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：朴素贝叶斯、EM算法
- 规则归纳
- 与分类有关的问题

朴素贝叶斯分类

那么既然是朴素贝叶斯**分类算法**，它的核心算法又是什么呢？

是下面这个贝叶斯公式：

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

换个表达形式就会明朗很多，如下：

$$\mathrm{p}(\text{类别}|\text{特征}) = \frac{p(\text{特征}|\text{类别})\mathrm{p}(\text{类别})}{\mathrm{p}(\text{特征})}$$

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

编号	色泽	根蒂	敲声	纹理	脐部	触感	密度	含糖率	好瓜
测1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	0.697	0.460	?

朴素贝叶斯模型

- 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。

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

- 假设现在有两枚硬币1和2，，随机抛掷后正面朝上概率分别为P1，P2。为了估计这两个概率，做实验，每次取一枚硬币，连掷5下，记录下结果

硬币	结果	统计
1	正正反正反	3正-2反
2	反反正正反	2正-3反
1	正反反反反	1正-4反
2	正反反正正	3正-2反
1	反正正反反	2正-3反

- 可以很容易地估计出P1和P2，如下：
- P1 = （3+1+2）/ 15 = 6/15=0.4
  P2= （2+3）/10 = 5/10=0.5

硬币	结果	统计
Unknown	正正反正反	3正-2反
Unknown	反反正正反	2正-3反
Unknown	正反反反反	1正-4反
Unknown	正反反正正	3正-2反
Unknown	反正正反反	2正-3反

- 还是估计P1和P2，要怎么做呢？

- 瞎猜：硬币1正面朝上的概率P1 = 0.2
  硬币2正面朝上的概率P2 = 0.7

Unknown	正正反正反	3正-2反
Unknown	反反正正反	2正-3反
Unknown	正反反反反	1正-4反
Unknown	正反反正正	3正-2反
Unknown	反正正反反	2正-3反

- 瞎猜：硬币1正面朝上的概率P1 = 0.2
  硬币2正面朝上的概率P2 = 0.7
- 如果是硬币1，得出3正2反的概率为
  0.2*0.2*0.2*0.8*0.8 = 0.00512
  如果是硬币2，得出3正2反的概率为
  0.7*0.7*0.7*0.3*0.3=0.03087

Unknown	正正反正反	3正-2反
Unknown	反反正正反	2正-3反
Unknown	正反反反反	1正-4反
Unknown	正反反正正	3正-2反
Unknown	反正正反反	2正-3反

- 如果是硬币1，得出3正2反的概率为
  0.2*0.2*0.2*0.8*0.8 = 0.00512
  如果是硬币2，得出3正2反的概率为
  0.7*0.7*0.7*0.3*0.3=0.03087
- 按照最大似然法则：0.03087 > 0.00512
  所以，第1轮中最有可能的是硬币2

Unknown	正正反正反	3正-2反
Unknown	反反正正反	2正-3反
Unknown	正反反反反	1正-4反
Unknown	正反反正正	3正-2反
Unknown	反正正反反	2正-3反

- 瞎猜：硬币1正面朝上的概率P1 = 0.2
  硬币2正面朝上的概率P2 = 0.7
- 如果是硬币1，得出3正2反的概率为
  0.2*0.2*0.8*0.8*0.8 = 0.02048
  如果是硬币2，得出3正2反的概率为
  0.7*0.7*0.3*0.3*0.3=0.01323

EM算法

- 瞎猜：硬币1正面朝上的概率P1 = 0.2
  硬币2正面朝上的概率P2 = 0.7

硬币1	硬币2
0.00512 即 0.2×0.2×0.2×0.8×0.8	0.03087 即 0.7×0.7×0.7×0.3×0.3
0.02048 即 0.2×0.2×0.8×0.8×0.8	0.01323 即 0.7×0.7×0.3×0.3×0.3
0.08192 即 0.2×0.8×0.8×0.8×0.8	0.00567 即 0.7×0.3×0.3×0.3×0.3
0.00512 即 0.2×0.2×0.2×0.8×0.8	0.03087 即 0.7×0.7×0.7×0.3×0.3
0.02048 即 0.2×0.2×0.8×0.8×0.8	0.01323 即 0.7×0.7×0.3×0.3×0.3

估计	实验结果	统计
硬币2	正 正 反 正 反	3正2反
硬币1	反 反 正 正 反	2正3反
硬币1	正 反 反 反 反	1正4反
硬币2	正 反 反 正 正	3正2反
硬币1	反 正 正 反 反	2正3反

