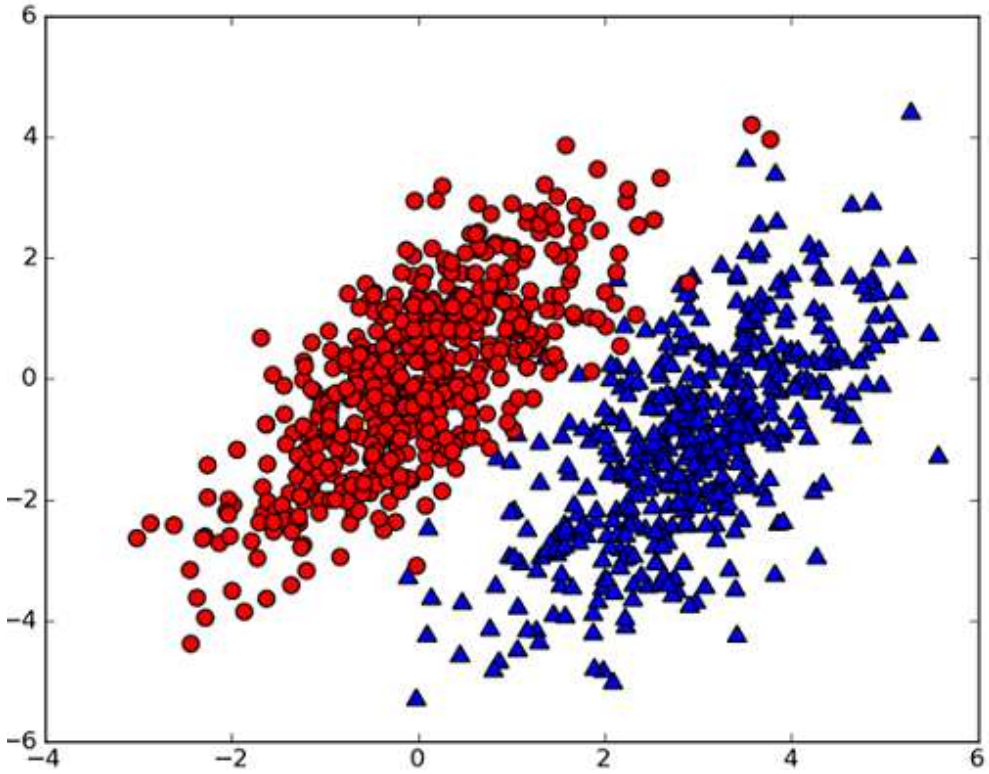
清華大學
Tsinghua University

第四章
贝叶斯分类器

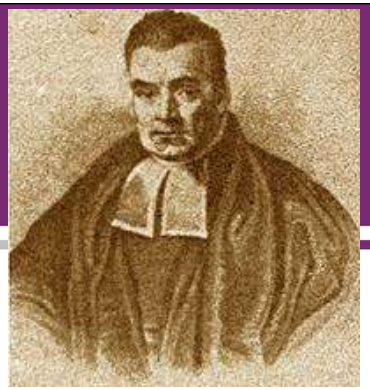
问题的提出



KNN？

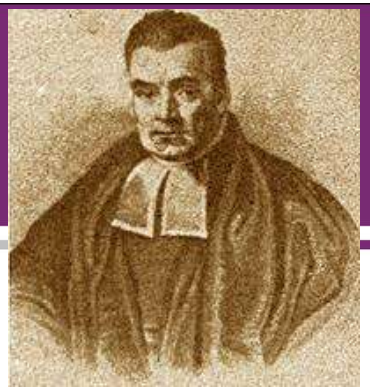
决策树?

概率方法?



贝叶斯

- 贝叶斯(约1701-1761) Thomas Bayes，英国数学家。约1701年出生于伦敦，做过神甫。1742年成为英国皇家学会会员。1761年4月7日逝世。贝叶斯在数学方面主要研究概率论。他首先将归纳推理法用于概率论基础理论，并创立了贝叶斯统计理论，对于统计决策函数、统计推断、统计的估算等做出了贡献。他死后，理查德·普莱斯(Richard Price)于1763年将他的著作《机会问题的解法》(An essay towards solving a problem in the doctrine of chances)寄给了英国皇家学会，对于现代概率论和数理统计产生了重要的影响



贝叶斯

- 贝叶斯决策就是在不完全情报下，对部分未知的状态用主观概率估计，然后用贝叶斯公式对发生概率进行修正，最后再利用期望值和修正概率做出最优决策。
- 贝叶斯决策理论方法是统计模型决策中的一个基本方法，其基本思想是：
- 1、已知类条件概率密度参数表达式和先验概率。
- 2、利用贝叶斯公式转换成后验概率。
- 3、根据后验概率大小进行决策分类。

贝叶斯网络的应用

- 最早的PathFinder系统，该系统是淋巴疾病诊断的医学系统，它可以诊断60多种疾病，涉及100多种症状;后来发展起来的Internist－I系统，也是一种医学诊断系统，但它可以诊断多达600多种常见的疾病。

- 1995年，微软推出了第一个基于贝叶斯网的专家系统，一个用于幼儿保健的网站OnParent (www.onparenting.msn.com)，使父母们可以自行诊断。

贝叶斯网络的应用

- (1)故障诊断(diagnose)
- (2)专家系统(expert system)
- (3)规划(planning)
- (4)学习(learning)
- (5)分类(classifying)

概率(回顾)

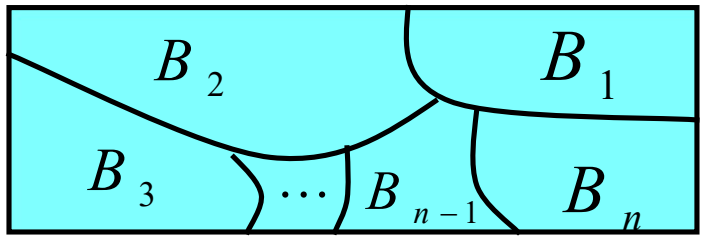
样本空间的划分

定义　设 Ω 为试验E的样本空间, $B_1, B_2, \cdots, B_n$ 为 E 的一组事件,若

1^0　$B_i B_j = \varnothing, i, j = 1, 2, \cdots, n;$

2^0　$B_1 \cup B_2 \cup \cdots \cup B_n = \Omega,$

则称 $B_1, B_2, \cdots, B_n$ 为样本空间 Ω 的一个划分.



概率(回顾)

全概率公式

定义 设Ω为试验E的样本空间, A为E的事件,

$B_1, B_2, \cdots, B_n$为Ω的一个划分,且$P(B_i) > 0$

$(i = 1, 2, \cdots, n)$,则

$$P(A) = P(A \mid B_1)P(B_1) + P(A \mid B_2)P(B_2) + \cdots + P(A \mid B_n)P(B_n)$$

$$= \sum_{i=1}^{n} P(B)P(A \mid B_i)$$

基本方法

- 训练数据集: $T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$

- 由X和Y的联合概率分布P(X,Y)独立同分布产生

- 朴素贝叶斯通过训练数据集学习联合概率分布P(X,Y)，
  - 即先验概率分布: $P(Y=c_k),\quad k=1,2,\cdots,K$
  - 及条件概率分布:
    $$P(X=x\,|\,Y=c_k)=P(X^{(1)}=x^{(1)},\cdots,X^{(n)}=x^{(n)}\,|\,Y=c_k),\quad k=1,2,\cdots,K$$
  - 注意: 条件概率为指数级别的参数: $K\prod_{j=1}^{n}S_j$

