

聚类与分类

IRLAB

聚类

大纲

- 聚类分析简介
- 层次聚类
 - 单连接和全连接聚类
 - 组平均聚类
 - 应用：改进语言模型
 - 自顶向下聚类
- 非层次聚类
 - K-均值
 - EM算法

什么是聚类分析?

- 聚类: 数据对象的集合
 - 在同一个类中, 数据对象是相似的
 - 不同类之间的对象是不相似的
- 聚类分析
 - 一个数据集合分组成几个聚类
- 聚类是一种无监督分类: 没有预定义的类
- 典型应用
 - 作为一个独立的工具 透视数据分布
 - 可以作为其他算法的预处理步骤

聚类在自然语言中的应用

- 探测数据分析（exploratory data analysis）
 - 例如词性标注，将相似的词作为同一种词性，对前置词比较有效
 - 对this和the 这种语法语义特征不一致的词，不总分在一组的词不适合
- 概化（generalization）
 - 等价类，可以使用相同的上下文环境，解决数据稀疏问题
 - 同时聚类是学习的一种方法（推理 Friday 的前置词）

聚类算法类型

- 层次聚类与非层次聚类
 - 层次聚类的每一个节点是其父节点的一个子类，叶节点对应的是类别中每一个单独的对象，常用算法自底向上与自上向下（凝聚与分裂）
 - 非层次聚类只是简单的包括了每类的数量，体现不了他们之间的层次关系，常用算法K-均值
- 软聚类与硬聚类
 - 硬聚类将每一个对象分到一个且只能是一个的类别中，例如K-均值
 - 软聚类刻画的是将对象归属不同类的程度，模糊聚类（EM算法）

层次聚类和非层次聚类的比较

- 层次聚类

- 适合于数据的详细描述
- 提供更多的信息
- 没有单一的最好的算法
- 效率没有非层次的好

- 非层次聚类

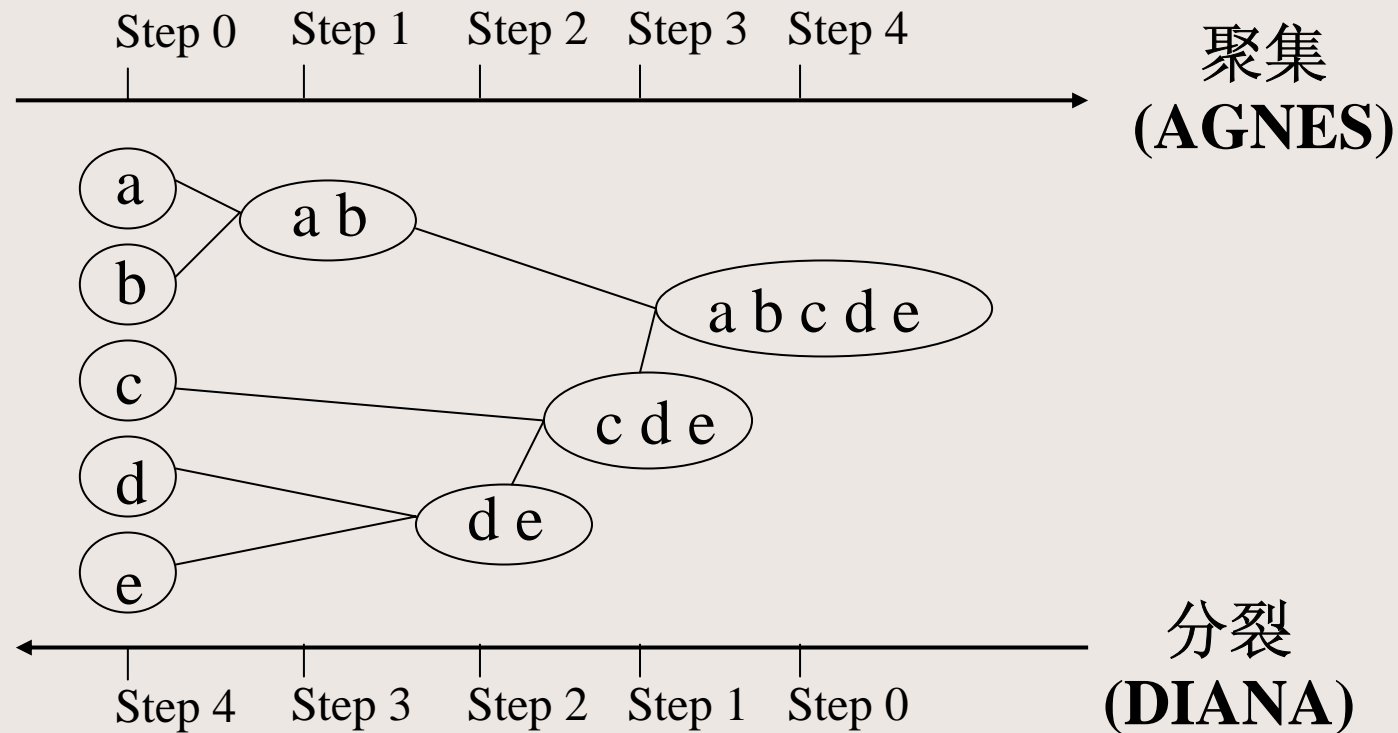
- 适合于大数据集合要求考虑效率较高的情况
- K-均值是一种最简单的方法，并且有效的
- K-均值采用欧氏距，不能表达更广泛的数据
- EM算法提供了类的定义以及基于复杂概率模型的数据的分配

层次聚类

- 自底向下的聚类(凝聚)
 - 每一项自成一类
 - 不断地将最近的两类合为一类
- 自顶向下的聚类（分裂）
 - 将所有项看作一类
 - 找出最不相似的项分裂出去成为两类

层次聚类

- 这种方法不需要输入参数K，但需要一个终止条件。例如：相似度阈值



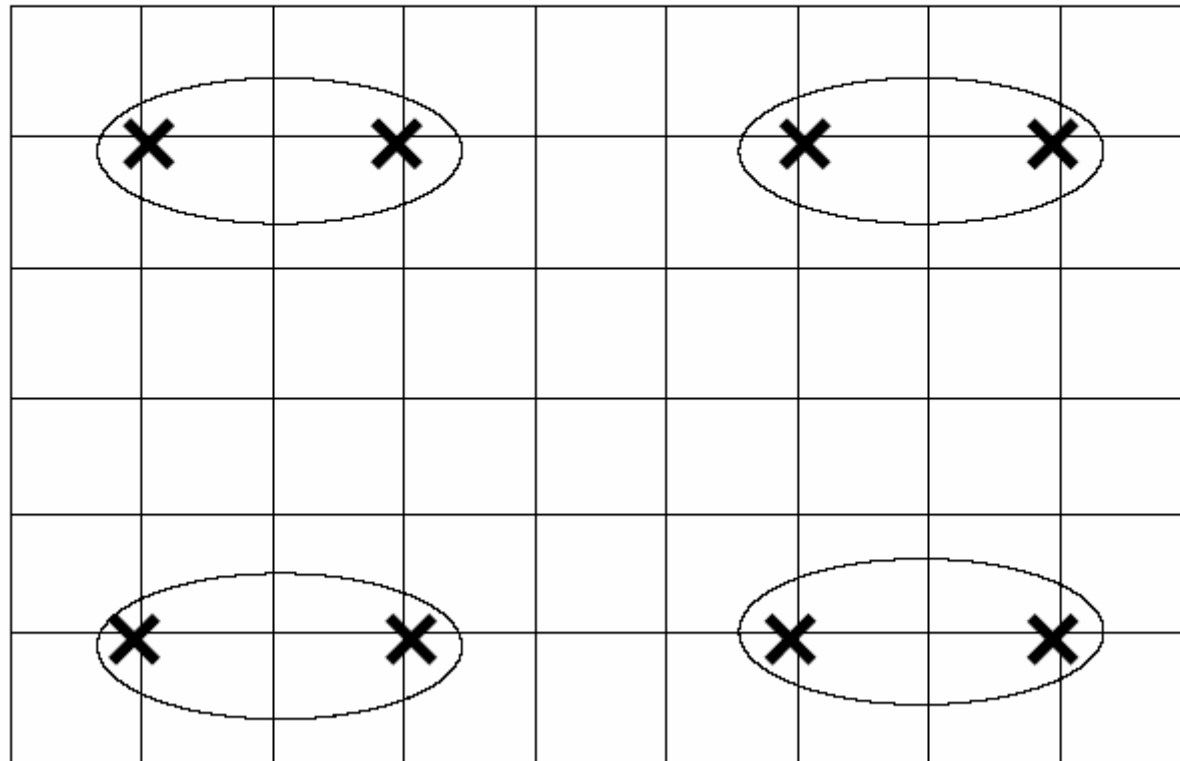
类的相似度度量

- 三种度量：
 - 单连接
 - 两个最近成员的相似度
 - 全连接
 - 两个最远成员的相似度
 - 组平均
 - 类成员的平均相似度
- 不同的度量会导致不同的聚类形状，适用于不同的问题
- 在大多数NLP问题中，基于全连接聚类更适用
- 基于组平均方法比全连接效率高，并且避免了单连接聚类的狭长形状

