2	3

$$GINI(Rain \cup Overcast) = 1 - \left(\frac{7}{9}\right)^2 - \left(\frac{2}{9}\right)^2$$

$$GINI = \frac{9}{14}GINI(Rain \cup Overcast) + \frac{5}{14}GINI(Sunny)$$



CART算法

- CART用Gini系数**最小**化准则来进行特征选择，生成二叉树。
- 输入：训练数据集 D ，停止计算的条件：
- 输出：CART决策树。
- 1、设结点的训练数据集为 D ，计算现有特征对该数据集的Gini系数。
- 2、在所有可能的特征 A 以及它们所有可能的切分点 a 中，选择Gini系数最小的特征及其对应的切分点作为最优特征与最优切分点。
- 3、对两个子结点递归地调用步骤1~2，直至满足停止条件。
- 4、生成CART决策树。



[fuquiui](#) / [lihang_algorithms](#)

[Code](#)[Issues](#) 0[Pull requests](#) 0[Projects](#) 0[Security](#)[Insights](#)

[lihang_algorithms](#) /

- [> AdaBoost/AdaBoost_sklearn.py](#)
- [README.md](#)
- [data/test.csv](#)
- [data/train.csv](#)
- [data/train_binary.csv](#)
- [decision_tree/C45.py](#)
- [decision_tree/ID3.py](#)
- [decision_tree/decision_tree_sklearn.py](#)
- [imgs/Adaboost_sklearn_result_1.png](#)
- [imgs/Adaboost_sklearn_result_2.png](#)
- [imgs/C45_result.png](#)
- [imgs/ID3_result.png](#)
- [imgs/decision_tree_sklearn_result.png](#)
- [imgs/knn_result.png](#)
- [imgs/knn_sklearn_result.png](#)
- [imgs/logistic_regression_result.png](#)



使用UCI数据集

■ UCI数据库是加州大学欧文分校(University of California Irvine)提出的用于机器学习的数据库，这个数据库目前共有335个数据集，其数目还在不断增加

以Iris鸢尾花数据集为例：

1. Iris数据集在右边方框【Most Popular Data Sets (hits since 2007)】中第一个。

UCI Machine Learning Repository
Center for Machine Learning and Intelligent Systems

About Citation Policy Donate a Data Set Contact

Search

Repository Web

[View ALL Data Sets](#)

Welcome to the UC Irvine Machine Learning Repository!


We currently maintain 394 data sets as a service to the machine learning community. You may [view all data sets](#) through our searchable interface. Our [old web site](#) is still available, for those who prefer the old format. For a general overview of the Repository, please visit our [About page](#). For information about citing data sets in publications, please read our [citation policy](#). If you wish to donate a data set, please consult our [donation policy](#). For any other questions, feel free to [contact the Repository librarians](#). We have also set up a [mirror site](#) for the Repository.

Supported By: In Collaboration With:

Latest News:

- 04-04-2013: Welcome to the new Repository admins Kevin Bache and Moshe Lichman!
- 03-01-2010: [Note](#) from donor regarding Netflix data
- 10-16-2009: Two new data sets have been added.
- 09-14-2009: Several data sets have been added.
- 07-23-2008: [Repository mirror](#) has been set up.
- 03-24-2008: New data sets have been added!
- 06-25-2007: Two new data sets have been added: UJI Pen Characters, MAGIC Gamma Telescope

Featured Data Set: Plants



Task: Clustering
Data Type: Multivariate
Attributes: 70
Instances: 22632

Data has been extracted from the USDA plants database. It contains all plants (species and genera) in the database and the states of USA and Canada where they occur.

Newest Data Sets:

- 08-28-2017: [UCI Burst Header Packet \(BHP\) flooding attack on Optical Burst Switching \(OBS\) Network](#)
- 07-23-2017: [UCI Eco-hotel](#)
- 07-23-2017: [UCI Las Vegas Strip](#)
- 07-20-2017: [UCI Parkinson Disease Spiral Drawings Using Digitized Graphics Tablet](#)
- 07-18-2017: [UCI PM2.5 Data of Five Chinese Cities](#)
- 07-16-2017: [UCI Sales Transactions Dataset Weekly](#)
- 06-29-2017: [UCI Data for Software Engineering Teamwork Assessment in Education Setting](#)
- 05-30-2017: [UCI chestnut - LARVIC](#)
- 05-24-2017: [UCI Epileptic Seizure Recognition](#)

Most Popular Data Sets (hits since 2007):

- 1521979: [Iris](#)
- 999594: [Adult](#)
- 760642: [Wine](#)
- 652904: [Car Evaluation](#)
- 585076: [Breast Cancer Wisconsin \(Diagnostic\)](#)
- 575930: [Forest Fires](#)
- 547848: [UCI Human Activity Recognition Using Smartphones](#)
- 529868: [Heart Disease](#)
- 523521: [Wine Quality](#)

<http://blog.csdn.net/shengchaohua163>



小结

- ID3: 特征划分基于信息增益
- C4.5: 特征划分基于信息增益率
- CART: 特征划分基于基尼指数
- 规则的产生: 一旦树被建立, 就可以把树转换成if-then规则。
 - 规则存储于一个二维数组中, 每一行代表树中的一个规则, 即从根到叶之间的一个路径。表中的每列存放着树中的结点。

第三章 分类方法

内容提要

- 分类的基本概念与步骤
- 基于距离的分类算法
- 决策树分类方法
- 贝叶斯分类
- 规则归纳
- 与分类有关的问题





贝叶斯原理

- 贝叶斯是用来描述两个条件概率直接的关系。

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

https://blog.csdn.net/weixin_37567451

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

https://blog.csdn.net/weixin_37567451

- 在分类中，把X理解成“具有某种特征”，把Y理解为“类别标签”

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

https://blog.csdn.net/weixin_37567451



贝叶斯分类

- 例如，假定数据样本域由**水果**组成，用它们的**颜色**和**形状**来描述。假定 X 表示红色和圆的， H 表示假定 X 是苹果，则 $P(H|X)$ 反映当我们看到 X 是红色并是圆的时，我们对 X 是苹果的确信程度。
- 贝叶斯分类器对两种数据具有较好的分类效果：一种是完全独立的数据，另一种是函数依赖的数据。



- 定义4-2 设 X 是类标号未知的数据样本。设 H 为某种假定，如数据样本 X 属于某特定的类 C 。对于分类问题，我们希望确定 $P(H|X)$ ，即给定观测数据样本 X ，假定 H 成立的概率。贝叶斯定理给出了如下计算 $P(H|X)$ 的简单有效的方法：

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

- $P(H)$ 是**先验概率**，或称 H 的先验概率。 $P(X|H)$ 代表假设 H 成立的情况下，观察到 X 的概率。 $P(H|X)$ 是**后验概率**，或称条件 X 下 H 的后验概率。



- **贝叶斯**方法使用概率统计的知识对样本数据集进行分类，特点是**结合先验概率和后验概率**，即避免了只使用先验概率的主管偏见，也避免了单独使用样本信息的过拟合现象。
- **朴素贝叶斯法**是基于贝叶斯定理与**特征条件独立**假设的分类方法。即假定给定目标值时属性之间相互条件独立。也就是说没有哪个属性变量对于决策结果来说占有着较大的比重，也没有哪个属性变量对于决策结果占有着较小的比重。虽然这个简化方式在一定程度上降低了贝叶斯分类算法的分类效果，但是在实际的应用场景中，极大地简化了贝叶斯方法的复杂性