基本方法

- 条件独立性假设：
$$P(X=x|Y=c_k)=P(X^{(1)}=x^{(1)},\cdots,X^{(n)}=x^{(n)}|Y=c_k)$$
$$=\prod_{j=1}^{n}P(X^{(j)}=x^{(j)}|Y=c_k)$$

- “朴素”贝叶斯名字由来，牺牲分类准确性。

- 贝叶斯定理：
$$P(Y=c_k|X=x)=\frac{P(X=x|Y=c_k)P(Y=c_k)}{\sum_k P(X=x|Y=c_k)P(Y=c_k)}$$

- 代入上式：
$$P(Y=c_k|X=x)=\frac{P(Y=c_k)\prod_j P(X^{(j)}=x^{(j)}|Y=c_k)}{\sum_k P(Y=c_k)\prod_j P(X^{(j)}=x^{(j)}|Y=c_k)}$$

基本方法

- 贝叶斯分类器:

$$y = f(x) = \arg\max_{c_k} \frac{P(Y = c_k)\prod_j P(X^{(j)} = x^{(j)} \mid Y = c_k)}{\sum_k P(Y = c_k)\prod_j P(X^{(j)} = x^{(j)} \mid Y = c_k)}$$

- 分母对所有c_k都相同:

$$y = \arg\max_{c_k} P(Y = c_k)\prod_j P(X^{(j)} = x^{(j)} \mid Y = c_k)$$

后验概率最大化的含义：

- 朴素贝叶斯法将实例分到后验概率最大的类中，等价于期望风险最小化，
- 假设选择0-1损失函数：f(X)为决策函数

$$L(Y, f(X)) = \begin{cases} 1, & Y \neq f(X) \\ 0, & Y = f(X) \end{cases}$$

- 期望风险函数：$R_{\exp}(f) = E[L(Y, f(X))]$

- 取条件期望：$R_{\exp}(f) = E_X \sum_{k=1}^{K} [L(c_k, f(X))] P(c_k \mid X)$

后验概率最大化的含义：

- 只需对X=x逐个极小化，得：

$$f(x) = \arg\min_{y \in \mathcal{Y}} \sum_{k=1}^{K} L(c_k, y) P(c_k \mid X = x)$$

$$= \arg\min_{y \in \mathcal{Y}} \sum_{k=1}^{K} P(y \neq c_k \mid X = x)$$

$$= \arg\min_{y \in \mathcal{Y}} (1 - P(y = c_k \mid X = x))$$

$$= \arg\max_{y \in \mathcal{Y}} P(y = c_k \mid X = x)$$

- 推导出后验概率最大化准则：　$f(x) = \arg\max_{c_k} P(c_k \mid X = x)$

朴素贝叶斯法的参数估

- 应用极大似然估计法估计相应的概率：
- 先验概率$P(Y=c_k)$的极大似然估计是：

$$P(Y=c_k)=\frac{\sum_{i=1}^{N} I(y_i=c_k)}{N},\quad k=1,2,\cdots,K$$

- 设第j个特征$x^{(j)}$可能取值的集合为：$\{a_{j1},a_{j2},\cdots,a_{jS_j}\}$
- 条件概率的极大似然估计：

$$P(X^{(j)}=a_{jl}\mid Y=c_k)=\frac{\sum_{i=1}^{N} I(x_i^{(j)}=a_{jl},y_i=c_k)}{\sum_{i=1}^{N} I(y_i=c_k)}$$

$$j=1,2,\cdots,n;\quad l=1,2,\cdots,S_j;\quad k=1,2,\cdots,K$$

朴素贝叶斯法的参数估

- 学习与分类算法Naïve Bayes Algorithm：
- 输入：
  - 训练数据集 $T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$

$x_i^{(j)}$ • 第i个样本的第j个特征 $x_i=(x_i^{(1)},x_i^{(2)},\cdots,x_i^{(n)})^T$

a_{jl} • 第j个特征可能取的第*l*个值 $x_i^{(j)}\in\{a_{j1},a_{j2},\cdots,a_{jS_j}\}$

- 输出： $y_i\in\{c_1,c_2,\cdots,c_K\}$
  - x的分类

朴素贝叶斯法的参数估

- 步骤
  - 1、计算先验概率和条件概率

$$P(Y=c_k)=\frac{\sum_{i=1}^{N} I(y_i=c_k)}{N}, \quad k=1,2,\cdots,K$$

$$P(X^{(j)}=a_{jl}\mid Y=c_k)=\frac{\sum_{i=1}^{N} I(x_i^{(j)}=a_{jl}, y_i=c_k)}{\sum_{i=1}^{N} I(y_i=c_k)}$$

$$j=1,2,\cdots,n; \quad l=1,2,\cdots,S_j; \quad k=1,2,\cdots,K$$

朴素贝叶斯法的参数估

- 步骤
  - 2、对于给定的实例 $x=(x^{(1)},x^{(2)},\cdots,x^{(n)})^{\mathrm{T}}$
  - 计算

$$P(Y=c_k)\prod_{j=1}^{n}P(X^{(j)}=x^{(j)}\mid Y=c_k),\quad k=1,2,\cdots,K$$

  - 3、确定x的类别

$$y=\arg\max_{c_k}P(Y=c_k)\prod_{j=1}^{n}P(X^{(j)}=x^{(j)}\mid Y=c_k)$$

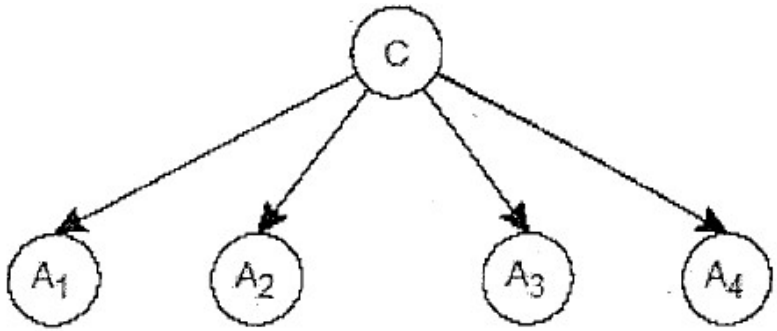
例子

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

例子

- 测试

<Outlook=sunny, Temperature=cool, Humidity=high, Wind=strong>



$$c(x) = \arg\max_{c \in \{yes,no\}} P(c)P(sunny \mid c)P(cool \mid c)P(high \mid c)P(strong \mid c)$$

例子

$P(yes)=(9+1)/(14+2)=10/16$

$P(sunny \mid yes)=(2+1)/(9+3)=3/12$

$P(cool \mid yes)=(3+1)/(9+3)=4/12$

$P(high \mid yes)=(3+1)/(9+2)=4/11$

$P(strong \mid yes)= (3+1)/(9+2)=4/11$

$P(no)=(5+1)/(14+2)=6/16$

$P(sunny \mid no)=(3+1)/(5+3)=4/8$

$P(cool \mid no)=(1+1)/(5+3)=2/8$

$P(high \mid no)=(4+1)/(5+2)=5/7$

$P(strong \mid no)=(3+1)/(5+2)=4/7$

$P(yes)P(sunny|yes)P(cool|yes)P(high|yes)P(strong|yes)=0.0069$

$P(no)P(sunny|no)P(cool|no)P(high|no)P(strong|no)=0.0191$

贝叶斯估计

- 考虑：用极大似然估计可能会出现所要估计的概率值为0的情况，这时会影响到后 验概率的计算结果，使分类产生偏差.解决这一问题的方法是采用贝叶斯估计。
- 条件概率的贝叶斯估计：

$$P_\lambda(X^{(j)}=a_{jl}\mid Y=c_k)=\frac{\sum_{i=1}^{N}I(x_i^{(j)}=a_{jl},y_i=c_k)+\lambda}{\sum_{i=1}^{N}I(y_i=c_k)+S_j\lambda}$$

- 先验概率的贝叶斯估计：

$$P_\lambda(Y=c_k)=\frac{\sum_{i=1}^{N}I(y_i=c_k)+\lambda}{N+K\lambda}$$

朴素贝叶斯网络的缺陷

- 考虑几个问题:
- 1、如果属性之间不相互独立?
- 2、如果属性A和属性B都很重要，但是相关?
- 3、如果属性A，属性B之间独立，但是在属性C下有关?
- 4、属性之间的条件概率究竟有多少个?
- 5、条件概率谬论?