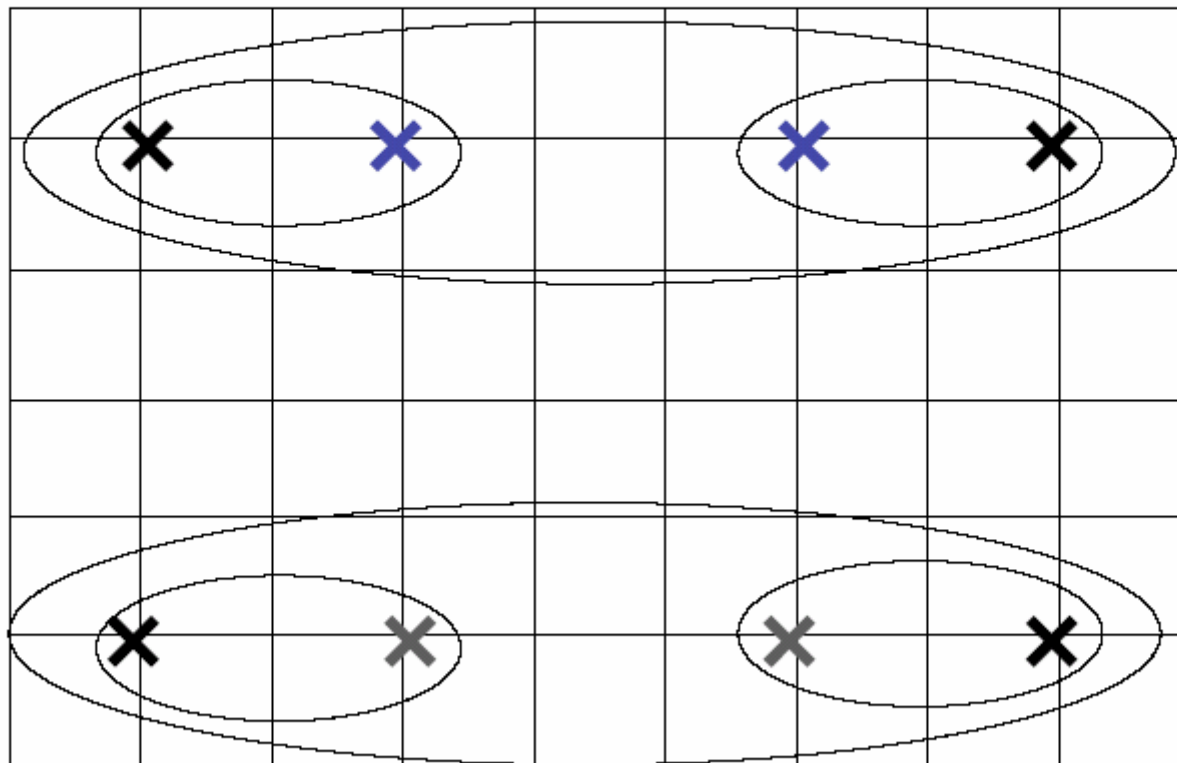
Single vs. Complete link

	x		x			x		x
	x		x			x		x

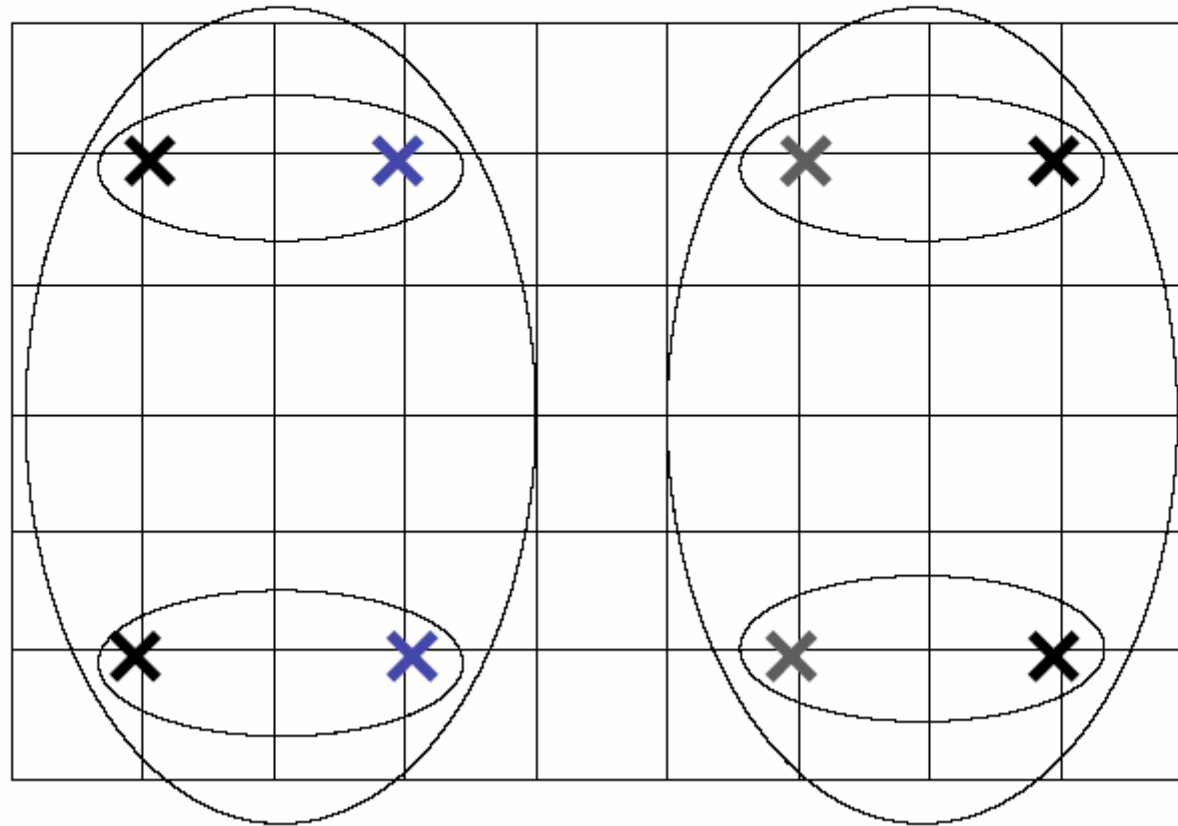
Single vs. Complete link



Single link



Complete link



应用：改进语言模型

- 聚类通过概化改进语言模型
- 通过聚类进行推理，扩大训练语料
- 使对稀疏事件处理据有较好的精度

非层次聚类

- 一般过程
 - 随机选择种子
 - 进行样本划分
 - 通过迭代将样本进行重新分配
 - 直到模型参数估计不再上升或呈下降趋势

非层次聚类

- K-均值
 - 硬聚类
 - 每个样本点完全属于某一类
 - 计算每个类的中心值
- 模糊k-均值
 - 软聚类
 - 每个样本点模糊隶属于某一类
 - 用EM算法计算 $P(c_i|w_1)$

K-均值

- 将n个向量分到k个类别中去

$$x_1=(2,1) \quad x_2=(1,3) \quad x_3=(6,7) \quad x_4=(4,7)$$

- 选择k个初始中心

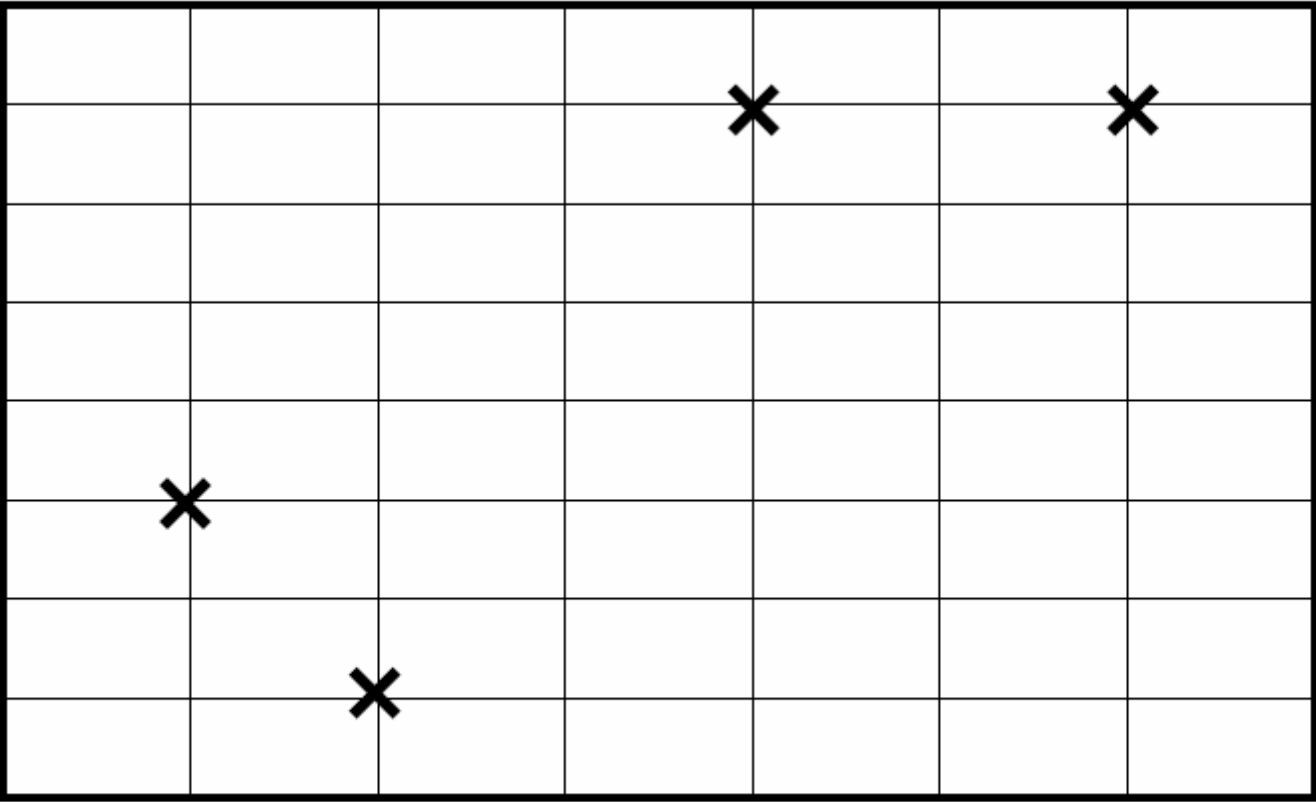
$$f_1=(4,3) \quad f_2=(5,5)$$

- 计算两项距离

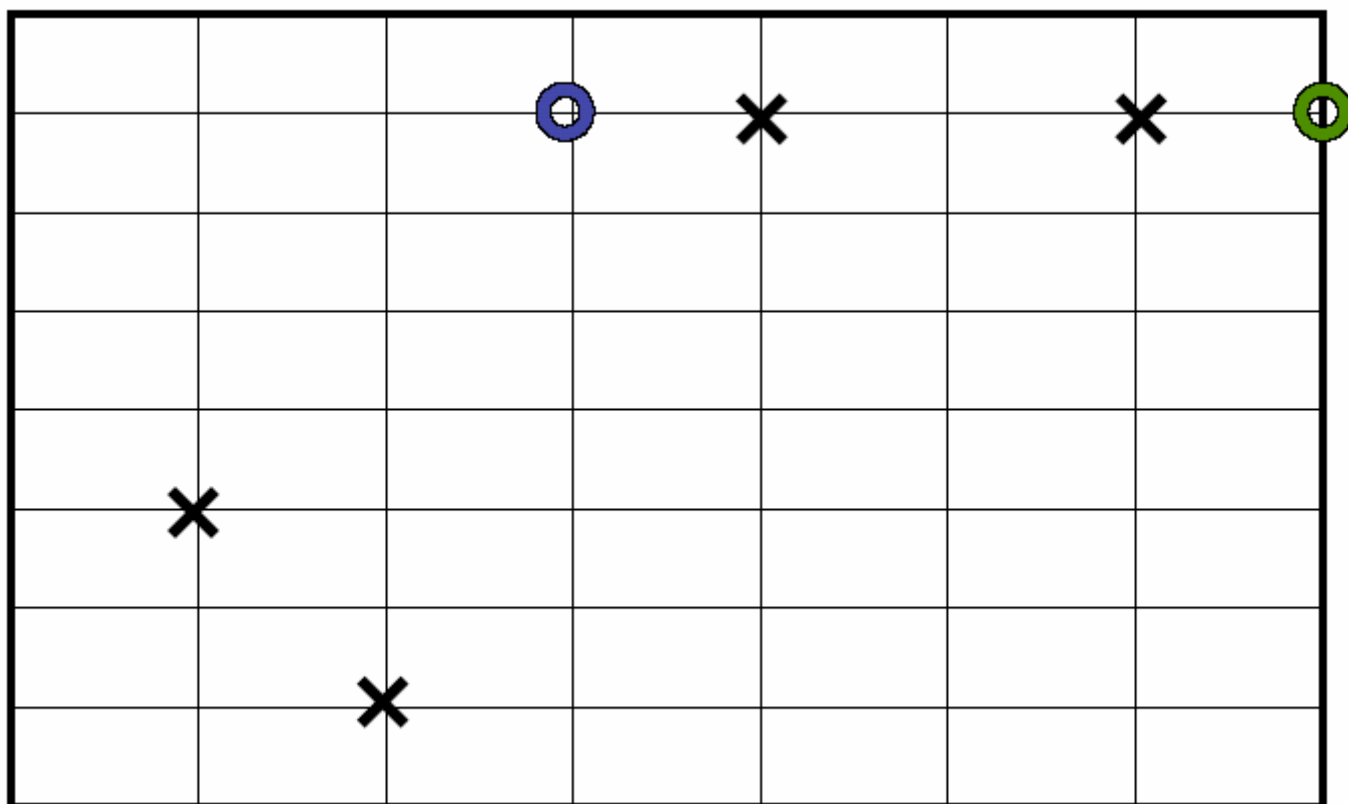
$$d(x_i, x_j) = \sqrt{\sum_{k=1}^n (x_{i_k} - x_{j_k})^2}$$