贝叶斯分类

- 设 X, Y 为两个随机变量，得到贝叶斯公式：

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

https://blog.csdn.net/weixin_37567451

- $P(Y)$ 叫做先验概率， $P(Y|X)$ 叫做后验概率， $P(Y, X)$ 是联合概率。
- **先验概率** (prior probability) 是指根据以往经验和分析得到的概率。
- **后验概率** (posterior probability)，事情已经发生了，事情发生可能有很多原因，判断事情发生时由哪个原因引起的概率。
- **联合概率** (joint probability) 是指表示两个事件共同发生的概率。



朴素贝叶斯分类

- 朴素贝叶斯分类的工作过程如下：
- (1) 每个数据样本用一个 n 维特征向量 $X = \{x_1, x_2, \dots, x_n\}$ 表示，分别描述对 n 个属性 A_1, A_2, \dots, A_n 样本的 n 个度量。
- (2) 假定有 m 个类 C_1, C_2, \dots, C_m ，给定一个未知的数据样本 X （即没有类标号），分类器将预测 X 属于具有最高后验概率（条件 X 下）的类。也就是说，朴素贝叶斯分类将未知的样本分配给类 C_i ($1 \leq i \leq m$) 当且仅当 $P(C_i | X) > P(C_j | X)$ ，对任意的 $j = 1, 2, \dots, m, j \neq i$ 。这样，最大化 $P(C_i | X)$ 。其 $P(C_i | X)$ 最大的类 C_i 称为最大后验假定。根据贝叶斯定理

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



朴素贝叶斯分类(续)

- (3) 由于 $P(X)$ 对于所有类为**常数**，只需要 $P(X|C_i)*P(C_i)$ 最大即可。如果 C_i 类的先验概率未知，则通常假定这些类是等概率的，即 $P(C_1)=P(C_2)=\dots=P(C_m)$ ，因此问题就转换为对 $P(X|C_i)$ 的最大化（ $P(X|C_i)$ 常被称为给定 C_i 时数据 X 的似然度，而使 $P(X|C_i)$ 最大的假设 C_i 称为**最大似然假设**）。否则，需要最大化 $P(X|C_i)*P(C_i)$ 。注意，类的先验概率可以用 $P(C_i)=s_i/s$ 计算，其中 s_i 是类 C_i 中的训练样本数，而 s 是训练样本总数。
- (4) 给定具有许多属性的数据集，计算 $P(X|C_i)$ 的开销可能非常大。为降低计算 $P(X|C_i)$ 的开销，可以做**类条件独立的朴素假定**。给定样本的类标号，假定属性值相互条件独立，即在属性间，不存在依赖关系。这样

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



朴素贝叶斯分类(续)

其中概率 $P(x_1 | C_j)$, $P(x_2 | C_j)$, ..., $P(x_n | C_j)$ 可以由训练样本估值。

- 如果 A_k 是离散属性, 则 $P(x_k | C_i) = s_{ik} / s_i$, 其中 s_{ik} 是在属性 A_k 上具有值 x_k 的类 C_i 的训练样本数, 而 s_i 是 C_i 中的训练样本数。
- 如果 A_k 是连续值属性, 则通常假定该属性服从高斯分布。因而,

$$P(x_k | C_i) = g(x_k, \mu_{c_i}, \sigma_{c_i}) = \frac{1}{\sqrt{2\pi}\sigma_{c_i}} e^{-\frac{(x_k - \mu_{c_i})^2}{2\sigma_{c_i}^2}}$$

$g(x_k, \mu_{c_i}, \sigma_{c_i})$ 是高斯分布函数, μ_{c_i}, σ_{c_i} 而分别为平均值和标准差。

- (5) 对未知样本 X 分类, 也就是对每个类 C_i , 计算 $P(X | C_i) * P(C_i)$ 。样本 X 被指派到类 C_i , 当且仅当 $P(C_i | X) > P(C_j | X)$, $1 \leq j \leq m$, $j \neq i$, 换言之, X 被指派到其 $P(X | C_i) * P(C_i)$ 最大的类。

表 4.3 西瓜数据集 3.0

编号	色泽	根蒂	敲声	纹理	脐部	触感	密度	含糖率	好瓜
1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	0.697	0.460	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	0.774	0.376	是
3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	0.634	0.264	是
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	0.608	0.318	是
5	浅白	蜷缩	浊响	清晰	凹陷	硬滑	0.556	0.215	是
6	青绿	稍蜷	浊响	清晰	稍凹	软粘	0.403	0.237	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	0.481	0.149	是
8	乌黑	稍蜷	浊响	清晰	稍凹	硬滑	0.437	0.211	是
9	乌黑	稍蜷	沉闷	稍糊	稍凹	硬滑	0.666	0.091	否
10	青绿	硬挺	清脆	清晰	平坦	软粘	0.243	0.267	否
11	浅白	硬挺	清脆	模糊	平坦	硬滑	0.245	0.057	否
12	浅白	蜷缩	浊响	模糊	平坦	软粘	0.343	0.099	否
13	青绿	稍蜷	浊响	稍糊	凹陷	硬滑	0.639	0.161	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	0.657	0.198	否
15	乌黑	稍蜷	浊响	清晰	稍凹	软粘	0.360	0.370	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	0.593	0.042	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	0.719	0.103	否



- 判断一个具有特征{色泽=青绿，根蒂=蜷缩，敲声=浊响，纹理=清晰，脐部=凹陷，触感=硬滑，密度=0.697，含糖率=0.460}的测试样例瓜是否为好瓜。
- 估计先验概率 $P(c)$
- $P(\text{好瓜}=\text{是})=8/17=0.471$
- $P(\text{好瓜}=\text{否})=9/17=0.529$



计算每个属性值的条件概率

$$P_{\text{青绿}|\text{是}} = P(\text{色泽} = \text{青绿} | \text{好瓜} = \text{是}) = \frac{3}{8} = 0.375,$$

$$P_{\text{青绿}|\text{否}} = P(\text{色泽} = \text{青绿} | \text{好瓜} = \text{否}) = \frac{3}{9} \approx 0.333,$$

$$P_{\text{蜷缩}|\text{是}} = P(\text{根蒂} = \text{蜷缩} | \text{好瓜} = \text{是}) = \frac{5}{8} = 0.375,$$

$$P_{\text{蜷缩}|\text{否}} = P(\text{根蒂} = \text{蜷缩} | \text{好瓜} = \text{否}) = \frac{3}{9} \approx 0.333,$$

$$P_{\text{浊响}|\text{是}} = P(\text{敲声} = \text{浊响} | \text{好瓜} = \text{是}) = \frac{6}{8} = 0.750,$$

$$P_{\text{浊响}|\text{否}} = P(\text{敲声} = \text{浊响} | \text{好瓜} = \text{否}) = \frac{4}{9} \approx 0.444,$$

$$P_{\text{清晰}|\text{是}} = P(\text{纹理} = \text{清晰} | \text{好瓜} = \text{是}) = \frac{7}{8} = 0.875,$$

$$P_{\text{清晰}|\text{否}} = P(\text{纹理} = \text{清晰} | \text{好瓜} = \text{否}) = \frac{2}{9} \approx 0.222,$$

$$P_{\text{凹陷}|\text{是}} = P(\text{脐部} = \text{凹陷} | \text{好瓜} = \text{是}) = \frac{6}{8} = 0.750,$$

$$P_{\text{凹陷}|\text{否}} = P(\text{脐部} = \text{凹陷} | \text{好瓜} = \text{否}) = \frac{2}{9} \approx 0.222,$$

$$P_{\text{硬滑}|\text{是}} = P(\text{触感} = \text{硬滑} | \text{好瓜} = \text{是}) = \frac{6}{8} = 0.750,$$

$$P_{\text{硬滑}|\text{否}} = P(\text{触感} = \text{硬滑} | \text{好瓜} = \text{否}) = \frac{6}{9} \approx 0.667,$$