贝叶斯分类器

- 条件概率的谬论

  假设 P(A|B) 大致等于 P(B|A)

例子:

P(disease) = 1% = 0.01

P(well) = 99% = 0.99

P(negative | disease) = 1%

P(positive | disease) = 99%

P(positive | well) = 1%

P(negative | well) = 99%

? P(disease|positive) = ?

$$p(d|p) = \frac{P(p|d)P(d)}{P(p|w)P(w) + P(p|d)P(d)}$$

$$= \frac{0.99 * 0.01}{0.01 * 0.99 + 0.99 * 0.01}$$

$$= 50\%$$

Problem: if P(d)=0.1% , P(d|P)=?

信息论相关概念

阻塞:一条路径被结点集 F 阻塞，是指在路径上存在一个结点 Z 满足下面三种情形之一：

(1) Z∈ F，并且路径中有一条有向弧指向 Z，另一条有向弧源自 Z；

(2)Z∈ F，并且路径中有两条有向弧源自 Z；

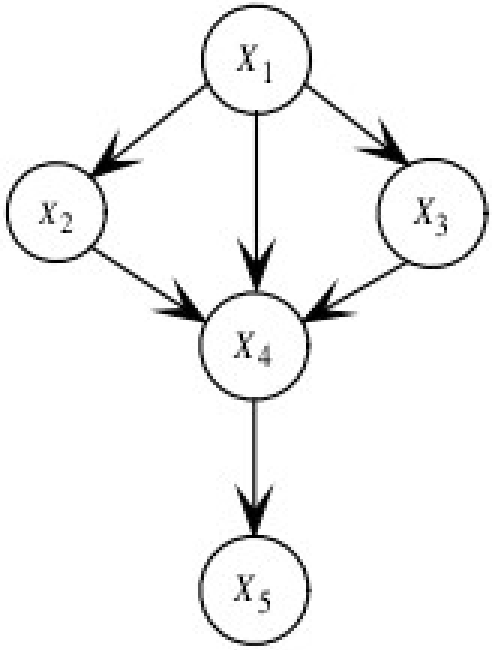
(3) Z 及 Z 的所有后继结点都不在 F 中，并且路径中有两条有向弧指 Z。

信息论相关概念

阻塞:

X = {X2}和Y = {X3}被Z = {X1}所分割。路径X2←X1→ X3被X1∈ Z阻塞; 同时路径X2→ X4← X3也被阻塞,因为X4及它的所有子孙都不在Z中。因此成立d(X2,X1,X3)。

然而，X和Y没有被Z= {X1, X5}所d分割，因为路径X2→ X4← X3由于X5的作用而成为活跃，X5在Z 中，是X4的子孙

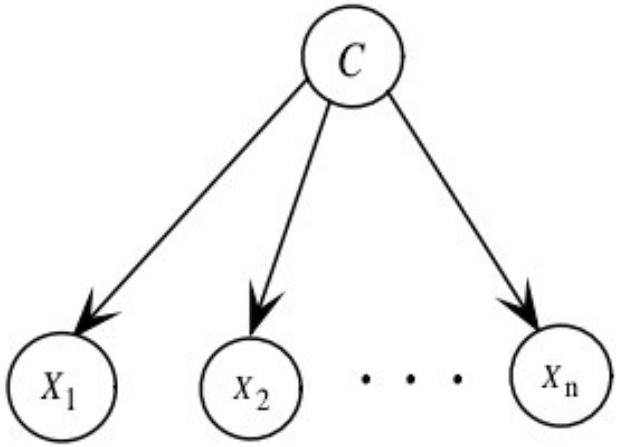


信息论相关概念

d-separation：令 X，Y 和 Z 是一个有向无环图 G 中三个不相交节点的子集，如果在集合 X 和 Y 中所有节点间的所有路径都被集合 Z 所阻塞，则称集合 X 和 Y 被 Z 集合 d-separation，表示为<X,Y|Z>G，也称 Z为 A 和 B 的切割集。否则，称在给定集合 Z 下集合 X 和 Y 图形依赖。

I-map：　假设 G 是以随机变量 Y1,Y2,…,Yn 为节点的一个有向无环图，P 是随机变量 Y1,Y2,…,Yn的联合概率函数，如果从图 G 中得到的每一个独立性假设(Yi在给定其父母节点变量的情况下独立于它的非后代节点)在联合概率 P 的计算中都成立，则称 G 是该概率分布 P 的一个独立映射(Independence-map, I-map)。

朴素贝叶斯网络分类器



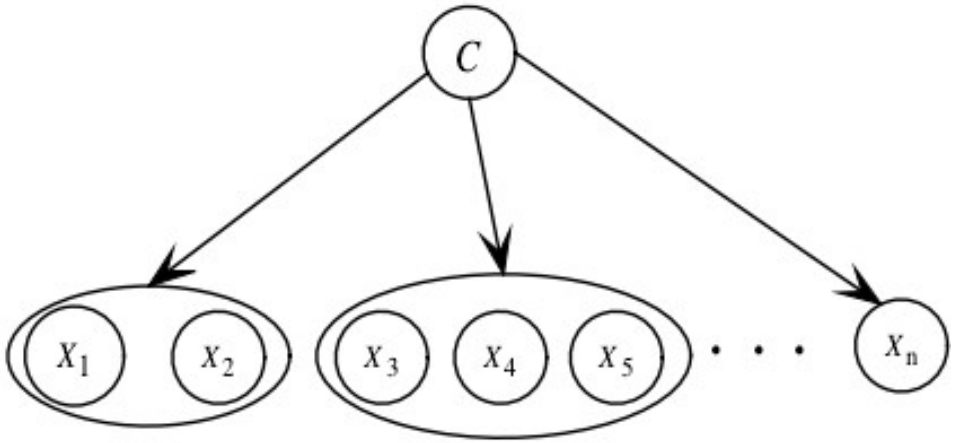
$$C^* = \arg\max P(c|x_1, x_2, \cdots, x_n) = \arg\max \prod_{i=1}^{n} P(x_i|c)P(c)$$

朴素贝叶斯网络分类器的改进算法

- 半朴素贝叶斯分类器 (SNBC：Semi-Naive Bayesian Classifier)
- 选择贝叶斯分类器(SBC: Selective Bayesian Classifier)
- 树增广朴素贝叶斯网络分类器(TAN: Tree Augmented Naive Bayes)
- 平均一依赖估测器(AODE：averaged one dependence estimators)学习算法
- 加权平均的一依赖估测器(WAODE：weightily averaged one dependence estimators)学习算法
- 无约束贝叶斯网络分类器(GBN：General Baynes Network)
- 隐藏扩展的朴素贝叶斯分类算法(HANB)

半朴素贝叶斯网络分类器

- SNBC： Semi-Naive Bayesian Classifier)，
  - SNBC在模型构建过程中，依照一定的标准将关联程度较大的特征属性合并在一起组合成新属性。
  - SNBC的各个组合属性之间相对于类别属性也是相互独立。
  - SNBC除了结构上的差别之外，计算推导过程与朴素贝叶斯相同。



选择贝叶斯分类器

- SBC: Selective Bayesian Classifier)
- SBC设计目标是为了提高朴素贝叶斯在属性冗余情况下的分类精度。
- SBC只使用属性集的子集作为决策过程中的属性结点，即选择贝叶斯分类器选择初始特征的子集作为属性结点。
- SBC通过搜索特征空间，去掉特征间具有较强依赖关系的属性。
- 前向（空集到全集），后向（全集到空集）
- 可能的搜索子集2^m

信息论相关概念

- 属性的相互独立性?
- 如何测量?