- 重新估计P1和P2，如下：

  P1 = （2+1+2）/ 15 = 5/15=0.433

  P2= （3+3）/10 = 6/10=0.6

硬币	结果	统计
Unknown	正正反正反	3正-2反
Unknown	反反正正反	2正-3反
Unknown	正反反反反	1正-4反
Unknown	正反反正正	3正-2反
Unknown	反正正反反	2正-3反

- 目标：估计P1和P2，要怎么做呢？
- 瞎猜估计值：硬币1正面朝上的概率P1 = 0.2
  硬币2正面朝上的概率P2 = 0.7
- 更新估计值：硬币1正面朝上的概率P1 = 0.43
  硬币2正面朝上的概率P2 = 0.6

硬币	结果	统计
Unknown	正正反正反	3正-2反
Unknown	反反正正反	2正-3反
Unknown	正反反反反	1正-4反
Unknown	正反反正正	3正-2反
Unknown	反正正反反	2正-3反

- 目标：估计P1和P2，要怎么做呢？
- 更新估计值：硬币1正面朝上的概率P1 = 0.43
  硬币2正面朝上的概率P2 = 0.6

Unknown	正正反正反	3正-2反
Unknown	反反正正反	2正-3反
Unknown	正反反反反	1正-4反
Unknown	正反反正正	3正-2反
Unknown	反正正反反	2正-3反

- 如果是硬币1，得出3正2反的概率为

  0.43*0.43*0.43*0.57*0.57 = 0.02583

  如果是硬币2，得出3正2反的概率为

  0.6*0.6*0.6*0.4*0.4=0.03456

- 按照最大似然法则：0.03456 > 0.02583

  所以，第1轮中最有可能的是硬币2

EM算法

- 估计：硬币1正面朝上的概率P1 = 0.43
  硬币2正面朝上的概率P2 = 0.6

硬币1	硬币2
0.02583 即 0.43×0.43×0.43×0.57×0.57	0.03456 即 0.6×0.6×0.6×0.4×0.4
0.02048 即 0.43×0.43×0.57×0.57×0.57	0.02304 即 0.6×0.6×0.4×0.4×0.4
0.04539 即 0.43×0.57×0.57×0.57×0.57	0.01536 即 0.6×0.4×0.4×0.4×0.4
0.02583 即 0.43×0.43×0.43×0.57×0.57	0.03456 即 0.6×0.6×0.6×0.4×0.4
0.03424 即 0.43×0.43×0.57×0.57×0.57	0.02304 即 0.6×0.6×0.4×0.4×0.4

估计	实验结果	统计
硬币2	正 正 反 正 反	3正2反
硬币2	反 反 正 正 反	2正3反
硬币1	正 反 反 反 反	1正4反
硬币2	正 反 反 正 正	3正2反
硬币1	反 正 正 反 反	2正3反

- 重新估计P1和P2，如下：

  P1 = （1+2）/ 10 = 3/10=0.3

  P2= （3+2+3）/15 = 8/15=0.533

- 假设现在有两枚硬币1和2，，随机抛掷后**正面朝上概率**分别为P1，P2。为了估计这两个概率，做实验，每次取一枚硬币，连掷5下，记录下结果

硬币	结果	统计
1	正正反正反	3正-2反
2	反反正正反	2正-3反
1	正反反反反	1正-4反
2	正反反正正	3正-2反
1	反正正反反	2正-3反

- 可以很容易地估计出P1和P2，如下：
- P1 = （3+1+2）/ 15 = 6/15=0.4
  P2= （2+3）/10 = 5/10=0.5

硬币	结果	统计
Unknown	正正反正反	3正-2反
Unknown	反反正正反	2正-3反
Unknown	正反反反反	1正-4反
Unknown	正反反正正	3正-2反
Unknown	反正正反反	2正-3反