对于连续值属性:

$$P_{\text{密度: 0.697|是}} = p(\text{密度} = 0.697 \mid \text{好瓜} = \text{是})$$

$$= \frac{1}{\sqrt{2\pi} \cdot 0.129} \exp \left(-\frac{(0.697 - 0.574)^2}{2 \cdot 0.129^2} \right) \approx 1.959 ,$$

$$P_{\text{密度: 0.697|否}} = p(\text{密度} = 0.697 \mid \text{好瓜} = \text{否})$$

$$= \frac{1}{\sqrt{2\pi} \cdot 0.195} \exp \left(-\frac{(0.697 - 0.496)^2}{2 \cdot 0.195^2} \right) \approx 1.203 ,$$

$$P_{\text{含糖: 0.460|是}} = p(\text{含糖率} = 0.460 \mid \text{好瓜} = \text{是})$$

$$= \frac{1}{\sqrt{2\pi} \cdot 0.101} \exp \left(-\frac{(0.460 - 0.279)^2}{2 \cdot 0.101^2} \right) \approx 0.788 ,$$

$$P_{\text{含糖: 0.460|否}} = p(\text{含糖率} = 0.460 \mid \text{好瓜} = \text{否})$$

$$= \frac{1}{\sqrt{2\pi} \cdot 0.108} \exp \left(-\frac{(0.460 - 0.154)^2}{2 \cdot 0.108^2} \right) \approx 0.066 .$$



- 计算待测试瓜分别属于好瓜和坏瓜的概率：

$$P(\text{好瓜} = \text{是}) \times P_{\text{青绿}|\text{是}} \times P_{\text{蜷缩}|\text{是}} \times P_{\text{浊响}|\text{是}} \times P_{\text{清晰}|\text{是}} \times P_{\text{凹陷}|\text{是}} \\ \times P_{\text{硬滑}|\text{是}} \times P_{\text{密度: 0.697}|\text{是}} \times P_{\text{含糖: 0.460}|\text{是}} \approx 0.038,$$

$$P(\text{好瓜} = \text{否}) \times P_{\text{青绿}|\text{否}} \times P_{\text{蜷缩}|\text{否}} \times P_{\text{浊响}|\text{否}} \times P_{\text{清晰}|\text{否}} \times P_{\text{凹陷}|\text{否}} \\ \times P_{\text{硬滑}|\text{否}} \times P_{\text{密度: 0.697}|\text{否}} \times P_{\text{含糖: 0.460}|\text{否}} \approx 6.80 \times 10^{-5}.$$

- 好瓜的概率 远大于 坏瓜的概率，于是分类器判断测试瓜为好瓜。



朴素贝叶斯模型

- **高斯模型**：处理特征是连续型变量的情况
- **多项式模型**：最常见，要求特征是离散数据
- **伯努利模型**：要求特征是离散的，且为布尔类型，即true和false，或者1和0



朴素贝叶斯分类举例

样本数据

Age	income	student	credit_rating	buys_computer
<=30	High	No	Fair	No
<=30	High	No	Excellent	No
31...40	High	No	Fair	Yes
>40	Medium	No	Fair	Yes
>40	Low	Yes	Fair	Yes
>40	Low	Yes	Excellent	No
31...40	Low	Yes	Excellent	Yes
<=30	Medium	No	Fair	No
<=30	Low	Yes	Fair	Yes
>40	Medium	Yes	Fair	Yes
<=30	Medium	Yes	Excellent	Yes
31...40	Medium	No	Excellent	Yes
31...40	High	Yes	Fair	Yes
>40	Medium	No	Excellent	No

数据样本用属性age, income, student和credit_rating描述。类标号属性buys_computer具有两个不同值（即{yes, no}）。设 C_1 对应于类buys_computer="yes", 而 C_2 对应于类buys_computer="no"。

我们希望分类的未知样本为:

$X = (\text{age} = "<=30", \text{income} = \text{"medium", student} = \text{"yes", credit_rating} = \text{"fair"})$ 。

(1) 我们需要最大化 $P(X|C_i) \cdot P(C_i)$, $i=1, 2$ 。每个类的先验概率 $P(C_i)$ 可以根据训练样本计算:

$$P(\text{buys_computer} = \text{"yes"}) = 9/14 = 0.643,$$

$$P(\text{buys_computer} = \text{"no"}) = 5/14 = 0.357.$$

(2) 为计算 $P(X|C_i)$, $i=1, 2$, 我们计算下面的条件概率:

$$P(\text{age} \leq 30 | \text{buys_computer} = \text{"yes"}) = 2/9 = 0.222,$$

$$P(\text{age} \leq 30 | \text{buys_computer} = \text{"no"}) = 3/5 = 0.600,$$

$$P(\text{income} = \text{"medium"} | \text{buys_computer} = \text{"yes"}) = 4/9 = 0.444,$$

$$P(\text{income} = \text{"medium"} | \text{buys_computer} = \text{"no"}) = 2/5 = 0.400,$$

$$P(\text{student} = \text{"yes"} | \text{buys_computer} = \text{"yes"}) = 6/9 = 0.677,$$

$$P(\text{student} = \text{"yes"} | \text{buys_computer} = \text{"no"}) = 1/5 = 0.200,$$

$$P(\text{credit_rating} = \text{"fair"} | \text{buys_computer} = \text{"yes"}) = 6/9 = 0.667,$$

$$P(\text{credit_rating} = \text{"fair"} | \text{buys_computer} = \text{"no"}) = 2/5 = 0.400.$$

(3) 假设条件独立性, 使用以上概率, 我们得到:

$$P(X | \text{buys_computer} = \text{"yes"}) = 0.222 * 0.444 * 0.667 * 0.667 = 0.044,$$

$$P(X | \text{buys_computer} = \text{"no"}) = 0.600 * 0.400 * 0.200 * 0.400 = 0.019,$$

$$P(X | \text{buys_computer} = \text{"yes"}) * P(\text{buys_computer} = \text{"yes"}) = 0.044 * 0.643 = 0.028$$

$$P(X | \text{buys_computer} = \text{"no"}) * P(\text{buys_computer} = \text{"no"}) = 0.019 * 0.357 = 0.007.$$

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

4. 朴素贝叶斯算法过程

以参数估计为极大似然估计为例:

样本集为:

$$D = \{(x_1^{(1)}, x_2^{(1)}, \dots, x_n^{(1)}, y_1), (x_1^{(2)}, x_2^{(2)}, \dots, x_n^{(2)}, y_2), \dots, (x_1^{(m)}, x_2^{(m)}, \dots, x_n^{(m)}, y_m)\}$$