信息增益（回顾）

- 设有随机变量(X,Y),其联合概率分布为:

$$P(X=x_i,Y=y_j)=p_{ij}\,,\quad i=1,2,\cdots,n\,;\quad j=1,2,\cdots,m$$

- 条件熵H(Y|X)：表示在已知随机变量X的条件下随机变量Y的不确定性，定义为X给定条件下Y的条件概率分布的熵对X的数学期望:

$$H(Y\mid X)=\sum_{i=1}^{n}p_iH(Y\mid X=x_i)$$

信息增益（回顾）

- 定义 (信息增益):特征A对训练数据集D的信息增益，g(D,A), 定义为集合D的经验熵H(D)与特征A给定条件下D的经验条件熵H(D|A)之差，即

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

- (Information gain)表示得知特征X的信息而使得类Y的信息的不确定性减少的程度.
- 一般地，熵H(Y)与条件熵H(Y|X)之差称为互信息（mutual information)
- 决策树学习中的信息增益等价于训练数据集中类与特征的互信息.

信息论相关概念

- 设信源X为离散随机变量，则用来度量X的不确定性的信息熵H(X)为

$$H(X) = -\sum_{X} P(x) \log P(x)$$

- 设(X,Y)均为离散随机变量，用来度量二元随机变量的不确定性联合信息熵H(X,Y)为

$$H(X,Y) = -\sum_{X}\sum_{Y} P(x,y) \log P(x,y)$$

信息论相关概念

- 条件信息熵H(X|Y)用来度量在收到随机变量Y 提供的信息后，随机变量X仍然存在的不确定性

$$H(X|Y) = -\sum_{X}\sum_{Y} P(x,y)\log P(x|y)$$

- 互信息I(X;Y)用来描述随机变量Y 提供的关于X的信息量的大小。

$$I(X;Y) = H(X) - H(X|Y) = \sum_{X}\sum_{Y} P(x,y)\log\frac{P(x,y)}{P(x)P(y)}$$

信息论相关概念

- 在已知Y 的前提下，随机变量X和Z之间的条件互信息定义为

$$I(X;Z|Y) = \sum_X \sum_Y \sum_Z P(x,y,z) \log \frac{P(x,y,z)P(y)}{P(x,y)P(z,y)}$$

$$= \sum_X \sum_Y \sum_Z P(x,y,z) \log \frac{P(x,z|y)}{P(x|y)P(z|y)}$$

信息论相关概念

- 条件独立：对概率模式 M，A，B 和 C 是 U 的三个互不相交的变量子集，如果对 $\forall x \in A$，$\forall y \in B$和 $\forall z \in C$都有 $p(x|y,z)= p(x|z)$，其中 $p(y,z)>0$，称给定 C 时 A 和 B 条件独立，记为 $I(A,C,B)_M$。
- 或：$p(x,y|z)=p(x|z)p(y|z)$

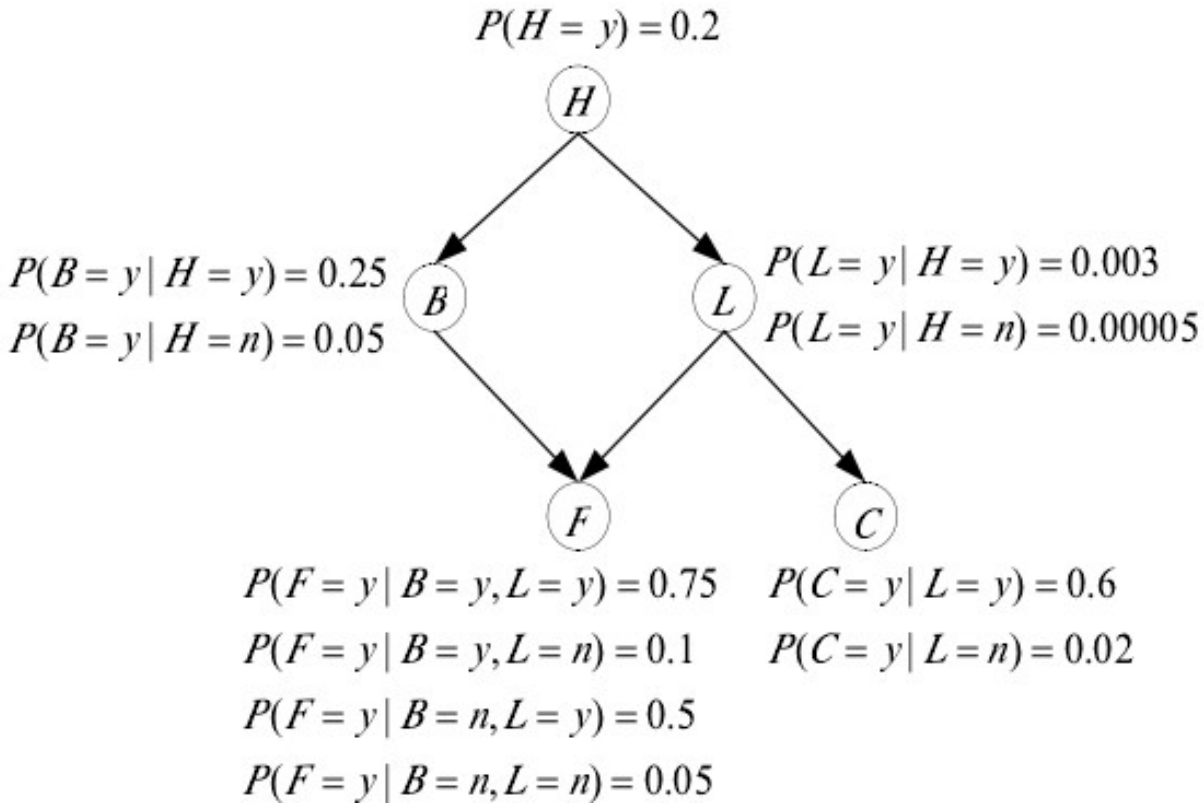
贝叶斯网络基本模型

- 贝叶斯网络是由图论和概率论结合而成的描述多元统计关系的模型，它为多个变量之间复杂依赖关系的表示提供了统一的框架，具有紧凑有效、简洁直观的特点。
- 由于贝叶斯网络对大规模复杂系统简约而紧凑的表示能力，使得其成为人工智能、专家系统、模式识别、数据挖掘和软件测试等领域的研究热点。

贝叶斯网络基本模型

- 父节点pa(Vi)
- 子节点ch(Vi)
- 邻居节点nb(Vi)
- 祖先an(vi) $V_i \in an(V_i)$
- 有向环
- 有向无环（directed acyclic Graph）
- BN二元组 $BN = (G, P)$， $G = (V, E)$