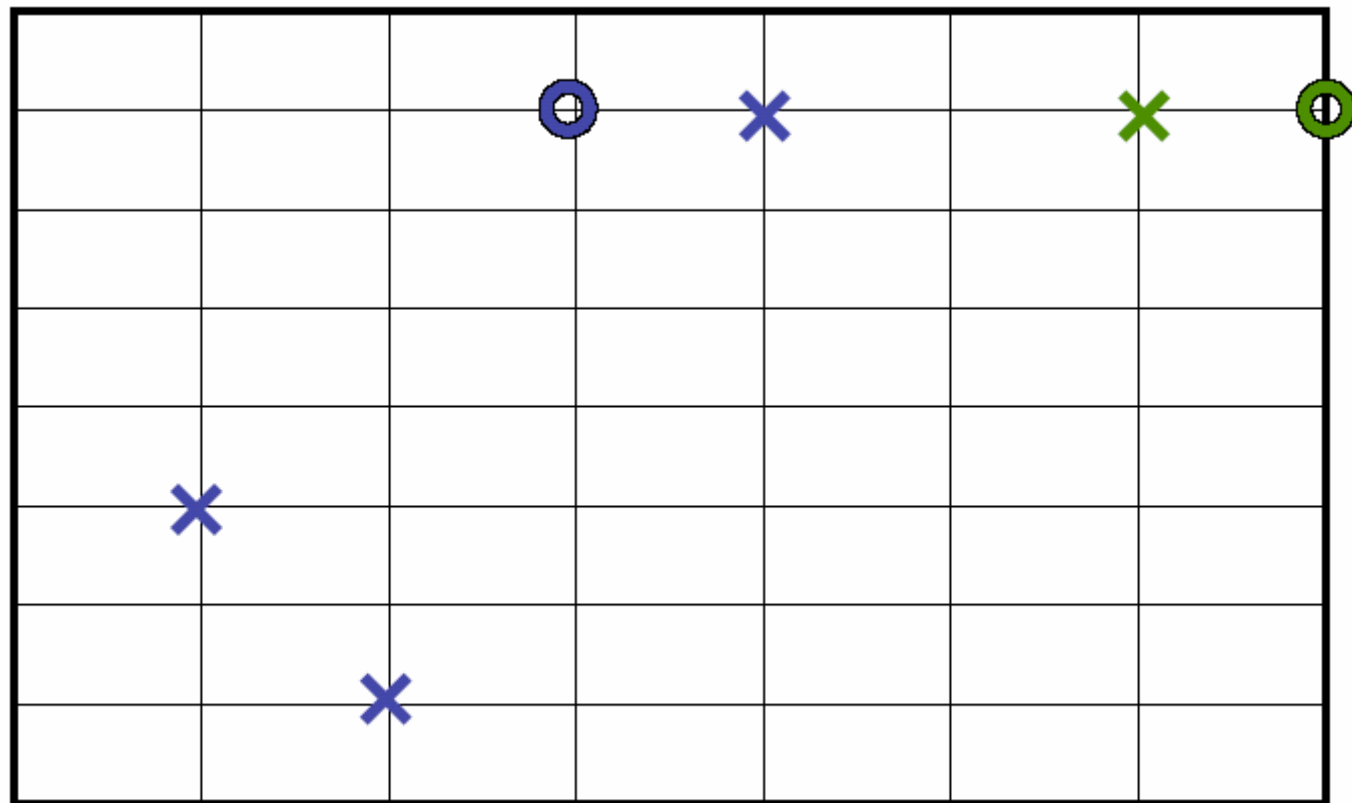
- 计算n个向量均值

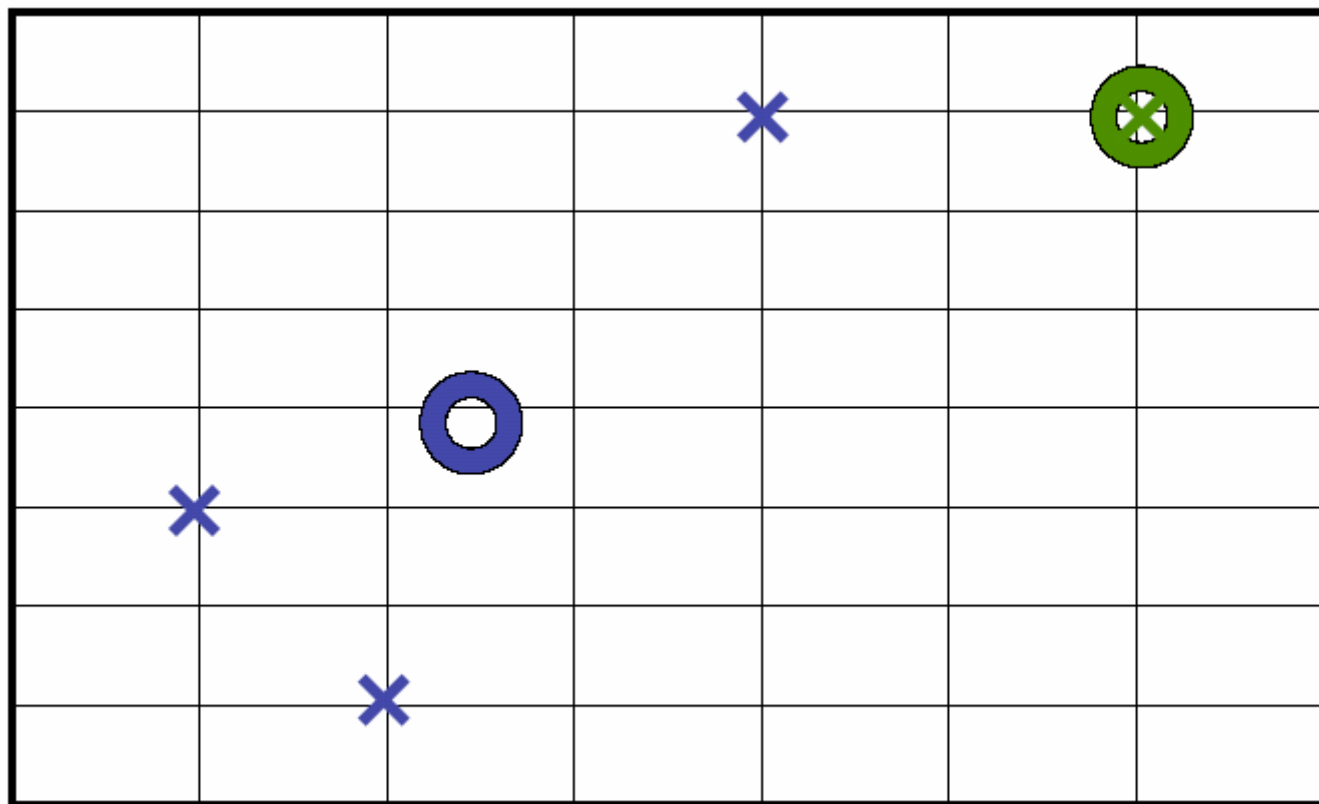
$$\mu(x_1, \dots, x_n) = \left(\frac{\sum_{i=1}^n x_{i_1}}{n}, \dots, \frac{\sum_{i=1}^n x_{i_m}}{n} \right)$$

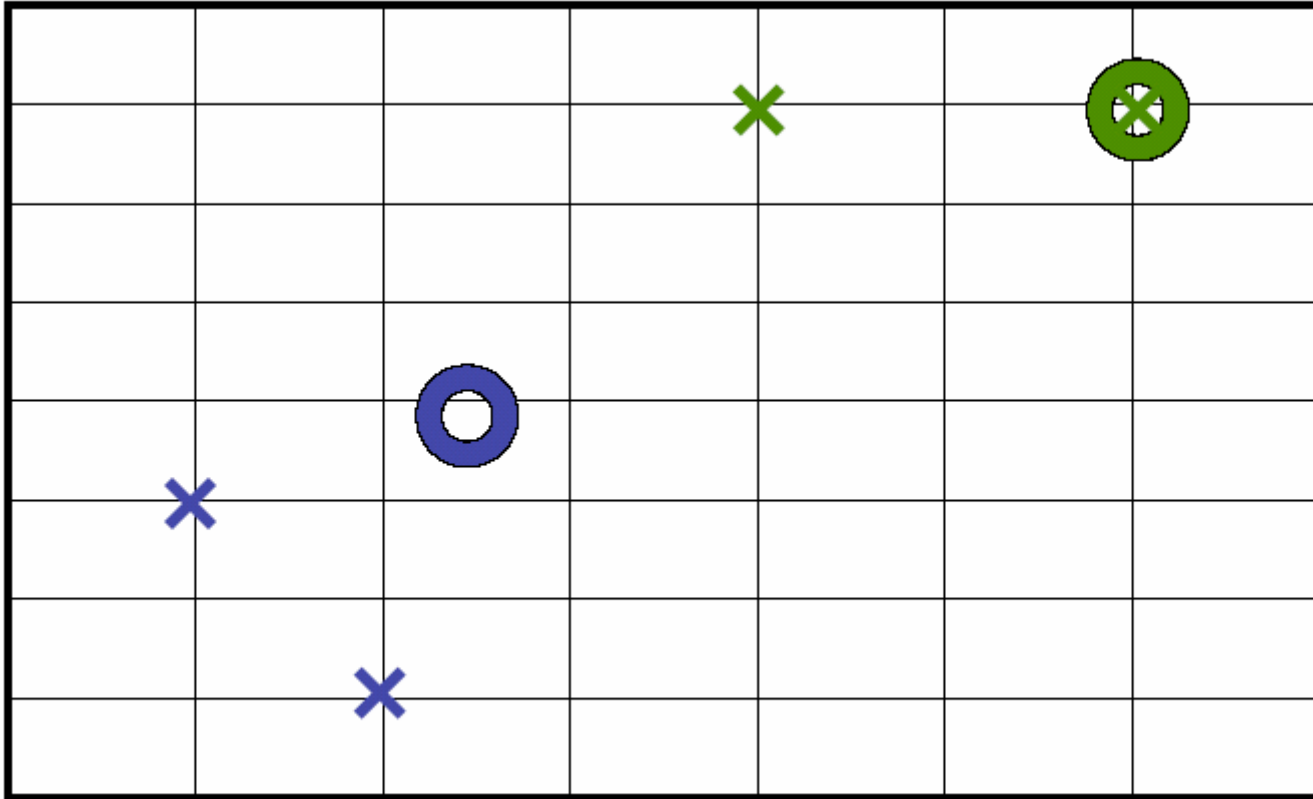


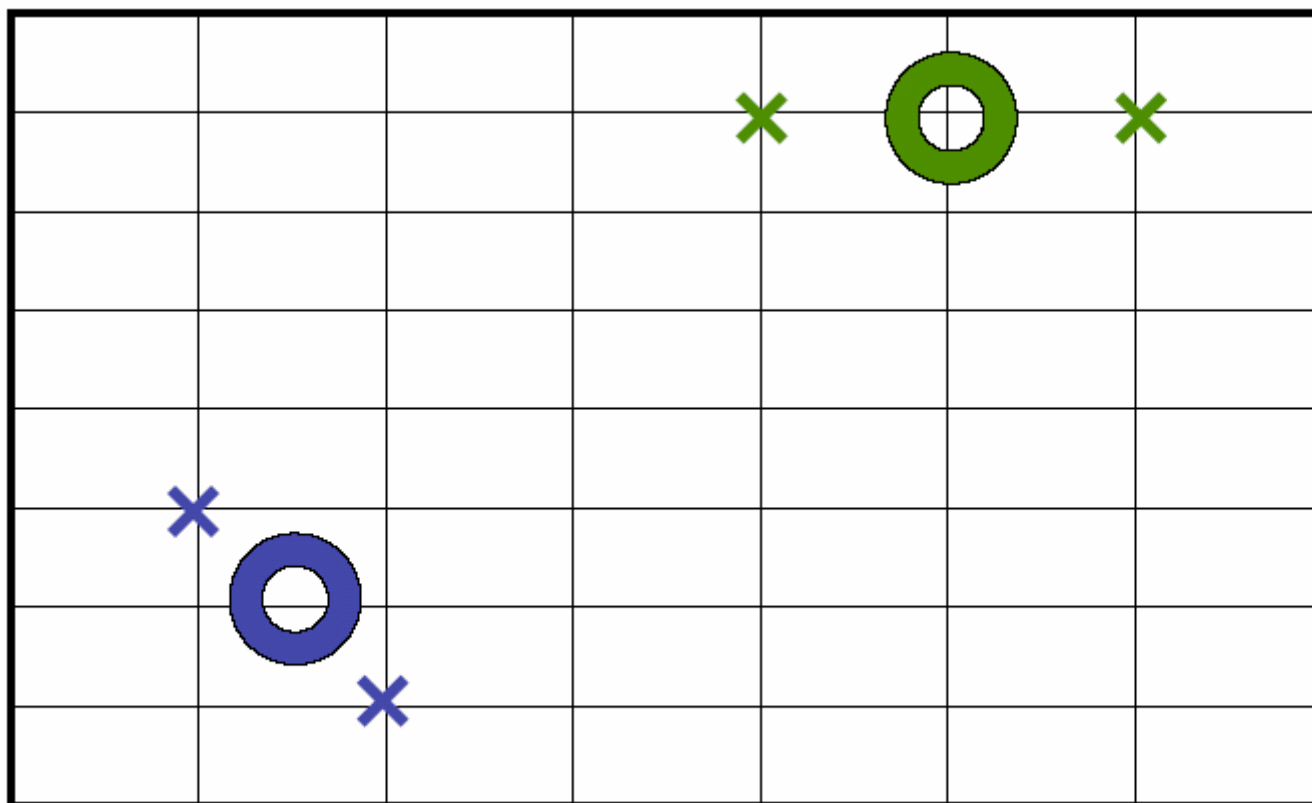
			X		X		
X							
		X					











K-均值算法

- 给定 k , k -均值算法包括4个步骤:
 - 将对象分成 k 个非空的子集
 - 计算每个类的平均值作为中心点.
 - 重新将对象, 将对象划分到离它最近的聚类
 - 重新计算聚类的中心, 重新划分对象, 直到所有的对象都不再发生变化.
- 注意与 k -中心点的区别