其中 $y_i, i = 1, 2, \dots, m$ 表示样本类别, 取值为 $\{C_1, C_2, \dots, C_K\}$ 。

(1) 计算先验概率: 求出样本类别的个数 K 。对于每一个样本 $Y = C_k$, 计算出 $P(Y = C_k)$ 。其为类别 C_k 在总样本集中的频率。

(2) 计算条件概率: 将样本集划分成 K 个子样本集, 分别对属于 C_k 的子样本集进行计算, 计算出其中特征 $X_j = a_{jl}$ 的概率: $P(X_j = a_{jl} | Y = C_k)$ 。其为该子集中特征取值为 a_{jl} 的样本数与该子集样本数的比值。

(3) 针对待预测样本 x^{test} , 计算其对于每个类别 C_k 的后验概率:
 $P(Y = C_k | X = x^{test}) = P(Y = C_k) \prod_{j=1}^n P(X_j = x_j^{test} | Y = C_k)$ 。概率值最大的类别即为待预测样本的预测类别。

第三章 分类方法

内容提要

- 分类的基本概念与步骤
- 基于距离的分类算法
- 决策树分类方法
- 贝叶斯分类
- EM算法
- 规则归纳
- 与分类有关的问题





EM算法举例

食堂的大师傅炒了一份菜，要等分成两份给两个人吃

——显然没有必要拿来天平一点一点的精确的去称分量，最简单的办法是先随意的把菜分到两个碗中，然后观察是否一样多，把比较多的那一份取出一点放到另一个碗中，这个过程一直迭代地执行下去，直到大家看不出两个碗所容纳的菜有什么分量上的不同为止



EM算法举例

- 假设男、女身高都服从正态分布，通过抽样调查男、女群体的身高平均值。
- 男人身高(cm): 170,180,180,190
- 女人身高(cm): 150,160,160,170
- 如果出现了意外，把抽样信息中男女的**标记给弄丢了**，男女身高数据混在了一起，那么还有没有办法把男女身高的平均值分别求出来呢？

身高 (x)	150	160	160	170	170	180	180	190
男人数目 (M)	?	?	?	?	?	?	?	?
女人数目 (F)	?	?	?	?	?	?	?	?



假设男、女身高均值分别为 μ_1 、 μ_2 ，这两个数的初值可以赋予任意两个不同的随机数，例如我们令初值为： $\mu_1=190$ ， $\mu_2=150$

- 根据这个初值，我们来重新估计每个 x_i 对应的 M_i 和 F_i 的期望值。假设男人和女人的概率密度函数分别为 p_m 和 p_f ，同一个身高数据 x_i 对应的男、女人数计算如下：

$$M_i = \frac{p_m(x_i)}{p_m(x_i) + p_f(x_i)} \quad F_i = \frac{p_f(x_i)}{p_m(x_i) + p_f(x_i)}$$

- 接下来更新 μ_1 、 μ_2 的值了，计算方法就是总身高除以总人数，算式如下：

$$\mu_1 = \frac{M_1 x_1 + \dots + M_8 x_8}{M_1 + \dots + M_8} \quad \mu_2 = \frac{F_1 x_1 + \dots + F_8 x_8}{F_1 + \dots + F_8}$$

- 重复迭代上面的过程，直到 μ_1 、 μ_2 收敛为止



EM算法

- 最大期望算法 (Expectation-maximization algorithm) 在概率模型中寻找参数最大似然估计或者最大后验估计的算法。用于寻找，依赖于不可观察的隐性变量的概率模型中，参数的最大似然估计。
- 最大期望算法经过两个步骤交替进行计算，第一步是**计算期望** (E)，利用对隐藏变量的现有估计值，计算其最大似然估计值；第二步是**最大化** (M)。最大化在E步上求得的最大似然值来计算参数的值。M步上找到的参数估计值被用于下一个E步计算中，这个过程不断交替进行。



a Maximum likelihood



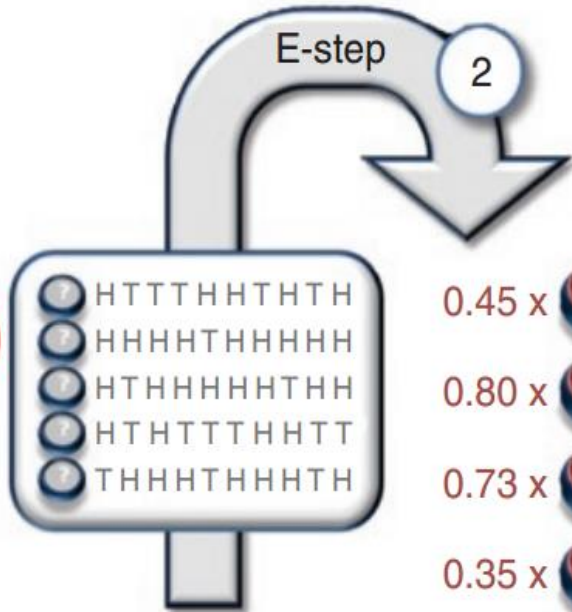
5 sets, 10 tosses per set

Coin A	Coin B
	5 H, 5 T
9 H, 1 T	
8 H, 2 T	
	4 H, 6 T
7 H, 3 T	
24 H, 6 T	9 H, 11 T

$$\hat{\theta}_A = \frac{24}{24 + 6} = 0.80$$

$$\hat{\theta}_B = \frac{9}{9 + 11} = 0.45$$

b Expectation maximization

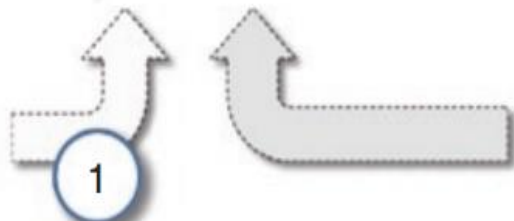


$$\hat{\theta}_A^{(0)} = 0.60$$

$$\hat{\theta}_B^{(0)} = 0.50$$

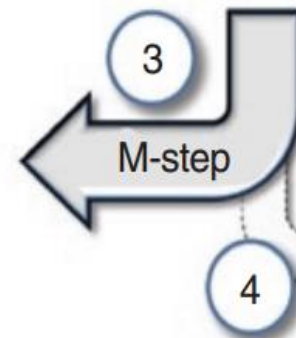


Coin A	Coin B
$\approx 2.2 \text{ H}, 2.2 \text{ T}$	$\approx 2.8 \text{ H}, 2.8 \text{ T}$
$\approx 7.2 \text{ H}, 0.8 \text{ T}$	$\approx 1.8 \text{ H}, 0.2 \text{ T}$
$\approx 5.9 \text{ H}, 1.5 \text{ T}$	$\approx 2.1 \text{ H}, 0.5 \text{ T}$
$\approx 1.4 \text{ H}, 2.1 \text{ T}$	$\approx 2.6 \text{ H}, 3.9 \text{ T}$
$\approx 4.5 \text{ H}, 1.9 \text{ T}$	$\approx 2.5 \text{ H}, 1.1 \text{ T}$
$\approx 21.3 \text{ H}, 8.6 \text{ T}$	$\approx 11.7 \text{ H}, 8.4 \text{ T}$



$$\hat{\theta}_A^{(1)} \approx \frac{21.3}{21.3 + 8.6} \approx 0.71$$

$$\hat{\theta}_B^{(1)} \approx \frac{11.7}{11.7 + 8.4} \approx 0.58$$



$$\hat{\theta}_A^{(10)} \approx 0.80$$

$$\hat{\theta}_B^{(10)} \approx 0.52$$



- EM的算法流程例如以下：
- 1、初始化分布参数
- 2、反复直到收敛：
 - E步骤：预计未知参数的期望值，给出当前的参数预计。
 - M步骤：又一次预计分布参数，以使得数据的似然性最大，给出未知变量的期望预计。