贝叶斯网络基本模型


P(H = y) = 0.2
H
P(B = y | H = y) = 0.25
P(B = y | H = n) = 0.05
B
L
P(L = y | H = y) = 0.003
P(L = y | H = n) = 0.00005
F
C
P(F = y | B = y, L = y) = 0.75
P(F = y | B = y, L = n) = 0.1
P(F = y | B = n, L = y) = 0.5
P(F = y | B = n, L = n) = 0.05
P(C = y | L = y) = 0.6
P(C = y | L = n) = 0.02

贝叶斯网络基本模型

- N个变量的BN, $X = \{x_1, x_2, \ldots, x_n\}$
- 变量xi有ri个取值
- 父节点pa(xi)的取值qi
- 网络参数: $\theta = \{\theta_{ijk} | i = 1, \ldots, n;\ j = 1, \ldots, q_i;\ k = 1, \ldots, r_i\}$

$$\theta_{ijk} = P(x_i = k \mid pa(x_i) = j)$$

$$l(\theta \mid D) = \log L(\theta \mid D) = \log \prod_{l=1}^{m} P(d_l \mid \theta) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} m_{ijk} \log \theta_{ijk}$$

m_{ijk} 是数据 D 中满足 $x_i = k$ 和 $pa(x_i) = j$ 的样本的数量

贝叶斯估计

- Θ表示所有参数组成的向量

$$\theta = \{\theta_{ijk} | i = 1, \ldots, n;\ j = 1, \ldots, q_i;\ k = 1, \ldots, r_i\}$$

- Θ_{ij*} 表示 $\theta_{ij1}, \theta_{ij2}, \ldots, \theta_{ijr_i}$ 子向量 $P(x_i \mid pa(x_i) = j)$

- Θ_{i**} 表示 $\theta_{i1*}, \theta_{i2*}, \ldots, \theta_{iq_i*}$ 子向量 $P(x_i \mid pa(x_i))$

贝叶斯估计

- P(Θ)的三个假设：
  - （全局独立）关于不同变量x_i的参数相互独立。

$$P(\theta) = \prod_{i=1}^{n} P(\theta_{i\cdot\cdot})$$

  - （局部独立）给定一个变量x_i对应于$pa(x_i)$ 的不同取值的参数相互独立。

$$P(\theta_{i\cdot\cdot}) = \prod_{j=1}^{q_i} P(\theta_{ij\cdot})$$

  - $P(\theta_{ij\cdot})$ 服从 Dirichlet 分布 $D[\alpha_{ij1}, \alpha_{ij2}, \ldots, \alpha_{ijr_i}]$
  - θ 的先验分布 $P(\theta)$ 等价于 $\prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \theta_{ijk}^{\alpha_{ijk}-1}$

贝叶斯估计

- 给定数据集D的条件下，

$$P(\theta \mid D) = \lambda \cdot P(\theta) \cdot P(D \mid \theta) = \lambda \cdot \prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \theta_{ijk}^{m_{ijk}+\alpha_{ijk}-1}$$

- 即：θ 的后验概率分布 P (θ|D)服从 Dirichlet 分布

$$D[m_{ij1} + \alpha_{ij1}, m_{ij2} + \alpha_{ij2}, \ldots, m_{ijr_i} + \alpha_{ijr_i}]$$

- 结合贝叶斯网络的可分解性得到参数估计值

$$\theta_{ijk}^{*} = \int P(x_i = k \mid pa(x_i) = j, \theta_{ijk}) P(\theta_{ijk} \mid D) d\theta_{ijk} = \int \theta_{ijk} P(\theta_{ijk} \mid D) d\theta_{ijk}$$

$$\theta_{ijk}^{*} = (m_{ijk} + \alpha_{ijk}) \Big/ \sum_{k=1}^{r_i} (m_{ijk} + \alpha_{ijk}).$$

贝叶斯估计

- Beta distribution
- 在α,β两个值的控制下x的概率分布，即控制Beta分布的性质

$$Beta(\alpha,\beta):\ prob(x|\alpha,\beta)=\frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}$$

$$B(\alpha,\beta)=\int_0^1 t^{\alpha-1}(1-t)^{\beta-1}dt \qquad B(\alpha,\beta)=\frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$$

- Gamma 函数 阶乘在实数域和复数域的扩展

$$\Gamma(z)=\int_0^\infty t^{z-1}e^{-t}\,\mathrm{d}t \qquad \Gamma(n)=(n-1)!$$

正整数的情况

贝叶斯网络结构学习

- BN 结构学习就是在给定一个数据样本集合 D 的前提下，寻找一个与训练样本集D匹配最好的网络结构，而搜索最优的网络结构是一个 NP 难问题，因此，从数据中学习贝叶斯网络结构一直是研究的热点。
- 基于评分搜索的方法
- Robinson 等证明了包含n个节点可能的 BN 结构数目

$$f(n)=\sum_{i=1}^{n}(-1)^{i+1}\cdot(n!/i!(n-i)!)\cdot 2^{i(n-i)}\cdot f(n-i)$$

  - K2 评分（又称 CH 评分），BD 评分，MDL（Minimum Description Length）评分，BIC（Bayesian Information Criterion）评分，AIC（Akaike Information Criterion）评分

基于信息理论的评分函数

- 基于信息理论的评分函数主要是利用编码理论和信息论中的最小描述长度（Minimum Description Length，MDL）原理来实现的．
- 其基本思想源自对数据的存储．假设D是一组给定的实例数据，如果要对其进行保存，为了节省存储空间，一般采用某种模型对其进行编码压缩，然后再保存压缩后的数据；
- 为了在需要的时候可以完全恢复这些实例数据，要求对所使用的模型进行保存；因此，需要保存的数据长度等于压缩后的数据长度加上模型的描述长度，该长度称为总的描述长度。

基于信息理论的评分函数

- MDL评分准则趋向于寻找一个结构较简单的网络，实现网络精度与复杂度之间的均衡。
- 利用参数个数作为网络结构复杂度的惩罚函数

$$C(G)=\frac{1}{2}\log m\sum_{i=1}^{n}(r_i-1)q_i.$$

- 采用海明码表示压缩后的数据长度，即就是关于数据D和模型的对数似然

$$LL_D(G)=\sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i}m_{ijk}\log(\frac{m_{ijk}}{m_{ij*}}).$$

- 得MDL得分

$$F_{MDL}(G\mid D)=\sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i}m_{ijk}\log(\frac{m_{ijk}}{m_{ij*}})-\frac{1}{2}\log m\sum_{i=1}^{n}(r_i-1)q_i.$$

基于信息理论的评分函数

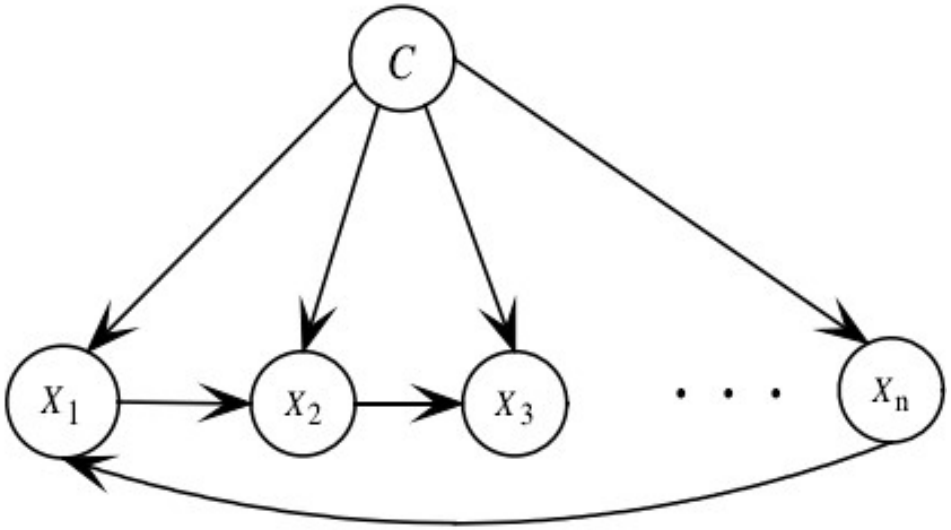
- 简化:

$$F_{AIC}(G \mid D) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} m_{ijk} \log(\frac{m_{ijk}}{m_{ij*}}) - \sum_{i=1}^{n} (r_i - 1) q_i$$