	瞎猜	更新	再更新		真实值
P1	0.2	0.43	0.3	……	0.4
P2	0.7	0.6	0.53	……	0.5

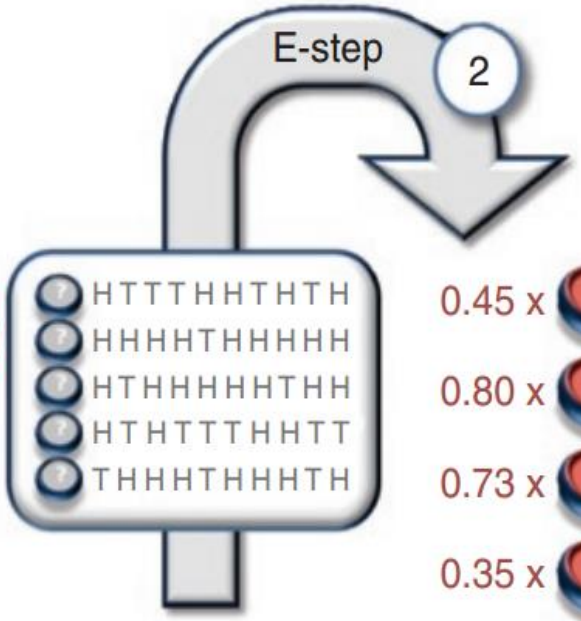
EM算法

- 最大期望算法（Expectation-maximization algorithm）是一种从不完全数据或有数据丢失的数据集（存在隐含变量）中求解概率模型参数的最大似然估计方法。

- 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预计值，计算其最大似然预计值；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步上找到的参数预计值被用于下一个E步计算中，这个过程不断交替进行。

- 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。

b Expectation maximization



	Coin A	Coin B
0.45 x A, 0.55 x B	≈ 2.2 H, 2.2 T	≈ 2.8 H, 2.8 T
0.80 x A, 0.20 x B	≈ 7.2 H, 0.8 T	≈ 1.8 H, 0.2 T
0.73 x A, 0.27 x B	≈ 5.9 H, 1.5 T	≈ 2.1 H, 0.5 T
0.35 x A, 0.65 x B	≈ 1.4 H, 2.1 T	≈ 2.6 H, 3.9 T
0.65 x A, 0.35 x B	≈ 4.5 H, 1.9 T	≈ 2.5 H, 1.1 T
	≈ 21.3 H, 8.6 T	≈ 11.7 H, 8.4 T

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

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

1

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

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

3 M-step

4

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

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

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]

规则归纳

- **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(续)

- 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算法原理

- 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

- 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。

AQR算法描述

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 disjuct 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。

AQR算法描述(续)

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∧y|x∈STAR，y∈EXTENSION}；/*令STAR= {x∧y|x∈STAR，y∈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∧y|x∈STAR，y∈EXTENSION}，因此，STAR={ size=huge∧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∧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∧type =bicycle* }，COVER ={ *size=huge∧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∧y|x∈STAR，y∈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∧type =bicycle }。
- 从STAR（SEED，NEG）返回。

（8）BEST={size=huge}，将BEST添加到COVER中，COVER ={ size=huge∧type =bicycle∨ 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← size=huge。

CN2算法描述

- 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。

CN2算法描述(续)

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∧y|x∈STAR，y∈EXTENSION}；
/*令NEWSTAR= {x∧y|x∈STAR，y∈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 Ci in NEWSTAR
（8） IF Ci is statistically significant when tested on E and better than BEST_CPX according to user-defined criteria when tested on E /*如果Ci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 Ci；/*将BEST_CPX替换为Ci*/
（10） REPEAT remove worst Complexes from NEWSTAR
（11） UNTIL size of NEWSTAR is <= 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算法

- 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∨……∨*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：

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

  其中，H，L_1，L_2，…，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$L_1 \wedge L_2 \wedge \ldots \wedge L_n$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Ture的成员；
（2）　Neg←Examples中Target_predicate为False的成员；
（3）　Learen_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算法介绍

- 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：

$$P(x_1, x_2, \ldots, x_k) \leftarrow L_1 \ldots L$$

其中，$L_1, \ldots, L_n$为当前规则前件中的文字，而$P(x_1, x_2, \ldots, x_k)$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(v1，…，vr)，其中Q为在Predicates中出现的任意谓词名，并且vi既可为新变量，也可为规则中已有的变量。vi中至少一个变量必须是当前规则中已有的。
  - 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算法介绍(续)

- **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：

- Pos={father(christopher，arthur)，father(christopher，victoria)}；
- Neg={father(penelope，arthur)， father(christopher，penelope)}；
- Learned_rules={}；
- 当Pos不为空，则学习NewRule
  a) NewRule={father(A，B)←}；
  b) NewRuleNeg={ father(penelope，arthur)， father(christopher，penelope)}；
  c) 当NewRuleNeg不为空，则增加特征化文字：
  - 由FOIL中的侯选特征化式的规则，根据father(A，B)←可生成的侯选文字为：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，Candidate_literals={ male(A)，male(B)，female(A)，female(B).....}，
  - 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，---，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）

分类数据预处理

- 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。

分类器性能的表示

- 分类器性能的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。

分类器性能的表示（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

分类器性能的评估

- 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份合在一起来建立模型，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。

分类方法的类型

- 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