第三章 分类方法

内容提要

- 分类的基本概念与步骤
- 基于距离的分类算法
- 决策树分类方法
- 贝叶斯分类
- 规则归纳
- 与分类有关的问题





- **规则归纳**是机器学习的一个领域，是从观察集中将形式规则提取出来。提取的规则可能代表了全面的科学数据模型，或者只是代表了数据的本地模式。
- IF[物品是冷冻食品], THEN[将物品放在冷冻袋中], ELES[将物品放在购物袋中]



- 常见的采用规则表示的分类器构造方法有：
 - 利用规则归纳技术直接生成规则
 - 利用决策树方法先生成决策树，然后再把决策树转换为规则；
 - 使用粗糙集方法生成规则；
 - 使用遗传算法中的分类器技术生成规则等。
- 本节将只讨论规则归纳方法。我们这里讨论的规则归纳算法，可以直接学习规则集合，这一点与决策树方法、遗传算法有两点关键的不同。
 - 它们可学习包含变量的一阶规则集合：这一点很重要，因为一阶子句的表达能力比命题规则要强得多。
 - 这里讨论的算法使用序列覆盖算法：一次学习一个规则，以递增的方式形成最终的规则集合。



- 规则归纳有四种策略：减法、加法，先加后减、先减后加策略。
 - 减法策略：以具体例子为出发点，对例子进行推广或泛化，推广即减除条件（属性值）或减除合取项（为了方便，我们不考虑增加析取项的推广），使推广后的例子或规则不覆盖任何反例。
 - 加法策略：起始假设规则的条件部分为空（永真规则），如果该规则覆盖了反例，则不停地向规则增加条件或合取项，直到该规则不再覆盖反例。
 - 先加后减策略：由于属性间存在相关性，因此可能某个条件的加入会导致前面加入的条件没什么作用，因此需要减除前面的条件。
 - 先减后加策略：道理同先加后减，也是为了处理属性间的相关性。
- 典型的规则归纳算法有AQ、CN2和FOIL等。



- AQ 算法是一种覆盖算法 (Covering Algorithms)，即通过用规则去覆盖样例数据，寻找覆盖特定目标值的规则，然后对每个目标值学习一个析取规则集。

1969年，Michalski提出了AQ学习算法，这是一种基于实例的学习方法。AQ算法生成的选择假设的析取，覆盖全部正例，而不覆盖任何反例。

1978年提出了AQ11

AQ18

AQ19

简单的AQ学习算法

- (1) 集中注意一个实例(作为种子);
- (2) 生成该实例的一致性泛化式(称作star);
- (3) 根据偏好标准, 从star选择最优的泛化式(假设)。如果需要, 特化该假设;
- (4) 如果该假设覆盖了全部实例, 则停止; 否则选择一个未被假设覆盖的实例, 转到 (2)。



- AQ算法利用覆盖所有正例，排斥所有反例的思想来寻找规则。比较典型的有Michalski的AQ11方法，洪家荣改进的AQ15方法以及洪家荣的AE5方法。
- AQR是一个基于最基本AQ算法的归纳算法。其实，在很多的算法中，都采用了基本AQ算法，象AQ11和GEM。但和其它方法比较而言，AQR更加简单一些，如在AQ11中使用一种复杂的包括置信度的规则推导方法。下面我们首先对AQR算法进行概念描述，然后介绍算法描述和相关例子，最后分析其性能。



AQR算法有关定义

- AQR为每一个分类推导出一条规则，每一条规则形式如下：
if <cover> then predict <class>。
- 在一个属性上的基本测试被称为一个Selector。下面是一些Selector的例子：<Cloudy=yes> 或 <Temp>60>。
- AQR允许测试做{=, ≤, ≥, ≠}。Selectors的合取被称为复合（Complex），Complexes之间的析取被称为覆盖（Cover）。如果一个表达式对某个样本为真，则我们称其为对这个样本的一个覆盖。这样，一个空Complex覆盖所有的样本，而一个空Cover不覆盖任何样本。
- 在AQR中，一个新样本被区分是看其属于哪个推导出来的规则。如果该样本只满足一条规则，则这个样本就属于这条规则；如果该样本满足多条规则，则被这些规则所预测的最频繁的分类被赋予这条规则；如果该样本不属于任何规则，则其分类为样本集中最频繁的分类。



算法 4-5 AQR

输入：正例样本POS；

反例样本NEG。

输出：覆盖COVER。

- (1) COVER = Φ ; //初始化COVER为空集 Φ
- (2) WHILE COVER does not cover all positive examples in POS DO BEGIN
- (3) Select a SEED; /选取一个种子SEED，例如没有被COVER覆盖的一个正样例
- (4) Call procedure STAR (SEED, NEG); //产生一个能覆盖种子而同时排除所有反例的星
- (5) Select the best Complex BEST from the STAR according to user-defined criteria;
- /*从星中选取一个最好的复合*/
- (6) Add BEST as an extra disjunct to COVER /*把最好的复合与COVER合取，形成新的COVER*/
- (7) END
- (8) RETURN COVER.

在算法AQR中调用了过程STAR，来排除所有的反例，产生覆盖种子的星。



算法 4-6 STAR

输入：种子SEED；反例NEG。

输出：星STAR。

- (1) 初始化STAR为空Complex
- (2) WHILE one or more Complexes in STAR covers some negative examples in NEG BEGIN /*如果STAR中的一个或多个Complex覆盖NEG中的负样例*/
 - (3) Select a negative example Eneg covered by a Complex in STAR; /*选取一个被STAR中的Complex覆盖的负样例*/
 - (4) Let EXTENSION be all Selectors that cover SEED but not ENEG; /*令EXTENSION为那些覆盖SEED但不覆盖ENEG的Selectors; */
 - (5) Let STAR be the set $\{x \wedge y | x \in \text{STAR}, y \in \text{EXTENSION}\}$;
/*令STAR = $\{x \wedge y | x \in \text{STAR}, y \in \text{EXTENSION}\}$; */
 - (6) Remove all Complexes in STAR subsumed by other Complexes in STAR;
/*从STAR中除去被其他Complexes所包含的Complexes; */
 - (7) Remove the worst Complexes from STAR UNTIL size of STAR is less than or equal to user-defined maximum (maxstar) /*删除STAR中最坏的Complex直到STAR的大小等于或小于用户定义的最大数目maxstar*/
- (8) END
- (9) RETURN STAR. /*返回一系列覆盖SEED但不覆盖NEG的规则*/