树增广朴素贝叶斯网络分类器

- 树增广朴素贝叶斯网络分类器(TAN: Tree Augmented Naive Bayes)
  - 在属性之间增添连接弧，以消除朴素贝叶斯分类器关于条件独立的假设，称为扩展弧。
  - 从结点Xi到Xj的扩展弧表示属性Xj对分类的影响也取决于Xi的取值。
  - 可能有这样的情况: 待分类样本的属性值Xi和Xj对分类的影响都不大，P(xi|c)和P(xj|c)的值比较低，但是P(xj|xi, c)的值却很高，这时朴素贝叶斯分类器会过度降低样本属于类c概率，而扩展的朴素贝叶斯网络分类器却可以避免这一点。

树增广朴素贝叶斯网络分类器



树增广朴素贝叶斯网络分类器

- TAN的构造过程：
- 步骤1. 通过训练集计算属性对之间的条件互信息$I(X_i,X_j|C)$ $(i \neq j)$。
- 步骤2. 建立一个以$X_1, X_2, \cdots, X_n$为结点的加权完全无向图，结点X_i, X_j之间的权重为$I(X_i,X_j|C)(i \neq j)$。
- 步骤3. 利用求最大权生成树算法，建立该无向图最大权重跨度树。首先把边按权重由大到小排序，之后遵照被选择的边不能构成回路的原则，按照边的权重由大到小的顺序选择边，这样由所选择的边构成的树便是最大权重跨度树。

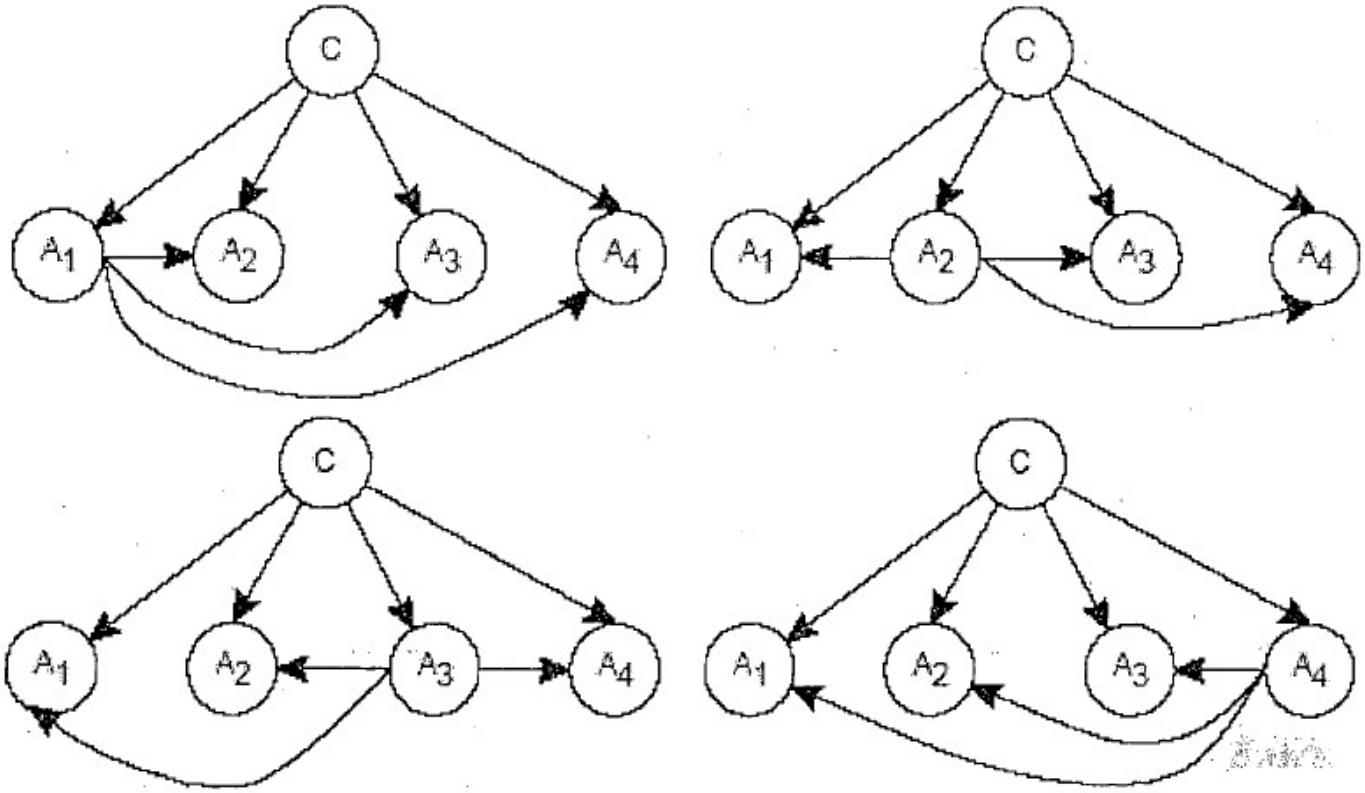
树增广朴素贝叶斯网络分类器

- 步骤4. 指定一个属性结点作为根结点，将所有边的方向设置成由根结点指向外，把无向图转换成有向图。
- 步骤5. 加入类结点C，并添加从C指向每个属性结点Xi的弧。

平均一依赖估测器

- 平均一依赖估测器(averaged one dependence estimators,简记为AODE)的学习算法。
- AODE首先针对每一个属性结点学习一个“特殊”的树扩展的朴素贝叶斯分类器,之所以特殊,是因为它直接把该属性结点当成是其他所有属性结点的父亲结点。然后,把这些树扩展的朴素贝叶斯分类器进行平均。