模糊聚类

- 经典的k-均值聚类算法在每一步迭代中，每一个样本点都被认为是完全属于某一类别
- 模糊聚类放松这一条件，假定每个样本是模糊隶属于某一类的
 - 每类是一个高斯分布
 - 样本集合模拟为高斯混合分布

EM算法

- 点集 x_1, \dots, x_n
- K 个类
- Z 为二维数组, z_{ij} 为1表示 x_i 在 j 类中, 否则为0
- 每个类定义为一个高斯分布

$$n(x; \mu_j, \sigma_j) = \frac{1}{\sqrt{2\pi}\sigma_j} \exp \left[\frac{-(x - \mu_j)^2}{2\sigma_j^2} \right]$$

EM算法

- 用先前的概率累加

$$\rho_j \cdot n(x; \mu_j, \sigma_j)$$

- 任意一项 x_i 的概率

$$p(x_i) = \sum_{j=1}^k \rho_j \cdot n(x_i, \mu_j, \sigma_j)$$

EM算法

- 参数

$$\theta_j = (\mu_j, \sigma_j, \rho_j)$$

$$\Theta = (\theta_1, \dots, \theta_k)$$

- 给定参数下x的值

$$L(X | \Theta) = \log \prod_i P(x_i) = \log \prod_i \sum_j \rho_j \cdot n(x_i; \mu_j, \sigma_j)$$
$$\sum_i \log \sum_j \rho_j \cdot n(x_i; \mu_j, \sigma_j)$$

EM算法

- 计算 z_{ij} 的期望值 (E-step)并用它计算最大似然估计(M-step), 反复迭代, 直到收敛。

$$E(z_{ij} | x_i; \Theta) = \frac{\rho_j \cdot n(x_i; \mu_j, \sigma_j)}{p(x_i)}$$

$$\mu'_j = \frac{\sum_i E(z_{ij} | x_i) \cdot x_i}{\sum_i E(z_{ij} | x_i)} \quad \sigma'_j = \frac{\sum_i E(z_{ij} | x_i) \cdot (x_i - \mu'_j)^2}{\sum_i E(z_{ij} | x_i)}$$

EM算法特点

- 算法族
- 可以用于任意的概率模型的参数估计
- 结果是局部最优的
- K-均值是用EM算法求解高斯混合分布的特例



分类

IRLAB

大纲

- 分类技术在自然语言中的应用
- 决策树
- 贝叶斯分类
- 最大熵模型
- K近邻
- 其他方法

自然语言中的分类问题

问题	对象	分类
词性标注	词的上下文	词性
词义消歧	词的上下文	词义
介词附着	句子	分析树
命名实体识别	句子	实体类别
作者识别	文档	作者
语言识别	文档	语言类型
文本分类	文档	主题

分类的一般过程

- 训练集
- 数学模型
- 训练过程
- 测试集
- 评价

本课介绍的几种方法

- 决策树
- 贝叶斯分类
- 最大熵模型
- K近邻
- 神经网络简介

决策树

- 简介
- 决策树表示法
- 决策树学习的适用问题
- 基本的决策树学习算法
- 决策树学习中的假想空间搜索
- 决策树学习的常见问题

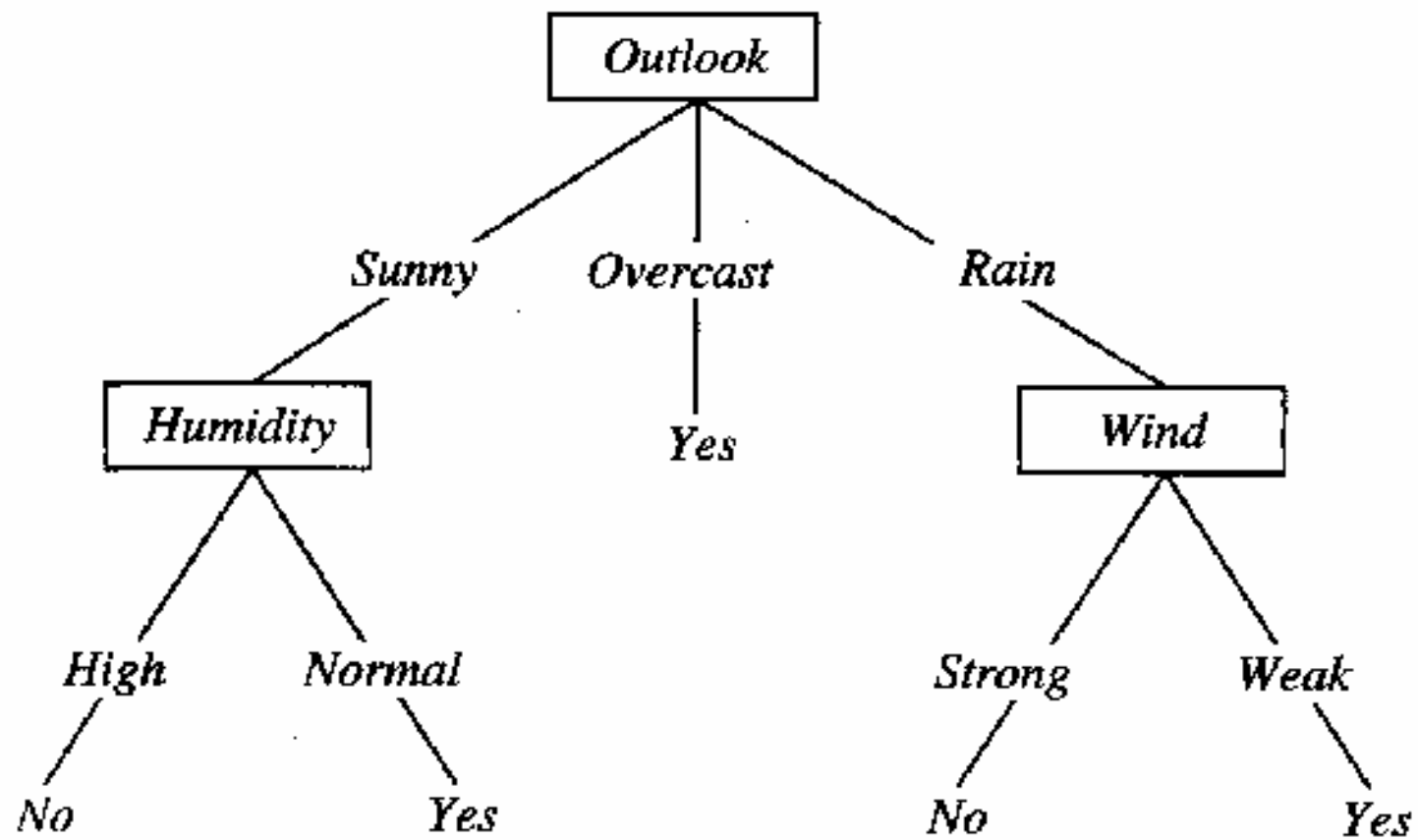
简介

- 决策树方法的起源是概念学习系统CLS，然后发展到ID3方法而为高潮，最后又演化为能处理连续属性的C4.5。有名的决策树方法还有CART和Assistant。
- 应用最广的归纳推理算法之一
- 一种逼近离散值目标函数的方法
- 对噪声数据有很好的健壮性且能学习析取表达式

决策树的表示法

- 决策树通过把实例从根节点排列到某个叶子节点来分类实例，叶子节点即为实例所属的分类。
- 树上的每一个节点说明了对实例的某个属性的测试，并且该节点的每一个后继分支对应于该属性的一个可能值

决策树表示举例



表达式

$(\text{Outlook} = \text{Sunny} \wedge \text{Humidity} = \text{Normal})$

$\vee (\text{Outlook} = \text{Overcast})$

$\vee (\text{Outlook} = \text{Rain} \wedge \text{Wind} = \text{Weak})$

决策树学习的适用问题

- 实例是由属性-值对表示的
- 目标函数具有离散的输出值
- 可能需要析取的描述
- 训练数据可以包含错误
- 训练数据可以包含缺少属性值的实例

属性选择

- 构造好的决策树的关键在于如何选择好的逻辑判断或属性。
- 对于同样一组例子，可以有很多决策树能符合这组例子。
- 一般情况下或具有较大概率地说，树越小则树的预测能力越强。
- 要构造尽可能小的决策树，关键在于选择恰当的逻辑判断或属性。
- 由于构造最小的树是NP-难问题，因此只能采取用启发式策略选择好的逻辑判断或属性

用熵度量样例的均一性（纯度）

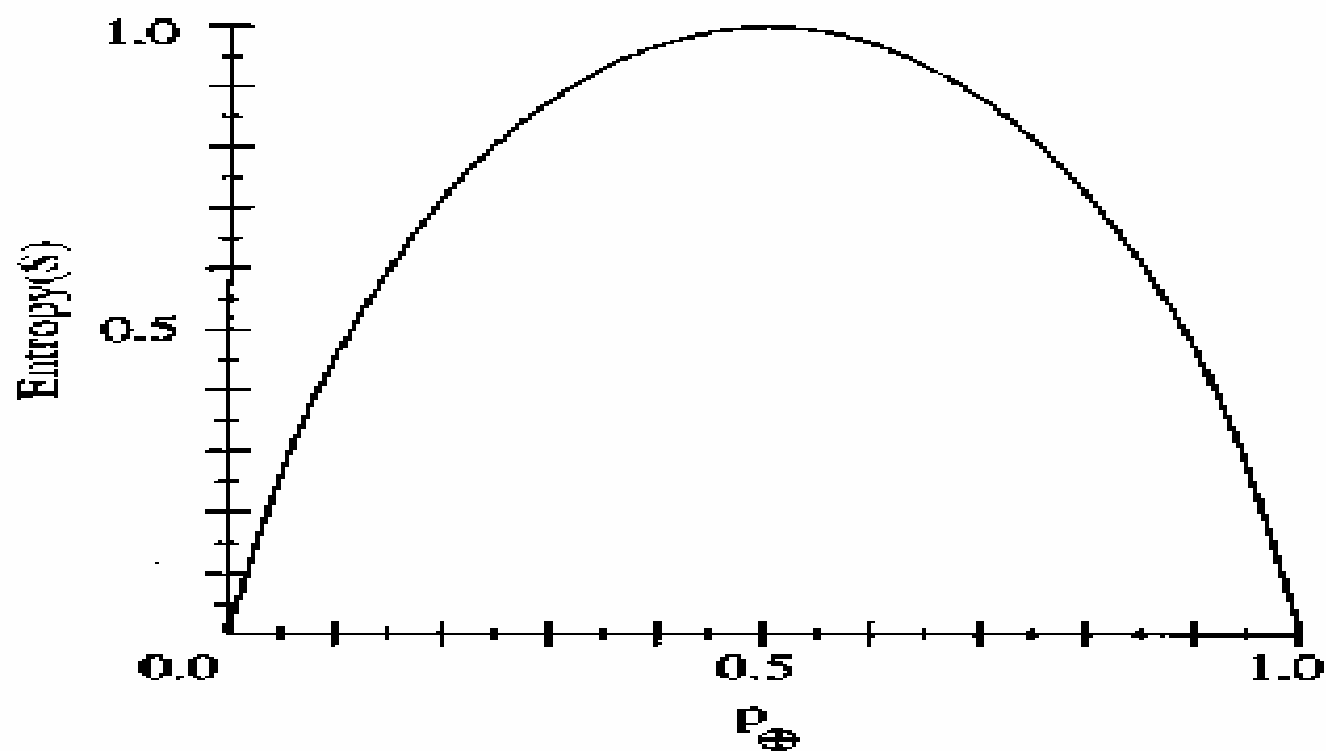
- 熵的定义

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

- 举例

$$\begin{aligned} Entropy([9+, 5-]) &= -(9/14) \log_2(9/14) - (5/14) \log_2(5/14) \\ &= 0.940 \end{aligned}$$