假设现有一个训练集，其包含两种属性：

size （属性值：micro, tiny, mid, big, huge, vast）

type （属性值：bicycle, motorcycle, car, prop, jet, glider）

现有正例、反例样本分别如表4-6，表4-7所示：

正例样本

size	type	class
Huge	bicycle	giant 2-wheeler
Huge	motorcycle	giant 2-wheeler

反例样本

size	type	class
Tiny	motorcycle	conventional transportation
tiny	car	conventional transportation
mid	car	conventional transportation
micro	jetfast	plane
Tiny	jetfast	plane
Mid	jetfast	plane

下面给出用AQR算法对 *giant 2-wheeler* 类的规则进行获取过程，具体步骤如下：

- (1) COVER={ }。
- (2) 空cover不覆盖任何样本，进入循环。
- (3) 一开始COVER并没有覆盖任何正例，假定从正例中选取的SEED 为 { size=huge, type =bicycle }。
- (4) 调用STAR (SEED, NEG) 去产生一个覆盖SEED但不包含NEG的STAR集合；
 - 初始化STAR为空，即STAR={ }。
 - 空的complex覆盖所有样例，STAR覆盖多个负样例，进入循环。
 - a) 选取一个被STAR中的复合覆盖的负样例ENEG，假定被选取的是 $Eneg = \{ size = tiny, type = motorcycle \}$ 。
 - b) 使EXTENSION为所有覆盖SEED但不覆盖ENEG的选择，则EXTENSION包括size= huge和type =bicycle，则又根据 $STAR = \{ x \wedge y | x \in STAR, y \in EXTENSION \}$ ，因此， $STAR = \{ size = huge \wedge type = bicycle \}$ 。
 - c) 在这里定义maxstar为2，可不对STAR进行精简。
 - d) 接着选取另一个被STAR中的复合覆盖的负样例ENEG，显然已经没有这样的负样例，因此， $STAR = \{ size = huge \wedge type = bicycle \}$ 。
 - 从STAR (SEED, NEG) 返回。

假设现有一个训练集，其包含两种属性：

size （属性值：micro, tiny, mid, big, huge, vast）

type （属性值：bicycle, motorcycle, car, prop, jet, glider）

现有正例、反例样本分别如表4-6，表4-7所示：

正例样本

size	type	class
Huge	bicycle	giant 2-wheeler
Huge	motorcycle	giant 2-wheeler

反例样本

size	type	class
Tiny	motorcycle	conventional transportation
tiny	car	conventional transportation
mid	car	conventional transportation
micro	jetfast	plane
Tiny	jetfast	plane
Mid	jetfast	plane

(5) $BEST = \{ size = huge \wedge type = bicycle \}$, $COVER = \{ size = huge \wedge type = bicycle \}$ 。

(6) 显然COVER不能覆盖所有的正例，从正例中选取另一个SEED= $\{ size = huge, type = motorcycle \}$ 。

(7) 调用STAR (SEED, NEG) 去产生一个覆盖SEED但不包含NEG的STAR集合。

- 初始化STAR为空，即STAR= $\{ \}$ ；
- 空的complex覆盖所有样例，所以STAR覆盖负样例，进入循环；
 - a) 假定被选取的是 $Eneg = \{ size = tiny, type = motorcycle \}$ ；
 - b) 使EXTENSION为所有覆盖SEED但不覆盖Eneg的选择，则EXTENSION包括 $size = huge$ ，则又根据 $STAR = \{ x \wedge y | x \in STAR, y \in EXTENSION \}$ ，因此， $STAR = \{ size = huge \}$ ；
 - c) 接着选取另一个被STAR中的复合覆盖的负样例Eneg，显然已经没有这样的负样例，因此， $STAR = \{ size = huge \}$ ；
 - d) 接着选取另一个被STAR中的复合覆盖的负样例ENEG，显然已经没有这样的负样例，因此， $STAR = \{ size = huge \wedge type = bicycle \}$ 。
- 从STAR (SEED, NEG) 返回。

(8) $BEST = \{ size = huge \}$ ，将BEST添加到COVER中， $COVER = \{ size = huge \wedge type = bicycle \vee size = huge \} = \{ size = huge \}$ 。

(9) 这时，COVER已经覆盖到全部的正例，则算法结束。输出规则为 $gaint\ 2-wheeler \leftarrow size = huge$ 。



- CN2使用一种基于噪音估计的启发式方法，使用这种方法可以不用对所有的训练样本进行正确的区分，但是规约出的规则在对新数据的处理上有很好的表现。

算法 4-7 CN2

输入：E /*E为训练样本*/

输出：RULE_LIST /*返回一个覆盖若干样例的规则*/

- (1) Let RULE_LIST be the empty list; /* 初始化RULES_LIST为空; */
- (2) REPEAT
- (3) Let BEST_CPX be Find_Best_Complex (E) ;
/*寻找最佳的规则Find_Best_Complex (E) 并将其结果放入BEST_CPX中; */
- (4) IF BEST_CPX is not nil THEN BEGIN
- (5) Let E' be the examples covered by BEST_CPX; /*令E'为BEST_CPX覆盖的所有样例*/
- (6) Remove from E the examples E' covered by BEST_CPX;
/*从训练样本E中除去E'，即E=E-E'; */
- (7) Let C be the most common class of examples in E';
/*令C为样本子集E'中最频繁的分类标号; */
- (8) Add the rule 'if BEST_CPX then class=C' to the end of RULE_LIST;
/*将规则 'if BEST_CPX then class=C'添加到RULES_LIST中*/
- (9) END
- (10) UNTIL BEST_CPX is nil or E is empty. /*直到BEST_CPX为空或者训练样本E为空*/
- (11) RETURN RULE_LIST

算法CN2需要通过调用函数Find_Best_Complex，它的描述写成下面算法4-8。



算法 4-8 Find_Best_Complex

输入: E /*E为训练样本*/

输出: BEST_CPX /*返回最佳的规则BEST_CPX */

- (1) Let the set STAR contain only the empty Complex; /*初始化集合STAR为空Complex; */
- (2) Let BEST_CPX be nil; /*初始化BEST_CPX为空*/
- (3) Let SELECTORS be the set of all possible Selectors;
/*集合SELECTOR为所有可能的选择*/
- (4) WHILE STAR is not empty DO BEGIN
- (5) Let NEWSTAR be the set $\{x \wedge y | x \in \text{STAR}, y \in \text{EXTENSION}\}$;
/*令NEWSTAR = $\{x \wedge y | x \in \text{STAR}, y \in \text{EXTENSION}\}$; */
- (6) Remove all Complexes in NEWSTAR that are either in STAR or are null;
/*从NEWSTAR中除去包括在STAR中的Complex或者为空的Complex*/
- (7) FOR every complex C_i in NEWSTAR
- (8) IF C_i is statistically significant when tested on E and better than BEST_CPX
according to user-defined criteria when tested on
E /*如果 C_i 在统计上有意义,
并且对训练集E测试后符合用户定义的条件且优于BEST_CPX*/
- (9) THEN replace the current value of BEST_CPX by C_i ; /*将BEST_CPX替换为 C_i */
- (10) REPEAT remove worst Complexes from NEWSTAR
- (11) UNTIL size of NEWSTAR is \leq user-defined maximum maxstar;
/*逐步移去在NEWSTAR中最坏的complex直到NEWSTAR的大小等于或小于用户
定义的最大数目maxstar*/
- (12) Let STAR be NEWSTAR; /*令STAR=NEWSTAR*/
- (13) END
- (14) RETURN BEST_CPX. /*返回BEST_CPX*/