平均一依赖估测器



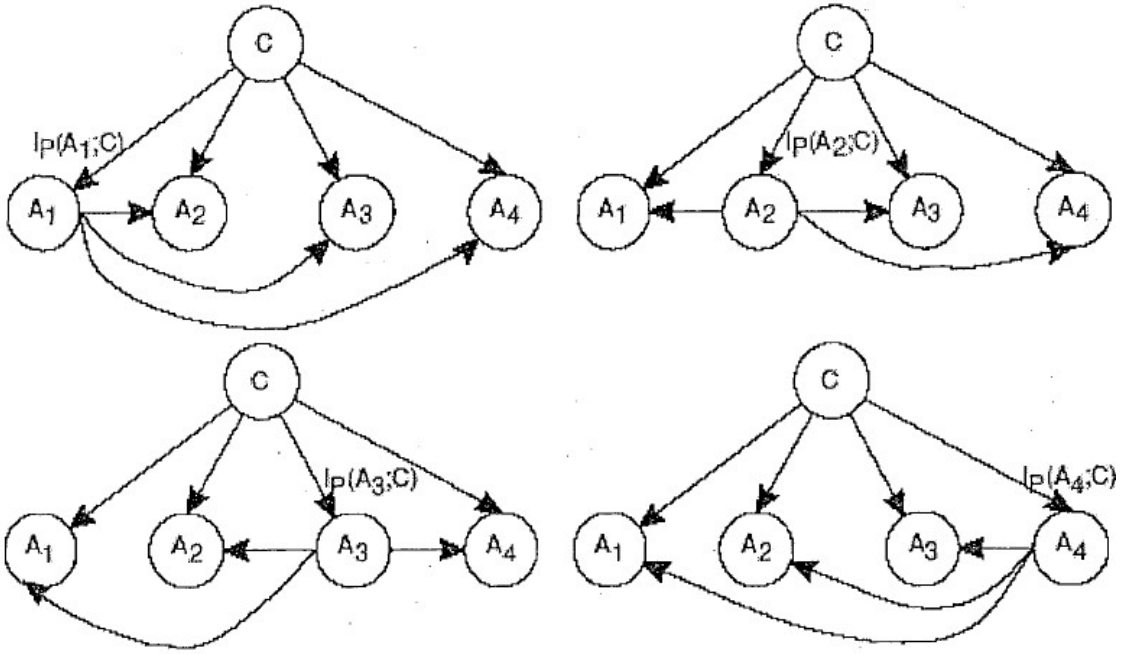
平均依赖估测器

$$c(x) = \arg\max_{c \in C} \frac{\sum_{i=1 \wedge F(a_i) \geq 30}^{m} P(a_i, c) \prod_{j=1 \wedge j \neq i}^{m} P(a_j \mid a_i, c)}{numParent}$$

- 在AODE模型中,所有的树扩展的朴素贝叶斯分类器对最终分类的贡献是一样的,这似乎不太合理

加权平均一依赖估测器

- 加权平均的一依赖估测器(weightily averaged one dependence estimators,简记WAODE）



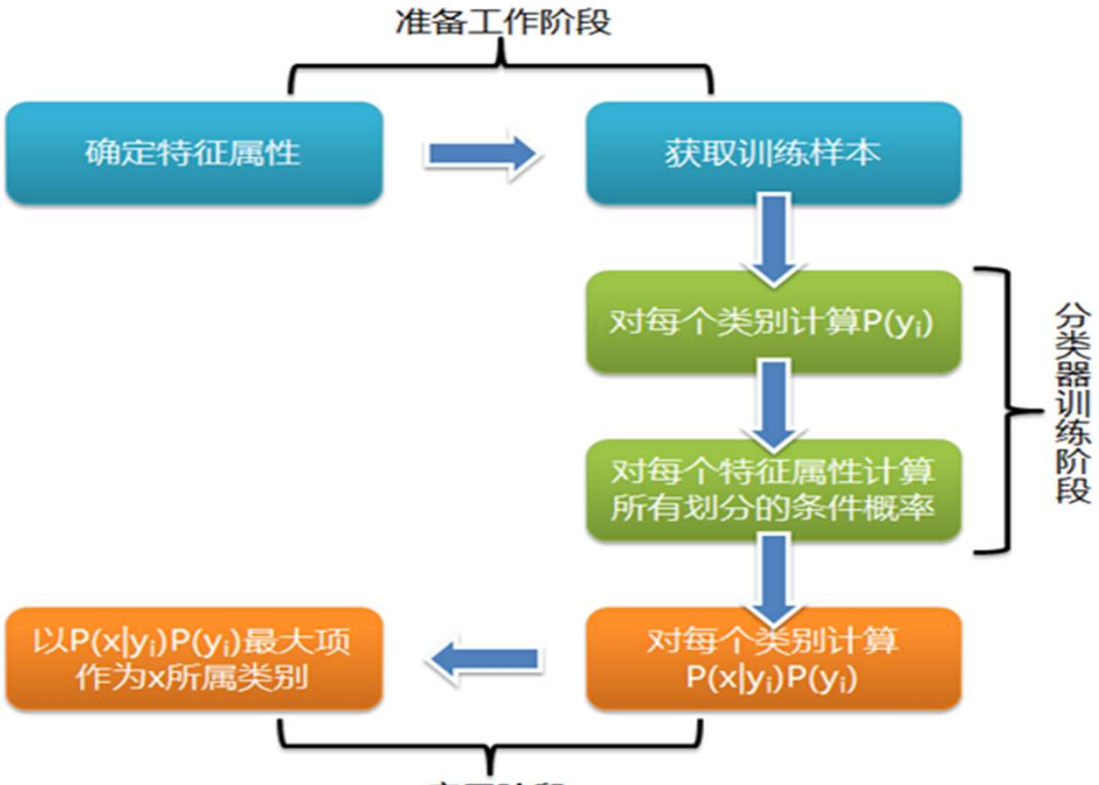
加权平均--依赖估测器

- 加权平均的一依赖估测器(weightily averaged one dependence estimators,简记WAODE)

$$c(x) = \arg\max_{c \in C} \frac{\sum_{i=1}^{m} W_i P(a_i, c) \prod_{j=1 \wedge j \neq i}^{m} P(a_j \mid a_i, c)}{\sum_{i=1}^{m} W_i}$$

$$W_i = I_P(A_i; C) = \sum_{a_i, c} P(a_i, c) \log \frac{P(a_i, c)}{P(a_i) P(c)}$$

贝叶斯算法处理流程：



使用Python 进行文本分类

```
def loadDataSet():
    postingList=[['my', 'dog', 'has', 'flea', 'problems', 'help', 'please'],
                 ['maybe', 'not', 'take', 'him', 'to', 'dog', 'park', 'stupid'],
                 ['my', 'dalmation', 'is', 'so', 'cute', 'I', 'love', 'him'],
                 ['stop', 'posting', 'stupid', 'worthless', 'garbage'],
                 ['mr', 'licks', 'ate', 'my', 'steak', 'how', 'to', 'stop', 'him'],
                 ['quit', 'buying', 'worthless', 'dog', 'food', 'stupid']]
    classVec = [0,1,0,1,0,1]    #1 is abusive, 0 not
    return postingList,classVec
```

使用Python 进行文本分类

```
def createVocabList(dataSet):
    vocabSet = set([])   #create empty set
    for document in dataSet:
        vocabSet = vocabSet | set(document)    #union of the two sets
    return list(vocabSet)
```

训练算法：从词向量计算概率

$$p(c_i|w) = \frac{p(w|c_i)p(c_i)}{p(w)}$$

$$p(w_0, w_1, w_2 .. w_N|c_i) \Rightarrow p(w_0|c_i)p(w_1|c_i)p(w_2|c_i)...p(w_N|c_i)$$

- def trainNB0(trainMatrix,trainCategory):
- numTrainDocs = len(trainMatrix)
- numWords = len(trainMatrix[0])
- pAbusive = sum(trainCategory)/float(numTrainDocs)
- p0Num = ones(numWords); p1Num = ones(numWords) #change to ones()
- p0Denom = 2.0; p1Denom = 2.0 #change to 2.0

训练算法：从词向量计算概率

```
for i in range(numTrainDocs):
    if trainCategory[i] == 1:
        p1Num += trainMatrix[i]
        p1Denom += sum(trainMatrix[i])
    else:
        p0Num += trainMatrix[i]
        p0Denom += sum(trainMatrix[i])
p1Vect = log(p1Num/p1Denom)          #change to log()
p0Vect = log(p0Num/p0Denom)          #change to log()
return p0Vect,p1Vect,pAbusive
```

测试和使用算法

```
def classifyNB(vec2Classify, p0Vec, p1Vec, pClass1):
    p1 = sum(vec2Classify * p1Vec) + log(pClass1)    #element-wise mult
    p0 = sum(vec2Classify * p0Vec) + log(1.0 - pClass1)
    if p1 > p0:
        return 1
    else:
        return 0
```