关于某布尔分类的熵函数



用信息增益度量期望熵最低

- 一个属性的信息增益就是由于使用这个属性分割样例而导致的期望熵的降低

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

举例

$Values(Wind) = Weak, Strong$

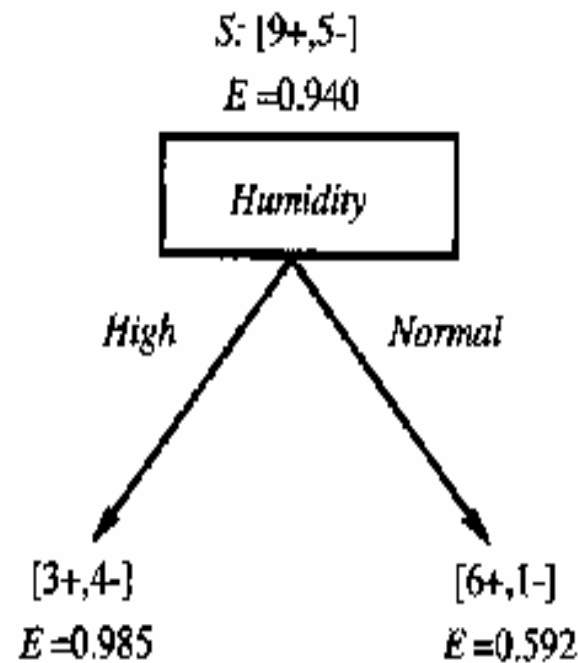
$S = [9+, 5-]$

$S_{Weak} \leftarrow [6+, 2-]$

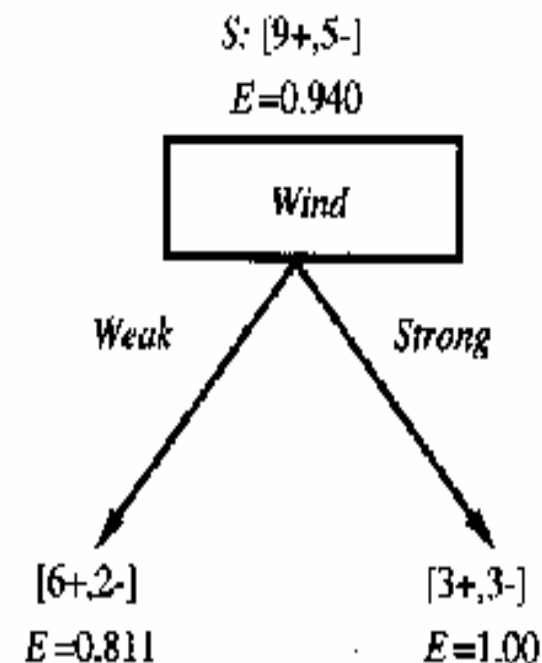
$S_{Strong} \leftarrow [3+, 3-]$

$$\begin{aligned} Gain(S, Wind) &= Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} Entropy(S_v) \\ &= Entropy(S) - (8/14) Entropy(S_{Weak}) \\ &\quad - (6/14) Entropy(S_{Strong}) \\ &= 0.940 - (8/14)0.811 - (6/14)1.00 \\ &= 0.048 \end{aligned}$$

Which attribute is the best classifier?



$$\begin{aligned} \text{Gain}(S, \text{Humidity}) &= .940 - (7/14) \cdot 0.985 - (7/14) \cdot 0.592 \\ &= .151 \end{aligned}$$

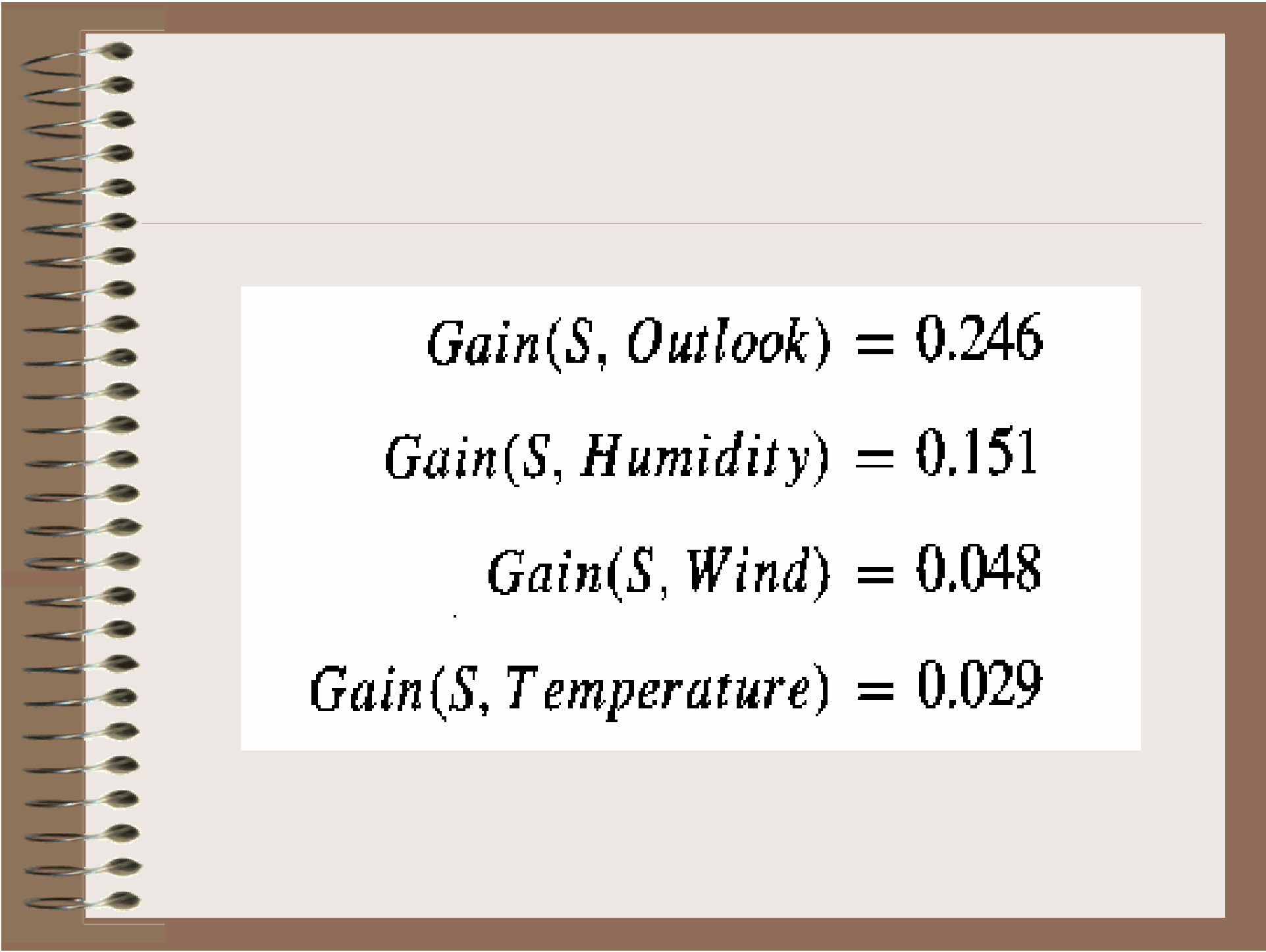


$$\begin{aligned} \text{Gain}(S, \text{Wind}) &= .940 - (8/14) \cdot 0.811 - (6/14) \cdot 1.0 \\ &= .048 \end{aligned}$$

Day	<i>Outlook</i>	<i>Temperature</i>	<i>Humidity</i>	<i>Wind</i>	<i>PlayTennis</i>
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

TABLE 3.2

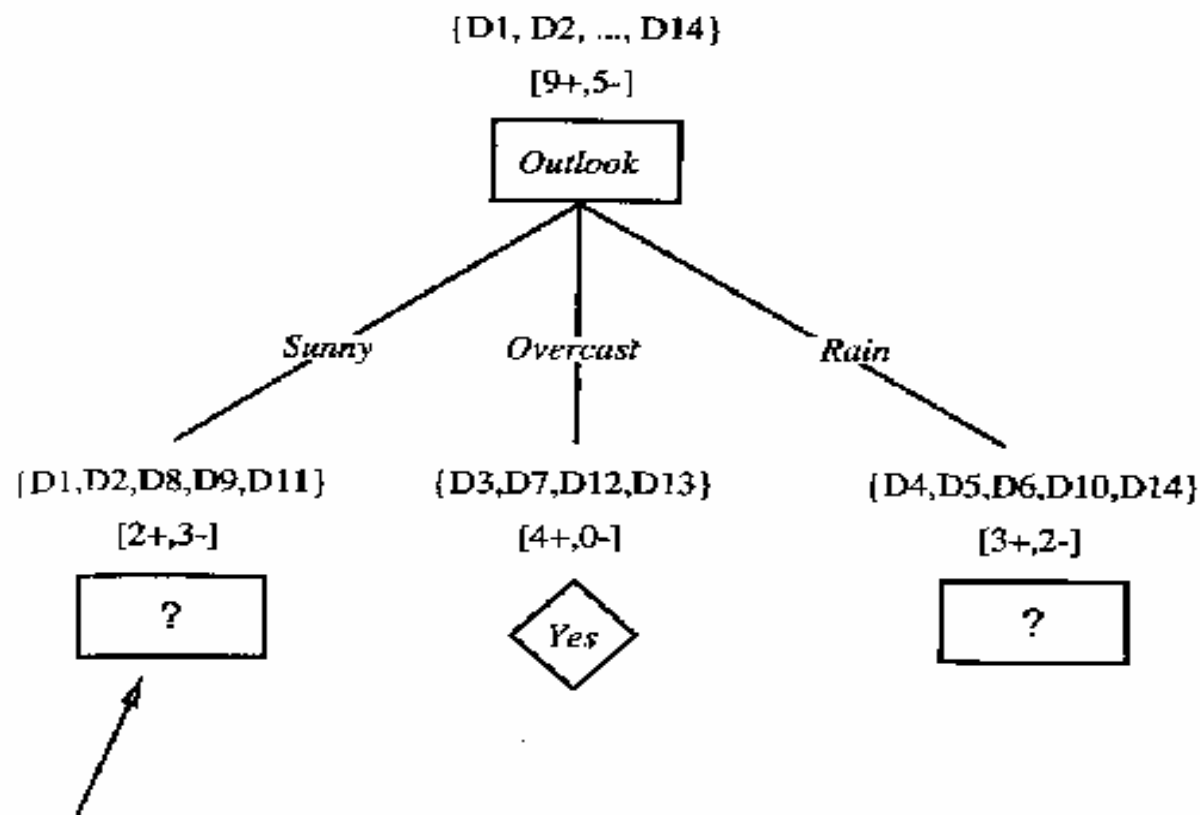
Training examples for the target concept *PlayTennis*.