- FOIL学习系统已经被广泛地应用在逻辑规约领域。FOIL是用来对无约束的一阶Horn子句进行学习。一个概念的定义是由一系列的子句组成，而其中每一个子句描述了一种证明一个样本是这个概念的实例的唯一方法。每个子句由一些文字的析取组成。
- FOIL由一系列的外部定义的断言开始，其中之一被确定为当前学习的概念，而其他作为背景文字。FOIL从这些外部定义的断言中获取一系列包括文字的子句。
- FOIL算法由一个空子句开始查找，其不断的向当前的子句中追加文字直到没有负样例被子句所覆盖。之后，FOIL重新开始一个子句的查找，直到所有的正样例均被已经生成的子句所覆盖。FOIL计算每一个外部定义断言的信息熵（Information Gain）和合法的变量（Legal Variabilization）用来决定哪一个文字添加到子句中。



一阶Horn子句的主要术语

■ 一阶Horn子句所涉及的主要术语有：

- 所有表达式由**常量**（如*Mary*、*23*或*Joe*）、变量（如*x*）、谓词（如在*Female*（*Mary*）中的*Female*和函数（如在*age*（*Mary*）中的*age*）组成；
- **项**（Term）为任意常量、任意变量或任意应用到项集合上的函数。例如，*Mary*，*x*，*age*（*Mary*），*age*（*x*）；
- **文字**（Literal）是应用到项集合上的任意谓词或其否定。例如，*Female*（*Mary*），*Greater_than*（*age*（*Mary*），*20*）；
- **基本文字**（Ground Literal）是不包括任何变量的文字；
- **负文字**（Negative Literal）是包括否定谓词的文字；
- **正文字**（Positive Literal）是不包括否定谓词的文字；
- **子句**（Clause）是多个文字的析取式， $M_1 \vee \dots \vee M_n$ ，其中所有变量是全程量化的；



一阶Horn子句的表达

- Horn子句是一个如下形式的表达式：

$$H \leftarrow (L_1 \wedge \dots \wedge L_n)。$$

其中， H ， L_1 ， L_2 ， \dots ， L_n 为正文字。 H 被称为Horn子句的头（Head）或推论（Consequent）。文字合取式 $L_1 \wedge L_2 \wedge \dots \wedge L_n$ 被称为Horn子句的体（Body）或者先行词（Antecedents）。

- 置换（Substitution）是一个将某些变量替换为某些项的函数。例如，置换 $\{x/3, y/z\}$ 把变量 x 替换为项3并且把变量 y 替换为项 z 。给定一个置换 θ 和一个文字 L ，我们使用 L_θ 代表应用置换 θ 到 L 得到的结果。



FOIL算法描述

算法 4-9 FOIL (Target_predicate, Predicates, Examples)

输入: Examples; /*样本数据*/; Predicates /*断言集合*/; Target_predicate /*目标断言*/

输出: 规则

- (1) Pos ← Examples中Target_predicate为True的成员;
- (2) Neg ← Examples中Target_predicate为False的成员;
- (3) Learn_rules ← {};
- (4) WHILE Pos不空 DO BEGIN /*学习NewRule*/
- (5) NewRules ← 没有前件的谓词Target_predicate规则;
- (6) NewRuleNeg ← Neg;
- (7) WHILE NewRuleNeg不空 BEGIN /*增加新文字以特化NewRule*/
- (8) Candidate_literals ← 基于Predicates对NewRule生成新文字
- (9) Best_literal ← argmax Foil_Gain (L, NewRule); /*获取最佳文字*/
- (10) 把Best_literal加入到NewRule的前件;
- (11) NewRuleNeg ← NewRuleNeg中满足NewRule前件的子集
- (12) END;
- (13) Learned_rules ← Learned_rules + NewRule;
- (14) Pos ← Pos - {被NewRule覆盖的Pos成员}
- (15) END;
- (16) 返回Learned_rules



- FOIL中的候选特征化式的生成：为生成当前规则的候选特征式，FOIL生成多个不同的新文字，每个可被单独地加到规则前件中。更精确地讲，假定当前规则为：

$$P(x_1, x_2, \dots, x_k) \leftarrow L_1 \dots L_n$$

其中， L_1, \dots, L_n 为当前规则前件中的文字，而 $P(x_1, x_2, \dots, x_k)$ 为规则头（或后件）。FOIL生成该规则的候选特征化式的方法是考虑符合下列形式的新文字 L_{n+1} ：

- $Q(v_1, \dots, v_r)$ ，其中 Q 为在Predicates中出现的任意谓词名，并且 v_i 既可为新变量，也可可为规则中已有的变量。 v_i 中至少一个变量必须是当前规则中已有的。
- $\text{Equal}(x_j, x_k)$ ，其中 x_j 和 x_k 为规则中已有的变量。
- 上述两种文字的否定。



■ Foil_Gain函数

- FOIL使用评估函数以估计增加新文字的效用，它基于加入新文字前后的正例和反例的约束数目。更精确地讲，考虑某规则R和一个可能被加到R的规则体的后选文字L。令R'为加入文字L到规则R后生成的规则。Foil_Gain(L, R)的值定义为：

$$Foil_Gain(L, R) = t(\log_2 \frac{p_1}{p_1 + n_1} - \log_2 \frac{p_0}{p_0 + n_0})$$

其中， p_0 为规则R的正例约束数目， n_0 为R的反例约束数目， p_1 是规则R'的正例约束数， n_1 为规则R'的反例约束数目。最后， t 是在加入文字L到R后仍旧能覆盖的规则R的正例约束数目。当加入L引入了一个新变量到R中时，只要在R'的约束中的某些约束扩展了原始的约束，它们仍然能被覆盖。



FOIL算法举例

假设学习目标文字 fathe(A, B) 的规则集例子。训练数据包括下列简单的断言集合：

Predicates: //断言集合

male(christopher), male(arthur)
female(victoria), female(penelope)
parent(christopher, arthur),
parent(christopher, victoria)
parent(penelope, arthur),
parent(penelope, victoria)

Examples: /*样本数据*/

positive:

father(christopher, arthur)
father(christopher, victoria)

negative:

father(penelope, arthur)
father(christopher, penelope)

则根据FOIL算法：

- $\text{Pos} = \{\text{father(christopher, arthur)}, \text{father(christopher, victoria)}\};$
- $\text{Neg} = \{\text{father(penelope, arthur)}, \text{father(christopher, penelope)}\};$
- $\text{Learned_rules} = \{\};$
- 当Pos不为空，则学习NewRule
 - a) $\text{NewRule} = \{\text{father(A, B)} \leftarrow \};$
 - b) $\text{NewRuleNeg} = \{\text{father(penelope, arthur)}, \text{father(christopher, penelope)}\};$
 - c) 当NewRuleNeg不为空，则增加特征化文字：
 - 由FOIL中的候选特征化式的规则，根据 $\text{father(A, B)} \leftarrow$ 可生成的候选文字为：male(A), not(male(A)), male(B), not(male(B)), female(A), not(female(A)), female(B), not(female(B)), parent(A, A), not(parent(A, A)), parent(B, B), not(parent(B, B)), parent(A, B), not(parent(A, B)), parent(B, A), not(parent(B, A)), parent(A, C), not(parent(A, C)), parent(C, A), not(parent(C, A)), parent(B, C), not(parent(B, C)), parent(C, B), not(parent(C, B));
 - 因此， $\text{Candidate_literals} = \{\text{male(A)}, \text{male(B)}, \text{female(A)}, \text{female(B)}, \dots\};$
 - 之后计算最佳文字Best_literal，具体计算过程如下表所示($p_0=2, n_0=2$)。