测试算法和修改分类器

$$p(w_0|c_i)p(w_1|c_i)p(w_2|c_i)...p(w_N|c_i)$$

- `p0Num = ones(numWords); p1Num = ones(numWords)`
  `p0Denom = 2.0; p1Denom = 2.0`
- `ln(a*b) = ln(a)+ln(b)`

- def classifyNB(vec2Classify, p0Vec, p1Vec, pClass1):
- p1 = sum(vec2Classify * p1Vec) + log(pClass1) #element-wise mult
- p0 = sum(vec2Classify * p0Vec) + log(1.0 - pClass1)
- if p1 > p0:
- return 1
- else:
- return 0

测试算法和修改分类器

```
def testingNB():
    listOPosts,listClasses = loadDataSet()
    myVocabList = createVocabList(listOPosts)
    trainMat=[]
    for postinDoc in listOPosts:
        trainMat.append(setOfWords2Vec(myVocabList, postinDoc))
    p0V,p1V,pAb = trainNB0(array(trainMat),array(listClasses))
    testEntry = ['love', 'my', 'dalmation']
    thisDoc = array(setOfWords2Vec(myVocabList, testEntry))
    print testEntry,'classified as: ',classifyNB(thisDoc,p0V,p1V,pAb)
    testEntry = ['stupid', 'garbage']
    thisDoc = array(setOfWords2Vec(myVocabList, testEntry))
    print testEntry,'classified as: ',classifyNB(thisDoc,p0V,p1V,pAb)
```

文档词袋模型

```
def bagOfWords2VecMN(vocabList, inputSet):
    returnVec = [0]*len(vocabList)
    for word in inputSet:
        if word in vocabList:
            returnVec[vocabList.index(word)] += 1
    return returnVec
```

切分文本

```
>>> mySent='This book is the best book on Python or M.L. I have ever laid
➥ eyes upon.'
>>> mySent.split()
['This', 'book', 'is', 'the', 'best', 'book', 'on', 'Python', 'or', 'M.L.',
 'I', 'have', 'ever', 'laid', 'eyes', 'upon.']


>>> import re
>>> regEx = re.compile('\\W*')
>>> listOfTokens = regEx.split(mySent)
>>> listOfTokens
['This', 'book', 'is', 'the', 'best', 'book', 'on', 'Python', 'or', 'M',
'L', '', 'I', 'have', 'ever', 'laid', 'eyes', 'upon', '']


>>> [tok.lower() for tok in listOfTokens if len(tok) > 0]
['this', 'book', 'is', 'the', 'best', 'book', 'on', 'python', 'or', 'm',
'l', 'i', 'have', 'ever', 'laid', 'eyes', 'upon']
```

使用朴素贝叶斯方法进行交叉验证

```
Def spamTest():
    docList=[]; classList = []; fullText =[]
    for i in range(1,26):
        wordList = textParse(open('email/spam/%d.txt' % i).read())
        docList.append(wordList)
        fullText.extend(wordList)
        classList.append(1)
        wordList = textParse(open('email/ham/%d.txt' % i).read())
        docList.append(wordList)
        fullText.extend(wordList)
        classList.append(0)
    vocabList = createVocabList(docList)#create vocabulary
    trainingSet = range(50); testSet=[]          #create test set
```

使用朴素贝叶斯方法进行交叉验证

```
for i in range(10):
    randIndex = int(random.uniform(0,len(trainingSet)))
    testSet.append(trainingSet[randIndex])
    del(trainingSet[randIndex])
trainMat=[]; trainClasses = []
for docIndex in trainingSet:
    trainMat.append(setOfWords2Vec(vocabList, docList[docIndex]))
    trainClasses.append(classList[docIndex])
p0V,p1V,pSpam = trainNB0(array(trainMat),array(trainClasses))
errorCount = 0

for docIndex in testSet:
    wordVector = setOfWords2Vec(vocabList, docList[docIndex])
    if classifyNB(array(wordVector),p0V,p1V,pSpam) !=
 classList[docIndex]:
        errorCount += 1
print 'the error rate is: ',float(errorCount)/len(testSet)
```

2

Cla

RSS的使用：从RSS源收集数据

- Feedparser的安装 http://code.google.com/p/feedparser/
- 下载
- 网络学堂
- 安装：python setup.py install

```
>>> import feedparser
>>>ny=feedparser.parse('http://newyork.craigslist.org/stp/index.rss')

>>> ny['entries']
>>> len(ny['entries'])
100
```

RSS的使用：从RSS源收集数据

- 高频词去除

```
def calcMostFreq(vocabList,fullText):
    import operator
    freqDict = {}
    for token in vocabList:
        freqDict[token]=fullText.count(token)
    sortedFreq = sorted(freqDict.iteritems(), key=operator.itemgetter(1),\
                        reverse=True)
    return sortedFreq[:30]
```

1 Calculates frequency of occurrence

RSS的使用：从RSS源收集数据

- 两个RSS源作为参数

```
def localWords(feed1,feed0):
    import feedparser
    docList=[]; classList = []; fullText =[]
    minLen = min(len(feed1['entries']),len(feed0['entries']))
    for i in range(minLen):
        wordList = textParse(feed1['entries'][i]['summary'])
        docList.append(wordList)
        fullText.extend(wordList)
        classList.append(1)
        wordList = textParse(feed0['entries'][i]['summary'])
        docList.append(wordList)
        fullText.extend(wordList)
        classList.append(0)
```

RSS的使用：从RSS源收集数据

```
vocabList = createVocabList(docList)
top30Words = calcMostFreq(vocabList,fullText)
for pairW in top30Words:
    if pairW[0] in vocabList: vocabList.remove(pairW[0])
trainingSet = range(2*minLen); testSet=[]
for i in range(20):
    randIndex = int(random.uniform(0,len(trainingSet)))
    testSet.append(trainingSet[randIndex])
    del(trainingSet[randIndex])
trainMat=[]; trainClasses = []
```

RSS的使用：从RSS源收集数据

```
for docIndex in trainingSet:
    trainMat.append(bagOfWords2VecMN(vocabList, docList[docIndex]))
    trainClasses.append(classList[docIndex])
p0V,p1V,pSpam = trainNB0(array(trainMat),array(trainClasses))
errorCount = 0
for docIndex in testSet:
    wordVector = bagOfWords2VecMN(vocabList, docList[docIndex])
    if classifyNB(array(wordVector),p0V,p1V,pSpam) != \
        classList[docIndex]:
        errorCount += 1
print 'the error rate is: ',float(errorCount)/len(testSet)
return vocabList,p0V,p1V
```

RSS的使用：从RSS源收集数据

- 问题：
- 高频次设定的阈值对正确率的影响
- 消除各种语言的高频词

http://www.ranks.nl/resources/stopwords.html

分析数据：显示地域相关的用词

```
def getTopWords(ny,sf):
    import operator
    vocabList,p0V,p1V=localWords(ny,sf)
    topNY=[]; topSF=[]

for i in range(len(p0V)):
    if p0V[i] > -6.0 : topSF.append((vocabList[i],p0V[i]))
    if p1V[i] > -6.0 : topNY.append((vocabList[i],p1V[i]))
sortedSF = sorted(topSF, key=lambda pair: pair[1], reverse=True)
print "SF**SF**SF**SF**SF**SF**SF**SF**SF**SF**SF**SF**SF**SF**
for item in sortedSF:
    print item[0]
sortedNY = sorted(topNY, key=lambda pair: pair[1], reverse=True)
print "NY**NY**NY**NY**NY**NY**NY**NY**NY**NY**NY**NY**NY**NY **"
for item in sortedNY:
    print item[0]
```

实验作业三

- 见参考文献。

- Q&A?