A spiral-bound notebook with a brown cover and a light beige page. The spiral binding is on the left side. A white rectangular box is centered on the page, containing four lines of text.
$$\textit{Gain}(S, \textit{Outlook}) = 0.246$$

$$\textit{Gain}(S, \textit{Humidity}) = 0.151$$

$$\textit{Gain}(S, \textit{Wind}) = 0.048$$

$$\textit{Gain}(S, \textit{Temperature}) = 0.029$$



Which attribute should be tested here?

$$S_{\text{sunny}} = \{D1, D2, D8, D9, D11\}$$

$$\text{Gain}(S_{\text{sunny}}, \text{Humidity}) = .970 - (3/5) 0.0 - (2/5) 0.0 = .970$$

$$\text{Gain}(S_{\text{sunny}}, \text{Temperature}) = .970 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = .570$$

$$\text{Gain}(S_{\text{sunny}}, \text{Wind}) = .970 - (2/5) 1.0 - (3/5) .918 = .019$$

ID3算法

创建树的Root结点

开始

$A \leftarrow$ Attributes中分类能力最好的属性

Root的决策属性 $\leftarrow A$

对于每个可能值 v_i

在Root下加一个新的分支对应测试 $A=v_i$

令 $Examples_{v_i}$ 为 Examples 中满足 A 属性值为 v_i 的子集

如果 $Examples_{v_i}$ 为空

在这个新分支下加一个叶子结点, 节点的label=Examples中最普遍的目标属性值(target-attribute)

否则在这个新分支下加一个子树 $ID3(example_{v_i}, target-attribute, attributes-\{A\})$

返回 Root

C4.5

- C4.5是对ID3的改进算法
 - 对连续值的处理
 - 对未知特征值的处理
 - 对决策树进行剪枝
 - 规则的派生

决策树学习中的假设空间搜索

- 假设空间
 - 可能的决策树集合
- ID3算法中的假设空间包含所有的决策树
- 基本的ID3算法在搜索中不进行回溯
- ID3算法在搜索的每一步都使用当前的所有训练样例

决策树学习的常见问题(1)

- 避免过度拟合数据
 - 基本的决策树构造算法没有考虑噪声，生成的决策树完全与训练例子拟合
 - 有噪声情况下，完全拟合将导致过分拟合（Overfitting），即对训练数据的完全拟合反而不具有很好的预测性能。

解决方法

- 剪枝是一种克服噪声的技术，同时它也能使树得到简化而变得更容易理解。
 - 向前剪枝 (forward pruning)
 - 向后剪枝 (backward pruning)
- 理论上讲，向后剪枝好于向前剪枝，但计算复杂度大。
- 剪枝过程中一般要涉及一些统计参数或阈值，如停机阈值
- 有人提出了一种和统计参数无关的基于最小描述长度 (MDL) 的有效剪枝法

决策树学习的常见问题（2）

- 合并连续值属性
 - 把连续值的决策属性的值域分割为离散的区间集合，动态创建一个新的布尔属性 A_c 。
- 属性选择的其他度量标准
 - 信息增益比（gain ratio）、Gini-index、距离度量（distance measure）等。不同的度量有不同的效果，特别是对于多值属性。

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

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

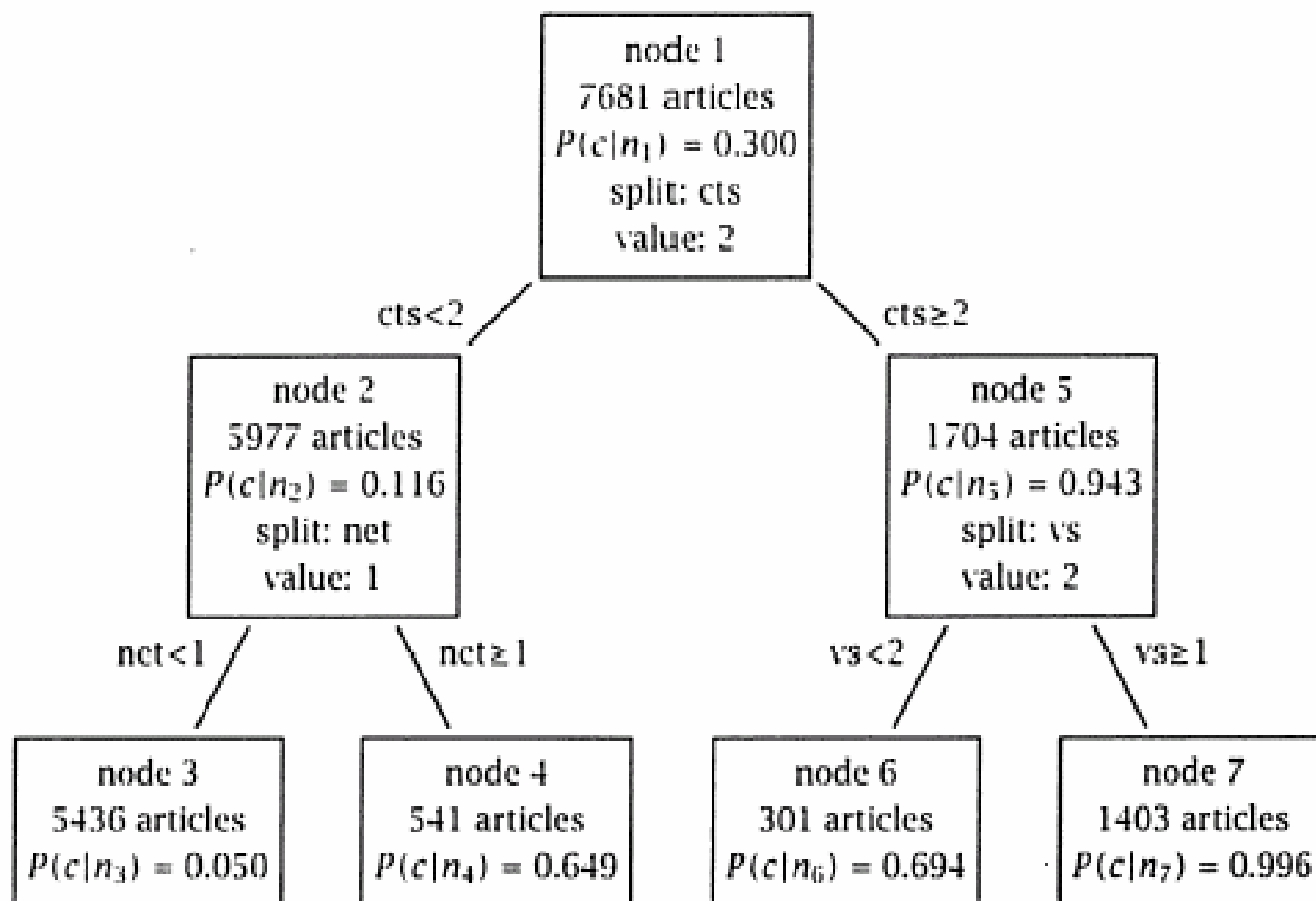
决策树的优点

- 可以生成可以理解的规则
- 计算量相对来说不是很大
- 可以处理连续和离散字段
- 决策树可以清晰的显示哪些字段比较重要
- 过程可见

不足之处

- 对连续性的字段比较难预测
- 当类别太多时，错误可能会增加的比较快
- 一般的算法分类的时候，只是根据一个属性来分类。
- 不是全局最优

举例：利用决策树进行文本分类



entropy at node 1, $P(C N) = 0.300$	0.611
entropy at node 2, $P(C N) = 0.116$	0.359
entropy at node 5, $P(C N) = 0.943$	0.219
weighted sum of 2 and 5	$\frac{5977}{7681} \times 0.359 + \frac{1704}{7681} \times 0.219 = 0.328$
information gain	$0.611 - 0.328 = 0.283$

Table 16.4 An example of information gain as a splitting criterion. The table shows the entropies for nodes 1, 2, and 5 in figure 16.1, the weighted sum of the child nodes and the information gain for splitting 1 into 2 and 5.

"earnings" assigned?	"earnings" correct?	
	YES	NO
YES	1024	69
NO	63	2143

Table 16.5 Contingency table for a decision tree for the Reuters category "earnings." Classification accuracy on the test set is 96.0%.

贝叶斯分类

- 给定训练集 D , h 的后验概率 $P(h|D)$ 遵循下面的 Bayes 定理

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- MAP (最大后验概率) 假定

$$h_{MAP} \equiv \operatorname{argmax}_{h \in H} P(h|D) = \operatorname{argmax}_{h \in H} P(D|h)P(h).$$

- 实际困难: 需要概率知识, 明显的计算代价

朴素Bayes分类器 (I)

- 假定：属性值对给定类的影响独立于其他属性

$$P(C_j|V) \propto P(C_j) \prod_{i=1}^n P(v_i|C_j)$$

- 如果只计算单个属性值的分布，大大地减少了计算量

朴素Bayesian分类器 (II)

给定训练集, 我们计算概率

Outlook	P	N		Humidity	P	N
sunny	2/9	3/5		high	3/9	4/5
overcast	4/9	0		normal	6/9	1/5
rain	3/9	2/5				
Temperature				Windy		
hot	2/9	2/5		true	3/9	3/5
mild	4/9	2/5		false	6/9	2/5
cool	3/9	1/5				

Bayesian分类

- 使用一个后验概率:

$P(C|X)$ = 样本 $X = \langle x_1, \dots, x_k \rangle$ 属于类 C 的概率

E.g. $P(\text{class}=\text{N} \mid \text{outlook}=\text{sunny}, \text{windy}=\text{true}, \dots)$

- 如果 $P(C|X)$ 是最大的 则指定 X 的类标识为 C

估算后验概率

- Bayes 定理:

$$P(C|X) = P(X|C) \cdot P(C) / P(X)$$

- $P(X)$ 对所有类来说，是一个常数
- $P(C)$ = C 类样本的相对频率
- C such that $P(C|X)$ is 最大 =
C such that $P(X|C) \cdot P(C)$ is 最大
- 问题: 计算 $P(X|C)$ 是不可行的!

朴素Bayesian分类

- 朴素假定：属性之间是独立的

$$P(x_1, \dots, x_k | C) = P(x_1 | C) \cdot \dots \cdot P(x_k | C)$$

- 如果 i -th 属性是分类属性：

$P(x_i | C) = s_{ik} / s_i$, s_{ik} 是在 i -th 属性上具有值 x_i 的类 C_i 的训练样本数，而 s_i 是 C_i 中训练样本数

- 如果 i -th 属性是 连续的：

$P(x_i | C)$ 用高斯密度函数估计

- 在这两种情况下，计算变得容易

Play-tennis 例子: 估算 $P(x_i|C)$

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

$$P(p) = 9/14$$

$$P(n) = 5/14$$

outlook	
$P(\text{sunny} \text{p}) = 2/9$	$P(\text{sunny} \text{n}) = 3/5$
$P(\text{overcast} \text{p}) = 4/9$	$P(\text{overcast} \text{n}) = 0$
$P(\text{rain} \text{p}) = 3/9$	$P(\text{rain} \text{n}) = 2/5$
temperature	
$P(\text{hot} \text{p}) = 2/9$	$P(\text{hot} \text{n}) = 2/5$
$P(\text{mild} \text{p}) = 4/9$	$P(\text{mild} \text{n}) = 2/5$
$P(\text{cool} \text{p}) = 3/9$	$P(\text{cool} \text{n}) = 1/5$
humidity	
$P(\text{high} \text{p}) = 3/9$	$P(\text{high} \text{n}) = 4/5$
$P(\text{normal} \text{p}) = 6/9$	$P(\text{normal} \text{n}) = 2/5$
windy	
$P(\text{true} \text{p}) = 3/9$	$P(\text{true} \text{n}) = 3/5$
$P(\text{false} \text{p}) = 6/9$	$P(\text{false} \text{n}) = 2/5$

Play-tennis例子: 分类 X

- 例子 $X = \langle \text{rain}, \text{hot}, \text{high}, \text{false} \rangle$
- $P(X|p) \cdot P(p) =$
 $P(\text{rain}|p) \cdot P(\text{hot}|p) \cdot P(\text{high}|p) \cdot P(\text{false}|p) \cdot P(p) =$
 $3/9 \cdot 2/9 \cdot 3/9 \cdot 6/9 \cdot 9/14 = 0.010582$
- $P(X|n) \cdot P(n) =$
 $P(\text{rain}|n) \cdot P(\text{hot}|n) \cdot P(\text{high}|n) \cdot P(\text{false}|n) \cdot P(n) =$
 $2/5 \cdot 2/5 \cdot 4/5 \cdot 2/5 \cdot 5/14 = 0.018286$
- Sample **X** is classified in class **n** (don't play)