FOIL算法举例(续)

文字的获益计算结果

Test	p_1	n_1	t	Gain
male(A)	2	1	2	0.83
not(male(A))	0	1	0	0.00
male(B)	1	1	1	0.00
not(male(B))	1	1	1	0.00
female(A)	0	1	0	0.00
not(female(A))	2	1	2	0.83
female(B)	1	1	1	0.00
not(female(B))	1	1	1	0.00
parent(A, A)	0	0	0	0.00
not(parent(A, A))	2	2	2	0.00
parent(A, B)	2	1	2	0.83
not(parent(A, B))	0	1	0	0.00
parent(B, B)	0	0	0	0.00
not(parent(B, B))	2	2	2	0.00
parent(B, A)	0	0	0	0.00
not(parent(B, A))	2	2	2	0.00
parent(A, C)	4	4	2	0.00
not(parent(A, C))	0	0	0	0.00
parent(C, B)	0	0	0	0.00
not(parent(C, B))	2	1	2	0.83



FOIL算法举例(续)

- 选择文字male(A) 添加到Best_literal,
- NewRule={ father(A, B)← male(A)}, 其覆盖两个正例和一个反例。
- NewRuleNeg改写为NewRuleNeg={father(penelope, arthur)},
- d) 当NewRuleNeg不为空, 则增加特征化文字:
 - 则下一个文字应添加parent(A, B)。
 - 再将Best_literal加为NewRule的前件, 则NewRule={ father (A, B)← male(A)∧parent(A, B)};
 - 这时NewRuleNeg中的所有成员满足NewRule前件的子集, 跳出内层循环。
- e) Learn_rules=Learn_rules+{ father (A, B)← male(A)∧parent(A, B)} ;
- f) 再从Pos中减去被NewRules覆盖的成员;
- 这时Pos为空, 算法结束。

第三章 分类方法

内容提要

- 分类的基本概念与步骤
- 基于距离的分类算法
- 决策树分类方法
- 贝叶斯分类
- 规则归纳
- 与分类有关的问题





处理连续值的属性

- 对于连续属性值，其处理过程如下：
 - 根据属性的值，对数据集排序；
 - 用不同的阈值将数据集动态的进行划分；
 - 当输出改变时确定一个阈值；
 - 取两个实际值中的中点作为一个阈值；
 - 取两个划分，所有样本都在这两个划分中；
 - 得到所有可能的阈值、增益及增益比；
 - 在每一个属性会变为取两个取值，即小于阈值或大于等于阈值。



其他处理

- 简单地说，针对属性有连续数值的情况，则在训练集中可以按升序方式排列。如果属性A共有 n 种取值，则对每个取值 v_j ($j=1, 2, \dots, n$)，将所有的记录进行划分：一部分小于 v_j ；另一部分则大于或等于 v_j 。针对每个 v_j 计算划分对应的增益比率，选择增益最大的划分来对属性A进行离散化。
- 样本中可以含有未知属性值，其处理方法是用最常用的值替代或者是将最常用的值分在同一类中。
 - 具体采用概率的方法，依据属性已知的值，对属性和每一个值赋予一个概率，取得这些概率，取得这些概率依赖于该属性已知的值。

决策树研究问题

关于过渡拟合

分类模型的误差

一般可以将分类模型的误差分为：

- 1、训练误差 (Training Error) ;
- 2、泛化误差 (Generalization Error)



- 分类的效果一般和数据的特点有关，有的数据噪声大，有的有空缺值，有的分布稀疏，有的字段或属性间相关性强，有的属性是离散的而有的是连续值或混合式的。目前普遍认为不存在某种方法能适合于各种特点的数据。因此，在分类以前需要做一些数据的预处理。

- **数据清理**

- 主要是消除或减少数据噪声和处理空缺值。

- **特征选择**

- 从已知一组特征集中按照某一准则选择出有很好的区分特性的特征子集
 - 或按照某一准则对特征的分类性能进行排序，用于分类器的优化设计。

- **数据变换**

- 就是通过平滑、聚集、数据概化、规范化、特征构造等手段将数据转化为适合于挖掘的形式。



分类器性能表示

- 分类器性能表示方法类似信息检索系统的评价方法，可以采用OC曲线和ROC曲线、混淆矩阵等。
- 定义4-3 给定一个类 C_j 和一个数据库元组 t_i ， t_i 可能被分类器判定为属于 C_j 或不属于 C_j ，其实 t_i 本身可能属于 C_j 或不属于 C_j ，这样就会产生如下一些情况：
 - 真正：判定 t_i 在 C_j 中，实际上的确在其中。
 - 假正：判定 t_i 在 C_j 中，实际上不在其中。
 - 真负：判定 t_i 不在 C_j 中，实际上不在其中。
 - 假负：判定 t_i 不在 C_j 中，实际上的确在其中。
- 在上述定义的基础上，人们经常使用OC曲线和ROC曲线表示“假正”和“真正”的关系。OC曲线通常用于通信领域来测试误报率。OC曲线的水平轴一般表示“假正”的百分比，另外一个轴表示“真正”的百分比。



分类器性能表示 (续)

- 混淆矩阵是另外一种表示分类准确率的方法。假定有 m 个类，混淆矩阵是一个 $m*m$ 的矩阵， $C_{i,j}$ 表明了 D 中被分到类 C_j 但实际类别是 C_i 的元组的数量。显然地，最好解决方案是对角线以外的值为全零。

混淆矩阵示例

	实际分类		
	矮	中等	高
矮	0	4	0
中等	0	5	3
高	0	1	2



- 保持法和交叉验证是两种基于给定数据随机选样划分的、常用的评估分类方法准确率的技术。

■ (1) 保持法

把给定的数据随机地划分成两个独立的集合：训练集和测试集。通常，三分之一的数据分配到训练集，其余三分之二分配到测试集。使用训练集得到分类器，其准确率用测试集评估。

■ (2) 交叉验证

先把数据随机分成不相交的 n 份，每份大小基本相等，训练和测试都进行 n 次。比如，如果把数据分成10份，先把第一份拿出来放在一边用作模型测试，把其他9份合在一起建立模型，然后把这个用90%的数据建立起来的模型用上面放在一边的第一份数据做测试。这个过程对每一份数据都重复进行一次，得到10个不同的错误率。最后把所有数据放在一起建立一个模型，模型的错误率为上面10个错误率的平均。



- 从使用的主要技术上看，可以把分类方法归结为四种类型：
 - 基于距离的分类方法
 - 决策树分类方法
 - 贝叶斯分类方法
 - 规则归纳方法。

- 本章将择选一些有代表性的方法和算法来介绍这四类分类方法。

第三章 分类方法

内容提要

- 分类的基本概念与步骤
- 基于距离的分类算法
- 决策树分类方法
- 贝叶斯分类
- 规则归纳
- 与分类有关的问题