举例

- 在Joachims(1996)的一个实验中，被应用于分类新闻组文章
- 20个电子新闻组，每个新闻组1000篇文章，形成2万个文档的数据集
- 2/3作训练集，1/3作测试集衡量性能
- 20个新闻组，随机猜测的分类精确度5%，由程序获得的精确度89%

讨论

- 朴素的贝叶斯假定在一个位置上出现的词的概率独立与另外一个位置的单词，这个假定有时并不反映真实情况
- 虽然独立性假设很不精确，别无选择，否则计算的概率项将极为庞大
- 幸运的是，在实践中朴素贝叶斯学习器在许多文本分类中性能非常好，即使独立性假设不成立

最大熵原理

- 熵定量的描述事物的不确定性
- 设随机变量，它有 A_1, A_2, \dots, A_n 共 n 个可能的结局，每个结局出现的机率分别为 p_1, p_2, \dots, p_n ，则的不确定程度，即信息熵为：

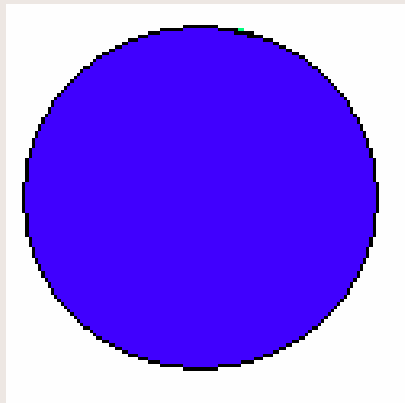
$$H(\xi) = -\sum_{i=1}^n p_i \log p_i$$

- 熵越大，越不确定
- 熵等于0，变量是确定的

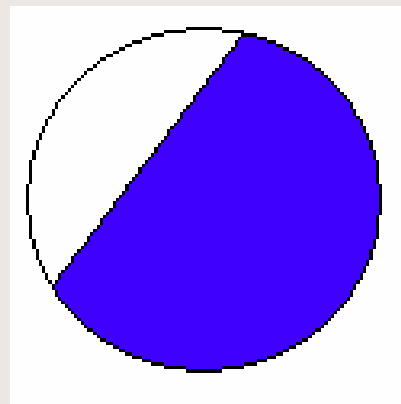
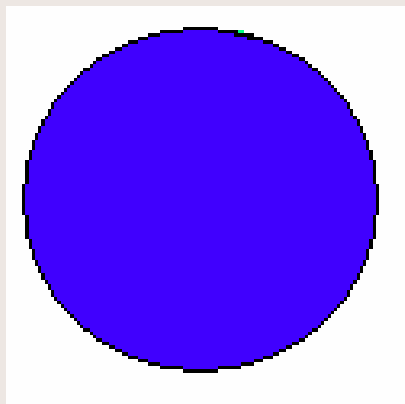
最大熵的基本思想

- 事物是约束和自由的统一体
- 熵是状态的不确定性的定量标尺，事物的状态越自由，熵越大。
- 最大熵原则认为：任何物质系统除了都受到或多或少的外部约束外，其内部总是具有一定的自由度，这种自由度导致系统内的各元素处于不同的状态。熵最大就是事物状态的丰富程度自动达到最大值。
- 事物总是在约束下争取（或呈现）最大的自由权，这其实也是自然界的根本原则。

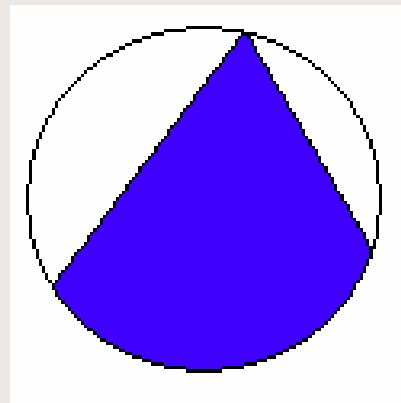
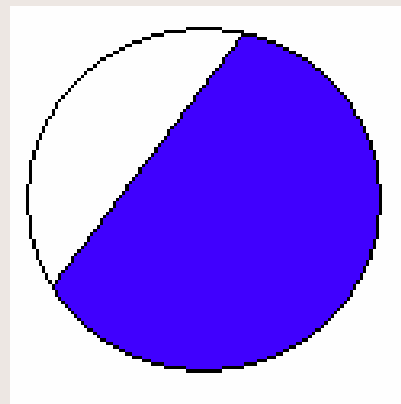
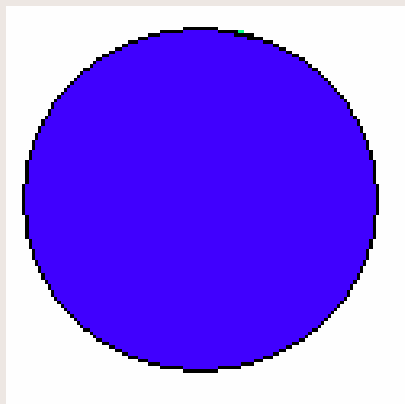
最大熵原则下点的分布



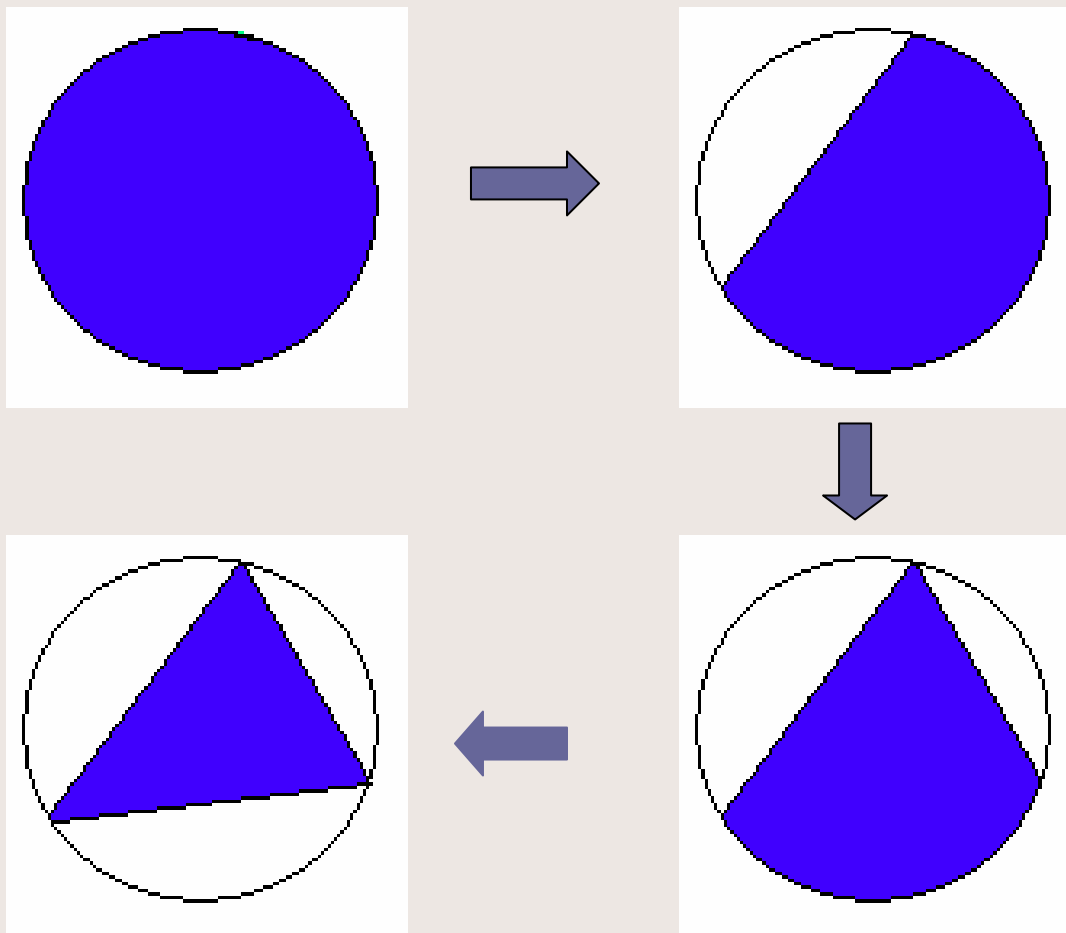
最大熵原则下点的分布



最大熵原则下点的分布



最大熵原则下点的分布



最大熵原则可用于模型选择

- 研究某个随机事件，即研究该事件的分布规律。
- 当随机事件的真实分布难以获得时，构造统计模型预测事件的分布规则。
- 构造的统计模型基于对事件已知的信息。
- 满足已知信息要求的模型可能有多个。
- 选择熵最大的模型是模型选择的一种方法。

为什么基于最大熵原则

- 事物总是在满足约束的条件下，使状态的丰富程度自动达到最大值
- Jaynes证明：对随机事件的所有相容的预测中，熵最大的预测出现的概率占绝对优势。
- 最大熵原则体现了“知之为知之，不知为不知”的处理问题的方法。

关于分数的例子

- 已知一个班的学生成绩有三个分数档：80分、90分、100分
- 又知平均成绩为90分。
- 这种情况下，三种分数档的概率分布并不是唯一的（有无数组解）
 - 80分（10%）、90分（80%）、100分（10%）
 - 80分（40%）、90分（20%）、100分（40%）
- 挑选标准可以使用最大信息熵原理
 - 80分（33%）、90分（33%）、100分（33%）

基于最大熵的统计建模

- 统计建模是根据已知输入和它所对应的输出，用统计方法学习输入与输出之间的关系。
- 基于最大熵的统计建模即发现满足已知条件的熵最大的模型

统计建模的基本步骤

- 特征空间的确定
 - 对于统计模型而言，只有特征不同的样本才是有区别的样本
- 特征选择
 - 原始特征可能数据过大，需从特征空间中挑选出最有效的特征以降低特征空间维数
- 根据选择的特征构建统计模型

基于最大熵原则的建模

- 遵循统计建模的基本步骤
- 特别之处在于建立的模型为熵最大的模型

在自然语言中的应用

- S.Pietra、V.Pietra等人提出了一种基于最大熵原理的单词聚类方法，首次将最大熵理论应用于自然语言处理
- A.L.Berger、S.Pietra、V.Pietra等人比较详细地介绍了最大熵的理论框架，并介绍了其在基于统计的机器翻译领域的一些应用
- S.Abney在统计属性--值文法(Attribute-value Grammars)中使用最大熵进行参数估计
- 李涓子、黄昌宁改进了最大熵的特征选择策略，并将其应用于汉语的词义消歧，取得了较好的效果
- A. Borthwick研究了基于最大熵的名实体(Named Entity)的识别
- 黄萱婧等研究的基于最大熵模型的名词短语识别

最大熵模型的符号与定义

- 已知训练样本集 $(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$, 其中 x 为输入, y 为输出
 $\tilde{p}(y | x)$ 指 x 出现的情况下, y 的经验概率, 也就是 y 在样本集中的概率

$p(y | x)$ 指 x 出现的情况下, y 的实际概率

- 随机事件的不确定性可以用条件熵来衡量:

$$H(p) \equiv - \sum_{x,y} \tilde{p}(x) p(y | x) \log p(y | x)$$

- 特征指 x 与 y 之间存在的某种特定关系, 可以用一个输出为0或1的特征函数表示。

- $$f(x, y) = \begin{cases} 1 & \text{if } x=1, y=1 \\ 0 & \text{otherwise} \end{cases}$$

最大熵模型的符号与定义

- 特征的经验概率为所有满足特征要求的 (x,y) 的经验概率之和,即:

$$\tilde{p}(f) = \sum_{x,y} \tilde{p}(x,y) f(x,y)$$

- 特征的期望概率,也就是特征在我们所学习的随机事件中的真实分布为:

$$p(f) = \sum_{x,y} \tilde{p}(x) p(y|x) f(x,y)$$

最大熵模型

- 选定的特征的重要性可通过下式体现：

$$p(f) = \tilde{p}(f)$$

- 体现了在模型中最大化表现已知信息的原则
- 上式表示，特征 f 的经验概率与期望概率一致，当样本足够多时，可信度高的特征的经验概率与期望概率是一致的，即：

$$\sum_{x,y} \tilde{p}(x) p(y|x) f(x,y) = \sum_{x,y} \tilde{p}(x,y) f(x,y)$$

- 根据随机事件的情况，约束等式可以有多组，约束等式的集合叫约束集，可表示为

$$C \equiv \{ p(f_i) = \tilde{p}(f_i) , i \in \{1, 2, \dots, n\} \}$$

- 最大熵模型，是满足约束集条件的所有模型中熵最大的模型，即：

$$p^* = \arg \max H(p)$$

其中 p 为满足约束集 C 条件的某一统计模型。

最大熵模型

- 因为约束集中的每一个特征的分布是最大似然估计，所以约束集中元素越多，统计模型从训练样本中学得的越多，其做出的预测也越依赖于样本集。
- 选择特征较多时，满足约束集要求的统计模型个数较少，当把样本中的所有 (x, y) 都作为特征时，模型唯一，为用极大似然估计求 $p(y|x)$ 所建立的模型。

最大熵模型求解

- 最大熵模型求解问题，实质是一个约束条件下求极值的问题。此类问题通常用拉格朗日乘子法确定：

$$\Lambda(p, \lambda) \equiv H(p) + \sum_i \lambda_i (p(f_i) - \tilde{p}(f_i))$$

- 其中：

$$H(p) \equiv - \sum_{x,y} \tilde{p}(x) p(y | x) \log p(y | x)$$

$$\tilde{p}(f) = \sum_{x,y} \tilde{p}(x, y) f(x, y)$$

$$p(f) = \sum_{x,y} \tilde{p}(x) p(y | x) f(x, y)$$

- 求导后变换得

$$p_{\lambda}(y | x) = \frac{1}{Z_{\lambda}(x)} \exp(\sum_i \lambda_i f_i(x, y))$$

其中

$$Z_{\lambda}(x) \equiv \sum_y \exp(\sum_i \lambda_i f_i(x, y))$$

- 最大值可通过求 $\lambda^* = \arg \max_{\lambda} \Psi(\lambda)$

$$\Psi(\lambda) \equiv \Lambda(p_{\lambda}, \lambda)$$

求解算法

- λ^* 没有解析解，Danroch 和Rateliff于1972年提出了一个称为GIS(Generalized Iterative Scaling Algorithm)算法。
- D.Pietra等改进了原有的最大熵模型求解算法，降低了求解算法的约束条件，提出了IIS(Improved Iterative Scaling Algorithm)算法，增加了算法的适用性
- IIS算法是目前最大熵参数求解中的常用算法。

最大熵统计模型的特点

- 最大熵统计模型获得的是所有满足约束条件的模型中信息熵极大的模型
- 其次最大熵统计模型可以灵活地设置约束条件。通过约束条件的多少可以调节模型对未知数据的适应度和对已知数据的拟合程度
- 另外最大熵模型还自然地解决了统计模型中参数平滑的问题

$$p_{\lambda}(y | x) = \frac{1}{Z_{\lambda}(x)} \exp\left(\sum_i \lambda_i f_i(x, y)\right)$$

举例:文本分类

Word w^i	Feature weight $\log \alpha_i$
vs	0.613
mln	-0.110
cts	1.298
:	-0.432
&	-0.429
000	-0.413
loss	-0.332
,	-0.085
"	0.202
3	-0.463
profit	0.360
dhrs	-0.202
-	-0.211
./	-0.260
is	-0.546
s	-0.490
that	-0.285
net	-0.300
lt	1.016
at	-0.465
f_{K+1}	0.009

分类结果

"earnings" assigned?	"earnings" correct?	
	YES	NO
YES	735	24
NO	352	2188

Table 16.9 Classification results for the distribution corresponding to table 16.8 on the test set. Classification accuracy is 88.6%.

在文本分类中应用最大熵模型的讨论

- 优点

- 相对于贝叶斯等分类方法，最大熵模型考虑了特征之间的依存关系
- 数据的平滑
- 可以提供特征选择与文本分类一体化框架

- 不足

- 上述结果在分类方法中不是最优的（决策树方法达到96%），因为二值化特征只是记录的特征的出现与否，而文本分类需要知道特征的强度
- 计算代价较大由于算法收敛速度较慢

K近邻 (KNN)

- 最近邻分类规则
 - 对于测试样本点 x ，在集合中距离它最近的的 x_1 。
最近邻分类就是把 x 分为 x_1 所属的类别
- 最近邻规则的推广- KNN
- 没有好的相似度矩阵不能用 KNN

KNN算法

- 目标：基于训练集N的对y分类
- 确定在N中与y最相似的元素x

$$sim_{MAX}(y) = MAX_{x \in N} sim(x, y)$$

- 得到k个最相似的集合

$$A = \{x \in N \mid sim(x, y) = sim_{\max}(y)\}$$

- 设n1,n2分别为集合中属于c1,c2的个数

$$p(c_1 | y) = \frac{n1}{n1 + n2} \quad p(c_2 | y) = \frac{n2}{n1 + n2}$$

- 如果 $p(c1|y) > p(c2|y)$,判为c1,否则判为c2

KNN在文本分类中的应用

"earnings" assigned?	"earnings" correct?	
	YES	NO
YES	1022	91
NO	65	2121

Table 16.12 Classification results for an INN categorizer for the "earnings" category. Classification accuracy is 95.3%.

特点

- 分类性能依赖于相似度矩阵
- 效率问题

神经网络

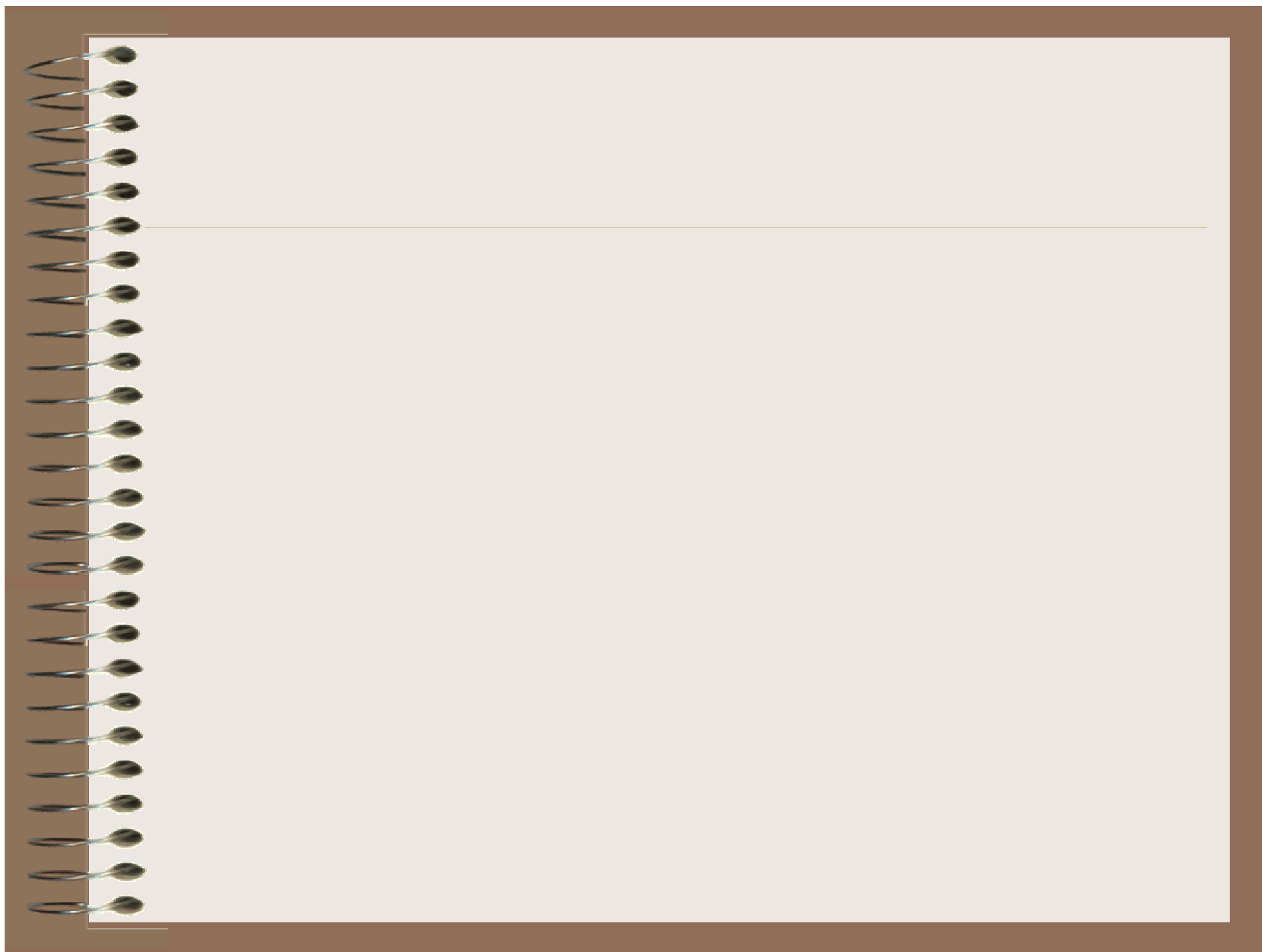
- 好处：
 - 预测准确率通常很高
 - 强健，当样本数据中包含错误和噪声
 - 输出可能是离散值，实数值或者离散值向量，实数值向量
 - fast evaluation of the learned target function
- 不足
 - 训练时间长
 - 很难理解学习函数（权）
 - 不易和领域知识融合

其他方法

- 支持向量集 (SVM)
- 遗传算法
- 粗糙集
- ○ ○ ○ ○ ○ ○

A spiral-bound notebook with a light beige, textured cover and a dark brown border. The spiral binding is on the left side. The word "Thanks!" is written in the center in a black, italicized serif font.

Thanks!



IIS算法

- IIS算法如下:

输入: 约束集, $C \equiv \{p(f_i) = \tilde{p}(f_i), i \in \{1, 2, \dots, n\}\}$

x, y 的经验概率分布 $\tilde{p}(x, y)$

输出: λ^*

- 1、 初始令 $\lambda_i = 0 \quad i \in \{1, 2, \dots, n\}$
- 2、 for $i=1$ to n 循环
- a) 令 $\Delta\lambda_i$ 为下面方程的解

$$\sum \tilde{p}(x)p(y|x)f_i(x, y)\exp(\Delta\lambda_i f^\#(x, y)) = \tilde{p}(f_i)$$

其中, $f^\#(x, y) \equiv \sum_{i=1}^{x/y} f_i(x, y)$ 由对 f 的定义可知在本文中为某一实例 (x, y) 包含的特征数量。

- b) $\lambda_i \leftarrow \lambda_i + \Delta\lambda_i$
- c) 重复 a)至 λ_i 收敛
- 3、 $\lambda^* = \langle \lambda_1, \dots, \lambda_n \rangle$ 算法结束

牛顿迭代法求解

$$g(a_n) = \sum_{x,y} \tilde{p}(x) p(y | x) f_i(x, y) \exp(a_n f^\#(x, y)) - \tilde{p}(f_i)$$

$$g'(a_n) = \sum_{x,y} \tilde{p}(x) p(y | x) f^\# \exp(a_n f^\#(x, y))$$

迭代算法

1 初始令 $i=0, a_i=0$

2
$$a_{i+1} = a_i - \frac{g(a_i)}{g'(a_i)}$$

3 当 $|g(a_{i+1}) - g(a_i)| >$ 某一较小常数, $i++$, 循环至2,

4 算法结束, a_i 为方程解, $a_i = \Delta\lambda_{\odot}$